THE UNIVERSITY *of* EDINBURGH

Resource Efficient Action Recognition in Videos

Shreyank Narayana Gowda

Doctor of Philosophy

THE UNIVERSITY OF EDINBURGH

2023

To my dearly departed dogs,

Lassie and Duncy

Abstract

This thesis traces an innovative journey in the domain of real-world action recognition, in particular focusing on memory and data efficient systems. It begins by introducing a novel approach for smart frame selection, which significantly reduces computational costs in video classification. It further optimizes the action recognition process by addressing the challenges of training time and memory consumption in video transformers, laying a strong foundation for memory efficient action recognition. The thesis then delves into zero-shot learning, focusing on the flaws of the currently existing protocol and establishing a new split for true zero-shot action recognition, ensuring zero overlap between unseen test classes and training or pre-training classes. Building on this, a unique cluster-based representation, optimized using reinforcement learning, is proposed for zero-shot action recognition. Crucially, we show that a joint visual-semantic representation learning is essential for improved performance. We also experiment with feature generation approaches for zero-shot action recognition by introducing a synthetic sample selection methodology extending the utility of zero-shot learning to both images and videos and selecting high-quality samples for synthetic data augmentation. This form of data valuation is then incorporated for our novel video data augmentation approach where we generate video composites using foreground and background mixing of videos. The data valuation helps us choose good composites at a reduced overall cost. Finally, we propose the creation of a meaningful semantic space for action labels. We create a textual description dataset for each action class and propose a novel feature generating approach to maximise the benefits of this semantic space. The research contributes significantly to the field, potentially paving the way for more efficient, resource-friendly, and robust video processing and understanding techniques.

Acknowledgements

I would like to express my deepest gratitude to all those who have supported me throughout this incredible journey of completing my Ph.D. I could not have accomplished it without the help and encouragement of numerous individuals. First and foremost, to my incredible girlfriend, words cannot express the depth of my appreciation. Through the toughest of times, you stood by my side, offering unwavering support and understanding. Your emotional support and the countless hours spent helping me draw diagrams for my papers made all the difference. Every conference deadline of mine became a shared journey, and your dedication to my success was nothing short of remarkable. To my dear brother, my role model. Your belief in me and your trust in my ability is much more than my own. You've been my rock, and I can't thank you enough. I must acknowledge my mother, who demanded that I pursue a Ph.D. Thank you for always pushing me to reach my full potential, even when I resisted. To my dad, you define humility and patience, things that helped me reach where I am. I cannot forget to mention my furry companions, my dogs, whose unwavering positivity and unconditional love provided much-needed respite from the rigors of research. To my master's supervisor, Dr Chun Yuan who made researching a relaxing rather than stressful experience, I can't thank you enough for introducing me to the world of academia and making me enjoy doing research. To my second supervisor, Dr. Frank Keller, thank you for all the support and paper editing you have had to do! I also want to express my gratitude to my collaborator, Dr. Marcus Rohrbach. Your consistent support and expertise in the field have been instrumental in shaping my research. The fruitful discussions and exchange of ideas have enriched my work and broadened my perspectives. A large chunk of my PhD was through the COVID period and I never got to attend conferences in person until the penultimate year of my PhD and for the experiences I finally had, I can't thank Dr. Davide Moltisanti enough. Like a big brother taking me around and introducing me to everyone, thank you. To all my lab mates and friends at the university thank you for making this whole experience so memorable! Lastly, but most importantly, to my amazing supervisor, Dr. Laura Sevilla-Lara, I owe a debt of gratitude. Being her first graduating student, specializing in zero-shot learning, I can't help but reflect on the irony of the situation. In a way, I was your own personal "zero-shot learning" experience as you navigated the uncharted territory of guiding someone through this complex process. Thank you for being my

trailblazing supervisor and for all the invaluable lessons along the way. And now, as I close this chapter and embark on new adventures, I look forward to continuing to make a positive impact on the world with the knowledge and skills I have gained during my Ph.D. journey.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Shreyank Narayana Gowda

Contents

Abstract iii

Acknowledgements iv

Declaration vi

Figures and Tables xiii

1 Introduction 1

1.1 Thesis Statement 6
1.2 Summary of Contributions 6
1.3 Thesis Structure 7

2 Related Work 9

2.1 Memory Efficient Action Recognition 9
2.1.1 Early Approaches to Action Recognition 10
2.1.2 Frame Selection 10
2.1.3 Attention and Relational Models 12
2.1.4 Transformers for Videos 13
2.1.5 Efficient Transformers in Videos 14
2.2 Data-Efficient Learning 15
2.2.1 Zero-Shot Learning in Images 16
2.2.2 Zero-Shot Learning in Videos 17
2.2.3 Semantic Embeddings 17
2.2.4 Evaluation Protocol Strategies for Zero-Shot Learning 18
2.2.5 Deep Approaches to Centroid Learning for Classification . . . 19
2.2.6 Reinforcement Learning for Data Valuation 20
2.2.7 Reinforcement Learning for Zero-Shot Learning 20
2.2.8 Data Augmentation for Video Action Recognition 21
2.2.9 Learning to Augment Data 21
2.2.10 Semi-supervised Video Action Recognition 22

3 Efficient Action Recognition by Key Frames Selection 23

3.1 Introduction . . . 23
3.2 SMART Frame Selection . . . 25
3.2.1 Feature Representation . . . 25
3.2.2 Single-frame Selector . . . 26
3.2.3 Global Selector . . . 27
3.3 Experimental Analysis . . . 29
3.3.1 Experimental Setup . . . 30
3.3.2 Ablation Study on the SMART Frame Selection . . . 32
3.3.3 Analysis of the Behavior of SMART Frame Selection . . . 33
3.4 Quantitative Results Analysis . . . 34
3.4.1 Generality of SMART on Additional Datasets . . . 34
3.4.2 Performance on Untrimmed Datasets . . . 35
3.4.3 Extension of SMART as a Pre-processing Step . . . 36
3.4.4 Improving Performances of Other Models . . . 36
3.5 Limitations . . . 37
3.6 Conclusion . . . 38

4 Time and Memory Efficiency of Transformer Architectures by Appropriate Initialization 41
4.1 Introduction . . . 41
4.2 Methodology . . . 44
4.2.1 Revisiting ViViT . . . 44
4.2.2 Our training strategy . . . 46
4.3 Experimental Analysis . . . 49
4.3.1 Datasets . . . 49
4.3.2 Implementation Details . . . 50
4.3.3 Ablation Study . . . 52
4.3.4 How important is the pre-training image dataset for action recognition performance? . . . 53
4.3.5 How many frames should we use for Stage 1? . . . 54
4.3.6 Does the proposed training increase convergence speed? . . . 55
4.3.7 Extending the image backbone to 1.5B parameters . . . 56
4.3.8 Comparison of Memory Usage with Standard ViViT Training and Proposed Method . . . 56
4.3.9 Comparison on all benchmarks to the baseline model . . . 57
4.4 Curriculum Training . . . 59

4.5 How long do we need to train the model? . . . 59
4.6 What about initializing standard ViViT models? . . . 61
4.7 Limitations . . . 61
4.8 Conclusion . . . 62

5 A New Split for Evaluating True Zero-Shot Action Recognition 63
5.1 Introduction . . . 63
5.2 ZSL preliminaries . . . 66
5.2.1 Visual and Semantic Embeddings . . . 67
5.3 Evaluated Methods . . . 68
5.4 Evaluation protocol . . . 69
5.4.1 Datasets . . . 69
5.4.2 TruZe Split . . . 69
5.5 Experimental Results . . . 72
5.5.1 Results on ZSL and Generalized ZSL . . . 72
5.5.2 Extension to Few-shot Learning . . . 73
5.5.3 Is overlap the reason for performance difference between Random and our TruZe split? . . . 74
5.5.4 Use of different backbone networks . . . 75
5.6 Implementation Details . . . 76
5.6.1 Visual features . . . 76
5.6.2 Semantic embedding . . . 77
5.6.3 Hyperparameters for evaluated methods . . . 77
5.7 Limitations . . . 77
5.8 Discussion and Conclusion . . . 78

6 Regularizing Representations through Clustering for Zero-Shot Action Recognition 79
6.1 Introduction . . . 79
6.2 Method Overview . . . 81
6.2.1 Problem Definition . . . 81
6.2.2 Visual-Semantic Representation . . . 82
6.2.3 CLASTER Representation . . . 83
6.2.4 Loss Function . . . 84
6.2.5 Optimization with Reinforcement Learning . . . 85
6.2.6 Use of Sentence2vec . . . 86
6.3 Implementation Details . . . 87

6.3.1 Visual features . . . 87
6.3.2 Network architecture . . . 87
6.3.3 Number of clusters . . . 87
6.3.4 RL optimization . . . 87
6.3.5 Semantic embeddings . . . 88
6.3.6 Rectification of the Semantic Embedding . . . 88
6.3.7 Nearest Neighbor Search . . . 89
6.4 Experimental Analysis . . . 89
6.4.1 Datasets . . . 89
6.4.2 Ablation Study . . . 89
6.4.3 Results on ZSL . . . 91
6.4.4 Results on GZSL . . . 92
6.4.5 Results on TruZe . . . 93
6.4.6 Analysis of the RL optimization . . . 94
6.5 Regularization Effect of Clustering . . . 95
6.6 Statistical Significance . . . 96
6.7 Average of Differences in Performance for Same Splits . . . 98
6.8 Number of Clusters . . . 98
6.9 Comparison of aggregation strategies . . . 99
6.10 Seen and Unseen Class Performance for GZSL . . . 99
6.10.1 Results on Images . . . 99
6.11 Limitations . . . 101
6.12 Conclusion . . . 101

7 Transferring Semantic Knowledge through Action Stories for Zero-Shot Action Recognition 102
7.1 Introduction . . . 102
7.2 The Story Dataset . . . 105
7.2.1 Building the Stories Dataset . . . 105
7.2.2 What do Stories Look Like? . . . 106
7.2.3 Learning from Stories . . . 107
7.3 SDR . . . 107
7.3.1 Zero-Shot and Generalized Zero-Shot Settings . . . 108
7.3.2 Zero-Shot Action Recognition with Feature Generation . . . 109
7.3.3 Data-driven Noise . . . 110
7.3.4 Ranking Loss for Feature Generation . . . 111

7.4 Experimental Analysis . . . 112
7.4.1 Datasets and Evaluation Protocol . . . 112
7.4.2 Implementation details . . . 113
7.4.3 Ablation Study . . . 114
7.4.4 Results without cleaning of data . . . 114
7.4.5 Comparison to State-of-the-Art on Zero-Shot . . . 115
7.4.6 Comparison to State-of-the-Art on Generalized Zero-Shot . . . 116
7.4.7 Exploring the Few-Shot Setting . . . 118
7.5 Studying the Effect of Stories as a General Semantic Embedding . . . 119
7.6 Why Not Just Use VAE for Feature Generation? . . . 119
7.7 Hyperparameter Selection . . . 121
7.8 Why Does Using a Single Model Work? . . . 121
7.9 Limitations . . . 122
7.10 Conclusion . . . 122

8 Selecting Informative Synthetic Features using Reinforcement Learning for Improved Generalized Zero-Shot Learning 123
8.1 Introduction . . . 123
8.2 Methodology . . . 125
8.2.1 Feature Generating Network (FGN) . . . 126
8.2.2 Selector . . . 126
8.2.3 Proximal Policy Optimization . . . 128
8.3 Experimental Analysis . . . 129
8.3.1 Implementation Details . . . 129
8.3.2 Ablation Study . . . 130
8.3.3 Images . . . 131
8.3.4 Videos . . . 132
8.3.5 Results on TruZe . . . 134
8.4 Limitations . . . 134
8.5 Conclusion . . . 135

9 Learning to Augment Video Datasets by Compositing Quality Videos for Action Recognition 136
9.1 Introduction . . . 136
9.2 Method . . . 138
9.2.1 Selector . . . 139
9.2.2 Semantic Matching (SM) . . . 140

9.2.3 Video Compositing(VC) . . . 141
9.3 Optimization of Learn2Augment . . . 142
9.3.1 Training the Selector . . . 142
9.3.2 Training the Classifier . . . 144
9.3.3 Experimental Details . . . 145
9.3.4 Architectural Changes for Different Settings . . . 146
9.3.5 Ablation Study . . . 146
9.3.6 Augmenting in the Semi-supervised Setting . . . 148
9.3.7 Augmenting in the Few-shot Setting . . . 149
9.3.8 Augmenting the Full Training Set . . . 149
9.4 Why Not Intra-class Augmentation? . . . 150
9.5 Distillation Loss for Semi-Supervised Learning . . . 151
9.6 Analysis of Number of Augmented Samples . . . 151
9.7 Other Selector Choices . . . 152
9.8 Why Re-train the Classifier Network? . . . 153
9.9 Examples of Selected and Discarded Samples . . . 154
9.10 Effect of Semantic Match in generalization ability . . . 155
9.11 Limitations and Future Work . . . 156
9.12 Conclusion . . . 156

10 Conclusion and Future Work **157**
10.1 Short-Term Improvements and Future Work . . . 158
10.2 Long Term Directions . . . 159
10.3 Ethical Impact . . . 160

Bibliography **163**

Figures and Tables

Figures

1.1 Our broad classification of the types of action recognition based on the amount of labelled data available. 5

3.1 Examining the impact of good frame selection using an oracle as shown by Huang et al. [77]. 24

3.2 Overview of the SMART frame selection. The model consists of two streams. The first considers the information of the frames one at a time and the second stream considers the entire video at a time. Here, $\mathscr{L}_{oracle}$ is the MSE loss used to distill frame scores from an expensive model and $\mathscr{L}_{cls}$ is the categorical cross entropy loss used for classification. 26

3.3 Steps involved in calculating the attention weights and intermediaries involved. 29

3.4 Ablation study to determine the effect of each of the components of the SMART frame selection network. We use 26 frames for all experiments. Something-something-v2 dataset. We see that using visual and language features performs better than visual only. We also see that the single frame selector alone helps boost performance at minimal costs whereas using the entire architecture with 2 frames input gives us the best results. 32

3.5 (a) Behavior of different sampling strategies with respect to number of frames. Orange represents SMART, blue represents uniform selection and red represents random selection (b) Comparison of the importance score of semantically similar actions. We can see a striking resemblance for all actions involving pushing. 39

3.6 Graphical comparison of how the two selection modules give importance scores. We can see that each selector is giving different importance weights to different parts of the video. The classes are (a) Spinning something and it continues spinning and (b) Throwing something in the air and catching it 40

3.7 Examples of frames not selected (top) and selected (bottom) for the class “pulling something so that it gets stretched". Frames from [67]. 40

4.1 Comparison of our initialization method vs conventional training of ViViT. Training time is scaled relative to setting ViViT-B training time to '1 unit' (7.93 hours). We see clear time saving using our initialization scheme and for larger models, the training time saved is much larger. 43

4.2 A visual comparison of (a) the factorized encoder version of the ViViT model as our baseline, and (b) our proposed model, which involves freezing the spatial transformer post-initialization, adding a small adapter model for further fine-tuning, and initializing the temporal transformer rather than training it from scratch. 45

4.3 STAGE 1: We first use the full ViViT-FE model on 8 frames by initializing the spatial transformer from an image checkpoint and the temporal transformer from scratch. STAGE 2: We then use this as our checkpoint to initialize the spatial and temporal transformer for models using more frames (such as 32, 64 or 128). We then freeze the spatial transformer and add an adapter model to finetune spatial transformer features. The temporal transformer is finetuned from the same checkpoint. 48

4.4 The effect of initializing with different numbers of frames (JFT, 2, 4, 8, 16, 32, and 48), freezing the spatial transformer and adding an adapter model and fine-tuning using 64, 96, and 128 frames. Results on Kinetics400 dataset, 'f' refers to frames. 54

4.5 Comparison of our initialization method vs conventional training of ViViT on Top-1 accuracy and loss on the Kinetics-400 dataset using 64 and 128 frames. We see that our initialization gives a significant headstart to the models. 55

4.6 Stacked bar chart representing the cumulative processing times of Models A-E. Each color within a bar corresponds to a specific sub-model ('a' in yellow, 'b' in blue, 'c' in green, 'd' in gray) contributing to the total computation time of each model. Model accuracies are indicated at the top of each respective bar. 'a' = ViViT-L-8f, 'b' = SFA-ViViT-L-32f, 'c' = SFA-ViViT-L-128f, 'd' = ViViT-L-128f. All results are using Kinetics-400 dataset and using ViViT-L variants. 60

5.1 An illustration of the overlap in *classes* of the pretraining dataset (grey), training split (yellow) and zero-shot test split (green). Current evaluation protocol (a) picks classes at random with 51 training classes and 50 test classes. There is always some overlap and also chances of an extremely high overlap. We propose a stricter evaluation protocol (b) where there is no overlap at test time, maintaining the ZSL premise. 65

5.2 A few examples of how the classes are selected. (a) is an example of an exact match between the testing dataset (in this case UCF101) and the pre-trained dataset (Kinetics). (b), (d) and (g) are examples of visual-semantic similar matches where the output and semantically closest classes are the same. (c), (e), (f) and (h) are examples of classes without overlap in terms of both visual and semantic similarity. 71

5.3 Graphical representation of the difference in performances of different models on GZSL. We see consistent positive difference in performance on the unseen classes and negative difference in the performance of the seen classes while using the 'TruZe'. The x-axis corresponds to difference in accuracy (Random splits accuracy - TruZe split accuracy) and the y-axis to different methods. 74

5.4 The difference of accuracy for different models using IDT and I3D using manual annotations as the semantic embedding. The larger the bar, the more significant the difference. We can see a clear difference when using I3D and this difference is due to the presence of overlapping classes in the test set. The y-axis corresponds to the difference in performance in percentage and the x-axis corresponds to various models. 75

6.1 Left: Learning curve for the seen classes. Right: Learning curve for the unseen classes. The clustering-based representation avoids overfitting, which in the case of seen classes means that the gap between validation and training accuracy is smaller than in the vanilla representation. This regularization effect improves the validation accuracy in unseen classes. 80

6.2 Overview of CLASTER. We map the semantic embedding $a(y_i)$ to the space of visual features x_i, and concatenate both to obtain a visual-semantic representation. We cluster these visual-semantic representations with K-means to obtain initial cluster centroids. Each video is represented as the sum of the visual-semantic representation and the centroid clusters, weighted by their distance (see Sec. 6.2.3 and Fig. 6.3). This is used as input for classification (Sec 6.2.4 and Eq. 6.3). Based on the classification result, we send a reward to optimize the cluster centroids using REINFORCE (Sec. 6.2.5). At test time, we first perform classification on the seen classes and then do a nearest neighbor (NN) search to predict the unseen class. 82

6.3 The proposed CLASTER Representation in detail (see Fig. 6.2 for the overview of the full method). The visual feature is mapped to match the space of the visual-semantic cluster centroids with an MLP and concatenation. Based on the distances to the cluster centroids the final representation ω is a weighted representation of the centroids, more robust to the out-of-distribution instances of the unseen test classes. The comparison block refers to comparing of the visual-semantic centroids and a data point to find the closest centroid. Details in Sec. 6.2.3 and Eq. 6.1. 84

6.4 **HMDB51** (a) Averaging word embeddings can produce poor results in certain cases. For example the nearest neighbor of “shoot ball" is “shoot gun", and of “sit up" is “sit down" which are not necessarily meaningful (b) Sentence2vec better captures phrase meanings: Nearest neighbors to “sit up" is now “push up", and for “shoot ball", is golf. **UCF101** (c) The same effect is observed, where after averaging word2vec representations, the nearest neighbor of “pommel horse" is "horse riding" (d) Sentence2vec helps capture phrase meanings: the while nearest neighbor of “pommel horse" is now “balance beam". The circles contain the nearest neighbor to the given unseen class and is for illustration purposes. 86

6.5 CLASTER improves the representation and clustering in unseen classes. The figure shows t-SNE [115] of video instances, where each color corresponds to a unique unseen class label. The RL optimization improves the representation by making it more compact: in (b) instances of the same class, i.e. same color, are together and there are less outliers for each class compared to (a). 94

6.6 Analysis of how RL optimization changes the cluster to which an instance belongs. The frequencies are represented as percentages of instances in each cluster. We can see that the clusters are a lot "cleaner" after the optimization by RL. 94
6.7 Left: Learning curve for the seen classes. Right: Accuracy curve for the unseen classes. The clustering-based representation avoids overfitting, which in the case of seen classes means that the gap between validation and training accuracy is smaller than in the vanilla representation. This regularization effect improves the accuracy in unseen classes. 95
6.8 Left: Learning curve for the seen classes using 35% of the data. Right: Learning curve for the unseen classes. The clustering-based representation avoids overfitting, which in the case of seen classes means that the gap between validation and training accuracy is smaller than in the vanilla representation. This regularization effect improves the accuracy in unseen classes. 96
6.9 Effect of using different number of clusters. The green line represents the standard deviation. The reported accuracy is on the UCF101 dataset. As can be seen, the average cluster accuracy increases till about 6 clusters and then remains more or less constant. The vertical lines correspond to the standard deviation. 98

7.1 Comparison of accuracy across state-of-the-art zero-shot approaches (CLASTER [64], E2E [15], OD [116], and the proposed SDR using different semantic embeddings (the proposed Stories, word2vec [120] (W2V) and elaborative definitions [23] (ER), on the UCF101 [157] dataset. Using the proposed Stories to create the semantic space of class labels improves the performance by a large margin, showing that it is model-agnostic. 103
7.2 Comparing nearest neighbors using Stories. We see an example where ER fails and Stories provides more context and helps in obtaining better neighbors. This is one example of where ER fails, there are multiple such examples. Dataset is UCF101. 105
7.3 Visualization of the features generated from the *Stories* embedding vs ER, using t-SNE [115]. 107

7.4 Overview of SDR. SDR generates video features that resemble real ones. It uses I3D to compute real visual features, x_i, and a "Projection Network" to generate synthetic semantic embeddings, $\hat{a}i$. The "Feature Generator" combines real semantic embeddings, ai, with noise to produce synthetic visual features, $\hat{x}i$. A discriminator is trained to distinguish between pairs $\hat{x}i$, $\hat{a}i$ and xi, a_i. The "Projection Network" is trained with a ranking loss, $\mathscr{L}_{rank}$, to separate synthetic semantic representations. The generator and discriminator engage in a joint training game. 108

7.5 Training the generator using data-driven noise converges much faster than using the standard Gaussian noise. Also, saving the feature generator at various stages and testing on HMDB, we see that with using DBN we obtain our best performance at epoch 30 and the performance in later epochs are also consistent. However, without DBN we see the best performance at 150 epochs and the accuracies are a bit erratic. 111

7.6 The generator loss using data-driven noise is much more stable, leading to faster convergence and better accuracy. We see that using DBN stabilizes the training of our feature generator whilst also ensuring we reach our optima quicker. 112

7.7 Comparison of accuracy across state-of-the-art zero-shot approaches (CLASTER [64], E2E [15], OD [116], and the proposed SDR using different semantic embeddings. Using the proposed *Stories* to create the semantic space of class labels improves the performance by a large margin, showing that it is model-agnostic. 115

7.8 Comparison of using different number of nearest neighbors on both (left) the data-based noise and (right) the ranking loss. 121

8.1 Comparison of feature-generating frameworks pipelines (a) standard pipeline where features that look “real" are used to train the generator (b) proposed pipeline where the generator is trained based on the performance of the seen class classifier. 124

8.2 Overall pipeline of our proposed SPOT. The feature generator generates features that the selector module ranks based on the seen class classifier’s performance. The selector is updated based on the performance of the classifier on the selected features. The proposed pipeline is model and data-agnostic. 125

8.3 Ablation comparison of different combinations of RL algorithms with selector choices (GRU or Transformer). 'G' = GRU, 'Tr' = Transformer, 'R' = REINFORCE, 'T' = TRPO and 'P' = PPO. 130

9.1 Standard video augmentation techniques generate data using hand-designed heuristics (left). We propose to learn to select videos for augmentation, based on how effective they will be for learning to classify (middle). Our approach, Learn2Augment, improves classification across datasets and settings, including UCF101 (right). 137

9.2 Overview of the proposed Learn2Augment. Given a pair of videos and their labels, a Selector network gives a score ω of the quality of the potential composited video. At training time, the Selector is trained with the validation loss of the classification network. Once the Selector is trained, pairs of videos are sampled, and only the promising combinations with high score ω are composited and used for training the classifier. 139

9.3 Pipeline for compositing a single frame. The foreground is from the class "soccer juggling" and the background from the class "soccer penalty", which are semantic class neighbors. We can see objects such as 'person' and 'ball' are detected as objects of interest. 140

9.4 Sample frames of rendered videos. While the segmentation contains errors, such as missing limbs or portions of the object, the action category remains clear. 141

9.5 Comparison of performance with increasing number of augmented samples. Results are for 10% and 20% of labeled data UCF101. We see that the performance increases initially, reaches a peak and slowly starts dropping. 152

9.6 Visualizing selected examples. From top to bottom as (foreground, background) pairs: (flic-flac, cartwheel), (smile, laugh), (playing violin, playing cello), (front crawl, swimming backstroke). The first two are examples from HMDB51 and the last two from UCF101. 154

9.7 Visualizing discarded examples. From top to bottom as (foreground, background) pairs: (somersault, diving), (climbing stairs, falling floor), (baby crawling, walking dog), (hammering, hammer throw). 155

Tables

3.1 Baseline frame sampling techniques vs our SMART with ResNet-152 backbone; #F: #frames. Results on UCF101 dataset 35
3.2 Baseline frame sampling techniques vs our SMART with ResNet-152 backbone; #F: #frames. Results on Kinetics dataset subsets: Temporal and Static 35
3.3 Results on ActivityNet and FCVID of the SMART frame selection. Compared to recent state-of-the-art methods, the proposed method outperforms their accuracy. #F: Number of frames, 10c corresponds to 10 clips used instead of frames 36
3.4 Extending SMART as a pre-processing step to state-of-the-art deep learning approaches. The '+ Kinetics' indicate that the backbone is pre-trained with Kinetics. 37
3.5 Extending SMART to other approaches 37

4.1 The hyperparameters utilized in the experiments conducted for the primary research paper are detailed here. If a regularisation method is not employed, it is represented by a "–". Constant values that are present across all columns are mentioned just once. For simplicity, abbreviations have been used to denote different datasets: Kinetics 400 is represented as K400, Kinetics 600 as K600, Moments in Time as MiT, Epic Kitchens as EK, Something-Something v2 as SSv2 and Something-Else as Selse. . . 51
4.2 Ablation study results illustrating the impact of various modifications to the ViViT-B model, including spatial and temporal transformer freezing, adapter addition, and initialization methods, on top-1 and top-5 accuracy. Dataset is Something-something v2. The Index is to refer to a particular row in Section 4.9.3.
4.3 Comparison of impact of different backbones for the spatial transformer. . 52
We use ResNet-ViT-L pre-trained on ImageNet21k, ViT-L pre-trained on ImageNet21k, ViT-L pre-trained on JFT and ViT-H pre-trained on JFT. Listed as (Top-1 accuracy/ Top-5 accuracy). 53
4.4 Memory usage in different ViViT training schemes is compared using the Kinetics400 dataset on 64 TPUs v3 with 16GB memory each. A ✓ indicates accessible frames given hardware constraints, while a $\times$ signals an out-of-memory (OOM) error. Here, 'f' in 8f, 16f and so on corresponds to the number of frames. 57

4.5 Performance Comparison of various versions of ViViT with the proposed training strategy for Kinetics-400, Kinetics-600 and Moments in Time. Accuracies listed as Top-1/Top-5. 58
4.6 Performance Comparison of ViViT-L with the proposed training strategy for Something-something v2, Something-Else and Epic-Kitchens. Accuracies listed as Top-1/Top-5, for Epic Kitchens Top-1 noun-verb/ Top-1 noun/ Top-1 Verb. 58
4.7 A summary of cross-dataset initialization of the proposed model and performance comparison. We use Kinetics400 and Something-something v2 as our datasets. 58
4.8 A comparison of top-1 and top-5 accuracies for the ViViT-g model with the proposed training strategy, which incorporates a larger spatial transformer backbone. All models use 48 frames for fair comparison. Results are on Kinetics400 dataset. 58
4.9 Comparison of near peak performance (NPP) epoch and best performance epoch for ViViT and SFA-ViViT for different datasets and models. All results are on Kinetics400 dataset. 60
4.10 Comparison of near peak performance (NPP) epoch and best performance epoch for initializing the full ViViT model with and without the 8f variant. We see the benefit of initialization as the “near-peak" performance is reached at a much earlier stage when initialized with the 8f variant. All results are on Kinetics400 dataset. 61

5.1 Datasets and their splits used for ZSL in action recognition. Traditionally, ‘Random Split’ was followed where the seen and unseen classes were randomly selected. However, we can see the extent of overlap in the ‘overlapping classes’ column. Using the extent of overlap we define our ‘TruZe split’. *Note that for the all experiments in this paper we use a random split which matches the number of classes of our TruZe, e.g. 29/22 for HMDB51. 70

5.2 Results with different splits for **Zero-Shot Learning (ZSL)**. Column 'Random' corresponds to the accuracy using splits in the traditional fashion (random selection of train and test classes, but with the same number of classes in train/test as in TruZe), 'TruZe' corresponds to the accuracy using our proposed split and 'Diff' corresponds to the difference in accuracy between using random splits and our proposed split. We run 10 independent runs for different random splits and report the average accuracy. We see positive differences in the 'Diff' column which we believe is due to the overlapping classes in Kinetics. 72

5.3 Results with different splits for **Generalized Zero-Shot Learning (GZSL)**. 'Rand' corresponds to the splits using random classes over 10 independent runs, 'TruZe' corresponds to the proposed split. Acc_U and Acc_S correspond to unseen class accuracy and seen class accuracy respectively. The semantic embedding used is sen2vec. 'diff' corresponds to the difference between 'Rand' and 'TruZe'. We see consistent positive difference in performance on the unseen classes and negative difference in the performance of the seen classes while using the 'TruZe'. 73

5.4 **Few Shot Learning (FSL)** with different splits on UCF101. Accuracies are reported for 5-way, 1, 2, 3, 4, 5-shot classification. 'SS' corresponds to the split used in [209, 133] and 'TruZe' corresponds to the proposed split. We can see that using our proposed split results in a drop in performance of up to 6.2 % for UCF101. This shows TruZe is much harder even in the FSL scenario. 73

5.5 **Few Shot Learning (FSL)** with different splits on HMDB51. Accuracies are reported for 5-way, 1, 2, 3, 4, 5-shot classification. 'SS' corresponds to the split used in [209, 133] and 'TruZe' corresponds to the proposed split. We can see that using our proposed split results in a drop in performance of up to 17.1 % for HMDB51. This shows TruZe is much harder even in the FSL scenario. 74

5.6 Results comparison using different backbones to extract visual features for the ZSL models. We evaluate OD, E2E and CLASTER using I3D, NL-I3D and SlowFast networks as backbones. All results are on the proposed split. We see that stronger backbones result in improved performance of the ZSL model. 76

6.1 Results of the ablation study of different components of CLASTER ZSL. The study shows the effect of clustering, using visual-semantic representations, and optimizing with different methods. All three components show a wide improvement over the various baselines, suggesting that they are indeed complementary to improve the final representation. 91
6.2 Results on ZSL. SE: semantic embedding, M: manual representation, W: word2vec embedding, S: sentence2vec, C: Combination of embeddings. The proposed CLASTER outperforms previous state-of-the-art across tasks and datasets. 92
6.3 Results on GZSL. SE: semantic embedding, M: manual representation, W: word2vec embedding, S: sentence2vec, C: combination of embeddings. The seen and unseen class accuracies are listed in the Section 6.10. 93
6.4 Results on TruZe. For ZSL, we report the mean class accuracy and for GZSL, we report the harmonic mean of seen and unseen class accuracies. All approaches use sen2vec annotations as the form of semantic embedding. 93
6.5 Comparison of the t-test for different pairs of models on the same random split. Lower the value of 'p', higher the significance. As we can see, our results are statistically significant in comparison to both OD [116] and WGAN [185] in both ZSL and GZSL. For GZSL, OD [116] also achieves results that are significant in comparison to WGAN [185]. 97
6.6 Comparing the average of the difference in performance for recent state-of-the-art approaches in zero-shot and generalized zero-shot action recognition on the same splits. All results were computed using sen2vec as the embedding. We can see that we outperform recent approaches in every scenario. 98
6.7 Results on different aggregation options for the semantic and visual embeddings. 99
6.8 Seen and unseen accuracies for CLASTER on different datasets using different embeddings. 'E' corresponds to the type of embedding used, wherein 'A', 'W', 'S' and 'C' refers to manual annotations, word2vec, sen2vec and combination of the embeddings respectively. 'u', 's' and 'H' corresponds to average unseen accuracy, average seen accuracy and the harmonic mean of the two. All the reported results are on the same splits. . . 100

6.9 Results on image datasets. For ZSL, we report the mean class accuracy and for GZSL, we report the harmonic mean of seen and unseen class accuracies. All approaches use manual annotations as the form of semantic embedding. 100

7.1 Ablation Study of Proposed Components. This table shows an ablation study evaluating the impact of each proposed component of SDR: using Stories embeddings, incorporating data-driven noise (DBN), and adding the ranking loss (Lrank). Results are reported for zero-shot (ZSL) and generalized zero-shot learning (GZSL) on HMDB and UCF101 datasets. The table demonstrates that each component provides gains over the baseline, and combining all three gives the best overall performance. . . 114
7.2 Results on zero-shot action recognition on the Olympics, HMDB51 and UCF101 datasets. 'SM' corresponds to the single model experiment. . . . 116
7.3 Results on zero-shot action recognition on Kinetics-220. 116
7.4 Results on TruZe. For zero-shot, we report the mean class accuracy, and for generalized zero-shot, we report the harmonic mean of seen and unseen class accuracies. The split refers to the train/test split used for each setting. 117
7.5 Results on generalized zero-shot setting. Reported results are the harmonic mean of the seen and unseen class accuracies. 'SM' corresponds to the single model experiment. 117
7.6 Seen and unseen accuracies for CLASTER on different datasets using different embeddings. 'SE' corresponds to the type of embedding used, wherein 'M', 'W', 'S', 'C' and 'Sto' refers to manual annotations, word2vec, sen2vec, combination of the embeddings and Stories respectively. 'u', 's' and 'H' corresponds to average unseen accuracy, average seen accuracy and the harmonic mean of the two. All the reported results are on the same splits. SDR+I3D corresponds to the backbone network being I3D and similarly for SDR+CLIP. 118
7.7 Results on the few-shot setting, using HMDB51 and UCF101. 119
7.8 Results on ZSL. SE: semantic embedding, M: manual representation, W: word2vec embedding, S: sentence2vec, Sto: Stories. 120
7.9 Comparing different choices for feature generator. Reported results are on 10 different runs and all models use the same split. Dataset is HMDB51. 120

8.1 Results on zero-shot image classification using recent feature-generating frameworks. . . . 131
8.2 Results on generalized zero-shot image classification on 4 challenging benchmarks. . . . 131
8.3 Results on zero-shot action recognition on the Olympics, HMDB51 and UCF101 datasets. . . . 133
8.4 Results on generalized zero-shot setting. Reported results are the harmonic mean of the seen and unseen class accuracies. . . . 134
8.5 Results on TruZe. We report the mean class accuracy for zero-shot; for generalized zero-shot, we report the harmonic mean of seen and unseen class accuracies. . . . 134

9.1 Ablation study to explore the impact of each proposed component. All settings use the same number of samples for training, so that they can be compared fairly. The # Videos (S) corresponds to the search space in each scenario. As we can see, we obtain the best accuracy using just 12K instead of the standard scenario which would have had 10.4M i.e. a reduction of over 1000x. . . . 147
9.2 Ablation study of compositing components. The version “w/o Inpaint" refers to pasting the foreground without first filling in the holes of removed objects in the background. The version “w/o Segmentation" refers to using bounding boxes instead of object segmentations. “w/o Objects" refers to copying and pasting only the humans in the scene, leaving the objects. . 147
9.3 Results on the semi-supervised setting. Results for TCL and ActorCut are obtained by us running the author’s code. All methods are run with a 3D ResNet-18 backbone for fair comparison. L2A+Pre-training refers to pre-training the selector and fixing it. . . . 148
9.4 Results on UCF101 for the Few-Shot Learning setting, with different splits. Accuracies are reported for 5-way, 1, 2, 3, 4, 5-shot classification. S corresponds to the split used in [209, 133] and T is the TruZe split [65], which avoids overlapping classes with Kinetics. . . . 149
9.5 Augmenting standard datasets improves classification even with a model pre-trained on the largest existing dataset (Sports1M). . . . 150

9.6 Comparison of approaches for the use of Selector. All results are reported on UCF101. 'Acc' corresponds to accuracy and 'SS' corresponds to the number of mixed videos that the Selector looks at. All results are on different percentage of labeled data in UCF101. 153
9.7 Comparison of jointly training classifier and re-training it. We see that there is a consistent large improvement in re-training the classifier. 153
9.8 Results on FSL using the proposed Semantic Matching vs random matching using the TruZe [65] split. 155

Chapter 1

Introduction

Action recognition in videos, the discipline that is dedicated to identifying and classifying specific actions within a sequence of video frames, is a significant topic within the broader field of computer vision. This topic has gained considerable attention due to its wide-ranging real-world applications, which span from surveillance systems and human-computer interaction interfaces to autonomous vehicles and robotic systems [153, 20]. In each of these domains, the capacity to efficiently and accurately recognize actions can drastically enhance system performance, thereby contributing to safer, more interactive, and increasingly automated environments. As such, the development of robust action recognition techniques holds substantial societal and technological implications.

Traditional approaches to action recognition have relied heavily on handcrafted features and shallow learning methods. While these methods have demonstrated some level of effectiveness in certain contexts, they are inherently limited in their adaptability to diverse real-world scenarios, due to their rigid reliance on predetermined features and assumptions [173]. The advent of deep learning has brought about a transformative shift in the landscape of action recognition. Deep learning-based systems have demonstrated their superiority over traditional methods, offering more flexible and adaptable solutions through their capacity for automated feature extraction and end-to-end learning [153]. This has facilitated the recognition of complex and nuanced actions across a range of diverse and challenging video data.

However, this performance has come at a cost. Training these models require a lot of memory. Hence, we first look at memory efficient training strategies to reduce memory costs involved with conventional training strategies. In action recognition models frames are sampled uniformly, without looking at their importance or value to the overall dataset. The typical trend is to then see models accessing more frames perform much better than models with fewer frames. However, not everyone has the

resources to access a lot of frames and one way of mitigating this problem is to first use a lightweight model to access all frames and rank them. Then use the most important frames to train the heavier models. We see with this type of training, that using more frames does not necessarily contribute to increased accuracy and we also see that we can actually improve performance by dropping redundant or unimportant frames. We show this type of training strategy can be used as a plug-in for many 2D and 3D networks to boost their accuracy whilst reducing their cost.

Recently, transformers [36] have revolutionized computer vision by offering an alternative to traditional CNNs, leading to the development of many new state-of-the-art architectures [36, 39]. Moreover, the flexibility of transformers have inspired researchers to adapt these models to more complex problems, including video understanding and action recognition [5, 113]. With the introduction of transformer based models, the memory based problem further aggravated. Transformer based models obtained much better accuracies than previous models but needed much more memory and time to train them. One efficient way of training transformers was proposed in ViViT [5], wherein the spatial and temporal transformers were separated and this form of training was called factorized encoder. While better than other variants of transformer training, this was still inefficient in comparison to older models. A problem with this was that initializing the spatial transformer was easy with the use of image based models but initializing the temporal transformer was not so straightforward.

We propose a training recipe to make training of this breed of transformer architectures more efficient. Our method has two stages. First we pretrain a cheaper version of the model using fewer frames. Then we fine tune with more frames, freezing the spatial encoder and adding an adapter between the frozen spatial representations and temporal transformer. This includes pretraining the temporal transformer, often overlooked in current models.

Along with requiring large memory, the success of deep learning models in action recognition is largely attributable to the existence of extensive labeled datasets. In computer vision, neural networks have existed for decades, but one of the enabling factors for the current revolution was the development of the large ImageNet [31]. In the video domain, manually collecting and annotating data can be a prohibitively expensive process. In video action recognition, for example, collecting data requires an immense amount of manual labor, as it involves finding suitable videos, trimming them and classifying them. Nonetheless, the process of annotating this data, par-

ticularly for videos, is both challenging and resource-intensive. This has led to the emergence of an area of study known as zero-shot learning (ZSL), a unique learning paradigm wherein the model is trained on a collection of seen classes and evaluated on a separate set of unseen classes.

In the context of videos, ZSL involves augmenting each class label with semantic embeddings. These embeddings, which can either be manually annotated or automatically generated by language models, provide rich descriptors for each class. During testing, these embeddings are utilized to match the prediction from the seen classes to the most similar unseen class, thereby allowing the model to infer the likely unseen class for a given video. By devising techniques that can learn from smaller datasets, we can expand the applicability of action recognition systems to a wider range of domains and practical use cases. Therefore, data-efficient methods play a vital role in enabling action recognition in scenarios where data availability is limited.

However, work in video zero-shot learning often uses a pre-trained model to represent videos. While pre-trained models help obtaining good visual representations, overlap with test classes can invalidate the premise of zero-shot learning, making it difficult to compare approaches fairly. We propose a dataset split to overcome this problem. To do this we do an extensive study of visually and semantically similar classes with Kinetics400 [20] and remove all classes from the testing set with overlaps. We also show how the accuracies of models are inflated using the traditional splits.

There are a number of ways to deal with data efficient settings, simple ones include synthetic data generation, data augmentation etc. We could also benefit by incorporating multi-modal training strategies and large unlabelled datasets (using self-supervised learning for example). Traditionally zero-shot learning approaches focused on improving visual representations or semantic representations individually [15, 116, 23]. We propose a representation learning strategy that learns them jointly. Our representation strategy involves joint-clustering of visual-semantic representations which regularizes the learning process and then optimizing these clusters using reinforcement learning (RL). We do this as we do not have any ground truth to tell us how good the clusters are and instead use RL to optimize them based on validation accuracy.

Existing methods assume distances in visual and semantic space are meaningful. Visual similarity arises naturally with presence of similar objects but semantic similarity of labels is harder, though some do overlap. Previous efforts used manual attributes, embedding functions like word2vec, and action definitions to improve se-

mantic space and transfer between seen and unseen classes. We build meaningful action label space using descriptions of steps to achieve each action. Steps contain objects, tools, scenes, verbs etc - the "common sense" associations for the action. E.g. he action penalty shot steps have ball, grass, kick etc. Compared to word2vec embeddings, this simple approach improves semantic space and facilitates transfer between seen and unseen classes, boosting performance. The embedding is general and improves all state-of-the-art methods.

As already mentioned, augmentation is a very common solution to deal with these kinds of problems. One such solution is the use of Generative Adversarial Networks (GAN) [215, 185, 187] to generate synthetic visual features. GANs have aided zero-shot learning but don't explicitly learn interpretable representations. They also suffer from mode collapse, class imbalance, expense for high-dimensional data, and generating samples without studying if they actually improve the classifier. To address the limitations of GANs for synthetic feature selection, we propose a novel reinforcement learning approach that automatically selects generated features to improve model performance. Specifically, we use a transformer model for synthetic sample selection, with validation classification accuracy as the reward for RL training. However, this does not address the cost of actually generating all synthetic features first.

We could also augment the data using the data itself. However, video augmentation is a very complex process. The complexity of coherent motion, backgrounds, and variability in videos poses difficulties for augmentation techniques to scale up effectively while generating useful and realistic samples that preserve training labels. We propose to synthesize realistic and diverse augmented training videos to improve model generalization. The core idea is to learn to composite two action videos in a physically plausible manner while preserving the original action labels. The compositing network takes two randomly sampled action videos as input and predicts blending weights to composite foreground into background in a spatially-varying manner. We also go one step better than before, by learning to choose what samples to composite before actually compositing them.

This thesis presents an in-depth examination of efficiency in the realm of action recognition, with efficiency being interpreted from both a data and memory perspective. For our work, we broadly classify action recognition to: supervised, semi-supervised, few/zero-shot as can be seen in Figure 1.1. On the data front, the work caters to

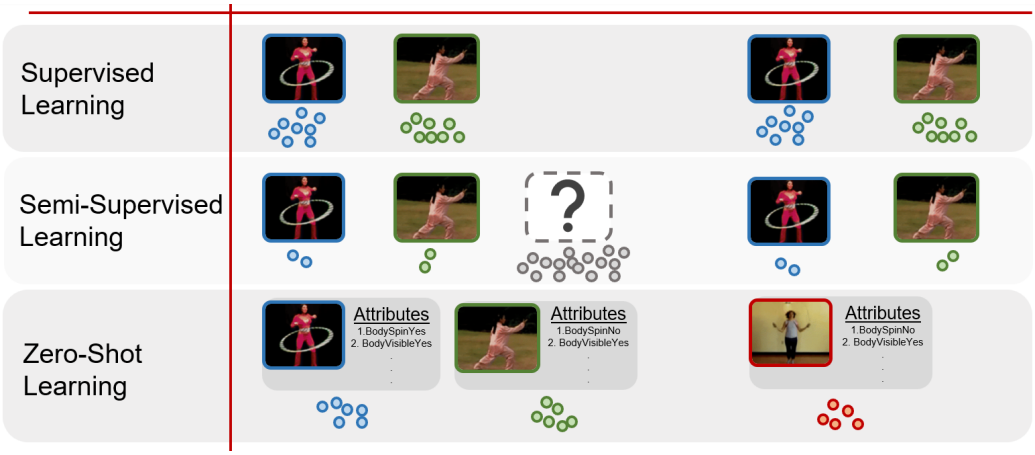


Figure 1.1: Our broad classification of the types of action recognition based on the amount of labelled data available.

scenarios where labeled data is scarce, thereby focusing on paradigms such as zero-shot, few-shot, and semi-supervised action recognition. In terms of memory efficiency, the objective is to refine and optimize deep learning models to deliver high accuracy while also significantly reducing memory usage and computational speed.

To ensure the widespread adoption and progress of data-efficient and memory-efficient action recognition techniques, it is essential to democratize access to these advancements. Deep learning models and training regimes that democratize access empower universities and smaller companies to participate in the development and implementation of action recognition systems.

By providing equal access to knowledge and resources, we foster diversity, creativity, and collaboration among a broader community of researchers and developers. This democratization leads to new perspectives, innovative ideas, and interdisciplinary collaborations, ultimately pushing the boundaries of what is possible in data-efficient and memory-efficient action recognition. With this in mind we formulate a thesis statement.

1.1 Thesis Statement

This thesis explores two key principles that effectively contributed to our goal of efficient action recognition. First, we find that peak performance requires only a subset of data rather than the full dataset. This is consistent across various settings: fully supervised, semi-supervised and zero-shot learning. More importantly, identifying these relevant subsets is something that can be learned. Second, richer features either through joint multi-modal learning or enriched from sophisticated language descriptions is crucial for zero-shot learning. This helps across datasets and models.

1.2 Summary of Contributions

In summary we propose a number of methodologies to tackle both data-efficient and memory-efficient action recognition. Our contributions are listed below:

- SMART Frame Selection [62]: a frame selection strategy that enhances action recognition accuracy and reduces computational costs in trimmed videos by considering frames jointly rather than individually using an attention and relation network to score frames. (AAAI-2021)
- Optimizing ViViT Training [59]: this presents two strategies to reduce the training time and memory requirements of the ViViT Factorised Encoder for action recognition: using a compact adapter model for fine-tuning image representations and initializing the temporal transformer with an 8-frame pre-trained model, achieving reduced training costs, memory usage, and potentially improved performance while enabling use of larger models and more frames. (In Review)
- TruZe [65]: True Zero-Shot (TruZe) Split for action recognition, eliminating class overlap between training and testing datasets, creating a more challenging benchmark that demonstrates lower unseen performance and changes the evaluation standard for zero- and few-shot action recognition. (GCPR 2021)
- CLASTER [64]: a novel cluster-based representation optimized by reinforcement learning, that enhances Zero-Shot action recognition by preventing overfitting to seen classes and improving generalization to unseen ones, outperforming state-of-the-art methods across standard Zero-Shot video datasets. (ECCV 2022)

- Stories [63]: The paper proposes a novel approach for zero-shot action recognition, leveraging language modeling and a WikiHow-based corpus to build a meaningful semantic space for action labels, involving the creation of action "stories" for a deeper understanding, accompanied by a new feature generation and ranking loss method. (In Review)
- Synthetic Sample Selector [58]: presents a novel, model- and data-agnostic approach using reinforcement learning for selecting synthetic features in Generalized Zero-Shot Learning (GZSL), enhancing recognition by reducing redundancy and improving performance across multiple benchmarks. (CVPR-W 2023)
- Learn2Augment [61]: this proposes a learned video augmentation strategy that selects high-quality foreground and background video pairs for compositing, enhancing video action recognition by reducing computational cost and improving accuracy across different training settings, achieving state-of-the-art results in limited data situations. (ECCV 2022)

1.3 Thesis Structure

The structure of the subsequent chapters in this work is each dedicated to a distinct paper. Each of these chapters delves deeply into a number of important aspects associated with the paper it focuses on.

Firstly, the motivation behind the idea presented in the paper is explored. This section elucidates the reasoning that drove the conception of the idea, the problems it aims to solve, and the perceived benefits or advancements it contributes to the field.

Next, the chapter outlines the specific notations associated with the paper. This could include any mathematical notations, technical terms, acronyms, or other symbolic representations that are pivotal to understanding and interpreting the paper. This section aims to ensure the reader is fully equipped to comprehend the technical specifics of the paper's methodologies or findings.

The chapter then moves on to discuss the experimental setup, detailing the research design, methodologies, materials, data collection processes, and any other critical factors that were involved in executing the paper's experiments. It allows the reader to understand how the research was conducted and the foundation upon which the results are based.

