Building and Optimizing a CNN for CIFAR-10 Classification from Scratch

Atheer Salim atheers@kth.se

Milad Farahani miladsf@kth.se

Pontus Amgren amgren@kth.se

Noel Tesfalidet Noelt@kth.se

Abstract

In this project, we create and train a Convolutional Neural Network (CNN) for image classification on CIFAR-10 and CIFAR-100 datasets using PyTorch. The initial model was based on the VGG architecture from the provided article, achieving a baseline accuracy of 73%. Various regularization techniques such as Dropout, L2 regularization, and data augmentation were applied to enhance performance, with data augmentation significantly boosting accuracy to 84%. Different optimizers and learning rate schedulers were explored, with Adam and AdamW showing improvements. Architectural upgrades were implemented using ResNet variants, achieving the highest accuracy of 95.59% on CIFAR-10. Symmetric Cross Entropy was used to handle noisy labels, maintaining reasonable accuracy even with high noise levels. The results demonstrate effective strategies for optimizing CNNs, contributing to the development of robust image classifiers.

1 Introduction

Image classification is an important problem in computer vision and has many types of use cases in a variety of applications. The project aims to contribute to solving this problem by training and testing different enhancements on CNNs on the Cifar datasets. The initial construction of the model followed the VGG baseline architecture which performed at the baseline accuracy of around 73%. We explored further different enhancements on this baseline to see the improvement. Three independent regularization techniques were tested on the baseline model, including, Dropout, L2 regularization, and, data augmentations such as horizontal flipping of images and translation shifts. The baseline was also used to test different learning rate schedulers, different optimizers such as Adam and AdamW, and batch normalization. Dropout improved performance to 82.67% after 100 epochs, L2 regularization maintained the accuracy around 73.34%. Data augmentation significantly boosted accuracy to 83.58%. Adam and AdamW optimizers improved accuracy to 82.13% and 82.89%, respectively. Warm-up with Cosine Annealing and Step Decay showed modest improvements, with Step Decay reaching the highest 74.10%. Cosine Annealing with restarts and Batch Normalization alone did not perform well, achieving 72.14% and 70.35%, respectively.

ResNet20, ResNet56, and ResNet110 showed improved accuracy with squeeze and excitation blocks and sophisticated data augmentation. Sophisticated data augmentation with ResNet110 achieved the highest accuracy at 95.50% on CIFAR-10 and 75.45% using SGD with momentum as the optimizer. Symmetric Cross Entropy (SL) was used to handle noisy labels and it showed that with the labels being contaminated between 20% and 60% it had better accuracy on VGG compared to using cross-entropy loss, these models trained with SL showed resilience to label noise.

2 Related Work

For the E-part of the project, the related work is the following, The CNN architecture implemented is VGG and the exploration on top of it comes from the tutorial [1]. Dropout as a regularization technique was used to reduce overfitting which is discussed in [2] and batch normalization[3] was also a regularization technique used but the main purpose is to help train deep networks and combat sensitive initialization. The other explorations explored further in the E-part have the following related work, two of the optimizers used were Adam[4] and AdamW[5] explained in more detail in each respective paper. Other learning rate schedulers was also explored such as cosine annealing[6] with and without restarts

For the A-part when it comes to **Architectural upgrades** the baseline network is ResNet which the paper[7] talks about in more detail. Furthermore, all of the suggested experiments on top of ResNet were implemented and here Squeeze-and-excitation networks[8] were used to understand how to implement the squeeze and excitation block. Also, Vision Transformer[9] was used to understand how images are provided as input to the transformer i.e. understand patchify and embed. Furthermore, to understand the sophisticated data augmentations such as CutMix[10] and MixUp[11] these papers were used to understand how these data augmentations work.

For the A-part when it comes to **Make training more robust to noisy labels** a new loss function was used known as symmetric cross entropy[12] to deal with the noisy labels.

