Student Number: 198680

1. Introduction

The CIFAR-10 dataset presents a multi-class classification challenge for machine learning (ML) researchers [6]. The dataset includes 60,000 32x32 full-colour images, which are mapped to 10 non-overlapping classes such as frog or airplane. Furthermore, a training-test split divides the labelled images into 50,000:10,000 sets. This facilitates researchers to train models on the training set and to report unbiased performance on the test set. Performance is reported in accuracy on this dataset, which is often not best practice for machine learning tasks. However, this is a multi-class classification problem with verifiably evenly-distributed classes (6000 images per class), therefore accuracy is a representative metric of model performance in this scenario.

An investigation into how deep convolutional neural networks (CNN) performed on this challenge was undertaken. These networks have been the gold-standard in image-based machine learning (ML) tasks [7, 3] over the past decade. Different architectures were explored and optimised for the CIFAR-10 task. Specifically, the Asynchronous Successive Halving Algorithm (ASHA) [8] was used to search for optimal network parameters, this algorithm improves resource allocation and employs aggressive early-stopping on unfavourable trials. A wide range of hyper-parameters were explored, alongside data augmentation techniques. During optimisation, model performance was validated against a held-out validation set.

Two architectural approaches were explored, these are referred to as the classic CNN approach and the ResNet approach [3]. The classic CNN approach explores a neural network with two convolutional layers and three densely connected layers to classify the CIFAR-10 classes. The ResNet approach explores a much deeper network with either 18 or 34 layers, it makes use of skip connections to perform well at such depth. 30 hyper-parameter trials for each approach were conducted, this was then followed up by a further 10 ResNet trials that included a learning rate scheduler. For readers convenience the best performing trials in each approach are reported in Table 1.

Model Name	CIFAR-10 Accuracy (%)
Classic CNN	64.4
ResNet	87.8
ResNet + LR Scheduler	92.3

Table 1: Results of each investigated model on the CIFAR-10 test set, reported in accuracy.

The report consists of four subsequent sections. A methods section detailing the approaches taken in this assignment (2). The results of these approaches are then reported, this includes the details of hyper-parameter optimisation (3). These results are then interpreted in the discussion section of the report, along with providing avenues for further work (4). Lastly, a conclusion of the report and it's findings are detailed in the final section (5).

2. Methods

This section details the methodological approach adopted in applying CNN models to the CIFAR-10 dataset. It is divided into four subsections: an implementation overview (2.1), data preprocessing (2.2), model architectures (2.3) and hyper-parameter tuning (2.4). Each subsection explores a different aspect in relation to the task methodology.

2.1. Implementation Overview

CNN models in this investigation were implemented using *Pytorch*, specifically, *Pytorch Lightning* [9] was used to reduce boilerplate code and generally improve the process of training many models. The CIFAR-10 dataset was downloaded directly from Pytorch; their official guide was especially useful in this regard [10]. The training set was split into a training-validation split of 90:10% respectively. Furthermore, the hyper-parameter tuning library *Ray Tune* [14] was used to optimise model parameters. This library allowed the Asynchronous Successive Halving Algorithm (ASHA) [8] to be implemented, this searched for network parameters to maximise validation set performance.

2.2. Data Preprocessing and Augmentation

The CIFAR-10 Dataset was downloaded directly from Pytorch as a Pytorch dataset. This lent itself to using Pytorch Transforms functionality for image processing. Image normalisation was applied to both training and test images, meaning that the dataset had a mean of 0 and a standard deviation of 1 for each channel, this centering of a dataset generally improves model performance which is why it was applied here over all sets [12]. However, the mean and standard deviation values were calculated over the training set only. The mean was found to be 0.491, 0.482, 0.447 (3.d.p) and the standard deviation 0.247, 0.243, 0.261 (3.d.p) for each channel (RGB) respectively.

Data augmentation techniques were also applied to training images, this is considered a regularisation technique that improves a models ability to generalise to new images (e.g. the unseen test set) [12, 11]. Several techniques were explored such as: random cropping, randomly flipping im-

ages horizontally and randomly rotating images. These augmentations are applied at retrieval time, meaning that the model must learn to adapt to slight variations in images during training, therefore, making it more robust. However, these augmentation techniques were considered a hyper-parameter, and turning them on or off to see how they influenced validation performance was explored in the hyper-parameter investigation.

