AI6103 Individual Assignment
Effects of Hyper-parameters on Optimization and Regularization of Deep Neural Networks

Nguyen Giang Son (G2404606D)
College of Computing and Data Science
Nanyang Technological University Singapore
c240066@e.ntu.edu.sg

Abstract

A key challenge in deep learning is balancing optimization and regularization. Optimization is determined by the hyper-parameters initial learning rate and its schedule, while regularization is affected by weight decay and data augmentation. In this report, we explore how these hyper-parameters impact deep neural networks using the MobileNet architecture and the CIFAR-100 dataset. We found that setting a good learning rate plays an important role in model training, using a Cosine Annealing schedule and Weight Decay improves model performance. Additional experiments reveal that using ReLU activation function helps avoid the vanishing gradient problem and leads to smooth loss reduction.

Introduction

Deep neural networks (DNNs) have achieved outstanding performance in multiple applications, butt optimization and regularization techniques remain a great challenge for deep learning practitioners. Optimization, controlled by hyper-parameters such as the initial learning rate and its schedule, enables efficient gradient-based updates, while regularization methods like weight decay and data augmentation help prevent overfitting and enhance generalization (Goodfellow, Bengio, and Courville 2016). However, achieving the right balance between these factors remains a challenge, especially for lightweight architectures designed for resource-limited environments (Howard et al. 2017).

This study investigates how these hyper-parameters influence training stability and model performance, focusing on the MobileNet architecture (Howard et al. 2017) and the CIFAR-100 dataset (Krizhevsky 2009). MobileNet, designed for mobile and embedded vision applications, employs depthwise separable convolutions to drastically reduce computational costs. However, its efficiency also makes it sensitive to suboptimal hyper-parameter configurations, necessitating a systematic analysis of optimization and regularization trade-offs.

Prior work has emphasized the critical role of learning rate scheduling in DNN convergence. For instance, (Loshchilov and Hutter 2017) demonstrated that cosine annealing—a strategy that smoothly reduces the learning rate—outperforms traditional step-based decay. Similarly, weight decay, a regularization technique that penalizes large weights, has been shown to stabilize training by implicitly limiting model complexity (Krogh and Hertz 1991).

Additionally, we examine architectural decisions, particularly activation functions, which play a crucial role in gradient flow. Sigmoid activations tend to cause vanishing gradients due to saturation effects (LeCun et al. 1998), whereas Rectified Linear Units (ReLUs) address this problem by maintaining linear gradients in their active region (Nair and Hinton 2010).

Background

MobileNet

MobileNet (Howard et al. 2017) is a family of lightweight convolutional neural networks designed for efficient deep learning on mobile and edge devices. It achieves comparable performance to previous models such as VGG-16 (Simonyan and Zisserman 2015) or ResNet-50 (He et al. 2015); while significantly reducing compute and the number of parameters due to key architectural innovations.

Depth-wise separable convolution and Point-wise convolution: Networks prior to MobileNet typically used standard convolutional layers, which compute filters across all input channels simultaneously.

Meanwhile, MobileNet splits convolution into two steps:

1. Depthwise Convolution: Applies a single $K \times K$ filter per input channel (no cross-channel interaction).
2. Pointwise Convolution: Uses 1×1 convolutions to combine outputs across channels.

Figure 1 compares the architectures of MobileNet and standard convolution. Using this architecture reduces the parameters count as well as floating-point operations (FLOPs) used by the network. The calculations are summarized in Table 1.

CIFAR-100

CIFAR-100 (Krizhevsky 2009) is a dataset comprising 60,000 color images categorized into 100 distinct classes, each containing 600 images. Each image has a relatively low resolution of 32×32. It is split into 50,000 images for training and 10,000 for testing, and we further divide the

Table 1: Comparison of Regular Convolution and MobileNet's Depthwise Separable Convolution

Operation	**# Parameters**	**# FLOPs**
Regular Convolution	$K^2 \times C_{in} \times C_{out}$	$K^2 \times C_{in} \times C_{out} \times M^2$
MobileNet: Depthwise + Pointwise	$K^2 \times C_{in} + C_{in} \times C_{out}$	$K^2 \times C_{in} \times M^2 + C_{in} \times C_{out} \times M^2$

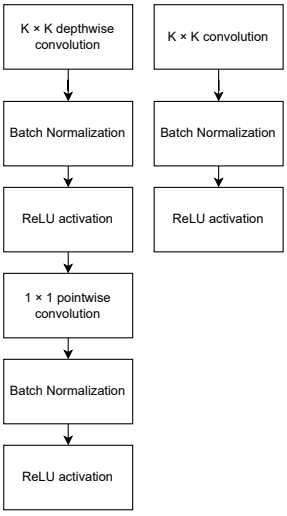

Figure 1: Architectural difference of MobileNet with depthwise and pointwise convolution (left) versus standard convolution (right)

50,000-image set into 40,000 images for training and 10,000 for validation.

