Deep Learning Final Report

Yufei Quan yufeiq@kth.se
Guanyuan Li guanl@kth.se
Kaixin Jia kaixinj@kth.se
Yanggtao Chen yangtao@kth.se
KTH Royal Institute of Technology, Stockholm, Sweden

Abstract

The project is based on the Assignment3 and we expand the network, including adding more convolutional layers, downsampling operations, and regularization techniques to prevent overfitting and improve performance by ResNet. Key improvements include the incorporation of VGG-style convolutional blocks, the use of the AdamW optimizer, data augmentation methods such as horizontal flipping and translation, and batch normalization. The project also investigates the performance of ResNet training on CIFAR-10 and CIFAR-100. There are also explorations on Supervised Contrastive Learning, solutions of noisy labels, and comparison with Vision Transformer.

1 Introduction

Our code space website is: https://github.com/Jackson0422/DD2424-Final-Project-Group110

1.1 E

The aim of this project is to improve the performance of the network. We constructed the networks through PyTorch and applied various layers such as batch-normalization or maxpooling. Significantly, the influence of regularization and data augmentation such as cutout, label smoothing, and flipping are explored. Meanwhile, warm-up and cosine annealing are used as a scheduler for training. Above these are essential to avoid the effect of overfitting, which is important in future experiments.

1.2 A

In this section, we use ResNet18 as a baseline model on CIFAR-10 and CIFAR-100, and further explore supervised contrastive learning to enhance feature representation. To address the challenge of noisy labels commonly found in real-world datasets, we implement a CoTeaching strategy for improved robustness. Additionally, we experiment with Vision Transformer (ViT) on CIFAR-100 to compare its performance with ResNet18, while noting that the small image size may limit its effectiveness. These experiments are significant to further solutions in the future when problems such as noisy labels occur.

2 Related work

Much research in image classification has focused on architectural innovations and training techniques to boost accuracy and robustness. Classical models like VGGNet showed that deeper networks with small filters can perform well but suffer from computational inefficiency and depth limitations.[1] ResNet addressed this by introducing skip connections, enabling much deeper networks like ResNet18, now a standard baseline.[2]

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Recent advances include feature recalibration methods such as Squeeze-and-Excitation.[3] And also hybrid models like ConvNeXt, combining convolutional backbones with Transformer-like designs for better global and channel-wise feature modeling. [4] Supervised Contrastive Learning improves feature embedding for more discriminative representations, especially in low-data or similar-class scenarios.[5] Robust training methods like CoTeaching help tackle label noise by letting two networks teach each other with clean samples. [6]

The main contribution of this report:

1. Architectural Enhancements: Integrated Squeeze-and-Excitation (SE) blocks into the ResNet18 architecture and re-construct the ConvNeXt module. To accommodate the 32×32 resolution of CIFAR-10 and CIFAR-100 datasets, we modified the entry convolution layer to a 3×3 kernel instead of 7×7 kernel originally applied to 224 ×224 resolution dataset, ensuring effective feature extraction.
2. Supervised Contrastive Learning (SCL) Pretraining: Employed SCL as a pretraining strategy to enhance feature discriminability, particularly in scenarios with limited labeled data or high class similarity, thereby improving model generalization.
3. Robustness to Label Noise with CoTeaching: Assessed the impact of label noise on model performance and demonstrated the effectiveness of the CoTeaching algorithm in mitigating these effects, providing quantitative evidence of its benefits in enhancing model robustness.

3 Data

3.1 E

1. CIFAR-10 dataset from PyTorch, with 50,000 training and 10,000 test samples.
2. Data augmentation: random flipping, panning, and Cutout to enhance generalization.
3. Additional preprocessing: `ToTensor` and normalization.

3.2 A

1. CIFAR-10 and CIFAR-100 from PyTorch, with 49,500 training, 500 validation, and 10,000 test samples.
2. Augmentations: horizontal flip, translation, color jitter, rotation, and Cutout.
3. ImageNette dataset from FastAI, with 9,000 training and 469 validation samples; full test set included.
4. Preprocessing: resize, center crop, random crop, horizontal flip.
5. All datasets include `ToTensor` and normalization.