3 Data

The data used for this project comes from the CIFAR-10 and CIFAR-100 datasets. These datasets are standard for computer vision and deep learning tasks. Both CIFAR-10 and CIFAR-100 consist of 60,000 32x32 labeled images of the 80 million tiny images dataset. [13] The images of the CIFAR-10 dataset are of 10 different classes, representing various categories such as cars, airplanes, dogs, cats, etc. Each class contains 6,000 images. 5000 images are for training and 1000 are for testing. For the CIFAR-100 dataset, there are 100 total classes divided into 20 superclasses with 600 images per class. There are 500 training images and 100 testing images for each class. [14]

Many models have utilized the datasets in focus. The top three best performing models for CIFAR-10 are *ViT-H/14* (99.5%), *DINOv2* (99.5%) and *μ2Net* (99.49%). *ViT-H* and *DINOv2* use transformer-based models while *μ2Net* focuses on existing pretraining methods and combinations of different techniques. [15] [16] [17] For CIFAR-100, the top three models are *EffNet-L2* (96.08%), *Swin-L + ML-Decoder* (95.1%) and *μ2Net* (94.95%). [18] In *EffNet-L2*, a procedure called "Sharpness-Aware Minimization" (SAM) was used which seeks parameters that lie in neighbourhoods having uniformly low loss. This results in a min-max optimization problem on which gradient descent can be used. [19] For *Swin-L + ML-Decoder*, an ML-Decoder was used, which is highly efficient and can scale to thousands of classes. The ML-Decoder predicts the existence of class labels via queries and enables better usage of spatial data compared to global average pooling. [20] *μ2Net* uses the same technique as for CIFAR-10. [17]

4 Methods

4.1 Grade E Part

The baseline CNN model is based on the model developed in [21], excluding the additional regularization techniques such as Dropout, Weight Decay, and Data Augmentation. One notable difference is that the tutorial uses the Keras library, while the model implemented in the project utilizes PyTorch. All hyper-parameters such as batch size, learning rate, etc. are the same in the project as in the tutorial. As part of the basic part of the project, additions were made to the baseline model. These extensions were: normalizing the data to have zero mean and standard deviation 1, replacing the SDG + momentum optimizer introduced in the tutorial with Adam and AdamW, trying different learning rate schedulers, and checking the optimal order to perform Dropout and Batch Norm and if they are complementary.

Normalization

In the tutorial, the input data is normalized by ensuring it is between 0 and 1. In addition, the input data can be experimented with and normalized to have zero mean and standard deviation 1. Two ways of doing this is possible. The first is using the approximate values of the mean and standard deviations of the dataset: μ: $(0.4914, 0.4822, 0.4465)$, σ: $(0.2023, 0.1994, 0.2010)$. The other way, which was used for the project, is to calculate the actual mean and standard deviation of the training set dynamically and then use the values to normalize the dataset. This method is preferred because it ensures that the normalize based on the actual statistics of the dataset.

Using Different Learning Rate Schedulers

Three learning rate schedulers were tested to improve the training on the Baseline CNN mentioned in the previous sections. The idea is to find a learning rate that optimizes the training and yields a higher final accuracy. Step decay, cosine with restart was fairly easy to implement because the learning rate schedulers were already in the PyTorch optimal library. These only required some research on the hyper parameter options. The Cosine annealing with warm up was not directly implemented in PyTorch and required some customization for the warm up phase. What was done is that normal cosine annealing was used from the optimal library but disregards that learning rate until the specified epoch is reached. During the warm up phase, the learning rate was calculated during the training and manually set.

Adam and AdamW

The tutorial uses SGD with momentum as an optimizer. Alternatives to this optimizer were explored by replacing it with Adam and AdamW optimizer. Without changing anything else, this allowed for comparing the optimizers' respective classification accuracy and time for convergence.