Subsequently, the chapter presents the results of the paper's experiments. This section elaborates on the outcomes of the research, the key details in running the experiments etc. It may also discuss the statistical significance of the results, their implications, and how they align with or contradict existing research.

Each chapter concludes with a thoughtful synthesis of the paper discussed, tying together the motivation, related work, notations, experimental setup, and results. This section offers final thoughts on the importance of the paper's contribution to its field, possible areas for further research, and the potential applications of its findings.

Chapter 2

Related Work

This chapter reviews prior work related to efficient approaches for action recognition. With a focus on memory-efficient and data-efficient techniques for video analysis, this chapter provides an overview of relevant literature on optimizing action recognition models and algorithms.

The literature on compressing action recognition networks, distilling knowledge from large models into smaller ones, and leveraging unlabeled or limited labeled data through semi-supervised, few-shot, and zero-shot learning is summarized. Pushing boundaries in efficient video understanding is the aim, so building on top of these works provides important contextualization.

We only briefly introduce the topics in this section. More technical details can be found in the chapters of each individual paper. This chapter serves as an overview to navigate the landscape of efficient action recognition techniques and applications.

We further break this down to two sections and a number of subsections. We divide the related work to memory efficient action recognition and data efficient action recognition.

2.1 Memory Efficient Action Recognition

In this section we discuss briefly various works relevant to our proposed works in the realm of memory-efficient action recognition. We discuss topics such as frame selection, attention and relation networks, transformers in videos and efficient transformers.

2.1.1 Early Approaches to Action Recognition

Before the advent of deep learning, action recognition in videos relied on traditional computer vision and machine learning techniques. These approaches typically involved the extraction of handcrafted features from video frames, followed by the application of classifiers or temporal models.

Commonly used features included Histogram of Oriented Gradients (HOG) [166], Histogram of Optical Flow (HOF) [95], and Spatiotemporal Interest Points (STIP) [96]. These features aimed to capture information about motion, shape, and appearance within video sequences.

Classification methods such as Support Vector Machines (SVM) [146], Hidden Markov Models (HMM) [14], and Bag-of-Words (BoW) [161] were frequently employed to recognize actions based on these features. Other approaches include proposing a generic framework for incorporating latent variables into object and action classification, accounting for localization and alignment, and improving classification through iterative expansion of the latent parameter space [13].

Additionally, in the realm of action recognition, Wang et al. introduced a spatiotemporal context modeling method [172], which effectively captured temporal dependencies and contextual information for improved action recognition in videos.While these pre-deep learning methods showed promise, they often struggled with handling complex and large-scale datasets, as well as achieving robust performance across diverse action categories. With the advent of deep learning, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), action recognition has made significant strides in both accuracy and scalability, revolutionizing the field.

2.1.2 Frame Selection

The field of identifying key frames for action recognition is a relatively nascent one, with substantial exploration and advancements yet to be made. Selecting key frames from video can enable more efficient action recognition compared to using all frames. Strategic frame sampling aims to pick frames that provide maximal information about the action while reducing redundant frames.

Earlier approaches select key frames by either capturing local motion features and their spatiotemporal arrangements [218], pyramidal motion feature extraction with temporal correlogram representation [111], or using a spatially-localizable poselet-like representation with HoG and BoW components learned from weak annotations [138] among other approaches.

Over time, numerous methodologies have been developed that rely on training a reinforcement learning (RL) agent. This approach revolves around the principle of examining individual frames sequentially, with the ultimate objective being to predict the number of frames that can be feasibly bypassed without compromising the integrity of the action recognition process.

AdaFrame[183] leverages RL, in combination with an LSTM that is augmented with memory that helps providing context information for selecting frames to use. Given a frame, it generates a prediction of the action class and it decides which frame to observe next and computes the expected reward of seeing more frames.

FastForward[40] is an end-to-end reinforcement learning approach. It consists of two sub networks: an adaptive stop network and fast forward network. The adaptive stop network can either let the frame sampling continue or stop. The fast forward network has a set of several actions (going backwards or going forward with varying seconds). The RL agent learns to skim through the video.

FrameGlimpse[199] follows the intuition that detecting an action is dependent on observation and refinement. FrameGlimpse relies on a recurrent neural network based agent that observes and decides where to look next. Given the current frame, the agent also decides whether to emit a prediction based on a confidence score. If the agent is not confident enough then it decides to look ahead.

Multi-agent Reinforcement Learning (MARL)[181] formulates the frame sampling procedure as multiple parallel Markov decision processes which pick frames by gradually adjusting an initial sampling. They have a context-aware observation network which models context information among nearby agents along with the historical states of a specific agent. They also have a policy network which generates a probability distribution over an existing predefined action space.

SCSampler[89] is a lightweight clip-sampler that can efficiently obtain the most salient temporal clips. They sample features directly from compressed videos and also from the audio obtained from the video. Attention aware sampling (AAS) [35] uses an agent which discards irrelevant frames using attention. They consider the frame selection procedure as a Markov decision process and train an agent without extra labels by utilising deep reinforcement learning.

While all these approaches showed great results, they have mostly focused on the scenario of untrimmed videos. SCSampler does however report results on Kinetics[20], however, it requires audio as an extra modality. Untrimmed videos contain significant parts of unnecessary data and discarding them is easier than discarding frames from trimmed videos.

In contrast to previous work, we propose a method that instead of selecting frames by considering one at a time, considers them jointly.

2.1.3 Attention and Relational Models

The concept of attention was introduced by [169] for the objective of machine translation. This concept of attention is based on the concept that the neural network will learn how relevant different samples are regarding the desired output state in a sequence, or image regions. These values of importance are specified as weights of attention and are generally calculated at the same time as other model parameters trained for a specific goal. This concept of attention works in a similar fashion for videos as we can capture frame to frame relationships effectively.

Attention has been used in first person action recognition by having a joint learning of gaze and actions [102], by using object-centric attention [158] or via event modulated attention [151]. The use of attention to weigh spatial regions representative of a task was done by generating spatial attention masks implicitly by training the network with video labels [150, 212, 54]. Temporal attention was used for action recognition by detecting change in gaze [134, 151].

LRCN [34] introduced a simple LSTMs for frame-aggregation across time for action recognition. Non-local Networks [178] introduce a residual self attention block in convolutional networks to aggregate information across all temporal and/or spatial locations.

Inspired by the relation-net[162], relation attention was proposed to deal with the task of facial emotion recognition [118]. They believed that having a global level representation of features in addition to the local level representation helps obtain better results. We improve upon this approach by adding relation-temporal attention to add a global representation to our temporal attention.

2.1.4 Transformers for Videos

Transformer architectures, initially devised for NLP tasks [169], have more recently been adapted for video understanding and action recognition tasks. This adaption has given rise to a host of state-of-the-art models such as TimeSformer [10], ViViT [5], VideoSwin [113], and Uniformer [99]. These transformer-based models, harnessing self-attention mechanisms, prove highly efficient in capturing intricate spatiotemporal patterns in action recognition tasks.

A pioneer in transformer-based models for video understanding, TimeSformer [10] adapts the transformer architecture to video by viewing it as a sequence of flattened image patches. ViViT [5] combines spatial and temporal transformers to effectively apprehend spatiotemporal information in video sequences. VideoSwin [113], a hierarchical transformer, applies local windowing for efficiency, enabling the model to manage extended video sequences. VideoBERT [159] is another transformer model that learns joint video and language representations in a self-supervised manner, which can be fine-tuned for various video understanding tasks, including action recognition.

Uniformer [99], a recent development, integrates 3D convolution and spatiotemporal self-attention, achieving a balance between computation and accuracy while addressing spatiotemporal redundancy and dependency. The multi-view transformer model, MTV [193], uses distinct encoders for each video "view", enhancing accuracy as the number of views increases.

The Multiscale Vision Transformers (MViT)[39] model optimizes computation and memory usage by operating at varying resolutions. It focuses on high-level features at lower resolutions and low-level details at higher ones, effectively leveraging both spatial and temporal information in visual tasks. TubeViT[135] introduces a method of sparsely sampling different-sized 3D segments from videos, enabling efficient joint image and video learning and allowing the adaptation of larger models to videos with fewer computational resources.

These models typically have FLOPs in the range of TFLOPs and require training times that span days even on the most potent GPUs/TPUs available. This makes them infeasible for use in settings with fewer resources such as academia. As a result, it is crucial to devise ways to train these models with limited resources while preserving their performance. To this end, our focus lies on the factorised encoder version of ViViT. The late-fusion approach followed serves as a foundation for state-of-the-art approaches in various tasks [193, 22, 175, 190, 194, 70]. We firmly believe that the initialization scheme proposed can be used for future methods working on similar architectures.

2.1.5 Efficient Transformers in Videos

Efficiency within the realm of computational models is a multifaceted concept [29]. There are several cost indicators associated with efficiency, including GFLOPs, inference time, training time, and memory usage. It is noteworthy that models that optimize efficiency in one aspect do not necessarily excel in other dimensions [29].

The TokenLearner model [143] exemplifies this notion. This model proposes a method that adaptively learns tokens, thereby enabling efficient performance in image and video understanding tasks. This method also facilitates effective modeling of pairwise attention over longer temporal horizons or extensive spatial content. The TokenLearner succeeds in reducing the GFLOPs required by ViViT by approximately half. However, it does not bring about any significant changes in the training or inference times of ViViT.

The Perceiver [80] is another Transformer-based model that embraces a comprehensive approach to handling multiple modalities. It does so by incorporating an attention bottleneck and connecting position and modality-specific features with inputs. This strategic integration enables flexible and efficient processing of high-dimensional data.

The Spatial Temporal Token Selection (STTS) framework [174] introduces a dynamic token selection mechanism for spatial and temporal dimensions. The framework ranks token importance using a lightweight scorer network and selects top-scoring tokens for downstream evaluation in an end-to-end training process. While the STTS does reduce the GFLOPs, it does not bring about any significant changes in the training or inference times.

Another similar approach is the TokShift [208], a zero-parameter, zero-FLOPs operator that models temporal relations in transformer encoders by temporally shifting partial token features across adjacent frames. Despite this efficiency, it still requires the same training time as the original model. The integration of TokShift into a plain 2D vision transformer results in a computationally efficient, convolution-free video transformer for video understanding.

The ST-Adapter [131], with similarities to our own work, makes use of built-in spatio-temporal reasoning in a compact design. It allows pre-trained image models to reason about dynamic video content with a small per-task parameter cost, surpassing existing methods in both parameter-efficiency and performance. However, it does not make any alterations to FLOPs or inference time.

Contrasting to the ST-Adapter, our approach employs a spatial only adapter which we demonstrate is sufficient to reproduce the performance of the baseline model while cutting the training time nearly in half. In particular, our proposed method significantly improves training time and training memory usage, addressing a key challenge faced by researchers and practitioners in training video models. However, it is important to note that our method does not alter the inference time in comparison to a standard ViViT model.

We consider overall train time for the same hyperparameters and use the same hardware for a direct comparison. We consider efficiency in this paper as the time saved in the overall training of the model.

2.2 Data-Efficient Learning

In this section we look at data-efficient learning and other concepts related to what we propose. Hence concepts such as deep approaches to centroid learning and data augmentation in videos are listed here. We start with an introduction to zero-shot learning in images and continue to videos and so on.

2.2.1 Zero-Shot Learning in Images

Zero-shot learning (ZSL) is a challenging problem in computer vision, where the task is to recognize object categories without any training examples for them. Various approaches have been proposed to solve this problem in images. One of the early works [93] in this field used attributes, such as color and shape, to describe the object categories and mapped them to a visual space. They then used a nearest-neighbor classifier to recognize unseen object categories. However, this approach suffers from the semantic gap problem, where the attributes do not always correlate well with the visual features.

To address this problem, more recent works have explored the use of deep learning techniques to learn a joint embedding space for the visual and semantic features. One such approach is proposed by Frome et al. (2013) [47], where they used a deep neural network to learn a joint embedding space for the visual and textual features of the objects. They then used a nearest-neighbor classifier to recognize unseen object categories. Another approach is proposed by Socher et al. (2013) [156], where they used a recursive neural network to learn a compositional representation of the textual descriptions of the object categories.

More recently, there has been a trend towards using generative models to solve the ZSL problem. One such approach is proposed by Xian et al. [185], where they used a generative adversarial network (GAN) to generate visual features for unseen object categories. They then used a joint embedding space to match the generated features with the semantic features and recognize unseen object categories. Another approach is proposed by Schonfeld et al. (2019) [145], where they used a GAN to generate visual features conditioned on the textual descriptions of the object categories. ZeroGen [196] uses pre-trained language models to synthesize a dataset for a zero-shot task and then trains a small task model on it. NereNet [108] generates unseen samples by combining noise from similar seen classes with unseen class attributes using GAN. CMC-GAN [195], performs data hallucination of unseen classes by performing semantics-guided intra-category knowledge transfer across image categories.

2.2.2 Zero-Shot Learning in Videos

The initial study by Rohrbach et al [141] utilized script data from cooking activities to facilitate the transfer of knowledge to unseen categories. Gan et al [50] considered each action class as a domain and tackled the problem of identifying semantic representations as a multisource domain generalization task. To extract semantic embeddings of class labels, popular approaches employ label embeddings such as word2vec [120], which solely requires class names. Several methods have used a shared embedding space between video features and class labels [192, 191], error-correcting codes [137], pairwise relationships between classes [48], interclass relationships [49], out-of-distribution detectors [116], synthetic features [121, 122] and graph neural networks [52].

ReST [104] jointly encodes video data and textual labels for zero-shot action recognition. In ReST, transformers are utilized to conduct modality-specific attention. On the other hand, JigSawNet [136] models visual and textual features jointly but disassembles videos into atomic actions in an unsupervised manner and establishes group-to-group relationships between visual and semantic representations instead of the one-to-one relationships that CLASTER and ReST establish.

2.2.3 Semantic Embeddings

To obtain semantic embeddings of action class labels, earlier works use word2vec [120] directly on the class labels. However, the word2vec approach averages the embedding for class labels with multiple words, giving equal weight to each word. However, this causes class names to lose context. For example, the class “pommel horse" is a gymnastic class, but using word2vec makes the embedding closer to “horse riding" or “horse racing" in the word2vec space. A recent solution is to use elaborate descriptions [23] based on the principle of Elaborative Rehearsals (ER), which replace each class name with a class definition. An object description was also used to describe the objects used in the particular action. This resulted in a significant boost in performance.

2.2.4 Evaluation Protocol Strategies for Zero-Shot Learning

In prior research endeavors within this domain, considerable attention has been paid to the effects of pre-training on classes that overlap with those utilized in the test set, particularly in the realm of image analysis [186]. This investigative work primarily involved the computation of the degree of overlap between testing datasets and ImageNet [31], the latter of which is a commonly used resource for pre-training models. The classes that presented a high degree of overlap with ImageNet are typically the ones used for training, while the classes that demonstrated little to no overlap are used for testing. Figure 5.1 offers a clear visual representation of the evaluation protocol proposed for managing this overlap issue.

This approach diverges significantly from traditional evaluation protocols, which typically involve the random selection of classes from the entire pool of classes, without any consideration for the level of overlap between these selected classes and those used in the pre-training dataset. In contrast, our approach involves a rigorous filtration process to exclude all classes that demonstrate a high threshold of either visual or semantic similarity (as elaborated upon in Sec. 5.4).

Roitberg et al. [142] proposed another approach to tackling this issue, but in the video space. They suggested scrutinizing the overlapping classes in video datasets and implementing a corrective method that would automatically exclude categories that are similar. This process involves the application of pairwise similarity analysis to labels within the same dataset.

While they demonstrated that using pre-trained models could lead to improved accuracy due to class overlap, their evaluation was limited to a single dataset and focused solely on the semantic similarity of labels. We believe that incorporating visual similarity into the analysis could reveal additional overlapping classes, such as "typing" in UCF101 and "using computer" in Kinetics. Therefore, in our proposed split, we include both semantic and visual similarity as key criteria for class selection.

Busto et al. [16] made a contribution to this discussion by offering a mapping of shared classes between UCF101 and Kinetics, as part of a domain adaptation problem. This involved manually identifying semantic matches based on class names. However, their approach did not consider visual and semantic similarity in their analysis, which

is evidenced by the inclusion of classes such as "typing" and "writing on board" in UCF101, classes that were not identified as similar to any Kinetics class. Moreover, they classified "floor gymnastics" as an unknown class in UCF101, even though Kinetics includes a very similar class, "gymnastics tumbling".

Based on our analysis of visual and semantic similarities, we found that the "typing" class in UCF101 has a significant overlap with the "using computers" class in Kinetics, a connection not identified by Busto et al. Consequently, our proposed set of classes differs slightly from theirs.

In recent research, end-to-end training has been proposed as a promising approach to ZSL in video classification [15]. To maintain the premise of ZSL, the authors suggest training a model on Kinetics after removing the set of overlapping classes (identified using semantic matching), and using this as a pre-trained model.

While this method certainly holds promise, it is also highly computationally intensive, and as such may not be feasible for all use cases. We also demonstrate that using a better backbone (as discussed in Sec 5.5.4) leads to improved accuracy, which further reinforces the computational demands of end-to-end training. Consequently, we propose that using our proposed class split, while allowing for the use of any backbone, could offer a more practical and efficient approach.

2.2.5 Deep Approaches to Centroid Learning for Classification

Centroid-based methods are a powerful approach for feature learning in classification tasks. By clustering the feature space and aggregating residuals to centroid vectors, methods like VLAD (Vector of Locally Aggregated Descriptors) [3] can create highly discriminative representations. VLAD encodes visual words by summing residuals between descriptors and vocabulary cluster centers. NetVLAD extended VLAD with learnable clusters and end-to-end training within CNNs. ActionVLAD [55] further adapted VLAD for video analysis, aggregating frame-level features across time to yield video descriptors. By accumulating frame features into video-level visual words through deep centroid alignment, ActionVLAD achieved state-of-the-art results in action recognition. The success of VLAD, NetVLAD, and ActionVLAD demonstrates how deep centroid learning enables robust video classification, providing global video representations without relying solely on local spatio-temporal features.

2.2.6 Reinforcement Learning for Data Valuation

The quantification of data value for a specific machine learning task, known as data valuation, has numerous applications such as domain adaptation, corrupted sample detection, and robust learning. Various techniques have been proposed to estimate data values based on different criteria, including influence functions, Shapley values, leave-one-out errors, and data deletion. However, these methods are computationally expensive, necessitate model perturbations or retraining, and do not consider the interactions among data points. Recently, an adaptive approach to data valuation using reinforcement learning [200], in which data values are jointly learned with the predictor model using a data value estimator that is trained using a reinforcement signal reflecting task performance.

Synthetic sample selection [197] for medical image segmentation is an under-investigated research area that focuses on the quality control of synthetic images for data augmentation purposes. Synthetic images are not always realistic and may contain misleading features that distort data distribution when mixed with real images. As a result, the effectiveness of synthetic images in medical image recognition systems cannot be ensured when they are randomly added without quality assurance. A reinforcement learning-based synthetic sample selection approach is proposed in which synthetic images containing reliable and informative features are chosen.

However, none of the above approaches consider the extreme case of zero-shot learning. In the case of zero-shot learning, synthetic features are often biased towards seen classes, and the generated synthetic features do not represent the true distribution. Training a model to generate realistic-looking features will produce features similar to the training distribution without any guarantee on the effect on zero-shot evaluation. Therefore, we propose a data valuation method for synthetic features based on classification performance rather than their realism.

2.2.7 Reinforcement Learning for Zero-Shot Learning

Reinforcement learning for zero-shot learning has only recently begun garnering attention as a promising research direction. In the domain of zero-shot image classification, pioneering work by Liu et al. [106] combined ontology information and reinforcement learning to adaptively determine the discriminative degree of hierarchical classification rules. Through this ontology-augmented reinforcement learning approach, they demonstrated initial success in tackling zero-shot recognition for images. Beyond

images, zero-shot text classification has also seen recent studies leveraging reinforcement learning for improved performance when no labeled examples are available for new classes. Ye et al. [198] developed a self-training method that employs reinforcement learning to learn an effective data selection strategy, thereby enabling more reliable selection of useful unlabeled instances. Additional efforts have harnessed reinforcement learning for zero-shot learning in diverse problem settings, including task generalization [129], active learning [41], and video object segmentation [60].

2.2.8 Data Augmentation for Video Action Recognition

Standard data augmentation techniques in action recognition include horizontal flip and cropping, where new videos are created by selecting a box at each frame, and then resizing the resulting video to have the same size as the original one. While this strategy helps, generated videos do not add much diversity to the training set.

Recent efforts such as ActorCut [223] and VideoMix [204] increase the diversity of new video samples by cutting and pasting the foreground of one video onto another. This general technique of combining two data samples has proven to be quite effective, even in the image domain [203]. However, the resulting videos are not very realistic, and are used for training regardless of their quality. Zhang et al. [215] go one step further and synthesize new samples using GANs, and use “self-paced selection” to train, starting with easy samples and progressively choosing harder samples.

Instead, we propose to create realistic data samples by segmenting, inpainting and blending the foreground of one video onto the background of another. Crucially, we learn to discard novel video samples that are not expected to be useful for classification, overall producing a more accurate data augmentation strategy.

2.2.9 Learning to Augment Data

The idea of learning to augment data has been used in other computer vision problems. In the image classification domain, this strategy has been done using the final classification loss as the training criterion [98], augmenting in feature space [32], and learning data augmentation policies [26]. As in this paper, in the image domain it has been noted that the search space for data samples can be large and thus expensive [27].

Other computer vision domains like low level vision, also struggle with data dependency, as creating ground truth is particularly hard. In optical flow, AutoFlow [160] recently introduced the strategy of learning to generate good training data for a target dataset.

2.2.10 Semi-supervised Video Action Recognition

Semi-supervised learning (SSL) also aims to reduce data dependence by learning from large sets of unlabeled samples and a small set of labeled ones. SSL in images has been widely explored. For example, some strategies include giving pseudo-labels [4, 97] to samples where the classifier has high confidence, and adding these to the labeled training data. Other common approaches use consistency regularization [91, 92, 165]. Approaches that combine consistency regularization and entropy minimization [68] have shown to be very effective in tackling the SSL task in images such as MixMatch [12] and RemixMatch [11].

SSL in videos however, has not been explored as much. One of the early works used extreme learning machines [79] to perform SSL on videos. Recently, VideoSSL [84] and Temporal Contrastive Learning (TCL) [154] leverage SSL in videos. VideoSSL [84] uses pseudo-labels and object cues from unlabeled samples to guide the learning process. TCL [154] use a two-pathway contrastive learning model using unlabeled videos at two different speeds with the intuition that changing video speeds do not change the action being performed.

Chapter 3

Efficient Action Recognition by Key Frames Selection

3.1 Introduction

Video processing is computationally expensive. At the same time, the amount of video content being generated is increasing fast and constitutes a large part of the computation of many big social media platforms. Traditionally, most efforts in action recognition have focused on improving accuracy by creating larger architectures. These architectures take as input either a frame or a set of frames (also called a *clip*) and produce a prediction. These predictions are then aggregated over time. The frames or clips are either sampled densely [153, 202] or randomly [176].

Videos, however, provide an opportunity for reducing computational cost in multiple ways. First, videos contain highly temporally redundant data, making it easier to skip parts without losing much information [40]. Second, some parts of a video can be more discriminative than others, due to their content, or other phenomena like blur, occlusions, etc. Supporting this intuition, experimentally that using an oracle to make an optimal selection of frames (or clips), produces more accurate classification results than using the entire video [77]. This is shown in Figure 3.1. Additionally, many action classes in standard datasets do not require motion or temporal information to be identified [149]. For a human observer, a few still frames are often discriminative enough. This suggests that large parts of a video can be discarded.

Several recent works [89, 183, 221] have successfully leveraged these principles to reduce computational cost at test time. These methods have used a common strategy: they use an inexpensive way to decide which regions of the video are important and discriminative, and only process those with an expensive method. This general problem has been referred to as frame or clip selection.

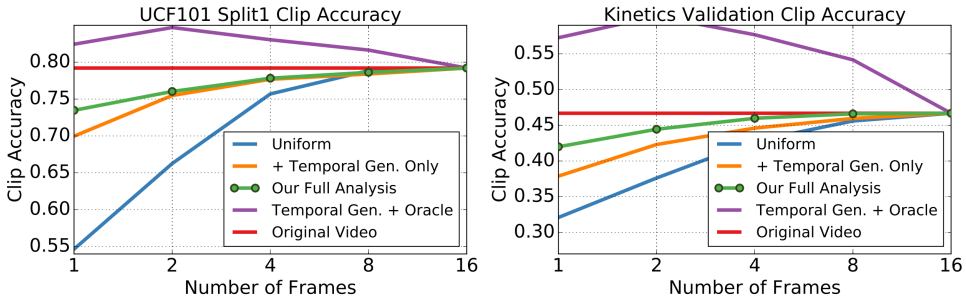


Figure 3.1: Examining the impact of good frame selection using an oracle as shown by Huang et al. [77].

While very successful, most frame and clip selection methods have focused on a particular domain of action recognition, namely long, and frequently sparse videos with a typical length of a minute or more, e.g. ActivityNet [17], Sports1M [85], FCVID [83], Youtube 8M [1]. This is indeed the domain where discarding portions of a video is easier and has potentially the largest effect. In contrast, the problem of frame selection in short videos of a few seconds remains much less explored, probably due to its difficulty.

In this chapter we propose a method to do frame selection in the core, standard activity classification setting of trimmed video clips. Part of the challenge in this setting is that “good" frames are often temporally close together within a video. Since most existing frame selection methods consider the value of choosing a frame one at a time, the selected frames tend to only represent part of the action. In other words, the diversity of frames, and their ability to tell a story are disregarded. We also show that using language features along with visual features helps improve the performance. The goal of this chapter is to learn to select the best set of frames to improve accuracy of the model whilst reducing the memory required to train such a model.

To handle these challenges, we propose a model that, in addition to considering the discriminative value of a single frame, also considers its relation to others in a video. We do this by using an attention and a relational network [118, 162], that examines the value of frames jointly. We learn our **S**ampling through **M**ulti-frame **A**ttention and **R**elations in **T**ime, which we dub the *SMART*selection network.

We test our SMART frame selection network on several trimmed action recognition datasets, including Something-something, UCF101 and subsets of Kinetics. We observe that in all of them the proposed method outperforms the baselines, including using the full video, while reducing the computational cost by a factor of 4 to 10,

depending on the dataset. We also test the proposed method on the untrimmed setting in ActivityNet and FCVID, where we get higher accuracies than all previous work on frame selection. Further, we extend our frame selection approach to select frames that are then passed at test time to deep action recognition models and show that we obtain state-of-the-art results on UCF101 and HMDB51 which are trimmed video datasets, showing that frame selection can be an important step to improve accuracy in trimmed action recognition.

3.2 SMART Frame Selection

The proposed approach is designed to use a small portion of the overall computational cost in selecting the best frames. These frames will then be classified using a more computationally expensive model. Therefore, we use a very lightweight representation of the frames as input to the SMART frame selection model.

The model consists of two streams. The first considers the information of the frames one at a time, and outputs a score δ_i for each frame i, which represents how useful the frame is for classification. The second stream considers the entire video at a time. It takes as input pairs of frames, and uses an attention and relational network to also obtain a score γ_i of how useful these pairs of frames are. Both scores are then multiplied, to obtain a final score of how good each frame is. Given a budget of n frames, we now select the top n frames with the highest discriminative score, and use an expensive, high quality classifier for the final prediction. An overview of the method can be seen in Fig. 3.2. We now describe each of the components in detail.

3.2.1 Feature Representation

We choose the lightweight MobileNet [144] to extract the visual features of each frame, to minimize the computational cost of this stage. We also make the observation that, in addition to the visual features, we can use language features associated with the content of the frame. The intuition behind this is to enrich the representation with terms that are related to the content of the image. One could imagine that if for an action class like “kayaking", having associated words like *water*, *boat*, or *paddle* can help discrimination in cases where the kayak is not apparent visually.

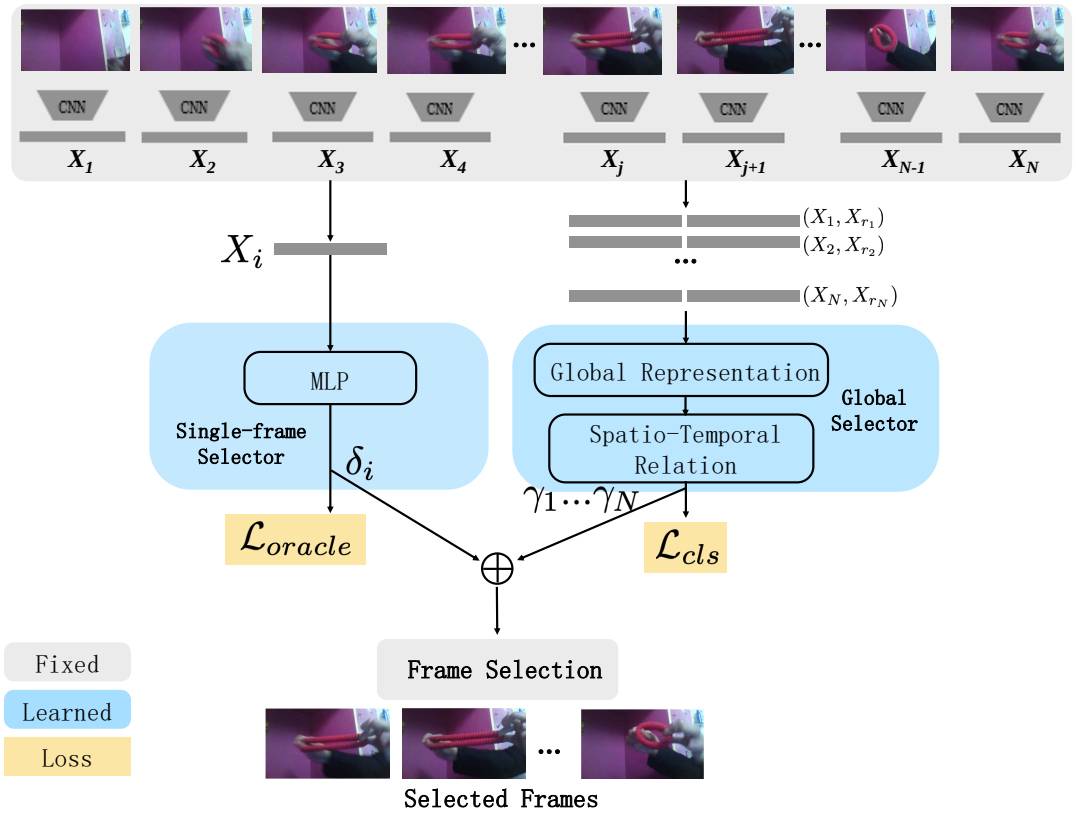


Figure 3.2: Overview of the SMART frame selection. The model consists of two streams. The first considers the information of the frames one at a time and the second stream considers the entire video at a time. Here, $\mathscr{L}_{oracle}$ is the MSE loss used to distill frame scores from an expensive model and $\mathscr{L}_{cls}$ is the categorical cross entropy loss used for classification.

We run Mobilenet pre-trained on Imagenet on the frames, and take the top 10 Imagenet classes with highest probability. The names of these classes are then embedded with a pre-trained GloVe [132] over Wikipedia 2014 and average over the 10 classes. The language embedding is then concatenated with the visual features resulting in a feature vector X_i for each frame i.

3.2.2 Single-frame Selector

This stream is designed to be extremely fast. We build on the observation from [77] that an oracle that looks at the predictions from an expensive network, and selects the frames with the highest confidence for the ground truth class, actually outperforms using the entire video for prediction. Thus, we use a simple multi-layer perceptron (MLP) that takes as input a feature vector X_i, and computes the confidence of the

classification for the ground truth. This MLP has 2 layers, and is trained using the oracle mentioned before wherein each frame outputs the probability of that frame with respect to the ground truth class. At training time we can obtain the ground truth probability of each frame using an expensive model trained on the dataset we are looking at. At test time δ_i is predicted by the trained model as the importance score of a particular frame.

Overall, this module aims to minimize the loss $\mathscr{L}_{oracle}$ that is described in Eq. 3.1, given ground truth classification scores of the expensive model y_i and predicted scores by the lightweight model $\hat{y}_i$ and N is the total number of frames for a particular video. This loss is calculated per video..

$$\mathscr{L}_{oracle} = -\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.1}$$

3.2.3 Global Selector

The multi-frame discriminator is designed to use information across frames for selection. This is done by first obtaining a global representation of the video using an attention model over the entire video. Given this global representation, the temporal relationships across frames are learned using a relation model and a long short-term memory (LSTM) network. While lightweight and easy to learn, this network provides information about how useful frames are when considered globally. The global selector uses a relational model to learn temporal relationships across frames over the entire video. This produces an inexpensive global representation of the video.

Pairs of frames. Consider an input sequence $X = (X_1, ..., X_N), X_i$ represents the concatenated visual and categorical features in frame i and N represents the total number of frames. We use ':' to notate concatenation here. For each frame, we concatenate a second, randomly selected frame, $X_r^i, r \in \{1, ..., N\}$. The random frame is always chosen from the subsequent set of frames to capture the temporal changes that occur in actions. Some actions will be most recognizable when these pair of frames are only a few frames apart, while others will be more recognizable when they are further apart. This random choice allows the model to be flexible and capture the temporal changes in different classes. The input to the attention model is the concatenation of both vectors $Z_i = [X_i : X_r^i]$. The output of the network are a set of temporal relation-attention weights $\gamma_1, \gamma_2, .., \gamma_N$. This helps our model to obtain temporal information.

Attention Module. The coarse self-attention weights α_i are first calculated using a fully connected layer and a sigmoid function [118]. The mathematical representation is in Eq. 3.2, where U are network parameters. We now aggregate the input features using these self-attention weights. We do this in order to obtain a global representation Z' of the frame features, as in Eq. 3.2.

Self-attention weights are learned using individual frames with the help of non-linear mapping. To obtain a more reliable form of attention, we need both local and global features to be used. Z' is aggregated from all local features and hence contains the global information of the video. Hence, by using Z' we can further refine the attention weights by modeling the relationship between local frame features and Z'.

$$\alpha_{\mathrm{i}} = \sigma(\mathrm{Z_i^T U}) \quad and \quad \mathrm{Z'} = \frac{\sum_{\mathrm{i=1}}^{\mathrm{N}} \alpha_{\mathrm{i}} \mathrm{Z_i}}{\sum_{\mathrm{i=1}}^{\mathrm{N}} \alpha_{\mathrm{i}}} \tag{3.2}$$

Relation Module. We can add a sample concatenation and another fully connected layer [162] to estimate a relation-attention weight β. The input is the concatenation of the pairs of frames Z_i and the global representation of the video Z' and the output for each Z_i is β_i Θ_1 is a parameter of the fully connected layer and σ represents the sigmoid function. Using this we have obtained frame attention weights. However, we also want temporal attention weights. We use an LSTM to capture sequential per frame changes. The input to the LSTM at each time step is the dynamic weighted sum using the relational self-attention weights 'ω_t'. This is represented in Eq. 3.3.

$$\beta_{\mathrm{i}} = \sigma([\mathrm{Z_i} : \mathrm{Z'}]^{\mathrm{T}} \Theta_1) \quad and \quad \omega_{\mathrm{t}} = \sum_{\mathrm{i=1}}^{\mathrm{t}} \beta_{\mathrm{i}} \mathrm{Z_i} \tag{3.3}$$

The temporal attention weights are then calculated as shown in Eq. 3.4 and Eq. 3.5. It is dependent on the previous time step output of the LSTM and the input at that time step. b is a bias vector.

$$\mathrm{h_t, m_t} = \mathrm{LSTM}(\omega_{\mathrm{t}}, \mathrm{h_{t-1}, m_{t-1}}) \tag{3.4}$$

$$\lambda_{\mathrm{t}} = \mathrm{softmax}(\mathrm{V h_t + b}) \tag{3.5}$$

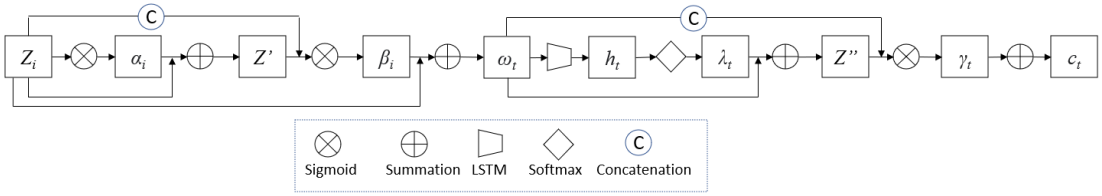


Figure 3.3: Steps involved in calculating the attention weights and intermediaries involved.

To compute the relational-temporal weights, we follow the procedure used to obtain relational-frame attention weights, as in Eq. 3.6. Here Θ_2 is simply a network parameter.

$$Z'' = \frac{\sum_{t=1}^{N} \lambda_t \omega_t}{\sum_{t=1}^{N} \lambda_t} \quad and \quad \gamma_t = \sigma([\omega_t : Z'']^T \Theta_2) \tag{3.6}$$

Using these γ_t we can obtain an attended content vector c_t at time 't' using Eq. 3.7. Here h_i refers to the hidden state of the LSTM at i. For classification, c_t is fed into an MLP to generate the predicted label y. Overall, this module aims to minimize the loss $\mathscr{L}_{cls}$ that is described in Eq. 3.8, given ground truth labels $\hat{y}_t$. Steps to calculate all the attention weights and intermediaries can be seen in Figure 3.3.

$$c_t = \sum_{i=1}^{t} \gamma_i h_i \tag{3.7}$$

$$\mathscr{L}_{cls} = -\sum_{i=1}^{C} \hat{y}_i \log(y_i) \tag{3.8}$$

3.3 Experimental Analysis

In this section we describe all experiments that we conduct to test the behavior of the proposed SMART frame selection network. In the qualitative results analysis subsection we describe the ablation experiments that led to the specific design of the network, justify each of the components and measure their impact. We analyze the behavior of the frame selector components individually (single frame and global selection). We show the generality of the proposed method on several datasets. Finally, we compare to other state-of-the-art frame sampling methods in the untrimmed setting, showing that the proposed method still produces higher accuracy.

3.3.1 Experimental Setup

Datasets. We use 6 of the most popular benchmark datasets throughout our experimental analysis. We use the Something-something-v2 dataset [67] for our extensive ablation study. The purpose of the ablation study is to drive the design choices we make through experimental evidence.

We choose this for the ablation study because we think that it contains the types of actions where relations of frames over time matter more. This allows us to truly evaluate the effect of the global model that we propose. In particular, the action classes in this dataset are designed to focus on the action, (eg.: “put something", “pushing something") instead of on an object (eg.: “playing guitar"). As a result, actions tend to have more temporal structure, and relations across frames may matter more.

The Something-something dataset has a total of 168,913 training videos and 24,777 validation videos with a total of 174 classes. After the ablation study, we show the generality of our approach by testing it in other datasets as well. The Kinetics [20] dataset is one of the most widely used large-scale datasets in action recognition.

We use two subsets [149] of Kinetics that have been identified as containing mostly temporal information and mostly static information. In our experiments we refer to these as Kinetics-Temporal and Kinetics-Static. These subsets were created using a human perceptual test, where users are asked to identify the class of a video where the frames are not in order, therefore removing temporal information. Static classes are those that users could identify without temporal information. Temporal classes are those that users were not able to identify when the frames were not in order. These two subsets provide an interesting setting for experimentation, because the statistics of the data (eg.: size of videos, frame rate, platform that the videos came from, etc) are common for both datasets, but the nature of the action classes differ. Each of the two splits contains 32 classes. The temporal subset consists of 26509 videos and the static subset consists of 23675.

For our generality tests, we also use the well-known UCF101 [157] dataset which contains 101 classes and about 13K videos. This was one of the first widely used action classification datasets and thus we think that it can give additional useful reference for comparison.

We also extend our approach as a pre-processing step for more complex models and compare performances on HMDB51 which contains 51 classes and 6849 video clips along with UCF101. Previous frame selection for action recognition have focused on untrimmed videos. Therefore, they have not tested on any of these datasets and have not released code so that we could test the methods on these datasets ourselves.

In order to compare with them, we use ActivityNet [17] and FCVID [83]. ActivityNet consists of 19994 videos, and contains 200 classes. As the testing labels are not available publicly, the reported performances are on the validation set. FCVID is made up of 91, 223 videos taken from YouTube having an average duration of 167 seconds, and these are annotated into 239 classes.

Implementation Details. For selecting frames, we use visual features obtained from MobileNet v2 [144]. Taking the 1000 class predicted vector, we then choose first 10 most probable classes and store in a vector. This is then converted to a one-hot encoding vector using a word embedding matrix vocab. We use pretrained GloVe [132] over wikipedia 2014 and do a dot product between the 2 vectors to obtain a language embedding which is then concatenated with the visual features to have a set of concatenated feature vectors. We show in our ablation study that having a semantic embedding helps our frame selector to make better informed decisions. After the frame selection is done, we can use a more expensive and high-quality feature representation. In our experiments, we use three different backbones: ResNet-152, ResNet-101 [74], and Inception-v3 [163]. The backbones are pre-trained either on ImageNet [31] or Kinetics. These architectures are representative of the state-of-the-art, and are chosen according to what other methods that we want to compare to have used. While having multiple backbones and pre-training datasets makes experiments more cumbersome, it allows us to simply swap the frame selection component and directly observe the effect on the accuracy without any other confounders.

We use Pytorch for implementation. All frames are resized to 224x224. We use mini-batch stochastic gradient descent, with a momentum of 0.9. We run 200 epochs on UCF101 and the Kinetics subsets, and 100 epochs on Something-something dataset and Activitynet due to the computational requirements for these larger scale datasets. We use a batch size of 128 for UCF101 and the Kinetics subsets and a batch size of 64 for the Activitynet and something-something datasets. The initial learning rate is set at 0.0001 and reduces by 10 after every 25 epochs.

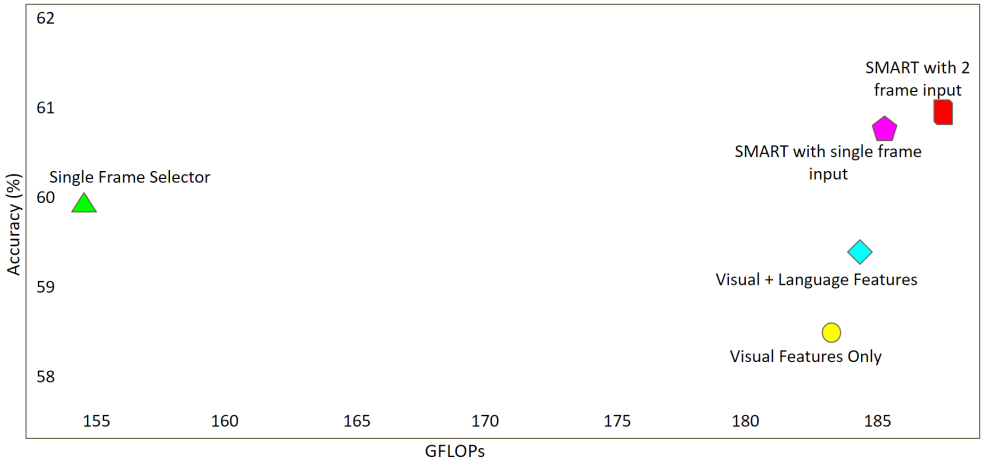


Figure 3.4: Ablation study to determine the effect of each of the components of the SMART frame selection network. We use 26 frames for all experiments. Something-something-v2 dataset. We see that using visual and language features performs better than visual only. We also see that the single frame selector alone helps boost performance at minimal costs whereas using the entire architecture with 2 frames input gives us the best results.

Baselines. We compare the performance of our frame selection model with that of random and uniform frame selection. Random frame selection picks frames uniformly at random from the entire video, while uniform frame selection picks frames that are evenly spaced. Once the frames are picked, we predict an action by average pooling the predictions of every selected frame. In addition to these baselines, we compare to other state-of-the-art frame sampling methods, including Adaframe [183], FastForward [40], FrameGlimpse [199] and MARL [181].

3.3.2 Ablation Study on the SMART Frame Selection

Here, we look at the impact of the feature representation (visual and categorical), the choice of frame selector (the global multi-frame selector and the single-frame discriminator), and the use of pairs of frames. We use the Something-something-v2 [67] dataset for this study. Fig 3.4 shows the results.

We first test and compare the use of the simple visual features (from MobileNet), then combining them with the categorical ones (from GloVe). We use the global selector for this initial test. We observe that the addition of semantic language features helps, supporting the intuition that using words related to the content of a frame actually helps in the context of frame selection. Using that, we examine the effect of different

selectors: the single-frame selector, the global selector, and the combination of both. We observe that the combination of both is the best choice, suggesting that these two selectors behave in different but complementary ways. We analyze this behavior further in the following section.

We also measure the impact of using pairs of frames as input to the global selector. While we use a relational component inside the selector, adding pairs would give an additional mechanism to consider frames jointly. We observe that this does indeed help. Since we use random frames, we report the average accuracy in Figure 3.4. The standard deviation on the Something-something-v2 [67] dataset on 10 random runs was 0.067 using Inception v3 and 0.082 using Resnet-152.

3.3.3 Analysis of the Behavior of SMART Frame Selection

Number of Selected Frames. First, we measure the impact of selecting different number of frames. For this, we vary the number of selected frames between 10 and 50, and measure the impact on accuracy and GFLOPs and compare with random and uniform sampling. The results are in Fig. 3.5 (a). We choose the Something-something dataset, and the Inception-v3 as backbone.

For fair comparison, we use the same features (language embeddings concatenated) for both random and uniform sampling as well. We see that as the number of frames increases, the uniform and random frame selection perform strictly. The proposed method performs much better than these baselines across frames. It is also interesting that the accuracy increases and reaches a peak, and then slowly drops in performance. This behavior confirms the intuition that there is a sweet spot in the number of frames, and that using more than that, will include frames that are harder to classify, which will pollute the prediction.

Frame Selection Across Similar Classes. We now plot the combined frame score from both selectors, to analyze its behavior. We plot the frame score of classes that are semantically related, to compare if frames scores are also similar. The Something-something dataset contains groups of classes that are very related. We sample 25 videos within a class, for 5 classes and average the importance scores. The plots are shown in Fig. 3.5. We see a strong resemblance for actions involving “pushing", suggesting that the general structure of the action has been captured by the model. We also compare actions involving “throwing", which show peaks for frame importance at similar time intervals.

Selecting Frames with the Global Selector vs. the Single-frame Selector. We measure whether the pattern of frame selection from the global selector tends to be different from the pattern from the single-frame selector. For this, we randomly sample 25 videos within a class, and score each of their frames with the two selectors. We plot the average score at each frame, in Fig. 3.6. Again we use the Something-something dataset and Inception-v3. While the scores from the single-frame selector change more erratically, the score from the global selector seems to be more temporally consistent. This suggests that frames scores from the global selector are actually more structured.

Selected frames. It is not only intriguing but also insightful to delve into an analysis of the frames that have been chosen for one particular class, as depicted in Figure 3.7. The class is "pulling something so that it gets stretched". As we examine the figure, we notice that it's not simply an assortment of frames chosen at random. Rather, these individual frames, though they may be few in number, provide a concise and clear narrative of the action in question.

3.4 Quantitative Results Analysis

3.4.1 Generality of SMART on Additional Datasets

UCF101. Results for UCF101 can be seen in Table 3.1. As in the Something-something dataset we observe that the SMART selection outperforms the baselines of random and uniform, regardless of the number of frames. We also see that it outperforms using the full video (for all except for using 10 frames) while the "sweet spot" of number of frames is slightly larger than in the Something-something dataset. This is consistent with the fact that videos in UCF101 are about 7 seconds long, while Something-something are closer to 3 seconds. Therefore it makes sense that the proportion of "good frames" stays the same.

Subsets of Kinetics. We also show results on the subsampled 32 temporal classes of Kinetics and the 32 static classes [149]. These two subsets are described in detail in the Datasets section. Results are shown in Table 3.2. We see that the pattern is similar to all other experiments: SMART outperforms the other sampling baselines, and for the optimal number of frames, it outperforms the full video as well. Also the proposed method behaves slightly differently in these two subsets of Kinetics. In the

Method	Accuracy			GFLOPs		
#F	*10*	*26*	*50*	*10*	*26*	*50*
Random	63.2	68.3	70.2	110	277	652
Uniform	63.8	69.1	70.7	110	277	652
SMART	72.8	**75.3**	**75.5**	164	331	706
All frames	**74.6**	74.6	74.6	1969	1969	1969

Table 3.1: Baseline frame sampling techniques vs our SMART with ResNet-152 backbone; #F: #frames. Results on UCF101 dataset

Method	Temporal, Acc			Static, Acc			GFLOPs		
#F	*10*	*26*	*50*	*10*	*26*	*50*	*10*	*26*	*50*
Uniform	59.4	60.3	62.1	60.1	60.6	61.2	110	227	652
Random	60.1	60.8	62.7	60.3	60.9	61.7	110	227	652
SMART	63.1	**64.8**	**65.4**	61.4	61.9	**62.6**	185	353	728
All frames	**64.1**	64.1	64.1	**62.4**	**62.4**	62.4	2761	2761	2761

Table 3.2: Baseline frame sampling techniques vs our SMART with ResNet-152 backbone; #F: #frames. Results on Kinetics dataset subsets: Temporal and Static

Kinetics-Static subset, where temporal information is less rich, the improvement is smaller. This is consistent with the expected behavior of the proposed method, which considers the entire video globally, and is able to make selections that are more temporally aware.

3.4.2 Performance on Untrimmed Datasets

Finally, we compare the performance of the proposed method to previous work for untrimmed video. Therefore, we test on the ActivityNet [17] and the FCVID [83] datasets. We show that using fewer frames than recent state-of-the-art approaches such as Adaframe [183], FastForward [40] and FrameGlimpse [199] we can obtain a higher accuracy. However, we access all frames which makes our approach slower than these. We also compare with LiteEval [182] which is a lightweight action recognition model. We also compare our approach to Multi-agent Reinforcement Learning (MARL) [181] approach and Dynamic Sampling Networks (DSN) [220] by using a model pretrained on Kinetics for fair comparison. Table 3.3 shows the results.

