THE UNIVERSITY *of* EDINBURGH

Computation and Data Efficient Techniques for Training Computer Vision Methods

Mustafa Taha Koçyiğit

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2023

Abstract

Increased use of data and computation have been the main drivers in Deep Learning for improving performance, solving previously unsolvable problems and unlocking new capabilities. At the same time cost of training and obtaining annotations have been increasing exponentially for large scale models where in certain circumstances it is not even feasible for many research organizations to reproduce ground breaking results in the field let alone improve them. This thesis focuses on problems that arise from scaling computational and data requirements for training computer vision algorithms and proposes efficient training techniques in three scenarios. First where data is available but labels are very limited, second where data is abundant but no labels are available and finally where both labels and data is abundant but computation is limited. We provided model-agnostic and efficient solutions for these urgent bottlenecks.

Obtaining labeled data is one of the most expensive parts of training deep learning methods which are notoriously data hungry. We have identified that batch normalization statistics calculated from limited labeled data does not reflect the true data distribution and prevent training deep learning models effectively. We proposed a semi-supervised learning method where batch normalization calculations are augmented with unlabeled examples to alleviate this problem and train deep learning models more accurately.

Recently self-supervised learning has been used to learn augmentation invariant representations from unlabelled data. These representations have been shown to effectively transfer to tasks where only limited labeled data is available. This eliminates the data bottleneck while exacerbating the computational bottleneck since these methods are much more computationally hungry than their supervised alternatives. In order to accelerate the training for these methods we introduce three strategies: a progressive augmentation and resolution schedule, a fast hard augmentations mining scheme and a matching accelerated learning rate schedule. We show that our training strategies reduce the cost of training for many self-supervised methods while maintaining the same accuracy as standard training.

In problems where annotation cost is low and large scale labeled data is available more sophisticated model architectures like Vision transformers can be used which require longer and computationally more demanding training settings. We exploit the intuition that not all the samples carry similar amount of information, and introduce a fast online importance sampling technique that allows faster training. We show that we can accelerate the supervised training on many datasets and backbone architectures while reaching the same level of accuracy as standard training.

Acknowledgements

In the name of Allah the most Gracious and most Merciful. I praise him for giving me the opportunity, passion and love to seek knowledge and ask him to give me the most beneficial knowledge and use it to produce useful works.

Every great scientist sits on the shoulder of giants, and since the sweetness of the honey comes not from bee but the kaleidoscope of flowers it visits. So too, I find myself indebted to the significant figures who nurtured my knowledge journey, lighting the path for me to follow.

I would like to extend my heartfelt appreciation to my supervisors Hakan Bilen and Timothy Hospedales. Their invaluable feedback and astute insights not only enhanced my work, but also fostered a space for me to evolve as a researcher. In many ways, I was an unconventional student, yet they embraced my uniqueness with patience and understanding. Their role in my academic journey has been indispensable. I am deeply indebted to the Turkish Ministry of Education for their financial support throughout my PhD. Additionally, the Turkish General Education Attaché in Edinburgh has been a pillar of support, and for this, I am immensely grateful.

I would like to show my gratitude and respect towards my Family who are the coolness in my eyes. My dear father Yakup and my mother Sabire Koçyiğit who cared for me when I was little and helpless and raised me up, cared for me and supported my education on so many levels. I am grateful to be married with dearest friend Meryem Güzide who was always with me in every sense of the word supporting me. I also owe sincere thanks to my little daughter Amine Zehra and my son Mahir Yakup who gave me joy every day and reminded me that little lives depended on me and gave me the grit to push forward. I thank my friend and brother Muhammed Yusuf for always being there for me when I needed him. I also thank my sisters Rana, Sena and Sude for their sincere prayers.

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.

(*Mustafa Taha Koçyiğit*)

Table of Contents

Chapter 1

Introduction

The rapid increase in compute resources was the main driver for AI development in the past decade Fig. 1.1. The bitter lesson in prior AI progress Sutton (2019) it that general methods that can better leverage compute resources ultimately were responsible for the major breakthroughs in the field. It has been shown that scalable architectures like self-attention and convolution can be trained with more computational resources on larger datasets to improve performance in many tasks including translation, language modeling, image processing, and speech recognition Hestness et al. (2017). While the limits of scale for these architectures are still unknown Kaplan et al. (2020), some have hypothesized that further scaling of data and computation will solve current problems in the field Branwen (2021).

Recent progress in large scale self-supervised pre-training in language modelling (GPT-3) Brown et al. (2020) and computer vision (SEER) Goyal et al. (2021, 2022) have demonstrated many emergent phenomenon like few-shot learning, robustness and generalization. In both cases pre-training models that have more than a billion parameters with self-supervised learning resulted in models that can solve many difficult downstream tasks with minimal supervision. Surprising scaling effects have also been demonstrated for Autoregressive Text-to-Image models Yu et al. (2022) and Acoustic models Droppo and Elibol (2021).

1.1 Research Problem

However the cost of training large scale models has been increasing exponentially reaching millions of dollars for just training one of the largest modelsHernandez and Brown (2020). This prevents many research organizations from reproducing and

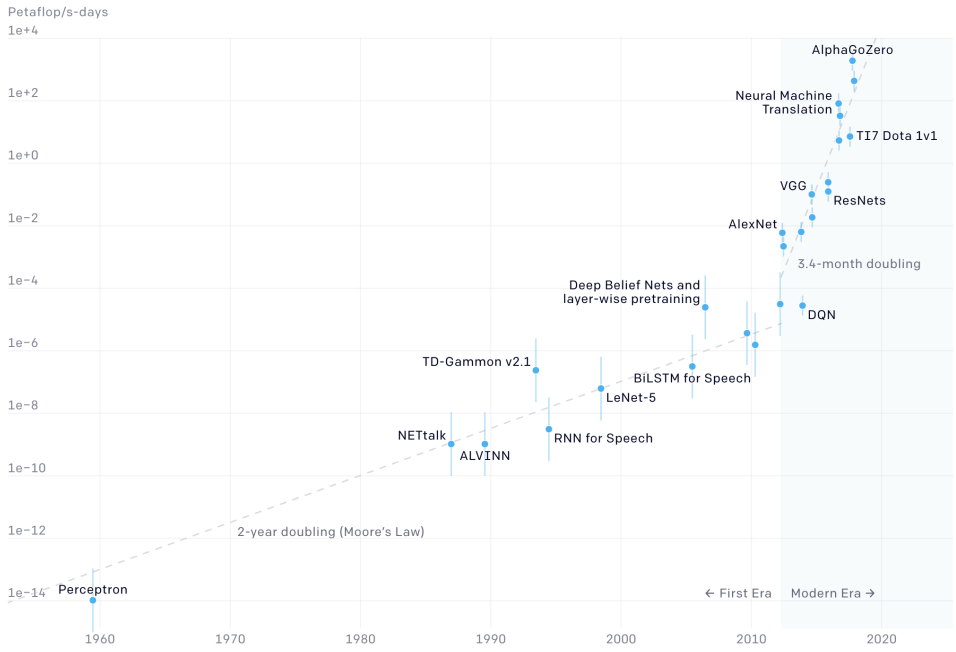


Figure 1.1: The computational requirement for AI tasks has been increasing exponentially. The computational demands have doubled every 3.4 months in the Modern which started with AlexNet Krizhevsky et al. (2009). Source: Amodei and Hernandez (2018)

improving ground breaking results in the field and limits the reach of these amazing developments to select institutions thus slowing down progress.

This thesis seeks to address this challenge by proposing and examining novel, model-agnostic efficient training methodologies. Our main aim is to drastically cut down the computational costs and resources needed for training large-scale artificial intelligence models without compromising their accuracy.

The comprehensive breadth of this thesis is evident in the three distinct contributions it offers. Each contribution is meticulously designed to apply to different regimes of machine learning tasks: dense prediction problems where data is abundant but high annotation costs are high, self-supervised learning tasks with abundant data and no annotation costs, and supervised learning with large datasets and low-cost annotations. This wide-ranging scope is a testament to the versatility and adaptability of the proposed model-agnostic efficient training methods, thereby underscoring the potential of these methods to significantly accelerate AI development across a spectrum of different computational challenges.

The quest to decrease the cost of training neural networks over the last decade

has primarily focused on three computational trends: Moore's Law, Cloud Computing which enabled sharing of the resources efficiently and increasing hardware utilization and finally Parallelization of computation using specialized hardware (GPUs and TPUs) and software (CUDA NVIDIA et al. (2020) and PyTorch Paszke et al. (2019)). However, improvements in algorithmic efficiency played a far more significant role in reducing cost of training Hernandez and Brown (2020) making this an attractive target for further research. Optimization improvements for large batch size training Akiba et al. (2017) allowed large number of accelerators to be used in parallel. Architecture innovations like batch normalizationIoffe and Szegedy (2015) and residual connectionsHe et al. (2015) allowed much deeper networks to be trained effectively while neural architecture search Zoph and Le (2016) and architecture scaling Tan and Le (2019) allowed learned architectures to be used increasing training efficiency.

Improvements for algorithmic efficiency are urgently required due to the increased demand for larger models, larger datasets and smaller number of annotations. Dense prediction tasks have high annotation costs while self-supervised learning doesn't have any annotation cost but it is at least an order of magnitude more computationally demanding. In tasks where the annotation is relatively cheap like classification much larger datasets and more sophisticated architectures like Vision Transformers Dosovitskiy et al. (2020) are being used which are more computationally demanding. In each setting techniques that reduce training cost while not being tied down to any particular model or architecture offer the greatest potential. In this thesis we focus on scaling bottlenecks for computation and data and provide novel model-agnostic efficient training methods for three computer vision tasks that reduce the training cost significantly while maintaining model accuracy.

1.2 Outline

Chapter 2 "Unsupervised Batch Normalization" focuses on dense prediction problems where obtaining annotations are expensive and proposes a semi-supervised learning method where batch normalization calculations are augmented with unlabeled examples. We show that this method allows using half the number of annotated examples to reach comparable performance to supervised baselines. This work has been published in the 2020 Proceedings of IEEE/CVF Conference for Computer Vision and Patter Recognition, Workshop for Visual Learning with Limited Labels.

Chapter 3 "Accelerating Self-Supervised Learning via Efficient Training Strategies"

focuses on the self-supervised learning task and proposes three strategies for accelerating training: a progressive augmentation and resolution schedule, a fast hard augmentations mining scheme and a matching accelerated learning rate schedule. We show that our method can speed up training up to 2.7 times without losing accuracy. This work has been published in the 2023 Proceedings of IEEE/CVF Winter Conference on Applications of Computer Vision.

Chapter 4 "Fast Online Minibatch Selection" focuses on supervised learning on tasks with large datasets and low cost for annotation and introduces a fast online importance sampling technique that allows faster training. We subsample from the minibatch the most useful examples using a low cost approximation and still reach baseline performance. We show that our method can accelerate training up to 1.7 times for many models and a wide range of datasets.

In Chapter 5 we summarize our findings and discuss the limitations of our work and how our work can be extended in the future. Our research has made significant strides towards efficient AI model training, shedding light on potential areas for cost and resource optimization. In acknowledging the limitations of our current methods, we open the door to continued exploration and refinement of these techniques, laying the groundwork for future development in this dynamic and critical field. Our versatile approach to addressing various machine learning regimes further accentuates the applicability and potential of our research in advancing AI development.

Chapter 2

Unsupervised Batch Normalization

Abstract

Batch Normalization is a widely used tool in neural networks to improve the generalization and convergence of training. However, on small datasets due to the difficulty of obtaining unbiased batch statistics it cannot be applied effectively. In some cases, even if there is only a small labeled dataset available, there are larger unlabeled datasets from the same distribution. We propose using such unlabeled examples to calculate batch normalization statistics, which we call Unsupervised Batch Normalization (UBN). We show that using unlabeled examples for batch statistic calculations results in a reduction of the bias of the statistics, as well as regularization leveraging the data manifold. UBN is easy to implement, computationally inexpensive and can be applied to a variety problems. We report results on monocular depth estimation, where obtaining dense labeled examples is difficult and expensive. Using unlabeled samples, and UBN, we obtain an increase in accuracy of more than 6% on the KITTI dataset, compared to using traditional batch normalization only on the labeled samples.

2.1 Introduction

Large-scale labeled data and modern learnable representations have driven progress on many computer vision problems in the last decade. The dependency on labeled data is a fundamental bottleneck for many vision problems, as obtaining ground truth annotations can either be extremely expensive (*e.g.* pixel-wise annotations for semantic segmentation) or require specialized equipment and controlled environments (*e.g.* depth estimation, 3D pose estimation, optical flow).

In these cases, research often resorts to different strategies. A common approach is to pre-train from scratch a deep network on "realistic" synthetic data and then fine-tune on the available labeled data. This strategy is commonly used for example by state-of-the-art optical flow methods Sun et al. (2018). While this method alleviates overfitting, it requires creating such realistic data, and does not provide the opportunity to leverage the availability of unlabeled data. While data may be difficult and expensive to label, sometimes additional unlabeled data samples (*e.g*. images from the same distribution) can be easily available for free. Another popular technique is transfer learning, where large models are pre-trained on self-supervised tasks (*e.g*. Zhang et al. (2017b)) and then fine-tuned on the small amount of labeled data available. Although the pre-training stage helps the optimization starting point, the fine-tuning stage still risks overfitting to the small amount of labeled data. Another effective strategy is semi-supervised learning Zhu (2005), which aims at learning both from labeled and unlabeled data at the same time. This approach is successful but is usually not general, and requires specific adaptation to each vision problem.

In this paper, we propose a method that, we hypothesize, could be generalized, as it does not require adaptation to any new vision problem, leverages additional unlabeled data and does not require fine-tuning. In particular, we adapt the widely used batch normalization (BN) technique to use unlabeled data.

BN is widely used on many architectures to normalize the statistics of the features, and therefore stabilize the optimization process. However, these statistics can also overfit to the labeled dataset and not reflect the statistics that would normalize the actual dataset. This would cause new images to appear as out of distribution examples for the model.

Instead, we use unlabeled samples, which typically are a much larger number than the labeled ones, to compute the normalization statistics, in a process we call *Unsupervised Batch Normalization* (UBN). We test our method in the problem of monocular depth estimation, and obtain more than 6% improvement over the baseline of using vanilla batch normalization. While our results show promising outcomes in the realm of monocular depth estimation, further investigations are required to establish the effectiveness of UBN across a wider range of vision tasks. We hope that the simplicity of our method will make it a useful component of future learning systems.

2.2 Related Work

Normalization methods. Batch normalization Ioffe and Szegedy (2015) is the most standard normalization method in the state-of-the-art classification architectures on the ImageNet LSVRC Tan and Le (2019); Xie et al. (2017). Recently, new alternatives have been developed to extend its scope in different ways and making BN more general. Batch Renormalization Ioffe (2017) enables training with smaller batch sizes and correlated batches by calculating moving averages of batch statistics during training. Mode Norm Deecke et al. (2018) enables the use of BN for multi-modal data. This is done by calculating multiple batch statistics and adaptively normalizing examples through a gating function and calculating weighted normalization statistic for each example. Instance Norm Ulyanov et al. (2016) has been used for style transfer and normalizes the style of the content image Huang and Belongie (2017) by calculating statistics over each channel and sample in the dataset. Adaptive Instance Norm Huang and Belongie (2017) transfers style between two images by calculating the batch statistics form one example and applying them to another image. Weight Norm Salimans and Kingma (2016) conditions the optimization problem by reparameterizing the weight values into magnitude and direction while Layer Normalization Ba et al. (2016) calculates normalization statistics per example for feature maps in each layer to enable normalization in recurrent models.

While all these variants make BN more usable and stable, to our knowledge, our is the first to extend BN to leverage unlabeled data.

Understanding Batch Normalization. While BN has been widely and successfully used, there remains some questions about why and when it works. The initial explanation of Ioffe and Szegedy Ioffe and Szegedy (2015) that BN reduces internal co-variate shift has been questioned and explored in recent work.

For example, Bjorck et. al. Bjorck et al. (2018) argues that BN enables training with larger learning rates by preventing exploding and vanishing gradients. They also show that unnormalized networks have ill-conditioned activations due to random initialization which results in large singular values of the activation matrices and predictions which are independent from the inputs. Similarly Santurkar et. al. Santurkar et al. (2018) argue that the loss landscape of the normalized networks are smoother. Luo et. al. Luo et al. (2018a) showed that batch normalization applies population normalization with data dependant gamma decay that promotes uncorrelated features.

Monte Carlo Batch Normalization Teye et al. (2018) shows that uncertainty estimates

can be obtained from networks with BN layers by sampling batch statistics.

Training on small datasets. Training the classification networks on ImageNet requires millions of balanced labeled images which many other vision problems lack. Many semi-supervised methods that are designed for classification problems Berthelot et al. (2019); Xie et al. (2019); Lee (2013); Tarvainen and Valpola (2017) are not readily applicable to regression due to the difficulty in generating pseudo-labels and augmentation strategies. Training unsupervised methods like geometric consistency based unsupervised trainingKuznietsov et al. (2017); Guizilini et al. (2019); Amiri et al. (2019) with supervised training in order to mitigate labeled data requirements has also results in problem specific solutions. While pre-training is a widely used technique in segmentation and detection Chen et al. (2018) when the target tasks is not similar to the source tasks the transfer learning has limited success Zamir et al. (2018). Another approach is to use synthetic data to pre-train the networks and fine-tune on real data. For tasks like optical flow Sun et al. (2018) and depth estimation Kundu et al. (2018); Luo et al. (2018b); Guo et al. (2018) synthetic data is much more accessible and does not contain the various artifacts and sampling noises that real data has. However, this approach suffers from the domain shift problem where the data distribution of synthetic and real data can differ.

2.3 Method

Since our method is a simple modification of the standard BN method, we start recalling BN and then describe our UBN method.