Order change of Dropout and Batch Normalization and Complementary check

The final model presented in [21] Incorporated batch normalization, dropout and data augmentation, it used batch normalization after each convolution and after the max pool used a dropout. We first explore if only using batch normalization or dropout in the same manner as above had an impact. Then tested further if switching batch normalization and dropout operation had an impact this was done by having dropout after the convolution and then batch normalization, no dropout was used after the max pool

4.2 Grade A Part

Architectural Upgrades

The first part of the architectural upgrades was to train a ResNet and this was done in two ways, the first way was using the same network and parameters applied in [7] on Cifar-10 (section 4.2) this included using the same data augmentations, SGD with momentum, cross-entropy loss, weight decay, HE initialization and learning rate started with eta 0.1 but at epoch 100 and 150 its was reduced by a factor 10 as according to the paper. We trained three ResNets namely ResNet20, ResNet56, and ResNet110 these differ in how deep the networks are and they were trained for 200 epochs in total. The second way was to comply with the assignments essentially same network and parameters were used but the AdamW optimizer was used instead and label smoothing was used in the cross-entropy loss with the value set 0.1. Note that for the improvements they were also trained using these two ways in both cases the original 50000 training images were split into 45000 training data and 5000 validation data.
In the paper [7] they used option A when one would scale down the number of channels from one layer to the next which expected a smaller number of channels, we instead chose option B since that was easier to implement.

Continuing all of the three suggested improvements were explored, adding the squeeze and excitation was done by using the baseline ResNet architecture but the ResNet block was changed with the squeeze-and-excitation version of the ResNet block this meant adding the squeeze and excitation operation before the residual connection as specified in [8] and here the reduction ratio 16

was chosen. Continuing further adding patchify and embed operation which corresponds to the first operation in Visual Transformers [9] was the first operation to run on the baseline ResNet and here the patch size was chosen to be 2x2.

Finally, the third improvement was to add more sophisticated data augmentations done on the baseline ResNet and here random erasing was added but also CutMix[10] and MixUp[11] were used, these two data augmentations work on a batch and not single image like the previous data augmentations and for each batch there was a random choice between which one of these would be used for that specific batch.
Finally, all of these improvements including baseline were evaluated on both Cifar-10 and Cifar-100

Make training more robust to noisy labels

Being able to train on noisy data is great, as it opens up the possibility to train on more data, instead of being restricted to datasets of only the highest quality. A model robust to noise is also important as even high-quality datasets may contain incorrectly labeled data [12]. One method of dealing with noisy labels, presented in [12] is called Symmetric Cross Entropy Learning (SL). The paper introduces a method for calculating loss by combining cross-entropy with reverse cross-entropy. We implemented this loss function and trained a VGG-based model with the same parameters as those given in the paper. The core model is an 8-layer CNN, consisting of 6 convolutional layers and 2 fully connected layers. For the optimizer, SGD with momentum 0.9, and weight decay 10^{-4} was used. The learning rate was initialized to 0.01, but reduced by a factor of 10 after 40 and once again after 80 epochs, when training for a total of 120 epochs.

Two different types of label contamination are mentioned in the paper. The first is symmetric (uniform), which entails flipping the labels of a given proportion of the training data to another label uniformly. The second type is asymmetric (class-dependent), where the contaminated labels are flipped to a class, that is dependent on the original class label. It is important to note that this contamination should only be present in the training set, leaving the validation and test datasets clean. Since symmetric noise was covered to a greater extent in the paper, we therefore chose to explore only this type of contamination. The level of contamination ranged from 0-80%, tested at 0%, 20%, 40%, 60%, and 80%. The dataset used was CIFAR-10. Although the paper does not mention any training/validation split, we chose to use 5000 samples (10%) of the training set for validation.

5 Experiments

This section covers the results and the discussion of the conducted experiments on image classification on the Cifar-10 and Cifar-100 datasets. The comparisons will mainly be on the baseline accuracy and model architecture.

Baseline Network Performance and Improvements

Below one can see Table 1 which shows that our results are similar to the results achieved which indicates that we have done things correctly even though we used PyTorch.