4 Method

4.1 E

4.1.1 Network Architecture with Regularization

The input images (3 channels, 32×32) are first processed by a convolutional patchify layer with 64 filters of size 2×2 and stride 2×2, reducing spatial dimensions to 16×16 and increasing channels to 64, followed by batch normalization and ReLU activation. The backbone uses a VGG-style block sequence $[64, 64, \mathrm{M}, 128, 128, \mathrm{M}, 256, 256, \mathrm{M}]$, where integers represent convolutional layers with 3×3 kernels, batch normalization, and ReLU, and `M` denotes max-pooling with stride 2. After flattening the $256 \times 2 \times 2$ feature map, a fully connected layer with 128 units and ReLU is applied, followed by dropout ($p = 0.5$) and a final linear layer with 10 outputs.

Initially, adding the VGG blocks caused gradient issues and test accuracy dropped to 0.1, adding batch normalization resolved this problem. Figures 4 and 5 illustrate the code structure, while Figures 6a and 6b present the results related to gradient issues, shown in the appendix.

The model is trained with the AdamW optimizer and L2 regularization. A cyclic learning rate scheduler adjusts the learning rate dynamically to aid convergence. Dropout, weight decay, and data augmentation (random flips and translations) improve generalization. Batch normalization after convolutional and fully connected layers stabilizes training. The CIFAR-10 dataset is augmented during training and normalized during evaluation.

4.1.2 Exploration

To reduce overconfidence, label smoothing softens target labels, while Cutout augmentation improves robustness by randomly masking patches in training images. Three learning rate schedulers—warm-up with cosine annealing, step decay, and cosine annealing with restarts—were tested to enhance training. Perform downsampling by applying the second convolution with stride 2 in each VGG block and removing the max-pooling layer. The fully connected layer after the last convolution was replaced with global average pooling to evaluate its impact on regularization and accuracy.

4.2 A

4.2.1 Architectural and Regularization Investigations

The SE-ResNet integrates Squeeze-and-Excitation (SE) blocks into the standard ResNet architecture to improve channel-wise feature representation. The SE block applies global average pooling followed by two fully connected layers and a sigmoid activation to recalibrate channel importance. This block is added after the second convolution in each residual block.

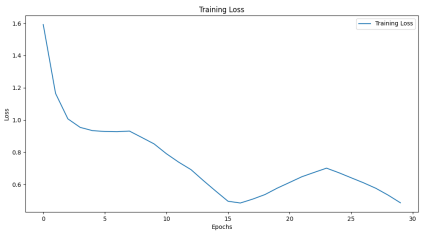
The ConvNeXt model was adapted for CIFAR-100 by replacing the input layer with a smaller 3×3 convolution to accommodate the 32×32 images. Down sampling is performed using strides convolutions, and the network employs ConvNeXt blocks consisting of depthwise convolutions, layer normalization, and GELU activations. The final features are normalized and passed through a linear classification head.

4.2.2 Training with a Different Loss

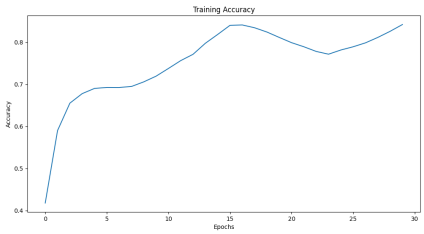
In this experiment, we explore supervised contrastive learning as an alternative to traditional cross-entropy loss for training a neural network. The primary goal of supervised contrastive learning is to learn a vector representation of images in such a way that images from the same class are closer in the feature space, while images from different classes are positioned farther apart. This is achieved through a loss function that encourages the model to bring together similar images while pushing apart dissimilar ones.
The key advantage of this approach lies in the use of a contrastive loss (denoted as L_{out}) to pre-train the network, as opposed to the more conventional cross-entropy loss L_in. The contrastive loss improves the model's ability to distinguish between different classes by focusing on the relationship between samples, thus enhancing the feature representations. However, to make computations feasible, compromises are often required, such as reducing batch sizes or applying data augmentation techniques, which play a critical role in the success of the learning process.
Once the model has been pre-trained using this feature representation, a small Multi-Layer Perceptron (MLP) can be trained on top of the learned features to perform image classification. After training the network, the performance is compared with a similar-sized network trained purely using cross-entropy loss. This comparison allows us to evaluate the effectiveness of supervised contrastive learning in producing more robust and discriminative feature representations for image classification tasks.