2.3.1 Batch Normalization

Given an input batch of height H and width W with N samples and C channels $x \in \mathbb{R}^{N\times C\times H\times W}$, BN normalizes the mean and standard deviation for each individual feature channel c during training:

$$\mathrm{BN}(x)_c = \gamma_c \left(\frac{x_c - \mu(x_c)}{\sigma(x_c)} \right) + \beta_c \tag{2.1}$$

where $\gamma, \beta \in \mathbb{R}^C$ are affine parameters learned from data; μ, σ are the mean and standard deviation, computed across batch size and spatial dimensions independently for each feature channel.

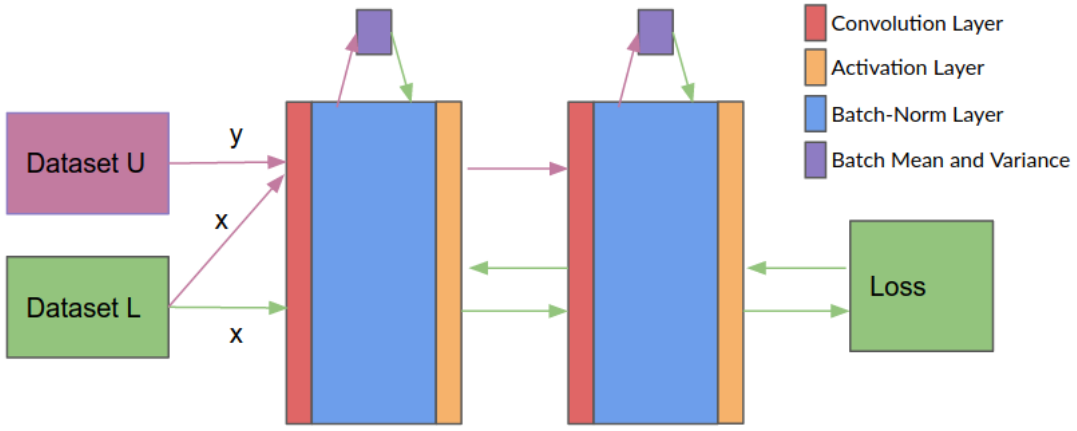


Figure 2.1: Unsupervised Batch Norm disentangles the normalization from the weight updates by separating them into 2 steps. In the first step a batch is sampled from Dataset U and Dataset L and the normalization statistics are updated. In the second step the sampled batch from Dataset L is used to updated the weights of the network.

The moving average of the mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$ are updated using a momentum rate λ during training and used to normalize feature maps during testing.

$$\hat{\mu_c} = \lambda\hat{\mu_c} + (1-\lambda)\mu(x_c) \tag{2.2}$$

$$\hat{\sigma_c} = \lambda\hat{\sigma_c} + (1-\lambda)\sigma(x_c) \tag{2.3}$$

2.3.2 Unsupervised Batch Normalization

UBN is based on updating first the batch statistics than the weights. Given an batch $x \in \mathbb{L}$ from labeled dataset $\mathbb{L}$ and a batch $y \in \mathbb{U}$ from unlabeled dataset $\mathbb{U}$ the first step builds a combined batch $n = \{x, y\}$ and makes a forward pass to update the normalization statistics. The second step makes forward-backward pass while using x and the updated normalization statistics calculated at the earlier step. 2.1

UBN normalizes the mean and standard deviation for each individual feature channel with:

$$\mathrm{UBN}(x_c) = \gamma\left(\frac{x_c - \mu(n_c)}{\sigma(n_c)}\right) + \beta \tag{2.4}$$

where $\mu(n_c), \sigma(n_c)$ are the mean and standard deviation, computed across batch size and spatial dimensions independently for each feature channel from the combined batch n. The $\hat{\mu_c}, \hat{\sigma_c}$ values are calculated similarly to BN using a moving average.

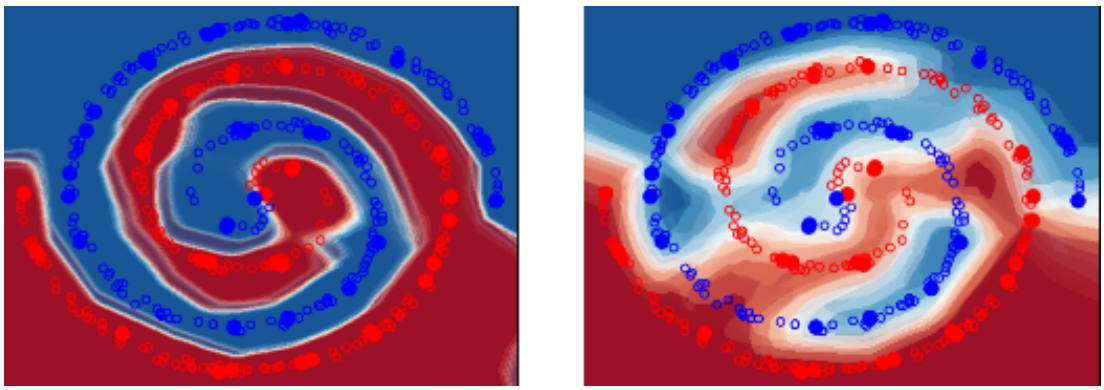


Figure 2.2: Results of classifying with BN and UBN using the Spiral dataset, with 25 labeled points of each class (filled) and 200 unlabeled points per class which (hollow). Using UBN (left) the model can interpolate between these points while BN (right) learns islands around labeled examples.

2.3.3 Analysing UBN with a Toy Example

We illustrate the principles behind UBN and how it differs from BN using a toy example. We train a simple 5-layer fully connected network using BN layers on the Spiral Dataset[1], with 50 labeled examples and 400 unlabeled examples. Using this simple dataset allows us to do a deeper analysis which would not be straightforward using the depth estimation task.

We see from Fig. 2.2 that supervised training with BN learns the distribution of the labeled examples well and therefore learns accurate decision boundaries. However, those decision boundaries do not reflect the true data distribution. Instead, using UBN the network learns to align its decision boundary with the data manifold and learns to interpolate between the labeled examples by utilizing the batch statistics of unlabeled examples. In the coming sections we will examine the mechanism behind UBN through this example.

2.3.3.1 Bias in Batch Statistics

Batch statistics are calculated from labeled examples in BN. When there is only a limited number of labeled examples, this may cause the normalization statistics to be biased and end up hurting the model performance. We recorded the feature distributions from different layers of regularly spaced points, labeled points and unlabeled points while the network normalization statistics using the unlabeled examples. 2.3.

[1] http://playground.tensorflow.org

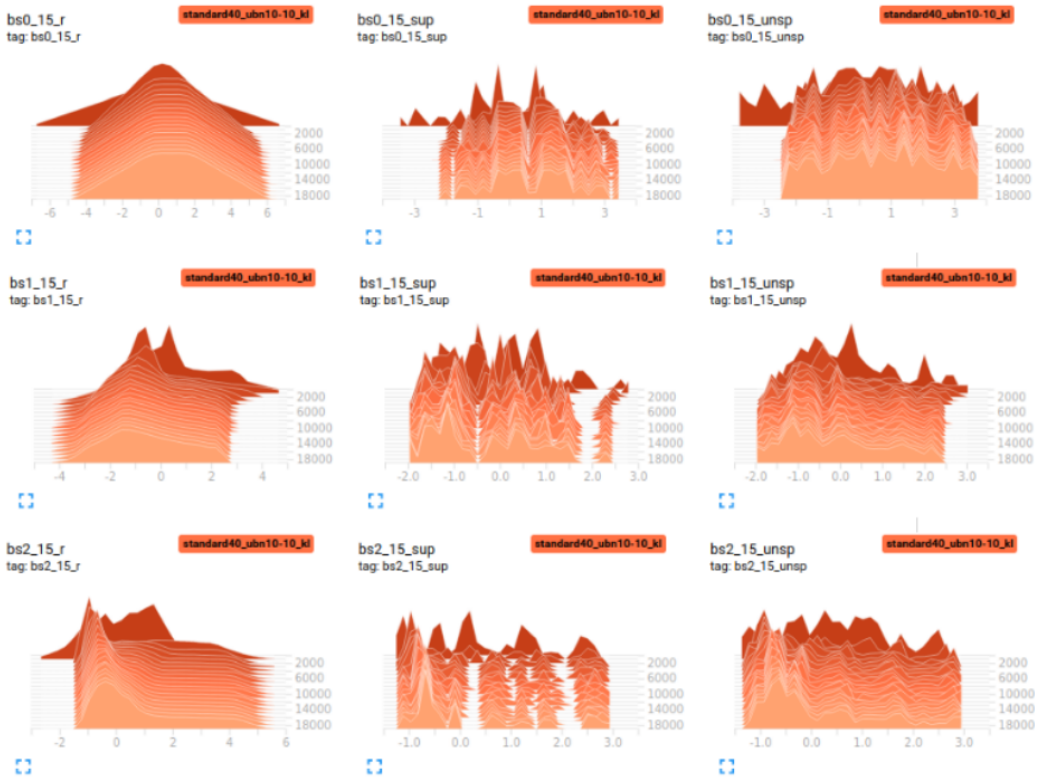


Figure 2.3: Histogram of hidden representations during training of the regularly sampled points (left column) the labeled points (middle column) and the unlabeled points (right column) after first (first row) second (second row) and third (third row) Batch Normalization layers of a random channel.

We have observed from the histogram of feature maps that the distribution of unlabeled and labeled examples are drastically different from each other. Calculating batch statistics with respect to only the labeled examples will cause a significant amount of bias in the normalization values.

In the statistical analysis presented in Table 2.1, we notice significant differences in the distributions of the labeled and unlabeled samples across the layers of the neural network. This is corroborated by various statistical measures employed.

The Pearson Correlation Coefficient, measuring the linear relationship between the labeled and unlabeled samples, varies across the layers and stays close to zero. This implies that there isn't a strong linear relationship between the labeled and unlabeled sample distributions at any of the layers.

The Kullback-Leibler (KL) Divergence and its reverse, which provide a measure of the dissimilarity between the distributions, reveal an interesting pattern. We see that the reverse KL divergence is significantly smaller that the KL divergence. Which

Layer	Pearson Corr.	KL Div.	Rev. KL Div.	Wasserstein	Unsup-Wass.
Layer 1	0.092	0.120	0.068	0.206	0.087
Layer 2	0.126	0.132	0.060	0.253	0.053
Layer 3	0.126	0.129	0.055	0.298	0.044
Layer 4	0.037	0.024	0.017	0.153	0.064

Table 2.1: In this table we compare the distribution of the features in different layers of the network for labeled and unlabeled samples we use the following distances: Pearson Correlation Coefficient quantifies the linear relationship between the two sets of samples; KL Divergence and its reverse provide a measure of the relative entropy between the labeled and unlabeled sets indicating their dissimilarity; Wasserstein Distance provides a measure of the minimum cost of transforming one distribution into the other; Unsup-Wasserstein provides a measure for the minimum cost of transforming the unlabeled distribution into a different subset of the unlabeled distribution. The aim is to use these measures to quantitatively compare the distributions of the two sample sets at each layer of the network and lastly to have a sanity check of the distance values. We have seen that the distribution between the labeled and unlabeled samples is significantly different which provides one of the primary motivations for our method.

means that that the unlabelled data distribution can explain the labeled data distribution much better and many examples of unlabelled distribution are regarded as low value examples.

The Wasserstein Distance, quantifying the minimum cost of transforming one distribution into another, remains consistent across the layers. Interestingly, the Wasserstein distance between unsupervised samples (Unsup-Wasserstein), which measures the minimum cost of transforming the unlabeled distribution into a different subset of the unlabeled distribution, is much smaller than the Wasserstein distance between the labeled and unlabeled sets. This suggests that there is distributional differences between the features of the labeled samples and unlabeled samples.

We also apply the Kolmogorov-Smirnov test for the feature map distributions and find that for %15 of the features in layer 2 and 3 have different distributions which are statistically significant. This confirms our earlier observations for KL divergence, Wasserstein distance and Pearson Correlation. These all indicate that the distribution of the labeled samples can show small sample bias and lead to different distribution statistics when the number of labeled samples are small.

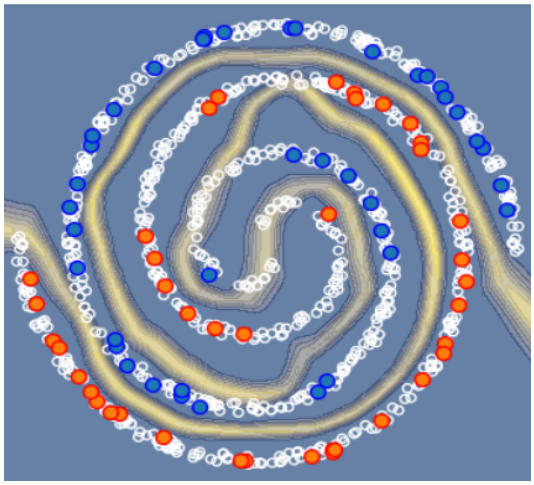


Figure 2.4: Uncertainty in the decision boundaries when unlabeled batch statistics are changed.

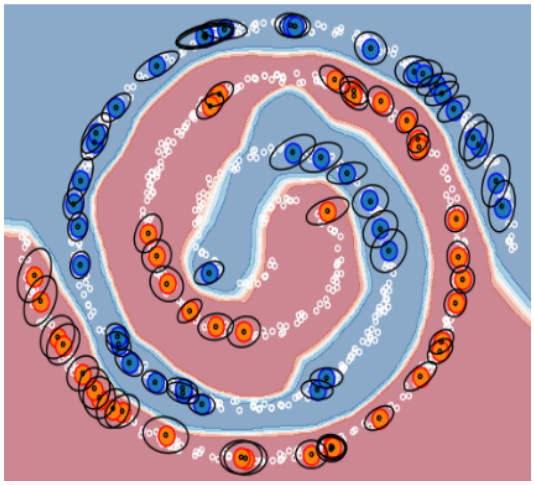


Figure 2.5: We visualize the data driven implicit augmentation via the noise that is induced by changing batch statistics.

We postulate that the difficulty of obtaining correct batch statistics will be an important cause for the failure of Deep Learning methods in problems where the data is limited. Our method alleviates this problem by calculating the normalization values with respect to a larger variety of examples that better reflect the true data distribution.

2.3.3.2 Regularization in UBN

Our understanding is that the regularizing effect of UBN and similarly of BN comes from the noise that is induced by changing batch norm statistics which makes the network invariant to this type of noise. This specific noise is also different from simpler random variations that is applied to gradients Neelakantan et al. (2015) and feature maps Gal and Ghahramani (2016) of the network since it comes from the actual data distribution and extends the data boundaries in the manifold that is predicted by the network. Other method for injecting learned noises are also not applicable in the semi-supervised setting where the small number of labels doesn't allow for such networks to be trained.

We visualized the uncertainty that is induced by changing batch statistics by making multiple predictions using the same inputs while updating the batch statistics Teye et al. (2018). This gives us a distribution of prediction from which we can calculate the standard deviation of our predictions. We observe that the uncertainty of the network follows the data manifold and aligns well with incorrect predictions and regions where there are no data examples 2.4.

We analyse the induced noise of changing batch statistics by making a forward pass of the network using labelled examples saving the network predictions updating the batch statistics and updating the inputs using back propagation in the feature space until the predictions are the same with the earlier batch statistics. This gives a distribution of starting points that have the same effect as changing the batch statistics. We observe that the augmentation follows the data manifold except for regions where the decision boundary is too close the the data manifold where in that case it pushes the decision boundary away from the labeled examples 2.5. Thus, Batch normalization layers apply an implicit augmentation that is dependant on the data manifold.

2.4 Experiments

KITTI dataset Geiger et al. (2013) contains stereo RGB images at 256x832 resolution and ground truth depth maps collected from Velodyne HDL-64E rotating 3D laser scanner at 10Hz. We have utilized the Kitti Raw dataset with 42304 training images and 2480 testing images.

The architecture for our method is based based on the SfMLearner architecture Zhou et al. (2017b) which uses a modified version of the DispNet architecture Mayer et al. (2016). It is a fully convolutional encoder-decoder architecture with skip connections and predictions in multiple resolutions. The building block of the architecture consists of 3x3 convolution operation, BN layers and a ReLU activation. The loss is calculated in different resolution by scaling the depth maps to match the prediction resolutions. The last layer before a prediction a sigmoid activation is used. The result of the sigmoid function is than scaled, shifted and inverted to obtain the depth prediction. The depth prediction D_i for a feature map X_i using a scaling factor of $s > 0$ and a minimum disparity value m such that $0 < m < 1$ can be calculated as follows

$$\hat{D}_i = \frac{1}{s * sigm(X_i) + m} \tag{2.5}$$

The network is trained with a learning rate of 0.0002 using the ADAM optimizer Kingma and Ba (2014) for 160 epochs with batch size of 128. The scaling parameter s is 0.4 and the minimum disparity value m is 0.0125 (max depth 80m) throughout the experiments.

In order to evaluate the effect of unlabeled examples on the BN statistics we compare the performance of the depth prediction network on the KITTI dataset where a certain fraction of the depth maps are missing while we keep the number of images the same.