2.3. Model Architectures

This subsection explores the different CNN architectures that were implemented. In particular, two different categories of CNNs were explored: a classic CNN architecture and a ResNet CNN architecture. This section begins with an overview of the loss function used for both approaches, followed by a detailed examination of each approach.

2.3.1 Loss Function

This problem is a multi-class classification problem with 10 classes, meaning each image passed to the model can belong exclusively to one class. This meant that the Categorical Cross Entropy (CCE) Loss Function was appropriate for this task [12]. This loss function is described as a soft-max over the model output, followed by the cross entropy loss. Where:

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{1}$$

$$\text{CCE Loss} = -\sum_{\forall x} y(x) \cdot log(\hat{y}(x)) \tag{2}$$

2.3.2 Classic CNN Approach

A CNN was implemented with two convolutional layers, each coupled with a max pooling layer and followed by three dense layers. Additionally, the ReLU activation function was applied over each of the dense layers. The final dense layer had 10 outputs, one for each class. This network is similar (but shallower) to the AlexNet [7] approach proposed in 2012, this architecture massively popularised the deep CNN approach. This architecture proposed multiple convolutional layers followed by three dense layers and conducted training with a GPU, the same approach is used here only with fewer convolutional layers.

The best hyper-parameters were investigated for this approach, such as: optimisation choice (Stochastic Gradient Descent - SGD - or Adam [5]), batch normalisation [4], drop out layers [13], weight decay, dense layer weight initialisation strategies [2], and data augmentation. However, some hyper-parameters were fixed such as the batch size of 64, to minimise training time, and the choice of non-linearity function (ReLU) was also kept fixed. Furthermore, the main shape of the models architecture was kept fixed (e.g. number of layers and layer width), this was mainly for simplicity as it allowed deeper investigation into numerical hyper-parameters whilst tuning. Table 2 details the investigated parameters and their possible ranges. Models were run for 30 epochs, however, ASHA employs aggressive early stopping and so many trials were run for far fewer epochs.

Parameter	Possible Values
Learning Rate	1e-4 to 1e-2
Optimiser	Adam or SGD
Weight Decay	0.001 to 0.01
Drop out	0.1, 0.3, 0.5
Batch Norm	True or False
Xavier Weight Init	True or False
Random Weight Seed	7, 42, 77
Data Augmentation	True or False

Table 2: Investigated hyper-parameters for the Classic CNN approach. Fixed parameters are not listed.

2.3.3 ResNet CNN Approach

A CNN inspired by the ResNet architecture [3] was implemented. The ResNet authors note neural networks implementations often worsen as they increase in depth, however, theoretically networks should only ever improve with depth as they have more parameters to express the problem function. At the very least they need to learn the identity function of previous layers to perform equally as well as their shallower counter-parts. The authors postulate that the problem with these deeper networks is their inability to accurately learn this identity function. They propose a solution of 'skip connections' where the output from a previous layer is summated with a deeper layer in the network, essentially providing the identity function for free and allowing the model to only learn useful differences in output.

In this report an 18 and 34 layer ResNet are explored, this consists almost entirely of convolutional layers with one dense layer at the end for classification. Furthermore, batch normalisation [4] was applied several times throughout the network as recommended in the ResNet paper. Hyper-parameters were tuned in a similar fashion to the previous approach (with ASHA). However, due to the depth of these models and their reliance on some hyper-parameters a slightly restricted set was investigated. Table 3 details which parameters were investigated. Again, each model was trained for a maximum of 30 epochs with a batch size of 64.

Parameter	Possible Values
Learning Rate	1e-4 to 1e-2
Optimiser	Adam or SGD
Weight Decay	0.001 to 0.01
Drop out	0.1, 0.3, 0.5
Random Weight Seed	7, 42, 77
Data Augmentation	True or False
ResNet Depth	18 or 34

Table 3: Investigated hyper-parameters for the ResNet approach. Fixed parameters are not listed.