In all our experiments, we will utilize the validation set of CIFAR-100 for hyper-parameter tuning and and the test set for model evaluation.

Experimental Setup

As the goal is to understand the effects of certain hyper-parameters, we will keep the following settings fixed unless stated otherwise:

- We use SGD with momentum (Sutskever et al. 2013), setting $\beta = 0.9$ as standard.
- We use a batch size of 128.
- Standard augmentations (random cropping and random horizontal flip) are applied (Krizhevsky, Sutskever, and Hinton 2012).

Experiments

Learning Rate

The first experiment concerns the learning rate. In gradient descent optimization, the learning rate η is the hyper-parameter that controls the magnitude of updates for model parameters w:

$$w_t = w_{t-1} - \eta \left. \frac{\partial \mathcal{L}}{\partial w} \right|_{w_{t-1}} \tag{1}$$

Selection of the learning rate can affect the speed and stability of model training. If the learning rate is too low, the weights can take longer to converge, and the model can get stuck in bad local minima. If the learning rate is too high, the model can potentially overshoot the optimal regions in the loss landscape, causing divergence.

Table 2: Final losses and accuracies for different learning rates

Learning Rate	0.01	0.05	0.2
Final Train Loss	1.953	**1.762**	1.838
Final Valid Loss	2.207	**2.042**	2.059
Final Train Accuracy	46.314	**50.481**	48.510
Final Valid Accuracy	42.250	**46.040**	44.800

For this part of the experiment, we train three models using the MobileNet architecture with three different learning rates — 0.2, 0.05 and 0.01 — for 15 epochs each and examine the model's accuracy and loss on both the training and validation set. The accuracy and loss curves for each setting are presented in Figure 2 and the final losses and accuracy values are reported in Table 2.

Regarding the training losses and accuracies, the configuration with 0.05 learning rate performed best across all epochs, followed by the 0.2 and 0.01 configurations. The same is also true on the validation set. We can see There are minor spikes in the validation loss of learning rates 0.2 and 0.05 while the 0.01 model's loss decreases stably, indicating the instability associated with higher learning rate discussed earlier. After 15 epochs, we observe that the model with learning rate set to 0.05 achieved the best final loss (1.762) and accuracy (50.482%) on the training set. This configuration seems to have achieved a good balance between 0.01 and 0.2 learning rates.

For subsequent experiments, we will employ 0.05 as the initial learning rate.

Learning Rate Schedule

The second experiment concerns learning rate schedule. Keeping the learning constant is not always ideal when training deep neural networks. Instead, we can utilize a learning rate schedule. The high-level idea is start with a certain learning rate ensure fast training initially and avoid poor local minima, then decrease the learning rate as the loss comes

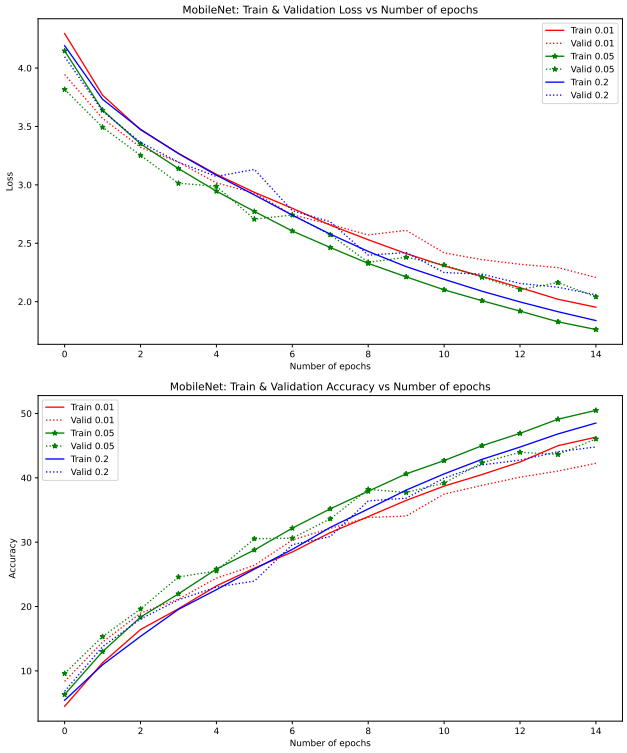

Figure 2: Training and Validation Losses and Accuracies for MobileNet using Different Learning Rates.