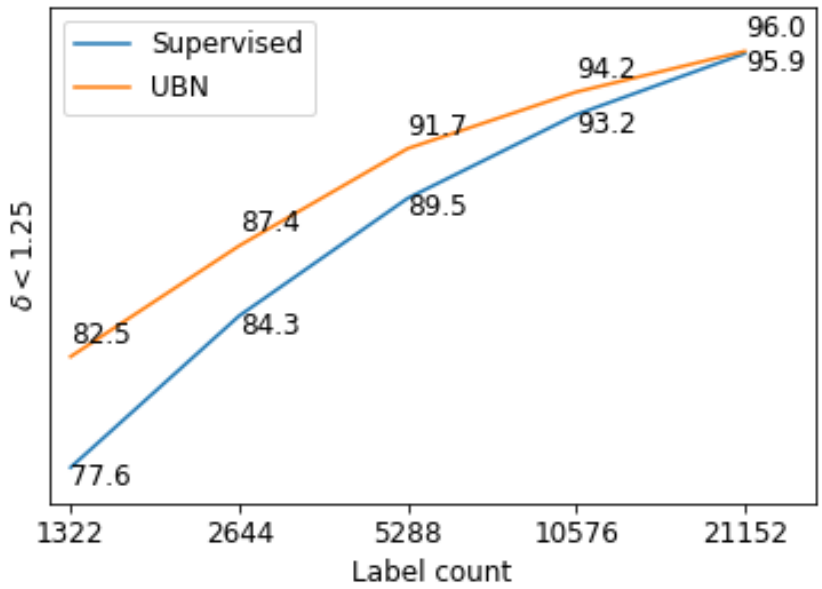


Figure 2.6: UBN shows a larger improvement as the number of labeled images decreases.

Removing depth annotation in this way let us keep the variation in the dataset stable while reducing the total number of annotated examples drastically. The missing depth maps are designed to mimic a depth sensor with lower frame rate.

For the metrics we used accuracy δ with threshold τ : $\delta = max(\frac{\hat{D}_i}{D_i}, \frac{D_i}{\hat{D}_i}) < \tau$ for $\tau = 1.25$.

UBN shows progressively larger improvement as the number of labeled examples shrink 2.6 and outperforms the supervised BN training. If the number of labeled images is close to the unlabeled images the two methods become effectively the same which makes our method more suitable for problems with limited labels.

2.5 Conclusion

We have shown that UBN reduces the bias in batch statistics and applies a regularization that utilizes the data manifold. Our method is easy to implement, computationally inexpensive and is effective in problems where obtaining annotations is difficult.

While these results are promising, it's worth noting that further experimentation and validation will be necessary in future work to support the potential universality of this approach across a variety of problem types, including object recognition, image

segmentation, and pose estimation.

Nonetheless, our research underscores the potential of UBN as a vital tool for enhancing the training and generalization capabilities of neural networks on small datasets, especially when unlabeled data from the same distribution is readily available.

Chapter 3

Accelerating Self-Supervised Learning via Efficient Training Strategies

Abstract

Recently the focus of the computer vision community has shifted from expensive supervised learning towards self-supervised learning of visual representations. While the performance gap between supervised and self-supervised has been narrowing, the time for training self-supervised deep networks remains an order of magnitude larger than its supervised counterparts, which hinders progress, imposes carbon cost, and limits societal benefits to institutions with substantial resources. Motivated by these issues, this paper investigates reducing the training time of recent self-supervised methods by various model-agnostic strategies that have not been used for this problem. In particular, we study three strategies: an extendable cyclic learning rate schedule, a matching progressive augmentation magnitude and image resolutions schedule, and a hard positive mining strategy based on augmentation difficulty. We show that all three methods combined lead up to 2.7 times speed-up in the training time of several self-supervised methods while retaining comparable performance to the standard self-supervised learning setting.

3.1 Introduction

Learning representations without manual human annotations that can be successfully transferred to various downstream tasks has been a long standing goal in machine learning Fukushima and Miyake (1982); Wiskott and Sejnowski (2002). Self-supervised

learning (SSL) aims at learning such representations discriminatively through pretext tasks such as identifying the relative position of image patches Doersch et al. (2015) and solving jigsaw puzzles Noroozi and Favaro (2016). The recent success of SSL methods He et al. (2020); Chen et al. (2020a,c) builds on contrastive learning where the representations are in a latent space invariant to various image transformations such as cropping, blurring and colour jittering. Contrastive learned representations have been shown to obtain on par performance with their supervised counterparts when transferred to various vision tasks including image classification, object detection, semantic segmentation Chen et al. (2020b); Grill et al. (2020), and extended to medical imaging Azizi et al. (2021) as well as multi-view Tian et al. (2020a) and multi-modal learning Radford et al. (2021).

Despite the remarkable progress, an important downside of SSL methods is their high training cost which hampers the development and adoption of these promising techniques. Even the most efficient SSL methods require at least an order of magnitude more computation to reach the performance of supervised methods, *e.g.* training BYOL Grill et al. (2020) requires 23 times more computation resulting from 8.8 times more iterations and 2.6 times more computation at each iteration than its supervised counterpart to reach similar accuracy. The large training cost is largely due to the challenging task of learning invariant representations over a large set of images and various augmentation transforms. In this paper, we focus on developing algorithms that can speed up the training of SSL while maintaining their performance.

Prior work in efficient supervised training reduced the cost of training by gradually increasing the training resolution together with the augmentations magnitude using Progressive Learning Tan and Le (2021). While augmentation is a regularization mechanism in supervised learning it is the main source of supervision in SSL and therefore plays a much more important role. Additionally, Super Convergence Smith and Topin (2017) is introduced which uses a cyclic learning rate and anti-phased momentum schedule that is used to accelerate convergence in supervised learning tasks. The longer duration of training makes the application of Super Convergence more difficult and using it together with Progressive Learning causes instabilities in the early stage of training.

In this work, we aim to increase the training efficiency of self-supervised training methods by optimizing the learning rate, resolution, and augmentation schedules to allow reaching the same level of performance with a smaller computational budget. Inspired from Smith and Topin (2017) and Tan and Le (2021) that were used in su-

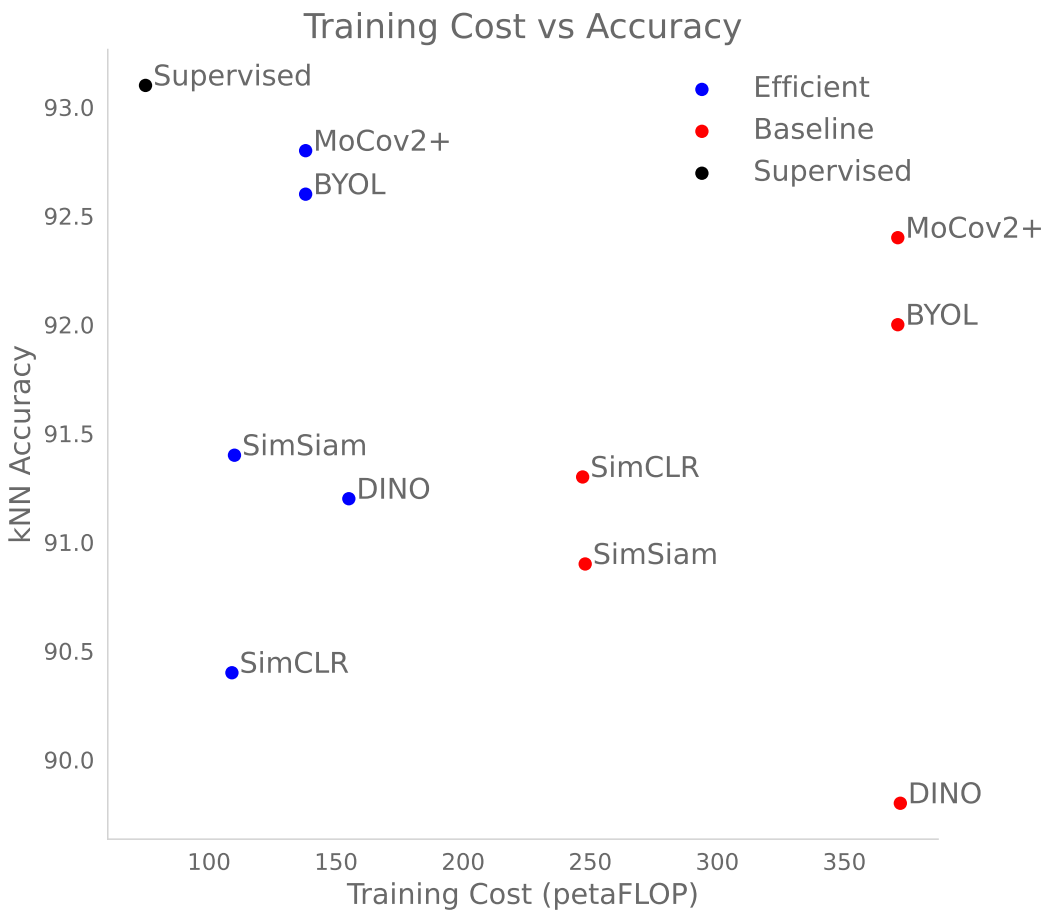


Figure 3.1: Classification accuracy *v.s.* training cost of various SSL methods, which are trained on a subset of ImageNet by using ResNet50 (see Section 3.4.2 for experimental details). Our method (drawn in blue) successfully significantly accelerates the SSL methods (drawn in red) without any significant drop in their performance.

pervised learning we propose a combined learning rate and resolution schedule for self-supervised learning. Additionally, we modify the augmentation strategy used in self-supervised learning for faster training.

Our contributions are three folds. Fixed 1-cycle Learning Rate schedules (see Section 3.3.1) allows Super-Convergence Smith and Topin (2017) to take place for longer training times by fixing the duration of its warm-up phase while extending the decay phase and comparing it against alternative learning rate schedules used in self-supervised learning. Super Progressive Learning (see Section 3.3.2) proposes a suitable Progressive Learning schedule Tan and Le (2021) for SSL by adding a warm-up stage with full resolution at the beginning of the resolution schedule while increasing the augmentation magnitude gradually during training so that it works well with our

proposed learning rate schedule. Hard Augment (see Section 3.3.3) selects a hard augmentation pair dynamically from multiple low resolution augmentations in order to boost the learning signal and regularize the training.

We show that our method accelerates the training of different self-supervised learning methods and architectures while maintaining comparable performance to standard training. We provide ablation studies for analysing the contribution of different methods and determining the optimum hyperparameters of our methods and add theoretical justifications.

3.2 Related Works

Generative models like DBN Hinton et al. (2006), VAE Kingma and Welling (2013) and GANs Goodfellow et al. (2014) have been used for unsupervised representation learning however discriminative SSL have proven to be more effective. While early work on SSL focused on designing various pretext tasks such as solving jigsaw puzzles Noroozi and Favaro (2016), colouring Zhang et al. (2016); Larsson et al. (2017); Zhang et al. (2017c) and predicting rotation Gidaris et al. (2018), more recent techniques focused on learning augmentation invariant representation Wu et al. (2018) Tian et al. (2020b) to replace the loss functions that require manual supervision. Recently, Contrastive learning methodsHe et al. (2020); Chen et al. (2020a,c) have shown promising results for various target tasks and caused renewed interest in this area. However, at least an order of magnitude, more computation is required when training these methods in order to get comparable performance to supervised learning. In fact, we demonstrate our method in these recent models and show that SimSiam Chen and He (2020), BYOL Grill et al. (2020), MoCov2 Chen et al. (2020c), SimCLR Chen et al. (2020a) and DINO Caron et al. (2021) training can significantly be sped up.

By adaptation gradient noise during training large scale data parallel training has been used for increasing training speed in large deep learning methods. Using many accelerators together and increasing the total batch size training speed can be scaled up almost linearly with the number of accelerators Li et al. (2020). This way, supervised ImageNet training can be done in minutes Smith et al. (2017) or hours Goyal et al. (2017) instead of days even though the total training cost remains roughly the same. This required adapting the hyperparameters so that gradient noise was managed properly Smith and Le (2017); Jastrzębski et al. (2017); Chaudhari and Soatto (2018) by using the linear relationship between the batch size and learning rate Goyal et al. (2017);

Smith et al. (2017). It was shown that training with larger learning rates had also a regularization effect Jastrzębski et al. (2017); Chaudhari and Soatto (2018); Lewkowycz et al. (2020) and prevented convergence to sharper minima Keskar et al. (2016). We use the relationship between gradient noise and learning rate schedules to optimize for training efficiency instead of speed where we synchronize the learning rate schedule with our resolution schedule to reduce the total cost of training. We use a cyclic learning rate schedule Smith and Topin (2017) which combines cyclic learning rate and cyclic momentum adaptation together as our learning rate schedule due to its efficient training performance and adapt it to longer training regimes common in SSL training.

Curriculum learning Bengio et al. (2009) has been used to accelerate convergence for deep learning methods and when applying it to self-supervised learning, augmentation strength and image resolution are the most relevant parameters. Image resolution directly affects accuracy Hoffer et al. (2019); Touvron et al. (2020) while reducing run-time quadratically. Previous work Howard (2018) on efficient supervised learning gradually increased the image resolution in the DAWN benchmark Coleman et al. (2017) to accelerate training while having slightly lower performance. Recently Tan and Le (2021) proposed gradually increasing the training image resolution while increasing augmentations strength to allow for both faster and accurate training. While augmentations are used for regularization in supervised learning they are the main source of supervision for recent self-supervised methods and determine task difficulty. We build on this intuition and adapt Progressive Learning for efficient self-supervised training since in order to reduce the training cost especially in the initial phase while increasing the difficulty of the task gradually. To the best of our knowledge having a good schedule for augmentation magnitude and its relation to resolution hasn't been examined for self-supervised learning.

Hard positive mining is another technique that can be used to accelerate training. Importance sampling based on sample loss has been shown to accelerate supervised learning Katharopoulos and Fleuret (2018); Jiang et al. (2019). In object detection Shrivastava et al. (2016) selects regions with the highest loss in order to select useful positives and balance positive to negative regions effectively. Gong et al. (2020) applied multiple augmentations and back-propagated only the augmented sample with the maximum loss in order to improve the adversarial robustness of their method while increasing supervised performance.Caron et al. (2020) have introduced a multi-crop strategy for self-supervised learning that combines low resolution and high resolution crops in order to show increased performance while trying to keep the computational

cost limited. We mine hard augmentations during training by utilizing the loss based importance sampling technique to dynamically select the most useful augmentations. Since our focus is on efficient training, we used down sampled versions of the augmented images in the selection pass to minimize the overhead.

3.3 Method

In this section we introduce the techniques used to accelerate self-supervised training. Each technique can be used to accelerate training on its own but they can be combined together in a synergistic way to enable faster training. We introduce Fixed 1-cycle Learning Rate Schedule in Section 3.3.1, Super Progressive Learning in Section 3.3.2 and Hard Augment in Section 3.3.3.

We formulate a contrastive loss function that measures difference between pairs of images and optimizing this loss allows us to learn parameters θ for a deep neural network f_θ that produces similar representations for augmentations of the same image and have representation that is going to be useful for various target tasks. Let D be an unlabeled dataset consisting of $|D|$ images with resolution r. We randomly sample two image transformations τ and $\bar{\tau}$ from the transformation space T for each training image $\boldsymbol{x}$, apply them to $\boldsymbol{x}$ to obtain two views $\boldsymbol{v}$ and $\bar{\boldsymbol{v}}$, extract their features through the deep neural network $\boldsymbol{z} = f_\theta(\boldsymbol{v})$ and $\bar{\boldsymbol{z}} = f_\theta(\bar{\boldsymbol{v}})$ respectively. To learn the network weights θ, we minimize the loss function L that represents the mismatch between two representations $\boldsymbol{z}$ and $\bar{\boldsymbol{z}}$ over the dataset and transformation space:

$$\min_{\theta} \mathbb{E}_{x\sim D,(\tau,\bar{\tau})\sim T} L(\boldsymbol{z},\bar{\boldsymbol{z}}) \tag{3.1}$$

We use minibatch Stochastic Gradient Descent (SGD) optimizer with momentum:

$$L^{(B)}(\theta) = \frac{1}{|B|} \sum_{\boldsymbol{x}\sim B,(\tau,\bar{\tau})\sim T} L(\boldsymbol{z},\bar{\boldsymbol{z}}), \tag{3.2}$$

where the minibatch B consists of $|B|$ randomly sampled images from D and the loss is averaged over the samples to obtain a noisy yet unbiased estimate of the true gradient.

The update rule for θ is given as:

$$\begin{aligned} \mu_t &= \beta_t \mu_{t-1} - \varepsilon_t \nabla_{\theta_t} L^{(B)}(\theta_t) \\ \theta_{t+1} &= \theta_t - \mu_t \end{aligned} \tag{3.3}$$

where ε_t is the learning rate and β_t is the momentum weighs at step t. The values for the learning rate and momentum is given by a learning rate scheduler $(\varepsilon_t, \beta_t) = S(t)$.

Eq. (3.3) can be interpreted as stochastic differential equation (*e.g.* Smith and Le (2017)):

$$\frac{d\theta}{dt} = \mu, \quad \frac{d\mu}{dt} = \beta_t \mu - \frac{dL}{d\theta} + \eta_t \tag{3.4}$$

where $\eta(t) \sim \mathcal{N}(0, g\boldsymbol{F}(\theta)/|D|)$ is an additive Gaussian noise originating from the stochasticity and $\boldsymbol{F}(\theta)$ is the covariance matrix for gradients of the samples and g is the "noise scale" which is given by $g \approx \frac{\varepsilon_t |D|}{b(1-\beta_t)}$. For our analysis we will utilize the noise in the gradients which is proportional with the learning rate ε_t while being inversely proportional with batch size b and $1-\beta_t$. This relationship plays an important role when adapting hyperparameters to different setups and for giving us a theoretically grounded understanding for learning rate schedules and their relationship with Progressive Learning.