Method	Pre-trained	Back-bone	ActivityNet #F	ActivityNet Acc	FCVID #F	FCVID Acc
FastForward	Imagenet	Inc v3	9.61	58.1	15.34	73.3
FrameGlimpse	Imagenet	VGG16	9.42	62.8	9.26	71.7
Adaframe	Imagenet	Res101	8.65	71.5	8.21	80.2
LiteEval	Imagenet	Res101	-	72.7	-	80.0
SMART	Imagenet	Res101	8	71.4	8	80.8
SMART	Imagenet	Res101	10	**73.1**	10	**82.1**
DSN	Kinetics	Res18	10c	68.0	-	-
DSN	Kinetics	Res34	10c	82.6	-	-
MARL	Kinetics	Res152	25	83.8	-	-
SMART	Kinetics	Res152	24	**84.4**	-	-

Table 3.3: Results on ActivityNet and FCVID of the SMART frame selection. Compared to recent state-of-the-art methods, the proposed method outperforms their accuracy. #F: Number of frames, 10c corresponds to 10 clips used instead of frames

3.4.3 Extension of SMART as a Pre-processing Step

We look at the results of using our approach as a pre-processing step to Temporal Segment Networks (TSN) [176] and using the selected frames at inference in Table 3.4. To the best our knowledge this gives us state-of-the-art results on UCF101 and HMDB51. We compare with other recent state-of-the-art approaches such as two-stream networks [153, 57], DynaMotion [7], I3D [20] and Knowledge Integration network (KINet) [213] which are among the latest state-of-the-art approaches. We also add comparison with AAS [35] as a frame selection approach.

3.4.4 Improving Performances of Other Models

Here, we show that using our model to select frames and pass the selected frames at inference helps to improve the performance of models such as I3D [20], STM-ResNet [44] and ISTPAN [37]. This can be seen in Table 3.5.

Method	Backbone	UCF101	HMDB51
Two-stream	VGG	92.5	62.4
I3D	Inc v3	98.0	80.7
DynaMotion + I3D	Inc v3	98.4	84.2
TSN	BN-Inc	94.2	69.9
KI-Net	Res-152	97.8	78.2
AAS	TSN	94.6	71.2
SMART	TSN	**95.8**	**74.6**
AAS	TSN+Kinetics	96.8	77.3
SMART	TSN+Kinetics	**98.6**	**84.3**

Table 3.4: Extending SMART as a pre-processing step to state-of-the-art deep learning approaches. The '+ Kinetics' indicate that the backbone is pre-trained with Kinetics.

Method	UCF101	HMDB51
ISTPAN	95.5	70.7
ISTPAN + SMART	**96.4**	**72.1**
I3D	98.0	80.0
I3D + SMART	**98.2**	**81.1**
STM-Resnet	94.2	68.9
STM-Resnet + SMART	**94.9**	**69.7**

Table 3.5: Extending SMART to other approaches

3.5 Limitations

One limitation of SMART is that it still requires running the frame selection model over all frames in a video, which can be computationally expensive especially for long videos. Although we use a lightweight model, processing all frames limits the approach's ability to scale to large video datasets. More recent work has looked at ways to avoid extracting features for all frames upfront. For example, AdaFrame [183] proposes a reinforcement learning policy to select keyframes in a sequential manner, not needing to process the full video initially. Other methods like LiteEval [182] and Uniformer [99] focused on frame sampling or feature aggregation techniques to avoid feature extraction on all frames. These approaches help address SMART's limitation of having to process all frames even with a lightweight model. However, SMART's core ideas around using importance sampling through learning relationships among frames for action recognition are still relevant. More recent methods such as SMS

[216] and AFNet [214] build on similar concepts but tackle the computational challenges in different ways, like through heurestic searching combined with supervised learning or adaptive frame resolutions. So newer approaches address SMART's limitations around efficiency but build on its core ideas around intelligent frame selection.

3.6 Conclusion

We have proposed a method for frame selection that we refer to as SMART frame selection. The method addresses the issue of considering all frames in a video at once, instead of individually, therefore making decisions globally. The proposed method outperforms the accuracy of the baselines on different action classification datasets and reduces the computation cost up to 4 times. Further, it outperforms recent frame selection approaches on untrimmed videos. Also, it can be extended as a pre-processing step to obtain state-of-the-art accuracy on 2 benchmarks.

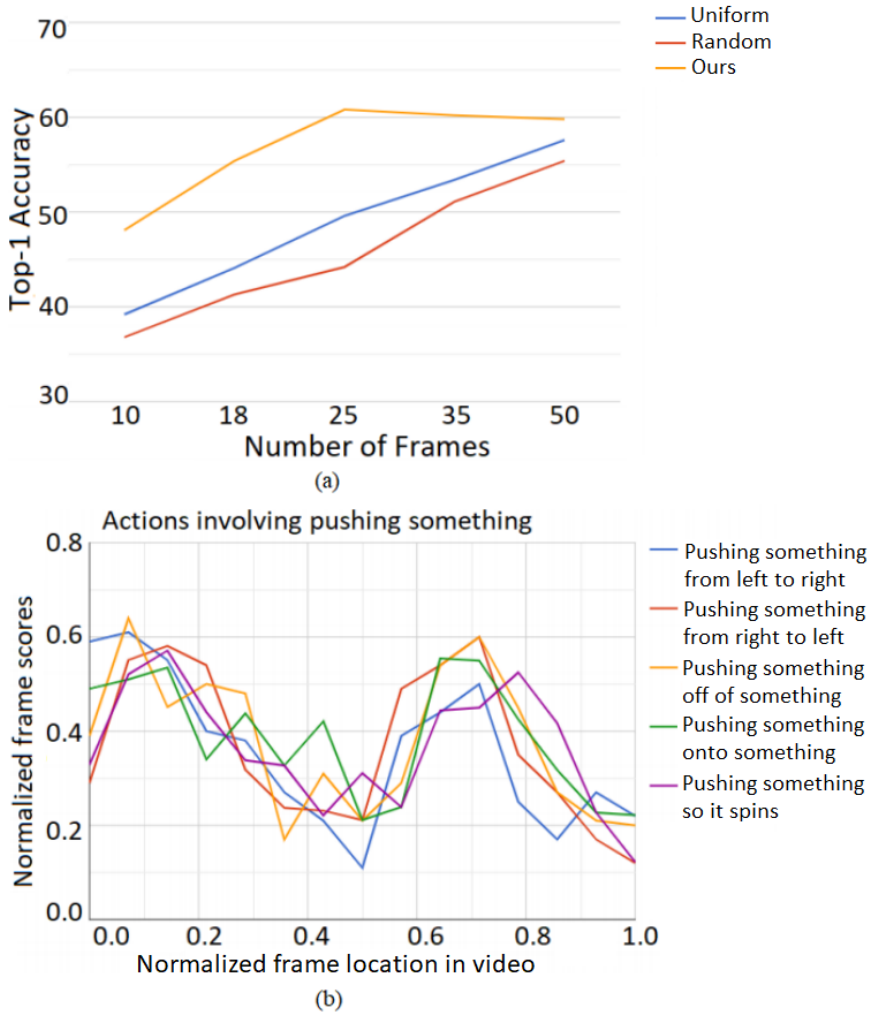


Figure 3.5: (a) Behavior of different sampling strategies with respect to number of frames. Orange represents SMART, blue represents uniform selection and red represents random selection (b) Comparison of the importance score of semantically similar actions. We can see a striking resemblance for all actions involving pushing.

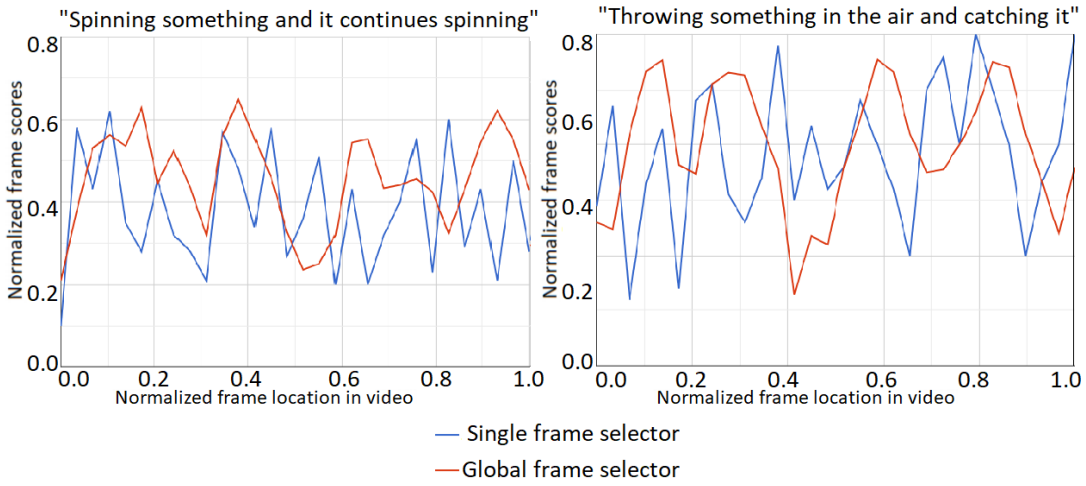


Figure 3.6: Graphical comparison of how the two selection modules give importance scores. We can see that each selector is giving different importance weights to different parts of the video. The classes are (a) Spinning something and it continues spinning and (b) Throwing something in the air and catching it

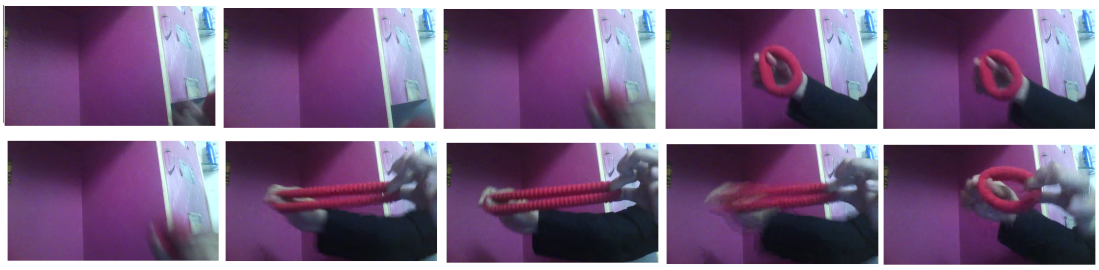


Figure 3.7: Examples of frames not selected (top) and selected (bottom) for the class “pulling something so that it gets stretched". Frames from [67].

Chapter 4

Time and Memory Efficiency of Transformer Architectures by Appropriate Initialization

4.1 Introduction

Recently, transformers have revolutionized computer vision by offering an alternative to traditional CNNs for various tasks. The self-attention mechanism in transformers enables effective modeling of long-range dependencies, leading to the development of many new state-of-the-art architectures [36, 18, 112]. Moreover, the flexibility of transformers have inspired researchers to adapt these models to more complex problems, including video understanding and action recognition [5, 113].

Transformers, however, are notoriously expensive, and Video Transformer-based architectures [10, 5, 113], which integrate information across space and time are even more so. And memory consumption and training times become even more significant when working with large-scale video datasets with long sequences [20].

Video transformers integrate spatial and temporal transformers, enabling them to efficiently capture complex spatiotemporal patterns in action recognition tasks. This combination, however, exacerbates the computational expense due to the increased processing requirements for visual information within individual frames and temporal dependencies between consecutive frames. The demands on memory consumption and training times become even more significant when working with large-scale video datasets with long sequences [20].

Video Transformer-based architectures [10, 5, 113], have made significant advancements in video action recognition. The self-attention mechanism in these architectures, which exhibits quadratic complexity with respect to the input sequence length, contributes to the substantial GPU memory consumption and extended training times [87]. These high computational costs present a particular challenge for researchers with limited resources, especially those from universities and smaller companies. The goal of our work is therefore to cut the cost of training: we want to train transformer-based video models with fewer resources or use larger model variants and handle more frames with the same resources. As a result, it is crucial to explore more efficient and scalable alternatives that can achieve high performance without excessive computational costs.

We have chosen the ViViT model [5] as our baseline upon which to improve. Specifically, we focus on the "factorized encoder" variant of ViViT which has separate spatial and temporal transformer stages, where the spatial transformer is responsible for extracting features from individual frames, while the temporal transformer processes the temporal dynamics across frames. We choose this factorized encoder design because it is more efficient compared to, e.g., the variant of ViViT using all-to-all spatiotemporal attention, while still achieving high accuracies and has thus been adopted as the building block for recent state-of-the-art architectures on various tasks [193, 22, 175, 190, 194, 70].

To address the challenge of reducing training time and memory usage without compromising the sophistication and accuracy of the original model, our approach is based on the simple idea of freezing the spatial backbone. Freezing the spatial backbone has many advantages: by not backpropagating through this transformer, training is faster and requires less memory (allowing for the model to handle more frames). We also inherit the benefits of pretraining the spatial transformer on a large dataset (such as JFT [75]). Naively implemented, however, we show that this approach falls very short in accuracy. Instead, with a few simple (but important) tweaks to the above idea, we propose a method that has the same advantages of freezing the spatial transformer, but does not compromise on accuracy.

Our method proceeds in two stages. In the first stage we pretrain a cheap version of the model using fewer frames, e.g., 8 frames as opposed to, e.g., 32 frames. Critically this includes training the temporal transformer which is often overlooked in current video models which typically initialize this component from scratch. However our experiments show that this step is critical if we wish to not sacrifice performance. In

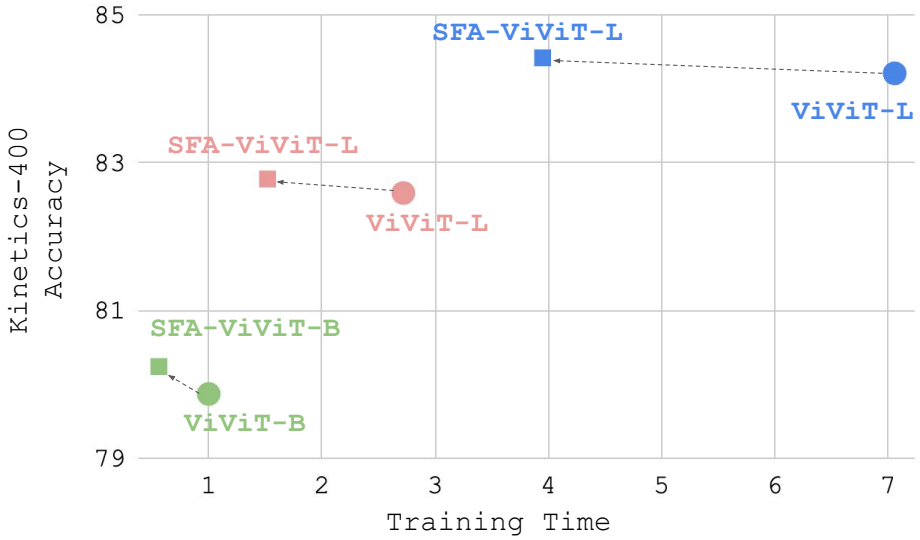


Figure 4.1: Comparison of our initialization method vs conventional training of ViViT. Training time is scaled relative to setting ViViT-B training time to '1 unit' (7.93 hours). We see clear time saving using our initialization scheme and for larger models, the training time saved is much larger.

the second stage we fine tune this model with more frames, which is more expensive but in this stage we freeze the spatial encoder and introduce a compact "adapter" model connecting frozen spatial representations to the temporal transformer, negating the need for end-to-end training of the spatial transformer. The adapter model is a lightweight neural network that connects the frozen spatial transformer outputs to the temporal transformer. Critically this includes pre-training the temporal transformer (by initializing from stage 1) which is often overlooked in current video models which typically initialize this component from scratch. However our experiments show that this step is critical if we wish to not sacrifice performance.

Drawing parallels with curriculum learning [9], our methodology can also be viewed as progressively training on tasks of increasing complexity, beginning with a ViViT model pre-trained on 8 frames — our "easy examples". As we progress, the model effectively handles larger frame counts up to 128 frames - our "difficult examples". This approach not only sustains the intricacy of the original model but also significantly reduces resource demands. Thus our approach enables entities with limited resources to emulate high-performance models using affordable GPUs.

With our training recipe, we match or slightly outperform conventional training of ViViT at roughly half the cost as seen in Figure 4.1. A notable benefit of our training recipe, is its ability to process up to 80 frames on typical university-grade GPUs, a significant leap from the previous capacity of 16 frames. This expansion in processing power broadens the range of video data manageable under resource-constrained settings. As we elaborate in Section 4.8, our research underscores the potential to democratize access to advanced video transformer models. Another notable benefit is the model's ability to now use even larger models as the spatial transformer, we introduce ViViT-g. This accessibility paves the way for future video action recognition research, irrespective of resource constraints. Hereafter, we refer to our version of ViViT as SFA-ViViT, where SFA denotes '**S**patial **F**rozen and **A**dapter Initialized'. In summary, we propose an efficient training strategy for video transformers that reduces training costs and memory usage by freezing the spatial transformer and initializing the temporal transformer.

4.2 Methodology

4.2.1 Revisiting ViViT

The Video Vision Transformer (ViViT) extends the Vision Transformer architecture to handle video data by incorporating spatio-temporal reasoning. The idea behind ViViT is to process video input as a sequence of image patches, combining spatial and temporal information through a series of transformer layers, which include multi-head self-attention, layer normalization, and feed-forward networks. The output is used for video classification.

There are broadly 4 versions of the ViViT model namely:

- Spatio-temporal attention model: This model forwards all spatio-temporal tokens extracted from the video through the transformer encoder. It models long-range interactions across the video from the first layer but due to its quadratic complexity, it necessitates the development of more efficient architectures.
- Factorised encoder model: The architecture consists of two separate transformer encoders, one modeling interactions between tokens from the same temporal index and the other modeling interactions between tokens from different temporal indices. Despite having more parameters, it requires fewer floating

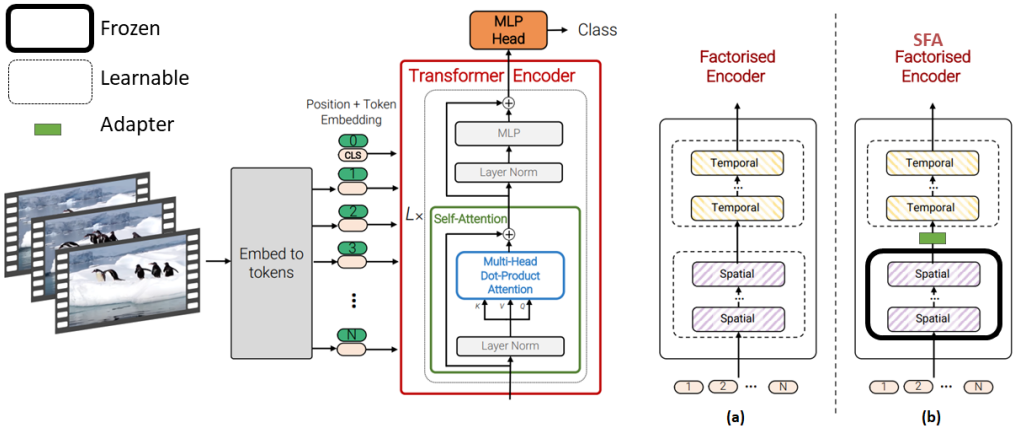


Figure 4.2: A visual comparison of (a) the factorized encoder version of the ViViT model as our baseline, and (b) our proposed model, which involves freezing the spatial transformer post-initialization, adding a small adapter model for further fine-tuning, and initializing the temporal transformer rather than training it from scratch.

point operations (FLOPs) than the first model. The paper also proposes two more variations of the factorised model, but as mentioned before the factorised encoder model has the best speed/accuracy trade-off and is used for all our experiments.

- Factorised self-attention model: This model maintains the same number of transformer layers as the first model but computes self-attention spatially and temporally more efficiently by factorizing the operation over two smaller sets of elements. This results in the same computational complexity as the second model. The order of spatial-then-temporal self-attention or vice versa does not affect the model's performance.
- Factorised dot-product attention model: This model maintains the same computational complexity as the second and third models, while retaining the same number of parameters as the first model. It achieves this by factoring the multi-head dot-product attention operation over the spatial and temporal dimensions separately. Half of the attention heads attend over tokens from the spatial dimension and the rest over the temporal dimension.

As our training strategy focuses on separating the spatial transformer and temporal transformer, we use the factorized encoder version of ViViT. The factorized encoder version of ViViT disentangles spatial and temporal encodings to better exploit their individual properties. This architecture embodies a "late fusion" approach to incorporating temporal information, with the initial spatial encoder being identical to the

one used for image classification. This is similar to CNN architectures that first extract per-frame features and subsequently aggregate them into a final representation before classification. Our proposed version of the factorised encoder is visualized in Figure 3.2.

The factorized encoder version is the most efficient variant of ViViT models due to its reduced computational complexity. By separately handling spatial and temporal information, it significantly lowers the number of floating-point operations (FLOPs) needed for processing, resulting in a more efficient model that maintains high performance in video understanding tasks. Because the Factorised Encoder variant strikes a good balance point between accuracy and processing speed, it has also been adopted as the foundation for other architectures [193, 22, 175, 190, 194, 70], reinforcing its utility and robustness.

4.2.2 Our training strategy

We concentrate on the factorised encoder variant of ViViT as it is already the most efficient version of the baseline. Henceforth, when we talk about ViViT we refer to this variant of ViViT. Consider the ViViT model that contains a spatial transformer with parameters $\theta_{spatial}$ and a temporal transformer with parameters $\theta_{temporal}$. X_{in} refers to the input video frames that have dimensions (N, C, H, W) where N is the number of frames, C is the number of channels (3 for RGB), and H and W are the height and width of each frame. This input X_{in} passes through the spatial transformer to extract per-frame feature representations. The output of the spatial transformer, denoted $X_{spatial}$, is then passed into the temporal transformer, which outputs embeddings X_{out} with dimensions (N, D) where N is the number of frames and D is the hidden dimension of the temporal transformer.

$$\begin{aligned} X_{spatial} &= T_{spatial}(X_{in}; \theta_{spatial}) \\ X_{out} &= T_{temporal}(X_{spatial}; \theta_{temporal}). \end{aligned} \tag{4.1}$$

In conventional ViViT training, $\theta_{spatial}$ is initialized from an image pre-trained checkpoint such as ImageNet-21k [140] or JFT [75] and the $\theta_{temporal}$ is initialized from scratch. During backpropagation, the gradient flows through the entire model. This entails training two sizable transformer models end-to-end, which is a highly resource-intensive process, as the transformer architecture is inherently computationally demanding, especially with more frames and larger ViViT variants (e.g., ViViT-H).

One approach to reducing training time is to freeze the parameters of the spatial transformer $\theta_{spatial}$. By not backpropagating through $\theta_{spatial}$, gradient updates are faster and require less memory, allowing us to access more frames without encountering out-of-memory issues. But as we show in experiments, the accuracy of the resulting model with frozen $\theta_{spatial}$ is not competitive (in accuracy) with the baseline training approach.

We present a two stage approach (see Fig. 4.3) to training ViViT models that inherits the same benefits of freezing the spatial transformer, while not compromising on model quality.

Stage 1. In Stage 1, we pretrain our ViViT model on a reduced number of frames initializing the spatial transformer using a pre-trained image checkpoint. We do not freeze the spatial transformer during this stage, but critically, Stage 1 serves to also initialize the temporal transformer.

To set the number of frames at this stage, we must balance the goal of efficiency (using fewer frames) against our finding in experiments that pre-training on too few frames can lead to suboptimal results. In our ablations, we identify a sweet spot at 8 frames.

Full results of this ablation study can be seen in Section 4.3.3.

Stage 2. In Stage 2, we fine tune our ViViT model on the full frame count (e.g. 128 frames) initializing both spatial and temporal transformer parameters learned in Stage 1. Because this stage is significantly more expensive, in stage 2, we freeze the spatial transformer parameters $\theta_{spatial}$ and add a lightweight adapter module with parameters $\theta_{adapter}$ following the spatial transformer:

$$\begin{aligned} X_{spatial} &= T_{spatial}(X_{in}; \theta_{spatial}) \\ X_{adapter} &= A_{adapter}(X_{spatial}; \theta_{adapter}) \\ X_{out} &= T_{temporal}(X_{adapter}; \theta_{temporal}) \end{aligned} \tag{4.2}$$

In this setting, by backpropagating only through the temporal transformer and the lightweight adapter module (in our experiments, a two layer MLP), we effectively cut total training time by half.

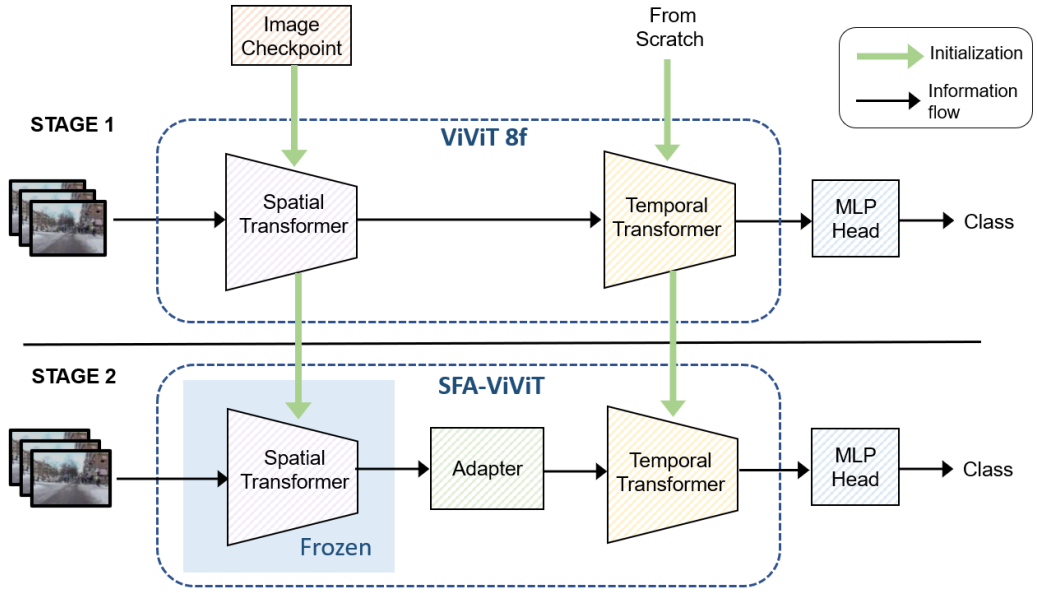


Figure 4.3: STAGE 1: We first use the full ViViT-FE model on 8 frames by initializing the spatial transformer from an image checkpoint and the temporal transformer from scratch. STAGE 2: We then use this as our checkpoint to initialize the spatial and temporal transformer for models using more frames (such as 32, 64 or 128). We then freeze the spatial transformer and add an adapter model to finetune spatial transformer features. The temporal transformer is finetuned from the same checkpoint.

The crucial finding here is that the spatial transformer requires only short-term context for initialization (after which it remains frozen), whereas the temporal transformer necessitates long-term context to achieve its optimal performance. Further details and empirical analysis can be found in the next section.

4.3 Experimental Analysis

Through a series of comprehensive experiments which we now present, we investigate the significance of the spatial transformer, examining the impact of pre-training datasets and how larger models affect action recognition performance. We also explore the importance of initializing the temporal transformer by employing various initialization schemes and datasets, assessing whether the number of frames is critical for initializing larger models, initializing full ViViT models, and initializing models on one dataset while fine-tuning on another.

4.3.1 Datasets

We evaluate on all the datasets considered in [5] (specifically, Kinetics-400 [20], Kinetics-600 [19], EPIC-Kitchens [28], Something-something v2 [66] and Moments-in-time [124]) as well as the Something-Else [117] dataset.

As Kinetics consists of YouTube videos which may be removed by their original creators, we note the exact sizes of our dataset.

Kinetics-400 [20]: Kinetics-400 is a large-scale video dataset with 400 classes introduced by Google's DeepMind. It has 235693 training samples and 53744 validation and test samples. The dataset encompasses various categories, such as object manipulation, human-object interaction, and body movements. Each class contains approximately 400 video samples, with each video lasting around 10 seconds.

Kinetics-600 [19]: Kinetics-600 is an extension of the Kinetics-400 dataset, with an increased number of classes, totaling 600 human action classes. This dataset contains approximately 380735 training samples and 56192 validation and test samples. The additional classes broaden the scope of the dataset, thereby providing more diverse training data for video recognition tasks.

EPIC Kitchens [28]: EPIC Kitchens is a large-scale dataset focusing on egocentric (first-person) videos of daily kitchen activities. It consists of 55 hours of video captured by 32 different participants in their own kitchens, with 67217 training samples and 22758 samples for validation and testing. The dataset includes 97 verb classes and 300 noun classes. Epic Kitchens is particularly useful for understanding human-object interactions and fine-grained actions in everyday settings.

Something-something v2 [66]: The Something-something v2 dataset is a collection of short video clips focused on common objects and human actions. It contains around 168913 training clips and 24777 test clips distributed across 174 action classes. This dataset aims to capture more abstract and high-level understanding of actions, as well as temporal relationships among objects.

Moments in Time [124]: The Moments in Time dataset is a large-scale video dataset containing one million short video clips, each lasting three seconds. It covers 339 classes of dynamic events and aims to provide a diverse set of visual and auditory representations of these events with 791297 training samples and 33900 test samples. This dataset is particularly useful for understanding the temporal aspects of various activities and events, as well as their associated contexts.

Something-else [117]: Something-Else utilizes the videos from SomethingSomething-V2 as its foundation, and introduces novel training and testing partitions for two new tasks that examine the ability to generalize: compositional action recognition and few-shot action recognition. Our attention is solely on the compositional action recognition task, which aims to prevent any object category overlap between the 54919 training videos and the 57876 validation videos.

4.3.2 Implementation Details

We use Scenic [30] for our implementation. Since we build on ViViT, we directly work on top of the codebase and stick to the default parameters used by ViViT in terms of hyperparameters. Full details can be seen in Table 4.1.

Our adapter is a two-layer fully connected network that takes as input the output from the spatial transformer and the output from the adapter is passed as input to the temporal transformer.

	K400	K600	MIT	Epic-Kitchens	SSv2	Selse
Optimisation						
Optimiser	Synchronous SGD					
Momentum	0.9					
Batch size	128					
Learning rate schedule	cosine with linear warmup					
Linear warmup epochs	2.5					
Base learning rate	0.1	0.1	0.25	0.5	0.5	0.5
Epochs	30	30	10	50	35	35
Data augmentation						
Random crop probability	1.0					
Random flip probability	0.5					
Scale jitter probability	1.0					
Maximum scale	1.33					
Minimum scale	0.9					
Colour jitter probability	0.8	0.8	0.8	-	-	-
Rand augment number of layers [27]	-	-	-	2	2	-
Rand augment magnitude [27]	-	-	-	15	20	-
Other regularisation						
Stochastic droplayer rate, p_{drop} [78]	-	-	-	0.2	0.3	-
Label smoothing [164]	-	-	-	0.2	0.3	-
Mixup [207]	-	-	-	0.1	0.3	-

Table 4.1: The hyperparameters utilized in the experiments conducted for the primary research paper are detailed here. If a regularisation method is not employed, it is represented by a "–". Constant values that are present across all columns are mentioned just once. For simplicity, abbreviations have been used to denote different datasets: Kinetics 400 is represented as K400, Kinetics 600 as K600, Moments in Time as MiT, Epic Kitchens as EK, Something-Something v2 as SSv2 and Something-Else as Selse.

The hyper-parameters of the transformer models are set to the standard: the number of heads are 12/16/16/16, number of layers are 12/24/32/40, hidden sizes are 768/1024/1280/1408 and MLP dimensions are 3072/4096/5120/6144 for the base, large, huge and giant versions respectively. The 8-frame ViViT model is trained for 30 epochs. We also experiment with initializing larger models with an 8-frame model trained for 10 epochs. Details of this can be found in Section 4.4.

For our hardware, we use 64 v3 TPUs for all experiments. However, we also show results using 8 NVIDIA GeForce 2080 Ti (w/12 GB memory). This is a typical setting in a small academic lab.

Index	Spatial Frozen	Adapter	Temporal Frozen	Temporal Init	Top-1 Acc	Top-5 Acc	Train Time
-	×	×	×	×	64.45	87.48	14.17 h
I	✓	×	×	×	27.75	56.73	0.5x()
II	×	×	✓	×	25.80	53.07	0.5x()
III	✓	✓	×	×	38.77	68.93	0.53x()
IV	✓	✓	×	$VMAE$	58.54	85.83	2.51x()
V	✓	✓	×	$ViViT-8f$	63.85	87.62	0.62x()

Table 4.2: Ablation study results illustrating the impact of various modifications to the ViViT-B model, including spatial and temporal transformer freezing, adapter addition, and initialization methods, on top-1 and top-5 accuracy. Dataset is Something-something v2. The Index is to refer to a particular row in Section 4.3.3.

4.3.3 Ablation Study

We first address two critical aspects: the significance of fine-tuning the spatial transformer and the importance of initializing the temporal transformer. To do so, we conduct a series of experiments in various scenarios, which are detailed below. Our analysis focuses on the Something-something dataset, utilizing the large version of the ViViT model, referred to as ViViT-L.

Consider the ViViT model that contains a spatial transformer with parameters $\theta_{spatial}$ and a temporal transformer with parameters $\theta_{temporal}$. In conventional ViViT training, $\theta_{spatial}$ is initialized from an image pre-trained checkpoint such as JFT or Imagenet21K and the $\theta_{temporal}$ is initialized from scratch. We examine four main elements that modify the structure of conventional ViViT training and these are mentioned with indices in Table 4.2 namely: I. The freezing of the spatial transformer ($\theta_{spatial}$ is initialized and then frozen), II. The freezing of the temporal transformer ($\theta_{temporal}$ is frozen), III. The addition of an adapter (lightweight module with parameters $\theta_{adapter}$), IV. Next, we initialize the temporal transformer using VideoMAE [167], while keeping the spatial transformer frozen and the adapter incorporated and V. The initialization of the temporal transformer ($\theta_{spatial}$ and $\theta_{temporal}$ are initialized using the 8-frame version of the baseline).

It is important to note that the VideoMAE training is an extremely expensive process as can be seen in the table. But combined with the line below it, these two models, which significantly outperform lines I, II and III, show that properly initializing the temporal transformer is the critical issue at hand.

Table 4.3: Comparison of impact of different backbones for the spatial transformer. We use ResNet-ViT-L pre-trained on ImageNet21k, ViT-L pre-trained on ImageNet21k, ViT-L pre-trained on JFT and ViT-H pre-trained on JFT. Listed as (Top-1 accuracy/ Top-5 accuracy).

Backbone	16-frames	32-frames	48-frames
ResNet-ViT-L (ImageNet21k)	66.09/88.30	66.63/88.50	66.88/88.65
ViT-L (ImageNet21k)	65.59/85.86	68.34/87.80	70.09/88.91
ViT-L (JFT)	69.76/88.41	73.98/90.88	75.08/91.69
ViT-H (JFT)	73.68/90.23	75.85/91.53	77.90/92.72

Based on the findings, we conclude that the most considerable enhancement in performance is achieved through the initialization of the temporal transformer. Additionally, initializing the spatial transformer yields further improvement. The adapter plays a vital role in augmenting performance when the spatial transformer is frozen, and due to its lightweight nature, it will be an essential component of our training methodology moving forward.

4.3.4 How important is the pre-training image dataset for action recognition performance?

While we know from the original ViViT paper [5] that using larger ViT [36] backbones result in better performances, we do a more thorough ablation here by considering variations of the ViT model such as the hybrid ViT (ResNet-ViT-L pre-trained on ImageNet21k [140]), ViT-L pre-trained on ImageNet21k, ViT-L pre-trained on JFT and ViT-H pre-trained on JFT. We report these results in Table 4.3, with the obvious conclusion of larger backbones pre-trained on larger datasets yields highest accuracies. We report top-1 and top-5 accuracies on the Kinetics-400 dataset and we freeze the spatial transformer here without any fine-tuning or adapter. We also keep the temporal transformer fixed in size here for fair comparison. Essentially, the performance difference is purely from the output of the spatial transformer changing due to different backbones.

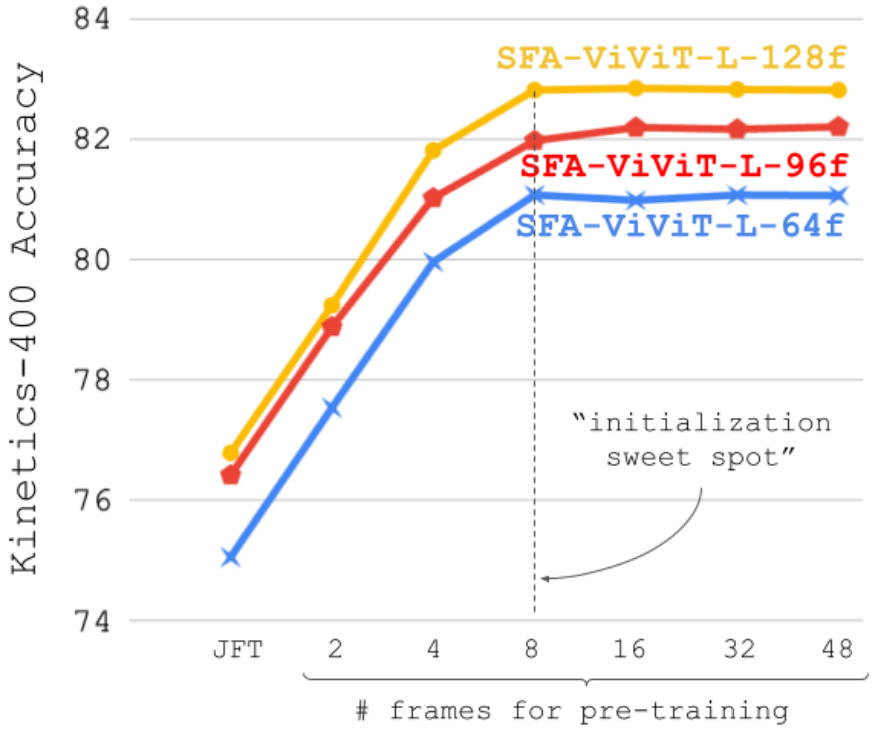


Figure 4.4: The effect of initializing with different numbers of frames (JFT, 2, 4, 8, 16, 32, and 48), freezing the spatial transformer and adding an adapter model and fine-tuning using 64, 96, and 128 frames. Results on Kinetics400 dataset, 'f' refers to frames.

4.3.5 How many frames should we use for Stage 1?

Next we experiment with various frame counts for stage 1 training, We test seven variants: JFT [75] checkpoint (image-based), 2, 4, 8, 16, 32, and 48-frame ViViT checkpoints. We then fine-tune these with a frozen spatial transformer and add an adapter model using 64, 96, and 128 frames (see Figure 4.4). Results show that using too few frames for Stage 1 training can underperform (with image-only initialization from a JFT [75] checkpoint performing the worst). Thus we deduce that short term temporal context is essential for initializing the spatial transformer. Performance also plateaus after 8 frames, and given that using more frames increases training time, we settle on using 8 frames as our "sweet spot" for Stage 1 training.

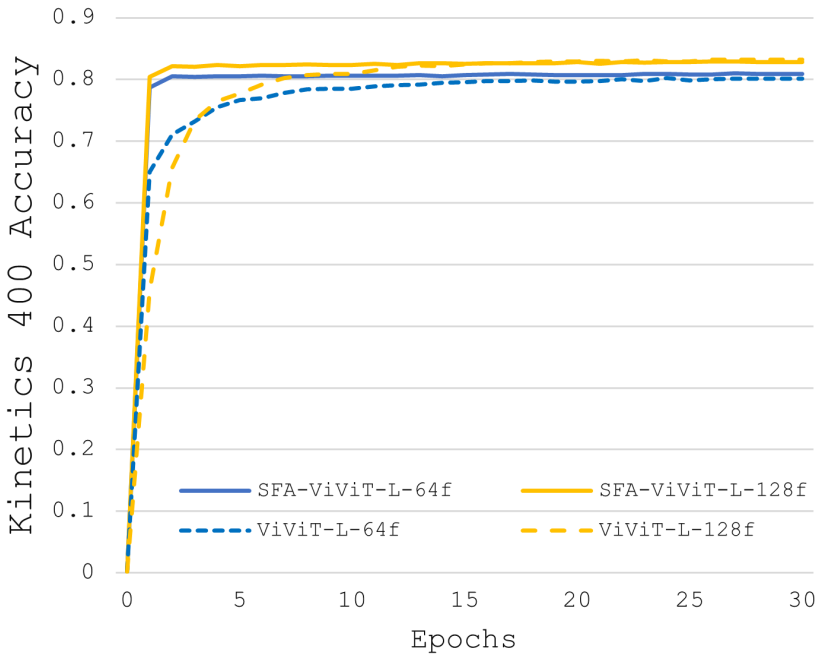


Figure 4.5: Comparison of our initialization method vs conventional training of ViViT on Top-1 accuracy and loss on the Kinetics-400 dataset using 64 and 128 frames. We see that our initialization gives a significant headstart to the models.

4.3.6 Does the proposed training increase convergence speed?

Another potential question that may arise concerns the impact of this initialization method in terms of convergence speed, if any. This specific aspect holds considerable significance owing to its potential ability to drastically curtail the duration of time required for training and the number of epochs necessary to effectively train the model. Moreover, an important element to take into account is the effect of freezing the spatial transformer. This approach decreases the memory needed to store the model but also considerably enhances the training speed. To provide a clearer picture, we have plotted the validation curves with and without initialization, which can be seen in Figure 4.5. Note that with the proposed initialization, we get a significant head start in overall accuracy.

4.3.7 Extending the image backbone to 1.5B parameters

An intriguing consequence of our approach is the ability to incorporate larger backbones into the spatial transformer, made possible by the additional memory available to us as a result of freezing the spatial transformer during training. Consequently, we introduce ViViT-g, which integrates the ViT-g model (with 1.5B parameters) as its backbone. To ensure a fair comparison, we focus solely on training and inference using 48 frames, and abstain from employing multiview or multicrop testing. Our objective is to investigate the potential impact of a more substantial spatial transformer backbone on the overall performance and show the potential of larger spatial backbones that are possible due to our training process. It is essential to note that the full ViViT-g model could not process more than 8 frames due to memory limitations. However, our proposed strategy allows processing up to 48 frames. A comparison of top-1 and top-5 accuracies is presented in Table 4.3.9 along with the number of steps needed to reach the best performance. Dataset used is Kinetics-400 and all the ViT checkpoints are JFT-pretrained [75].

4.3.8 Comparison of Memory Usage with Standard ViViT Training and Proposed Method

In this study, we compare the number of frames that can be accessed using the standard ViViT training scheme against our proposed scheme, employing a set of 64 v3 TPUs that have 16 GB each. We further evaluate the performance of ViViT variants, including H, and g, in comparison with the SFA-ViViT using the same variant configurations. Maintaining identical hyperparameters, we ensure a local batch size of 1. Our findings indicate that the conventional ViViT training approach restricts frame accessibility to 96 frames for the ViViT-H model, and a mere 8 frames for the ViViT-g model, before reaching memory limitations. Conversely, our proposed method enables access to 128 frames for ViViT-H, and up to 48 frames when utilizing ViViT-g with the same hardware. Furthermore, we investigate the impact of utilizing university grade GPUs by conducting ViViT experiments on an NVIDIA Tx 2080 Ti GPU farm equipped with 8 GPUs having 12 GB each. Under these circumstances, ViViT can only process 16 frames using a local batch size of 1. However, our proposed training strategy enables a notable improvement, expanding the frame capacity to 80 frames

Model	4f	8f	16f	32f	48f	64f	96f	128f
ViViT-H	✓	✓	✓	✓	✓	✓	✓	×
SFA-ViViT-H	✓	✓	✓	✓	✓	✓	✓	✓
ViViT-g	✓	✓	×	×	×	×	×	×
SFA-ViViT-g	✓	✓	✓	✓	✓	×	×	×

Table 4.4: Memory usage in different ViViT training schemes is compared using the Kinetics400 dataset on 64 TPUs v3 with 16GB memory each. A ✓ indicates accessible frames given hardware constraints, while a × signals an out-of-memory (OOM) error. Here, 'f' in 8f, 16f and so on corresponds to the number of frames.

helping us reproduce ViViT results on lower end GPUs. This enhancement provides a valuable opportunity for researchers with limited resources to attain performance levels comparable to those with extensive resources. We show a comparison of number of frames accessible with and without our training recipe in Table 4.4.

4.3.9 Comparison on all benchmarks to the baseline model

In this section, we present a comprehensive comparative analysis, focusing on the proposed approach and the baseline model. We report the Top-1 accuracy, Top-5 accuracy and the overall training time. The evaluation is conducted on the large and huge variants of ViViT across three datasets, namely Kinetics400, Kinetics600, and Moments in Time (MiT), with the summarized results tabulated in Table 4.5. The findings indicate a slight enhancement in accuracy for both Kinetics400 and Kinetics600 datasets, whereas a notable 1.79% increase in top-1 accuracy is observed for the MiT dataset using the proposed method. Furthermore, the proposed approach showcases a significant reduction in training time, accounting for approximately 56% of the original duration. This reduction emphasizes the advantageous nature of the proposed approach. To calculate the total training time for the SFA version, the train time of the 8 frame (Stage 1) ViViT model is combined with the train time of the (Stage 2) SFA-ViViT model. Conversely, the total training time for the standard ViViT encompasses the total train time for the same number of frames that SFA-ViViT is trained on for fair comparison.

We further examine the performance of ViViT-L incorporating our proposed training strategy in comparison to the original version on three additional datasets: Something-something, Something-Else, and Epic-Kitchens. A consistent trend is observed, with the modified approach outperforming the baseline model, at only a 62% cost of the

Model	Kinetics-400		Kinetics-600		Moments in Time		
	Accuracy	Train Time	Accuracy	Train Time	Accuracy	Train Time	
ViViT-L	82.59/93.09	1x(21.57 h)	83.29/**95.82**	1x(26.14 h)	-	-	-
ViViT-L + **SFA**	**82.78**/**94.03**	**0.56x()**	**83.47**/95.29	**0.56x()**	-	-	
ViViT-H	84.21/94.66	1x(56.71 h)	84.18/95.68	1x(60.45 h)	38.17/62.84	1x(110.79 h)	
ViViT-H + **SFA**	**84.42**/**94.72**	**0.57x()**	**84.39**/**96.20**	**0.57x()**	**39.96**/**64.39**	**0.59x()**	

Table 4.5: Performance Comparison of various versions of ViViT with the proposed training strategy for Kinetics-400, Kinetics-600 and Moments in Time. Accuracies listed as Top-1/Top-5.

Model	Something-something		Something-Else		Epic-Kitchens	
	Accuracy	Train Time	Accuracy	Train Time	Accuracy	Train Time
ViViT-L	**64.45**/87.48	1x(14.17 h)	53.14/73.98	1x(3.84 h)	43.53/56.55/**65.40**	1x(5.61 h)
ViViT-L + **SFA**	63.85/**87.62**	**0.62x()**	**53.60**/**74.47**	**0.62x()**	**43.54**/**56.78**/65.16	**0.63x()**

Table 4.6: Performance Comparison of ViViT-L with the proposed training strategy for Something-something v2, Something-Else and Epic-Kitchens. Accuracies listed as Top-1/Top-5, for Epic Kitchens Top-1 noun-verb/ Top-1 noun/ Top-1 Verb.

baseline training time. In summary, our proposed training strategy demonstrates promising potential by yielding comparable or slightly improved performance across all datasets. This is obtained while maintaining a training cost ranging from 56% to 62% of the original model, thus highlighting its effectiveness. Results can be seen in Table 4.6.

Checkpoint	SSv2	K400
K400-init	44.71/74.53	**82.81**/**93.98**
SSv2-init	**63.85**/**87.62**	76.79/92.35

Table 4.7: A summary of cross-dataset initialization of the proposed model and performance comparison. We use Kinetics400 and Something-something v2 as our datasets.

Backbone	Top-1	Top-5	Steps
ViViT-L	79.64	91.73	48k steps
ViViT-H	81.02	93.09	39k steps
ViViT-g	**81.81**	**94.55**	29k steps

Table 4.8: A comparison of top-1 and top-5 accuracies for the ViViT-g model with the proposed training strategy, which incorporates a larger spatial transformer backbone. All models use 48 frames for fair comparison. Results are on Kinetics400 dataset.

4.4 Curriculum Training

We consider variants of the “curriculum" training we talk about in the paper. There are various forms that we can consider. For instance, we can train the standard ViViT 8 frame model for just 10 epochs and use that to initialize our model. In the paper all initializations are done using 8 frame model trained for 30 epochs. Further, we could initialize smaller versions of SFA-ViViT like a 32 frame version for 10 epochs and then initialize SFA-ViViT 128 frames using this 32 frame version. We plot this in Figure 4.6 and see various versions and conclude that in the end, the best speed-accuracy trade-off was obtained when the standard ViViT 8 frame model was trained on 30 epochs and then the 128 frame model is initialized using this.

We define the models in the figure as follows:

- Model A: ViViT-L-8f for 10 epochs + SFA-ViViT-L-32f for 10 epochs + SFA-ViViT-L-128f for 10 epochs
- Model B: ViViT-L-8f for 10 epochs + SFA-ViViT-L-128f for 20 epochs
- Model C: ViViT-L-8f for 10 epochs + SFA-ViViT-L-128f for 30 epochs
- Model D: ViViT-L-8f for 30 epochs + SFA-ViViT-L-128f for 30 epochs
- Model E: ViViT-L-128f for 30 epochs

We see that training the ViViT-L-8f model for the full 30 epochs and then using that to initialize the SFA-ViViT-L-128f model gave us the best results. But we could potentially reduce the cost of training to 0.25x if we sacrifice 2% accuracy. All results are on the Kinetics400 dataset.