2.4. Hyper-parameter Tuning

A trial of 30 models for each approach was launched to find optimal hyper-parameters. Parameters were sampled from the possible value ranges detailed in Table 2 and Table 3. The ASHA algorithm) was used to improve resource allocation and employ early stopping over the trials. The models were assessed against a held-out validation set, where accuracy was used as the performance metric. The accuracy and loss were stored for both the training and validation sets over all trials.

3. Results

This section details the findings of this report. Specifically, it displays the findings of the hyper-parameter tuning for each of the two detailed approaches. It then reports the parameters that achieved best performance on the held-out validation set. Finally, the best performing models from each approach are assessed against the CIFAR-10 test set and the accuracy score for each model is reported.

3.1. Hyper-parameter tuning results

Please refer to Figure 1 and Figure 2 in the appendix for the full results of hyper-parameter tuning across the 30 trials for each approach.

3.1.1 Best Hyper-parameters

The highest performing model in the classic CNN hyper-parameter trials achieved 67.1% validation accuracy after 27 epochs. It used the SGD optimiser with a $\sim$ 0.0039 learning rate, 0.9 momentum value and a $\sim$ 0.006 weight decay value. It used drop out layers to randomly ignore 10% of nodes at each dense layer. Interestingly, it did not perform better with batch normalisation, Xavier weight initialisation or data augmentation.

The highest performing model in the ResNet approach hyper-parameter trials achieved 87.6% validation accuracy after 28 epochs. It made use of the deep 34 layer ResNet architecture. Along with the Adam optimiser and a $\sim$ 0.0001 learning rate and a $\sim$ 0.0024 weight decay value. A single drop out layer (30%) was applied before the final dense layer. Additionally, the training data was augmented with random cropping, horizontal flipping and random rotation.

3.1.2 Further-Training: Learning Rate Schedule

After 30 trials on each architecture was conducted, a validation performance seemed to be plateauing. It was hypothesised that the learning rate might be too large to descend any further in the loss landscape. Therefore, one further study of 10 trials on the more promising 34 layer ResNet architecture was conducted with the addition of a learning rate scheduler. This decreased the learning rate during training to improve convergence, this technique was used in the original ResNet paper [11, 3]. The models in these trials were also trained longer (50 epochs). Additionally, a smaller subset of hyper-parameters was explored: learning rate, weight decay, data augmentation, drop out and optimiser choice. The results of these trials can be seen in Figure 3 in the appendix. The addition of a learning rate scheduler consistently improved model performance. With the best model achieving a validation accuracy of 92.7%. Optimal hyper-parameters for this method were the SGD optimiser with an initial 0.002 lr and weight decay of 0.005. The best model also applied data augmentation to the training images, but it did not apply drop out to the last linear layer. This concurs with similar findings to the ResNet authors [3].

Class	CNN	ResNet	ResNet LRS
'bird'	52.4	85.1	88.1
'car'	65.2	92.2	96.4
'cat'	41.5	79.1	84.2
'deer'	62.4	87.9	92.9
'dog'	60.2	74.8	87.9
'frog'	80.1	92.8	94.5
'horse'	63.8	93.4	95.4
'plane'	77.0	83.8	93
'ship'	72.0	94.5	95.6
'truck'	69.2	94.1	95.7
Avg. Accuracy	**64.4**	**87.8**	**92.37**
Std. Deviation	**10.8**	**6.5**	**6.0**

Table 4: Results of each investigated model on the CIFAR-10 test set, reported in accuracy per class. The ResNet with a learning rate scheduler is being referred to as ResNet LRS.

3.2. Model Test Results

The highest performing model from each trial was saved, these were then tested on the CIFAR-10 test set. This allows an unbiased assessment of performance as the model

hasn't interacted with this data at all. The final performance on the test set is displayed in Table 4 above for both models on a by-class basis. The ResNet approach significantly out performs the classic CNN and has less variance between classes. The learning-rate variant also substantially improved this performance.

4. Discussion and Further Work

The classic CNN approach performed moderately well on this task and well above a random baseline. However, the ResNet approach significantly outperformed the other model with up to 92.3% accuracy (1.d.p). This is an impressive achievement as human performance is set around $\sim$ 94%, so this gap is certainly comparable. With a larger investigation of models, it would seem probable to close this performance gap. Unlike the classic CNN, the ResNet approach managed to consistently achieve scores of 90+% on the training data whilst not massively over-fitting the validation data, showing that strong features were being learnt over the data. This was especially true when a learning rate schedule was applied.