3.3.1 Fixed 1-cycle Learning Rate Schedule

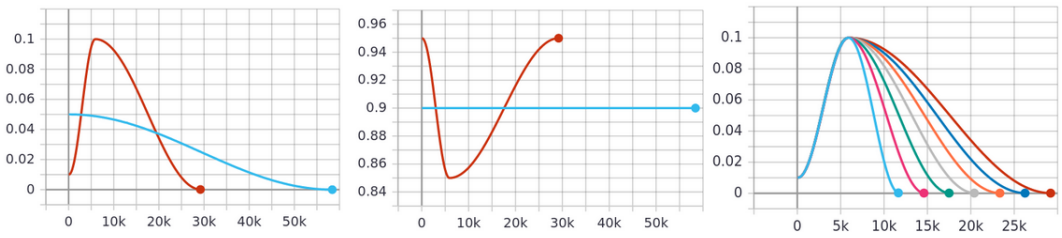


Figure 3.2: Learning rate (left) and momentum (middle) for standard cosine annealing learning rate schedule in blue and F1-CLR schedule in red are visualized. Learning rate for F1-CLR schedule with different length of training instances is visualized (right)

Cosine annealing learning rate schedule which has been used widely in SSL He et al. (2020); Chen et al. (2020c); Chen and He (2020) decays the learning rate starting from a maximum learning rate value ε_{max} using a cosine function and can be described as,

$$\varepsilon_t = \frac{1}{2}\varepsilon_{max}(\cos(\frac{t}{L}\pi) + 1), \tag{3.5}$$

where L is the total number of iterations.

Methods that perform better with larger batch sizes train with larger learning rates in order to maintain the gradient "noise scale" $g \propto \frac{\varepsilon_t}{b}$ during training. In order to prevent instabilities in the early stages of training due to rapidly changing parameters values Goyal et al. (2017) a linear warm-up phase for the learning rate is added Chen et al.

(2020a); Grill et al. (2020); Chen et al. (2021); Zbontar et al. (2021); Caron et al. (2020, 2021),

$$\varepsilon_t = \begin{cases} \frac{t}{t_w}\varepsilon_{max}, & t < t_w \\ \frac{1}{2}\varepsilon_{max}(\cos(\frac{t-t_w}{L-t_w}\pi)+1), & \text{otherwise} \end{cases} \tag{3.6}$$

where t_w is the number of warm up steps.

On the other hand, Cyclic Learning Rate (CLR) schedule introduced by Smith (2015) and later refined as the 1-cycle learning rate (1-CLR) schedule Smith and Topin (2017); Smith (2018) starts with a small learning rate and than increases the learning rate to a very high value and than decays it gradually to a very small value while changing the momentum at the opposite direction as the learning rate. The learning rates and momentum values at each step using a cosine window can be calculated by

$$\varepsilon_t = \begin{cases} \varepsilon_{max} - \frac{1}{2}\varepsilon_{max}(\cos(\frac{t}{\rho L}\pi)+1) & t < \rho L \\ \frac{1}{2}\varepsilon_{max}(\cos(\frac{t-\rho L}{L(1-\rho)}\pi)+1) & \text{otherwise} \end{cases} \tag{3.7}$$

$$\beta_t = \begin{cases} \beta_l + \frac{1}{2}(\beta_h - \beta_l)(\cos(\frac{t}{\rho L}\pi)+1) & t < \rho L \\ \beta_h - \frac{1}{2}(\beta_h - \beta_l)(\cos(\frac{t-\rho L}{L(1-\rho)}\pi)+1) & \text{otherwise} \end{cases}$$

where ρ is the time percentage of time allocated for the first phase, while β_l and β_h are lower and higher limits for momentum respectively. We use the learning rate (LR) range test proposed by Smith (2018) to set the maximum learning rate ε_{max} please see the details in supplementary material A.3.

A phenomenon called Super Convergence has been demonstrated in supervised classification where using larger learning rates and the 1-CLR schedule training time to reach a specified performance has been decreased dramatically Howard (2018). However, unlike supervised learning where longer training times do not generally result in better performance and can sometimes even cause worse performance due to over-fitting, in SSL the quality of the representation typically improves with longer training time Chen et al. (2020c,a). A problem with the current 1-CLR is that the percentage of the first phase is being determined in proportion to the full training duration which causes an extremely long warm-up phase in longer training settings wasting compute time. To address this problem, we propose to extend the annealing phase of 1CLR while keeping the warm up length t_w the same which we call *Fixed 1-cycle Learning Rate* (F1-CLR) schedule,

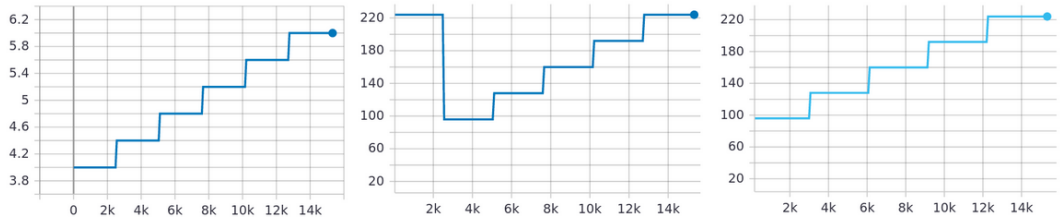


Figure 3.3: Augmentation magnitude and input resolution schedule where augmentation magnitude is gradually increased while the input size starts large and than gradually increases from its minimum value mimicking the inverse of the F1-CLR schedule. (left) Augmentation Magnitude, (middle) Super Progressive Learning resolution schedule, (right) Progressive Learning resolution schedule

$$\varepsilon_t = \begin{cases} \varepsilon_{max} - \frac{1}{2}\varepsilon_{max}(\cos(\frac{t}{t_w}\pi) + 1 & t < t_w \\ \frac{1}{2}\varepsilon_{max}(\cos(\frac{t-t_w}{L-t_w}\pi) + 1) & \text{otherwise} \end{cases} \tag{3.8}$$

$$\beta_t = \begin{cases} \beta_l + \frac{1}{2}(\beta_h - \beta_l)(\cos(\frac{t}{t_w}\pi) + 1) & t < t_w \\ \beta_h - \frac{1}{2}(\beta_h - \beta_l)(\cos(\frac{t-t_w}{L-t_w}\pi) + 1). & \text{otherwise} \end{cases}$$

Fig. 3.2 shows a comparison between cosine schedule and F1-CLR schedule in terms of learning rate (left) and momentum (middle) as well as illustrating the F1-CLR learning rate schedule for different lengths of training (right). The anti-phased movement of momentum in F1-CLR allows larger learning rates to be achieved while keeping gradient noise in check ($g \propto \varepsilon_t/(1-\beta_t)$ see Eq. (3.4)). Note that we do not extend the warm up phase, as the warm up is used as a stabilizer, while the learning rate and gradient noise scale are increasing ($g \propto \varepsilon_t$ see Eq. (3.4)).

3.3.2 Super Progressive Learning

Input resolution is very important factor that effects training time. Typically image resolution is a trade-off hyperparameter between performance and computational load, *i.e.* higher resolution, higher performance but also more computations. In SSL, the standard practice He et al. (2020); Chen et al. (2020c); Chen and He (2020); Chen et al. (2020a); Grill et al. (2020); Chen et al. (2021); Zbontar et al. (2021); Caron et al. (2020, 2021) is to train with fixed resolution and fixed augmentation settings.

We hypothesize that higher resolution inputs are only necessary when there is small amount of noise in the gradients, and hence, propose to employ a learning rate aware

progressive learning strategy inspired by Tan and Le (2021). However, as starting training with small input resolution as in Tan and Le (2021) results in additional noise in the gradients which prevent it from being trivially incorporated to F1-CLR due to the instabilities in the warm-up phase. Hence, we propose train with full resolution images during the warm-up phase before going into the linearly increasing resolution schedule while we gradually increase the augmentation magnitude as shown in Fig. 3.3 (left, middle). We call this strategy *Super Progressive Learning*. When the learning rate is high there is a large amount of noise in the gradients $g \propto \varepsilon_t$ and this allows us to use low resolution images at that point without resulting in bad performance however since the training starts with small learning rates we need to adapt our schedule and use large resolution images.

Note that the speedup of this strategy depends on the input resolution schedule (r_t). Assuming a quadratic relationship between training time and resolution, the speedup M can be calculated as:

$$M = \frac{1}{L}\sum_{t}(\frac{r_{max}}{r_t})^2, \tag{3.9}$$

where r_t is the input image resolution at time t, $r_{\max}$ is the final resolution. We discretize the resolution steps by 32 due to the fact that most network architectures have 5 pooling/dilation layers. The minimum value for the resolution schedule is determined empirically.

3.3.3 Hard Augment

So far we focused on varying the effective step size and input resolution. Informativeness of training samples is another important factor that can speed up the training by providing more efficient gradients. We propose Hard Augment to boost the learning signal in self-supervised learning. We generate m augmentations $\tau^m = \{\tau_{1..m} : \tau_{1..m} \sim T\}$ for each image and evaluate the loss for all pairs of augmentations $p = \binom{\tau^m}{2}$ using a forward pass to select the augmentation pair that results in the largest amount of loss for back-propagation. By focusing on the samples that produce the most lost and ignoring a large fraction of the augmentations that produce a small loss and do not make a significant contribution to the training we can accelerate training dramatically. The overall objective that we optimize is

$$L_{ha}^{(B)}(\theta) = \frac{1}{|B|}\sum_{\boldsymbol{x}\sim D,\tau_{1..m}\sim T}\left[\max_{\tau_p \in p} L(\boldsymbol{z_i}, \boldsymbol{z_j})\right]. \tag{3.10}$$

Gong et al. (2020) have shown that using the maximum of multiple augmentations applies a regularisation on the gradient-norm with respect to the input images $\left\|\nabla_{\boldsymbol{x}} L(\boldsymbol{z_i}, \boldsymbol{z_j})\right\|_2$ that is in the order of $\sigma\sqrt{\log m}$ where σ is the strength of the augmentation under the assumption that $\tau(x) \sim \mathcal{N}(x, \sigma^2 I)$. The regularization effect in our method can be seen as a corollary of their theorem that uses a pair of augmentations instead with a number of equivalent augmentations as the number of pairs $|p|$.

Crucially in order to decrease the overhead of selection we propose to down sample the images during the selection pass to r_{sel}. The image pairs that have the highest loss will than be used in training in their original resolution r. We do not need to have full resolution for the selection pass as we only need to have sufficient resolution to rank the augmentation pairs with respect to their loss and to find the highest loss pair which requires far less precision than is required for obtaining good quality gradient updates during training.

$$M_{ha} = \frac{r^2 C}{r^2 C + m r_{sel}^2} \tag{3.11}$$

We find the minimum resolution r_{sel} that we can use for the selection pass empirically. On a typical setting in ImageNet where the original resolution is $r = 224$ and the reduced selection resolution is $r_{sel} = 64$ with $m = 4$ augmentations and 6 possible pairs and the fraction for full training iteration cost to a forward pass cost is $C = 6$ we only add $1 - M_{ha} = \%5$ overhead on the training while significantly boosting the training speed by focusing on the most valuable augmentations.

We have provided the pseudo code for our method in Algorithm 1. Our method can easily be adopted to different SSL methods without any major difficulty as it operates at the level of augmentations, optimizer parameters and sampling strategies.

Algorithm 1 Efficient SSL Training, PyTorch-like

```
# f: backbone + projection layers
# criterion: loss function for SSL method
# aug: augmentation function
# F1CLR: Super Convergence Schedule (Section 3.1)
# SP: Super Progressive Schedule (Section 3.2)

t=0 # iteration
for e in range(epochs):
    for x in loader: # load a minibatch x with |B| samples
       eta, beta = F1CLR(t) # Update learning rate and momentum weight
       r, m = SP(t) # Update resolution(r) and augmentation magnitude(m)
       vs = [aug(x, r, m) for i in range(n)] # n random augmentation
       vi, vj = hard_augment(vs) # select hardes augmentations
       zi, zj = f(vi), f(vj) # forward-propagation

       L = criterion(zi,zj) # Loss calculation
       L.backward() # back-propagation
       update(f, eta, beta) # SGD update
       t+=1

def hard_augment(vs): # negative cosine similarity
    vs_small = interpolate(vs, r_min) # down sample views to r_min
    zs = f(vs) # forward prop for all views

    indexes = arg_max_loss_pair(zs) # determine pairs with maximum loss
    vi, vj = hard_select(vs, indexes) # select pairs by tensor indexing
    return vi, vj
```

3.4 Experiments

In this section, we first study evaluate our method on different backbone architectures in Section 3.4.1 and on various existing SSL methods in Section 3.4.2, finally we analyze the effect of each proposed component in our method in Section 3.4.3 and run ablation studies that analyze the hyperparameters and different trade-offs of our method in Section 3.4.4.

We evaluate the experiments on the ImageNet dataset Deng et al. (2009) in terms of linear probing accuracy. This is done by freezing the backbone after pre-training and training a linear classification layer in a supervised way following Chen and He (2020). The evaluation on the Imagenette dataset[1] is done using online kNN accuracy of the feature maps. When calculating speed up we consider the number of steps used in training and multiply that by the number of floating point calculations made in each step and compare it against the baseline.

Implementation Details Here we provide details on the default parameters that we have used during our experiments where we use the SimSiam Chen and He (2020) model

[1]ImageNet subset that consist of 10 classes (tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, parachute) avalilable at `https://github.com/fastai/imagenette`

	Baseline		Efficient	
Architecture	Acc.(%)	Cost (pF)	Acc.(%)	Speedup
ResNet18	51.1	3756	51.3	2.3x
ResNet50	69.2	8494	69.3	2.3x

Table 3.1: Our efficient strategies applied to different backbone architectures using SimSiam model on Imagenet dataset. Speed up is reported based on the reduction in training cost which is measured in terms of petaFLOPs

	Baseline		Efficient	
Model	Acc.(%)	Cost (pF)	Acc.(%)	Speedup
SimSiam	90.9	248	91.4	2.3x
BYOL	92.0	371	92.6	2.7x
SimCLR	91.3	247	90.4	2.3x
MoCov2+	92.4	371	92.8	2.7x
DINO	89.8	372	91.2	2.4x

Table 3.2: Our efficient strategies applied to different SSL models training on ResNet50 backbone architecture on Imagenette dataset. Speed up is reported based on the reduction in training cost which is measured in terms of petaFLOPs

with ResNet50 backbone using the SGD optimizer with .9 momentum. On ImageNet experiments we use a batch size of 512 and weight decay of 0.0001. For the baseline experiments we use the Cosine Annealing learning rate schedule with learning rate of 0.1 (η_{max}) and trained them for 200 epochs (t_{max}) and for our accelerated F1-CLR schedule we use a learning rate of 0.16 (η_{max}) with 10 epoch warm-up (t_w) and train them for 120 epochs (t_{max}). For Imagenette experiments we use a batch size of 128, weight decay of 0.0005, learning rate of 0.1 (η_{max}) with Cosine Annealing scheduler and train them for 800 epochs (t_{max}). For our accelerated setting we use the F1-CLR schedule with learning rate of 0.2 (η_{max}) with 80 epochs for warm up (t_w) and train for 480 epochs (t_{max}). For Hard Augment we use 6 augmentations generated using SimCLRChen et al. (2020a) transforms and scale the colour jittering linearly where the standard scale is defined as 5. For Super Progressive learning we use 96 as the minimum resolution and use 6 stages each a multiple of 32. Additional details can be seen in Supplementary Sec. A.2.

We have observed that GPU pre-processing plays an important role in obtaining

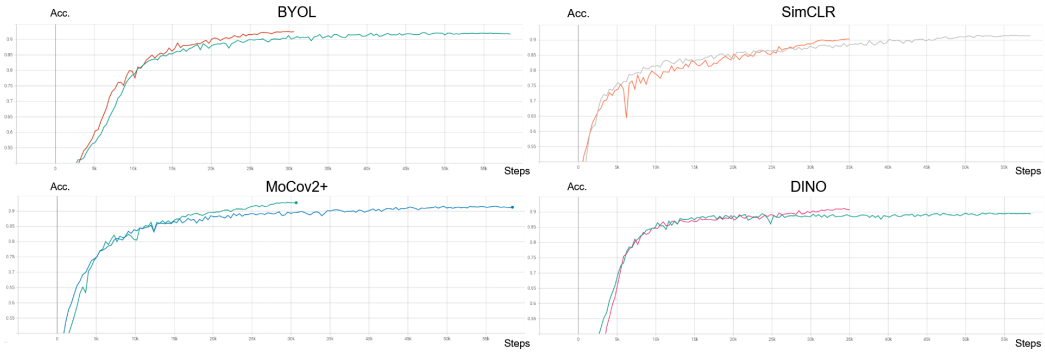


Figure 3.4: We provide the training curves for different models and their efficient trained versions using the ResNet50 backbone architecture on Imagenette dataset. The baseline methods are trained for 800 epochs while their efficient versions are trained for 420 epochs in BYOL and MoCov2+ and 480 epochs in DINO and SimCLR. We observe that our method is especially effective during the mid-phase and end-phase of the training. While the baseline strategies stagnate our methods show increased training speed.

speedups in real time. Since loading images and making multiple augmentations can become a bottleneck when training with faster GPU cards like the Nvidia V100. We use the Nvidia DALI[2] library for fast loading and augmentation of the images in the accelerators.

3.4.1 Backbone Architecture Analysis

In order to show the generality of our method to different different backbone architectures we applied our training strategies to SimSiam Chen and He (2020) model on the ImageNet dataset and report linear evaluation results and speed-up in Table 3.1.

Our results show that our method can accelerate both ResNet-18 and ResNet-50 architectures by 2.3 times while maintaining the baseline accuracy.