4.5 How long do we need to train the model?

We showed in the paper that using SFA based initialization helps us reach “near-peak” performance really quickly. We define this near-peak performance as 1 % less than the eventual best performance of the model. All results are using the test set of the datasets. Thus another natural question is: in order to save time, why not stop training the SFA version earlier? We note that although the standard ViViT model trains for ‘x’ epochs (see Table. 4.1 for exact number), it often reaches this “peak” performance much earlier and hence for fair comparison with the standard ViViT model, in the paper, we run on the same number of epochs. These results can be seen in Table 4.9.

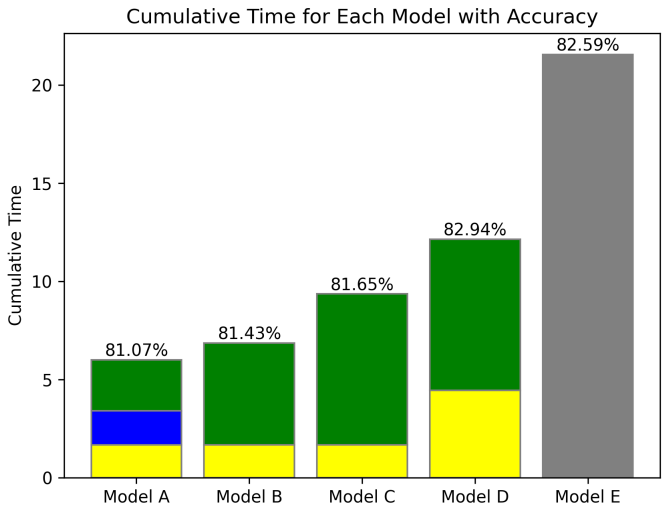


Figure 4.6: Stacked bar chart representing the cumulative processing times of Models A-E. Each color within a bar corresponds to a specific sub-model ('a' in yellow, 'b' in blue, 'c' in green, 'd' in gray) contributing to the total computation time of each model. Model accuracies are indicated at the top of each respective bar. 'a' = ViViT-L-8f, 'b' = SFA-ViViT-L-32f, 'c' = SFA-ViViT-L-128f, 'd' = ViViT-L-128f. All results are using Kinetics-400 dataset and using ViViT-L variants.

Model	Dataset	NPP Epoch	Best Epoch
ViViT-L	K400	20	29
SFA-ViViT-L	K400	5	28
ViViT-L	K600	21	28
SFA-ViViT-L	K600	5	23
ViViT-L	SSv2	29	35
SFA-ViViT-L	SSv2	4	24

Table 4.9: Comparison of near peak performance (NPP) epoch and best performance epoch for ViViT and SFA-ViViT for different datasets and models. All results are on Kinetics400 dataset.

Model	Dataset	NPP Epoch	Best Epoch
ViViT-L	K400	20	29
ViViT-L init with 8f ViViT-L	K400	4	25
ViViT-H	K400	22	27
ViViT-H init with 8f ViViT-H	K400	5	22

Table 4.10: Comparison of near peak performance (NPP) epoch and best performance epoch for initializing the full ViViT model with and without the 8f variant. We see the benefit of initialization as the “near-peak" performance is reached at a much earlier stage when initialized with the 8f variant. All results are on Kinetics400 dataset.

4.6 What about initializing standard ViViT models?

Since our method proposes an initialization scheme, we also test it on the standard ViViT models that do not have their spatial transformer frozen. In this particular scenario, we only want to check if the peak performance can be reached faster. However, it is important to note that with our proposed training scheme we also reduce the overall training time by close to half. This can be seen in Table 4.10.

4.7 Limitations

This method has some limitations. First, it still requires training the full model for a reduced number of frames (e.g. 4-8 frames) in the first stage before freezing the spatial transformer, adding modest overhead. Compared to newer work like TokenLearner [143], ST-Adapter [131], and Uniformer [99] that focus more on reducing FLOPs and parameters, SFA is complementary as it reduces training time and memory usage. These newer methods don’t directly build on SFA, as they aim to improve efficiency along different dimensions like GFLOPs rather than training costs. The core ideas of SFA in freezing the spatial transformer and using a compact adapter are still relevant today for reducing training requirements. However, limitations remain in needing to train some version of the full model initially. Future work could explore techniques to avoid training the full spatial transformer at all, or find ways to optimize the adapter training process. Overall, SFA makes important strides in efficient transformer training, but opportunities exist to address its limitations in subsequent research.

4.8 Conclusion

We have investigated the challenges posed by the substantial training time and memory consumption of video transformers, particularly focusing on the factorised encoder variant of the ViViT model as our baseline. To address these challenges, we proposed two effective strategies: utilizing a compact adapter model for fine-tuning image representations instead of end-to-end training of the spatial transformer, and initializing the temporal transformer using the baseline model trained with 8 frames. Our proposed training strategy has demonstrated the potential to significantly reduce training costs and memory consumption while maintaining, or even slightly improving, performance compared to the baseline model. Furthermore, we observed that with proper initialization, our baseline model can achieve near-peak performance within the first 10% of training epochs. The advancements made in this work have the potential to propel research in the video understanding domain by enabling access to more frames and the utilization of larger image models as the spatial transformer, all while maintaining the same memory consumption. Our findings provide valuable insights for researchers and practitioners with limited resources, paving the way for more efficient and scalable alternatives in the action recognition field. Future work may focus on further optimizing and refining these strategies, and exploring their application to other video transformer architectures and tasks in the computer vision domain.

Chapter 5

A New Split for Evaluating True Zero-Shot Action Recognition

5.1 Introduction

The substantial advancements and strides in the field of action recognition that have been achieved recently can be chiefly attributed to the rise in the accessibility and availability of substantial, extensively annotated datasets. However, the practical limitations of the task cannot be ignored, namely, the challenge and impracticality of collecting, collating, and analyzing thousands upon thousands of videos for the purpose of identifying and recognizing a singular class label.

To circumvent this practical constraint, researchers have pivoted their focus to the complex and intriguing problem of zero-shot learning (ZSL). Each class label, within this realm of ZSL, is coupled with corresponding semantic embeddings. These embeddings are either manually annotated, that is human-labeled, or are inferred through the application of semantic knowledge employing word embeddings [15, 116, 64].

These embeddings serve a crucial purpose, forming a key cornerstone of the ZSL approach. They facilitate the acquisition of relationships between the various training classes, which possess a large multitude of samples, and the testing classes, which conversely possess zero samples.

Generally speaking, in a typical application of this approach, the model is designed and configured to predict the semantic embedding of the input video. Subsequently, it matches this prediction to a test class by utilizing the nearest neighbor's search methodology. In recent times, the concept and practice of ZSL with videos have gained a substantial amount of traction, with numerous promising works being published in this field, as evidenced by several recent publications and studies [15, 116, 64].

Nevertheless, there are some potential drawbacks and limitations associated with this approach. As noted, many of the studies and works conducted in the domain of video ZSL have been observed to frequently rely on a pre-existing, pre-trained model to represent videos. In addition to this, they also tend to employ random splits for the purpose of evaluation. This combined methodology creates certain complexities and challenges, particularly when it comes to comparing the relative efficacy and performance of different approaches in a fair, unbiased manner.

The use of pre-trained models can certainly yield advantages, particularly when it comes to obtaining robust, accurate, and effective visual representations. However, these models also run the risk of overlapping with test classes, which may undermine and invalidate the very premise of zero-shot learning [15, 116, 64].

This issue, as it turns out, is not confined solely to the domain of videos. In fact, it has also been observed in the realm of image-based models, as evidenced by several studies in the image domain [177, 186]. These models are typically pre-trained on ImageNet [31].

In the context of image ZSL, Xian et al. demonstrated that if there is an overlap between the pre-training dataset and the test set, this leads to an over-inflated level of accuracy during the test phase. Such artificial inflation effectively breaks the ZSL premise, thus calling into question the validity of the approach [186]. As a potential solution to this problem, the authors proposed a new split, one that is designed to avoid this overlap problem. This newly proposed split has since been widely adopted within the field.

Following a similar vein, it is worth noting that most video models are pre-trained on Kinetics-400 [20]. Unfortunately, this dataset shares a large degree of overlap with the typical ZSL action recognition benchmarks, including UCF101, HMDB51, and Olympics. Such overlap leads to inflated accuracies, mirroring the issue identified within the image domain. Consequently, there is a discernible and pressing need for the implementation of a new split in this context, to ensure the validity and reliability of the ZSL approach.

Figure 5.1 provides a graphic illustration of these overlap issues, offering a clear visual representation of the challenges posed by this methodological complication.

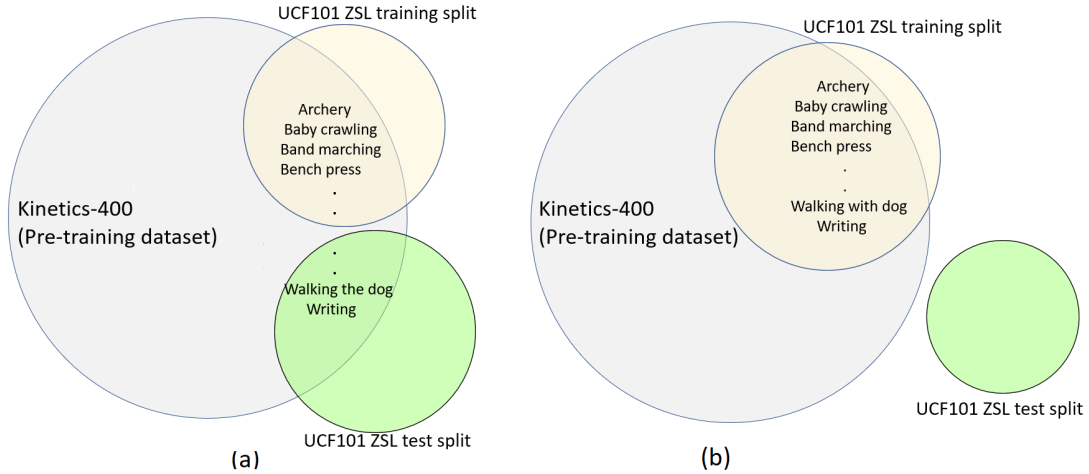


Figure 5.1: An illustration of the overlap in *classes* of the pretraining dataset (grey), training split (yellow) and zero-shot test split (green). Current evaluation protocol (a) picks classes at random with 51 training classes and 50 test classes. There is always some overlap and also chances of an extremely high overlap. We propose a stricter evaluation protocol (b) where there is no overlap at test time, maintaining the ZSL premise.

Contributions: First, we highlight a significant difference in performance that occurs when we pre-train models on classes also present in the test set, a pattern we observe across all networks and datasets. Next, we quantify the overlap between the Kinetics-400 dataset, often used for pre-training, and the common ZSL testing datasets like UCF101, HMDB51, and the Olympics datasets. We do this by computing both the visual and semantic similarities between the classes. Upon identifying this overlap, we propose a more balanced split of the classes that accounts for this overlap, ensuring we maintain the core premise of zero-shot learning (ZSL). We find that current models perform less effectively under this new split, underscoring the importance of addressing this overlap issue. We hope that our proposed split not only proves useful to the research community but also eliminates the need for arbitrary splits, paving the way for a more accurate comparison among different methodologies.

5.2 ZSL preliminaries

Consider the instance where S represents the training set comprising of seen classes. This set is formed by tuples, denoted as $(x, y, a(y))$. Within each tuple, x signifies the visual features characteristic to each sample within S. These are generally spatio-temporal features when dealing with video content. The y component corresponds to the class label found within the established set of seen class labels, represented as Y_s. The final component, $a(y)$, symbolizes the semantic embedding of the class labelled as y. These semantic embeddings are typically either manually annotated or calculated utilizing language-based embedding strategies such as word2vec [120] or sen2vec [130].

Further, let's denote U as the set comprising of unseen classes. These classes are comprised of tuples delineated as $(u, a(u))$, where u represents a class found within the set of label Y_u, while $a(u)$ embodies the associated semantic representations. It's important to note that Y_s and Y_u exhibit no overlap. This concept is encapsulated in the following expression:

$$Y_s \bigcap Y_u = \emptyset \tag{5.1}$$

Within the context of zero-shot learning (ZSL) for video classification, given an input video, the objective is to predict a class label that belongs to the unseen set of classes. This can be represented as $f_{ZSL} : X \rightarrow Y_u$. A derivative of this problem is found in the generalized zero-shot learning (GZSL) setting, where given a video, the task is to predict a class label within the combined set of the seen and unseen classes, which is expressed as $f_{GZSL} : X \rightarrow Y_s \cup Y_u$.

In instances where a pre-trained model is employed to generate visual features, the set of pre-trained classes is defined as Y_p. For the premise of ZSL to be preserved, there should be no overlap between this set and the unseen classes, which is expressed mathematically as:

$$Y_p \bigcap Y_u = \emptyset. \tag{5.2}$$

The core problem we address in this paper is that while prior work generally adheres to Eq. 5.1, recent use of pre-trained models does not adhere to Eq. 5.2. Instead, we propose the TruZe split in Section 5.4.2, which adheres to both Eq. (5.1) *and* Eq. (5.2).

5.2.1 Visual and Semantic Embeddings

In the early stages of computing **visual embeddings** or representations, techniques such as hand-crafted features were widely utilized. An example of these features is Improved Dense Trajectories (IDT) [173]. This particular method encompasses the tracking of detected interest points trajectories within a video sequence, accompanied by four descriptors to provide a comprehensive representation. However, more contemporary research has gravitated towards the application of deep features, particularly those generated from 3D convolutional networks (like I3D [20] or C3D [168]). These 3D CNNs serve a crucial role in learning the spatio-temporal representation of the video content, providing a more nuanced understanding of the subject matter. For the sake of our experiments, we have decided to utilize both these types of visual representations to cover a broad range of methodologies.

When it comes to the generation of **semantic embeddings**, prior research [110] has often relied on manual attribute annotations associated with each class. To give an illustration of this concept, for the action of 'kicking', attributes would include the motion of the leg and the twisting movement of the upper body. However, such detailed and specific attributes are not readily available across all datasets. To circumvent this issue, an alternate approach has been proposed which involves the use of word embeddings such as word2vec [120] for each class label. This effectively eradicates the necessity of manual attribute annotation, greatly simplifying the process. More recently, it was seen [64] that the use of sen2vec [130] instead of word2vec could yield superior results, given that action labels typically comprise multiple words and the act of averaging them using word2vec can potentially result in loss of contextual meaning. In light of these findings, we have chosen to implement sen2vec for our experiments.

These visual and semantic embeddings will form the basis for identifying overlapping classes between the pre-training and test sets. As we will describe in Section 5.4.2, we leverage both visual embeddings from a pre-trained I3D model and semantic embeddings from a pre-trained sen2vec model. By computing visual and semantic similarity between the classes in the pre-training Kinetics dataset and the test datasets like UCF101, we can identify classes that overlap substantially. This allows us to separate out these overlapping classes when creating our proposed TruZe split, ensuring there is no class overlap between the pre-training and test sets. In this way, the visual and semantic embeddings introduced here enable the construction of a truly zero-shot split, which we will describe shortly.

5.3 Evaluated Methods

Our evaluation process takes into consideration both early and recent state-of-the-art methodologies. From the early methods that utilize IDT features, we consider Zero-Shot Learning by bi-directional latent embedding learning (BiDiLEL), Zero-Shot Learning by single latent embedding (Latem) [184], and synthesized classifiers (SYNC) [21]. The application of features that are not learned allows us to regulate the effect of pre-training when using random splits and the proposed split (PS).

Subsequently, we also evaluate recent state-of-the-art methods such as feature generating networks (WGAN) [185], out-of-distribution detection networks (OD) [116], end-to-end learning for Zero-Shot Learning (E2E) [15], and a clustering-based approach (CLASTER) [64]. Let's delve briefly into each of these methods for a better understanding.

LatEM [184] employs a piece-wise linear compatibility function to decipher the visual-semantic embedding relationship, aided by the use of latent variables. Each latent variable encodes to accommodate the various visual properties of the input data, enabling a comprehensive understanding of the dataset. The authors project the visual embedding into the semantic embedding space to optimize the interpretability of the data.

Unlike Latem, BiDiLEL [177] chooses to project both the visual and semantic embeddings into a shared latent space rather than only into the semantic space. This strategy aims to uphold the intrinsic relationship between the visual and semantic elements.

SYNC [21] utilizes a weighted bipartite graph to learn a projection between the semantic embeddings and the classifier model space. They construct this graph using a set of "phantom" classes synthesized to ensure the alignment of the semantic embedding space and the classifier model space, thus minimizing the distortion error.

By applying a Wasserstein GAN, ClsWGAN [185] synthesizes the unseen features of classes. Additionally, it integrates extra losses in the form of cosine and cycle-consistency losses to enhance the feature generation process.

OD [116] trains an out-of-distribution detector to distinguish the generated features from the features of the seen classes, which subsequently assists with classification in the generalized zero-shot learning setting.

As a recent technique, E2E [15] harnesses end-to-end training to mitigate the issue of overlapping classes. The method involves eliminating all overlapping classes in the pre-training dataset and subsequently employing a CNN, trained on the remaining classes, to generate the visual features for the Zero-Shot Learning videos.

CLASTER [64] involves the clustering of visual-semantic embeddings, the optimization of which is facilitated by reinforcement learning.

5.4 Evaluation protocol

5.4.1 Datasets

The three most popular benchmarks for ZSL in videos are UCF101 [157], HMDB51 [90] and Olympics [127]. The typical evaluation protocol in video ZSL is to use a 50-50 split of each dataset, where 50 % of the labels are used as the training set and 50 % as the test set. In order to provide comparisons to prior work [177, 184, 21, 116, 15, 64] and for the purpose of communicating replicable research results, we study UCF101, HMDB51, and Olympics, as well as the the relationship to the pre-training dataset Kinetics-400.

In our experiments (see Section 5.5), we find overlapping classes between Kinetics-400 and each of the ZSL datasets, and move them to the training split. Thus, instead of using 50-50, we need to use 70-31 (number of labels for train and test) for UCF101 and 29-22 (number of labels for train and test) for HMDB51. We see that the number of overlapping classes in the case of Olympics is 13 out of 16, and hence we choose not to proceed further with it. More details can be found in Table 5.1. For a fair comparison between the TruZe and random split, we use the same proportions (i.e., 70-31 in UCF101 and so on) in the experiments with random splits. We create ten such random splits and use these same splits for all models.

5.4.2 TruZe Split

We now describe the process of creating the proposed TruZe split, to avoid the coincidental influence of pre-training on ZSL. First, we identify overlapping classes between the pre-training Kinetics-400 dataset and each ZSL dataset. To do this, we compute visual and semantic similarities, and discard those classes that are too similar.

Dataset	Videos	Classes	Random Split* (Seen/Unseen)	Overlapping classes with Kinetics	TruZe Split (Seen/Unseen)
Olympics	783	16	8/8	13	-
HMDB51	6766	51	26/25	29	29/22
UCF101	13320	101	51/50	70	70/31

Table 5.1: Datasets and their splits used for ZSL in action recognition. Traditionally, 'Random Split' was followed where the seen and unseen classes were randomly selected. However, we can see the extent of overlap in the 'overlapping classes' column. Using the extent of overlap we define our 'TruZe split'. *Note that for the all experiments in this paper we use a random split which matches the number of classes of our TruZe, e.g. 29/22 for HMDB51.

To calculate visual similarity, we use an I3D model pre-trained on Kinetics-400 and evaluate all video samples in UCF101, HMDB51 and Olympics using the Kinetics labels. This helps us to detect similarities that are often not recognized in terms of semantic similarities. Some examples include typing (class in UCF101) that the model detects as using computer (class in Kinetics), applying eye makeup (class in UCF101) that the model detects as filling eyebrows (class in Kinetics).

To calculate semantic similarity, we use a sen2vec model pre-trained on Wikipedia that helps us compare action phrases and outputs a similarity score. We combine the visual and semantic similarity to obtain a list of extremely similar classes to the ones present in Kinetics. The classes that even have a slight overlap or are a subset of a class in Kinetics are all chosen as part of the seen set (for example, cricket bowling and cricket shot in UCF101 are part of the seen set due to the superclass playing cricket in Kinetics). A few examples of the selection of classes is show in Figure 5.2.

We discard classes from the test set based on the following rules:

- Discard exact matches. For example, archery in UCF101 is also present in Kinetics.
- Discard matches that can be either superset or subset. For example, UCF101 has classes such as cricket shot and cricket bowling while Kinetics has playing cricket (superset). We manually do this based on the output of the closest semantic match.
- Discard matches that predict the same visual and semantic match. For example, *apply eye makeup* (UCF101 label) predicts *filling eyebrows* as the visual match using Kinetics labels and the closest semantic match to classes in Kinetics is also *filling eyebrows*. We also manually confirm this.

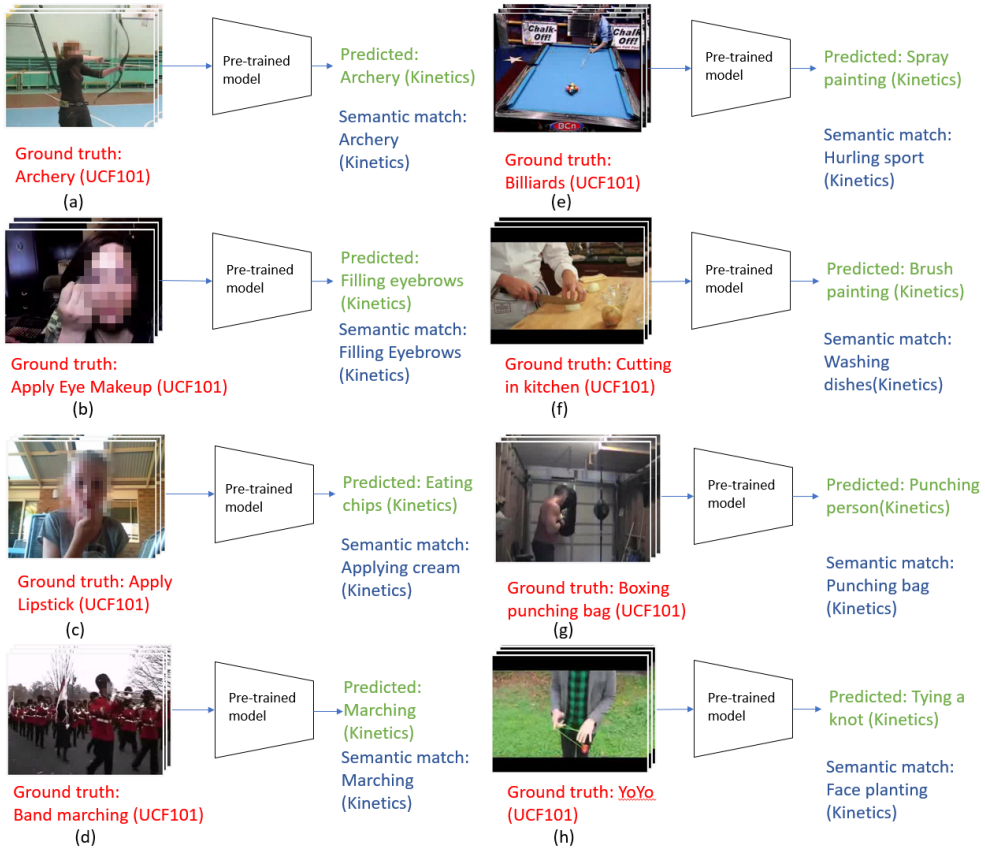


Figure 5.2: A few examples of how the classes are selected. (a) is an example of an exact match between the testing dataset (in this case UCF101) and the pre-trained dataset (Kinetics). (b), (d) and (g) are examples of visual-semantic similar matches where the output and semantically closest classes are the same. (c), (e), (f) and (h) are examples of classes without overlap in terms of both visual and semantic similarity.

We move all the discarded classes to the training set. This leaves a 70-31 split on UCF101 and a 29-22 split on HMDB51. We also see that in the Olympics dataset, there are 13 directly overlapping classes out of 16 classes and hence dropped the dataset from further analysis. One particular interesting scenario is the “pizza tossing" class in UCF101. In Kinetics, there is a class called “making pizza", however, the action of tossing is not performed in them and hence we use “pizza tossing" as an unseen class.

Method	UCF101			HMDB51		
	Random	TruZe	Diff	Random	TruZe	Diff
Latem [184]	21.4	15.5	5.9	17.8	9.4	8.4
SYNC [21]	22.1	15.3	6.8	18.1	11.6	6.5
BiDiLEL [177]	21.3	15.7	5.6	18.4	10.5	7.9
OD [116]	28.4	22.9	5.5	30.6	21.7	8.9
E2E [15]	46.6	45.5	1.1	33.2	31.5	1.7
CLASTER [64]	47.1	45.2	1.9	36.6	33.2	3.4

Table 5.2: Results with different splits for **Zero-Shot Learning (ZSL)**. Column 'Random' corresponds to the accuracy using splits in the traditional fashion (random selection of train and test classes, but with the same number of classes in train/test as in TruZe), 'TruZe' corresponds to the accuracy using our proposed split and 'Diff' corresponds to the difference in accuracy between using random splits and our proposed split. We run 10 independent runs for different random splits and report the average accuracy. We see positive differences in the 'Diff' column which we believe is due to the overlapping classes in Kinetics.

5.5 Experimental Results

5.5.1 Results on ZSL and Generalized ZSL

We first consider the results on ZSL. Here, as explained before, only samples from the unseen class are passed as input to the model at test time. Since TruZe separates the overlapping classes from the pre-training dataset, we expect a lower accuracy on this split compared to the traditionally used random splits. We compare BiDiLEL [177], Latem [184], SYNC [21], OD [116], E2E [15] and CLASTER [64] and report the results in Table 5.2. As expected, we see in the 'Diff' column for both UCF101 and HMDB51 a positive difference, indicating that the accuracy is lower for the TruZe split.

Generalized ZSL (GZSL) looks at a more realistic scenario, wherein the samples at test time belong to both seen and unseen classes. The reported accuracy is then the harmonic mean of the seen and unseen class accuracies. Since we separate out the overlapping classes, we expect to see an increase in the seen class accuracy and a decrease in the unseen class accuracy. We report GZSL results on OD, WGAN and CLASTER in Table 5.3. The semantic embedding used for all models is sen2vec. We use 70 classes for training chosen at random along with 31 test classes (also chosen at random) for UCF101 and 29 training with 22 testing for HMDB51. As expected,

Method	Acc_U			Acc_S			Harmonic mean			Dataset
	Rand	TruZe	Diff	Rand	TruZe	Diff	Rand	TruZe	Diff	
WGAN [185]	27.9	21.3	6.6	58.2	63.2	-5.0	37.7	31.8	5.9	HMDB51
WGAN [185]	28.2	23.9	4.3	74.9	75.6	-0.7	41.0	36.3	4.7	UCF101
OD [116]	34.1	24.7	9.4	58.5	62.8	-4.3	43.1	35.5	7.6	HMDB51
OD [116]	32.6	29.1	3.5	76.1	78.4	-2.3	45.6	42.4	3.2	UCF101
CLASTER [64]	41.8	38.4	3.4	52.3	53.1	-0.8	46.4	44.5	1.9	HMDB51
CLASTER [64]	37.5	35.6	1.9	68.8	70.6	-1.8	48.5	47.3	1.2	UCF101

Table 5.3: Results with different splits for **Generalized Zero-Shot Learning (GZSL)**. 'Rand' corresponds to the splits using random classes over 10 independent runs, 'TruZe' corresponds to the proposed split. Acc_U and Acc_S correspond to unseen class accuracy and seen class accuracy respectively. The semantic embedding used is sen2vec. 'diff' corresponds to the difference between 'Rand' and 'TruZe'. We see consistent positive difference in performance on the unseen classes and negative difference in the performance of the seen classes while using the 'TruZe'.

Method	SS					TruZe					Diff				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
C3D-PN [155]	57.1	66.4	71.7	75.5	78.2	50.9	61.9	67.5	72.9	75.4	6.2	4.5	4.2	2.6	2.8
ARN [209]	66.3	73.1	77.9	80.4	83.1	61.2	70.7	75.2	78.8	80.2	5.1	2.4	2.7	1.6	2.9
TRX [133]	77.5	88.8	92.8	94.7	96.1	75.2	88.1	91.5	93.1	93.5	2.5	0.7	1.3	1.6	2.6

Table 5.4: Few Shot Learning (FSL) with different splits on UCF101. Accuracies are reported for 5-way, 1, 2, 3, 4, 5-shot classification. 'SS' corresponds to the split used in [209, 133] and 'TruZe' corresponds to the proposed split. We can see that using our proposed split results in a drop in performance of up to 6.2 % for UCF101. This shows TruZe is much harder even in the FSL scenario.

the average unseen class accuracy drops in the proposed split and the average seen class accuracy increases. We expect this as the unseen classes are more disjoint in the proposed split than using random splits. For easier understanding, we convert the differences in Table 5.3 to a graph and this can be seen in Figure 5.3.

5.5.2 Extension to Few-shot Learning

Few-shot learning (FSL) is another scenario we consider. Since the premise is the same as ZSL, except that we have a few samples instead of zero. Again, usually, the splits used are random, and as such, the pre-trained model has seen hundreds of samples of classes that are supposed to belong to the test set. We report results on the 5-way, 1,2,3,4,5-shot case for temporal relational cross-transformers (TRX) [133], action relation network (ARN) [209], and C3D prototypical net (C3D-PN) [155]. Results are reported in Table 5.4 and Table 5.5. The standard split (SS) used here is

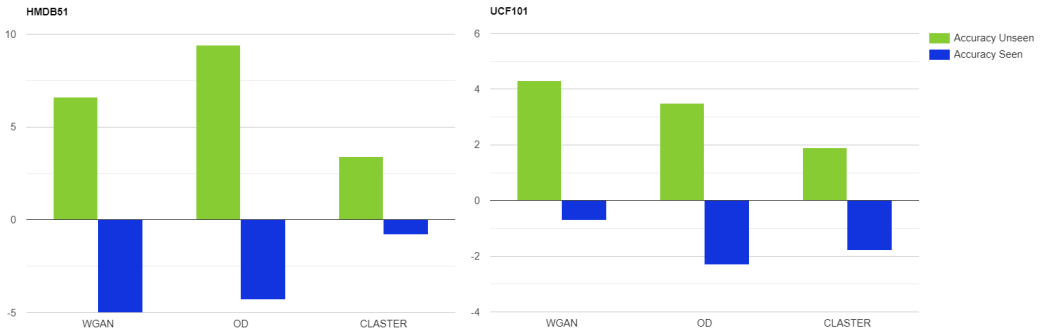


Figure 5.3: Graphical representation of the difference in performances of different models on GZSL. We see consistent positive difference in performance on the unseen classes and negative difference in the performance of the seen classes while using the 'TruZe'. The x-axis corresponds to difference in accuracy (Random splits accuracy - TruZe split accuracy) and the y-axis to different methods.

Method	SS					TruZe					Diff				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
C3D-PN [155]	38.1	47.5	50.3	55.6	57.4	28.8	38.5	43.4	46.7	49.1	9.3	9.0	6.9	8.9	8.3
ARN [209]	45.5	50.1	54.2	58.7	60.6	31.9	42.3	46.5	49.8	53.2	12.6	7.8	7.7	8.9	7.4
TRX [133]	50.5	62.7	66.9	73.5	75.6	33.5	46.7	49.8	57.9	61.5	17.0	16.0	17.1	15.6	14.1

Table 5.5: Few Shot Learning (FSL) with different splits on HMDB51. Accuracies are reported for 5-way, 1, 2, 3, 4, 5-shot classification. 'SS' corresponds to the split used in [209, 133] and 'TruZe' corresponds to the proposed split. We can see that using our proposed split results in a drop in performance of up to 17.1 % for HMDB51. This shows TruZe is much harder even in the FSL scenario.

taken from the one proposed in ARN [209]. Similar to the SS, we divide the classes in UCF101 and HMDB51 to (70,10,21) and (31,10,10), respectively, where the order corresponds to the number of training classes, validation classes and test classes. We see that the proposed split is much harder than SS. Consistent drops in performance can be seen on every split and for every model. Performance drops of upto 6.2% on UCF101 and 17.1% on HMDB51 can be seen.

5.5.3 Is overlap the reason for performance difference between Random and our TruZe split?

In order to understand the difference in model performance due to the overlapping classes, we compare the performance of each model for the random split (with five runs) vs the proposed split by using visual features represented by IDT and I3D. We depict the difference in performance in the form of a bar graph for better visual understanding. This is seen in Figure 5.4. The higher the difference, the bigger the

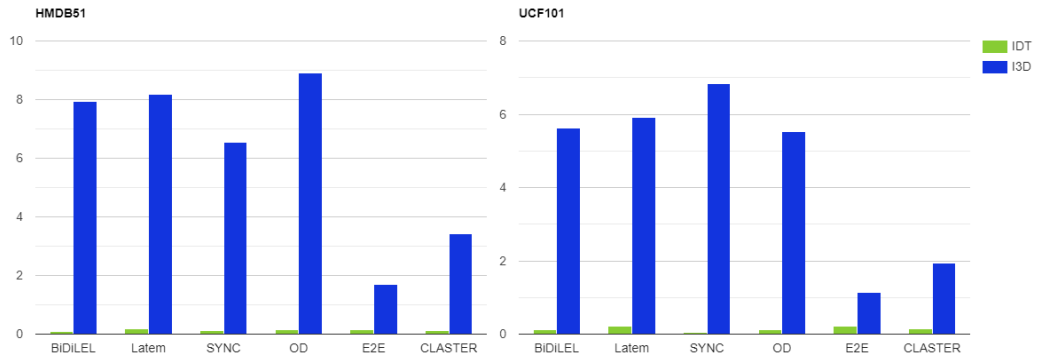


Figure 5.4: The difference of accuracy for different models using IDT and I3D using manual annotations as the semantic embedding. The larger the bar, the more significant the difference. We can see a clear difference when using I3D and this difference is due to the presence of overlapping classes in the test set. The y-axis corresponds to the difference in performance in percentage and the x-axis corresponds to various models.

impact of performance. We can see that there is a big difference when using I3D features compared to using IDT features (where there is a minimal difference). Since IDT features are independent of any pre-training model, the difference in performance is negligible. The difference while using I3D features can be attributed to the presence of overlapping classes in the random splits compared to the proposed split.

5.5.4 Use of different backbone networks

An end-to-end approach was proposed in [15] where a 3D CNN was trained in an end-to-end manner on classes in Kinetics not semantically similar to UCF101 and HMDB51 to overcome the overlapping classes conundrum. While this approach is useful, training more complex models end-to-end is not feasible for everyone due to the high computational cost involved. We show that using more recent state-of-the-art approaches as the backbone, there is a slight improvement in model performance and hence believe that having a proposed split instead of training end-to-end would be more easily affordable for the general public. Table 5.6 shows the results of using different backbones for extracting visual features on some of the recent state-of-the-art ZSL approaches. We use Non-Local networks [178] that build on I3D by adding long-term spatio-temporal dependencies in the form of non-local connections (referred as NL-I3D in Table 5.6). We also use slow-fast networks [43] that is a recent state-of-the-art approach that uses two pathways, a slow and a fast, to capture motion

Method	Backbone	UCF101 Accuracy	HMDB51 Accuracy
WGAN [185]	I3D	22.5	21.1
WGAN [185]	NL-I3D	22.7	21.3
WGAN [185]	SlowFast	**23.1**	**21.5**
OD [116]	I3D	22.9	21.7
OD [116]	NL-I3D	23.2	22.0
OD [116]	SlowFast	**23.4**	**22.5**
CLASTER [64]	I3D	45.2	33.2
CLASTER [64]	NL-I3D	45.3	33.6
CLASTER [64]	SlowFast	**45.5**	**33.9**

Table 5.6: Results comparison using different backbones to extract visual features for the ZSL models. We evaluate OD, E2E and CLASTER using I3D, NL-I3D and SlowFast networks as backbones. All results are on the proposed split. We see that stronger backbones result in improved performance of the ZSL model.

and fine temporal information. We can see minor but consistent improvements using stronger backbones, and this suggests that having a proposed split is an economical way of maximising the use of state-of-the-art models as backbone networks. We see gains of up to 0.6% in UCF101 and 0.8% in HMDB51.

5.6 Implementation Details

5.6.1 Visual features

We use either IDT [173] or I3D [20] for the visual features. Using the fisher vector obtained from a 256 component Gaussian mixture model, we generate visual feature representations using IDT (contains four different descriptors). To reduce this, PCA is used to obtain a 3000-dimensional vector for each descriptor. Concatenating these (all four descriptors), we obtain a 12000-dimensional vector for each video. In the case of I3D features, we use RGB and flow features taken from the *mixed 5c* layer from a pre-trained I3D (pre-trained on Kinetics-400). The output of the flow network is averaged across the temporal dimension and pooled by four in the spatial dimension, and then flattened to a vector of size 4096. We then concatenate the two.

5.6.2 Semantic embedding

While manual annotations are available for UCF101 in the form of a vector of size 40, there is no such annotation available for HMDB51. Hence, we use sen2vec embeddings of the action classes where the sen2vec model is pre-trained on Wikipedia. While most approaches use word2vec and average embeddings for each word in the label, we use sen2vec which obtains an embedding for the entire label.

5.6.3 Hyperparameters for evaluated methods

We use the optimal parameters reported in BiDiLEL [177]. The values for α and $k_G$values are set to 10 and d_y is set to 150. SYNC [21] has a parameter σ that models correlation between a real class and a phantom class and this is set to 1, while the balance coefficient is set to 2^{-10}. For Latem [184] the learning rate, number of epochs, and number of embeddings are 0.1, 200 and 10 respectively. For OD [116], WGAN [185], E2E [15] and CLASTER [64] we follow the settings provided by the authors. For few-shot learning, we use the hyperparameters defined in the papers [133, 209]. We compare against the standard split proposed in [209]. For the proposed split, we change the classes slightly for fair comparison to the standard split. Now the splits for HMDB51 and UCF101 are (31,10,10) and (70,10,21) where the order corresponds to (train,val,test).

5.7 Limitations

While TruZe provides a strict split to avoid overlapping classes between pre-training and test sets, it does have some limitations. One is that several recent approaches use Kinetics-600/700 [15, 126, 136] training which includes a majority of the classes from the “TruZe” test set. An alternative approach proposed in some recent work is to pre-train the model from scratch on Kinetics while excluding the overlapping classes [15]. However, this requires expensive end-to-end pre-training and limits model selection. Other recent methods have built on TruZe by incorporating its proposed split into their evaluations [64, 38, 63]. However, some recent approaches are orthogonal and do not directly address the overlapping class issue, instead focusing on new model architectures or training schemes [136, 126] An open question remains whether directly

accounting for overlapping classes is necessary as models become more powerful. Still, TruZe provides an important strict benchmark for evaluating adherence to the zero-shot premise. Recent methods have generally found it beneficial to incorporate the TruZe split, even as they explore other directions like new models.

5.8 Discussion and Conclusion

As we see in Figure 5.4 using IDT features which do not require pre-training on Kinetics resulted in a negligible change in performance comparing the TruZe split vs the random splits. However, using I3D features saw a stark difference due to the overlapping classes in the pre-trained dataset.

We see that the proposed split is harder in all scenarios (ZSL, GZSL, and FSL) whilst maintaining the premise of the problem. The differences are significant in most cases: between 0.7-6.2 % for UCF101 and 7.4-17.1 % for HMDB51 in FSL, an increase of 1.2-4.7 % for UCF101 and 1.9-7.6 % for HMDB51 (with respect to the harmonic mean of seen and unseen classes) in GZSL and an increase of 1.1-6.8 % for UCF101 and 1.7-8.1 % for HMDB51 in ZSL. It is also important to note that different methods are differently affected, suggesting that some method in the past have claimed improvements due to not adhering to the zero-shot premise, which is highly concerning.

We also see that changing the backbone network increases the performance slightly for each model, and as a result, the end-to-end pre-training [15] can prove very expensive. As such, having a proposed split makes things easier as we can directly use pre-trained models off the shelf. We see gains of up to 1.3% in UCF101 and 1.1% in HMDB51.

Chapter 6

Regularizing Representations through Clustering for Zero-Shot Action Recognition

6.1 Introduction

Research on action recognition in videos has made rapid progress in the last years, with models becoming more accurate and even some datasets becoming saturated. Much of this progress has depended on large scale training sets. However, it is often not practical to collect thousands of video samples for a new class. This idea has led to research in the Zero-Shot learning (ZSL) domain, where training occurs in a set of seen classes, and testing occurs in a set of unseen classes. In particular, in the case of video ZSL, each class label is typically enriched with semantic embeddings.

These embeddings are sometimes manually annotated, by providing attributes of the class, and other times computed automatically using language models of the class name or class description. This semantic information attached to each class is used to map the seen training classes to new unseen classes.

While ZSL is potentially a very useful technology, the standard pipeline poses a fundamental representation challenge. It requires learning a transfer function that discriminates unseen classes. This transfer function is defined using the semantic embeddings of each class, and therefore it is defined at the class level. This is especially difficult in the case of large intra-class variability, and at boundaries of similar classes, like “horse racing" and “horseback riding". The transfer of knowledge from a manually annotated semantic representation naturally yields promising results.

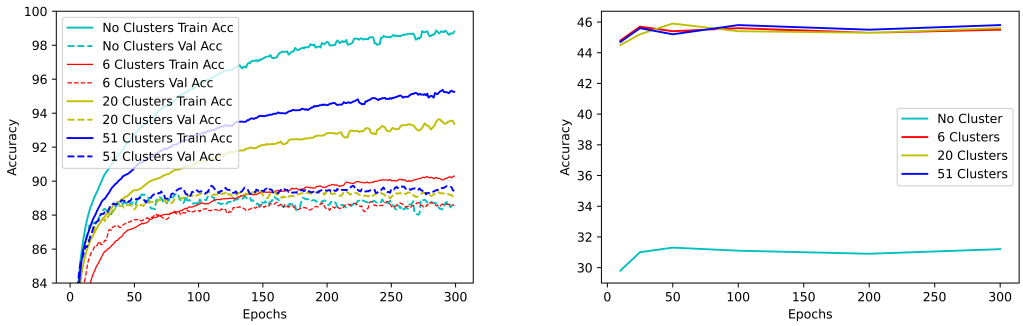


Figure 6.1: Left: Learning curve for the seen classes. Right: Learning curve for the unseen classes. The clustering-based representation avoids overfitting, which in the case of seen classes means that the gap between validation and training accuracy is smaller than in the vanilla representation. This regularization effect improves the validation accuracy in unseen classes.

Neural networks have proven extraordinarily powerful at learning complex discriminative functions of classes with many modes. In other words, instances of the same class can be very different and still be projected by the neural network to the same category. While this works well in supervised training, it can be a problem in Zero-Shot recognition, where the highly specialized discriminative function might not transfer well to instances of unseen classes. In this work, we adress this representation problem using three main ideas.

First, we turn to clustering by considering visual-semantic features of the training set as datapoints to cluster, and use the centroids of the clusters to represent a video. We argue that centroids are more robust to outliers, and thus help regularize the representation, avoiding overfitting to the space of seen classes. Figure 6.1 shows that the gap between training and validation accuracy is smaller when using clustering in seen classes (left). As a result, the learned representation is more general, which significantly improves accuracy in unseen classes (right).

Second, our representation is a combination of a visual and a semantic representation. The standard practice at training time is to use a visual representation, and learn a mapping to the semantic representation. Instead, we use both cues, which we show yields a better representation. This is not surprising, since both visual and semantic information can complement each other.

Third, we use the signal derived from the process of classification and repurpose it to serve as direct supervision for clustering, by using Reinforcement Learning (RL). Specifically, we use the REINFORCE algorithm to directly update the cluster centroids. This optimization improves the clustering significantly and leads to less noisy and more compact representations for unseen classes.

These three pieces are essential to learn a robust, generalizable representation of videos for Zero-Shot action recognition, as we show in the ablation study. Crucially, none of them have been used in the context of Zero-Shot or action recognition. They are simple, yet fundamental and we hope they will be useful to anyone in the Zero-Shot and action recognition communities.

Based on our proposed contributions, we call the method CLASTER, for *CLustering with reinforcement learning for Action recognition in zero-ShoT lEaRning*, and show that it significantly outperforms all existing methods across all standard Zero-Shot action recognition datasets and tasks.

6.2 Method Overview

We now describe the proposed CLASTER, which leverages clustering of visual and semantic features for video action recognition and optimizes the clustering with RL. Figure 6.2 shows an overview of the method.

6.2.1 Problem Definition

Let S be the training set of seen classes. S is composed of tuples $(x, y, a(y))$, where $x \in X$ represents the spatio-temporal features of a video (in our experiments, a pre-trained I3D [20] network, using both RGB and optical flow, concatenated) in the space of possible video representation X, represents the spatio-temporal features of a video, y represents the class label in the set of Y_S seen class labels, and $a(y)$ denotes the category-specific semantic representation of class y. These semantic representations are either manually annotated or computed using a language-based embedding of the category name, such as word2vec [120] or sentence2vec [130].

Let U be the set of pairs $(u, a(u))$, where u is a class in the set of unseen classes Y_U and $a(u)$ are the corresponding semantic representations. The seen classes Y_S and the unseen classes Y_U do not overlap.

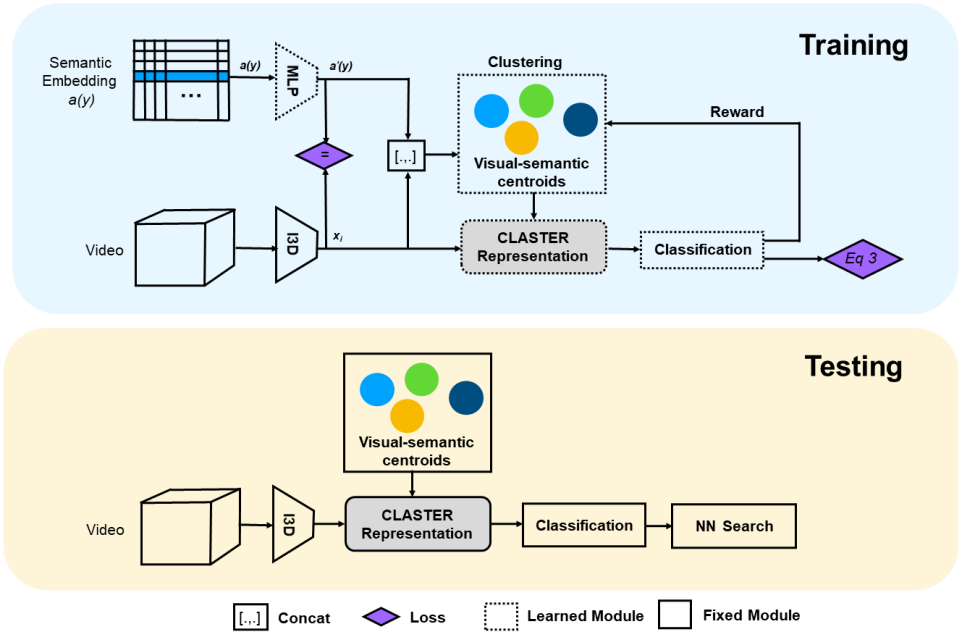


Figure 6.2: Overview of CLASTER. We map the semantic embedding $a(y_i)$ to the space of visual features x_i, and concatenate both to obtain a visual-semantic representation. We cluster these visual-semantic representations with K-means to obtain initial cluster centroids. Each video is represented as the sum of the visual-semantic representation and the centroid clusters, weighted by their distance (see Sec. 6.2.3 and Fig. 6.3). This is used as input for classification (Sec 6.2.4 and Eq. 6.3). Based on the classification result, we send a reward to optimize the cluster centroids using REINFORCE (Sec. 6.2.5). At test time, we first perform classification on the seen classes and then do a nearest neighbor (NN) search to predict the unseen class.

In the Zero-Shot Learning (ZSL) setting, given an input video the task is to predict a class label in the unseen classes, as $f_{ZSL} : X \rightarrow Y_U$. In the related generalized Zero-Shot learning (GZSL) setting, given an input video, the task is to predict a class label in the union of the seen and unseen classes, as $f_{GZSL} : X \rightarrow Y_S \cup Y_U$.

6.2.2 Visual-Semantic Representation

We obtain a representation which is not only aware of the visual features but also the semantic embedding. For this, we create a visual-semantic representation as follows. Given video i, we compute visual features x_i and a semantic embedding $a(y_i)$ of their class y_i (see Sec. 6.3 for details). The goal is to map both to the same space,

so that they have the same dimensionality and magnitude, and therefore they will have a similar weight during clustering. We learn this mapping with a simple multi-layer perceptron (MLP) [211], trained with a least-square loss. This loss minimizes the distance between x_i and the output from the MLP, which we call $a'(y)$.

Finally, we concatenate x_i and $a'(y)$ to obtain the visual-semantic representations that will be clustered. The result is a representation which is not only aware of the visual information but also the semantic. Note that we keep the MLP fixed after this initial training.

6.2.3 CLASTER Representation

We now detail how we represent videos using the proposed CLASTER representation, which leverages clustering as a form of regularization. The intuition behind this design is that representing data w.r.t. the visual-semantic centroids of the training set will lead to a representation which generalizes well to unseen classes. This would avoid overfitting to the training classes as it often happens with hidden representations. In other words, a representation w.r.t centroids is more robust to outliers, which is helpful since all instances of the unseen classes are effectively outliers w.r.t. the training distribution. Still, the initial clustering is not optimal and we discuss in the Section 6.2.5 how we further optimize our centroid-based representation with RL.

We initialize the clustering of the training set S using K-means [46]. Each resulting cluster j has a centroid c_j, that is the average of all visual-semantic samples in that particular cluster. The CLASTER representation of a given video is the sum of the visual-semantic representation and the centroids, weighted by the inverse of the distance, such that closer clusters will have more weight. Figure 6.3 shows this process in detail.