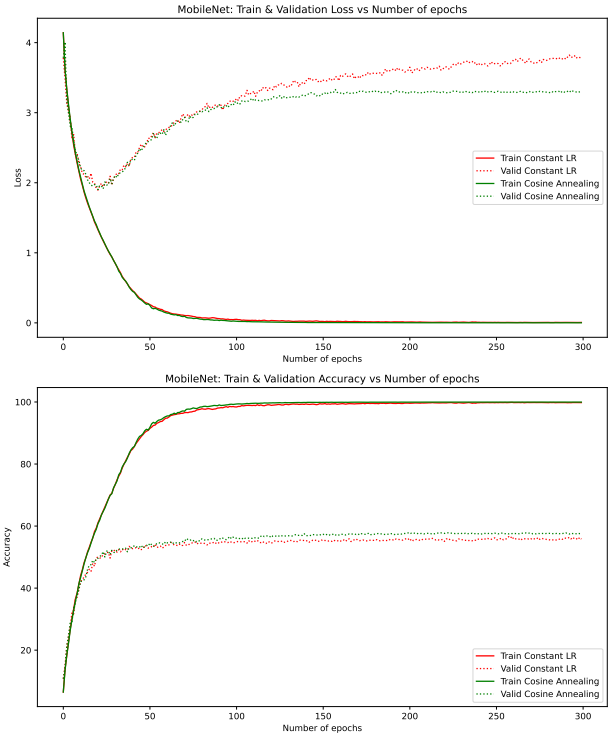

Figure 3: Training and Validation Losses and Accuracies for MobileNet using Constant Learning Rate versus Cosine Annealing Schedule. (initial_lr=0.05)

close to minima to stabilize training and prevent overshooting.

One strategy for learning rate schedule is Cosine Annealing (Loshchilov and Hutter 2017). During training, the learning rate is decayed using the formula:

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{t}{T}\pi\right)\right) \quad (2)$$

where η_{max} is the initial learning rate, η_{min} is the minimum learning rate (usually set to 0 or a small positive value), t is the current epoch and T is the total epochs. The decrease in learning rate follows the shape of a half cycle of a cosine function. In early training and towards the end, the learning rate decays slowly, but in the middle stages it drops steeply.

We train the two models for 300 epochs using 0.05 as the initial learning rate, as discovered during the previous section. The first model is trained with a constant learning rate, while the second uses the Cosine Annealing Learning Rate Scheduler, which decreases the learning rate to zero over all training epochs. Figure 3 shows the learning curves of the two configurations. Table 3 summarizes the final losses and accuracies.

At a glance, training losses for both strategies level off around epoch 50. However a zoomed-in look at the losses will reveal that the losses when constant learning rate fluctuated wildly, whereas the loss curve for cosine annealing is comparatively more stable. In the validation set, the loss

Table 3: Final Losses and Accuracies for Constant LR and Cosine Annealing Schedule

Learning Rate Schedule	Constant LR	Cosine Annealing
Final Train Loss	0.007	**0.001**
Final Valid Loss	3.786	**3.301**
Final Train Accuracy	99.800	**99.965**
Final Valid Accuracy	55.800	**57.510**

actually increased from epoch 30, indicating that both models overshot optimal regions in the loss landscape, but the loss when using cosine annealing plateaus from epoch 100 onward, whereas the loss of a constant learning rate continues to increases. This is because in later epochs, the cosine annealing scheduler decreases the loss to close to zero, making update steps extremely small and preventing the model from further overshooting. We can see a similar stabilization effect in the training and loss accuracies of the model with cosine annealing scheduler compared to the one with constant learning rate.

After training for 300 epochs, we observe that the model with Cosine Annealing Schedule outperformed the model using constant learning rate in terms of training loss and accuracy.

Table 4: Final losses and accuracies for different weight decay values

Weight Decay	1e-4	5e-4
Final Train Loss	**0.003**	0.012
Final Valid Loss	1.778	**1.394**
Final Train Accuracy	**99.985**	99.942
Final Valid Accuracy	60.320	**67.950**

Weight Decay

The third experiment involves weight decay. Weight decay promotes generalization by constraining model complexity, ensuring the network learns robust features rather than noise. Specifically, in each gradient descent update, we further subtract $\eta\lambda w_t$ from the weight w:

$$w_{t+1} = w_t - \eta\frac{\partial\mathcal{L}(w_t)}{\partial w_t} - \eta\lambda w_t \tag{3}$$

where λ is a hyper-parameter that controls the strength of the regularization effect.