3.4.2 Model Experiments

We have evaluated our acceleration strategies on different SSL models in Table 3.2 by pre-taining on a ResNet 50 backbone on the Imagenette dataset. We used the default hyperparameters for SimSiamChen and He (2020), BYOLGrill et al. (2020), MoCov2+ Chen et al. (2020c), DINO Caron et al. (2021) and SimCLR Chen et al. (2020a) models while we used the same optimizer and augmentation hyper-parameters for all of them

[2]available at `https://github.com/NVIDIA/DALI`

LR schedule	Sup. Prog.	Hard Aug.	kNN-Acc.(%)	Speedup
CA	-	-	90.9	1.0x
F1-CLR	-	-	90.2 (-.7)	1.7x
CA	-	✓	89.8 (-1.1)	1.8x
F1-CLR	✓	-	90.1 (-.8)	2.7x
F1-CLR	-	✓	90.3 (-.6)	1.5x
F1-CLR	✓	✓	91.4 (+.5)	2.4x

Table 3.3: Comparing the contribution of each component in our experimental setup. Speed up is reported based on the reduction in training cost. Using SimSiam model with ResNet50 backbone architecture on Imagenette dataset. CA stands for the baseline Cosine Annealing learning rate schedule.

which can be seen in the supplementary Sec. A.2.

We have observed that that our method can accelerate the training of different models. We note that methods with a momentum encoder typically have a higher accuracy and training cost even when trained with the same number of steps due to the additional forward passes made with the momentum encoder which helps to stabilize the training. Our method is especially effective for models with a momentum encoder where we can accelerate the training up to 2.7 times.

We visualize the results of this experiment in Fig. 3.4. We have seen that our training strategies are most effective during the mid phases and towards the end of the training. While part of the training speed comes from training for fewer number of steps part of it comes from doing less computation in each step thanks to the lower resolution of the images. We can accelerate training many models without sacrificing accuracy.

3.4.3 Experimental Setup

Here we compare the contribution of each component in our experimental setup in Table 3.3 training with ResNet50 backbone on the Imagenette dataset utilizing default hyperparameters whenever possible. We have observed that our improvements are generally compatible with each other and allow us to train close to the comparable accuracy while reducing the training cost by 2.4 times.

A key insight that can be gleaned from the results is that using Hard Augment alone may not yield the best results. Intriguingly, the model that employed only Hard Augment demonstrated a slight decrease in accuracy (-1.1%), although it achieved

a moderate speedup (1.8x). Therefore, the data suggests that while Hard Augment contributes to improving efficiency, its combination with F1-CLR yields better results in terms of accuracy (-0.6%).

While Super Progressive Learning is the main driver in reducing computational cost (2.7x when used with F1-CLR), without Hard Augment it can't reach baseline accuracy (-0.6%). We note that Super Progressive Learning and Hard Augment also work well synergistically. This allows us to surpass the baseline accuracy (+0.5%) while yielding a substantial speedup (2.4x). This synergy is demonstrated by Hard Augment's ability to select from a harder set of alternatives towards the end of the training, which further boosts performance.

It should be noted that this ablation study included one experiment in each setting. The primary objective of this study is to demonstrate the operative mechanism of these techniques, not to claim statistically significant results. Thus, our findings provide meaningful insights into the effectiveness and functionality of the techniques rather than asserting definitive evidence of statistical significance.

3.4.4 Ablations

In the ablation studies use the Imagenette dataset with the parameters described in the previous section. We aim to identify various parameters of our model and examine its behaviour in different circumstances.

Learning Rate Schedule Comparison We compared different learning rate schedulers in isolation in order to see what percentage of improvement is attributable to the different learning rate schedulers on the ImageNet dataset in Table 3.4. We have seen that our Extended Super Convergence method still gives a reliable performance increase in this setup compared to Cosine Annealing and Cosine Annealing with linear warm up.

Schedule	Warm-Up	Epochs	Acc.(%)
Cosine Decay	-	200	67.7
Cosine Decay	-	100	66.1
Cosine Decay	Linear-10	100	66.0
F1-CLR (3.3.1)	F1-CLR (3.3.1)	100	66.4

Table 3.4: Comparing different learning rate schedules. Cosine decay with linear warm-up and Super convergence both have 10 epochs of warm-up in order to make the comparison more direct.

F1-CLR Warm Up Here we make an ablation study where we train the SimSiam model for 320 epochs with different number of warm up epoch lengths in Table 3.5. We observe that 80 epochs is the optimal warm up length and we use this setting in our future experiments and ablations that utilize F1-CLR schedule.

Warm Up	32	48	64	80	96	112
Acc(%)	86.8	86.5	86.9	**87.1**	86.8	86.6

Table 3.5: Super Convergence training where the number of warm up epochs is changed.

Minimum Resolution

An important hyper-parameter that determines how much speed-up can be achieved with Super Progressive Learning is the minimum resolution. We made a ablation study in Table 3.6 to see the maximum speedup that can be achieved with our training strategies by increasing the minimum resolution by 32 increments starting from 64. The smallest resolution that does not result in a drop in performance is 96 which we used in our other experiments while 64 was giving better speedups it wasn't maintaining the level of performance we are seeking.

Min. Res.	Speed up	Acc.(%)
128	1.40x	86.6
96	1.60x	86.6
64	1.82x	86.1
0	3x	N/A

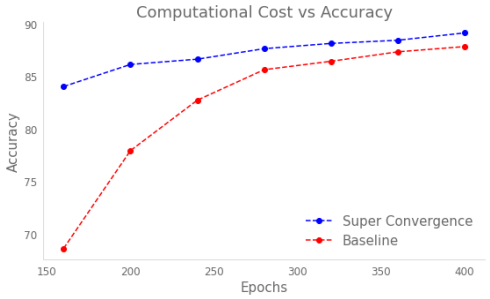
Table 3.6: Ablation study for the Super Progressive learning speedup and accuracy trade-off with different values for minimum resolution using 224 as the maximum resolution.

F1-CLR Training Length

To build a better understanding of 1CLR schedule we change the length of the training from 160 epochs to 800 epochs using the cosine annealing learning rate schedule and compare against how the proposed extended super convergence schedule performs. Table 3.7 shows that the difference in performance is stark when the training duration is short and gradually becomes smaller. Additionally, our F1-CLR setting leads to between 1.3x-1.6x speedup of the training to reach equivalent performance on the baseline.

Number of Positives

In order to understand the effect of number of augmentations we have made an



Baseline		Super Convergence	
Epochs	Acc.(%)	Speedup	Acc.(%)
160	68.8	1.61x	84.0
200	77.9	1.53x	86.1
240	82.7	1.37x	86.6
280	85.6	1.37x	87.6
320	86.4	1.42x	88.1
360	87.3	1.41x	88.4
400	87.8	1.59x	89.1

Table 3.7: (left):Online validation accuracy of training instances of different lengths that are trained with F1-CLR (right): Total length of training given the fixed duration for warm-up of 80 epochs. Speedups are calculated based on linearly interpolated training length in the baseline given that the baselines has at 800 epochs has %90.0 accuracy

ablation study that examines the trade-off between increased overhead and the effect on accuracy in Table 3.8. We have used the SimSiam Chen and He (2020) model with F1-CLR schedule and our default hyperparameters. We have observed that 6 positives gives good balance between maximizing accuracy while minimizing the overhead so we use this setting in other experiments.

# of Positives	4	5	6	8
Accuracy(%)	89.1	89.0	89.6	89.8
Overhead(%)	5.4	6.8	8.2	10.9

Table 3.8: Ablation study for the numbers of positives used in Hard Augment. 6 augmentations result in good performance increase while maintaining reasonable overhead.

Multi-Crop Comparison

Multi-crop augmentation proposed by SwAV Caron et al. (2020) uses multiple augmentations with a combination of high resolution and low resolution crops where only high resolution crops are used for back-propagation and low resolution ones are used only as extra comparison targets. A slight modification to our Hard Augmentation method will produce an intermediate algorithm useful for a comparison where the additional low resolution augmentations that do not have the highest loss can be used as extra targets. We made an ablation study that compares the multi-crop augmentation

strategy at the last line of Table 3.9 against our method and our method with extra targets. We train our method with Extended Super Convergence for 400 epochs and used 4 augmentations for each image.

Min Res.	Selection	Extra Target	Acc(%)
64	Hard		89.1
64	Hard	✓	88.8
96	Hard	✓	89.7
96	Random	✓	88.4

Table 3.9: Progressive learning speedup with different values for minimum to maximum resolution fraction

We have seen that extra targets are useful when 96 resolution is used for the selection pass and hard augment with extra targets and hard augment outperforms multi-crop augmentation. We have also seen that the standard resolution of 64 that is enough for ranking loss pairs in Hard Augment is not enough for generating accurate targets.

Since increasing selection resolution has a direct effect on the overhead for Hard Augment and the additional improvement that can be obtained from extra targets is small we have decided not to include extra targets in our method.

3.5 Conclusion

We have demonstrated that self-supervised training can be accelerated by adapting the learning rate, augmentation and resolution schedules for self supervised training and boosting the training signal by hard positive mining on the augmentations. Our method have shows an training speed-up between 2.3 and 2.7 times. which allows a much wider community to reproduce and contribute to the self-supervised learning literature, reduce the financial and carbon cost of training these models. However there is a risk that using efficient training methods the community will adopt larger and computationally more expensive benchmarks which will eliminate some of the intended benefits of our method. As future work we aim to apply our method to recent methods Caron et al. (2021); Chen et al. (2021) that used Vision Transformer Dosovitskiy et al. (2020) backbones.

Chapter 4

Fast Online Minibatch Selection

Abstract

The success of deep learning has increasingly become dependent on more data and computations in the last decade. In this paper, we aim to accelerate the training time of deep networks and propose a fast online minibatch selection method that efficiently selects hard samples by efficiently approximating loss values of training samples on downscaled images followed by a subsampling step. We show that this strategy, when combined with a late uniform sampling, achieves significant improvement over the state-of-the-art, accelerating training 1.3 to 1.5 times without any performance drop on large image classification datasets and modern backbone architectures. Our method is backbone agnostic and can be integrated into popular deep learning libraries with few additional lines of code.

4.1 Introduction

In the last decade, large-scale datasets Deng et al. (2009); Sun et al. (2017) and high capacity models Simonyan and Zisserman (2015); He et al. (2016); Dosovitskiy et al. (2020) have become the norm to obtain state-of-the-art results in computer vision problems Hestness et al. (2017); Kaplan et al. (2020). However, their success comes at the price of rapidly increasing computational costs and large carbon footprint. In this paper, we look at reducing the training cost of the deep networks without sacrificing their performance by focusing on more informative samples.

Importance sampling –a strategy for preferential sampling of more important examples– has been a popular technique to accelerate the training process for gen-

eral convex minimization problems Bordes et al. (2005); Nesterov (2012); Richtárik and Takáč (2014). In deep learning, Bengio *et al*. Bengio et al. (2009) propose a human-inspired sampling scheme that selects samples with increasing difficulty for training. More recent ones Schaul et al. (2015); Loshchilov and Hutter (2015); Katharopoulos and Fleuret (2018); Alain et al. (2015), closely related to our work, propose to sample hard samples, which produce high loss values in training, based on their loss value or gradient magnitude. However, hard example sampling has at least two challenges. First, keeping track of loss values of all samples, which are required for computing their sampling probability, can be computationally expensive and lead to a significant overhead in training. Previous works Loshchilov and Hutter (2015); Schaul et al. (2015); Johnson and Guestrin (2018) mitigate this problem by keeping a history of loss values from previous iterations, and computing and updating them occasionally. However, as the model is constantly updated, the importance of samples may vary significantly, and their recorded values may get outdated, leading to poor sample selection. Second, focusing only on hard examples may bias the model towards learning different data distribution than the original by paying more attention to difficult classes and samples, including outliers. Hence previous works either fail to boost training speed due to the additional overhead Kumar et al. (2010); Zhou and Bilmes (2018); Paul et al. (2021); Katharopoulos and Fleuret (2017) or fail to obtain comparable performance to the standard training Jiang et al. (2019); Raju et al. (2021).

To this end, we propose a fast online sampling method that samples examples in proportion to their loss as in previous work Loshchilov and Hutter (2015); Schaul et al. (2015); Katharopoulos and Fleuret (2018). However, unlike them, our method can accurately approximate the loss values of examples while being efficient and learning a model that is not biased towards harder samples. In particular, we use a two-step online sample selection mechanism that first uniformly samples a large minibatch of examples from a uniform distribution and measures their approximate loss using resolution-downsampled versions. Then we subsample this minibatch using a multinomial probability function based on this approximate hardness. The uniform sampling prevents the exclusive focus on only a few hard examples, and the online loss approximation via downsampled examples enables efficiently tracking their loss values as model parameters change, resulting in a significantly smaller overhead than the prior work. Finally, we introduce a cool-down period at the end of our training schedule that employs uniform sampling to mitigate any data bias due to the hard example sampling. We show that our method can be successfully used to accelerate the

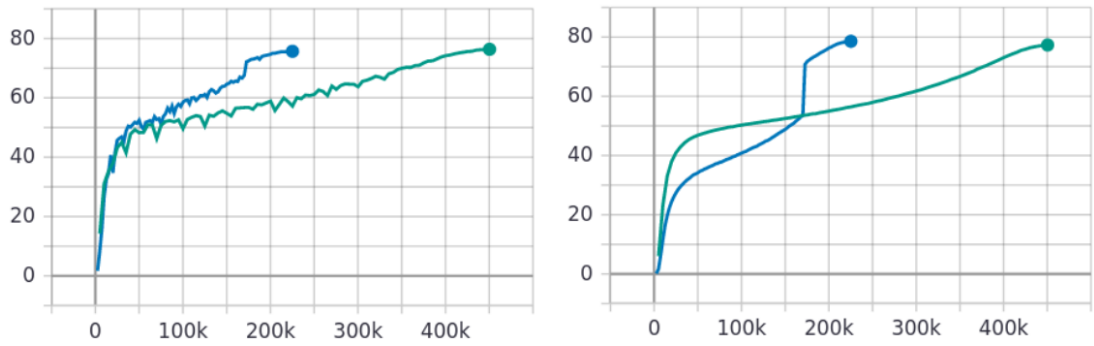


Figure 4.1: Comparative analysis showing the effect of hard sample selection and cool-down. The training curves of uniform sampling (green) against our method (blue) using a ResNet50 backbone architecture on ImageNet. We show per epoch validation accuracy (left) and training accuracy (right) on y axis and training iterations on x axis for both settings. We can see that in the selection stage training accuracy for our method is lower due to training with harder examples even though the validation accuracy training with hard examples results in faster convergence while having lower training accuracy and in the cool-down stage both the training and validation accuracy get a big boost and adapt to the actual data distribution.

training of multiple architectures on large-scale datasets and achieves higher speed-up and better performance than the prior work. Compared to standard training, we achieve up to 1.5 times speed-up while obtaining better or comparable performance. Finally, our method can be easily integrated into existing deep learning libraries by adding only a few lines of code.

4.2 Related Works

Non-uniform sample selection Importance sampling has a long history in deep learning where Hinton (2007) reduces training times of simple neural networks by sampling images proportional to their loss and re-weighting them with the inverse of their selection probability to get unbiased estimate of the original loss. Bengio et al. (2009) proposes curriculum learning to accelerate convergence of deep networks by mimicking how children are taught, starting the training from easy examples and than gradually moving to hard examples. SPL Kumar et al. (2010) and MCL Zhou and Bilmes (2018) follows a similar approach but measure the difficulty of the samples based on the per sample loss and select samples in a bi-level optimization framework.

Dataset Pruning An alternative strategy for accelerating supervised learning is to

explicitly remove unimportant samples from the training set Paul et al. (2021); Raju et al. (2021). However these methods have to first evaluate the sample importance based on gradient norm, error magnitude Paul et al. (2021) or uncertainty Raju et al. (2021) of the samples that is open to changes during training and especially when more sophisticated augmentation strategies like RandAugment Cubuk et al. (2019) are used. Hence they require pruning at multiple training phases which results in additional cost and complexity.

Sampling from Memory Many works model sample importance by keeping a memory of prediction values and define a selection distribution based on the prediction history Loshchilov and Hutter (2015); Gao et al. (2015); Schaul et al. (2015); Alain et al. (2015). Even though this approach reduces the computational overhead with selection, sampling a stale memory still suffers from the problems associated with changes in the network weights and sample augmentations. Alain et al. (2015) proposes a distributed setting where examples are sampled proportional to stale selection weights. The weights are generated by multiple workers in parallel based on the gradient norms of the individual samples to speed up the selection process. Chang et al. (2017) argues that it is better to select hard examples for easy tasks, while it is counter-productive for tasks with many outliers. Hence they propose uncertainty based importance weights that model the variance of the predictions and their proximity to the decision boundary. DIHCL Zhou et al. (2020) extends this by using a moving average for the importance weights of the samples and selecting samples proportional to their importance weight.

Online Sampling Having fast approximations for sample importance enables online batch selection to be used for accelerating training. Katharopoulos and Fleuret (2018) uses the gradient magnitude of the last layer, which they show as an upper bound for the full gradient norm for selecting samples in an online setting. Most related work to our work Selective-Backprop Jiang et al. (2019) ranks the samples based on their loss to determine their sampling rates and additionally keeps a queue of previous loss values to determine the correct percentile rank for the samples. To reduce the high computational overhead of online selection, they propose to use stale values for the selection weights, which they update only at every k-th epoch while reusing the previous values in between. Unlike Jiang et al. (2019), we reduce selection overhead by downsampling a set of images to estimate the loss values at each iteration. This allows for accurate loss estimation by considering the updated model weights at each iteration while being efficient. In addition, our downsampling strategy does not require a large memory queue, unlike Jiang et al. (2019). Our method also includes a cool-down

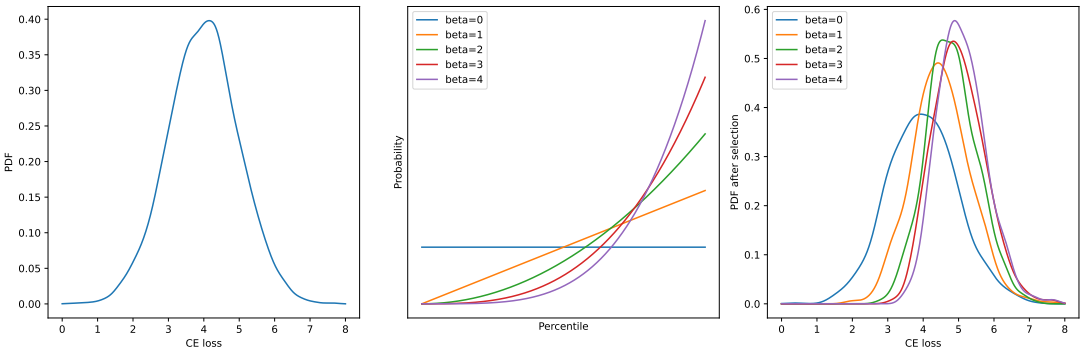


Figure 4.2: (left) Illustrative initial loss distribution (Gaussian loss with mean 4). (middle) Selection coefficients $p(\boldsymbol{x}_i)$ of each sample against their percentile ranking for different values of β. (right) Loss distribution after selection for different values of β.

period at the end of the training, where the examples are uniformly sampled, to reduce potential bias in data distribution caused by the hard example sampling and to obtain a better performing model for evaluation.