The ResNet approach also had less variance in its by-class performance compared to the classic CNN: 36.9 to 117.7 (1.d.p) respectively. This could mean that the classic CNN model is over-predicting certain classes at the detriment of others. For example, the 'cat' class only had a 41.5% accuracy compared to the 'frog' class with 80.1% accuracy. This is a worrying inter-class difference in performance, leading to question the robustness of the model. In general, both models fared worse on the animal classes compared to the vehicle classes. For the animal classes the classic CNN had an average accuracy of 60.0% (1.d.p) and the ResNet LRS 90.5% (1.d.p). Whereas, in the vehicle classes they scored 70.9 (1.d.p) and 95.2 (1.d.p) respectively. It could be hypothesised that these classes have more over-lapping features compared to the vehicle classes. For example, the deer, horse, cat and dog classes clearly share significant overlap in visual features since they are all four-legged land mammals.

The use of different hyper-parameters clearly had significant impact on model performance. It was surprising to see that many regularisation strategies such as batch norm and data augmentation did not improve the classical CNN's performance. However, it is also important to view these results through the lens of the models performance on the training set. This model didn't seem to learn strong features that could even predict the training data, meaning that model regularisation could be less effective. This could have been explored further, to first try and get good performance over the training data. Comparatively, the ResNet model often over-fit on the training data at the detriment to the validation set (as seen in Figure 2 and Figure 3). Therefore, it could be hypothesised regularisation techniques such as data augmentation would be more key to improving performance in this case. Furthermore, the learning rate schedule undoubtedly improved the overall performance demonstrating that learning rate is a significant factor in model training.

Since the deepest network (34 layers) achieved the best performance, naturally, a next step would be to further increase the network depth. In the original ResNet paper these deeper architectures yield better performance. Furthermore, there were several important hyper-parameters that were not investigated. Namely, many data augmentation strategies such as mixup or cutout [12] were not explored, these can add regularisation to model training and therefore improve the models performance on unseen data. It was also not totally apparent at which point models were over-fitting the training data, conducting fewer trials with a higher maximum epoch may provided more interpretable results. Additionally, a more insightful analysis may have created a classic CNN with varying depth, it is not entirely appropriate to compare the classic CNN against the deeper ResNet when they differ so vastly in architecture.

One particular avenue for future work that seems promising is the recently released normalisation-free networks (NFNets) [1], these recently released models eschew batch normalisation for a different system. They are faster to train and have significant performance gains over their batch-norm variants. It has been shown that a NF ResNet variant outperforms the standard counterpart, showing that this is a promising avenue for future work.

5. Conclusion

The CIFAR-10 dataset offers an insightful challenge for computer vision researchers. In this report a comparison between a deep ResNet CNN and a shallower CNN were investigated. It was shown that the ResNet counterpart significantly outperformed the shallower variant. It was shown that regularisation strategies seemed to yield better performance for this deeper network, yet, not for the shallower classic CNN. Although, this result seemed inconclusive as the classic CNN wasn't over-fitting to the training data, therefore further research is required. By applying a learning rate scheduler to the deeper ResNet architecture significant performance gains were observed. Through this strategy a model comparable to human performance was created for this dataset.

References

[1] A. Brock, S. De, S. L. Smith, and K. Simonyan. High-performance large-scale image recognition without normalization. 2 2021. 4

[2] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. 2

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. volume 2016-December, pages 770–778. IEEE Computer Society, 12 2016. 1, 2, 3

[4] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. volume 1, pages 448–456. International Machine Learning Society (IMLS), 2 2015. 2

[5] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. International Conference on Learning Representations, ICLR, 12 2015. 2

[6] A. Krizhevsky, V. Nair, and G. Hinton. Learning multiple layers of features from tiny images, 2009. 1

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks, 2012. 1, 2

[8] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, and A. Talwalkar. A system for massively parallel hyperparameter tuning. 10 2018. 1

[9] P. Lightning. Pytorch lightning documentation — pytorch lightning 1.3.0rc2 documentation. 1

[10] Pytorch. Training a classifier — official pytorch cifar-10 tutorial. 1

[11] V. Sharmanska. Week 5: Convolutional neural networks i: Neural networks [20/21], 2021. 1, 3

[12] V. Sharmanska. Week 6: Convolutional neural networks ii: Neural networks [20/21], 2021. 1, 2, 4

[13] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting, 2014. 2

[14] R. Tune. Trial schedulers (tune.schedulers) asha — ray v2.0.0.dev0. 1

6. Appendix

Continued in pages below.