This is equivalent to performing optimization on a loss function with an added L2 regularization term $\frac{1}{2}\lambda\|w\|^2$. The introduction of the regularization term proportional to the squared magnitude of the weights penalizes larger weight values and thus forces the network to learn robust features rather than noise.

Using the initial learning rate of 0.05 with Cosine Annealing Schedule, we train two models using the weight decay coefficients $\lambda = 5 \times 10^{-4}$ and 1×10^{-4}. The models' learning curves are in Figure 4 and final performance on the train and test set are reported in Table 4.

In the losses graph, we observe that the model with larger weight decay $\lambda = 5 \times 10^{-4}$ takes longer to converge compared to the one with $\lambda = 1 \times 10^{-4}$. The validation loss for the 5×10^{-4} model, while fluctuating during the middle stages of training (from epoch 25 to around epoch 225), was consistently lower than the loss of the 1×10^{-4} model. A possible explanation is that a larger weight decay coefficient forces the model to converge more slowly because it reduces the magnitude of the update. It also enforces a simpler model and therefore prevents overfitting.

Interestingly, the model with 1×10^{-4} achieved better performance on the training set compared to the 5×10^{-4} model, but trailed in validation set evaluations. Again, this demonstrates the regularization effect of larger weight decay coefficient preventing overfitting. It is also noteworthy that the 5×10^{-4} model improved upon the previous model (without weight decay) in validation accuracy by 10.44%, indicating better generalization.

Performance on Test Set

We use the previously trained models with Cosine Annealing and Weight Decay to make predictions on the hold-out test set. As the test set has never been seen by the models, accuracy on this set is indicative of performance on future unseen data. 5 shows the accuracy of the models on the test set.

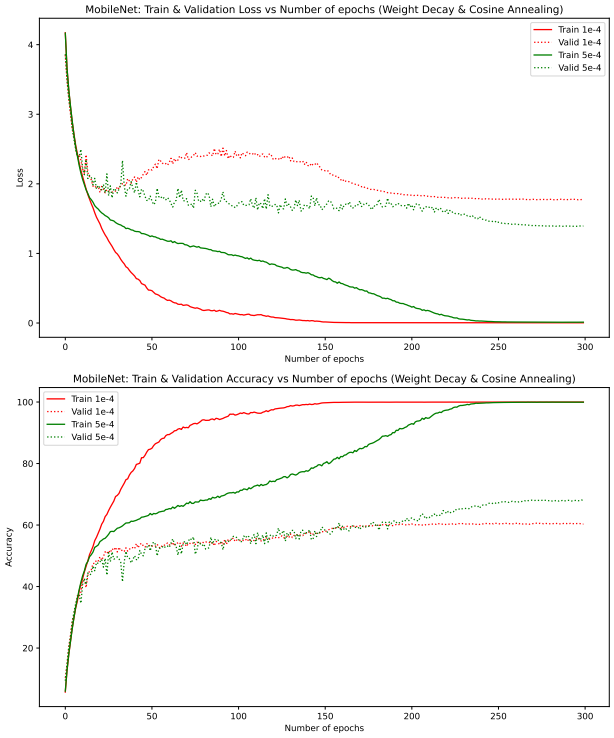

Figure 4: Training and Validation Losses and Accuracies for MobileNet using Different Weight Decay Coefficients and Cosine Annealing Schedule. (initial_lr=0.05)

Table 5: Test accuracies for different weight decay values

Weight Decay	1e-4	5e-4
Test Accuracy	61.99	**68.12**

In the end, the model using weight decay 5×10^{-4} performed better on the hold-out set, achieving 68.12% accuracy.

Activation Function

Additionally, we perform an experiment regarding the activation function of the model. Instead of the ReLU activation used in the previous experiments, we set the blocks [4, 5, 6, 7, 8, 9, 10] to use Sigmoid activation.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{4}$$

Other hyper-parameters follow the best-performing configurations we have discovered in the last sections: Cosine Annealing with initial learning rate set to 0.05, and weight decay with $\lambda = 5 \times 10^{-4}$. We train this model for 300 epochs. The learning curves of the new model is plotted together with the counterpart model using ReLU activation in Figure 5.

We also examined the 2-norm of the gradient vector of 'model.layers[8].conv1.weight' during training of the model using Sigmoid activation in Figure 6.