4.3 Method

Let D be labeled dataset consisting of $|D|$ image-label pairs $(\boldsymbol{x}, y)$. Our goal is to learn a prediction rule $f_{\boldsymbol{\theta}}$ instantiated as a deep neural network with parameters $\boldsymbol{\theta}$ that takes in an image $\boldsymbol{x}$ and predicts the corresponding class y. Assuming that the samples are identically independently distributed (IID), the parameters $\boldsymbol{\theta}$ can be learned by solving the following optimization problem:

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{x},y)\sim D} \ \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y) \tag{4.1}$$

where ℓ is cross-entropy loss and y is the ground-truth class label for image $\boldsymbol{x}$.

In practice, the optimization for large datasets is often solved by minimizing loss values averaged over a minibatch of samples through a stochastic gradient descent (SGD) procedure:

$$L_{\theta}^{(B)} = \frac{1}{|B|} \sum_{(\boldsymbol{x},y)\in B} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y)), \tag{4.2}$$

where B is a mini-batch consisting $|B|$ image-label pairs ($|B| \ll |D|$) which are sampled with uniform probability distribution without replacement from D. Once all the samples are seen (an epoch), the minibatch sampling restarts from the whole D.

In this paper, we hypothesize that not all training samples are equally informative and their importance varies through the training. To this end, we propose sampling

them based on their difficulty level, i.e. harder images are more likely to be sampled during training. We define the sample difficulty based on the cross-entropy loss value $\ell(f_{\boldsymbol{\theta}_t})$ at each optimization step t. There are at least two main challenges to designing a sampling strategy based on difficulty. First, a naive implementation requires tracking the loss values of all samples at each iteration t which is prohibitively expensive. In other words, a naive solution would need to forward pass all the training images to the deep model after each parameter update and only then select the ones with higher loss values. Second, assuming that train and test images come from the same data distribution, using a non-uniform distribution to sample training samples can be seen as effectively training the model on a different data distribution than the original one, which can lead to a discrepancy between train and test distributions and hence poor generalization performance.

To address the first challenge, we propose two speed-up strategies. First, we downsample training images, input the lower resolution images to the deep network f_θ to obtain loss values more efficiently. Note that most modern image classification deep networks (*e.g.* convolution based He et al. (2015) and transformer based Dosovitskiy et al. (2020) with minor modifications in positional embeddings) can ingest varying input sizes up to a minimum size. In Section 4.4, we provide further implementation details and analyze the effect of resolution on the accuracy of the loss computation. Though reducing resolution provides significantly faster loss computation, it can still be expensive to compute the loss over a large training set. Instead we propose to first sample a large minibatch $\bar{B}$ from D via uniform sampling, downsample those images and compute their loss values only. The key idea is to have a online approximation the loss distribution of the full dataset without incurring a large overhead using a large enough minibatch to make the loss calculations.

$$L_{\boldsymbol{\theta}_t}^{(\hat{B})} = \{\ell(f_{\boldsymbol{\theta}_t}(\hat{\boldsymbol{x}}_1), y_1), \ldots, \ell(f_{\boldsymbol{\theta}_t}(\hat{\boldsymbol{x}}_{|\bar{B}|}), y_{|\bar{B}|})\} \tag{4.3}$$

where $\hat{\boldsymbol{x}}$ denotes downscaled $\boldsymbol{x}$. We compute the empirical cumulative density function (P) over $L_{\boldsymbol{\theta}_t}^{(\hat{B})}$ based on the percentile rank of each sample within the batch $\hat{B}$ to approximate the loss distribution. In order to compute the sampling probability $p(\boldsymbol{x})$:

$$p(\boldsymbol{x}_i) := \left[P\big(\ell(f_{\boldsymbol{\theta}_t}(\hat{\boldsymbol{x}}_j), y_j) \leq \ell(f_{\boldsymbol{\theta}_t}(\hat{\boldsymbol{x}}_i, y_i))\big)\right]^{\beta} \quad \forall 1 \leq j \leq |\bar{B}| \tag{4.4}$$

where β is a hardness hyperparameter that determines how aggressively hard examples are selected.

We want to prioritise the more useful samples by shifting the sampling distribution towards samples with higher loss. Once the ranking of the initial samples $\bar{B}$ is

computed we can apply our prioritisation. We use our approximate loss distribtion to define a multinomial probability distribution over $\bar{B}$ and subsample a smaller minibatch $B \sim \left(\begin{smallmatrix} b \\ p(\boldsymbol{x}_i),\ldots,p(\boldsymbol{x}_{\bar{b}}) \end{smallmatrix}\right)$. Note that the sizes of two minibatches, $\bar{b} = |\bar{B}|$ and $b = |B|$ are hyperparameters where $\bar{b} > b$. Finally, we use the samples of the minibatch B in their original resolutions to compute the loss in Eq. (4.2) and backpropagate its gradients to update the model parameters $\boldsymbol{\theta}$. We depict our method in algorithmic format in Algorithm 2 and also provide PyTorch style pseudo-code for our method in Algorithm 3 for the benefit of the practitioner. The algorithm is a simplification of the actual code used to run our experiments and can be integrated in the training of additional models and datasets effortlessly.

We illustrate the effect of β on the loss using a Gaussian loss distribution in Fig. 4.2. The initial loss distribution is depicted on the left. For different β values different amounts of distributional shift is going to be applied using Eq. (4.4) to subsample the batch B. The scaling for each probability value is based on its percentile rank which is visualized in the middle. Once the selection have been applied the distribution of the losses is shown on the right. $\beta = 0$ corresponds to uniform sampling, and as β gets bigger, harder samples are selected more frequently.

There is a balance for the value β similar to the exploration exploitation in prioritised experience replay Schaul et al. (2015). Having too large values for β causes the model to use the highest shot term valuable samples similar to the exploitation strategy while using low values for β allows the model to try different samples irrespective of their loss values which is similar to the exploration strategy. We will analyse the balance for the β value in Sec. 4.4.4.

Discussion In our experiments, we observed that a forward pass requires approximately a third of the computation time of a forward-backward pass. The approximate relative overhead of running a forward pass on the lower resolution images w.r.t. the original resolution can be calculated as $\frac{1}{3}(\frac{\hat{r}}{r})^2$ for convolutional neural networks and $\frac{1}{3}(\frac{\hat{r}}{r})^3$ for transformer based networks where r and $\hat{r}$ denote the input sizes in one dimension (*e.g.* width) for original and low-res images.

To mitigate the second challenge, the discrepancy between train and test distributions, we introduce an additional training stage, *cool-down* after the model parameters are learned on the data distribution that is obtained with the proposed sampling strategy we apply uniform sampling until the end of the training. Since the cool-down stage is using uniform sampling it doesn't contribute to the sampling overhead. We demonstrate the effect of having a cool-down stage visually in Fig. 4.3. We show the training and

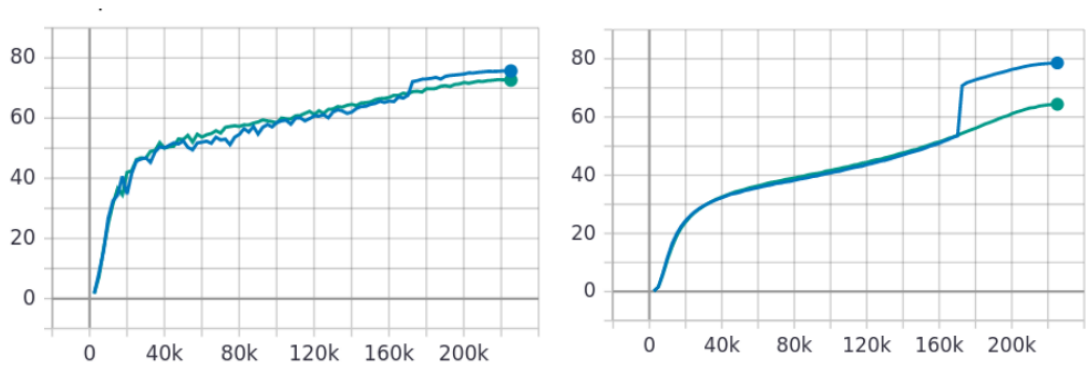


Figure 4.3: Comparison showing the benefit of cool-down with ResNet50 architecture on ImageNet. Training curves for component ablation without cool-down (green) and our method with cool-down (blue) is shown. We show per epoch validation accuracy (left) and training accuracy (right) on y axis and training iterations on x axis for both setting.

validation accuracy for our sampling strategy with (blue) and without cool-down (green). We can see that the performance of both methods are nearly identical until the start of the cool-down period. When the cool-down period starts there is a dramatic increase in both the training and validation accuracy for our method. We hypothesize that this effect is a result of the model shifting to the actual data distribution after being trained on harder examples and seeing all classes and sample difficulties.

As also observed in Chang et al. (2017), hard example sampling can bias the model towards a subset of the dataset that contains particularly hard classes. To this end, we further train the model towards the end of our training schedule by using the standard uniform sampling. In addition, we observed that hard-example optimization of certain deep network architectures like ViTDosovitskiy et al. (2020) can be unstable at early iterations. Similar findings have been shown theoretically and empirically in Katharopoulos and Fleuret (2018). Hence, we include a short *warm-up* stage at the beginning of the training, where the uniform sampling is applied.

4.4 Results

In this section we compare our method to the prior work in Section 4.4.1, evaluate our method on several backbone architectures in Section 4.4.2 and large-scale image classification datasets in Section 4.4.3 and analyze the performance of our method in ablation studies in Section 4.4.4. We evaluate our method in terms of top-1 classification performance (ACC), number of training steps (#TS) – which does not contain the

Algorithm 2 Fast Online Minibatch Selection

given $D, \beta, \bar{b}, b, t_{warm-up}, t_{cool-down}$
initialize $t \leftarrow 1$
repeat
  if $t > t_{warm-up}$ **and** $t < t_{cool-down}$ **then**
    $\bar{B} \leftarrow \bar{b}$ {uniformly sampled datapoints from D }
    $\hat{B} \leftarrow downsample(\bar{B})$
    use Eq. (4.3) to calculate $L_{\theta_t}^{(\hat{B})}$ {loss for downsampled initial batch }
    $P \leftarrow sort(L_{\theta_t}^{(\hat{B})})$ {get empirical Cumulative Distribution Function P }
    use Eq. (4.4) to calculate $p(\boldsymbol{x}_i) \quad \forall i \in \hat{\mathcal{B}}$ {calculate selection coefficients }
    $B \leftarrow \bar{B}[\binom{b}{p(\boldsymbol{x}_i),\ldots,p(\boldsymbol{x}_{\bar{b}})}]$ {sample b images with Multinomial dist. from $\bar{B}$ }
  else
    $B \leftarrow b$ {uniformly sampled datapoints from D }
  end if
  $\theta_t \leftarrow \text{sgd\_step}(B, \theta_{t-1})$
  $t \leftarrow t + 1$
until convergence

selection overhead, total computation cost (FLOPs), and hours of training in a single NVIDIA V100 (GPUh)

Implementation Details Here we provide the details for the default setting where we use ResNet-50 architecture He et al. (2015), and SGD optimizer with momentum weight 0.9 on ImageNet Deng et al. (2009). We use the cosine annealing learning rate scheduler with a learning rate of 0.1, training batch size ($|B|$) of 256, and a weight decay value of 0.0001 for all the baselines and our method. We set the hardness parameter β to 1, the large minibatch size ($\bar{B}$) to be double the small minibatch size, the image resolution to be 224 pixels, the downsampled image resolution for subsampling ($\hat{r}$) to be 96 pixels, warm-up duration to be 10k training steps and cool-down duration to be around 10% of the total training steps. The training duration for the ImageNet baselines is set as 90 epochs, and for our method, we train until we reach the baseline performance and report the acceleration in the number of steps. [1]

[1] Source code will be released upon publication.

Algorithm 3 Fast Online Minibatch Selection, PyTorch-like

```
# f: network parameterized by theta
# criterion: Loss function
# beta: selection difficulty
# init_batch_size: initial batch size
# batch_size: training batch size
# r_sel: selection resolution
# warm_up: number of warm up epochs in the beginning
# cool_down: start epoch for cool down period at the end

probs = torch.pow(torch.arange(init_batch_size)/init_batch_size, beta)

for e in range(epochs):
    for x in loader: # load initial batch B uniformly from D
      if e > warm_up and e < cool_down:
        x = fast_select(x) # Importance Sampling
      y_hat = f(x) # forward-propagation

      L = criterion(y_hat) # loss calculation
      L.backward() # back-propagation

def fast_select(x): # Callback on_after_batch_transfer
    x_sel = torch.nn.functional.interpolate(x, r_sel) # downsample to r_sel
    y_sel = f(x_sel) # forward propagate initial batch with low-res approx.
    l_sel = criterion(y_sel) # loss calculation
    sorted_idx = torch.argsort(l_sel, dim=-1, descending=False)
    # Sample based on percentile rank
    sel_idx = torch.multinomial(probs, batch_size, replacement=False)
    selection = sorted_idx[sel_idx]
    return x[selection]
```

4.4.1 Comparison to State-of-the-art

First, we compare our method to the standard uniform sampling baseline and the state-of-the-art methods, MCL Zhou and Bilmes (2018), DIHCL Zhou et al. (2020), Selective-Backprop Jiang et al. (2019) that also employ non-uniform sample selection to accelerate training. We evaluate all the methods in the ImageNet dataset using the ResNet50 backbone and report the results in Table 4.1. MCL and DIHCL results are copied from the original papers, as they report on the same backbone and dataset. We report Selective-Backprop results based on our implementation, as the authors do not report on ImageNet. Note that MCL and DIHCL originally compared their method to a baseline with significantly longer training time (200 epochs), which is easier to obtain gains in training time. Here, we follow a more competitive setting and follow the default training procedure originally proposed by the authors of ResNet50 He et al. (2015) (90 epochs).

Our results achieve a significant speed-up of 1.3 over the uniform sampling baseline while achieving slightly better classification accuracy. While DIHL outperforms MCL both in terms of classification accuracy and training speed, Selective-Backprop achieves better speed-up. Finally, our method outperforms the prior works while providing better

Model	ACC(%)	#TS	PFLOP	GPUh
Standard Uniform	76.1	450k	1314	77
MCL Zhou and Bilmes (2018)	75.1	285k (0.6x)	N/A	85 (1.1x)
DIHCL Zhou et al. (2020)	76.3	285k (0.6x)	N/A	70 (0.9x)
Selective-Backprop Jiang et al. (2019)	75.5	300k (0.7x)	935 (0.8x)	55 (0.8x)
Ours	76.4	300k (0.7x)	926 (0.8x)	54 (0.8x)

Table 4.1: Comparison to state-of-the-art methods in classification accuracy (ACC), number of training steps (#TS), number of floating point operations in petaflops during training (PFLOP), and training time in hours on a single GPU (GPUh) on ImageNet trained using ResNet-50. When calculating speed-up for prior art, we used the Standard Uniform setting as our baseline for training cost. We report re-implemented results for Selective-Backprop

classification and speed-up tradeoff. Interestingly we observe that Selective-Backprop obtains lower classification accuracy than the uniform baseline. We reason that the sampling strategy in Selective-Backprop leads to focusing on harder examples, while our method closes this performance gap by a later cool-down stage.

4.4.2 Backbone Architecture Analysis

Here we evaluate our method on various state-of-the-art image classification deep network architectures including ResNet50, ResNet152 He et al. (2015), DenseNet161 Huang et al. (2017), RegNetX3.2gf Radosavovic et al. (2020) and ViT-S/16-90e Dosovitskiy et al. (2020) in ImageNet and report results in Table 4.2. We use the default hyperparameters for ResNet50, ResNet152, and DenseNet161. For the RegNetX3.2gf Radosavovic et al. (2020) architecture we used a batch size of 512, learning rate of 0.4 with cosine annealing learning rate schedule with 5 epochs of linear warm up and weight decay of 0.00005. For the ViT architecture Dosovitskiy et al. (2020) we followed Beyer et al. (2022) hyper-parameter setting on ImageNet-1k Deng et al. (2009) training since the original paper did not provide details for running this architecture on the ImageNet-1k dataset. We use batch size 1024, weight decay of 0.0001 learning rate of 0.001 with cosine annealing learning rate with 10 epoch warm-up and RandAugment Cubuk et al. (2019) for training augmentations. We did not use MixUp Zhang et al. (2017a) since it interfered with our selection algorithm and added a 20 epoch warm-up

Architecture	Uniform(%)	Ours(%)	xTS	xFLOPs
ResNet50	76.1	76.1	x1.5	x1.4
ResNet152	78.4	78.3	x1.5	x1.4
DenseNet161	78.9	78.5	x1.4	x1.3
RegNetX3.2gf	77.8	77.8	x1.7	x1.5
ViT-S/16	73.3	73.2	x1.5	x1.5

Table 4.2: Analysis of backbone architecture in ImageNet in terms of classification accuracy (ACC), acceleration in training steps (xTS) and floating point operations(xFLOPs).

stage where we used standard training in the beginning. We use a convolutional Patch Encoder as well as a fixed 2D cos-sin positional embeddings as described in Chen et al. (2021). In order to process images at multiple resolutions we calculate multiple fixed positional embedding at different resolutions and depending on the input use the appropriate one.