4.2.3 Make Training More Robust to Noisy Labels

In this experiment, we focus on addressing the problem of noisy labels in training data, which can significantly hinder the performance of the network. To make the training process more robust in the presence of noisy labels, we explore CoTeaching, a method designed to mitigate the negative effects of label noise.
CoTeaching works by training two neural networks simultaneously, each network independently learning from a subset of the data. These networks exchange their predictions, and each network only uses the most confident predictions from the other network to update its own parameters. The underlying idea is that the networks will tend to agree on the clean (correctly labeled) samples while ignoring the noisy labels, leading to more robust learning.



(a) Basic - Training Loss



(b) Basic - Training Accuracy

Figure 1: E - Basic

4.2.4 Exploration of ViT

The Vision Transformer (ViT) adapts the Transformer architecture, typically used in natural language processing, for image classification by treating an image as a sequence of smaller patches. Each patch is flattened into a vector and linearly embedded into a fixed-dimensional space. To preserve spatial information, positional encodings are added to the patch embeddings, allowing the model to understand the relative positions of the patches within the image.
ViT offers a self-attention mechanism for Transformers to capture long-range dependencies between patches, enabling it to model global relationships across the entire image. Patch representations are processed through several Transformer encoder layers and a classification token (CLS token) is used to produce the final prediction.

5 Experiments

5.1 E

5.1.1 Basic with Regularization

Using the AdamW optimizer with a weight decay of 1×10^{-3} for L2 regularization, the model was trained with a cyclic learning rate scheduler varying between 5×10^{-5} and 3×10^{-3} over 3900 steps. Dropout with a rate of 0.5, combined with data augmentation techniques of random horizontal flips (probability 0.5) and translations. Batch normalization was applied throughout the network. The final test accuracy achieved was 83.92%. Training loss and accuracy curves are presented in Figure 1.

5.1.2 Exploration

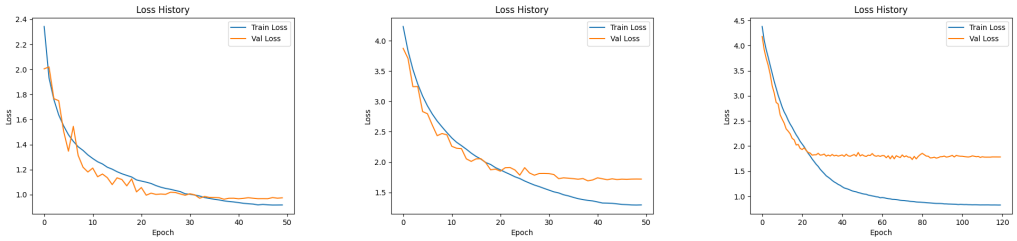
Label smoothing (factor 0.1) softens target labels to reduce overconfidence, while Cutout randomly masks an 8×8 patch in training images to improve model robustness. The test Result achieved is 83.10%. The train loss is shown in Figure 10.

Different learning rate schedulers were tested, including warm-up with cosine annealing, step decay (reducing learning rate by 0.1 every 10 epochs), and cosine annealing with restarts using PyTorch's `CosineAnnealingWarmRestarts` ($T_0 = 10$, $T_{mult} = 1$) to improve training dynamics. The test accuracies for the three learning rate schedulers are compared in Table 4. Among the learning rate schedulers tested, cosine annealing with restarts achieved the highest test accuracy. However, the original model using the cyclic learning rate scheduler still demonstrates the best performance. The train loss is shown in Figure 11.