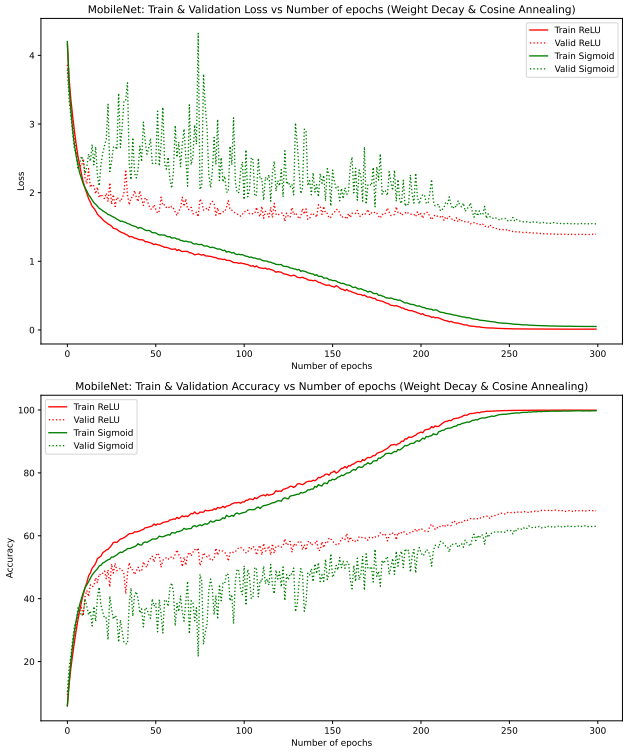

Figure 5: Training and Validation Losses and Accuracies for MobileNet using Different Activation Functions for layers 4 to 10, with Weight Decay and Cosine Annealing

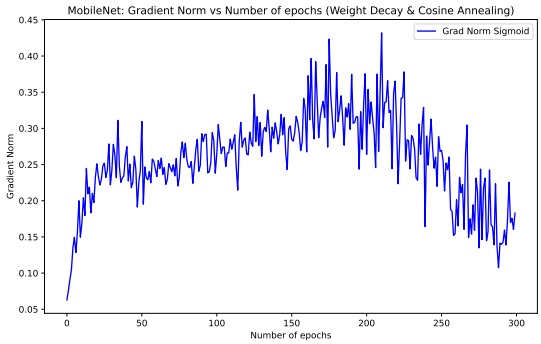

Figure 6: 2-norm of the gradient vector of 'model.layers[8].conv1.weight' of the Sigmoid model.

Overall, we can see that the model using ReLU activation achieved better loss and accuracy than then one using Sigmoid. Furthermore, there are frequent large spikes in the loss curves of the Sigmoid model, signifying inconsistent updates.

This could be due to the fact that sigmoid gradients vanish for extreme inputs due to saturation, leading to vanishing gradients. Neurons may alternate between saturated and active states, resulting in erratic loss fluctuations. We can also see oscillatory behavior in the gradient norm due to the same reason.

Meanwhile, the gradient of the ReLU function is 1 for positive inputs and 0 for negative inputs. This avoids vanishing gradients for active neurons, enabling consistent updates when inputs are positive, leading to smoother loss reduction.

Conclusion

In this report, we have experimented with several hyper-parameters that affected the optimization and regularization of MobileNet model, namely: learning rate, learning rate schedule, weight decay and activation function. Below are some key takeaways from this assignment:

- Choosing a suitable learning rate is important. A learning rate too high causes the model to overshoot and lead to divergence. A learning rate too low causes slow training and the model can get stuck in poor minima.
- Using a learning rate schedule to decrease the learning rate during training proves advantageous. Setting a high initial learning rate helps the model escapes poor local minima, gradually decreasing the learning rate towards the end stabilizes training and ensures smooth convergence.
- Using weight decay is an effective regularization strategy, and can result in better generalization. Setting a good decay coefficient is key, as a unsuitable value may hurt performance on unseen data.
- For deep networks, using a ReLU activation helps avoid the vanishing gradient problem found with Sigmoid activation.

Overall, this has been an insightful exercise in experimenting with hyper-parameters. These findings provide practical guidelines for training efficient architectures like MobileNet, particularly in resource-constrained settings.

References

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. The MIT Press. ISBN 0262035618.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR*, abs/1704.04861.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F.; Burges, C.; Bottou, L.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Krogh, A.; and Hertz, J. 1991. A Simple Weight Decay Can Improve Generalization. In Moody, J.; Hanson, S.; and Lippmann, R., eds., *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86: 2278–2324.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Nair, V.; and Hinton, G. E. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In Fürnkranz, J.; and Joachims, T., eds., *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, 807–814. Omnipress.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the importance of initialization and momentum in deep learning. In Dasgupta, S.; and McAllester, D., eds., *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, 1139–1147. Atlanta, Georgia, USA: PMLR.