Specifically, given video i, we compute the visual representation x_i. We estimate the semantic vector ϕ_i using an MLP. This is a necessary step, as during test time we do not have any semantic information. Concatenating the visual x_i and semantic ϕ_i we obtain the intermediate representation ψ_i, which is in the same space as the cluster centroids.

We compute the Euclidean distance $d_{i,j}$ between the visual-semantic point ψ_i and each cluster j, which we refer to as $d_{i,j}$. We take the inverse $1/d_{i,j}$ and normalize them using their maximum and minimum values, such that they are between 0 and 1. We refer to these normalized values as $\eta_{i,j}$, and they are used as the weights of each

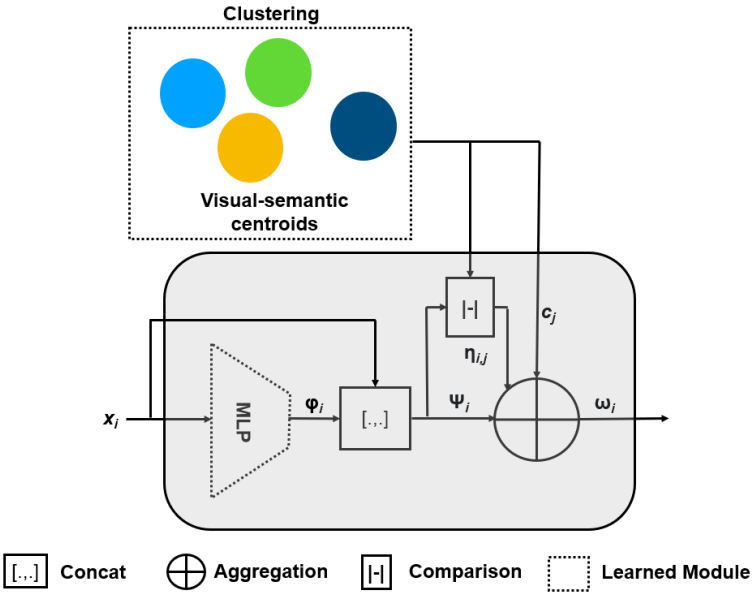


Figure 6.3: The proposed CLASTER Representation in detail (see Fig. 6.2 for the overview of the full method). The visual feature is mapped to match the space of the visual-semantic cluster centroids with an MLP and concatenation. Based on the distances to the cluster centroids the final representation ω is a weighted representation of the centroids, more robust to the out-of-distribution instances of the unseen test classes. The comparison block refers to comparing of the visual-semantic centroids and a data point to find the closest centroid. Details in Sec. 6.2.3 and Eq. 6.1.

cluster centroid in the final CLASTER representation ω_i:

$$\omega_i = \psi_i + \sum_{j=1}^{k} \eta_{i,j} c_j. \tag{6.1}$$

6.2.4 Loss Function

Given the CLASTER representation ω_i we predict a seen class using a simple MLP, V. Instead of the vanilla softmax function, we use semantic softmax [82], which includes the semantic information $a(y_i)$ and thus can transfer better to Zero-Shot classes:

$$\hat{y}_i = \frac{e^{a(y_i)^T V(\omega_i)}}{\sum_{j=1}^{S} e^{a(y_j)^T V(\omega_i)}}. \tag{6.2}$$

The output $\hat{y}_i$ is a vector with a probability distribution over the S seen classes. We train the classifier, which minimizes the cross-entropy loss with a regularization term:

$$\min_W \sum_{i=1}^{N} \mathscr{L}(x_i) \tag{6.3}$$

where W refers to all weights in the network.

6.2.5 Optimization with Reinforcement Learning

As there is no ground truth for clustering centroids, we use the classification accuracy as supervision. This makes the problem non-differentiable, and therefore we cannot use traditional gradient descent. Instead, we use RL to optimize the cluster centroids. For this, we compute two variables that will determine each centroid update: the reward, which measures whether the classification is correct and will determine the direction of the update, and the classification score, which measures how far the prediction is from the correct answer and will determine the magnitude of the update.

Given the probabilistic prediction $\hat{y}_i$ and the one-hot representation of the ground truth class y_i, we compute the classification score as the dot product of the two: $z_i = y_i \hat{y}_i$. To obtain the reward, we check if the maximum of $\hat{y}_i$ and y_i lie in the same index:

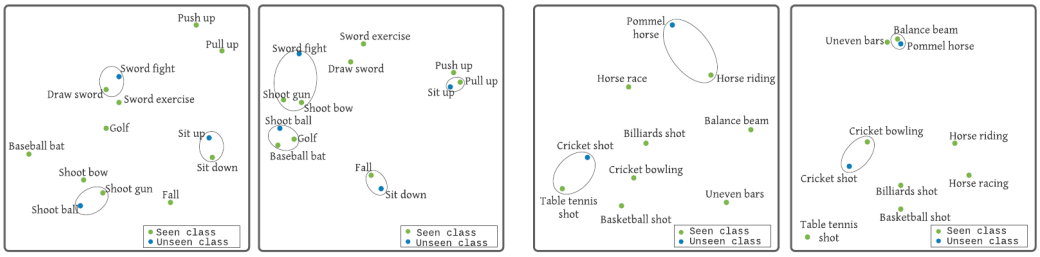
$$r = \begin{cases} 1 \; if \; \arg\max \hat{y}_i = \arg\max y_i \\ -1 \; otherwise \end{cases} \tag{6.4}$$

This essentially gives a positive reward if the model has predicted a correct classification and a negative reward if the classification was incorrect. This formulation is inspired by Likas [103], which was originally proposed for a different domain and the problem of competitive learning.

For each data point ψ_i we only update the closest cluster centroid c_j. We compute the update Δc_j using the REINFORCE [103] where $p_j = 2(1 - f(\eta_{i,j}))$ and $f(x) = \frac{1}{1+e^{-x}}$ taken from Likas [103] as:

$$\Delta c_j = \alpha \; r \; (z_i - p_j) \; (\psi_i - c_j). \tag{6.5}$$

For further details on this derivation, please see Likas [103]. The main difference between our model and Likas' is that we do not consider cluster updates to be Bernoulli units. Instead, we modify the cluster centroid with the classification score z_i, which is continuous in the range between 0 and 1.



(a) word2vec (b) sentence2vec (c) word2vec (d) sentence2vec

Figure 6.4: HMDB51 (a) Averaging word embeddings can produce poor results in certain cases. For example the nearest neighbor of “shoot ball" is “shoot gun", and of “sit up" is “sit down" which are not necessarily meaningful (b) Sentence2vec better captures phrase meanings: Nearest neighbors to “sit up" is now “push up", and for “shoot ball", is golf. **UCF101** (c) The same effect is observed, where after averaging word2vec representations, the nearest neighbor of “pommel horse" is "horse riding" (d) Sentence2vec helps capture phrase meanings: the while nearest neighbor of “pommel horse" is now “balance beam". The circles contain the nearest neighbor to the given unseen class and is for illustration purposes.

6.2.6 Use of Sentence2vec

Some datasets, such as HMDB51 [90], do not have semantic manual annotations. Thus, semantic representations are often computed using a word2vec model [120]. In most action recognition datasets, labels are phrases (e.g. “playing guitar") and thus the semantic embeddings are computed by averaging the word2vec embeddings of each word. This works in some cases, however, simple averaging does not always capture the inter-dependency of action classes. For example, “horse riding" and “horse racing" lie far apart in the word2vec space.

To alleviate this, we propose using sentence2vec [130], a model designed to capture the meaning of sentences. Specifically, sentence2vec learns embeddings with respect to the sentence context. It represents the sentence context as n-grams and optimizes the additive combination of the word vectors to obtain sentence embeddings. Figure 6.4 illustrates how class neighbors become more meaningful when we move from word2vec to sentence2vec.

We show in Sec. 6.4.3 and 6.4.4 that sentence2vec significantly improves performance of some of the recent state-of-the-art approaches, reaching performance close to using manual semantic representation. This suggests the potential of sentence2vec to automatically annotate large scale datasets. We also observe that combining multiple of these embeddings (e.g., through averaging) leads to a small but consistent improvement, suggesting they may contain complementary information.

6.3 Implementation Details

6.3.1 Visual features

We use RGB and flow features extracted from the *Mixed 5c* layer of an I3D network pre-trained on the Kinetics [20] dataset. The *Mixed 5c* output of the flow network is averaged across the temporal dimension and pooled by four in the spatial dimension and then flattened to a vector of size 4096. We then concatenate the two.

6.3.2 Network architecture

The MLP that maps the semantic features to visual features consists of two fully-connected (FC) layers and a ReLU. The MLP in the CLASTER Representation module, which maps the visual feature to the semantic space is a two-layer FC network, whose output after concatenation with the video feature has the same dimensions as the cluster representatives. The size of the FC layers is 8192 each. The final classification MLP (represented as a classification block in Figure 6.2) consists of two convolutional layers and two FC layers, where the last layer equals the number of seen classes in the dataset we are looking at. All the modules are trained with the Adam optimizer with a learning rate of 0.0001 and weight decay of 0.0005.

6.3.3 Number of clusters

Since the number of clusters is a hyperparameter, we evaluate the effect of the number of clusters on the UCF101 dataset for videos and choose 6 after the average performance stablizes as can be seen in the Section 6.8. We then use the same number for the HMDB51 and Olympics datasets.

6.3.4 RL optimization

We use 10,000 iterations and the learning rate α is fixed to 0.1 for the first 1000 iterations, 0.01 for the next 1000 iterations and then drop it to 0.001 for the remaining iterations.

6.3.5 Semantic embeddings

We experiment with three types of embeddings as semantic representations of the classes. We have human-annotated semantic representations for UCF101 and the Olympic sports dataset of sizes 40 and 115 respectively. HMDB51 does not have such annotations. Instead, we use a skip-gram model trained on the news corpus provided by Google to generate word2vec embeddings. Using action classes as input, we obtain a vector representation of 300 dimensions. Some class labels contain multiple words. In those cases, we use the average of the word2vec embeddings. We also use sentence2vec embeddings, trained on Wikipedia. These can be obtained for both single words and multi-word expressions. The elaborate descriptions are taken from [23] and only evaluated for fair comparison to them. For the elaborative descriptions, we follow ER [23] and use the provided embeddings in their codebase.

6.3.6 Rectification of the Semantic Embedding

Sometimes, in ZSL, certain data points tend to appear as nearest-neighbor of many other points in the projection space. This is referred to as the hubness problem [152]. We avoid this problem using semantic rectification [114], where the class representation is modified by averaging the output generated by the projection network, which in our case is the penultimate layer of the classification MLP. Specifically, for the unseen classes, we perform rectification by first using the MLP trained on the seen classes to project the semantic embedding to the visual space. We add the average of projected semantic embeddings from the k-nearest neighbors of the seen classes, specifically as follows:

$$\hat{a}(y_i) = a'(y_i) + \frac{1}{k} \sum_{n \in N} cos\left(a'(y_i), n\right) \cdot n, \tag{6.6}$$

where $a'(y)$ refers to the embedding after projection to the visual space, $cos(a,n)$ refers to the cosine similarity between a and n, the operator $\cdot$ refers to the dot product and N refers to the k-nearest neighbors of $a'(y_{u_i})$.

6.3.7 Nearest Neighbor Search

At test time in ZSL, given a test video, we predict a seen class and compute or retrieve its semantic representation. After rectification, we find the nearest neighbor in the set of unseen classes. In the GZSL task, class predictions may be of seen or unseen classes. Thus, we first use a bias detector [52] which helps us detect if the video belongs to the seen or unseen class. If it belongs to a seen class, we predict the class directly from our model, else we proceed as in ZSL.

6.4 Experimental Analysis

In this section, we look at the qualitative and quantitative performance of the proposed model. We first describe the experimental settings, and then show an ablation study, that explores the contribution of each component. We then compare the proposed method to the state-of-the-art in the ZSL and GZSL tasks, and give analytical insights into the advantages of CLASTER.

6.4.1 Datasets

We choose the Olympic Sports [127], HMDB-51 [90] and UCF-101 [157], so that we can compare to recent state-of-the-art models [50, 116, 137]. We follow the commonly used 50/50 splits of Xu et al. [191], where 50 percent are seen classes and 50 are unseen classes. Similar to previous approaches [222, 50, 137, 119, 88], we report average accuracy and standard deviation over 10 independent runs. We also report on the recently introduced TruZe [65]. This split accounts for the fact that some classes present on the dataset used for pre-training (Kinetics [20]) overlap with some of the unseen classes in the datasets used in the Zero-Shot setting, therefore breaking the premise that those classes have not been seen.

6.4.2 Ablation Study

Table 6.1 shows the impact of using the different components of CLASTER.

Impact of Clustering

We consider several baselines. First, omitting clustering, which is the equivalent of setting $\omega_i = \psi_i$, in Eq. 6.1. This is, ignoring the cluster centroids in the representation. This is referred to in Table 6.1 as “No clustering". Second, we use random clustering, which is assigning each instance to a random cluster. Finally, we use the standard K-means. We observe that using clusters is beneficial, but only if they are meaningful, as in the case of K-means.

Impact of using a visual-semantic representation

We compare to the standard representation, which only includes visual information, and keep everything the same. This is, clustering and classification are done using only the visual features. This is referred to in the table as “CLASTER w/o SE". We observe that there is a very wide gap between using and not using the semantic features at training time. This effect is present across all datasets, suggesting it is a general improvement in the feature learning. We also show a comparison of aggregation strategies and interaction between visual and semantic features in the Section 6.9.

Impact of different optimization choices

We make cluster centroids learnable parameters and use the standard SGD to optimize them (‘CLASTER w/o RL") We also test the use of the related work of NetVLAD to optimize the cluster (“CLASTER w/ NetVLAD"). We see that the proposed model outperforms NetVLAD by an average of 4.7% and the CLASTER w/o RL by 7.3% on the UCF101 dataset. A possible reason for this difference is that the loss is back-propagated through multiple parts of the model before reaching the centroids. However, with RL the centroids are directly updated using the reward signal. Section 6.4.6 explores how the clusters change after the RL optimization. In a nutshell, the RL optimization essentially makes the clusters cleaner, moving most instances in a class to the same cluster.

Component	HMDB51	Olympics	UCF101
No clustering	25.6 ± 2.8	57.7 ± 3.1	31.6 ± 4.6
Random clustering (K=6)	20.2 ± 4.2	55.4 ± 3.1	24.1 ± 6.3
K-means (K=6)	27.9 ± 3.7	58.6 ± 3.5	35.3 ± 3.9
Soft K-means (K=6)	27.7 ± 3.9	58.1 ± 3.9	35.5 ± 3.6
Gumbel Softmax	26.6 ± 3.1	55.5 ± 3.1	29.8 ± 2.4
CLASTER w/o SE	27.5 ± 3.8	55.9 ± 2.9	39.4 ± 4.4
CLASTER w/o RL	30.1 ± 3.4	60.5 ± 1.9	39.1 ± 3.2
CLASTER w/ NetVLAD	33.2 ± 2.8	62.6 ± 4.1	41.7 ± 3.8
CLASTER	**36.8 ± 4.2**	**63.5 ± 4.4**	**46.4 ± 5.1**

Table 6.1: Results of the ablation study of different components of CLASTER ZSL. The study shows the effect of clustering, using visual-semantic representations, and optimizing with different methods. All three components show a wide improvement over the various baselines, suggesting that they are indeed complementary to improve the final representation.

6.4.3 Results on ZSL

Table 8.3 shows the comparison between CLASTER and several state-of-the-art methods: the out-of-distribution detector method (OD) [116], a generative approach to Zero-Shot action recognition (GGM) [122], the evaluation of output embeddings (SJE) [2], the feature generating networks (WGAN) [185], the end-to-end training for realistic applications approach (E2E) [15], the inverse autoregressive flow (IAF) based generative model, bi-directional adversarial GAN(Bi-dir GAN) [121] and prototype sampling graph neural network (PS-GNN) [53]. To make results directly comparable, we use the same backbone across all of them, which is the I3D [20] pre-trained on Kinetics.

We observe that the proposed CLASTER consistently outperforms all other state-of-the-art methods across all datasets. The improvements are significant: up to 3.5% on HMDB51 and 13.5% on UCF101 with manual semantic embedding. We also measure the impact of different semantic embeddings, including using sentence2vec instead of word2vec. We show that sentence2vec significantly improves over using word2vec, especially on UCF101 and HMDB51. Combination of embeddings resulted in average improvements of 0.3%, 0.8% and 0.9% over the individual best performing embedding of CLASTER.

Method	SE	Olympics	HMDB51	UCF101
SJE [2]	M	47.5 ± 14.8	-	12.0 ± 1.2
Bi-Dir GAN [121]	M	53.2 ± 10.5	-	24.7 ± 3.7
IAF [121]	M	54.9 ± 11.7	-	26.1 ± 2.9
GGM [122]	M	57.9 ± 14.1	-	24.5 ± 2.9
OD [116]	M	65.9 ± 8.1	-	38.3 ± 3.0
WGAN [185]	M	64.7 ± 7.5	-	37.5 ± 3.1
CLASTER (ours)	M	**67.4 ± 7.8**	-	**51.8 ± 2.8**
SJE [2]	W	28.6 ± 4.9	13.3 ± 2.4	9.9 ± 1.4
IAF [121]	W	39.8 ± 11.6	19.2 ± 3.7	22.2 ± 2.7
Bi-Dir GAN [121]	W	40.2 ± 10.6	21.3 ± 3.2	21.8 ± 3.6
GGM [122]	W	41.3 ± 11.4	20.7 ± 3.1	20.3 ± 1.9
WGAN [185]	W	47.1 ± 6.4	29.1 ± 3.8	25.8 ± 3.2
OD [116]	W	50.5 ± 6.9	30.2 ± 2.7	26.9 ± 2.8
PS-GNN [53]	W	61.8 ± 6.8	32.6 ± 2.9	43.0 ± 4.9
E2E [15]*	W	61.4 ± 5.5	33.1 ± 3.4	46.2 ± 3.8
CLASTER (ours)	W	**63.8 ± 5.7**	**36.6 ± 4.6**	**46.7 ± 5.4**
CLASTER (ours)	S	**64.2 ± 3.3**	**41.8 ± 2.1**	**50.2 ± 3.8**
CLASTER (ours)	C	**67.7 ± 2.7**	**42.6 ± 2.6**	**52.7 ± 2.2**
ER [23]	ED	60.2 ± 8.9	35.3 ± 4.6	51.8 ± 2.9
CLASTER (ours)	ED	**68.4 ± 4.1**	**43.2 ± 1.9**	**53.9 ± 2.5**

Table 6.2: Results on ZSL. SE: semantic embedding, M: manual representation, W: word2vec embedding, S: sentence2vec, C: Combination of embeddings. The proposed CLASTER outperforms previous state-of-the-art across tasks and datasets.

6.4.4 Results on GZSL

We now compare to the same approaches in the GZSL task in Table 6.3, the reported results are the harmonic mean of the seen and unseen class accuracies. Here CLASTER outperforms all previous methods across different modalities. We obtain an improvement on average of 2.6% and 5% over the next best performing method on the Olympics dataset using manual representations and word2vec respectively. We obtain an average improvement of 6.3% over the next best performing model on the HMDB51 dataset using word2vec. We obtain an improvement on average performance by 1.5% and 4.8% over the next best performing model on the UCF101 dataset using manual representations and word2vec respectively. Similarly to ZSL, we show generalized performance improvements using sentence2vec. We also report results on the combination of embeddings. We see an improvement of 0.3%, 0.6% and 0.4% over the individual best embedding for CLASTER. The seen and unseen accuracies are shown in Section 6.10.

Method	SE	Olympics	HMDB51	UCF101
Bi-Dir GAN [121]	M	44.2 ± 11.2	-	22.7 ± 2.5
IAF [121]	M	48.4 ± 7.0	-	25.9 ± 2.6
GGM [122]	M	52.4 ± 12.2	-	23.7 ± 1.2
WGAN [185]	M	59.9 ± 5.3	-	44.4 ± 3.0
OD[116]	M	66.2 ± 6.3	-	49.4 ± 2.4
CLASTER (ours)	M	**68.8 ± 6.6**	-	**50.9 ± 3.2**
IAF [121]	W	30.2 ± 11.1	15.6 ± 2.2	20.2 ± 2.6
Bi-Dir GAN [121]	W	32.2 ± 10.5	7.5 ± 2.4	17.2 ± 2.3
SJE [2]	W	32.5 ± 6.7	10.5 ± 2.4	8.9 ± 2.2
GGM[122]	W	42.2 ± 10.2	20.1 ± 2.1	17.5 ± 2.2
WGAN [185]	W	46.1 ± 3.7	32.7 ± 3.4	32.4 ± 3.3
PS-GNN [53]	W	52.9 ± 6.2	24.2 ± 3.3	35.1 ± 4.6
OD [116]	W	53.1 ± 3.6	36.1 ± 2.2	37.3 ± 2.1
CLASTER (ours)	W	**58.1 ± 2.4**	**42.4 ± 3.6**	**42.1 ± 2.6**
CLASTER (ours)	S	**58.7 ± 3.1**	**47.4 ± 2.8**	**48.3 ± 3.1**
CLASTER (ours)	C	**69.1 ± 5.4**	**48.0 ± 2.4**	**51.3 ± 3.5**

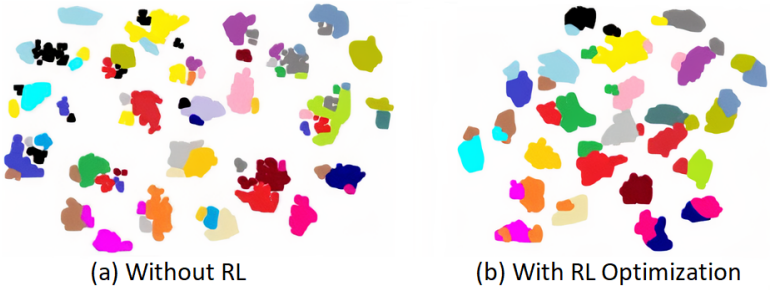
Table 6.3: Results on GZSL. SE: semantic embedding, M: manual representation, W: word2vec embedding, S: sentence2vec, C: combination of embeddings. The seen and unseen class accuracies are listed in the Section 6.10.

6.4.5 Results on TruZe

We also evaluate on the more challenging TruZe split. The proposed UCF101 and HMDB51 splits have 70/31 and 29/22 classes (represented as training/testing). We compare to WGAN [185], OD [116] and E2E [15] on both ZSL and GZSL scenarios. Results are shown in Table 6.4.

Method	UCF101		HMDB51	
	ZSL	GZSL	ZSL	GZSL
WGAN	22.5	36.3	21.1	31.8
OD	22.9	42.4	21.7	35.5
E2E	45.5	45.9	31.5	38.9
CLASTER	**45.8**	**47.3**	**33.2**	**44.5**

Table 6.4: Results on TruZe. For ZSL, we report the mean class accuracy and for GZSL, we report the harmonic mean of seen and unseen class accuracies. All approaches use sen2vec annotations as the form of semantic embedding.



(a) Without RL (b) With RL Optimization

Figure 6.5: CLASTER improves the representation and clustering in unseen classes. The figure shows t-SNE [115] of video instances, where each color corresponds to a unique unseen class label. The RL optimization improves the representation by making it more compact: in (b) instances of the same class, i.e. same color, are together and there are less outliers for each class compared to (a).

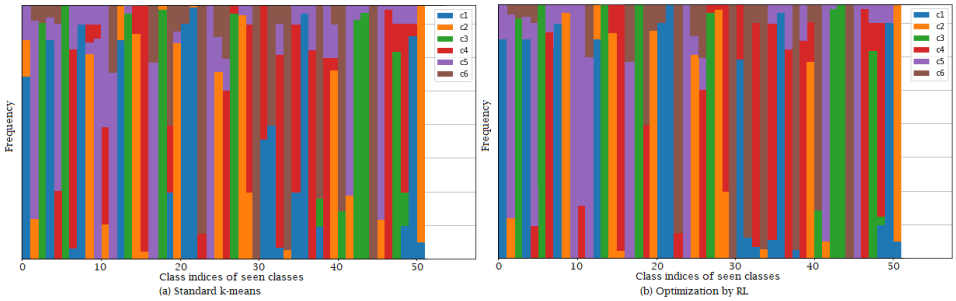


Figure 6.6: Analysis of how RL optimization changes the cluster to which an instance belongs. The frequencies are represented as percentages of instances in each cluster. We can see that the clusters are a lot "cleaner" after the optimization by RL.

6.4.6 Analysis of the RL optimization

We analyze how optimizing with RL affects clustering on the UCF101 training set. Figure 6.5 shows the t-SNE [115] visualization. Each point is a video instance in the unseen classes, and each color is a class label. As it can be seen, the RL optimization makes videos of the same class appear closer together.

We also do a quantitative analysis of the clustering. For each class in the training set, we measure the distribution of clusters that they belong to, visualized in the Fig 6.6. We observe that after the RL optimization, the clustering becomes “cleaner". This is, most instances in a class belong to a dominant cluster. This effect can be measured using the purity of the cluster:

$$Purity = \frac{1}{N}\sum_{i=1}^{k} max_j \left| c_i \cap t_j \right|, \tag{6.7}$$

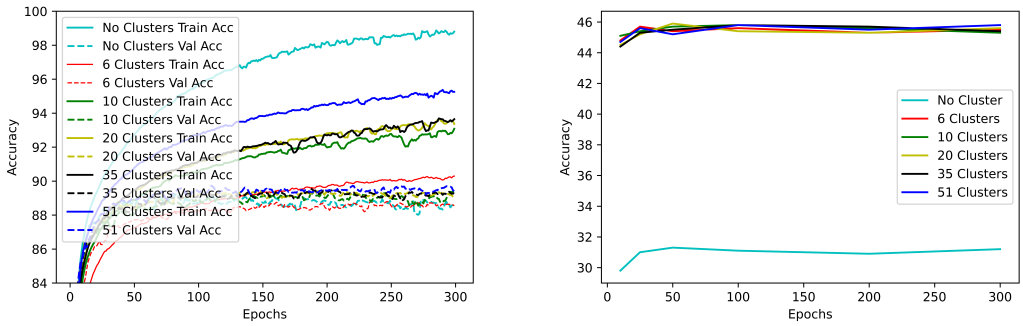


Figure 6.7: Left: Learning curve for the seen classes. Right: Accuracy curve for the unseen classes. The clustering-based representation avoids overfitting, which in the case of seen classes means that the gap between validation and training accuracy is smaller than in the vanilla representation. This regularization effect improves the accuracy in unseen classes.

where N is the number of data points (video instances), k is the number of clusters, c_i is a cluster in the set of clusters, and t_j is the class which has the maximum count for cluster c_i. Poor clustering results in purity values close to 0, and a perfect clustering will return a purity of 1. Using K-means, the purity is 0.77, while optimizing the clusters with RL results in a purity of 0.89.

Finally, we observe another interesting side effect of clustering. Some of the most commonly confused classes before clustering (e.g. “Baby crawling" vs. “Mopping floor", “Breaststroke" vs. “front crawl", “Rowing vs. front crawl") are assigned to different clusters after RL, resolving confusion. This suggests that clusters are also used as a means to differentiate between similar classes.

6.5 Regularization Effect of Clustering

Here, we look at the same effect with 20 and 35 clusters as well. We see consistent improvements of over 15% in accuracy for the unseen classes using the proposed CLASTER representation compared to no clustering.

In addition, we show that using only 35% of the data of seen classes for training also benefits from clustering on the unseen classes. This can be seen in Figure 6.8 While in the case of seen classes, using no clustering has the highest validation accuracy, at test time for the unseen classes, clustering leads to the best results. There are a few interesting points to note here. First, no clustering results in clear overfitting. The

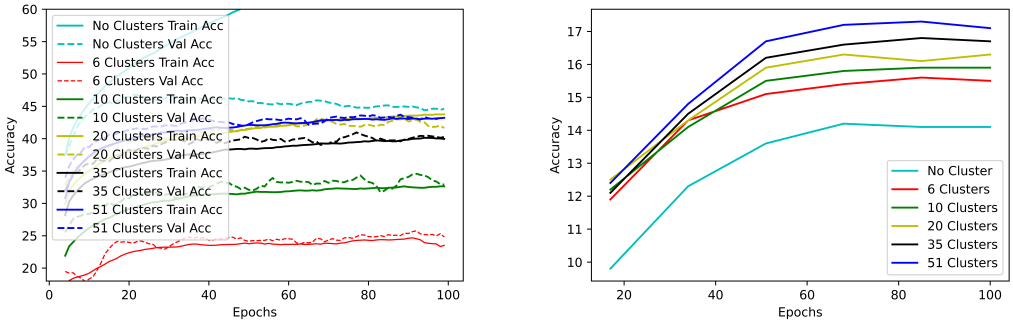


Figure 6.8: Left: Learning curve for the seen classes using 35% of the data. Right: Learning curve for the unseen classes. The clustering-based representation avoids overfitting, which in the case of seen classes means that the gap between validation and training accuracy is smaller than in the vanilla representation. This regularization effect improves the accuracy in unseen classes.

training accuracy reaches over 80% while the validation accuracy reaches a peak of 46% before dropping. However, using clustering results in the training and validation curves to be really close to each other. Another interesting point is that when there is a limited number of samples, having more clusters results in better performance at test time. This was not the case when we had all samples for the seen classes. When having all samples at training time, the number of clusters resulted in the same average accuracy as can be seen in Section 6.8.

6.6 Statistical Significance

We consider the dependent t-test for paired samples. This test is utilized in the case of dependent samples, in our case different model performances on the same random data split. This is a case of a paired difference test. This is calculated as shown in Eq 6.8.

$$t = \frac{\bar{X}_D - \mu_0}{s_D/\sqrt{n}} \tag{6.8}$$

Where $\bar{X}_D$ is the average of the difference between all pairs and s_D is the standard deviation of the difference between all pairs. The constant μ_0 is zero in case we wish to test if the average of the difference is different; n represents the number of samples, $n = 10$ in our case. The comparisons can be seen in Table 6.5. The lower the value of 'p', higher the significance.

Pairs	Dataset	t-value	Statistical significance($p<0.05$)	Type
CLASTER and OD [116]	UCF101	-15.77	Significant, $p<0.00001$	ZSL
CLASTER and WGAN [185]	UCF101	-9.08	Significant, $p<0.00001$	ZSL
CLASTER and E2E [15]	UCF101	-0.67	Not Significant, p = 0.26	ZSL
OD [116] and WGAN [185]	UCF101	-1.70	Not Significant, $p=0.12278$	ZSL
CLASTER and OD [116]	HMDB51	-4.33	Significant, $p=0.00189$	ZSL
CLASTER and WGAN [185]	HMDB51	-5.54	Significant, $p=0.00036$	ZSL
CLASTER and E2E [15]	HMDB51	-3.77	Significant, p = 0.00219	ZSL
OD [116] and WGAN [185]	HMDB51	-3.71	Significant, $p=0.00483$	ZSL
CLASTER and OD [116]	Olympics	-9.06	Significant, $p<0.00001$	ZSL
CLASTER and WGAN [185]	Olympics	-11.73	Significant, $p<0.00001$	ZSL
CLASTER and E2E [15]	Olympics	-2.72	Significant, p = 0.012	ZSL
OD [116] and WGAN [185]	Olympics	-2.47	Significant, $p=0.03547$	ZSL
CLASTER and OD [116]	UCF101	-4.51	Significant, $p=0.00148$	GZSL
CLASTER and WGAN [185]	UCF101	-5.49	Significant, $p=0.00039$	GZSL
OD [116] and WGAN [185]	UCF101	-3.16	Significant, $p=0.01144$	GZSL
CLASTER and OD [116]	HMDB51	-5.08	Significant, $p=0.00066$	GZSL
CLASTER and WGAN [185]	HMDB51	-7.51	Significant, $p=0.00004$	GZSL
OD [116] and WGAN [185]	HMDB51	-5.27	Significant, $p=0.00051$	GZSL
CLASTER and OD [116]	Olympics	-5.79	Significant, $p=0.00026$	GZSL
CLASTER and WGAN [185]	Olympics	-8.39	Significant, $p=0.00002$	GZSL
OD [116] and WGAN [185]	Olympics	-6.22	Significant, $p=0.00014$	GZSL

Table 6.5: Comparison of the t-test for different pairs of models on the same random split. Lower the value of 'p', higher the significance. As we can see, our results are statistically significant in comparison to both OD [116] and WGAN [185] in both ZSL and GZSL. For GZSL, OD [116] also achieves results that are significant in comparison to WGAN [185].

As we can see, our results are statistically significant in comparison to both OD [116] and WGAN [185] in both ZSL and GZSL. We also see that our results are statistically significant for both HMDB51 and Olympics in comparison to E2E [15]. In GZSL, OD [116] also achieves results that are significantly different in comparison to WGAN [185].

6.7 Average of Differences in Performance for Same Splits

Since the performance of the model varies for each random split (as witnessed by the standard deviation values), we average the difference in performance between CLASTER, OD, WGAN and E2E on the same splits. We believe that this gives us a better metric to check the performance of CLASTER with the other approaches. The results are depicted in Table 6.6.

Models	Setting	Olympics	HMDB51	UCF101
Ours and WGAN [185]	ZSL	17.5 $\pm$ 4.5	7.0 $\pm$3.8	17.4 $\pm$ 5.7
Ours and OD [116]	ZSL	13.6 $\pm$ 4.5	2.4 $\pm$ 1.6	14.3 $\pm$ 2.7
Ours and E2E [15]	ZSL	2.6 $\pm$ 2.8	3.7 $\pm$ 2.8	0.4 $\pm$ 1.8
Ours and WGAN [185]	GZSL	11.2 $\pm$ 4.0	9.3 $\pm$ 3.7	8.1 $\pm$ 4.4
Ours and OD [116]	GZSL	4.6 $\pm$ 2.4	5.2 $\pm$ 3.1	2.7 $\pm$ 1.8

Table 6.6: Comparing the average of the difference in performance for recent state-of-the-art approaches in zero-shot and generalized zero-shot action recognition on the same splits. All results were computed using sen2vec as the embedding. We can see that we outperform recent approaches in every scenario.

6.8 Number of Clusters

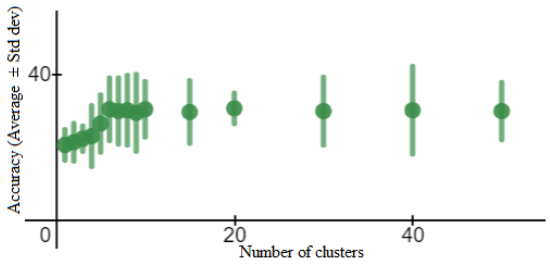


Figure 6.9: Effect of using different number of clusters. The green line represents the standard deviation. The reported accuracy is on the UCF101 dataset. As can be seen, the average cluster accuracy increases till about 6 clusters and then remains more or less constant. The vertical lines correspond to the standard deviation.

We test using different number of clusters on the UCF-101 dataset and show the results in Figure 6.9. These are for 5 runs on random splits. As we can see, the average accuracy increases until 6 clusters, and after that remains more or less constant. Thus, we use 6 clusters and continue with the same number for both HMDB51 and Olympics. For images, similarly, we used 5 random splits of CUB and found the performance stabilizes after having 9 clusters and use the same number of clusters for the other image datasets.

6.9 Comparison of aggregation strategies

Below, in Table 6.7 we show other aggregation options such as averaging and dot product. All results are using ED as semantic embedding.

Method	HMDB51
Average	33.1 $\pm$ 2.9
Dot Product	33.9 $\pm$ 3.2
Weighted Average	35.3 $\pm$ 3.6
Concatenation	43.2 $\pm$ 1.9

Table 6.7: Results on different aggregation options for the semantic and visual embeddings.

6.10 Seen and Unseen Class Performance for GZSL

In order to better analyze performance of the model on GZSL, we report the average seen and unseen accuracies along with their harmonic mean. The results using different embeddings and on the UCF101, HMDB51 and Olympics datasets are reported in Table 6.8. The reported results are on the same splits for fair comparison [65].

6.10.1 Results on Images

Our method also generalizes to the image domain as shown in Table 6.9. CLASTER outperforms previous work in five out of eight tasks and obtains comparable results in the remaining three tasks. We compare our approach to several state-of-the-art methods: Region Graph Embedding Network with the Parts Relation Reasoning branch

Model	E	Olympics			HMDB51			UCF-101		
		u	s	H	u	s	H	u	s	H
WGAN [185]	A	50.8	71.4	59.4	-	-	-	30.4	83.6	44.6
OD [116]	A	61.8	71.1	66.1	-	-	-	36.2	76.1	49.1
CLASTER	A	66.2	71.7	68.8	-	-	-	40.2	69.4	50.9
WGAN [185]	W	35.4	65.6	46.0	23.1	55.1	32.5	20.6	73.9	32.2
OD [116]	W	41.3	72.5	52.6	25.9	55.8	35.4	25.3	74.1	37.7
CLASTER	W	49.2	71.1	58.1	35.5	52.8	42.4	30.4	68.9	42.1
WGAN [185]	S	36.1	66.2	46.7	28.6	57.8	38.2	27.5	74.7	40.2
OD [116]	S	42.9	73.5	54.1	33.4	57.8	42.3	32.7	75.9	45.7
CLASTER	S	49.9	71.3	58.7	42.7	53.2	47.4	36.9	69.8	48.3
CLASTER	C	66.8	71.6	69.1	43.7	53.3	48.0	40.8	69.3	51.3

Table 6.8: Seen and unseen accuracies for CLASTER on different datasets using different embeddings. 'E' corresponds to the type of embedding used, wherein 'A', 'W', 'S' and 'C' refers to manual annotations, word2vec, sen2vec and combination of the embeddings respectively. 'u', 's' and 'H' corresponds to average unseen accuracy, average seen accuracy and the harmonic mean of the two. All the reported results are on the same splits.

(RGEN) and without the branch (R-PRR) [189], the evaluation of output embeddings(SJE) [2], the feature generating networks (WGAN) [185], the feature generating framework (VAEGAN) [187] and the latent embedding feedback method (LEF) [125]. We compare to the most recent approaches, but this is not the exhaustive suite of all relevant methods.

Method	CUB		SUN		AWA2		APY	
	ZSL	GZSL	ZSL	GZSL	ZSL	GZSL	ZSL	GZSL
SJE [2]	53.9	33.6	53.7	19.8	61.9	14.4	32.9	6.9
WGAN [185]	57.3	52.3	60.8	40.0	68.2	60.2	-	-
VAEGAN [187]	72.9	68.9	65.6	43.1	70.3	65.2	-	-
LEF [125]	74.3	**70.7**	66.7	**46.3**	73.4	66.7	-	-
R-PRR [189]	75.0	64.7	63.4	36.2	72.5	69.7	43.9	36.3
RGEN [189]	76.1	66.1	63.8	36.8	73.6	71.5	**44.4**	37.2
CLASTER	**76.4**	69.8	**67.2**	44.8	**74.1**	**72.7**	44.1	**40.8**

Table 6.9: Results on image datasets. For ZSL, we report the mean class accuracy and for GZSL, we report the harmonic mean of seen and unseen class accuracies. All approaches use manual annotations as the form of semantic embedding.

6.11 Limitations

CLASTER has some limitations that could be addressed in future work. The RL optimization for the cluster centroids can be unstable and time-consuming to train. More recent methods have explored different directions beyond CLASTER. For example, [123] surveys more sample-efficient RL techniques that could help stabilize training. JigsawNet and X-CLIP uses stronger vision-language backbones and that is orthogonal to CLASTER. These methods address some limitations of CLASTER's RL unstability while tackling the zero-shot action recognition problem from new angles. However, CLASTER's core ideas of leveraging clustering and visual-semantic representations remain relevant as can be seen in recent works [42, 210, 51]. These approaches focus on individual vision or language optimization of clusters without RL. Extending CLASTER with newer training techniques or network architectures could be promising future work.

6.12 Conclusion

Zero-Shot action recognition is the task of recognizing action classes without any visual examples. The challenge is to map the knowledge of seen classes at training time to that of novel unseen classes at test time. We propose a novel model that learns clustering-based representation of visual-semantic features, optimized with RL. We observe that all three of these components are essential. The clustering helps regularizing, and avoids overfitting to the seen classes. The visual-semantic representation helps improve the representation. And the RL yields better, cleaner clusters. The results is remarkable improvements across datasets and tasks over all previous state-of-the-art, up to 11.9% absolute improvement on HMDB51 for GZSL.

Chapter 7

Transferring Semantic Knowledge through Action Stories for Zero-Shot Action Recognition

7.1 Introduction

Action recognition technology has improved remarkably over the years, with methods becoming more accurate and even pushing the boundaries to include novel tasks [69]. However, one of the main challenges that remain today is the dependency of these methods on annotated data for novel categories. In practice, obtaining hundreds of annotated examples for each new class we aim to recognize is not realistic. This is particularly true as we grow the number of classes and wish to incorporate more flexible, natural language, for example, for retrieval. This general problem has led to research in the zero-shot domain.

In the typical zero-shot setting, there are seen classes that contain visual examples and their class label, and there are unseen classes where only the class labels are available at training time. Given a visual sample of the unseen set at test time, the task is to output the corresponding class label. Approaches typically learn a mapping from the visual space to the class labels using the seen classes and leverage that mapping in different ways to approximate the mapping in the unseen classes. Some examples of solutions include learning to map visual information and class labels to the same space or learning to generate visual features using a generative adversarial network (GAN) [185].

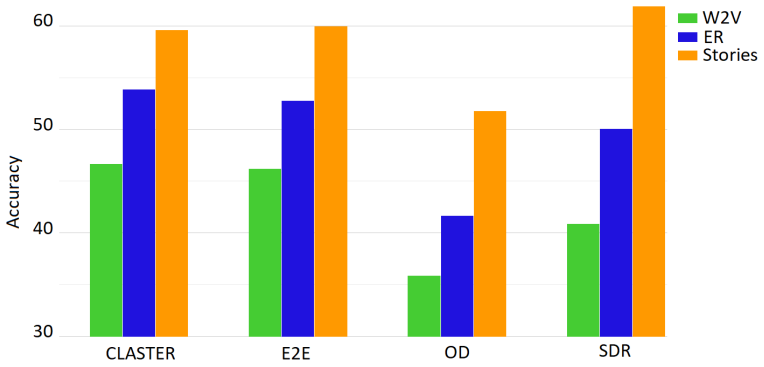


Figure 7.1: Comparison of accuracy across state-of-the-art zero-shot approaches (CLASTER [64], E2E [15], OD [116], and the proposed SDR using different semantic embeddings (the proposed Stories, word2vec [120] (W2V) and elaborative definitions [23] (ER), on the UCF101 [157] dataset. Using the proposed Stories to create the semantic space of class labels improves the performance by a large margin, showing that it is model-agnostic.

One of the underlying assumptions of these approaches is that the distances between the data points are meaningful both in the visual space and the semantic space. In other words, data points that are close together should be similar in content across both seen and unseen classes. In visual space, this tends to happen naturally as related classes will share objects, scenes, etc. For example, if we compare classes such as “penalty shot" and “playing soccer", they will share the ball, the soccer field, etc. However, in the space of action labels, which we also refer to as semantic space, this property is not straightforward to achieve. While some similar classes will contain overlapping terms (such as “horse-back riding" and “horse racing"), some others might be similar but not contain overlapping words (such as the previous example of “penalty shot" and “playing soccer"). This makes the step of transferring knowledge between seen and unseen classes harder. Previous efforts to improve the semantic space of class labels have included the use of manually annotated attributes or embedding functions trained on language corpora, such as word2vec [120], sentence2vec [130] and using definitions of actions [23].

In this work, we address the problem of building a meaningful space of action labels by leveraging the story around each action. In particular, we use the descriptions of the steps needed to achieve each action, which is the *Story* around this action and encode them using a language embedding. These steps typically contain the objects, tools, scenes, verbs, adjectives, etc., associated with the action label. One could think of all these additional pieces of information around an action as the “common sense"

of associations humans would typically consider. For example, in the case of the penalty shot, the steps would describe to first place the ball, run the hand through the grass, fluff the ball, take some steps back, kick, etc. When playing soccer, the steps include kicking the ball with the inside of your shoe for short passes across the grass, tapping the ball from foot to foot, etc. When we compare the stories of steps around these two classes, the overlap of terms becomes much more obvious. It is more likely that these related classes are closer in semantic space, facilitating the transfer of knowledge between seen and unseen classes. We show that this relatively simple approach to creating a semantic space is extremely effective across datasets and methods, improving performance by up to 20% compared to the standard word2vec [120]. Figure 7.1 shows that all state-of-the-art methods improve significantly and therefore the proposed semantic embedding is general and model-agnostic.

We leverage the richness of this semantic space to build a method for zero-shot action recognition. In particular, following feature-generating approaches of the past [116, 185], we use a GAN [56] to synthesize visual data points from these semantic embeddings. These synthetic visual data points are then used to learn a mapping from visual to semantic space in the unseen classes. Although this general approach works well, training GANs is notoriously slow and unstable. Part of the reason for the difficult training is the Gaussian distribution of noise used for the GAN [109], which is not realistic in the case of video datasets with several classes. This leads to many of the generated samples being obviously synthetic and, therefore, spending too much time during training. Instead, we propose to use a data-driven noise module, where noise is created using the real distribution of data points in the seen set. In particular, we train a variational autoencoder (VAE) to reconstruct visual features. We use the encoding part of the VAE to map visual features of seen classes to a subspace and use them as “noise" for the GAN [109]. We call this data-driven noise or data-based noise (DBN). We observe that this is a more faithful distribution of visual data points and therefore helps to reduce the time it takes for the GAN to train by about 70% while also improving accuracy. Finally, we train our method using a novel contrastive loss, which we call ranking loss, that pushes apart semantic features of different classes and pulls together those of the same class. This loss also helps across datasets, improving by up to 5%.

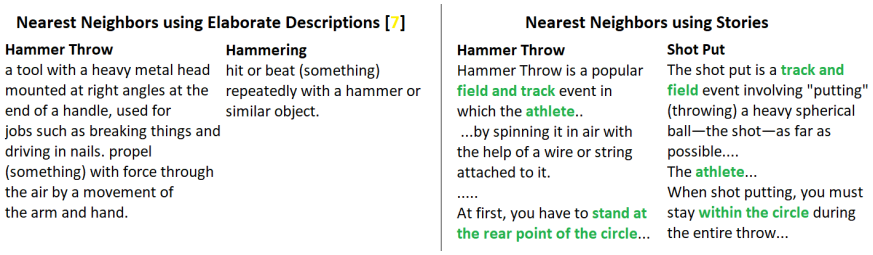


Figure 7.2: Comparing nearest neighbors using Stories. We see an example where ER fails and Stories provides more context and helps in obtaining better neighbors. This is one example of where ER fails, there are multiple such examples. Dataset is UCF101.

The result is a method which we call SDR for Stories, Data-driven noise and Ranking loss. Experimental results show that the proposed SDR improves state-of-the-art by up to 6.1%, across different datasets, and other tasks, including the generalized zero-shot task. We expect these three technical contributions to greatly help advance zero-shot action recognition technology.

7.2 The Story Dataset

Research in semantic representation of action labels has shown over the years that more sophisticated representations help to build a meaningful semantic space for zero-shot action recognition. Here we go beyond previous work by representing not only the class label but the story around it. This is, all the steps needed to perform the action, which include objects, verbs, etc, typically associated with that action. We call these representations *Stories*, and we now describe how we build them.

7.2.1 Building the Stories Dataset

We leverage textual descriptions of actions from *WikiHow*[1], a website that gives instructions on how to perform actions. These instructions consist of long paragraphs that describe each step in completing the action. For example, for the action classes in the HMDB51 dataset, the *WikiHow* articles contain an average of 9.8 steps, ranging from 4 to 20 steps. The most closely related work to us, ER [23] uses a single

1. `https://www.wikihow.com/`

sentence per class. Instead, for the proposed *Stories*, the average number of lines is 14.4 for the classes in the Olympics dataset, 9.6 for HMDB51, 13.2 for UCF101, and 13.5 Kinetics. These rich descriptions inherently contain information of the objects needed to perform an action.

Not all classes have articles on *WikiHow*. For example, if we search for “draw sword" (a class in UCF101), we will get instructions on how to paint or sketch swords instead of the steps needed to essentially remove a sword from its sheath. Hence, collecting clean, meaningful articles requires a more complex process than a simple search. After scraping the articles from *WikiHow* corresponding to all classes, we use sentence-BERT encoders [139] to represent the sentences in the article and use cosine similarity to find the 25 that are the most similar to the class definition [23]. Next, we manually check the sentences for each class. If we find a mismatch between the article and the action class, as was the case with the “draw sword" example, we do a manual search to pick the most relevant article from other sources such as *Wikipedia*. However, these alternative articles do not tend to contain the sequence of steps and hence need more manual intervention to order the sentences into a sequence of steps. We finally clean each story by re-arranging the sentences in sequential order and removing irrelevant sentences. In total, we had 6 people who manually cleaned the descriptions after the initial stage of noisy collection and a further 10 who verified the descriptions. This was done using the Prolific[2] platform. The time taken for the cleaning Stories for UCF101, HMDB51 and Kinetics was 7.2 hours, 3.3 hours and 25.3 hours on average respectively. We followed this process to create a dataset of these textual representations for classes in UCF101, HMDB51, Olympics, and Kinetics-400, as they are the most commonly used datasets for zero-shot action recognition.