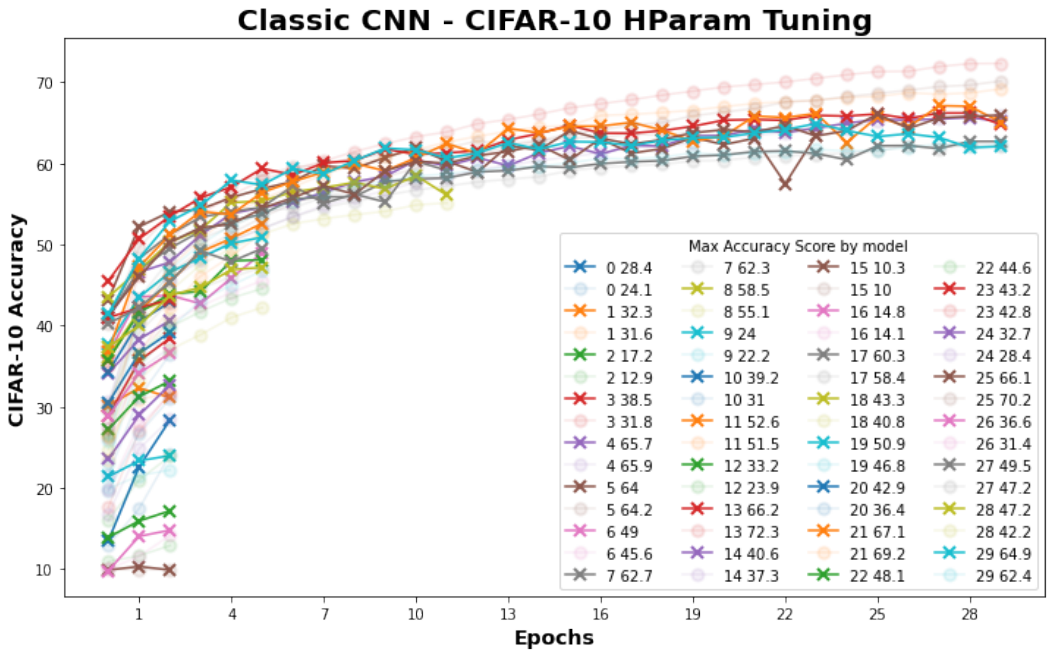

Figure 1: This figures displays N=30 trials using the Classic CNN architecture described under Methods 2.3.2. The bold lines represent each models performance over the held-out validation set over each epoch it was run. Adjacently to each line, an opaque line of the same colour displays the respective models performance on the training set. The legend displays maximum performance values for both validation and training sets per model over all epochs.