Experiment	Dataset	Test Data Accuracy	Epochs
Baseline	Cifar-10	73.42 %	100
Baseline + Dropout (constant dropout)	Cifar-10	82.67 %	100
Baseline + Weight Decay	Cifar-10	73.34 %	100
Baseline + Data Augmentation	Cifar-10	83.58 %	100
Combined Regularization	Cifar-10	89.28 %	400

Table 1: Performance of E-part without including the further explored

Further exploration

Table 2 below shows that changing the optimizer can have a large positive impact compared to the baseline since you only need to change one line of code. Also using dropout is very good since it's

easy to implement and also increases the accuracy by a hefty margin. It's also interesting to see that only using batch normalization the accuracy decreased and the same was when using cosine annealing with re-starts. The idea of the restart is to not get stuck in local minima and explore more effectively. However, the restarts did not improve the accuracy which could be because the restarts happened too often (every 10 epochs) causing an unstable initial training phase which lowered the total accuracy.

Experiment	Dataset	Test Data Accuracy	Epochs
Normalizing input data (0 mean, 1 std)	Cifar-10	73.66 %	100
Adam	Cifar-10	82.13 %	100
AdamW	Cifar-10	82.89 %	100
Warm-up + Cosine Annealing	Cifar-10	73.61 %	100
Step Decay	Cifar-10	74.10 %	100
Cosine Annealing with Re-starts	Cifar-10	72.14 %	100
Batch Normalization Only	Cifar-10	70.35 %	100
Dropout Only	Cifar-10	82.24 %	100
Changed Order Batch Normalization & Dropout	Cifar-10	81.50 %	200

Table 2: Performance of E-part including the further explored

Architectural Upgrades

Table 3 shows the performance of the architectural upgrades but here the network utilised the first way of training which is using the parameters defined in the ResNet paper[7], whilst table 4 shows the performance of the architectural upgrades but the network utilised the second way of training which was described by the assignment. The baseline indicates the network with the original ResNet network structure, SE indicates that Squeeze-and-excitation layers are used, PD indicates patchify and embed are used and finally, SDA indicates sophisticated data augmentations are used.

In table 3 comparing our result for the baseline ResNet to [7] we can see that our results are quite similar indicating that our implementation and training of the baseline ResNet was correct. Furthermore squeeze and excitation blocks helped increase the accuracy for both data across the various ResNet depths and patchify and embed were not helpful the accuracy decreased. Furthermore one can see that using sophisticated data augmentation bumps up the accuracy by a small margin.

In table 4 overall one can see when comparing to table 3 that the accuracy is less and this is mostly due to AdamW not being able to optimize as good as SGD with momentum with the settings that we have chosen, this can be directly observed in Appendix A. Furthermore in table 4 we can that using squeeze and excitation blocks have near similar performance as the baseline ResNet and patchify and embed helped but only for Cifar 100 with ResNet 20.

Make training more robust to noisy labels

Table 5 shows the test accuracy achieved when training data labels were contaminated to varying degrees, for both CE and SL loss. The performance recorded in the results of the paper [12] significantly outperforms our baseline model results. However, the paper does not mention any regularization methods, such as Dropout or Data Augmentation that could explain the high test accuracy. Therefore, we decided to use the model with Combined Regularization from Table 1 as our new baseline here. The test accuracies achieved by our model were comparable to the accuracies in the paper, with the exception of when noise was 80%. For this scenario, our CE-based model outperformed the one in the paper, while the SL model significantly underperformed. This might be explained by a regularization technique used in the paper but not in our solution.