Perform downsampling by applying the second convolution with stride 2 in each VGG block and delete max pooling layer. The test accuracy reaches 81.58% with 0.5455 test loss.

The fully connected layer after the last convolution was replaced by a global average pooling layer to evaluate effects on regularization and test accuracy. The test accuracy reaches 84.26%. The train loss is shown in Figure 12.

Combining label smoothing, Cutout, cosine annealing with restarts, and architectural changes like strided convolutions and global average pooling mentioned all above. The test accuracy is 79.45%,



(a) CIFAR-10 of SE (b) CIFAR-100 of SE (c) CIFAR-100 of ConvNeXt

Figure 2: Histories of Loss as Squeeze-and-Excite + ResNet18 and ConvNeXt

Architecture	CIFAR-10	CIFAR-100
Baseline ResNet18	0.9192	0.6889
SE + ResNet18	0.9203	0.7012
ConvNeXt	0.9159	0.7077

Table 1: Comparison of Test Accuracy Across Architectures on CIFAR-10 and CIFAR-100

suggesting that while individual modifications enhance performance, their combination may require further tuning for optimal results. The train loss is shown in Figure 13.

5.2 A

5.2.1 Baselines of CIFAR-10 and CIFAR-100

ResNet18 is separately trained and evaluated on the CIFAR-10 and CIFAR-100 datasets as baselines for the following experiments. And training loss and validating loss histories are shown in Figure 7, where the training processes have approached the boundary curve between overfitting. Notably, the test accuracies achieved on CIFAR-10 and CIFAR-100 are **91.92%** and **68.89%**, respectively.

5.2.2 Architectural Regularization Investigations

Integrating Squeeze-and-Excitation (SE) blocks into the ResNet18 architecture significantly enhances channel-wise feature representation and overall model performance. The resulting models achieved test accuracies of **92.03%** on CIFAR-10 and **70.12%** on CIFAR-100.

The adapted ConvNeXt model also demonstrated improved feature extraction capabilities and strong classification performance, achieving a test accuracy of **91.59%** on CIFAR-10 and **70.77%** on CIFAR-100. And training loss and validating loss histories are shown Figure 2.

5.2.3 Training with a Different Loss

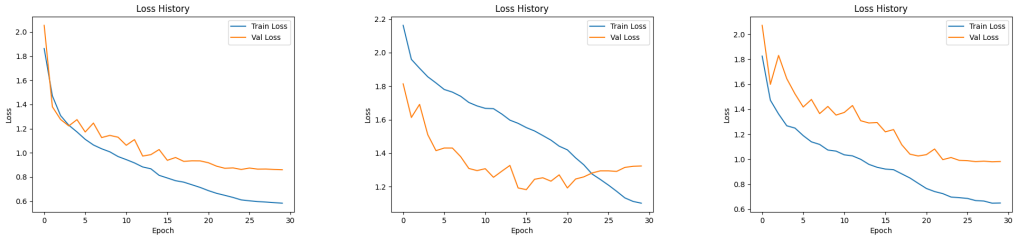
As the first stage of the training process is feature extraction, and we evaluate the performance of the strategy at the second stage - the performance of classification. The history of loss is shown in Figure 8. The rapid convergence speed at the beginning epochs infers that the first stage of feature extraction successfully concentrates the feature vectors so that the network can learn the pattern very quickly. And the accuracy result on the test set achieves 0.7120, which outperforms the baseline.

5.2.4 Make Training More Robust to Noisy Labels

In this section, three circumstances of noisy labels are compared. The details and results are shown in Table 2, which infers that the network performance will be significantly affected if label noise is involved and the CoTeaching Algorithm helps to decrease the influence of noise.
The loss histories of each case are shown in Figure 3, which shows the same trends that models training on clear data outperform other cases, and the use of CoTeaching algorithm can lower the influence of noise.