Note that all the networks are trained for 90 epochs, and ViT-S/16 originally reports higher results for significantly longer training time (300 epochs).

Our method boosts the training speed with all the backbones while retaining the original classification accuracy. This clearly shows that our method can be used with very diverse architectures, including convolutional and transformer based ones. ViT architectures have negligible selection overhead due to their cubic complexity in terms of image resolution, which is evident in their speed-ups in training steps and computation. Interestingly our method shows the highest speed-up over RegNetX3.2gf architecture. We hypothesize that our method leverages strong data augmentation in its training and generates examples more efficiently.

4.4.3 Additional Dataset Results

We evaluate our method on two additional datasets, Places365 Zhou et al. (2017a) and Food-101 Bossard et al. (2014) in Table 4.3, and employ the architectures (ResNet152 and ResNet50 respectively) that are commonly used in these datasets. We use the default hyper-parameter setting except for using RandAugment Cubuk et al. (2019) on the Places365 dataset. On the Food-101 dataset we train for 40K steps in the baseline while having 4k steps long warm-up stage for our method. As in ImageNet experiments, our method boosts the training speeds while preserving the original classification performance in both datasets. We observe that the speed-up is lower in

Dataset	Architecture	Uniform(%)	Ours(%)	xTS	xFLOPs
Places365	ResNet-152	56.9	56.9	x1.5	x1.4
Food-101	ResNet-50	80.1	80.0	x1.3	x1.2

Table 4.3: Analysis of different datasets in terms of classification accuracy for uniform and our sampling method, acceleration in training steps (xTS) and floating point operations(xFLOPs).

Food-101 dataset. We reason that the presence of noisy labels in the datasets reduces the efficiency, as those examples are sampled frequently due to their high loss values.

4.4.4 Ablation studies

Here we run a series of ablation studies for different components and hyperparameters. In all ablation studies, we train our model on ResNet50 architecture on ImageNet and follow the default hyperparameters setting except for the hyper-parameters being examined.

Components In order to understand the usefulness of each component of our method, we run an ablation study in Table 4.4. For the selection methods, we compare uniform sampling, greedy selection (top 50% samples selected w.r.t. loss) and our probabilistic selection described in Eq. (4.4). We see that using a greedy selection strategy is detrimental for training while having only our probabilistic selection gives better results it can only reach uniform sampling level with the same number of training steps. Adding a cool-down stage gives a boost to training accuracy in the end and allows us to reach full baseline accuracy. For a visual comparison between the baseline and our method see Fig. 4.1 and to see the effect of cool-down see Fig. 4.3 in the supplementary material.

Dataset Size We examine our method on subsets of ImageNet Deng et al. (2009) to see the effect of dataset size on our method. We used the standard ImageNet-1k dataset and its subsets ImageNet100[2] and ImageNet10 (Imagenette) [3] dataset that contain 100 and 10 classes respectively in Table 4.5. We use ResNet-50 for the backbone with the default hyper-parameters except for ImageNet10, where we used batch size of 128, learning rate of 0.03 and weight decay value of 0.0005. Our results indicate that we can train our method with up to 10K sample sizes on general image recognition problems in fewer steps without sacrificing performance.

[2]available at https://www.kaggle.com/datasets/ambityga/imagenet100

[3]available at https://github.com/fastai/imagenette

Selection	Cooldown	ACC(%)	#TS	PFLOP	GPUh
-	-	76.1	450K	1314	77
-	-	75.5	300K	876	51
Greedy	-	74.7	300K	912	53
Eq. (4.4)	-	75.5	300K	912	53
Eq. (4.4)	30k steps	76.1	300K	906	53

Table 4.4: Experimental setup to disentangle the contribution of each component in terms of classification accuracy (ACC), number of training steps (#TS), number of floating point operations in petaflops during training (PFLOP) and training time in hours on a single GPU (GPUh) trained on ImageNet with ResNet50 backbone.

Classes	Size	Uniform(%)	Ours(%)	xTS	xFLOPs
1000	1.2M	76.1	76.1	x1.5	x1.4
100	100k	84.1	83.6	x1.5	x1.4
10	10k	91.9	92.4	x1.4	x1.3

Table 4.5: Analysis for varying dataset size on ImageNet using ResNet50 as the backbone. Results are reported in terms of classification accuracy (ACC), acceleration in training steps (xTS) and floating point operations(xFLOPs).

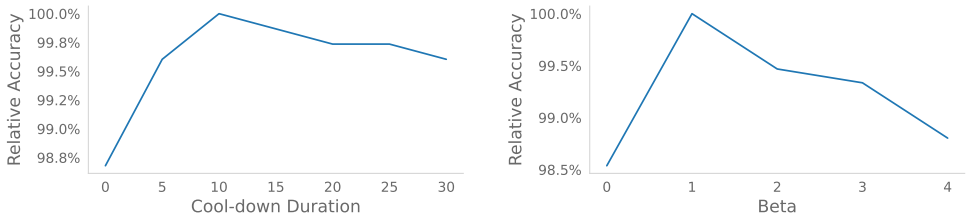


Figure 4.4: (left) Cool-down ablation where on x-axis different cool-down durations as a fraction of training time and the y-axis accuracy of the model is shown relative to the best performing result (right) Selection hardness ablation where on x-axis different values of the β parameter and on the y-axis accuracy is shown relative to the best performing result.

Cool-down duration Cool-down is an important portion of our method that allows the model to adapt to the original data distribution after being trained on the hard samples. To study the significance of the cool-down period, we run an ablation study with default hyper-parameters where we only change the cool-down duration as a percentage of training length in Fig. 4.4 (left). We see that the optimum value for the cool-down period duration is around 10% of training time. A shorter duration does not allow for adaptation to take place, while longer durations waste time with unnecessary uniform sampling. A comparison for the training curves between our method and the baseline can be seen in our supplementary material Fig. 4.3.

Selection Hardness Next we study the selection hardness which determined by β. We vary β while keeping the other hyperparameters at their default values, and show our results in Fig. 4.4 (right). We set the initial batch size to 5x of the training batch in this study in order to have a good approximation of the loss distribution even when a high value for β is used. We see that high values for β are detrimental for performance and the best results are obtained when we have a linear scaling ($\beta = 1$) for the selection coefficients. Normally since having a unnecessarily large initial batch causes additional selection overhead, we maintain a initial batch size that is at least $1 + \beta$ times the training batch size.

Selection Resolution We finally study the impact of the image resolution used in sample selection(Table 4.6). We maintain the default hyperparameters and only change the selection resolution. While reducing the image resolution speeds up training, it hurts the performance. We find the resolution 96×96 delivers the best performance and speed-up tradeoff.

ImageNet, ResNet50	$\hat{r} = 64$	$\hat{r} = 96$	$\hat{r} = 128$	$\hat{r} = 224$
Accuracy (%)	75.5	76.1	76.4	76.4
Overhead (%)	4.7	10.5	18.7	57.2

Table 4.6: The impact of image resolution $\hat{r}$ during importance weight estimation. Trained for 300K steps

4.5 Future Work

We propose a direction where our work can be extended by first analysing our contribution the limitations of our approach and than suggesting ways for improving it. Many

things in nature are distributed with the Pareto distribution where 80% of the benefit can be achieved by only 20% of the effort. If we assume that the utility of each sample in the batch similarly is distributed with the Pareto distribution having a Cumulative Density Function F,

$$F(x) = 1 - (\frac{x_{min}}{x})^{\alpha} \tag{4.5}$$

where x_{ming} and α are parameters of the distribution.

We can model the fast online minibatch selection algorithm described in Section 4.3 using a Pareto distribution and predict potential speedups that can be achieved. Fig. 4.5 Shows the CDF for different parameters of the distribution as well as potential speedups that can be achieved when this distribution is being used.

An interesting observation that can be made here is that in order to achieve higher speedups higher selection ratios need to be made while this is only beneficial up to a certain extent where beyond a certain value the overhead of selection simply becomes too large. This analysis also indicates that if we can improve the selection accuracy by having a larger Pareto Multiplier we can obtain significantly better speedups for online selection by using larger selection ratios.

We propose that more accurate selection method that uses a larger selection ratio has the potential to produce much better speedups. One can add a memory for gauging how fast a sample is being forgotten in order to improve the selection accuracy. Using gradient norm instead of loss value can also be used for improving selection accuracy.

4.6 Conclusion

We show a fast online importance sampling method that increases training speed between 1.3 and 1.5 times without sacrificing accuracy while reducing the computational overhead of selection by 5.4 times. Our method is architecture agnostic and can be easily integrated into existing deep learning methods to reduce training costs by adding a few lines of code. A limitation of our method is that it does not accelerate training when the MixUp augmentation Zhang et al. (2017a) is used because it interferes with the selection process. Addressing this issue and designing a higher quality approximation of sample importance can be done as future work.

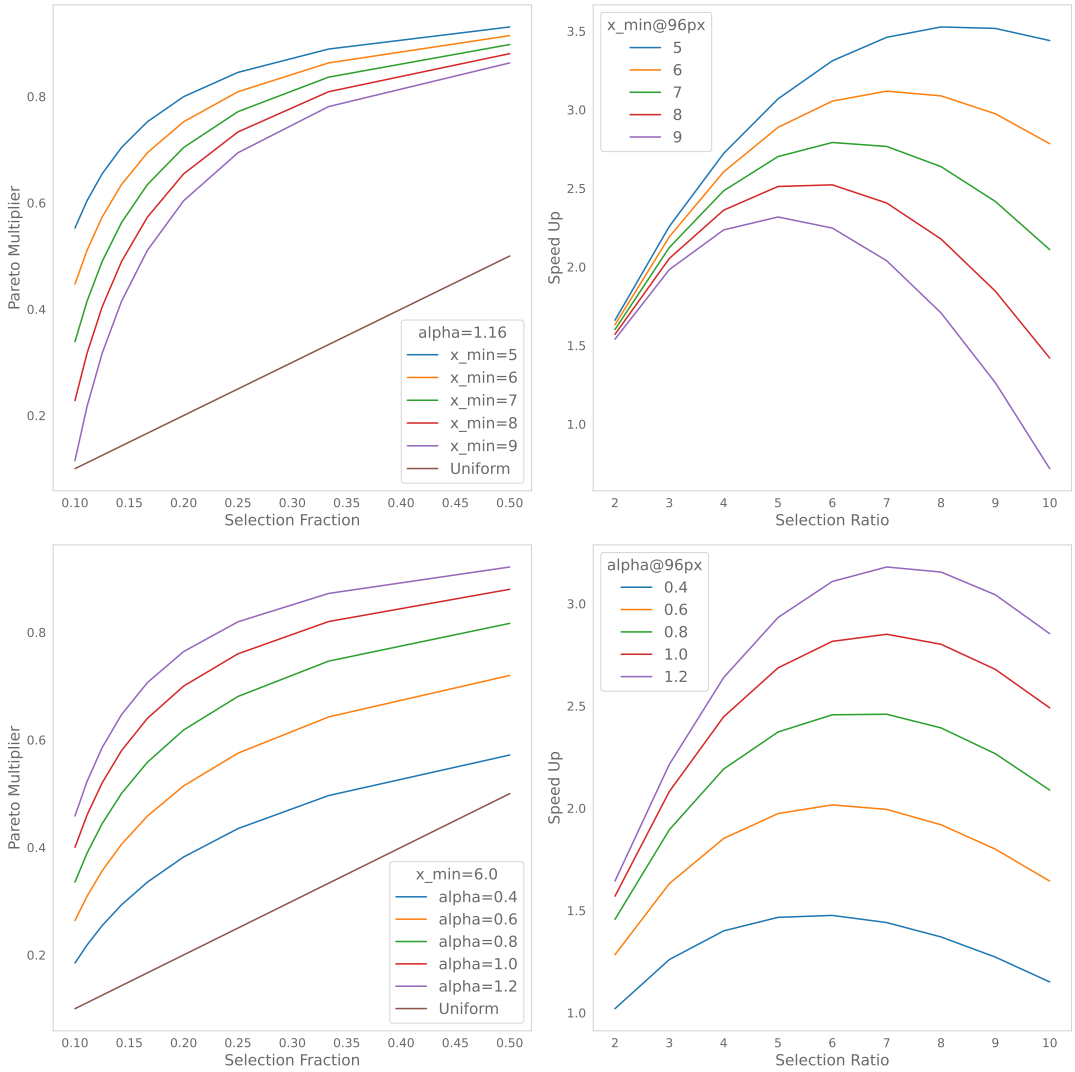


Figure 4.5: This figure aims to show the theoretical maximum speedup that can be achieved by a 2 stage online batch selection method assuming that the utility of each sample is distributed with the Pareto distribution. First column shows shows visualization of the Cumulative Density Function of the Pareto distribution for different values of x_{min} and α. The second column speedup that can be achieved for different values of x_{min} and α given the selection ratio when the selection resolution is 96. First row has $\alpha = 1.16$ while second row has $x_{min} = 6$

Chapter 5

Conclusion

In this thesis we have shown efficient training methods in three different settings that are all model-agnostic, easy to implement and have minimal overhead.

Chapter 2 demonstrates that UBN reduces the bias in batch statistics and applies a regularization that utilizes the data manifold. Our method is effective in problems where obtaining annotations is difficult and allows reaching the same accuracy with around half the number of annotated samples.

Chapter 3 shows that self-supervised training can be accelerated by adapting the learning rate, augmentation and resolution schedules for self supervised training and boosting the training signal by hard positive mining on the augmentations. Our method have shows an training speed-up between 2.3 and 2.7 times.

Chapter 4 introduces a fast online importance sampling for supervised learning method that increases training speed between 1.3 and 1.5 times without sacrificing accuracy while reducing the computational overhead of selection by 5.4 times.

Our work allows a much wider community to reproduce and contribute to computer vision literature, reduce the financial and carbon cost of training these models. Faster training times allows faster iteration between ideas and helps researchers make accelerated progress. By using these techniques at the large scale progress better performance can be obtained while using the same computational budget.

5.1 Limitations

Due to resource limitations that we had we were not able to examine all dataset and model combinations that we wanted to do. In Chapter 3 we used a subset of the original dataset in order to validate our method. It would be particularly interesting to apply our

method to recent methods Caron et al. (2021) that used Vision Transformer Dosovitskiy et al. (2020) backbones which we intend to pursue this as future work.

A limitation of fast online minibatch selection is that it does not accelerate training when the MixUp augmentation Zhang et al. (2018) is used because it interferes with the selection process since this is one of the regularization method that tends to be used fairly often it would be beneficial to have a complementary implementation of fast online minibatch selection.

Even though this was not part of our claims it would be valuable to see the generalizability of our methods outside of computer vision. The scaling challenges that we addressed are also present in different forms in Natural Language Processing and Reinforcement Learning.

5.2 Future Work

The research conducted in this thesis has highlighted several promising avenues for future work in the context of improving the efficiency of neural network training. These potential areas of further research are drawn from a diverse set of chapters, underscoring the broad impact of our findings. We organize them the order of the chapters.

5.2.1 Expansion and Validation of Unsupervised Batch Normalization

In our investigation of UBN, we observed a significant potential for enhancing the training and generalization capabilities of neural networks, particularly on smaller datasets. While the current results are promising, we suggest future work to further explore and validate the universality of UBN across a variety of problem types. Specifically, additional experimentation could be undertaken in areas such as object recognition, image segmentation, and pose estimation with popular benchmark datasets like ImageNetDeng et al. (2009), COCOLin et al. (2014), and MPII Human Pose DatasetAndriluka et al. (2014).

The analysis of UBN to reduce the bias in batch statistics and apply a regularization that utilizes the data manifold is certainly interesting with regards to understanding working mechanisms behind batch normalization. We also noted that it is particularly effective in scenarios where obtaining annotations is challenging, pointing towards potential applications in semi-supervised and unsupervised learning contexts. Successfully

applying UBN to these contexts could yield new insights into the manifold structures of these datasets and significantly improve the performance of models in cases where labeled data is scarce.

5.2.2 Scaling Efficient Self-Supervised Learning Strategies

Our exploration into accelerating self-supervised learning has opened up possibilities for making this training paradigm more accessible and less resource-intensive. We demonstrated substantial training speed-ups by adapting learning rate, augmentation, and resolution schedules, coupled with hard positive mining. We believe that these strategies could be further refined and combined with other efficiency-focused techniques.

One potential risk identified was that the wider adoption of efficient training methods might lead to the use of larger and more computationally expensive benchmarks, offsetting some of the benefits. Future work should aim to maintain a balance between performance and computational efficiency.

Additionally, applying our acceleration techniques to other cutting-edge methods, such as those that utilize Vision Transformer backbones Dosovitskiy et al. (2020), will be a critical step in pushing the boundaries of self-supervised learning. The use of more complex datasets, like ImageNet-21kRidnik et al. (2021), could also be considered to validate the scalability of these strategies.

This research trajectory holds promise to significantly reduce the entry barriers to self-supervised learning, opening doors for many more researchers and practitioners to contribute to this exciting field.

5.2.3 Refining Online Minibatch Selection for Supervised Learning

Our findings with regards to fast online minibatch selection underscored a new direction for supervised learning, achieving an increase in training speed without sacrificing accuracy. We anticipate that this strategy could be further developed, focusing on enhancing selection accuracy and optimizing the balance between selection ratio and selection overhead.