7.2.2 What do Stories Look Like?

Figure 7.2 gives a concrete example of what *Stories* looks like and shows why they improve the neighborhood of classes with respect to previous approaches. We use as an example the class “Hammer Throw" from the UCF101 [157] dataset, which is a sporting event in which the athlete throws a spherical object. If we retrieve the nearest

2. https://www.prolific.co/

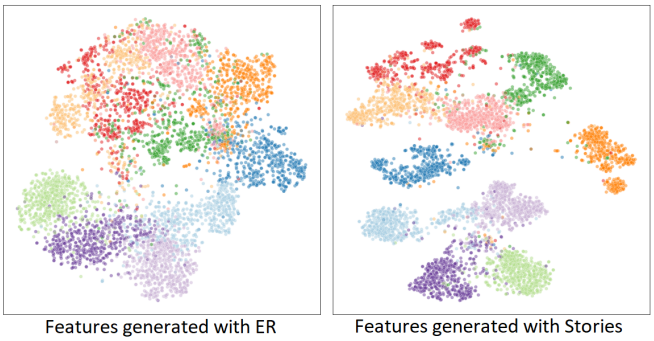


Figure 7.3: Visualization of the features generated from the *Stories* embedding vs ER, using t-SNE [115].

neighbor with ER [23], we obtain the class “Hammering", which is not actually related in meaning but both contain the description of the tool hammer. However, if we use *Stories*, the nearest neighbor is “Shot Put", which is also a sporting event where the athlete throws a spherical object.

7.2.3 Learning from Stories

To truly understand the effect of having richer semantic embeddings, we test the proposed *Stories* embeddings as input to several state-of-the-art methods. We compare to using the standard word2vec [120] embeddings and the more recent ER [23]. Figure 7.1 visualizes the gains obtained for multiple models. We observe gains of up to 21% over word2vec and 11.8% over ER. We visualize the effect of using ER and *Stories* to generate features, using the t-SNE [115] for 10 classes in UCF101, all related to gymnastics and therefore easier to confuse. We see that using *Stories* helps keep a more meaningful neighborhood for visual instances, and keep classes apart.

7.3 SDR

We now describe the three main components of the proposed SDR method. We first describe the setting, followed by the standard approach through feature generation, and finally we will delve into the technical details of the method. The overall feature-generating pipeline can be seen in Fig. 7.4.

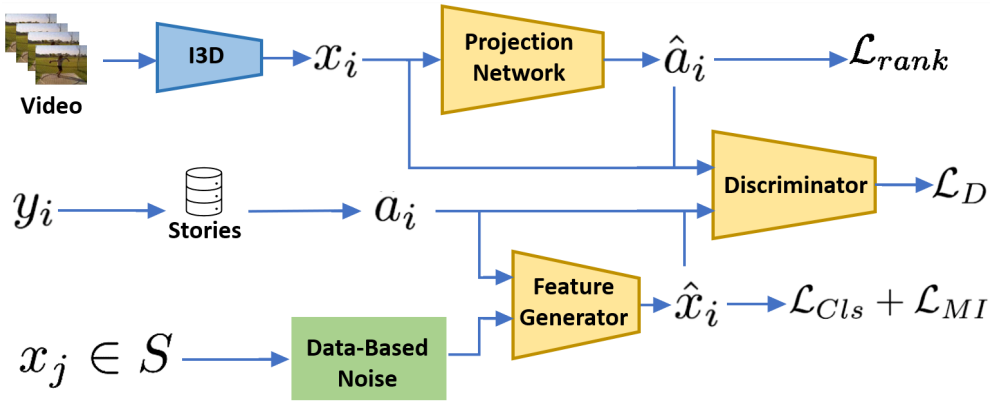


Figure 7.4: Overview of SDR. SDR generates video features that resemble real ones. It uses I3D to compute real visual features, x_i, and a "Projection Network" to generate synthetic semantic embeddings, $\hat{a}i$. The "Feature Generator" combines real semantic embeddings, ai, with noise to produce synthetic visual features, $\hat{x}i$. A discriminator is trained to distinguish between pairs $\hat{x}i$, $\hat{a}i$ and xi, a_i. The "Projection Network" is trained with a ranking loss, $\mathscr{L}_{rank}$, to separate synthetic semantic representations. The generator and discriminator engage in a joint training game.

7.3.1 Zero-Shot and Generalized Zero-Shot Settings

Let S be the training set of seen classes. S is composed of tuples $(x, y, a(y))$, where x represents the spatiotemporal features of a video, y represents the class label in the set of Y_S seen class labels, and $a(y)$ denotes the category-specific semantic representation of class y, which is either manually annotated or computed automatically, for example using word2vec [120] or the proposed *Stories*.

Let U be the set of pairs $(u, a(u))$, where u is a class in the set of unseen classes Y_U and $a(u)$ are the corresponding semantic representations. The seen classes Y_S and the unseen classes Y_U do not overlap.

In the zero-shot setting, given an input video, the task is to predict a class label in the unseen classes, such as $f_{ZSL} : X \rightarrow Y_U$. In the generalized zero-shot setting, given an input video, the task is to predict a class label in the union of the seen and unseen classes, as $f_{GZSL} : X \rightarrow Y_S \cup Y_U$.

7.3.2 Zero-Shot Action Recognition with Feature Generation

The general approach of the proposed SDR follows the standard feature generation pipeline for zero-shot recognition [185, 116, 122, 109, 72]. The high-level idea is to learn to generate visual features for unseen classes using a GAN [56], and then, given these synthetic features, train a classifier that takes in visual features and predicts unseen class labels.

The GAN comprises a generator (G), discriminator (D), and projection network (P). Generator G creates synthetic visual features ($\hat{x}_i$) from class label semantic embedding (a) and noise (z). P maps visual features (x) to a semantic embedding approximation ($\hat{a}$). D separates real from synthesized features. They're jointly trained via a mini-max game using an optimization function (see Eq 7.1).

$$\begin{aligned} \mathscr{L}_D = & \mathbb{E}_{(x,a)\sim p_{(x_S \times a_S)}}[D(x,P(x))] \\ & -\mathbb{E}_{z\sim p_z}\mathbb{E}_{a\sim p_a}[(D(G(a,z),a))] \\ & -\alpha\mathbb{E}_{z\sim p_z}\mathbb{E}_{a\sim p_a}\left[(\|\nabla_{\hat{x}}D(G(a,z))\|_2 - 1)^2\right], \end{aligned} \tag{7.1}$$

$p_{(x_S \times a_S)}$ is the joint distribution of visual and semantic descriptors for seen classes, p_a is the empirical distribution of their semantic embeddings, p_z is noise, and α is a penalty coefficient. Additional losses to enhance generated features are the classification regularized loss ($\mathscr{L}_{CLS}$) and the mutual information loss ($\mathscr{L}_{MI}$). These losses form the objective function minimized to train the vanilla pipeline (see Eq 7.2).

$$\min_G \min_P \max_D \mathscr{L}_{\mathscr{D}} + \lambda_1 \mathscr{L}_{CLS}(G) + \lambda_2 \mathscr{L}_{MI}(G). \tag{7.2}$$

Once these networks are trained on the seen classes, the generator is used to synthesize visual features for the unseen classes. The final step is to train a simple classifier using these synthetic visual features as input and the class labels. The loss is the standard cross-entropy loss, and the classifier is a simple multilayer perceptron (MLP).

7.3.3 Data-driven Noise

In the standard pipeline we have described, the typical generator takes as input attributes or semantic embeddings and normally distributed noise to generate visual features. The underlying assumption is that the normal distribution can represent all classes. However, this is not necessarily true [109]. Instead, we use the distribution of the seen classes' features to create the “noise" for the generator [109], such that the synthetic unseen classes will follow the same distribution.

To this end, we use a variational auto-encoder (VAE), which takes as input visual features from the seen classes and it is trained to reconstruct them. Here, the main role of the encoder is to convert the visual features to low-dimensional noise and the decoder then reconstructs the visual features from this noise. Once trained, we use the low-dimensional representation of the encoder as the “noise". More concretely, the loss function for the VAE is the standard:

$$\begin{aligned} \mathscr{L}(\theta,\phi) = &-\mathbb{E}_{z\sim q_\phi(z|x)}\left[p_\theta(x \mid z)\right] \\ &+D_{KL}\left(q_\phi(z \mid x) \parallel p_\theta(z)\right), \end{aligned} \tag{7.3}$$

where ϕ and θ are the parameters of the encoder p_θ and the decoder q_ϕ respectively, z is the sample latent vector (the “noise" in our case), x is the original feature and x' is the reconstructed feature.

Once the VAE is trained, to generate visual features of a particular unseen class, we retrieve the 3 most similar seen classes and sample visual features from these classes to generate the “noise", unlike earlier work [109], which uses random classes.

This simple change in the noise distribution, which we call data-driven or data-based noise (DBN), benefits in two ways. First, it improves the overall accuracy (Table 7.1). Most importantly, the time needed to train the feature generator is 65% lower compared to the standard method of using random Gaussian noise.

Figure 7.5 shows the effect on accuracy, where the model trained with data-driven noise converges much faster (50 epochs) compared to the standard model trained with Gaussian noise (150 epochs). The loss is also more stable, as shown in Fig. 7.6.

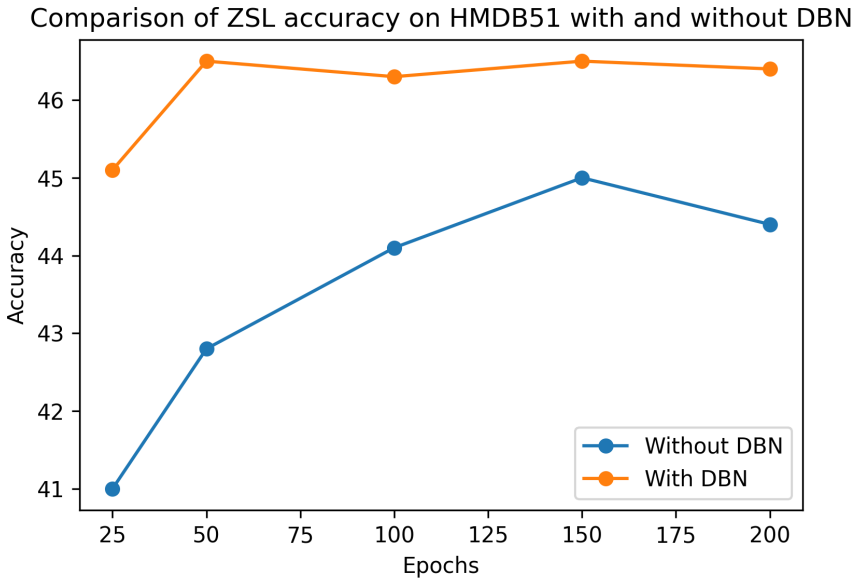


Figure 7.5: Training the generator using data-driven noise converges much faster than using the standard Gaussian noise. Also, saving the feature generator at various stages and testing on HMDB, we see that with using DBN we obtain our best performance at epoch 30 and the performance in later epochs are also consistent. However, without DBN we see the best performance at 150 epochs and the accuracies are a bit erratic.

7.3.4 Ranking Loss for Feature Generation

One of the risks of learning to generate semantic embeddings (through the “Projection Network" in Fig. 7.4) is that synthetic semantic embeddings can be too similar to each other. To avoid this, we introduce a ranking loss [47] that pushes apart the generated semantic representation ($\hat{a}_i$) from those of their neighboring classes:

$$\mathscr{L}_{rank} = \mathbb{E}[max(0, \delta - a^T \hat{a}_i + (a')^T \hat{a}_i)], \tag{7.4}$$

where a is the ground truth semantic embedding, a' is the semantic embedding of a class randomly sampled from the 5 classes (chosen empirically, more details in Section 7.7) closest to the ground truth and δ is a hyperparameter. Including this loss in the overall objective function, we obtain:

$$\begin{aligned} &\min_G \min_P \max_D \mathscr{L}_{\mathscr{D}} + \lambda_1 \mathscr{L}_{CLS}(G) \\ &+\lambda_2 \mathscr{L}_{rank}(P) + \lambda_3 \mathscr{L}_{MI}(G). \end{aligned} \tag{7.5}$$

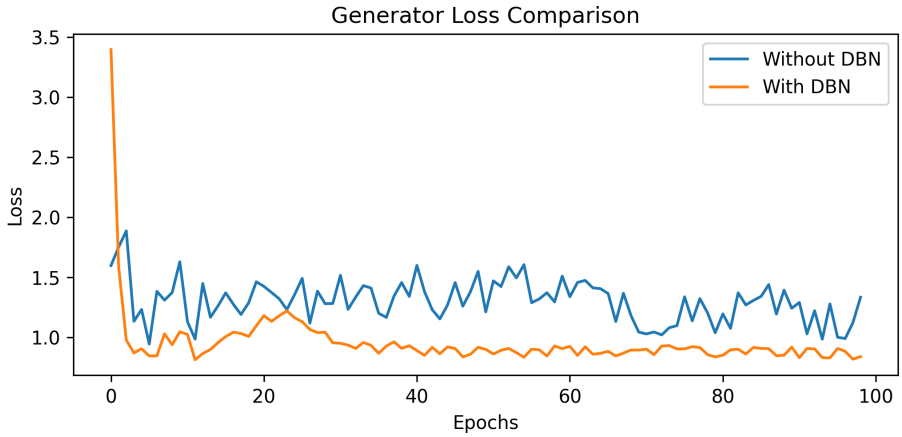


Figure 7.6: The generator loss using data-driven noise is much more stable, leading to faster convergence and better accuracy. We see that using DBN stabilizes the training of our feature generator whilst also ensuring we reach our optima quicker.

7.4 Experimental Analysis

7.4.1 Datasets and Evaluation Protocol

We use the Olympic Sports [127], HMDB-51 [90], UCF-101 [157] and Kinetics [20] datasets, as they are the standard choice in zero-shot action recognition, so that we can compare them with recent state-of-the-art models [50, 116, 137, 64, 136, 15].

These datasets contain 783, 6766 and 13320 videos, and have 16, 51 and 101 classes, respectively. We follow the commonly used 50/50 splits of Xu et al. [191], where 50% are the seen classes and 50 are the unseen classes. Similar to previous approaches [222, 50, 137, 119, 88], we report average accuracy and standard deviation over 10 independent runs. We also report on the recently introduced TruZe [65]. This split accounts for the fact that some classes present on the dataset used for pre-training (Kinetics [20]) overlap with some of the unseen classes in the datasets used in the zero-shot setting, therefore breaking the premise that those classes have not been seen. We also report on the Kinetics-220[20] split as proposed in ER [23]. Here, the 220 classes from Kinetics-600 [19] are treated as unseen classes to a model trained on the Kinetics-400 dataset.

7.4.2 Implementation details

Features

For our visual features we consider two scenarios. The first case, the appearance and flow features are extracted from the *Mixed 5c* layer of the RGB and flow I3D networks, respectively. Both I3D models are pre-trained on the Kinetics-400 dataset [20]. Given an input video, appearance and flow features extracted are averaged across the temporal dimension and pooled by 4 in the spatial dimension and then flattened to obtain a vector of size 4096 each. These vectors are then concatenated to obtain video features of size 8192. In the second case, we first train the X-CLIP-B/16 [126] on 16 frames of the non-overlapping classes of Kinetics [15] dubbed Kinetics-664 [15] using the proposed 'Stories' as the semantic embedding. For the text embeddings we use the large S-BERT [139], which is a sentence encoder. For ER we use the class definition as input to the S-BERT and use the 1024 sized vector output as the semantic embedding. In case of Stories, we use S-BERT for each sentence and average all the vectors to obtain a singe vector of size 1024.

Network Architecture

We use the Wasserstein GAN [185] which has been successful in both zero-shot image classification [186] and zero-shot action recognition [116] tasks. This also allows us to compare directly to OD [116] and Wasserstein GAN [185] in the experimental analysis. The feature generator G is a three-layer fully-connected network that has an output layer dimension equal to that of the video feature size. The hidden layers are of size 4096. The discriminator D is also a three-layer fully-connected network with hidden layers of size 4096. However, the output size equals 1. The projection network P is a fully-connected network that has an output layer size equal to the size of the semantic embeddings (in our case 1024).

Training Details

All the modules are trained using the Adam optimizer with a weight decay of 0.0005 and with an adaptive learning rate using a learning rate scheduler. We set λ_1 as 0.1, λ_2 as 0.9 and λ_3 as 0.1. At test time, we follow OD [116] and train a single classifier for ZSAR and two classifiers for GZSAR along with an out-of-distribution

Stories	DBN	$\mathscr{L}_{rank}$	ZSL_{HMDB}	ZSL_{UCF}	$GZSL_{HMDB}$	$GZSL_{UCF}$
×	×	×	29.1 ± 3.8	37.5 ± 3.1	32.7 ± 3.4	44.4 ± 3.0
×	×	✓	29.7 ± 3.5	38.0 ± 3.1	35.3 ± 3.1	47.1 ± 3.2
×	✓	×	30.6 ± 2.2	38.6 ± 3.4	33.3 ± 3.0	44.9 ± 2.9
✓	×	×	44.6 ± 2.9	60.4 ± 3.8	44.9 ± 3.6	51.0 ± 2.9
×	✓	✓	31.9 ± 3.2	40.9 ± 2.9	35.7 ± 2.9	47.9 ± 4.1
✓	×	✓	45.0 ± 2.5	60.9 ± 3.5	49.0 ± 3.2	54.4 ± 3.7
✓	✓	×	45.9 ± 2.7	61.4 ± 2.8	47.5 ± 2.6	53.7 ± 3.5
✓	✓	✓	**46.5 ± 5.3**	**61.9 ± 2.5**	**49.7 ± 2.9**	**54.9 ± 4.4**

Table 7.1: Ablation Study of Proposed Components. This table shows an ablation study evaluating the impact of each proposed component of SDR: using Stories embeddings, incorporating data-driven noise (DBN), and adding the ranking loss (Lrank). Results are reported for zero-shot (ZSL) and generalized zero-shot learning (GZSL) on HMDB and UCF101 datasets. The table demonstrates that each component provides gains over the baseline, and combining all three gives the best overall performance.

(OOD) detector. The classifiers are single-layer fully-connected networks with an input size equal to the video feature size and output sizes equal to the number of classes (seen or unseen). The OOD is a three-layer fully connected network with output and hidden layer sizes equal to the number of seen classes and 512, respectively.

7.4.3 Ablation Study

In order to truly understand the effect of the proposed components, we perform a thorough and extensive ablation study involving all proposed components. Results are shown in Table 7.1. We see that every proposed contribution benefits over the baseline. However, crucially, a combination of all three gives us our best results. We note that the improvement from the ranking loss is much more prominent in the generalized zero-shot setting than in the zero-shot.

7.4.4 Results without cleaning of data

In order to truly evaluate the effect of the Stories dataset, we evaluate multiple models on the noisy version of the Stories dataset and report results in Fig. 7.7. We see that using the noisy dataset improves the performance over ER by up to 4.6%. Using the cleaned version of Stories further improves performance over the noisy version by up to 7.2%. This shows us the cleaning of the data is not trivial and leads to significant gains.

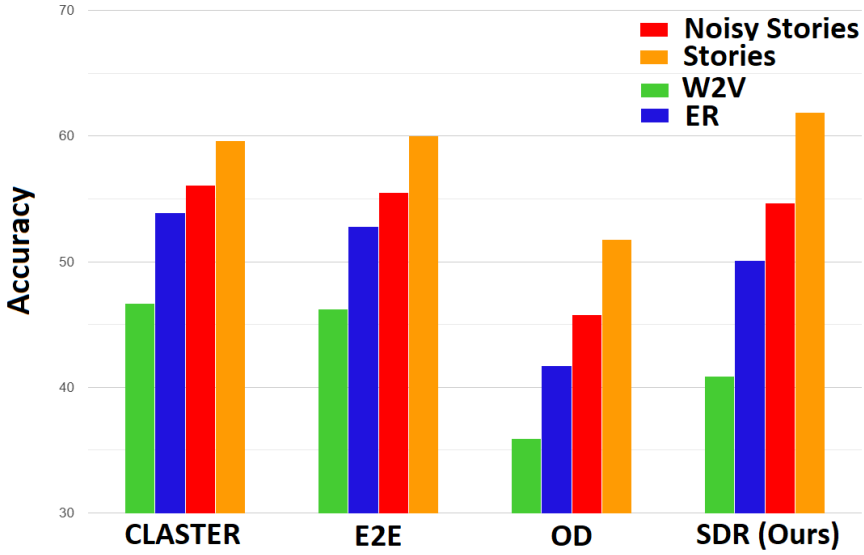


Figure 7.7: Comparison of accuracy across state-of-the-art zero-shot approaches (CLASTER [64], E2E [15], OD [116], and the proposed SDR using different semantic embeddings. Using the proposed *Stories* to create the semantic space of class labels improves the performance by a large margin, showing that it is model-agnostic.

7.4.5 Comparison to State-of-the-Art on Zero-Shot

Table 7.2 compares the proposed SDR to state-of-the-art on the zero-shot setting. We compare with the most recent best performing methods, which include the joint visual-semantic clustering method (CLASTER) [64], the out-of-distribution detector (OD) [116], the end-to-end training approach for realistic applications (E2E) [15], JigSawNet [136], among others.

We observe that the proposed SDR consistently outperforms all approaches across all datasets by gains of up to 6.1%. We experiment with using a single model for all datasets, by training on Kinetics and not doing any fine-tuning for the smaller datasets. This is the last row of the table, which we call “SDR (Ours) Single Model". It is remarkable and quite promising that, without the need to fine-tune, this single model achieves even better performance. We also evaluate on the Kinetics-220 dataset as proposed in ER [23]. There are fewer methods who report on this split, but it is interesting as it is much larger. Results are shown in Table 7.3. We observe that the proposed SDR outperforms all previous work, with significant gains of up to 5%. Finally, we evaluate on the stricter TruZe [65] split that ensures no overlap between pre-trained model and test classes. Results are shown in Table 7.4. We also observe a consistent improvement using our proposed approach.

Method	Olympics	HMDB51	UCF101
SJE [2]	47.5 ± 14.8	13.3 ± 2.4	12.0 ± 1.2
Bi-Dir GAN [121]	53.2 ± 10.5	21.3 ± 3.2	24.7 ± 3.7
IAF [121]	54.9 ± 11.7	19.2 ± 3.7	26.1 ± 2.9
GGM [122]	57.9 ± 14.1	20.7 ± 3.1	24.5 ± 2.9
OD [116]	65.9 ± 8.1	30.2 ± 2.7	38.3 ± 3.0
WGAN [185]	64.7 ± 7.5	29.1 ± 3.8	37.5 ± 3.1
PS-GNN [53]	61.8 ± 6.8	32.6 ± 2.9	43.0 ± 4.9
E2E [15]	61.4 ± 5.5	33.1 ± 3.4	46.2 ± 3.8
ER [23]	60.2 ± 8.9	35.3 ± 4.6	51.8 ± 2.9
CLASTER [64]	68.4 ± 4.1	43.2 ± 1.9	53.9 ± 2.5
JigSawNet [136]	-	38.7 ± 3.7	56.0 ± 3.1
ResT [104]	-	39.3 ± 3.5	58.7 ± 3.3
X-CLIP-B/16 [126]	-	44.6 ± 5.2	72.0 ± 2.3
SDR (Ours) + I3D	**72.5 ± 2.1**	**46.8 ± 5.0**	**62.9 ± 1.6**
SDR (Ours) + CLIP	**80.1 ± 2.3**	**52.7 ± 3.4**	**73.4 ± 2.7**
SDR (Ours) SM + I3D	**74.8 ± 2.3**	**48.9 ± 4.4**	**64.9 ± 2.1**
SDR (Ours) SM + CLIP	**82.2 ± 1.6**	**54.4 ± 4.1**	**75.5 ± 3.2**

Table 7.2: Results on zero-shot action recognition on the Olympics, HMDB51 and UCF101 datasets. 'SM' corresponds to the single model experiment.

Method	Top-1 Acc	Top-5 Acc
DEVISE [47]	23.8 ± 0.3	51.0 ± 0.6
SJE [2]	22.3 ± 0.6	48.2 ± 0.4
ER [23]	42.1 ± 1.4	73.1 ± 0.3
JigSawNet [136]	45.9 ± 1.6	78.8 ± 1.0
SDR (Ours)+I3D	**50.8 ± 1.9**	**82.9 ± 1.3**
SDR (Ours)+CLIP	**55.1 ± 2.2**	**86.1 ± 3.1**

Table 7.3: Results on zero-shot action recognition on Kinetics-220.

7.4.6 Comparison to State-of-the-Art on Generalized Zero-Shot

We evaluate SDR on the generalized setting where at test time both seen and unseen class samples are used. We train an out-of-distribution detector (OOD) following OD [116] and two separate classifiers for the seen and unseen classes along with the OOD network. Table 7.5 shows the results, with the harmonic mean of the seen and unseen class accuracies. The proposed SDR consistently outperforms all approaches across all datasets by gains of up to 5.4%.

Method	UCF101			HMDB51		
	Split	ZSL	GZSL	Split	ZSL	GZSL
WGAN	67/34	22.5	36.3	29/22	21.1	31.8
OD	67/34	22.9	42.4	29/22	21.7	35.5
CLASTER	67/34	45.8	47.3	29/22	33.2	44.5
SDR (Ours)	67/34	**49.7**	**51.3**	29/22	**34.9**	**45.5**
VCAP [38]	0/34	49.1	-	0/22	20.4	-
SDR (Ours)+I3D SM	0/34	**51.5**	**52.2**	0/22	**36.1**	**46.6**
SDR (Ours)+CLIP SM	0/34	**55.1**	**57.7**	0/22	**40.8**	**51.1**

Table 7.4: Results on TruZe. For zero-shot, we report the mean class accuracy, and for generalized zero-shot, we report the harmonic mean of seen and unseen class accuracies. The split refers to the train/test split used for each setting.

Method	Olympics	HMDB51	UCF101
Bi-Dir GAN [121]	44.2 $\pm$ 11.2	7.5 $\pm$ 2.4	22.7 $\pm$ 2.5
IAF [121]	48.4 $\pm$ 7.0	15.6 $\pm$ 2.2	25.9 $\pm$ 2.6
GGM [122]	52.4 $\pm$ 12.2	20.1 $\pm$ 2.1	23.7 $\pm$ 1.2
WGAN [185]	59.9 $\pm$ 5.3	32.7 $\pm$ 3.4	44.4 $\pm$ 3.0
OD[116]	66.2 $\pm$ 6.3	36.1 $\pm$ 2.2	49.4 $\pm$ 2.4
PS-GNN [53]	52.9 $\pm$ 6.2	24.2 $\pm$ 3.3	35.1 $\pm$ 4.6
CLASTER	69.1 $\pm$ 5.4	48.0 $\pm$ 2.4	51.3 $\pm$ 3.5
SDR (Ours)+I3D	**74.5** $\pm$ **3.9**	**49.7** $\pm$ **2.9**	**54.9** $\pm$ **4.4**
SDR (Ours)+CLIP	**79.5** $\pm$ **3.5**	**53.5** $\pm$ **3.3**	**57.8** $\pm$ **4.1**
SDR (Ours) SM+I3D	**76.6** $\pm$ **3.6**	**50.9** $\pm$ **2.6**	**57.2** $\pm$ **3.5**
SDR (Ours) SM+CLIP	**81.1** $\pm$ **3.0**	**56.1** $\pm$ **3.2**	**59.7** $\pm$ **3.1**

Table 7.5: Results on generalized zero-shot setting. Reported results are the harmonic mean of the seen and unseen class accuracies. 'SM' corresponds to the single model experiment.

In order to better analyze performance of the model on GZSL, we report the average seen and unseen accuracies along with their harmonic mean. The results using different embeddings and on the UCF101, HMDB51 and Olympics datasets are reported in Table 7.6. The reported results are on the same set of 10 random splits for fair compairson. There are no manual attributes for the HMDB dataset. We see that the proposed SDR approach obtains best results on all three categories. Another observation we can see is that the performance of all models using *Stories* is better than even the older manual attributes.

Model	SE	Olympics			HMDB51			UCF-101		
		u	s	H	u	s	H	u	s	H
WGAN [185]	M	50.8	71.4	59.4	-	-	-	30.4	**83.6**	44.6
OD [116]	M	61.8	71.1	66.1	-	-	-	36.2	76.1	49.1
CLASTER [64]	M	66.2	71.7	68.8	-	-	-	40.2	69.4	50.9
SDR	M	**71.6**	**76.9**	**74.2**	-	-	-	**43.1**	77.5	**54.6**
WGAN [185]	W	35.4	65.6	46.0	23.1	55.1	32.5	20.6	73.9	32.2
OD [116]	W	41.3	72.5	52.6	25.9	55.8	35.4	25.3	74.1	37.7
CLASTER	W	49.2	71.1	58.1	35.5	52.8	42.4	30.4	68.9	42.1
WGAN [185]	S	36.1	66.2	46.7	28.6	57.8	38.2	27.5	74.7	40.2
OD [116]	S	42.9	73.5	54.1	33.4	57.8	42.3	32.7	75.9	45.7
CLASTER	S	49.9	71.3	58.7	42.7	53.2	47.4	36.9	69.8	48.3
CLASTER	C	66.8	71.6	69.1	43.7	53.3	48.0	40.8	69.3	51.3
WGAN [185]	Sto	52.5	73.4	61.2	35.2	65.1	45.7	33.8	**84.2**	48.2
OD [116]	Sto	63.3	75.1	68.7	37.2	**67.5**	47.9	40.1	81.7	53.8
CLASTER	Sto	69.1	74.1	71.5	44.3	57.2	49.9	42.1	71.5	53.0
SDR+I3D	Sto	73.5	79.9	76.6	46.9	55.8	50.9	44.4	80.7	57.2
SDR+CLIP	Sto	**78.9**	**83.5**	**81.1**	**52.5**	60.4	**56.1**	**47.3**	81.2	**59.7**

Table 7.6: Seen and unseen accuracies for CLASTER on different datasets using different embeddings. 'SE' corresponds to the type of embedding used, wherein 'M', 'W', 'S', 'C' and 'Sto' refers to manual annotations, word2vec, sen2vec, combination of the embeddings and Stories respectively. 'u', 's' and 'H' corresponds to average unseen accuracy, average seen accuracy and the harmonic mean of the two. All the reported results are on the same splits. SDR+I3D corresponds to the backbone network being I3D and similarly for SDR+CLIP.

7.4.7 Exploring the Few-Shot Setting

Although it is not our goal, we explore the generalization ability of the proposed approach in the few-shot setting. In this setting we have a few examples of the unseen classes at train time. We use the same approach as for the zero-shot learning, except to train the classifier we also add the few labeled samples provided. Another approach is to use the generated features as additional data to train any of the current few-shot learning state-of-the-art models.

We compare to the current state-of-the-art in this setting, including CD3-PN [155], ARN [209], TRX [133], MTFAN [180] and HCL [219] on the UCF101, HMDB51 and Kinetics datasets. To the best of our knowledge GGM [122] is the only generative few-shot action recognition paper and we compare against them directly and see very large improvements. However, other approaches use meta-learning and few-shot specific learning paradigms and hence we use our generated features as additional data without changing their learning paradigm. We obtain state-of-the-art results on

	UCF101			HMDB51		
Method	1	3	5	1	3	5
GGM [122]	62.5	73.5	78.7	36.5	47.5	52.6
SDR (Ours)	**74.1**	**81.9**	**85.3**	**43.8**	**54.8**	**58.8**
ARN [209]	62.1	77.9	84.8	44.6	54.2	59.1
C3D-PN [155]	57.8	71.7	80.2	50.3	54.0	57.4
MTFAN [180]	**84.8**	-	95.1	59.0	-	74.6
HCL [219]	82.6	-	94.5	**59.1**	-	76.3
TRX [133]	81.3	92.8	95.9	52.0	66.9	75.6
TRX [133] + **SDR (Ours)**	84.1	**95.2**	**97.3**	58.0	**72.2**	**79.1**

Table 7.7: Results on the few-shot setting, using HMDB51 and UCF101.

the 3-shot and 5-shot setting and significant improvement on the 1-shot setting when training TRX with our features. However, MTFAN [180] and HCL [219] have not released their code yet and hence we can not check how much we could potentially improve.

7.5 Studying the Effect of Stories as a General Semantic Embedding

We look at the use of *Stories* as semantic embedding to a wide variety of models here. From older models, all the way to the most recent ones in the zero-shot learning literature. Here we look at the experimental results on a wide variety of models in all three datasets. We see that *Stories* is clearly model agnostic, old or new models improve using it. Results can be seen in Tab 7.8.

7.6 Why Not Just Use VAE for Feature Generation?

Another possible question is the use of the current feature generator model. There are multiple options to use as feature generators including VAEs, and other versions of GANs (not just WGAN [185] that we use).

Method	SE	Olympics	HMDB51	UCF101
SJE [2]	W	28.6 ± 4.9	13.3 ± 2.4	9.9 ± 1.4
SJE [2]	M	47.5 ± 14.8	-	12.0 ± 1.2
SJE [2]	Sto	**50.3 ± 4.1**	**19.1 ± 3.2**	**14.5 ± 1.6**
Bi-Dir GAN [121]	W	40.2 ± 10.6	21.3 ± 3.2	21.8 ± 3.6
Bi-Dir GAN [121]	M	53.2 ± 10.5	-	24.7 ± 3.7
Bi-Dir GAN [121]	Sto	**55.5 ± 6.5**	**27.2 ± 2.7**	**29.5 ± 3.2**
GGM [122]	W	41.3 ± 11.4	20.7 ± 3.1	20.3 ± 1.9
GGM [122]	M	57.9 ± 14.1	-	24.5 ± 2.9
GGM [122]	Sto	**59.7 ± 12.1**	**29.5 ± 2.6**	**27.1 ± 2.6**
WGAN [185]	W	47.1 ± 6.4	29.1 ± 3.8	25.8 ± 3.2
WGAN [185]	M	64.7 ± 7.5	-	37.5 ± 3.1
WGAN [185]	Sto	**66.2 ± 7.1**	**35.5 ± 2.8**	**40.1 ± 3.7**
OD [116]	W	50.5 ± 6.9	30.2 ± 2.7	26.9 ± 2.8
OD [116]	M	65.9 ± 8.1	-	38.3 ± 3.0
OD [116]	Sto	**69.1 ± 5.6**	**39.2 ± 2.8**	**50.3 ± 3.0**
CLASTER [64]	W	63.8 ± 5.7	36.6 ± 4.6	46.7 ± 5.4
CLASTER [64]	M	67.4 ± 7.8	-	51.8 ± 2.8
CLASTER [64]	Sto	**73.1 ± 6.6**	**45.5 ± 2.6**	**59.6 ± 2.8**

Table 7.8: Results on ZSL. SE: semantic embedding, M: manual representation, W: word2vec embedding, S: sentence2vec, Sto: Stories.

Feature Generator	Accuracy
VAE	25.5 ± 2.9
Vanilla GAN	31.5 ± 2.4
f-VAEGAN	45.9 ± 3.2
FREE	46.6 ± 3.5
SDR (Ours)	**48.1 ± 3.6**

Table 7.9: Comparing different choices for feature generator. Reported results are on 10 different runs and all models use the same split. Dataset is HMDB51.

We chose to adapt the WGAN for our feature generator based on two reasons. First, we wanted to compare directly to existing literature on zero-shot action recognition and to the best of our knowledge the most recent one has been the one used in OD [116]. However, for the sake of sanity we also ran additional experiments on the HMDB51 dataset incorporating f-VAEGAN [187], adapted FREE [24]: feature refinement of f-VAEGAN for zero-shot action recognition and using a simple VAE. The results of this can be seen in Tab 7.9.

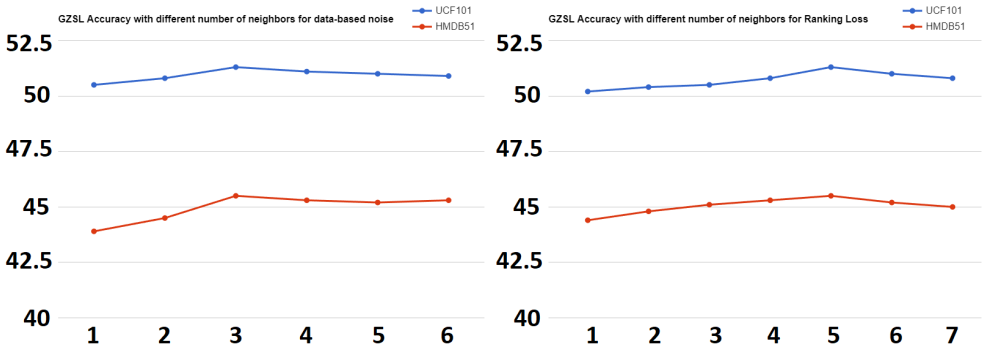


Figure 7.8: Comparison of using different number of nearest neighbors on both (left) the data-based noise and (right) the ranking loss.

7.7 Hyperparameter Selection

Choosing the number of nearest neighbors for both the data-based noise and the ranking loss is done empirically. We use the generalized zero-shot action recognition performance to decide these hyperparameters. We choose UCF101 as our dataset for the hyperparameter tuning, but also plot the results on HMDB51 as it also ended up following the same pattern. The results are shown in Figure 7.8. Based on these results, we choose the number of nearest neighbors as 3 for the data-based noise and 5 for the ranking loss. The results are on the TruZe split.

7.8 Why Does Using a Single Model Work?

One curious question to ponder would be why the single model trained on a large dataset like Kinetics-400 [20] results in better performance than the models fine-tuned on the smaller datasets. Our hypothesis is that the feature generator trained on a larger dataset has a better distribution of data to learn from as the data-driven noise that we use is more representative of the real visual world. This in turn generates more realistically distributed features, which in turn results in the improved performance.

7.9 Limitations

One possible limitation of our approach is that the Stories may focus on one specific way of performing each action, while other valid methods may exist. For example, the story for "shuffling cards" details the riffle shuffling technique, but other shuffling techniques could occur in videos of this class. Similarly, some parts of the stories may describe non-visual aspects like memorizing lines for the "acting in play" class that are not depicted in the videos. Our current approach does not explicitly address potential mismatches between the textual stories and visual contents. Still, we believe non-visual cues actually help making the semantic embeddings more distinct, as these cues are unique to each class's story. However, this remains a limitation worth noting. One area of future work is exploring how to make the stories more comprehensive by incorporating multiple variations of actions. We also plan to investigate techniques for identifying and excluding non-visual sentences that do not translate to visual features. Overall, handling the diversity of real videos compared to procedural descriptions remains an open challenge that we aim to address.

7.10 Conclusion

We propose a method called SDR, which makes three contributions for zero-shot action recognition. The main one is a more meaningful semantic space, which we call *Stories*, which a textual database of steps needed to perform the action. This novel semantic embedding helps significantly across models, tasks, and datasets. We also introduce a novel ranking loss and data-driven noise to improve performance and cut back convergence time to one third of the vanilla method. All these combined show improvements across datasets and tasks with improvements up to 13.8%. Overall, we hope these contributions can be helpful for future progress on low-shot action recognition.

Chapter 8

Selecting Informative Synthetic Features using Reinforcement Learning for Improved Generalized Zero-Shot Learning

8.1 Introduction

In Generalized ZSL, classifiers tend to be biased towards seen classes, leading to the misclassification of test samples from unseen classes as seen classes. To address the problem of the lack of visual data for unseen classes, researchers have proposed the use of Generative Adversarial Networks (GAN) [56] to generate synthetic visual features by leveraging attribute databases. However, while GANs have helped in zero-shot learning [170, 185, 201, 45], they do not explicitly learn to represent data in a structured way that is easily interpretable by humans or other models. They also suffer from the problems of mode collapse [81], class imbalance [6], and computational expense [71] when generating high-dimensional data. But what we care about most is that a lot of synthetic samples generated are used directly for training a classifier without studying if these samples actually help the classifier learn. Instead, these samples are chosen based on “realness". Figure 8.1 shows a comparison of the standard pipeline and our proposed pipeline for feature-generating approaches.

To address the limitations of GANs in synthetic feature selection, we propose a novel reinforcement learning-based approach that automatically selects generated features that improve model performance. Specifically, we use a transformer model [169] for synthetic sample selection and use validation classification accuracy as the reward for RL training. We employ the proximal policy optimization (PPO) [148] algorithm to

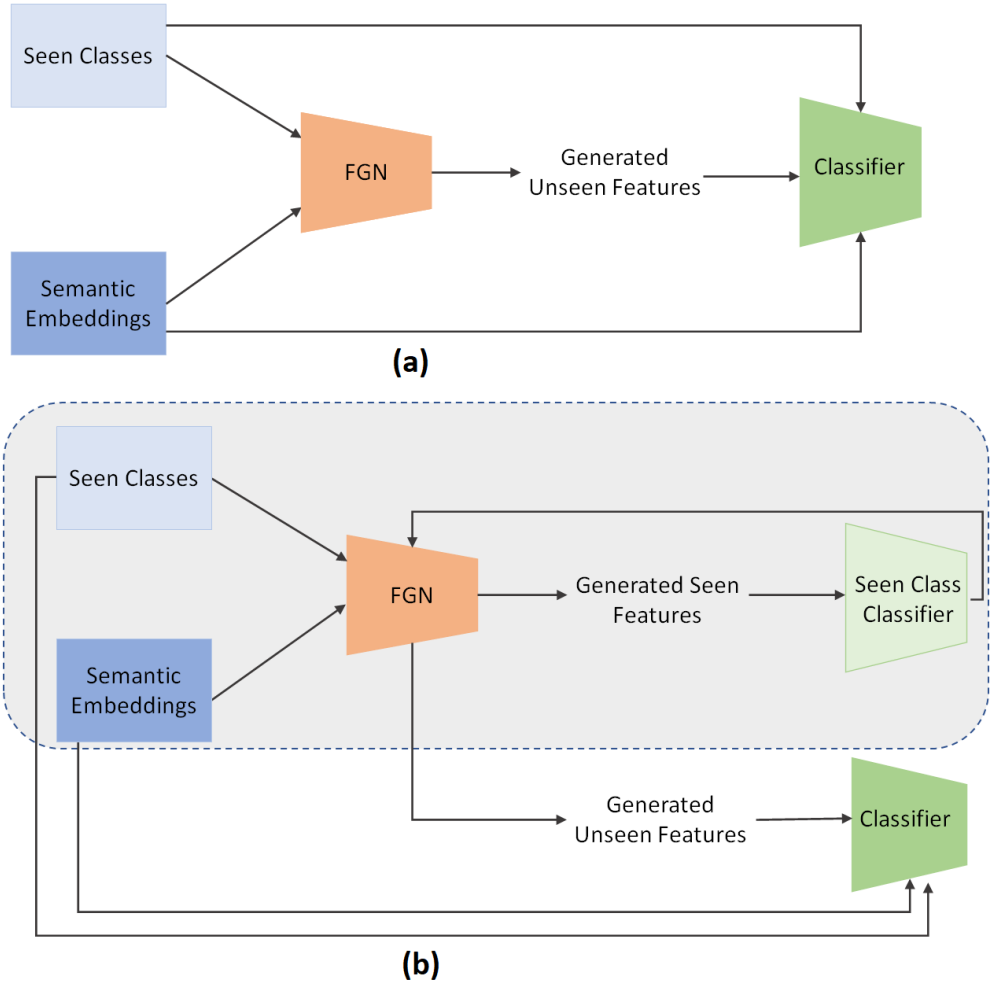


Figure 8.1: Comparison of feature-generating frameworks pipelines (a) standard pipeline where features that look “real" are used to train the generator (b) proposed pipeline where the generator is trained based on the performance of the seen class classifier.

update the model weights during training. The validation set is a subset of the original training set. Our proposed approach aims to pick samples that help classification and not just generate real-looking samples. We dub our synthetic sample selection method as “**SPOT**" for **S**election using **P**roximal policy **O**p**T**imization. The goal of this selector is to learn to pick synthetically generated samples that it believes would improve the classification performance of the classifier rather than picking the most “realistic" looking samples.

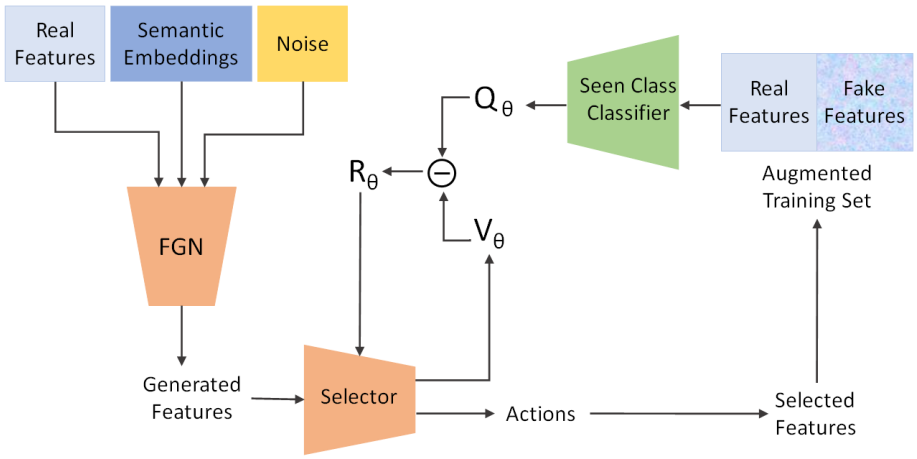


Figure 8.2: Overall pipeline of our proposed SPOT. The feature generator generates features that the selector module ranks based on the seen class classifier's performance. The selector is updated based on the performance of the classifier on the selected features. The proposed pipeline is model and data-agnostic.

Furthermore, our proposed approach is model-agnostic and data-agnostic, as we evaluate our method on multiple benchmark datasets in images and videos and various feature-generating models. Our comprehensive experiments demonstrate that our approach consistently improves model performance across different datasets and models, highlighting the effectiveness and versatility of our proposed method. By leveraging RL-based synthetic feature selection, we can more effectively generate synthetic data that captures the underlying structure of the data, improving the generalization performance of downstream models.

8.2 Methodology

The overall framework of the proposed method is visually depicted in Figure 8.2, which provides an illustrative overview of the various components employed. In this section, we delve deeper into the individual constituents of the model and explore in detail the novel SPOT selector that has been put forth. It is imperative to note that the proposed pipeline is model and data-agnostic, which implies that the choice of classifier model and network backbone is dependent solely on the feature-generating framework itself.

The development of the SPOT selector draws significant inspiration from the synthetic sample selector introduced in [197]. However, our approach seeks to tackle the more challenging task of zero-shot learning, where data from unseen classes is limited. Furthermore, we demonstrate that training the selector using data from seen classes can enhance the selection of better features for data from unseen classes. This approach represents a significant contribution and serves to bridge the gap between the seen and unseen class data.

8.2.1 Feature Generating Network (FGN)

As previously highlighted, it is worth noting that the proposed pipeline is entirely independent of the feature-generating approach itself. This aspect renders the framework highly versatile and adaptable to a diverse range of feature-generating models, which may be employed in place of the FGN utilized in this study.

Examples of alternate feature-generating models that could be employed include the WGAN [185] and Cycle-WGAN [45]. The utilization of such models would permit the proposed pipeline to be seamlessly integrated into a broader range of applications and extend the reach and scope of the framework. The flexibility afforded by this design decision represents a critical feature of the proposed pipeline and enables the framework to be readily adapted to a range of diverse use cases as shown with consistent improvements in multiple image and video datasets.

8.2.2 Selector

The reasoning behind selecting the particular selector is rooted in the interdependence of features among the potential images. We hypothesise that the sequence in which the features are generated is not completely autonomous, as the later additions must differentiate themselves from the earlier ones in order to ensure diversity across the entire set of augmented training data. We use a transformer-based architecture [36] to be our selector. The input is a feature vector of a dimension dependent on the FGN used. The goal of the selector is to tell us if the generated feature vector is good for classification performance. The selector takes in a feature vector and outputs a score that tells us how good that feature vector is for classification. To do this, the selector outputs a binary action: select or not select. However, we do not have ground truth to tell us how good the generated feature is and hence optimizing the selector is not trivial.

To address the possibility of a relationship between augmented images without relying heavily on sequential assumptions, we utilize the self-attention mechanism through the implementation of the transformer [36] model as our selector. The transformer architecture eliminates all recurrent structures, requiring feature vectors to be combined with their positional embeddings using sinusoidal functions prior to being input into the encoder layer of the transformer. The primary component of the transformer encoder is the multi-head attention block, comprised of n self-attention layers, where n denotes the number of heads. In each self-attention layer, input features are projected to three separate feature spaces - query Q, key K, and value V - by multiplying learnable weight matrices. The resulting attention map is obtained through the following process:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{8.1}$$

Each head of the multi-head attention block represents a distinct projected feature space for the input, achieved by multiplying the same input embedding with different weight matrices. These separate outputs are then concatenated to form the final attention map, which is expressed as:

$$MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O \tag{8.2}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{8.3}$$

Where, $W^Q \in \mathbb{R}^{d_{\text{input}} \times d_k}, W^K \in \mathbb{R}^{d_{\text{input}} \times d_k}, W^V \in \mathbb{R}^{d_{\text{input}} \times d_v}, W^O \in \mathbb{R}^{hd_v \times d_{\text{input}}}$ and h represents the number of heads.

Once we obtain the attention map, the context vector is then fed to the feed-forward layer as follows:

$$F(x) = max(((0, xW_1 + b_1)W_2 + b_2)W_3 + b_3...)W_n + b_n \tag{8.4}$$

Given that the objective of the selector is to produce a binary action for every input feature vector, the decoder for the transformer model is a linear layer that functions as the policy network. Overall, the use of the transformer as the selector within our image selection framework, based on reinforcement learning, is beneficial due to its self-attention mechanism that effectively captures the interdependencies among the input feature vectors. We conducted a thorough ablation study regarding the selection of the selector and this can be seen in Section 4.2.

To optimize the selector, we turn to reinforcement learning as this is a common solution [200, 61]. In particular, we use proximal policy optimization [148]. Details are explained in the next section.