Experiment	Dataset	Test Data Accuracy	Epochs
Baseline ResNet20	Cifar-10	91.47 %	200
Baseline ResNet56	Cifar-10	92.51 %	200
Baseline ResNet110	Cifar-10	93.45 %	200
Baseline ResNet20	Cifar-100	65.65 %	200
Baseline ResNet56	Cifar-100	68.61 %	200
Baseline ResNet110	Cifar-100	70.89 %	200
SE ResNet20	Cifar-10	91.64 %	200
SE ResNet56	Cifar-10	93.16 %	200
SE ResNet110	Cifar-10	93.76 %	200
SE ResNet20	Cifar-100	67.02 %	200
SE ResNet56	Cifar-100	70.51 %	200
SE ResNet110	Cifar-100	72.10 %	200
PE ResNet20	Cifar-10	86.99 %	200
PE ResNet56	Cifar-10	87.98 %	200
PE ResNet110	Cifar-10	89.59 %	200
PE ResNet20	Cifar-100	58.29 %	200
PE ResNet56	Cifar-100	59.49 %	200
PE ResNet110	Cifar-100	62.51 %	200
SDA ResNet20	Cifar-10	92.17 %	400
SDA ResNet56	Cifar-10	94.23 %	400
SDA ResNet110	Cifar-10	95.59 %	400
SDA ResNet20	Cifar-100	66.95 %	400
SDA ResNet56	Cifar-100	71.99 %	400
SDA ResNet110	Cifar-100	75.45 %	400

Table 3: Performance of architectural upgrades using the original parameters defined in ResNet paper

Experiment	Dataset	Test Data Accuracy	Epochs
Baseline ResNet20	Cifar-10	85.51 %	200
Baseline ResNet56	Cifar-10	90.86 %	200
Baseline ResNet110	Cifar-10	92.00 %	200
Baseline ResNet20	Cifar-100	19.89 %	200
Baseline ResNet56	Cifar-100	59.44 %	200
Baseline ResNet110	Cifar-100	64.44 %	200
SE ResNet20	Cifar-10	84.11 %	200
SE ResNet56	Cifar-10	88.60 %	200
SE ResNet110	Cifar-10	91.55 %	200
SE ResNet20	Cifar-100	28.62 %	200
SE ResNet56	Cifar-100	48.98 %	200
SE ResNet110	Cifar-100	59.54 %	200
PE ResNet20	Cifar-10	82.79 %	200
PE ResNet56	Cifar-10	85.25 %	200
PE ResNet110	Cifar-10	87.14 %	200
PE ResNet20	Cifar-100	30.15 %	200
PE ResNet56	Cifar-100	50.29 %	200
PE ResNet110	Cifar-100	51.84 %	200
SDA ResNet20	Cifar-10	78.23 %	400
SDA ResNet56	Cifar-10	88.86 %	400
SDA ResNet110	Cifar-10	89.79 %	400
SDA ResNet20	Cifar-100	20.47 %	400
SDA ResNet56	Cifar-100	21.59 %	400
SDA ResNet110	Cifar-100	61.95 %	400

Table 4: Performance of architectural upgrades using the second way of training including usage of AdamW optimizer and label smoothing

Loss Type	Noise Rate				
	0.0	0.2	0.4	0.6	0.8
CE	86.69%	80.71%	73.69%	63.70%	38.21%
SL	86.31%	85.07%	81.02%	72.20%	39.40%

Table 5: Performance of CE and SL loss on contaminated labels

6 Conclusion

We managed to train a CNN for image classification on CIFAR-10 and CIFAR-100 datasets using PyTorch. Some of the key findings Data augmentation significantly improved accuracy to 84%, while Dropout and L2 regularization provided smaller improvements. Adam and AdamW optimizers improved performance over SGD on the baseline model, and the Step Decay learning rate scheduler showed the highest performance over the other learning rate schedulers. ResNet110 with sophisticated data augmentation achieved the highest accuracy of 95.59% on CIFAR-10. Finally having labels contaminated between 20% to 60% whilst using symmetric cross-entropy has a noticeable accuracy improvement compared to just using cross-entropy.

A future extension to this project could be to train and implement Vision Transformers since in terms of accuracy they surpass CNN on Cifar-10 and Cifar-100, it would be interesting to explore those types of networks

References

[1] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Apr. 10, 2015. DOI: 10.48550/arXiv.1409.1556. arXiv: 1409.1556 [cs]. URL: http://arxiv.org/abs/1409.1556 (visited on 05/22/2024). preprint.