Notably, our method relied on a conservative selection ratio however our analysis have showed that much higher speed-ups can be achieved by improving selection accuracy. Future work could involve incorporating a memory component to gauge how quickly a sample is being forgotten, using gradient norm instead of loss value to

improve selection accuracy and allowing the use of larger selection ratios to obtain better speed-ups. These improvements, individually or in combination, could significantly advance the performance and efficiency of online minibatch selection. Implementing these improvements could open up an array of applications where rapid model training is critical.

Finally, a limitation in our method was the inability to accelerate training when the MixUp augmentation is used, as it interferes with the selection process. Future research could explore how to resolve this conflict and enable the use of this and other similar data augmentations with online minibatch selection. Overcoming this limitation could greatly broaden the applicability of our method, potentially introducing a new standard for data augmentation in the context of efficient training.

In summary, the research presented in this thesis opens several exciting avenues for future work. The progression in each of these areas will not only enhance our understanding of efficient neural network training techniques but also make these methods more accessible and environmentally sustainable.

Appendix A

Accelerating Self-Supervised Learning via Efficient Training Strategies

A.1 Speed Up Calculation

While Super Progressive Learning reduces the number of floating point calculations made by using a smaller resolution, Hard Augment adds an additional overhead for training which can be calculated by equations 3.9 and 3.11 respectively. Generally we find that we can reduce the number of steps by 1.7 times while reduce the number of floating point operations at each step by 1.4 times.

A.2 Additional Implementation Details

	Imagenette		ImageNet	
Hyper-parameter	Baseline	Efficient	Baseline	Efficient
Number of classes	10	10	1000	1000
Batch size	128	128	512	512
Weight decay	5×10^{-4}	5×10^{-4}	1×10^{-4}	1×10^{-4}
Learning rate	0.05	0.1	0.1	0.16
LR schedule	CA	ESC	CA	ESC
Warmup epochs	0	80	10	10
Momentum weight	0.9	0.85-0.95	0.9	0.85-0.95
Learning rate	0.05	0.05	0.05	0.16
Min aug. magnitude	5	4	5	4
Max aug. magnitude	5	6	5	6
Min view resolution	224	96	224	96
Number of positives	2	6	2	6
Selection resolution	N/A	64	N/A	64

Table A.1: Hyper parameters used in various setups. CA: Cosan Annealing, ESC: Extended Super Convergence

We have trained all our experiments in the single node setting with maximum 8 GPUs which restricted us to use 512 as the maximum batch size. Some of the SSL methods like SimCLR Chen et al. (2020a) and BYOL Grill et al. (2020) have been shown to perform better with larger batch sizes which we could not replicate and restricted ourselves to the same training setting in all our experiments.

In some of the ablations and analysis experiments where we study the effect of a certain parameter different values can be used. For Linear evaluation we use freeze the

encoder and re-initialize the last layer and train with batch size 2048 and learning rate 0.8 for 90 epochs using the LARS optimizer.

Our implementation is based on PyTorch Lightning Falcon et al. (2019) where we define Hard Augmentation and Super Progressive Learning Schedule as callbacks. We adapt the OneCycle Learning rate schedule implementation[1] for Extended Super Convergence. We will provide the code for our method upon publication. See the table for a full list of parameters Table A.1.

A.3 Learning rate range finder

We have used the Learning rate range finder test proposed by Smith and Topin (2017) to find good values for minimum and maximum learning rates. This test increase learning rate exponentially and keeps track of on the training and validation loss. The minimum value and maximum value are determined by the point at which the validation loss starts to decrease and the point at which is starts to diverge. We calculate the validation loss at each step on a batch of 4096 validation samples in order to keep the test manageable.

In the test shown in Figure A.1 we increase the learning rate from 1×10^{-3} to 1 in 200 steps. Both the training and validation losses plateau close to step 135 and learning rate .1 which we use in our experiments on Imagenette dataset.

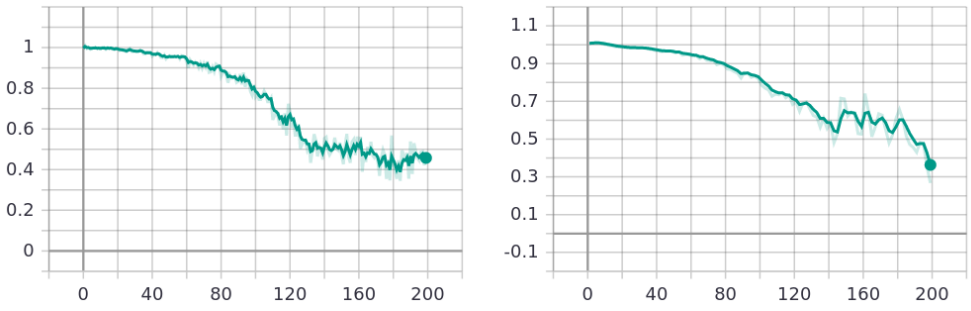


Figure A.1: Learning rate finder plot for training (left) and validation (right) losses

A.4 Augmentation Resolution Relationship

In order to examine the relationship between resolution and augmentation magnitude for self-supervised learning we have conducted a series of experiments. We trained

[1] https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.OneCycleLR.html

a BYOL Grill et al. (2020) model on the Imagenette dataset using the ResNet18 He et al. (2015) architecture with RandAugment Cubuk et al. (2019) augmentation with the cosine annealing learning rate schedule where we change the augmentation strength m while we change the input image resolution. Each entry in the table is a separate experiment with fixed input resolution and augmentation magnitude. The results in Table A.2 confirms that we can apply higher magnitude augmentations only on higher resolution images and the optimum augmentation magnitude increases with resolution. This confirms our intuition behind the relationship between resolution and augmentation magnitude in SSL and provides motivation for applying Super Progressive Learning and allows us to confirm the findings of Tan and Le (2021) for SSL.

	m=3	m=5	m=7	m=10	m=15
128	85.0	84.0	81.4	77.7	75.3
192	88.8	87.7	88.9	86.8	87.3
300	89.5	89.7	89.9	88.3	87.7

Table A.2: Resolution and Augmentation Magnitude relationship shown by training a self-supervised learning method on the combination of resolution and magnitudes and measuring the online classification accuracy of a linear layer

A.5 Progressive Augmentation Curriculum

However what values should be used as the minimum and maximum augmentation magnitude is an important question. Since we use the linearly scaled SimCLR Chen et al. (2020a) augmentations in our method we made a hyper-parameter search on these values in order to determine augmentation magnitudes empirically. We train our method for 320 epochs with Super Progressive learning schedule using 128 as the minimum resolution for various augmentation magnitudes and compare against the default augmentation magnitude of 5. Our experiment shows that a minimum value of 4 and maximum value of 6 perform better in our setting and outperform the fixed augmentation setting.

m_{min}	m_{max}	Acc(%)
5	5	88.6
2.5	4	88.4
3	4	88.7
4	5	88.7
5	6	88.7
4	6	88.9

Table A.3: Maximum and minimum augmentation magnitude when trained with Super Progressive learning

A.6 Future Work

This idea can be extended to other modalities where resolution is defined in different manners for example on sound with the sampling rate analogue which in a similar sense accelerates the training however than the amount of acceleration and cost benefit calculation will change. A similar case can also be made for tabular data where the most important features are processed first and later additional features are added however this is much more difficult to implement would probably require additional structure. Similarly training with a small vocabulary and than enlarging that can have a similar effect as well.

Bibliography

Akiba, T., Suzuki, S., and Fukuda, K. (2017). Extremely large minibatch sgd: Training resnet-50 on imagenet in 15 minutes. *arXiv preprint arXiv:1711.04325*.

Alain, G., Lamb, A., Sankar, C., Courville, A., and Bengio, Y. (2015). Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*.

Amiri, A. J., Loo, S. Y., and Zhang, H. (2019). Semi-supervised monocular depth estimation with left-right consistency using deep neural network.

Amodei, D. and Hernandez, D. (2018). Ai and compute. *OpenAI Research (blog)*.

Andriluka, M., Pishchulin, L., Gehler, P., and Schiele, B. (2014). 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, pages 3686–3693.

Azizi, S., Mustafa, B., Ryan, F., Beaver, Z., Freyberg, J., Deaton, J., Loh, A., Karthikesalingam, A., Kornblith, S., Chen, T., et al. (2021). Big self-supervised models advance medical image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3478–3488.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. (2019). Mixmatch: A holistic approach to semi-supervised learning. *arXiv:1905.02249*.

Beyer, L., Zhai, X., and Kolesnikov, A. (2022). Better plain vit baselines for imagenet-1k. *arXiv preprint arXiv:2205.01580*.

Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. (2018). Understanding batch normalization. In *Advances in Neural Information Processing Systems*, pages 7694–7705.

Bordes, A., Ertekin, S., Weston, J., Botton, L., and Cristianini, N. (2005). Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(9).

Bossard, L., Guillaumin, M., and Van Gool, L. (2014). Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*.

Branwen, G. (2021). The scaling hypothesis.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660.

Chang, H.-S., Learned-Miller, E., and McCallum, A. (2017). Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems*, 30.

Chaudhari, P. and Soatto, S. (2018). Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–10. IEEE.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020a). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E. (2020b). Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255.

Chen, X., Fan, H., Girshick, R., and He, K. (2020c). Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

Chen, X. and He, K. (2020). Exploring simple siamese representation learning. corr abs/2011.10566 (2020). *arXiv preprint arXiv:2011.10566*.

Chen, X., Xie, S., and He, K. (2021). An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9640–9649.

Coleman, C., Narayanan, D., Kang, D., Zhao, T., Zhang, J., Nardi, L., Bailis, P., Olukotun, K., Ré, C., and Zaharia, M. (2017). Dawnbench: An end-to-end deep learning benchmark and competition. *Training*, 100(101):102.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. (2019). Randaugment: Practical automated data augmentation with a reduced search space. arxiv e-prints, page. *arXiv preprint arXiv:1909.13719*.

Deecke, L., Murray, I., and Bilen, H. (2018). Mode normalization. *arXiv preprint arXiv:1810.05466*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Droppo, J. and Elibol, O. (2021). Scaling laws for acoustic models. *arXiv preprint arXiv:2106.09488*.

Falcon et al., W. (2019). Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3.

Fukushima, K. and Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Gao, J., Jagadish, H., and Ooi, B. C. (2015). Active sampler: Light-weight accelerator for complex data analytics at scale. *arXiv preprint arXiv:1512.03880*.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

Gidaris, S., Singh, P., and Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*.

Gong, C., Ren, T., Ye, M., and Liu, Q. (2020). Maxup: A simple way to improve generalization of neural network training. *arXiv preprint arXiv:2002.09024*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Goyal, P., Caron, M., Lefaudeux, B., Xu, M., Wang, P., Pai, V., Singh, M., Liptchinsky, V., Misra, I., Joulin, A., et al. (2021). Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. (2017). Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Goyal, P., Duval, Q., Seessel, I., Caron, M., Singh, M., Misra, I., Sagun, L., Joulin, A., and Bojanowski, P. (2022). Vision models are more robust and fair when pretrained on uncurated images without supervision. *arXiv preprint arXiv:2202.08360*.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., et al. (2020). Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*.

Guizilini, V., Li, J., Ambrus, R., Pillai, S., and Gaidon, A. (2019). Robust semi-supervised monocular depth estimation with reprojected distances.

Guo, X., Li, H., Yi, S., Ren, J., and Wang, X. (2018). Learning monocular depth by distilling cross-domain stereo networks. *Lecture Notes in Computer Science*, page 506–523.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. arxiv 2015. *arXiv preprint arXiv:1512.03385*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hernandez, D. and Brown, T. B. (2020). Measuring the algorithmic efficiency of neural networks. *arXiv preprint arXiv:2005.04305*.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M., Ali, M., Yang, Y., and Zhou, Y. (2017). Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*.

Hinton, G. E. (2007). To recognize shapes, first learn to generate images. *Progress in brain research*, 165:535–547.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hoffer, E., Weinstein, B., Hubara, I., Ben-Nun, T., Hoefler, T., and Soudry, D. (2019). Mix & match: training convnets with mixed image sizes for improved accuracy, speed and scale resiliency. *arXiv preprint arXiv:1908.08986*.

Howard, J. (2018). Training imagenet in 3 hours for usd 25; and cifar10 for usd 0.26.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Huang, X. and Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510.

Ioffe, S. (2017). Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *Advances in neural information processing systems*, pages 1945–1953.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jastrzębski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. (2017). Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*.

Jiang, A. H., Wong, D. L.-K., Zhou, G., Andersen, D. G., Dean, J., Ganger, G. R., Joshi, G., Kaminksy, M., Kozuch, M., Lipton, Z. C., et al. (2019). Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*.

Johnson, T. B. and Guestrin, C. (2018). Training deep models faster with robust, approximate importance sampling. *Advances in Neural Information Processing Systems*, 31.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Katharopoulos, A. and Fleuret, F. (2017). Biased importance sampling for deep neural network training. *arXiv preprint arXiv:1706.00043*.

Katharopoulos, A. and Fleuret, F. (2018). Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer.

Kumar, M., Packer, B., and Koller, D. (2010). Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23.

Kundu, J. N., Uppala, P. K., Pahuja, A., and Babu, R. V. (2018). Adadepth: Unsupervised content congruent adaptation for depth estimation.

Kuznietsov, Y., Stückler, J., and Leibe, B. (2017). Semi-supervised deep learning for monocular depth map prediction.

Larsson, G., Maire, M., and Shakhnarovich, G. (2017). Colorization as a proxy task for visual understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6874–6883.

Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2.

Lewkowycz, A., Bahri, Y., Dyer, E., Sohl-Dickstein, J., and Gur-Ari, G. (2020). The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*.

Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., Paszke, A., Smith, J., Vaughan, B., Damania, P., et al. (2020). Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Loshchilov, I. and Hutter, F. (2015). Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*.

Luo, P., Wang, X., Shao, W., and Peng, Z. (2018a). Towards understanding regularization in batch normalization. *arXiv preprint arXiv:1809.00846*.

Luo, Y., Ren, J., Lin, M., Pang, J., Sun, W., Li, H., and Lin, L. (2018b). Single view stereo matching.

Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048.

Neelakantan, A., Vilnis, L., Le, Q. V., Sutskever, I., Kaiser, L., Kurach, K., and Martens, J. (2015). Adding gradient noise improves learning for very deep networks.

Nesterov, Y. (2012). Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362.

Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer.

NVIDIA, Vingelmann, P., and Fitzek, F. H. (2020). Cuda, release: 10.2.89.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Paul, M., Ganguli, S., and Dziugaite, G. K. (2021). Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. (2020). Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10428–10436.

Raju, R. S., Daruwalla, K., and Lipasti, M. (2021). Accelerating deep learning with dynamic data pruning. *arXiv preprint arXiv:2111.12621*.

Richtárik, P. and Takáč, M. (2014). Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1):1–38.

Ridnik, T., Ben-Baruch, E., Noy, A., and Zelnik-Manor, L. (2021). Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*.

Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in neural information processing systems*, pages 901–909.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Shrivastava, A., Gupta, A., and Girshick, R. (2016). Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Smith, L. (2015). Cyclical learning rates for training neural networks. arxiv. *Preprint at https://arxiv. org/abs/1506.01186*.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*.

Smith, L. N. and Topin, N. (2017). Super-convergence: Very fast training of neural networks using large learning rates. arxiv e-prints, page. *arXiv preprint arXiv:1708.07120*.

Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

Smith, S. L. and Le, Q. V. (2017). A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*.

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852.

Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. (2018). Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Sutton, R. (2019). The bitter lesson. *Incomplete Ideas (blog)*, 13:12.

Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*.

Tan, M. and Le, Q. V. (2021). Efficientnetv2: Smaller models and faster training. *arXiv preprint arXiv:2104.00298*.

Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204.

Teye, M., Azizpour, H., and Smith, K. (2018). Bayesian uncertainty estimation for batch normalized deep networks. *arXiv preprint arXiv:1802.06455*.

Tian, Y., Krishnan, D., and Isola, P. (2020a). Contrastive multiview coding. In *European conference on computer vision*, pages 776–794. Springer.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. (2020b). What makes for good views for contrastive learning? *Advances in Neural Information Processing Systems*, 33:6827–6839.

Touvron, H., Vedaldi, A., Douze, M., and Jégou, H. (2020). Fixing the train-test resolution discrepancy: Fixefficientnet. *arXiv preprint arXiv:2003.08237*.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

Wiskott, L. and Sejnowski, T. J. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. (2018). Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742.

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. (2019). Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

Yu, J., Xu, Y., Koh, J. Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B. K., et al. (2022). Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*.

Zamir, A. R., Sax, A., Shen, W. B., Guibas, L. J., Malik, J., and Savarese, S. (2018). Taskonomy: Disentangling task transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017a). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *International Conference on Machine Learning*.

Zhang, R., Isola, P., and Efros, A. (2017b). Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings - 30th IEEE Conference on*

Computer Vision and Pattern Recognition, CVPR 2017, Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, pages 645–654, United States. Institute of Electrical and Electronics Engineers Inc. 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017 ; Conference date: 21-07-2017 Through 26-07-2017.

Zhang, R., Isola, P., and Efros, A. A. (2016). Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer.

Zhang, R., Isola, P., and Efros, A. A. (2017c). Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1058–1067.

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A. (2017a). Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhou, T. and Bilmes, J. (2018). Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. In *International Conference on Learning Representations*.

Zhou, T., Brown, M., Snavely, N., and Lowe, D. G. (2017b). Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858.

Zhou, T., Wang, S., and Bilmes, J. (2020). Curriculum learning by dynamic instance hardness. *Advances in Neural Information Processing Systems*, 33:8602–8613.

Zhu, X. J. (2005). Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Zoph, B. and Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.