Metrics	No Noise	30% Noise	30% Noise + CoTeaching
Accuracy	0.8400	0.7192	0.7842
Loss	0.9287	1.2990	0.9818

Table 2: Results in Different Cases of Noisy Labels



(a) 0 Noisy Label (b) 30% Noisy Labels (c) 30% Noisy Labels + CoTeach

Figure 3: Histories of Loss in Various Noise Cases

5.2.5 Exploration of ViT

ViT is compared with the baseline to explore if the advanced approach will perform better. The comparison of results is shown in Table 3, and the loss history is shown in Figure 9.
The results show that ViT converges much slower during the training process but can still converge to a good point gradually. However, it is surprising to find that ResNet outperforms ViT. One possible reason is that ViT learns better on data whose size is much larger so that it can easily extract global and cross-information. And ResNet18 performs better because it always works well on common tasks.

6 Conclusion

This project systematically explored deep learning architectures and training strategies for image classification on CIFAR-10, CIFAR-100. Key findings include:

- Regularized CNN Training and further enhancements: VGG-style blocks with batch normalization, AdamW, cyclic LR, dropout,weight decay, and data augmentations (flip, translation) achieved 83.92% test accuracy on CIFAR-10. Further exploring label smoothing, Cutout, cosine annealing schedulers, and architectural tweaks pushed accuracy to 84.26%
- Significant Performance Improvement: Adding SE blocks to baseline ResNet18 improved accuracy, which reaches 70.12% on CIFAR-100. ConvNeXt extends depth network and with 120 running epochs and finally reaches 70.77% on CIFAR-100.
- Special cases such as Noisy Labels and pre-training with Supervised Contrastive Learning (SCL) are explored. It is interesting to find that using CoTeaching algorithm can effectively solve the problem of label noise and SCL can improve the performance of the classification.
- Advanced work includes exploring advanced architectures like Vision Transformers, applying stronger data augmentation methods, and comparing with normal CNNs.

	Accuracy	Loss
ResNet18	0.6889	0.6448
ViT	0.6276	0.8876

Table 3: Comparison of Metrics Between ViT and Baseline

7 Appendix

This appendix provides additional visualizations and supporting results throughout the report.

All visualizations support the findings discussed in the main text and offer deeper insight into training behavior and model performance across different settings.

Figure 4: E Structure

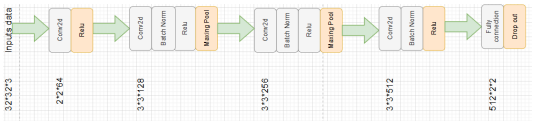
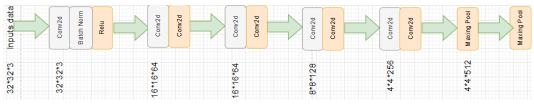
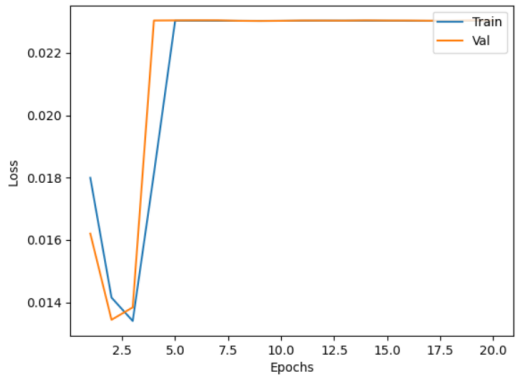
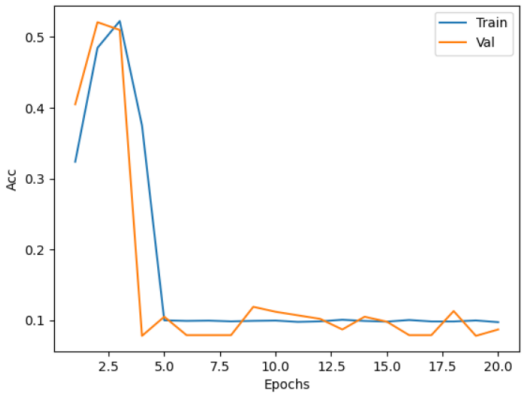