8.2.3 Proximal Policy Optimization

As previously stated, we turn to reinforcement learning approach to update the selector model. A proficient policy gradient method is fundamental to effectively utilize reward feedback as input to the selector in the reinforcement learning process. Among various policy gradient algorithms, Proximal Policy Optimization (PPO) [148] has gained popularity due to its computational efficiency and satisfactory performance, surpassing previous approaches like TRPO [147]. Additionally, PPO alleviates the instability encountered during RL training. PPO achieves comparable performance with reduced complexity by replacing the KL convergence constraint enforced in TRPO with a clipped probability ratio between the current and previous policy within a small interval around 1. At each time step t, with A_θ representing the advantage function, the objective function is defined as follows:

$$L(\theta) = E[min(\gamma_\theta(t)A_\theta(s_t,a_t), clip(\gamma_\theta(t), 1-\varepsilon, 1+\varepsilon)A_\theta(s_t,a_t))] \tag{8.5}$$

Here, $A_\theta(s_t,a_t) = Q_\theta(s_t,a_t) - V_\theta(s_t,a_t)$. As a component of the transformer output, the learned state-value $V_\theta(s_t,a_t)$ serves as a baseline for the q-value to mitigate the variance of rewards during the training process. The probability of actions is denoted by π. The q-value at time t, $Q_\theta(s_t,a_t)$, is defined as a smoothed version of the maximum validation accuracy observed among the last five epochs in the classification task. $\gamma_\theta(t)$ is the probability ratio between the previous and current policies. As our target tasks are trained on the seen class data and needs to generalize to the

unseen class data, it is crucial to obtain a robust estimation of the reward's changing pattern. To achieve this, we employ the Exponential Moving Average (EMA) algorithm to smooth the original reward curve. Thus, the final reward at time t is obtained as follows:

$$\hat{Q}_\theta(s_t, a_t) = \begin{cases} \mathrm{Q}_\theta(s_t, a_t), & t = 1 \\ \alpha \hat{Q}_\theta(s_{t-1}, a_{t-1}) + & \\ (1-\alpha)\mathrm{Q}_\theta(s_t, a_t), & t > 1 \end{cases}. \tag{8.6}$$

Drawing inspiration from the concept of importance sampling, the weight assigned to the current policy is influenced by earlier policies. The probability ratio between the previous and current policies, denoted by $\gamma_\theta(t)$, is mathematically defined as:

$$\gamma_\theta(t) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_\theta(a_{t-1} \mid s_{t-1})} \tag{8.7}$$

Here, $a_t \in \mathbb{R}^{N \times 2}$ refers to the number of synthetic samples in the candidate pool. If at any given timestep t, $a_i(t) = 0$ then i is discarded. Else, it is added to the original training set.

8.3 Experimental Analysis

8.3.1 Implementation Details

Since we propose a plug-and-play component to feature-generating networks, the backbone and technical details follow the exact same implementation that the feature-generating model uses. Here, we talk about the technical details of running SPOT.

The candidate features generated by the feature-generating framework are passed into the selector network which is a 8-layer encoder having a 8-head multi-head attention block. The output is a vector A_θ which is the action vector and a value vector V_θ that is used together to calculate the reward for the policy gradient algorithm.

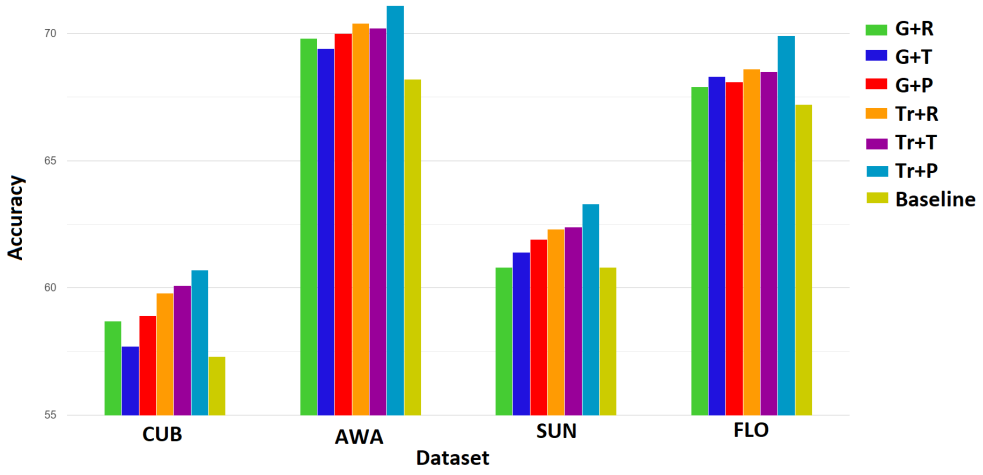


Figure 8.3: Ablation comparison of different combinations of RL algorithms with selector choices (GRU or Transformer). 'G' = GRU, 'Tr' = Transformer, 'R' = REINFORCE, 'T' = TRPO and 'P' = PPO.

The classifier network depends on the feature-generating network being used since our proposed method is model agnostic. Similar to [61, 197] we use the EMA-smoothed validation accuracy obtained from the last 5 epochs as the reward with α=0.5 when using the classifier on the validation set. As long as this average is increasing, we continue updating our policy. The policy function π is obtained from the softmax layer of the seen-class classifier model.

ε in Eq. 4 is set to 0.15 (see Ablations for empirical comparison), this helps to set an upper and lower bound at the current time step t and previous one $t-1$ for the ratio of the policy function. The number of selected synthetic features are dependent on the selector and varies according to model and dataset. However, we set the learning rate to be fixed for the PPO at 2e - 04.

8.3.2 Ablation Study

We have made a few choices with regards to the hyperparameter selection and choice of RL optimization algorithms and in this section, we show empirical reasons why the choices were made. Figure 8.3 shows the performance differences when using different RL optimization algorithms to modify the selector. Similar to [197], we also consider alternative choices such as GRU and LSTMs for the selector. In terms of RL algorithms, we compare to REINFORCE [179] and TRPO [147].

Model	CUB	AWA	SUN	FLO
WGAN	57.3	68.2	60.8	67.2
WGAN + **SPOT**	**60.7**	**71.1**	**63.3**	**69.9**
Cycle-WGAN	57.8	65.6	59.7	68.6
Cycle-WGAN + **SPOT**	**61.1**	**69.7**	**62.5**	**70.9**
f-VAEGAN	61.0	71.1	64.7	67.7
f-VAEGAN + **SPOT**	**62.8**	**72.7**	**66.0**	**69.2**
CMC-GAN	61.4	71.4	63.7	69.8
CMC-GAN + **SPOT**	**62.9**	**73.1**	**65.1**	**71.9**

Table 8.1: Results on zero-shot image classification using recent feature-generating frameworks.

8.3.3 Images

Model	CUB			AWA1			SUN			FLO		
	S	U	H	S	U	H	S	U	H	S	U	H
WGAN	43.7	57.7	49.7	57.9	61.4	59.6	42.6	36.6	39.4	59.0	73.8	65.6
WGAN+**SPOT**	**44.1**	**60.9**	**51.1**	**58.6**	**64.9**	**61.6**	**42.8**	**39.1**	**40.9**	**59.3**	**75.9**	**66.6**
Cycle-WGAN	46.0	60.3	52.2	56.4	63.5	59.7	**48.3**	33.1	39.2	59.1	71.1	64.5
Cycle-WGAN+**SPOT**	**46.5**	**62.9**	**53.5**	**56.9**	**66.1**	**61.1**	48.1	**36.2**	**41.3**	**59.4**	**74.4**	**66.1**
f-VAEGAN	48.4	60.1	53.6	57.6	70.6	63.5	45.1	38.0	41.3	56.8	74.9	64.6
f-VAEGAN+**SPOT**	**48.8**	**62.8**	**54.9**	**57.9**	**73.3**	**64.7**	**45.5**	**41.1**	**43.2**	**57.0**	**77.2**	**65.6**
CMC-GAN	52.6	65.1	58.2	63.2	70.6	66.7	48.2	40.8	44.2	64.5	80.2	71.5
CMC-GAN+**SPOT**	**53.1**	**66.7**	**59.1**	**63.3**	**73.8**	**68.1**	**48.9**	**44.1**	**46.4**	**64.6**	**82.8**	**72.6**
NereNET	51.0	56.5	53.6	-	-	-	45.7	38.1	41.6	-	-	-
NereNET+**SPOT**	**51.3**	**58.4**	**54.6**	-	-	-	**45.9**	**40.4**	**43.0**	-	-	-
FREE	**55.7**	59.9	57.7	62.9	69.4	66.0	47.4	37.2	41.7	67.4	84.5	75.0
FREE+**SPOT**	55.5	**62.2**	**58.6**	**63.1**	**72.1**	**67.3**	**47.8**	**39.9**	**43.5**	**67.8**	**86.3**	**75.9**
DAA	66.1	65.5	65.8	64.3	76.6	69.9	47.8	38.7	42.8	-	-	-
DAA+**SPOT**	**66.3**	**67.7**	**67.0**	**64.6**	**77.9**	**70.6**	**48.1**	**40.3**	**43.8**	-	-	-

Table 8.2: Results on generalized zero-shot image classification on 4 challenging benchmarks.

Datasets and Evaluation Protocol

Our method is evaluated on four challenging benchmark datasets, namely AWA [94], CUB (Caltech UCSD Birds 200) [171], SUN (SUN Attribute) [188], and FLO [128]. CUB and SUN are fine-grained datasets, while AWA and FLO are coarse-grained datasets. We adopt the same seen/unseen splits and class embeddings to ensure consistency with previous work as in [186]. AWA1 contains 30,475 instances across 50 categories. CUB comprises 11,788 images of 200 bird classes (150/50 for seen/unseen classes) with 312 attributes. SUN contains 14,340 images from 717 scene classes

(645/72 for seen/unseen classes) with 102 attributes. FLO consists of 8,189 images from 102 flower classes with an 82/20 class split for seen and unseen classes, respectively. These datasets are widely used in the literature, enabling a direct comparison of our results with those of previous studies.

Zero-Shot Learning

We compare strictly with recent state-of-the-art feature-generating approaches and as such compare to WGAN [185], Cycle-WGAN [45], f-VAEGAN [187], NereNet [108] and CMC-GAN [195]. Table 8.1 shows the results. We see consistent gains with increase of up to 4.1% when we add the proposed SPOT to any of the models.

Generalized Zero-Shot Learning

We perform a much more extensive comparison in the generalized setting as this is where most feature generating frameworks perform experiments. We compare against WGAN [185], Cycle-WGAN [45], f-VAEGAN [187], CMC-GAN [195], NereNet [108], FREE [24] and DAA [217]. We see consistent improvements on the unseen class accuracies as this is where selected features make a difference. We see gains of up to 3.3% on the unseen class accuracies. As a result, there is consistent improvement on the harmonic mean of the seen and unseen class accuracies as well.

8.3.4 Videos

Datasets and Evaluation Protocol

For videos, we use the widely adopted Olympic Sports [127], HMDB-51 [90], and UCF-101 [157] datasets to evaluate our method for zero-shot action recognition and compare it against recent state-of-the-art feature generating models [116, 185, 76]. The aforementioned datasets comprise 783, 6766, and 13320 videos and are associated with 16, 51, and 101 classes, respectively. To enable comparison with existing works [116, 185, 76, 122, 121], we adopt the widely used 50/50 splits proposed by Xu et al. [191], where half of the classes are considered as seen and the other half as unseen. We report the average accuracy and standard deviation over 10 independent runs, following previous approaches.

Method	Olympics	HMDB51	UCF101
Bi-Dir GAN [121]	53.2 ± 10.5	21.3 ± 3.2	24.7 ± 3.7
Bi-Dir GAN [121] + **SPOT**	**56.6** ± **10.1**	**25.1** ± **3.4**	**27.7** ± **3.5**
GGM [122]	57.9 ± 14.1	20.7 ± 3.1	24.5 ± 2.9
GGM [122] + **SPOT**	**62.4** ± **12.4**	**25.1** ± **2.8**	**27.4** ± **2.5**
OD [116]	65.9 ± 8.1	30.2 ± 2.7	38.3 ± 3.0
OD [116] + **SPOT**	**68.7** ± **7.5**	**34.4** ± **2.2**	**40.9** ± **2.6**
WGAN [185]	64.7 ± 7.5	29.1 ± 3.8	37.5 ± 3.1
WGAN [185] + **SPOT**	**68.1** ± **7.1**	**33.8** ± **2.4**	**40.6** ± **2.4**
FFG [76]	-	32.4 ± 2.3	27.6 ± 2.4
FFG [76] + **SPOT**	-	**35.9** ± **2.5**	**30.9** ± **2.2**

Table 8.3: Results on zero-shot action recognition on the Olympics, HMDB51 and UCF101 datasets.

Moreover, we extend our evaluation to include TruZe [65], which was recently introduced to address the issue of overlapping classes between the pre-training dataset (Kinetics [20]) and the unseen classes in zero-shot settings. The TruZe split acknowledges the presence of such overlapping classes, which contradicts the fundamental assumption that the unseen classes have not been previously seen.

Zero-Shot Learning

Table 8.3 shows the effect of using our proposed SPOT selector to enhance the performance of state-of-the-art feature-generating frameworks on the zero-shot setting. We compare with the most recent best-performing methods, which include the Bi-Dir GAN [121], GGM [122], OD [116], WGAN [185] and FFG [76] (fine-grained feature generation framework).

We observe that the proposed method consistently outperforms all approaches across all datasets by gains of up to 4.5%.

Generalized Zero-Shot Learning

We evaluate SPOT on the generalized setting where at test time both seen and unseen class samples are used. Table 8.4 shows the results, with the harmonic mean of the seen and unseen class accuracies. The proposed SPOT selector consistently improves all approaches across all datasets by gains of up to 4.2%.

Method	Olympics	HMDB51	UCF101
Bi-Dir GAN [121]	44.2 ± 11.2	7.5 ± 2.4	22.7 ± 2.5
Bi-Dir GAN [121] + **SPOT**	**48.4** ± **10.3**	**10.9** ± **3.1**	**25.1** ± **3.8**
GGM [122]	52.4 ± 12.2	20.1 ± 2.1	23.7 ± 1.2
GGM [122] + **SPOT**	**55.3** ± **11.9**	**23.4** ± **2.3**	**27.1** ± **3.1**
WGAN [185]	59.9 ± 5.3	32.7 ± 3.4	44.4 ± 3.0
WGAN [185] + **SPOT**	**62.4** ± **5.5**	**34.4** ± **2.9**	**46.2** ± **2.6**
OD[116]	66.2 ± 6.3	36.1 ± 2.2	49.4 ± 2.4
OD [116] + **SPOT**	**69.1** ± **6.5**	**38.2** ± **2.5**	**51.8** ± **2.5**
FFG [76]	-	37.4 ± 1.9	40.4 ± 2.2
FFG [76] + **SPOT**	-	**39.8** ± **1.4**	**42.8** ± **1.7**

Table 8.4: Results on generalized zero-shot setting. Reported results are the harmonic mean of the seen and unseen class accuracies.

Method	UCF101		HMDB51	
	ZSL	GZSL	ZSL	GZSL
WGAN	22.5	36.3	21.1	31.8
WGAN + **SPOT**	**25.3**	**39.1**	**23.8**	**33.3**
OD	22.9	42.4	21.7	35.5
OD + **SPOT**	**25.5**	**44.1**	**24.0**	**37.1**

Table 8.5: Results on TruZe. We report the mean class accuracy for zero-shot; for generalized zero-shot, we report the harmonic mean of seen and unseen class accuracies.

8.3.5 Results on TruZe

We also evaluate on the stricter TruZe [65] split that ensures no overlap between the pre-trained model and test classes. Results are shown in Table 8.5. We only evaluate on OD and WGAN as these are the two feature-generating approaches that have results reported on the TruZe split. Again, we see that using SPOT for selection consistently improves the performance of the feature-generating framework.

8.4 Limitations

While SPOT demonstrates improved performance over prior generative models, it has some limitations. First, it requires access to seen class data which may not always be available. Second, the selector model itself introduces additional complexity. More recent works [206, 101] take a different approach by learning a parameterized Mahalanobis distance in the VAEGAN framework, improving model structure to distinguish between seen and unseen samples. These avoid SPOT's dependence on

seen classes. However, SPOT's core idea of selecting synthetic samples that directly optimize the end goal remains relevant. Follow-up works [100, 8] adopt a similar strategy but on text-based zero-shot learning. So while later methods improve over SPOT, its insight on selection criteria is still influential.

8.5 Conclusion

In conclusion, although generative techniques have made significant progress in transforming traditional GZSL to fully supervised learning, they often generate redundant synthetic features, which can lead to reduced accuracy. To overcome this limitation, we have proposed an approach for synthetic feature selection using reinforcement learning, which involves training a transformer-based selector using proximal policy optimization (PPO) to select synthetic features based on the validation classification accuracy of seen classes as the reward. Our proposed method is model-agnostic and data-agnostic and hence is suitable for images and videos. The experimental results of our approach demonstrate its superiority over existing feature-generating methods, with improved overall performance observed across multiple benchmarks. Overall, our approach represents a significant contribution towards addressing the issue of synthetic feature redundancy in GZSL, and we believe that it has the potential to be widely applied in real-world scenarios.

Chapter 9

Learning to Augment Video Datasets by Compositing Quality Videos for Action Recognition

9.1 Introduction

Recent efforts in video focus on relieving the strong dependency of current methods to the size of labeled datasets. Some of these efforts [160, 223] involve increasing the number of data samples through data augmentation. This strategy aims to create new videos in the training set by performing transformations on the original annotated videos, where labels are known. This process adds diversity to the training data, while new videos are still realistic and plausible. In the simplest version of data augmentation in video, new data samples are generated by flipping the input video horizontally, or by cropping a subsection of the video. New methods [223, 204] propose more sophisticated processes like combining two videos. VideoMix [204] randomly crops regions of one video and pastes them onto another. ActorCut [223] goes one step further and uses the bounding box detections of humans on one video to paste them onto the background of another video. This increases the diversity of the new videos, and despite the lack of visual realism of the resulting videos, this strategy helps.

However, as datasets become larger, such data augmentation strategies become computationally expensive. The search space of possible video pairs and transformations is enormous and difficult to explore. The solution is often to sample the space randomly, or to manually design augmentation heuristics. Any exploration process is particularly burdening in the context of video data, where the augmentation process needs to be repeated in every frame, which may be orders of magnitude more expensive than for images.

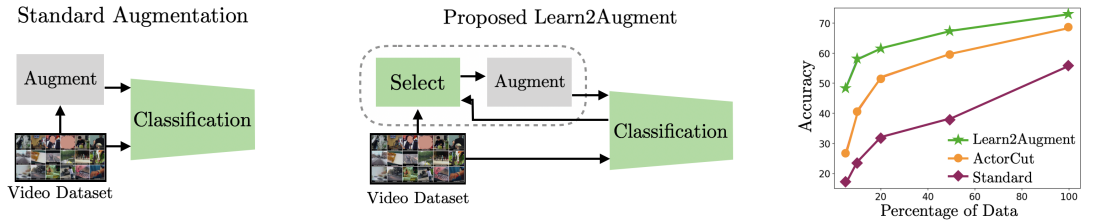


Figure 9.1: Standard video augmentation techniques generate data using hand-designed heuristics (left). We propose to learn to select videos for augmentation, based on how effective they will be for learning to classify (middle). Our approach, Learn2Augment, improves classification across datasets and settings, including UCF101 (right).

In this chapter, we address the problem of sampling for data augmentation, and propose to learn to select pairs of videos. We show that this reduces the search space of augmented data points by orders of magnitude and improves the final accuracy of the classifier significantly. We leverage two observations. First, not all data points are as useful for classification. This idea has been exploited in the context of frame or clip selection [62, 89, 77]. Second, we can learn to predict which data points will be useful without actually generating them. This is essential, as the space of transformations is huge, and if we needed to create each candidate augmented video, the process would be prohibitively expensive.

More concretely, we propose a data augmentation method which we call Learn2Augment. The proposed method contains a “Selector” network, which predicts a score of how useful a combination of two videos will be, without having to actually composite them. The Selector is trained using the accuracy of the classification as the cue. Since this metric depends on the classifier, it is not differentiable with respect to the Selector’s parameters. Therefore we optimize the network using reinforcement learning. Once the Selector network is trained, we use it to choose good pairs of videos, composite them, and train a classification network. In our experiments, for example in the case of the UCF101 dataset, using the Selector reduces the number of augmented videos by 92% while increasing the classification accuracy.

In the proposed method, each augmented video is created from a pair of videos using a composition of the segmented foreground of one video, including actor and objects, onto the background of the other video. This process yields diverse and realistic new data samples, which we demonstrate is important for learning. More concretely, results show an improvement of 4.4% over using a simpler transformation.

The Selector is indeed useful to reduce the number of videos for training the classifier. However, we also need to reduce the space of possible pairs for training the Selector network itself. For example, the number of possible pairs of videos in video datasets can be in the order of millions for small datasets or billions for large datasets. For this, we leverage the natural correlation between the occurrence of foreground activities and background scenes [25]. This is, it is more likely to find someone playing football in a football field than at a restaurant. Instead of sampling at random the pairs of videos to train the Selector on, we sample pairs from classes that are semantically similar. In particular, we use the class names to obtain a semantic embedding, and match each class to their nearest neighbor in this space. Experiments show that this extremely simple design choice of Semantic Matching reduces the space of possible pairs of videos by several orders of magnitude (from quadratic to linear on the number of videos). This yields better results than choosing pairs at random, which may result in non-plausible scenarios, or choosing pairs from the same class, which may not add as much diversity.

In summary, the proposed **Learn2Augment** contains three core components: a Selector that learns to choose good videos to augment, a semantic matching method that improves optimization, and a video compositing that composites video pairs for augmentation. Experimental results show that all components contribute to the performance of the system in different ways, and the overall method obtains state-of-the-art in all datasets, and in all settings that involve limited training data. In addition, in the setting which considers the full training set, the proposed data augmentation technique improves upon the baseline on all datasets, including UCF101, HMDB51, and the large-scale Kinetics-400.

9.2 Method

In this section we describe in detail the architecture of the proposed Learn2Augment. In a nutshell, the goal is to learn to augment novel data points which are realistic and diverse, such that we can train a better classifier with them. For this, we train a Selector network, which predicts a score of how useful a given pair of videos is for augmentation. We pick pairs that have a high score to be augmented. The transformation we use for augmentation is Video Compositing. Training the Selector

using the entire dataset is infeasible, and sampling pairs of videos at random will yield unlikely pairs. Thus we sample pairs of videos using Semantic Matching. Figure 9.2 shows an overview of the proposed method and in Sec. 9.3 we describe how we train our approach.

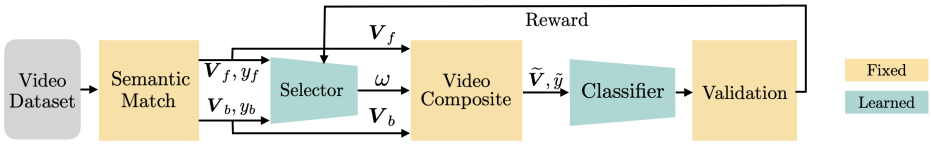


Figure 9.2: Overview of the proposed Learn2Augment. Given a pair of videos and their labels, a Selector network gives a score ω of the quality of the potential composited video. At training time, the Selector is trained with the validation loss of the classification network. Once the Selector is trained, pairs of videos are sampled, and only the promising combinations with high score ω are composited and used for training the classifier.

9.2.1 Selector

Given two input videos V_1 and V_2, the goal of the Selector is to predict a weight ω, rating the quality of the potential composited video. Note that the input to the Selector is two putative videos instead of the composited one. This means that at test time, we can predict how useful the composited video will be without having to actually create it.

The architecture of the Selector includes a standard video classification network to extract video features, which is ResNet3D-18 [74] followed by a simple multi-layer perceptron (MLP) with 3 hidden layers of sizes 2048, 1024 and 512. Two videos are input to the Selector at a time, and their features and labels are concatenated and input to the MLP.

Since there is no ground truth of how "good" a video sample is for learning, we train the Selector using the change in validation loss of the classifier. This is, we argue that a "good" training sample is one which, if used for training, improves the validation loss of the classification network. In other words, if we take one optimization step training the classifier, after updating the weights, the validation loss will go down. Section 9.3.1 describes the training process in detail.

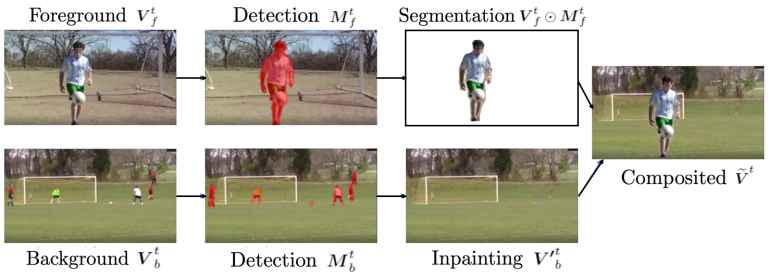


Figure 9.3: Pipeline for compositing a single frame. The foreground is from the class "soccer juggling" and the background from the class "soccer penalty", which are semantic class neighbors. We can see objects such as 'person' and 'ball' are detected as objects of interest.

At test time, we use the Selector by sampling pairs of videos, choosing those pairs with high score ω, and input to the Video Compositingmodule, which we describe in Sec. 9.2.3. The resulting video is finally used to augment the training set for the classification network.

9.2.2 Semantic Matching (SM)

The number of pairs in the full dataset can be very large, as it grows with the square of the number of videos. For Kinetics [20], for example, we would encounter 360 billion pairs. Training the classifier using these is clearly infeasible, and thus we use the Selector. But training the Selector itself with all these samples is infeasible too. Sampling uniformly is a reasonable solution, but many video pairs may not be useful for learning. We leverage the observation that all combinations of actions and backgrounds are not equally likely [25]. This natural correlation between actions and backgrounds helps to prune unlikely class combinations.

For this, we make the assumption that classes that are semantically similar are more likely to contain a foreground and a background that are plausible in the real world, and therefore more realistic for our data augmentation purposes. Thus, we use the class names to extract a language embedding using sen2vec [130], and use these embeddings to match each class to its nearest neighbor. We sample videos V_1 and V_2 from class c_1 and its closest neighbor c_2 respectively. This simple decision reduces the number of pairs to grow linearly with the size of the dataset, and furthermore

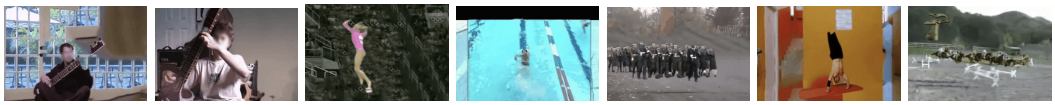


Figure 9.4: Sample frames of rendered videos. While the segmentation contains errors, such as missing limbs or portions of the object, the action category remains clear.

increases the accuracy significantly with respect to sampling video pairs at random. More details on the numerical impact can be found in Sec. 9.3.5. Semantic class pairs and additional experiments using intra-class augmentation can be found in the supplementary material.

9.2.3 Video Compositing(VC)

The goal of the augmentation process is to composite two videos, to produce realistic, plausible and diverse new videos, that will improve the classification. Figure 9.3 shows the overall pipeline for compositing a single frame.

Given two videos which will be used for foreground V_f and background V_b, we use a standard object segmentation network (MaskRCNN [73]) to segment out people and objects in every frame of both videos. Objects categories in action datasets are not completely contained in the image dataset COCO [105], which is used for training MaskRCNN. However, we observe that object detections with high confidence tend to correspond to actual objects, even if the category is not correct (boxing bag is often classified as fire hydrant), and therefore are useful to our purpose. We could also have selected only the humans in the video, as action categories tend to be focused on humans. However, we find that the presence of specific objects is highly correlated with action categories (musical instruments in the classes “playing guitar" or “playing violin"). Therefore removing the original objects from the background and adding the ones from the foreground is essential for recognition. See numerical results of the impact of these decisions in the ablation study of Sec. 9.3.5.

We remove the segmented objects from the background video and fill in the holes using image inpainting [107], to obtain a clean background video V'_b. Finally, we combine the foreground objects and the background at each frame by simple composition, as in:

$$\widetilde{V}^t = V^t_f \odot M^t_f + V'^t_b \odot (1 - M^t_f), \tag{9.1}$$

where $\widetilde{V}^t$ is the resulting composited frame at time t, ${V^t}_f$ and ${V'^t}_b$ are frames of the foreground and background videos respectively, ${M^t}_f$ is the binary mask with the union of all detected objects, and $\odot$ is the element-wise multiplication. Figure 9.4 shows sample frames of the resulting videos.

9.3 Optimization of Learn2Augment

The optimization of the proposed Learn2Augment method has two stages. In the first stage, we train the Selector network using RL, as described in Sec. 9.3.1. Once the Selector network is trained, in the second stage, we perform data augmentation to train the classifier. That is, we sample pairs of videos, pass them through the trained Selector, choose the pairs with high score, create new videos with these pairs through Video Compositing, and add them to the training set. We now describe the details of these two training stages.

9.3.1 Training the Selector

As mentioned before, there is no ground truth to tell us how good an augmented data sample is. Instead, we use the validation loss of the classification network to train the Selector network. This function is not differentiable with respect to the parameters of the Selector. A common solution to dealing with this is to use RL [200].

Specifically, the state s_t at time t is the batch of video pairs sampled using SM. The action a_t is the subset of these video pairs selected for compositing and is represented as a vector of values between 0 and 1. The environment is the classification network and the validation process. This environment is used to compute a reward $R(\theta)$ for choosing a particular action, where θ are the parameters of the Selector.

We calculate the reward in a single step, as the difference between the loss in the current batch and the moving average of losses in the previous S steps (where $S = 5$) denoted as δ, as in Eq. 9.2:

$$R(\phi) = \left(\frac{1}{|D_{\text{val}}|} \sum_{i=1}^{|D_{\text{val}}|} \mathscr{L}_{\text{cls}}(f_\phi(V_i), y_i) \right) - \delta \tag{9.2}$$

where $\mathscr{L}_{\text{cls}}$ is the classification cross-entropy loss, f_ϕ is the classifier network of parameters ϕ, V_i and y_i are an input video and its label respectively, D_{val} is the validation set and $|D_{\text{val}}|$ is the number of samples in D_{val}. The objective function that we want to maximize is the expected value of the reward:

$$J(\theta) = \mathbb{E}(R(\phi)). \tag{9.3}$$

To find the optimal policy, we would typically differentiate the objective function with respect to the parameters θ. However, the reward function is dependent on the validation loss, calculated with the classifier network, which does not involve θ. Instead, using REINFORCE [179], we approximate the objective function as:

$$\nabla_\theta J(\theta) \approx \frac{1}{M} \sum_{i=1}^{M} R_{\tau^i}(\phi) \left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_i^t | s_i^t) \right), \tag{9.4}$$

where, τ^i is the i^{th} state-action trajectory under the policy π_θ, M is the number of sample trajectories and T is the number of actions performed in a trajectory. Note that as we have single-step episodes, we can make several simplifications as $M = 1$, as $T = 1$, and as there is only one trajectory τ^i, and thus $R_{\tau^i}(\phi)$ is just $R(\phi)$. With these simplifications and substituting Eq. 9.3 in Eq. 9.4, we obtain:

$$\nabla_\theta J(\theta) \approx R(\phi) \nabla_\theta \log \pi_\theta(D_M | D_B), \tag{9.5}$$

where D_M corresponds to the subset of pairs of samples to composite and D_B to all the pairs of samples in the batch. The Selector is updated by $\alpha \nabla_\theta J(\theta)$ where α is the learning rate and δ is updated with the last calculated loss as seen in Eq 9.6.

$$\delta_t = \frac{S-1}{S} \delta_{t-1} + \frac{1}{|D_{\text{val}}|} \sum_{i=1}^{|D_{\text{val}}|} \left(\mathscr{L}_{cls}(f_\theta(V_i), y_i) \right). \tag{9.6}$$

Note that this training process does involve generating the composited videos for pairs in D_M, to input to the classifier and compute the loss. However, crucially, during training this is a small portion (one order of magnitude smaller) of how many videos would need to be generated if we were to composite all pairs of videos.

Once the Selector is trained, we use it for actually filtering good pairs. At that point, given two videos and their labels, the Selector network predicts a policy π of how likely it is to select the pair. The score ω is the value of π for each pair. We use a threshold on that score to select the pairs of videos to augment. In our experiments, we first determine a budget on the number of videos that we want to augment, and then pick the threshold to select the top-ranked video pairs. We use these selected pairs of videos as input to Video Compositing, add them to the training set, and use them to train the classifier.

9.3.2 Training the Classifier

Similar to previous work which combines multiple samples for augmentation [203, 223], composited/mixed samples should include mixed labels. We adopt the strategy of Cutmix [203], where the foreground label y_f and the background label y_b are combined using a ratio λ, as:

$$\tilde{y} = \lambda y_f + (1-\lambda) y_b, \tag{9.7}$$

to obtain the mixed label $\tilde{y}$. A simple way to choose λ is to use the ratio of the foreground mask with respect to the overall video. Given the foreground video V_f of dimensions $T \times H \times W$, and mask at each frame M_f, the foreground ratio would be $\gamma = \sum M_f/(THW)$. Instead of choosing λ to be directly proportional to the foreground ratio γ, we give slightly more weight to the foreground [223], as in Eq. 9.8, where $\alpha = 4$.

$$\lambda = -(\gamma-1)^{\alpha} + 1, \gamma \in [0,1] \tag{9.8}$$

We add composited videos $\widetilde{V}$, and their mixed labels $\tilde{y}$ to the training set, and train the classifier network using a standard cross-entropy loss, with stochastic gradient descent.

The choice of classifier is not tied to our method. In our experiments, we choose the widely used 3D ResNet-18 architecture, which allows us to compare directly to other approaches.

9.3.3 Experimental Details

Datasets. In order to provide comparison to prior work (e.g. [223, 154]), we use standard datasets for evaluation in action recognition, including HMDB51 [90], UCF101 [157], Kinetics-400 [20], and Kinetics-100, which includes the 100 classes with the largest amount of samples in Kinetics, as it is used in prior work [84] and helps us compare directly. For experiments on the effect of pre-training the Selector, we use Kinetics-400. For the semi-supervised setting, we split the datasets following the protocol of VideoSSL [84] and ActorCut [223]. For few-shot we use the standard split [209] and the Truze split [65] which ensures no overlap of novel classes with Kinetics-400.

Problem Settings. We test the proposed method in three different settings. In the *semi-supervised* setting, a portion of the training set is artificially held out, and the rest of the training data is assumed to be available, but unlabeled. Tests are performed on different percentages of held out data. In the *few-shot* setting, some classes (novel classes) are assumed to have a very small number of training samples (one to five instances), while other classes have the full number of samples (seen classes). We effectively change the n-shot learning problem to a $n+k$-shot problem where k is the number of augmented samples. Finally, in the standard *full set* setting, all training data is available.

Training Settings. We use mini-batch stochastic gradient descent, with momentum of 0.9 and weight decay 0.001. For each video, we use an 8-frame clip, where the frames are uniformly sampled. We use batch size of 8. For UCF101 and Kinetics100 in the SSL setting, we train the model for 400 epochs and for HMDB51, we train for 500 epochs. The initial learning rate is set to 0.1 and then decayed using cosine annealing policy. For the SSL setting, we use the data split proposed in VideoSSL [84]. For the few-shot setting, we use the default hyperparameters of TRX [133], ARN [209] and C3D-PN [155], respectively. In the fully supervised setting, we train R(2+1)D for 100 epochs on UCF101, HMDB51 and 50 epochs on Kinetics-400.

9.3.4 Architectural Changes for Different Settings

We briefly explain the structural adaptations of our approach for each of the settings.

Semi-supervised Learning. Similar to VideoSSL [84], we first train the classifier on the available labeled data using the categorical cross-entropy loss. Once this network is trained, we do a forward pass of the unlabeled examples and assign pseudo-labels to those samples with high confidence. We use these pseudo-labels as additional data for augmentation. We also add a knowledge distillation loss inspired by VideoSSL [84].

Few-shot Learning. We only augment the novel classes using Learn2Augment. We also do not perform label mixing and simply use the foreground label for the augmented sample. This incorporates our composited samples seamlessly into the meta-learning framework typically followed. We show results on the standard split, as on the recently proposed TruZe [65]. TruZe ensures that the novel classes do not overlap with Kinetics-400.

Fully-supervised Learning. This is the simplest setting, where the Selector is trained on the full training set, and used for data augmentation to train the classifier. We explore two scenarios: training the classifier from scratch and using a model pre-trained on Sports1M [86].

9.3.5 Ablation Study

Table 9.1 shows the ablation study of Learn2Augment, which illustrates the impact of each of the proposed elements in the design. The experiment is done on the UCF101 dataset, using 20% of the data i.e. in a semi-supervised setting. All three contributions (Selector, Semantic Matching and Video Compositing) improve accuracy. Crucially, Semantic Matching and the Selector also reduce greatly the number of possible video combinations, and the overall reduction is around three orders of magnitude. We see that Learn2Augment obtains a 13.4% improvement over the baseline. While there are improvements of up to 7.4% for each component, the combination of all three gives the best results.

Pairs Selector	Video Compositing	Semantic Matching	Accuracy in %	#Videos (S)
✓	✓	✓	**58.9**	12K
×	✓	✓	55.8	99K
✓	×	✓	54.5	12K
✓	✓	×	55.2	(1.2M)
✓	×	×	52.9	(1.2M)
×	✓	×	48.6	(10.4M)
×	×	✓	50.8	99K
×	×	×	45.5	(10.4M)

Table 9.1: Ablation study to explore the impact of each proposed component. All settings use the same number of samples for training, so that they can be compared fairly. The # Videos (S) corresponds to the search space in each scenario. As we can see, we obtain the best accuracy using just 12K instead of the standard scenario which would have had 10.4M i.e. a reduction of over 1000x.

Method	Accuracy
L2A	58.9
L2A w/o Inpaint	57.6
L2A w/o Segmentation	56.8
L2A w/o Objects	55.7
L2A w/o All	54.5

Table 9.2: Ablation study of compositing components. The version “w/o Inpaint" refers to pasting the foreground without first filling in the holes of removed objects in the background. The version “w/o Segmentation" refers to using bounding boxes instead of object segmentations. “w/o Objects" refers to copying and pasting only the humans in the scene, leaving the objects.

The Video Compositing module also has multiple components. In Table 9.2, we ablate these components and observe that removing objects is actually essential, and has the most significant impact, followed by using segmentation instead of a bounding box, and finally inpainting.

Although the compositing process is more computationally expensive than previous simpler mixing strategies, it is important to note that 1) the overall accuracy indeed improves, 2) the actual composition for training the classifier is done on a small subset of pairs of videos and 3) the Selector can be trained on a large dataset (e.g.: Kinetics) just once and can be reused for the smaller datasets without the need of fine-tuning (see Table 9.3).

		Kinetics 100				UCF101				HMDB51		
Method	Conference	50%	20%	10%	5%	50%	20%	10%	5%	60%	50%	40%
CutMix [203]	ICCV19	53.7	46.1	43.2	39.9	46.1	36.5	34.6	25.8	33.9	30.8	27.8
MixUp [207]	ICLR18	53.4	45.5	43.0	39.6	45.8	36.1	34.2	25.5	33.7	31.0	27.5
CutOut [33]	Arxiv17	52.8	45.1	42.3	38.8	45.2	35.6	33.9	24.6	33.0	30.5	27.1
ST-VideoMix [204]	Arxiv21	55.3	46.6	43.9	40.4	46.4	36.4	35.2	25.9	34.8	31.3	28.7
PseudoLabel [97]	ICMLW13	59.0	48.0	38.9	27.9	47.5	37.0	24.7	17.6	33.5	32.4	27.3
MeanTeacher [165]	Neurips17	59.3	47.1	36.4	27.8	45.8	36.3	25.6	17.5	32.2	30.4	27.2
S4L [205]	ICCV19	54.6	51.1	43.3	33.0	47.9	37.7	29.1	22.7	35.6	31.0	29.8
VideoSSL [84]	WACV21	65.0	57.7	52.6	47.6	54.3	48.7	42.0	32.4	37.0	36.2	32.7
ActorCut [223]	Arxiv21	68.7	61.2	56.8	52.7	59.9	51.7	40.2	27.0	38.9	38.2	32.9
ActorCut+ID [223]	Arxiv21	72.2	68.7	63.9	59.1	64.7	57.4	53.0	45.1	40.8	39.5	35.7
TCL [154]	ICCV21	70.4	64.7	61.1	58.2	62.1	55.4	52.1	42.8	41.2	40.4	34.8
L2A		**75.9**	**72.1**	**67.5**	**63.7**	72.1	60.3	56.1	48.0	44.5	43.2	37.9
L2A+Pre-training		-	-	-	-	**73.3**	**64.8**	**60.1**	**50.9**	**47.1**	**46.3**	**42.1**

Table 9.3: Results on the semi-supervised setting. Results for TCL and ActorCut are obtained by us running the author's code. All methods are run with a 3D ResNet-18 backbone for fair comparison. L2A+Pre-training refers to pre-training the selector and fixing it.

9.3.6 Augmenting in the Semi-supervised Setting

In this setting we artificially hold out a portion of the training set, with the goal of observing the behavior of different methods as the size of the training set changes. In this setting, we use the remaining part of the dataset by producing pseudo-labels, similar to VideoSSL [84]. Table 9.3 shows results in this semi-supervised setting. The L2A version of the method uses a Selector and a classifier trained only on the target dataset (in this case UCF101, HMDB51 or Kinetics-100). We observe that Learn2Augment improves on all settings over all previous methods.

The "L2A+Pre-training" row refers to Learn2Augment where the Selector has been pre-trained on Kinetics-400, without fine-tuning on the target dataset. We make two observations: First that pre-training on a large dataset helps, as the results from the pre-trained model are higher for all datasets and settings. Second that the Selector trained on Kinetics generalizes quite well to the smaller datasets without the need for fine-tuning. We do not test on Kinetics-100 with the pre-trained model, as this would mix training and testing sets.

		UCF101					HMDB51				
Method	Split	1	2	3	4	5	1	2	3	4	5
C3D-PN [155]	S	57.1	66.4	71.7	75.5	78.2	38.1	47.5	50.3	55.6	57.4
C3D-PN + L2A	S	**60.8**	**68.9**	**73.3**	**76.6**	**79.1**	**39.8**	**48.9**	**51.5**	**57.3**	**58.2**
ARN [209]	S	66.3	73.1	77.9	80.4	83.1	45.5	50.1	54.2	58.7	60.6
ARN + L2A	S	**67.7**	**74.2**	**79.6**	**81.1**	**84.4**	**47.3**	**51.7**	**55.5**	**60.1**	**61.8**
TRX [133]	S	77.5	88.8	92.8	94.7	96.1	50.5	62.7	66.9	73.5	75.6
TRX + L2A	S	**79.2**	**89.2**	**93.2**	**95.0**	**96.3**	**51.9**	**63.8**	**68.2**	**74.4**	**77.0**
C3D-PN [155]	T	50.9	61.9	67.5	72.9	75.4	28.8	38.5	43.4	46.7	49.1
C3D-PN + L2A	T	**52.5**	**63.8**	**70.1**	**75.2**	**78.2**	**29.9**	**40.1**	**44.5**	**47.7**	50.8
ARN [209]	T	61.2	70.7	75.2	78.8	80.2	31.9	42.3	46.5	49.8	53.2
ARN + L2A	T	**63.9**	**73.1**	**77.4**	**80.4**	**81.3**	**33.6**	**43.7**	**48.0**	**51.1**	**53.8**
TRX [133]	T	75.2	88.1	91.5	93.1	93.5	33.5	46.7	49.8	57.9	61.5
TRX + L2A	T	**76.8**	**88.9**	**92.7**	**93.8**	**94.1**	**35.0**	**48.1**	**51.1**	**59.2**	**62.1**

Table 9.4: Results on UCF101 for the Few-Shot Learning setting, with different splits. Accuracies are reported for 5-way, 1, 2, 3, 4, 5-shot classification. S corresponds to the split used in [209, 133] and T is the TruZe split [65], which avoids overlapping classes with Kinetics.

9.3.7 Augmenting in the Few-shot Setting

We also explore the impact of the proposed method on the more extreme few-shot setting, where there are only a few examples per class. This is interesting because few-shot methods are already designed to address data scarcity.

We compare with the current state of the art in this setting, including CD3-PN [155], ARN [209] and TRX [133], on the UCF101 and HMDB51 datasets. We observe that the proposed Learn2Augment method improves upon all existing approaches, suggesting data augmentation is complementary to few-shot methods. Table 9.4 shows the results of the experiments.

9.3.8 Augmenting the Full Training Set

We finally explore the effect of augmenting the full dataset, both for smaller datasets, and the large-scale Kinetics. Results can be found on Table 9.5. Again, Learn2Augment improves the performance on all datasets even for a pre-trained model.

Augmentation	Dataset	Pretrained	Top-1
Standard	UCF101	No Pretraining	55.7
ActorCut [223]	UCF101	No Pretraining	68.3
L2A	UCF101	No Pretraining	**73.1**
Standard	HMDB51	No Pretraining	40.8
ActorCut [223]	HMDB51	No Pretraining	44.5
L2A	HMDB51	No Pretraining	**46.4**
Standard	UCF101	Sports1M	93.6
L2A	UCF101	Sports1M	**95.3**
Standard	HMDB51	Sports1M	66.6
L2A	HMDB51	Sports1M	**68.4**
Standard	Kinetics	Sports1M	75.4
L2A	Kinetics	Sports1M	**76.3**

Table 9.5: Augmenting standard datasets improves classification even with a model pre-trained on the largest existing dataset (Sports1M).

9.4 Why Not Intra-class Augmentation?

One other possibility we explored is intra-class augmentation instead of using semantic classes. However, when we followed the same procedure on 20% labeled data of UCF101 we obtain an accuracy of 41.4% in comparison to 58.9% when using semantically similar classes. Similarly, in Kinetics100 we obtain an accuracy of 50.1% and 54.4% using 5% and 10% labeled data respectively. That is 9.4% and 8.9% lower than the results using semantic neighbors. We believe there to be two main concerns in intra-class augmentation. The first is that Cutmix [203] has been shown to be an excellent regularization technique. This is aided by having samples that have soft labels (since they are a ratio of samples from different classes). However, using intra-class augmentation would force the labels to be the same as the ground truth class. The second reason is that samples of a particular class are clips that were part of the same video. This is the case in both HMDB51 and UCF101 and not so in Kinetics100. If we cut the background from one sample and paste the foreground onto this, it results in an identical sample to the original foreground sample. This is because the background is the same in both cases. All we end up doing then is training the model on multiple instances of the same data which leads to overfitting and hence a poor accuracy at test time. However, since the results are much worse for Kinetics100 as well, we believe that this could be a smaller contributing factor.

9.5 Distillation Loss for Semi-Supervised Learning

Given frame a from video v, to distill appearance information of objects of interest, we use the softmax predictions of a ResNet [74] image classifier. This network is pre-trained on Imagenet and not modified during training. Let the output of the ResNet be denoted as $h(a) \in \mathbb{R}^M$ where M = 1000 which is the number of classes in Imagenet. We randomly select a frame from all videos (labeled, unlabeled and augmented) for training. The classifier model in our architecture, produces an embedding $q(v) \in \mathbb{R}^M$ which is of the same dimensions and space of $h(a)$. We train $q(v)$ to match the output of $h(a)$ by using a soft cross-entropy loss that treats the ResNet outputs as soft labels. This loss $\mathscr{L}_d$ can be seen in Eq. 9.9. Our final loss function is a combination of $\mathscr{L}_d$ and $\mathscr{L}_s$ (categorical cross-entropy loss for video samples). This is done following the work in VideoSSL [84].

$$\mathscr{L}_d = - \sum_{v \varepsilon (X \cup Z), a \varepsilon v} h(a) \log \left(q(v) \right) \tag{9.9}$$

9.6 Analysis of Number of Augmented Samples

We see a common pattern when adding augmented samples to the different SSL settings. This basically refers to increasing the number of augmented samples in the training set. We see that the accuracy increases initially, reaches a peak performance and then starts dropping slowly as can be seen in Figure 9.5. This makes sense as we don't expect every mixed example to be helpful for training. In fact, this helps us to define ω_i for the selector. We can see Figure 9.5 for the results from 0 augmentations to 5000 for 10% and 20% labeled data on UCF101. The sweet spot for the 10% labeled data is around 1200 augmentations and for the 20% labeled data is around 2000 augmentations. Both of which are obtained using $\omega_i = 0.6$. We decide the value of ω_i based on these and results and use the same for HMDB51 and Kinetics100 for all settings. If we increase the value of ω_i we obtain fewer samples and decreasing the value of ω_i results in more number of samples for training. The value of ω_i thus determines the number of augmented samples and also their quality.

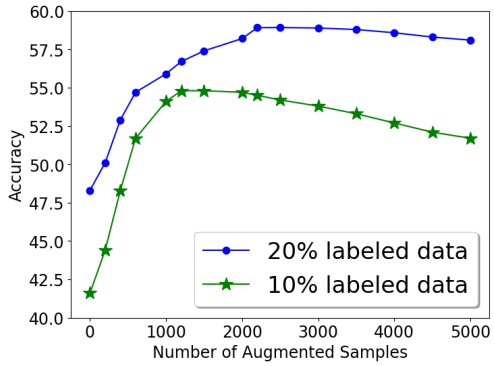


Figure 9.5: Comparison of performance with increasing number of augmented samples. Results are for 10% and 20% of labeled data UCF101. We see that the performance increases initially, reaches a peak and slowly starts dropping.

9.7 Other Selector Choices

The design of the selector is a crucial aspect of our model. We want the selector to be able to learn what makes a good pair of videos for mixing without actually having to mix every single pair. However, for lower percentages of labeled data, we can generate all possible samples of semantic classes and convert a state-of-the art frame selection model (SMART) [62] to do sample importance instead of frame importance. We also consider a simple baseline of using a discriminator network to pick only realistic samples. We report the results in Table 9.6. Another approach was to randomly pick a certain amount of samples to train the classifier network.