[2] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: ().

[3] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. Mar. 2, 2015. DOI: 10.48550/arXiv.1502.03167. arXiv: 1502.03167 [cs]. URL: http://arxiv.org/abs/1502.03167 (visited on 05/22/2024). preprint.

[4] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. DOI: 10.48550/arXiv.1412.6980. arXiv: 1412.6980 [cs]. URL: http://arxiv.org/abs/1412.6980 (visited on 05/22/2024). preprint.

[5] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. Jan. 4, 2019. DOI: 10.48550/arXiv.1711.05101. arXiv: 1711.05101 [cs, math]. URL: http://arxiv.org/abs/1711.05101 (visited on 05/22/2024). preprint.

[6] Ilya Loshchilov and Frank Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts*. May 3, 2017. DOI: 10.48550/arXiv.1608.03983. arXiv: 1608.03983 [cs, math]. URL: http://arxiv.org/abs/1608.03983 (visited on 05/22/2024). preprint.

[7] Kaiming He et al. *Deep Residual Learning for Image Recognition*. Dec. 10, 2015. DOI: 10.48550/arXiv.1512.03385. arXiv: 1512.03385 [cs]. URL: http://arxiv.org/abs/1512.03385 (visited on 05/22/2024). preprint.

[8] Jie Hu et al. *Squeeze-and-Excitation Networks*. May 16, 2019. DOI: 10.48550/arXiv.1709.01507. arXiv: 1709.01507 [cs]. URL: http://arxiv.org/abs/1709.01507 (visited on 05/22/2024). preprint.

[9] Alexey Dosovitskiy et al. *An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale*. June 3, 2021. DOI: 10.48550/arXiv.2010.11929. arXiv: 2010.11929 [cs]. URL: http://arxiv.org/abs/2010.11929 (visited on 05/22/2024). preprint.

[10] Sangdoo Yun et al. *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features*. Aug. 7, 2019. DOI: 10.48550/arXiv.1905.04899. arXiv: 1905.04899 [cs]. URL: http://arxiv.org/abs/1905.04899 (visited on 05/22/2024). preprint.

[11] Hongyi Zhang et al. *Mixup: Beyond Empirical Risk Minimization*. Apr. 27, 2018. DOI: 10.48550/arXiv.1710.09412. arXiv: 1710.09412 [cs, stat]. URL: http://arxiv.org/abs/1710.09412 (visited on 05/22/2024). preprint.

[12] Yisen Wang et al. "Symmetric cross entropy for robust learning with noisy labels". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 322–330.

[13] Antonio Torralba, Rob Fergus, and Bill Freeman. *80 Million Tiny Images*. https://groups.csail.mit.edu/vision/TinyImages/. June 2020. (Visited on 05/22/2024).

[14] Alex Krizhevsky. *CIFAR-10 and CIFAR-100 Datasets*. https://www.cs.toronto.edu/~kriz/cifar.html. Sept. 2017. (Visited on 05/22/2024).

[15] Alexey Dosovitskiy et al. *Papers with Code - An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale*. https://paperswithcode.com/paper/an-image-is-worth-16x16-words-transformers-1. (Visited on 05/22/2024).

[16] Maxime Oquab et al. *DINOv2: Learning Robust Visual Features without Supervision*. Feb. 2024. DOI: 10.48550/arXiv.2304.07193. arXiv: 2304.07193 [cs]. (Visited on 05/22/2024).

[17] Andrea Gesmundo and Jeff Dean. *Papers with Code - An Evolutionary Approach to Dynamic Introduction of Tasks in Large-scale Multitask Learning Systems*. https://paperswithcode.com/paper/an-evolutionary-approach-to-dynamic. (Visited on 05/22/2024).

[18] *Papers with Code - CIFAR-100 Benchmark (Image Classification)*. https://paperswithcode.com/sota/image-classification-on-cifar-100?tag_filter=0. (Visited on 05/22/2024).