Figure 5: A Structure



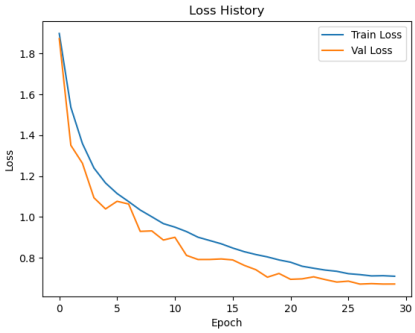


(a) Gradients exploring loss

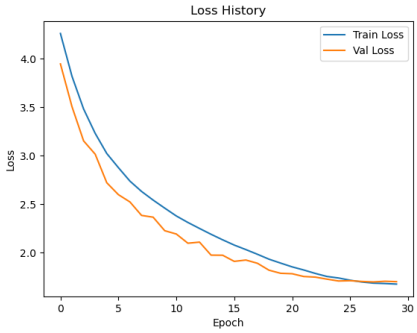


(b) Gradients exploring accuracy

Figure 6: Loss and Accuracy exploration side-by-side



(a) History of CIFAR-10



(b) History of CIFAR-100

Figure 7: Histories of Loss as Baselines

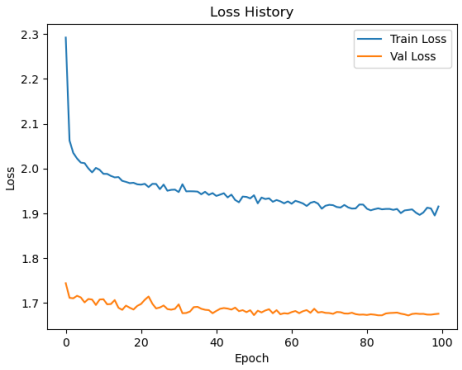


Figure 8: loss$_history - scl$

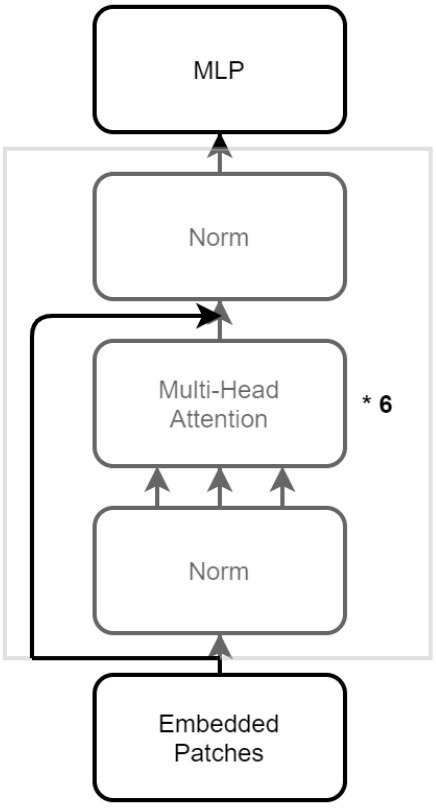


Figure 9: Network Structure of ViT

Learning Rate Model	Test accuracy
Warm-up + cosine annealing	0.7997
Step decay	0.3079
Cosine annealing with restarts	0.8162