We not only outperform all alternative approaches, we also do this by saving on both memory and computation cost. For example, in the 20 percent setting, SMART sees 99K videos and these 99k videos have to be precomputed and stored before training SMART. However, the proposed approach only needs 12K videos and outperforms SMART by up to 1.4%. This analysis is only to show a comparison to possible alternatives when storing data is feasible. The idea of trying these alternatives is only feasible in low percentage labeled data of small datasets like UCF101 and HMDB51. Even 50% labeled data in UCF101, results in having to mix over 400k videos while large scale datasets like Kinetics400 would lead to millions of mixes being needed making it practically unfeasible.

	50%		20%		10%		5%	
Method	Acc	SS	Acc	SS	Acc	SS	Acc	SS
Random	61.9	430K	56.2	99K	51.8	44K	42.3	9.7K
Discriminator	62.8	430K	57.3	99K	52.2	44K	41.1	9.7K
SMART [62]	68.9	430K	58.9	99K	57.8	44K	46.5	9.7K
Proposed	**72.1**	39K	**60.3**	12K	**56.1**	5.2K	**48.0**	1.2K

Table 9.6: Comparison of approaches for the use of Selector. All results are reported on UCF101. 'Acc' corresponds to accuracy and 'SS' corresponds to the number of mixed videos that the Selector looks at. All results are on different percentage of labeled data in UCF101.

9.8 Why Re-train the Classifier Network?

Here, we are talking about the classifier network in our proposed architecture that the selector learns from (based on the validation loss). Training the Selector and the Classifier together is also possible. But we decide against this for 2 reasons. First, and the most important reason is that we want to save out on computational cost needed to generate an augmented sample. We showed that the selector network looks only at a fraction of samples before it understands what makes a good pair. Hence, we first train the selector by generating augmented samples taken from random samples of semantically similar classes. Once the selector is trained, we don't need to generate the mixed sample for all possible pairs and only generate the mixed samples for good pairs (the selector need not have seen these pairs before). We then augment the original dataset by samples that the selector believes will improve the classifiers performance We compare the performance of the joint training and re-training of the classifier network in Table 9.7. We see that re-training the classifier network always yields the best performance.

Method	50%	20%	10%	5%
Jointly trained	66.5	57.4	53.1	44.7
Retrained	**72.1**	**60.3**	**56.1**	**48.0**

Table 9.7: Comparison of jointly training classifier and re-training it. We see that there is a consistent large improvement in re-training the classifier.

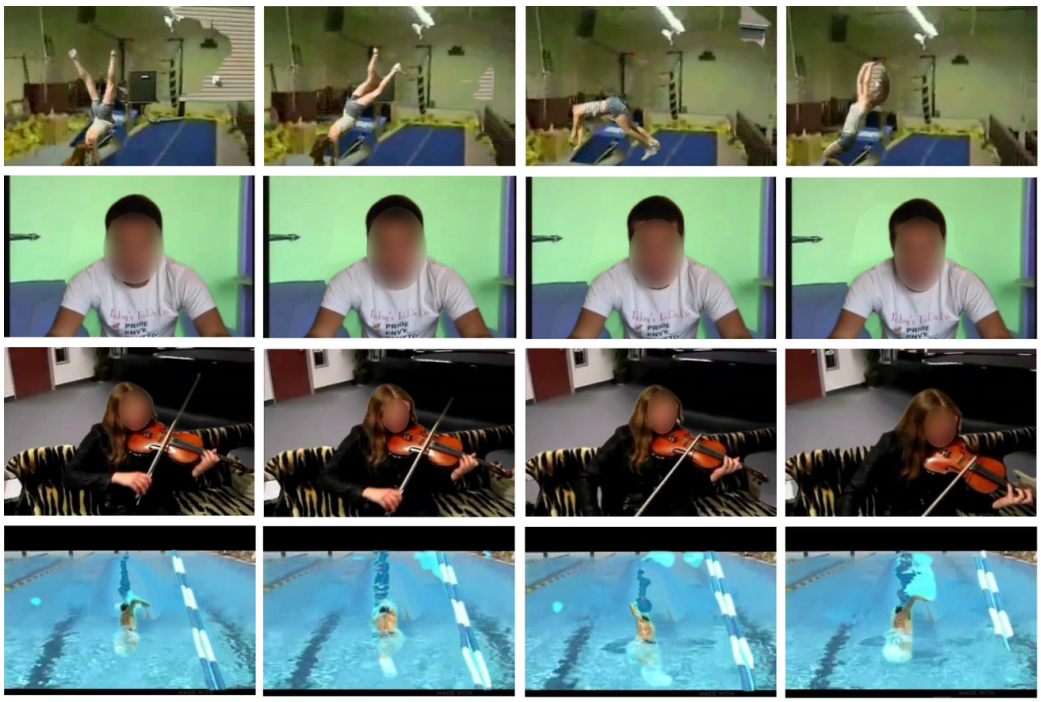


Figure 9.6: Visualizing selected examples. From top to bottom as (foreground, background) pairs: (flic-flac, cartwheel), (smile, laugh), (playing violin, playing cello), (front crawl, swimming backstroke). The first two are examples from HMDB51 and the last two from UCF101.

9.9 Examples of Selected and Discarded Samples

To understand what made a good sample we visualize a few samples that were selected by the selector model and a few samples that were discarded. These can be seen in Figure 9.6. The samples are displayed as 4 frames for better visualization. Based on the small subset of examples seen, we believe that for good pairs to be selected some of the criteria could be coherent inpainting, similar camera movement, not too drastic a background change.

We see some samples of discarded examples in Figure 9.7. Based on the small subset of examples seen we think possible bad pairs are due to bad video compositing (example 2 in Figure 9.7), varying camera movements (example 3 in Figure 9.7) or a drastic change in background (example 1 in Figure 9.7). These are however based on the few examples we see.

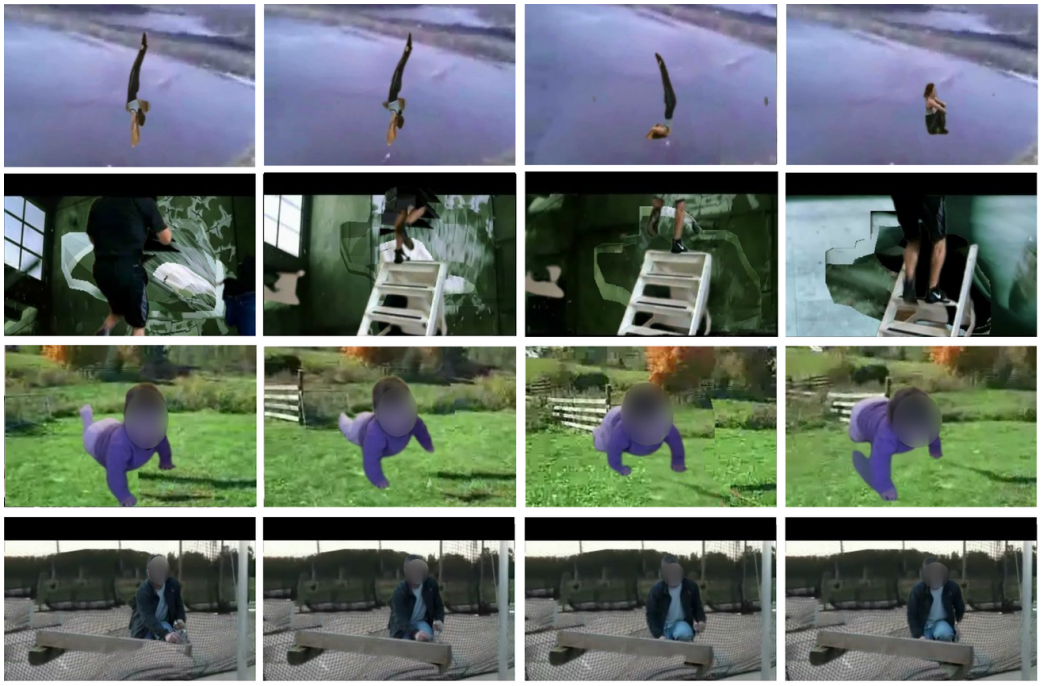


Figure 9.7: Visualizing discarded examples. From top to bottom as (foreground, background) pairs: (somersault, diving), (climbing stairs, falling floor), (baby crawling, walking dog), (hammering, hammer throw).

9.10 Effect of Semantic Match in generalization ability

We test the generalization ability of the semantic matching by comparing it with random matching which would correspond to row 4 of Table 1. We observe that the performance does decrease. To strengthen this test, we tried the same experiment in the FSL setting, which is an extreme case for generalization. We augment data for two different methods, using the proposed L2A, using both semantic and random matching of classes. We observe that even in this setting, which is the most susceptible to overfitting, the semantic matching outperforms random matching. We will add this to the final version.

Method	Class Matching	1-shot	3-shot	5-shot
C3D-PN	Random	28.1	42.9	47.7
C3D-PN	Semantic	29.9	44.5	50.8
TRX	Random	33.5	49.9	60.3
TRX	Semantic	35.0	51.1	62.1

Table 9.8: Results on FSL using the proposed Semantic Matching vs random matching using the TruZe [65] split.

9.11 Limitations and Future Work

The main area of improvement is the time needed for training. Optimizing the Selector with RL is time-consuming, and so is compositing the initial samples for training it. Future work could address this by parameterizing the composition process and learn these parameters instead of compositing the pairs directly. It could also learn to select particular frames in a video, and avoid the computational cost of temporal redundancy. Finally, another possible direction is to learn what samples to discard from the initial dataset itself.

9.12 Conclusion

While standard data augmentation strategies in action recognition are hand-crafted, we propose to learn which pairs of videos are good to composite. In order to do this, our approach leverages three components. We train a Selector optimized with RL to choose which pairs of videos are good to composite. We reduce the search space by using samples from semantically similar classes. We perform a clean segmentation for mixing samples and remove actors as well as objects from foreground and background samples. With this, we obtain state-of-the-art results in semi-supervised and few-shot action recognition settings, and improve in the fully supervised setting. In particular, we see gains of up to 8.6% and 3.7% in the semi-supervised and few-shot settings. We also see an improvement of up to 17.4% when compared to standard augmentation in the fully supervised setting when training from scratch.

Chapter 10

Conclusion and Future Work

In this thesis, we focused on efficient action recognition and address efficiency in both data and memory. We showed two key principles that helped effectively contribute towards these goals.

The first is that we only need a subset of the data to obtain best performance for a model and contrary to popular belief more data can actually deter accuracy. Crucially, this subset is learnable i.e. we can learn how to choose this subset. In context of memory efficiency, we see that selecting the most important frames is a crucial step in fully supervised action recognition of 2D and 3D models. This can reduce overall cost of both training and inference of the model whilst boosting the accuracy at the same time. In context of data efficiency, we show that we can learn to choose synthetic samples of generated samples and such an approach is both data and model-agnostic i.e. it improves feature generating approaches in both images and videos. Further, we show that we can learn to select pairs of videos for compositing as a form of data augmentation without actually compositing them saving significant time and memory.

Second, we show that obtaining richer features by enriching feature learning through either joint multimodal learning or sophisticated language descriptions is crucial for zero-shot learning. Traditional works train visual encoders and semantic encoders separately and use a common latent space in either visual or semantic space to train the model. We show that learning this jointly can significantly boost performance. Also, a lot of research is being done on the visual side of learning with less importance on the semantic side. We propose a large scale textual dataset that we show can improve multiple zero-shot learning models.

Along with this, we have proposed a new split for true zero-shot action recognition, ensuring zero overlap between unseen test classes and training or pre-training classes. This split is essential for evaluating the true performance of zero-shot action recognition algorithms.

We have also proposed a unique training regime for factorized encoder variants of video transformer architectures. First we pretrain a cheaper version of the model using fewer frames. Then we fine tune with more frames, freezing the spatial encoder and adding an adapter between the frozen spatial representations and temporal transformer. This includes pretraining the temporal transformer, often overlooked in current models.

We believe that these contributions make significant advances in the field of action recognition. They pave the way for more efficient, resource-friendly, and robust video processing and understanding techniques.

10.1 Short-Term Improvements and Future Work

The works presented in the PhD make significant contributions towards more data-efficient and computationally-efficient video action recognition, with demonstrated state-of-the-art results on multiple benchmarks. However, there remain opportunities for further enhancements and extensions of these methods in the short-term.

For the frame selection strategy SMART, while it achieves strong improvements in efficiency, there is room to explore more advanced relation modeling between frames, such as graph neural networks or transformer-based approaches. The current frame scoring mechanism could also potentially be improved with attention over spatial regions rather than entire frames.

The training optimizations for ViViT could be expanded to other video Transformer models, assessing their impact on larger and more complex architectures. The adapter-based fine-tuning may also benefit from exploring different adapter designs or integration strategies.

For zero-shot learning methods like TruZe, CLASTER and Stories, while they push towards more realistic evaluation, there is scope for further improving generalization on unseen classes. Semi-supervised learning using unlabeled videos may help learn more universal visual representations. Exploring different semantic spaces beyond Word2Vec embeddings could also be beneficial.

The data augmentation approach Learn2Augment could be extended to generate more complex transformations like mixing background scenes or manipulating foreground objects. The integration between foreground and background could also be improved with more advanced compositing techniques like matting.

The synthetic sample selection strategy Spot currently operates on pre-generated features. An end-to-end approach jointly optimizing feature extraction and sample selection may further enhance efficiency and performance. Expanding the method to other data modalities like text could also be worthwhile.

Overall, there are significant opportunities to build on these works to further advance efficient and generalizable video understanding. Focused investigation of neural architectural improvements, enhanced objective functions, and multi-modal representations will likely yield additional gains.

10.2 Long Term Directions

Stepping back, the PhD research points towards some promising longer-term directions for video analysis. Continued efforts toward generalizable representations that can effectively handle novel classes and domains seems inevitable. This entails learning robust visual-semantic spaces, regularizing for redundancy, and leveraging massive unlabeled video data. Tighter integration of visual, textual and audio understanding. Learning joint representations across modalities can enhance generalization and mirror real-world semantic understanding.

Scaling up video transformer architectures as can be seen in recent works such as ViVit [5] and Uniformer [99]. Designing efficient attention mechanisms and sparser interactions to enable longer sequences, more frames, and higher resolutions. Video generation for data augmentation. As generation models advance, synthesized video

could provide unlimited data for training to handle long-tail novelty. Online adaptation and lifelong learning for video models. Enabling systems to incrementally update over time to handle changing distributions and new classes, rather than training from scratch.

Video understanding for embodiment and robotics related tasks is another promising direction. Moving beyond passive observation to agents that can learn grounded, interactive representations through physical experience. Creating new video benchmark datasets with greater diversity. Current datasets have limited generality - creating varied benchmarks with long-tailed distributions can drive progress in generalization. Video understanding with less supervision. Leveraging massive unlabeled video along with self-supervision and multi-task learning to reduce annotation dependence.

In summary, advancing scalable, generalizable video understanding will remain a key challenge in computer vision research. The PhD works provide promising steps in this direction, while highlighting many open problems for continued investigation. Developing robust and adaptable video intelligence could enable transformative applications across domains like healthcare, education, home automation, and transportation.

10.3 Ethical Impact

Data-efficient techniques like zero-shot learning, few-shot learning, semi-supervised learning, and memory-efficient methods have significant ethical implications in the context of action recognition in video, with implications for privacy, sustainability, and broader ethical considerations in AI. Let's discuss these aspects in detail:

- **Privacy Concerns**:

  Zero-Shot Learning: Zero-shot learning enables models to recognize actions without the need for large labeled datasets. This can be beneficial for privacy because it reduces the reliance on personally identifiable data. In applications like surveillance or video analysis, where privacy concerns are paramount, zero-shot learning can be used to extract information from video feeds without storing sensitive data about individuals.

Few-Shot Learning: Similar to zero-shot learning, few-shot learning also reduces the dependence on extensive labeled data. It allows AI systems to adapt quickly to new tasks with only a few examples. This adaptability can be applied to privacy-preserving scenarios where recognizing specific actions without collecting large amounts of personal data is crucial.

- **Sustainability**:

  Efficiency Gains: Data-efficient techniques contribute to the sustainability of AI systems by reducing the need for large-scale data collection. Collecting and storing massive datasets can have a significant environmental impact due to the energy required for data centers and the carbon footprint associated with data storage. Data-efficient methods consume fewer resources, making AI models more sustainable.

- **Ethical Aspects**:

  Bias Mitigation: Data-efficient techniques can help mitigate biases present in large labeled datasets. When action recognition models are trained with limited data, there is less chance of reinforcing biases present in the data. This can lead to more fair and equitable AI systems.

  Transparency: Using memory-efficient techniques, such as compact models, can make AI systems more interpretable and transparent. Understanding how a model recognizes actions can enhance accountability, as it is essential to know how decisions are made, particularly in applications like law enforcement or healthcare.

  Accessibility: Data-efficient techniques make AI more accessible to smaller organizations and researchers who may not have the resources to collect or label vast datasets. This democratization can lead to more diverse and ethical AI development.

- **Regulatory Compliance**:

  In some regions, there are strict regulations regarding data privacy and protection. Data-efficient methods can help organizations comply with these regulations by minimizing the need to store and process sensitive data.

- **Potential Ethical Concerns**:

  While data-efficient techniques can have ethical advantages, they also have challenges. Models trained on limited data may not perform as well as those trained on larger datasets, leading to reduced accuracy, which can impact fairness and safety.

There is a risk that data-efficient techniques might be used to create AI systems with limited generalization and accuracy, potentially leading to unintended consequences, especially in critical applications like autonomous vehicles or medical diagnosis.

In summary, data-efficient techniques in action recognition have several ethical implications. They can enhance privacy, sustainability, fairness, and transparency in AI systems. However, careful consideration and monitoring are necessary to ensure that these techniques are used responsibly, balancing the benefits with the potential limitations and risks associated with reduced data usage. Ethical guidelines and regulatory frameworks should be in place to guide the development and deployment of data-efficient AI methodologies in action recognition and other domains.

Bibliography

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[2] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of output embeddings for fine-grained image classification. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2927–2936, 2015.

[3] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5297–5307, 2016.

[4] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *Int. Joint Conf. Neural Netw.*, pages 1–8. IEEE, 2020.

[5] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid. Vivit: A video vision transformer. In *Int. Conf. Comput. Vis.*, pages 6836–6846, 2021.

[6] S. Arora and Y. Zhang. Do gans actually learn the distribution? an empirical study. *arXiv preprint arXiv:1706.08224*, 2017.

[7] S. Asghari-Esfeden, M. Sznaier, and O. Camps. Dynamic motion representation for human action recognition. In *Winter Conf. Comput. Vis.*, pages 557–566, 2020.

[8] J. Bao, M. Kudo, K. Kimura, and L. Sun. Robust embedding regression for semi-supervised learning. *Pattern Recognition*, page 109894, 2023.

[9] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Int. Conf. Mach. Learn.*, pages 41–48, 2009.

[10] G. Bertasius, H. Wang, and L. Torresani. Is space-time attention all you need for video understanding? In *Int. Conf. Mach. Learn.*, volume 2, page 4, 2021.

[11] D. Berthelot, N. Carlini, E. D. Cubuk, A. Kurakin, K. Sohn, H. Zhang, and C. Raffel. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *Int. Conf. Learn. Represent.*, 2019.

[12] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Adv. Neural Inform. Process. Syst.*, 32, 2019.

[13] H. Bilen, V. P. Namboodiri, and L. Van Gool. Object and action classification with latent variables. In *Brit. Mach. Vis. Conf.*, volume 2, page 3, 2011.

[14] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *Int. Conf. Comput. Vis.*, volume 2, pages 1395–1402. IEEE, 2005.

[15] B. Brattoli, J. Tighe, F. Zhdanov, P. Perona, and K. Chalupka. Rethinking zero-shot video classification: End-to-end training for realistic applications. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4613–4623, 2020.

[16] P. P. Busto, A. Iqbal, and J. Gall. Open set domain adaptation for image and action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(2):413–429, 2018.

[17] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 961–970, 2015.

[18] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *Eur. Conf. Comput. Vis.*, pages 213–229. Springer, 2020.

[19] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018.

[20] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[21] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha. Synthesized classifiers for zero-shot learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5327–5336, 2016.

[22] J. Chen and C. M. Ho. Mm-vit: Multi-modal video transformer for compressed video action recognition. In *Winter Conf. Comput. Vis.*, pages 1910–1921, 2022.

[23] S. Chen and D. Huang. Elaborative rehearsal for zero-shot action recognition. In *Int. Conf. Comput. Vis.*, pages 13638–13647, 2021.

[24] S. Chen, W. Wang, B. Xia, Q. Peng, X. You, F. Zheng, and L. Shao. Free: Feature refinement for generalized zero-shot learning. In *Int. Conf. Comput. Vis.*, pages 122–131, 2021.

[25] J. Choi, C. Gao, J. C. Messou, and J.-B. Huang. Why can't i dance in the mall? learning to mitigate scene bias in action recognition. In *Adv. Neural Inform. Process. Syst.*, 2019.

[26] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2019.

[27] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 702–703, 2020.

[28] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Eur. Conf. Comput. Vis.*, pages 720–736, 2018.

[29] M. Dehghani, A. Arnab, L. Beyer, A. Vaswani, and Y. Tay. The efficiency misnomer. *arXiv preprint arXiv:2110.12894*, 2021.

[30] M. Dehghani, A. Gritsenko, A. Arnab, M. Minderer, and Y. Tay. Scenic: A jax library for computer vision research and beyond. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 21393–21398, 2022.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 248–255. Ieee, 2009.

[32] T. DeVries and G. W. Taylor. Dataset augmentation in feature space. In *Int. Conf. Learn. Represent.*, 2017.

[33] T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[34] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2625–2634, 2015.

[35] W. Dong, Z. Zhang, and T. Tan. Attention-aware sampling via deep reinforcement learning for action recognition. In *AAAI*, volume 33, pages 8247–8254, 2019.

[36] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[37] Y. Du, C. Yuan, B. Li, L. Zhao, Y. Li, and W. Hu. Interaction-aware spatio-temporal pyramid attention networks for action classification. In *Eur. Conf. Comput. Vis.*, pages 373–389, 2018.

[38] V. Estevam, R. Laroca, D. Menotti, and H. Pedrini. Tell me what you see: A zero-shot action recognition method based on natural language descriptions. *arXiv preprint arXiv:2112.09976*, 2021.

[39] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer. Multiscale vision transformers. In *Int. Conf. Comput. Vis.*, pages 6824–6835, 2021.

[40] H. Fan, Z. Xu, L. Zhu, C. Yan, J. Ge, and Y. Yang. Watching a small portion could be as good as watching all: Towards efficient video classification. In *IJCAI*, 2018.

[41] Y. Fan, F. Tian, T. Qin, J. Bian, and T.-Y. Liu. Learning what data to learn. *arXiv preprint arXiv:1702.08635*, 2017.

[42] Y. Fei, P. Nie, Z. Meng, R. Wattenhofer, and M. Sachan. Beyond prompting: Making pre-trained language models better zero-shot learners by clustering representations. *arXiv preprint arXiv:2210.16637*, 2022.

[43] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. In *Int. Conf. Comput. Vis.*, pages 6202–6211, 2019.

[44] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4768–4777, 2017.

[45] R. Felix, I. Reid, G. Carneiro, et al. Multi-modal cycle-consistent generalized zero-shot learning. In *Eur. Conf. Comput. Vis.*, pages 21–37, 2018.

[46] E. W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *biometrics*, 21:768–769, 1965.

[47] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. *Adv. Neural Inform. Process. Syst.*, 26, 2013.

[48] C. Gan, M. Lin, Y. Yang, G. De Melo, and A. G. Hauptmann. Concepts not alone: Exploring pairwise relationships for zero-shot video activity recognition. In *AAAI*, 2016.

[49] C. Gan, M. Lin, Y. Yang, Y. Zhuang, and A. G. Hauptmann. Exploring semantic inter-class relationships (sir) for zero-shot action recognition. In *Proceedings of the National Conference on Artificial Intelligence*, 2015.

[50] C. Gan, T. Yang, and B. Gong. Learning attributes equals multi-source domain generalization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 87–97, 2016.

[51] J. Gao, Y. Hou, Z. Guo, and H. Zheng. Learning spatio-temporal semantics and cluster relation for zero-shot action recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[52] J. Gao, T. Zhang, and C. Xu. I know the relationships: Zero-shot action recognition via two-stream graph convolutional networks and knowledge graphs. In *AAAI*, volume 33, pages 8303–8311, 2019.

[53] J. Gao, T. Zhang, and C. Xu. Learning to model relationships for zero-shot video classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[54] R. Girdhar and D. Ramanan. Attentional pooling for action recognition. In *Adv. Neural Inform. Process. Syst.*, pages 34–45, 2017.

[55] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 971–980, 2017.

[56] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[57] S. N. Gowda. Human activity recognition using combinatorial deep belief networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1–6, 2017.

[58] S. N. Gowda. Synthetic sample selection for generalized zero-shot learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 58–67, 2023.

[59] S. N. Gowda, A. Arnab, and J. Huang. Optimizing vivit training: Time and memory reduction for action recognition. *arXiv preprint arXiv:2306.04822*, 2023.

[60] S. N. Gowda, P. Eustratiadis, T. Hospedales, and L. Sevilla-Lara. Alba: Reinforcement learning for video object segmentation. *arXiv preprint arXiv:2005.13039*, 2020.

[61] S. N. Gowda, M. Rohrbach, F. Keller, and L. Sevilla-Lara. Learn2augment: Learning to composite videos for data augmentation in action recognition. In *Eur. Conf. Comput. Vis.*, pages 242–259. Springer, 2022.

[62] S. N. Gowda, M. Rohrbach, and L. Sevilla-Lara. Smart frame selection for action recognition. In *AAAI*, volume 35, pages 1451–1459, 2021.

[63] S. N. Gowda and L. Sevilla-Lara. Telling stories for common sense zero-shot action recognition. *arXiv preprint arXiv:2309.17327*, 2023.

[64] S. N. Gowda, L. Sevilla-Lara, F. Keller, and M. Rohrbach. Claster: clustering with reinforcement learning for zero-shot action recognition. In *Eur. Conf. Comput. Vis.*, pages 187–203. Springer, 2022.

[65] S. N. Gowda, L. Sevilla-Lara, K. Kim, F. Keller, and M. Rohrbach. A new split for evaluating true zero-shot action recognition. *arXiv preprint arXiv:2107.13029*, 2021.

[66] R. Goyal, S. Ebrahimi Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Int. Conf. Comput. Vis.*, pages 5842–5850, 2017.

[67] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Int. Conf. Comput. Vis.*, 2017.

[68] Y. Grandvalet, Y. Bengio, et al. Semi-supervised learning by entropy minimization. *CAP*, 367:281–296, 2005.

[69] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, M. Martin, T. Nagarajan, I. Radosavovic, S. K. Ramakrishnan, F. Ryan, J. Sharma, M. Wray, M. Xu, E. Z. Xu, C. Zhao, S. Bansal, D. Batra, V. Cartillier, S. Crane, T. Do, M. Doulaty, A. Erapalli, C. Feichtenhofer, A. Fragomeni, Q. Fu, C. Fuegen, A. Gebreselasie, C. Gonzalez, J. Hillis, X. Huang, Y. Huang, W. Jia, W. Khoo, J. Kolar, S. Kottur, A. Kumar, F. Landini, C. Li, Y. Li, Z. Li, K. Mangalam, R. Modhugu, J. Munro, T. Murrell, T. Nishiyasu, W. Price, P. R. Puentes, M. Ramazanova, L. Sari, K. Somasundaram, A. Southerland, Y. Sugano, R. Tao, M. Vo, Y. Wang, X. Wu, T. Yagi, Y. Zhu, P. Arbelaez, D. Crandall, D. Damen, G. M. Farinella, B. Ghanem, V. K. Ithapu, C. V. Jawahar, H. Joo, K. Kitani, H. Li, R. Newcombe, A. Oliva, H. S. Park, J. M. Rehg, Y. Sato, J. Shi, M. Z. Shou, A. Torralba, L. Torresani, M. Yan, and J. Malik. Ego4d: Around the World in 3,000 Hours of Egocentric Video. In *CVPR*, 2022.

[70] A. Gritsenko, X. Xiong, J. Djolonga, M. Dehghani, C. Sun, M. Lučić, C. Schmid, and A. Arnab. End-to-end spatio-temporal action localisation with video transformers. *arXiv preprint arXiv:2304.12160*, 2023.

[71] W. Guo, J. Wang, and S. Wang. Deep multimodal representation learning: A survey. *IEEE Access*, 7:63373–63394, 2019.

[72] Z. Han, Z. Fu, G. Li, and J. Yang. Inference guided feature generation for generalized zero-shot learning. *Neurocomputing*, 430:150–158, 2021.

[73] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Int. Conf. Comput. Vis.*, 2017.

[74] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016.

[75] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[76] M. Hong, X. Zhang, G. Li, and Q. Huang. Fine-grained feature generation for generalized zero-shot video classification. *IEEE Transactions on Image Processing*, 2023.

[77] D.-A. Huang, V. Ramanathan, D. Mahajan, L. Torresani, M. Paluri, F. F. Li, and J. C. Niebles. What makes a video a video: Analyzing temporal information in video understanding models and datasets. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7366–7375, 06 2018.

[78] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *Eur. Conf. Comput. Vis.*, pages 646–661. Springer, 2016.

[79] A. Iosifidis, A. Tefas, and I. Pitas. Semi-supervised classification of human actions based on neural networks. In *Int. Conf. Pattern Recog.*, pages 1336–1341. IEEE, 2014.

[80] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira. Perceiver: General perception with iterative attention. In *Int. Conf. Mach. Learn.*, pages 4651–4664. PMLR, 2021.

[81] A. Jahanian, L. Chai, and P. Isola. On the" steerability" of generative adversarial networks. In *Int. Conf. Learn. Represent.*, 2019.

[82] Z. Ji, Y. Sun, Y. Yu, J. Guo, and Y. Pang. Semantic softmax loss for zero-shot learning. *Neurocomputing*, 316:369–375, 2018.

[83] Y.-G. Jiang, Z. Wu, J. Wang, X. Xue, and S.-F. Chang. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(2):352–364, 2017.

[84] L. Jing, T. Parag, Z. Wu, Y. Tian, and H. Wang. Videossl: Semi-supervised learning for video classification. In *Winter Conf. Comput. Vis.*, pages 1110–1119, January 2021.

[85] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1725–1732, 2014.

[86] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1725–1732, 2014.

[87] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Int. Conf. Mach. Learn.*, pages 5156–5165. PMLR, 2020.

[88] E. Kodirov, T. Xiang, Z. Fu, and S. Gong. Unsupervised domain adaptation for zero-shot learning. In *Int. Conf. Comput. Vis.*, pages 2452–2460, 2015.

[89] B. Korbar, D. Tran, and L. Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In *Int. Conf. Comput. Vis.*, pages 6232–6242, 2019.

[90] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *Int. Conf. Comput. Vis.*, pages 2556–2563. IEEE, 2011.

[91] C.-W. Kuo, C.-Y. Ma, J.-B. Huang, and Z. Kira. Featmatch: Feature-based augmentation for semi-supervised learning. In *Eur. Conf. Comput. Vis.*, pages 479–495. Springer, 2020.

[92] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[93] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 951–958. IEEE, 2009.

[94] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(3):453–465, 2013.

[95] I. Laptev. On space-time interest points. *Int. J. Comput. Vis.*, 64:107–123, 2005.

[96] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1–8. IEEE, 2008.

[97] D.-H. Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Int. Conf. Mach. Learn.*, volume 3, page 896, 2013.

[98] J. Lemley, S. Bazrafkan, and P. M. Corcoran. Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869, 2017.

[99] K. Li, Y. Wang, P. Gao, G. Song, Y. Liu, H. Li, and Y. Qiao. Uniformer: Unified transformer for efficient spatiotemporal representation learning. *arXiv preprint arXiv:2201.04676*, 2022.

[100] M. Li, Y. Zhang, Z. Li, J. Chen, L. Chen, N. Cheng, J. Wang, T. Zhou, and J. Xiao. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *arXiv preprint arXiv:2308.12032*, 2023.

[101] X. Li, Y. Zhang, S. Bian, Y. Qu, Y. Xie, Z. Shi, and J. Fan. Vs-boost: Boosting visual-semantic association for generalized zero-shot learning. 2023.

[102] Y. Li, M. Liu, and J. M. Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *Eur. Conf. Comput. Vis.*, pages 619–635, 2018.

[103] A. Likas. A reinforcement learning approach to online clustering. *Neural computation*, 11(8):1915–1932, 1999.

[104] C.-C. Lin, K. Lin, L. Wang, Z. Liu, and L. Li. Cross-modal representation learning for zero-shot action recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 19978–19988, 2022.

[105] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2014.

[106] B. Liu, L. Yao, Z. Ding, J. Xu, and J. Wu. Combining ontology and reinforcement learning for zero-shot classification. *Knowledge-Based Systems*, 144:42–50, 2018.

[107] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Eur. Conf. Comput. Vis.*, pages 85–100, 2018.

[108] J. Liu, H. Bai, H. Zhang, and L. Liu. Near-real feature generative network for generalized zero-shot learning. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021.

[109] J. Liu, H. Bai, H. Zhang, and L. Liu. Beyond normal distribution: More factual feature generation network for generalized zero-shot learning. *IEEE MultiMedia*, 2022.

[110] J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3337–3344. IEEE, 2011.

[111] L. Liu, L. Shao, and P. Rockett. Boosted key-frame selection and correlated pyramidal motion-feature representation for human action recognition. *Pattern recognition*, 46(7):1810–1818, 2013.

[112] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Int. Conf. Comput. Vis.*, pages 10012–10022, 2021.

[113] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu. Video swin transformer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3202–3211, 2022.

[114] C. Luo, Z. Li, K. Huang, J. Feng, and M. Wang. Zero-shot learning via attribute regression and class prototype rectification. *IEEE Transactions on Image Processing*, 27(2):637–648, 2017.

[115] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[116] D. Mandal, S. Narayan, S. K. Dwivedi, V. Gupta, S. Ahmed, F. S. Khan, and L. Shao. Out-of-distribution detection for generalized zero-shot action recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9985–9993, 2019.

[117] J. Materzynska, T. Xiao, R. Herzig, H. Xu, X. Wang, and T. Darrell. Something-else: Compositional action recognition with spatial-temporal interaction networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1049–1059, 2020.

[118] D. Meng, X. Peng, K. Wang, and Y. Qiao. frame attention networks for facial expression recognition in videos. In *IEEE Int. Conf. Image Process.*, pages 3866–3870. IEEE, 2019.

[119] P. Mettes and C. G. Snoek. Spatial-aware object embeddings for zero-shot localization and classification of actions. In *Int. Conf. Comput. Vis.*, pages 4443–4452, 2017.

[120] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Adv. Neural Inform. Process. Syst.*, pages 3111–3119, 2013.

[121] A. Mishra, A. Pandey, and H. A. Murthy. Zero-shot learning for action recognition using synthesized features. *Neurocomputing*, 390:117–130, 2020.

[122] A. Mishra, V. K. Verma, M. S. K. Reddy, S. Arulkumar, P. Rai, and A. Mittal. A generative approach to zero-shot and few-shot action recognition. In *Winter Conf. Comput. Vis.*, pages 372–380. IEEE, 2018.

[123] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118, 2023.

[124] M. Monfort, A. Andonian, B. Zhou, K. Ramakrishnan, S. A. Bargal, T. Yan, L. Brown, Q. Fan, D. Gutfreund, C. Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(2):502–508, 2019.

[125] S. Narayan, A. Gupta, F. S. Khan, C. G. Snoek, and L. Shao. Latent embedding feedback and discriminative features for zero-shot classification. *arXiv preprint arXiv:2003.07833*, 2020.

[126] B. Ni, H. Peng, M. Chen, S. Zhang, G. Meng, J. Fu, S. Xiang, and H. Ling. Expanding language-image pretrained models for general video recognition. In *Eur. Conf. Comput. Vis.*, pages 1–18. Springer, 2022.

[127] J. C. Niebles, C.-W. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *Eur. Conf. Comput. Vis.*, pages 392–405. Springer, 2010.

[128] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[129] J. Oh, S. Singh, H. Lee, and P. Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *Int. Conf. Mach. Learn.*, pages 2661–2670, 2017.

[130] M. Pagliardini, P. Gupta, and M. Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of NAACL-HLT*, pages 528–540, 2018.

[131] J. Pan, Z. Lin, X. Zhu, J. Shao, and H. Li. St-adapter: Parameter-efficient image-to-video transfer learning. *Adv. Neural Inform. Process. Syst.*, 35:26462–26477, 2022.

[132] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods Nat. Lang. Process.*, pages 1532–1543, 2014.

[133] T. Perrett, A. Masullo, T. Burghardt, M. Mirmehdi, and D. Damen. Temporal-relational crosstransformers for few-shot action recognition. *arXiv preprint arXiv:2101.06184*, 2021.

[134] A. Piergiovanni, C. Fan, and M. S. Ryoo. Learning latent subevents in activity videos using temporal attention filters. In *AAAI*, 2017.

[135] A. Piergiovanni, W. Kuo, and A. Angelova. Rethinking video vits: Sparse video tubes for joint image and video learning. *arXiv preprint arXiv:2212.03229*, 2022.

[136] Y. Qian, L. Yu, W. Liu, and A. G. Hauptmann. Rethinking zero-shot action recognition: Learning from latent atomic actions. In *Eur. Conf. Comput. Vis.*, pages 104–120. Springer, 2022.

[137] J. Qin, L. Liu, L. Shao, F. Shen, B. Ni, J. Chen, and Y. Wang. Zero-shot action recognition with error-correcting output codes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2833–2842, 2017.

[138] M. Raptis and L. Sigal. Poselet key-framing: A model for human activity recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2650–2657, 2013.

[139] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Empirical Methods Nat. Lang. Process.* Association for Computational Linguistics, 11 2019.

[140] T. Ridnik, E. Ben-Baruch, A. Noy, and L. Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021.

[141] M. Rohrbach, M. Regneri, M. Andriluka, S. Amin, M. Pinkal, and B. Schiele. Script data for attribute-based recognition of composite activities. In *Eur. Conf. Comput. Vis.*, 2012.

[142] A. Roitberg, M. Martinez, M. Haurilet, and R. Stiefelhagen. Towards a fair evaluation of zero-shot action recognition using external data. In *Eur. Conf. Comput. Vis.*, pages 0–0, 2018.

[143] M. Ryoo, A. Piergiovanni, A. Arnab, M. Dehghani, and A. Angelova. Tokenlearner: Adaptive space-time tokenization for videos. *Adv. Neural Inform. Process. Syst.*, 34:12786–12797, 2021.

[144] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4510–4520, 2018.

[145] E. Schonfeld, S. Ebrahimi, S. Sinha, T. Darrell, and Z. Akata. Generalized zero-and few-shot learning via aligned variational autoencoders. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8247–8255, 2019.

[146] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *Int. Conf. Pattern Recog.*, volume 3, pages 32–36. IEEE, 2004.

[147] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Int. Conf. Mach. Learn.*, pages 1889–1897. PMLR, 2015.

[148] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[149] L. Sevilla-Lara, S. Zha, Z. Yan, V. Goswami, M. Feiszli, and L. Torresani. Only time can tell: Discovering temporal data for temporal modeling. In *Winter Conf. Comput. Vis.*, pages 535–544, 2021.

[150] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*, 2015.

[151] Y. Shen, B. Ni, Z. Li, and N. Zhuang. Egocentric activity prediction via event modulated attention. In *Eur. Conf. Comput. Vis.*, pages 197–212, 2018.

[152] Y. Shigeto, I. Suzuki, K. Hara, M. Shimbo, and Y. Matsumoto. Ridge regression, hubness, and zero-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 135–151. Springer, 2015.

[153] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Adv. Neural Inform. Process. Syst.*, pages 568–576, 2014.

[154] A. Singh, O. Chakraborty, A. Varshney, R. Panda, R. Feris, K. Saenko, and A. Das. Semi-supervised action recognition with temporal contrastive learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 10389–10399, 2021.

[155] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

[156] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. *Adv. Neural Inform. Process. Syst.*, 26, 2013.

[157] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[158] S. Sudhakaran and O. Lanz. Attention is all we need: Nailing down object-centric attention for egocentric activity recognition. *arXiv preprint arXiv:1807.11794*, 2018.

[159] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid. Videobert: A joint model for video and language representation learning. In *Int. Conf. Comput. Vis.*, pages 7464–7473, 2019.

[160] D. Sun, D. Vlasic, C. Herrmann, V. Jampani, M. Krainin, H. Chang, R. Zabih, W. T. Freeman, and C. Liu. Autoflow: Learning a better training set for optical flow. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[161] X. Sun, M. Chen, and A. Hauptmann. Action recognition via local descriptors and holistic features. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 58–65. IEEE, 2009.

[162] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1199–1208, 2018.

[163] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.

[164] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2818–2826, 2016.

[165] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017.

[166] C. Thurau and V. Hlavác. Pose primitive based human action recognition in videos or still images. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1–8. IEEE, 2008.

[167] Z. Tong, Y. Song, J. Wang, and L. Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *arXiv preprint arXiv:2203.12602*, 2022.

[168] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Int. Conf. Comput. Vis.*, pages 4489–4497, 2015.

[169] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Adv. Neural Inform. Process. Syst.*, 30, 2017.

[170] V. K. Verma, G. Arora, A. Mishra, and P. Rai. Generalized zero-shot learning via synthesized examples. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4281–4289, 2018.

[171] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[172] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3169–3176, 2011.

[173] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Int. Conf. Comput. Vis.*, pages 3551–3558, 2013.

[174] J. Wang, X. Yang, H. Li, L. Liu, Z. Wu, and Y.-G. Jiang. Efficient video transformers with spatial-temporal token selection. In *Eur. Conf. Comput. Vis.*, pages 69–86. Springer, 2022.

[175] J. Wang, Z. Yang, X. Hu, L. Li, K. Lin, Z. Gan, Z. Liu, C. Liu, and L. Wang. Git: A generative image-to-text transformer for vision and language. *Transactions of Machine Learning Research*, 2023.

[176] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Eur. Conf. Comput. Vis.*, pages 20–36. Springer, 2016.

[177] Q. Wang and K. Chen. Zero-shot visual recognition via bidirectional latent embedding. *Int. J. Comput. Vis.*, 124(3):356–383, 2017.

[178] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7794–7803, 2018.

[179] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[180] J. Wu, T. Zhang, Z. Zhang, F. Wu, and Y. Zhang. Motion-modulated temporal fragment alignment network for few-shot action recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9151–9160, 2022.

[181] W. Wu, D. He, X. Tan, S. Chen, and S. Wen. Multi-agent reinforcement learning based frame sampling for effective untrimmed video recognition. In *Int. Conf. Comput. Vis.*, pages 6222–6231, 2019.

[182] Z. Wu, C. Xiong, Y.-G. Jiang, and L. S. Davis. Liteeval: A coarse-to-fine framework for resource efficient video recognition. In *Adv. Neural Inform. Process. Syst.*, pages 7778–7787, 2019.

[183] Z. Wu, C. Xiong, C.-Y. Ma, R. Socher, and L. S. Davis. Adaframe: Adaptive frame selection for fast video recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1278–1287, 2019.

[184] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele. Latent embeddings for zero-shot classification. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 69–77, 2016.

[185] Y. Xian, T. Lorenz, B. Schiele, and Z. Akata. Feature generating networks for zero-shot learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5542–5551, 2018.

[186] Y. Xian, B. Schiele, and Z. Akata. Zero-shot learning-the good, the bad and the ugly. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4582–4591, 2017.

[187] Y. Xian, S. Sharma, B. Schiele, and Z. Akata. f-vaegan-d2: A feature generating framework for any-shot learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 10275–10284, 2019.

[188] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3485–3492. IEEE, 2010.

[189] G.-S. Xie, L. Liu, F. Zhu, F. Zhao, Z. Zhang, Y. Yao, J. Qin, and L. Shao. Region graph embedding network for zero-shot learning. In *Eur. Conf. Comput. Vis.*, pages 562–580. Springer, 2020.

[190] X. Xiong, A. Arnab, A. Nagrani, and C. Schmid. M&m mix: A multimodal multiview transformer ensemble. *arXiv preprint arXiv:2206.09852*, 2022.

[191] X. Xu, T. Hospedales, and S. Gong. Transductive zero-shot action recognition by word-vector embedding. *Int. J. Comput. Vis.*, 123(3):309–333, 2017.

[192] X. Xu, T. M. Hospedales, and S. Gong. Multi-task zero-shot action recognition with prioritised data augmentation. In *Eur. Conf. Comput. Vis.*, pages 343–359. Springer, 2016.

[193] S. Yan, X. Xiong, A. Arnab, Z. Lu, M. Zhang, C. Sun, and C. Schmid. Multiview transformers for video recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3333–3343, 2022.

[194] A. Yang, A. Nagrani, P. H. Seo, A. Miech, J. Pont-Tuset, I. Laptev, J. Sivic, and C. Schmid. Vid2seq: Large-scale pretraining of a visual language model for dense video captioning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023.

[195] F.-E. Yang, Y.-H. Lee, C.-C. Lin, and Y.-C. F. Wang. Semantics-guided intra-category knowledge transfer for generalized zero-shot learning. *Int. J. Comput. Vis.*, pages 1–15, 2023.

[196] J. Ye, J. Gao, Q. Li, H. Xu, J. Feng, Z. Wu, T. Yu, and L. Kong. Zerogen: Efficient zero-shot learning via dataset generation. *arXiv preprint arXiv:2202.07922*, 2022.

[197] J. Ye, Y. Xue, L. R. Long, S. Antani, Z. Xue, K. C. Cheng, and X. Huang. Synthetic sample selection via reinforcement learning. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part I 23*, pages 53–63. Springer, 2020.

[198] Z. Ye, Y. Geng, J. Chen, J. Chen, X. Xu, S. Zheng, F. Wang, J. Zhang, and H. Chen. Zero-shot text classification via reinforced self-training. In *Assoc. Comput. Linguistics*, pages 3014–3024, 2020.

[199] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2678–2687, 2016.

[200] J. Yoon, S. Arik, and T. Pfister. Data valuation using reinforcement learning. In *Int. Conf. Mach. Learn.*, pages 10842–10851. PMLR, 2020.

[201] Y. Yu, Z. Ji, J. Han, and Z. Zhang. Episode-based prototype generating network for zero-shot learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 14035–14044, 2020.

[202] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4694–4702, 2015.

[203] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Int. Conf. Comput. Vis.*, 2019.

[204] S. Yun, S. J. Oh, B. Heo, D. Han, and J. Kim. Videomix: Rethinking data augmentation for video classification. *arXiv preprint arXiv:2012.03457*, 2020.

[205] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer. S4l: Self-supervised semi-supervised learning. In *Int. Conf. Comput. Vis.*, pages 1476–1485, 2019.

[206] C. Zhang, M. Jin, Q. Yu, H. Xue, and X. Jin. Metric learning for projections bias of generalized zero-shot learning. *arXiv preprint arXiv:2309.01390*, 2023.

[207] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[208] H. Zhang, Y. Hao, and C.-W. Ngo. Token shift transformer for video classification. In *ACM Int. Conf. Multimedia*, pages 917–925, 2021.

[209] H. Zhang, L. Zhang, X. Qi, H. Li, P. H. Torr, and P. Koniusz. Few-shot action recognition with permutation-invariant attention. In *Eur. Conf. Comput. Vis.* Springer, 2020.

[210] J. Zhang, Q. Li, Y.-a. Geng, W. Wang, W. Sun, C. Shi, and Z. Ding. A zero-shot learning framework via cluster-prototype matching. *Pattern Recognition*, 124:108469, 2022.

[211] L. Zhang, T. Xiang, and S. Gong. Learning a deep embedding model for zero-shot learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2021–2030, 2017.

[212] P. Zhang, J. Xue, C. Lan, W. Zeng, Z. Gao, and N. Zheng. Adding attentiveness to the neurons in recurrent neural networks. In *Eur. Conf. Comput. Vis.*, pages 135–151, 2018.

[213] S. Zhang, S. Guo, L. Wang, W. Huang, and M. R. Scott. Knowledge integration networks for action recognition. *arXiv preprint arXiv:2002.07471*, 2020.

[214] Y. Zhang, Y. Bai, H. Wang, Y. Xu, and Y. Fu. Look more but care less in video recognition. *Adv. Neural Inform. Process. Syst.*, 35:30813–30825, 2022.

[215] Y. Zhang, G. Jia, L. Chen, M. Zhang, and J. Yong. Self-paced video data augmentation by generative adversarial networks with insufficient samples. In *ACM Int. Conf. Multimedia*, MM '20, page 1652–1660, New York, NY, USA, 2020. Association for Computing Machinery.

[216] M. Zhao, Y. Yu, X. Wang, L. Yang, and D. Niu. Search-map-search: A frame selection paradigm for action recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 10627–10636, 2023.

[217] X. Zhao, Y. Shen, S. Wang, and H. Zhang. Generating diverse augmented attributes for generalized zero shot learning. *Pattern Recognition Letters*, 2023.

[218] Z. Zhao and A. M. Elgammal. Information theoretic key frame selection for action recognition. In *Brit. Mach. Vis. Conf.*, pages 1–10, 2008.

[219] S. Zheng, S. Chen, and Q. Jin. Few-shot action recognition with hierarchical matching and contrastive learning. In *Eur. Conf. Comput. Vis.*, pages 297–313. Springer, 2022.

[220] Y.-D. Zheng, Z. Liu, T. Lu, and L. Wang. Dynamic sampling networks for efficient action recognition in videos. *IEEE Transactions on Image Processing*, 29:7970–7983, 2020.

[221] L. Zhu, D. Tran, L. Sevilla-Lara, Y. Yang, M. Feiszli, and H. Wang. Faster recurrent networks for efficient video classification. In *AAAI*, volume 34, pages 13098–13105, 2020.

[222] Y. Zhu, Y. Long, Y. Guan, S. Newsam, and L. Shao. Towards universal representation for unseen action recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9436–9445, 2018.

[223] Y. Zou, J. Choi, Q. Wang, and J. Huang. Learning representational invariances for data-efficient action recognition. *CoRR*, abs/2103.16565, 2021.