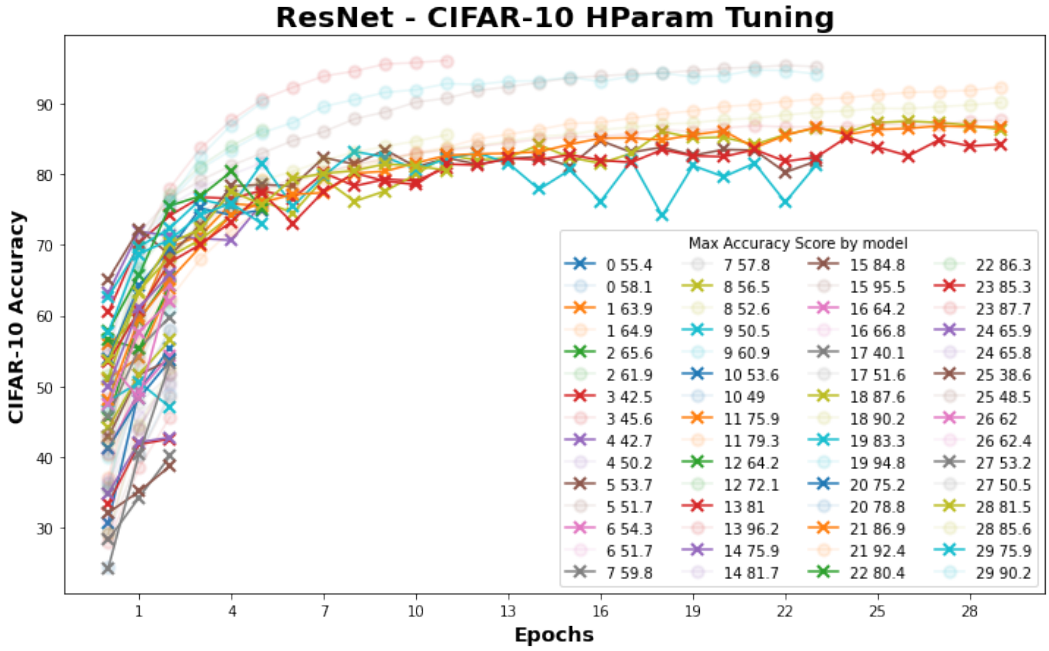

Figure 2: This figures displays N=30 trials using the Classic CNN architecture described under Methods 2.3.2. The bold lines represent each models performance over the held-out validation set over each epoch it was run. Adjacently to each line, an opaque line of the same colour displays the respective models performance on the training set. The legend displays maximum performance values for both validation and training sets per model over all epochs.

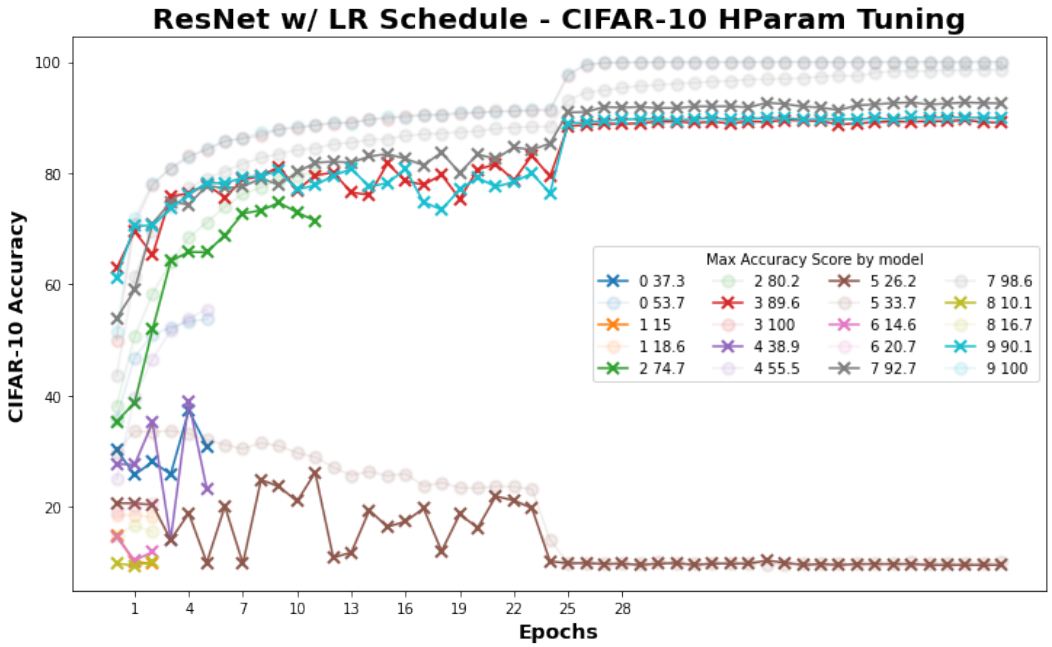

Figure 3: This figures displays N=30 trials using the Classic CNN architecture described under Methods 2.3.2. The bold lines represent each models performance over the held-out validation set over each epoch it was run. Adjacently to each line, an opaque line of the same colour displays the respective models performance on the training set. The legend displays maximum performance values for both validation and training sets per model over all epochs.