Table 4: E - Learning rate exploration

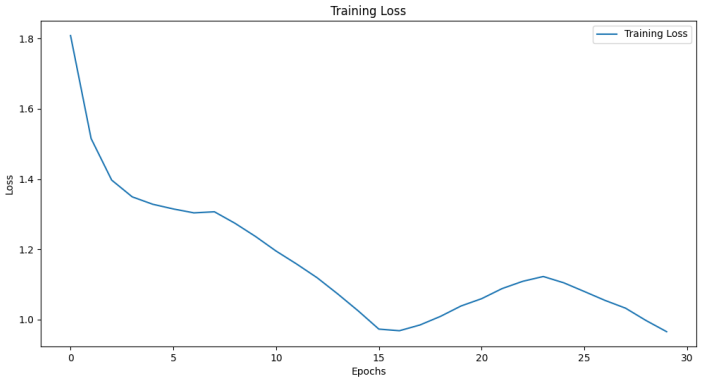
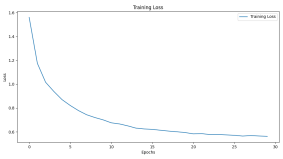
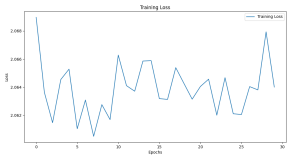


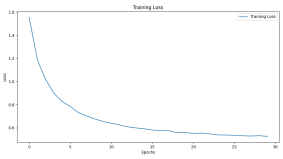
Figure 10: Histories of Loss in Label Smoothing + cutout



(a) Warm-up + Cosine Annealing

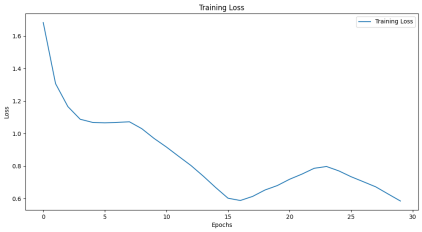


(b) Step Decay

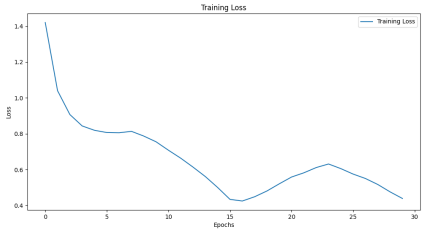


(c) Cosine Annealing with Restarts

Figure 11: Loss histories for three learning rate scheduling strategies



(a) Loss history for downsampling via strided convolution



(b) Loss history for replacing the fully connected layer

Figure 12: Loss histories for two architectural modifications

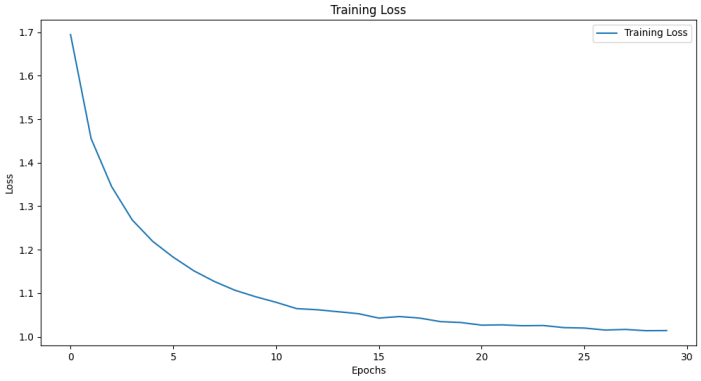


Figure 13: Histories of Loss in all combination of exploration

References

[1] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[2] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 770–778.

[3] Jie Hu et al. "Squeeze-and-Excitation Networks". In: *arXiv preprint arXiv:1709.01507* (2019). Version 4, May 16.

[4] Colin Wei, Yining Chen, and Tengyu Ma. "Mitigating Label Noise via Noise-Aware Learning". In: *arXiv preprint arXiv:2201.03545* (2022). URL: https://arxiv.org/abs/2201.03545.

[5] Prannay Khosla et al. "Supervised Contrastive Learning". In: *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*. 2020, pp. 18661–18673. URL: https://proceedings.neurips.cc/paper/2020/file/d89a66c7c80a29b1bdbab0f2a1a94af8-Paper.pdf.

[6] Jeremy Howard. *Imagenette: A smaller subset of 10 easily classified classes from Imagenet*. Mar. 2019. URL: https://github.com/fastai/imagenette.