[19] Pierre Foret et al. *Papers with Code - Sharpness-Aware Minimization for Efficiently Improving Generalization*. https://paperswithcode.com/paper/sharpness-aware-minimization-for-efficiently-1. (Visited on 05/22/2024).

[20] Tal Ridnik et al. *Papers with Code - ML-Decoder: Scalable and Versatile Classification Head*. https://paperswithcode.com/paper/ml-decoder-scalable-and-versatile. (Visited on 05/22/2024).

[21] Jason Brownlee. "How to develop a cnn from scratch for cifar-10 photo classification". In: *Machine Learning Mastery* (2019).

7 Appendix A

In fig 1 and fig 2 we can clearly observe that AdamW was not able to achieve 100 % accuracy on the training data and the accuracy on the validation data was roughly 60 % whilst when using SGD with momentum the validation accuracy was roughly 70 %. Furthermore SGD with momentum achieved lower loss in general when comparing to using AdamW.

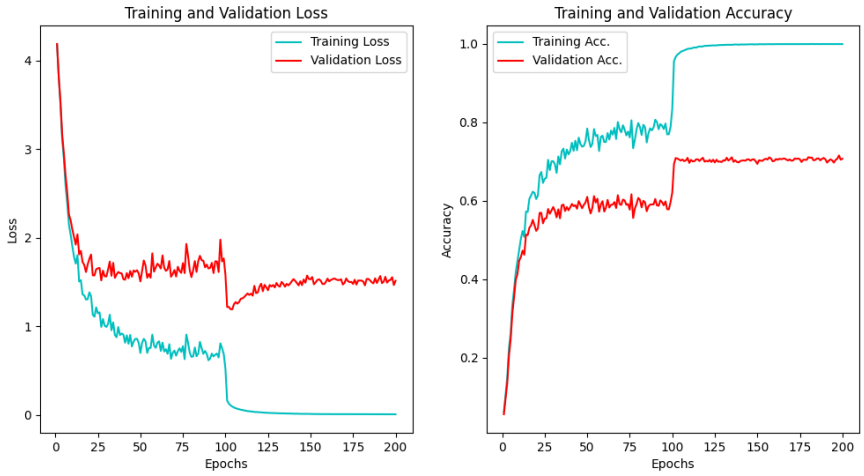

Figure 1: Loss & accuracy plot of Baseline ResNet110 trained using SGD with momentum

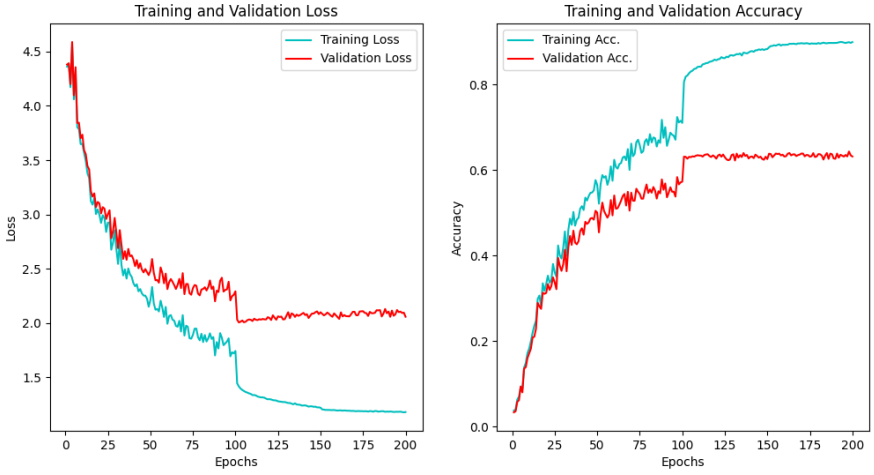

Figure 2: Loss & accuracy plot of Baseline ResNet110 trained using AdamW