# Optimizing Runtime in Deep Learning: Early Stopping and Learning Rate Schedulers

May 06, 2025

## Summary

The report investigates techniques for optimizing runtime in deep learning models, mainly on early stopping and learning rate schedulers. These methods aim the reduce training time and improve model performance. We explored the trade-offs between training efficiency and model accuracy, as well as the effectiveness of different optimization strategies. By conduct an experimental study using a CNN model on the MNIST data set, we found that early stopping significantly reduces training time without compromising accuracy, and learning rate schedulers also affects the performance of the model.

## Introduction

The deep learning models have achieved remarkable success in tasks such as image classification and natural language processing in the past years. However, training these models cost lots for time, due to the large data sets and complex architectures (Goodfellow, Bengio, and Courville 2016). How to reduce the running time is very important for making deep learning more accessible. The early stopping and learning rate scheduling are two widely used techniques. Early stopping is a form of regularization used to prevent overfitting in machine learning and deep learning models. It involves stopping the training process before the model starts to overfit The learning rate schedulers are algorithms that allow you to control your model's learning rate according to some pre-set schedule or based on performance improvements. This report investigates these techniques through an experimental study using the MNIST data set, the study aligns with best practices for deep learning.

## Current Research

Recent research highlights the importance of runtime optimization in a deep-learning. Early stopping is the most important technique for tracking validation metrics, including loss or accuracy, it will stop the training when no there is no more improvement, thus preventing overfitting and computational wastage (Prechelt 1998). While learning rate scheduling involves changing the learning rate based on given rules or performance criteria. Some common approaches include the step decay, exponential decay, and the adaptive one called ReduceLROnPlateau, which reduces the learning rate once the monitored metric stops improving (Bengio 2012). While Smith proposed that cyclical learning rates improve convergence by allowing the model to escape local minima Smith (2017), Loshchilov and Hutter proposed SGDR (Stochastic Gradient Descent with Warm Restarts) to enhance training efficiency Loshchilov and Hutter (2017). Such methods fit well with automatic hyperparameter tuning and performance monitoring as they dynamically adjust according to the training process. However, these methods' success depends on factors like dataset specifics, model architecture, and hyperparameter settings and, hence, have to be empirically evaluated.

## Data Collection / Model Development

### Data Collection

This experimental study used the MINIST data set, a classical data set for image classification tasks(LeCun et al. 1998). The data set contains 60,000 gray scale images of handwriting digits (0-9) for training and 10,0000 images for testing, the size of each image is $28 \times 28$ pixels. For simplicity and reducing running time, I used a subset of 10,000 images for training and 2,000 images for testing. We also applied normalization to the images.

### Model Development

We build a CNN model with the following architechure:

- **Input Layer**: Accepts $28 \times 28 \times 1$ grayscale images.
- **Convolutional Layer**: 32 filters, $3 \times 3$ kernel, ReLU activation.
- **MaxPooling Layer**: $2 \times 2$ pool size.
- **Flatten Layer**: Converts 2D feature maps to a 1D vector.
- **Dense Layer**: 64 units, ReLU activation.
- **Output Layer**: 10 units, softmax activation for classification.

Then we compiled the model with Adam optimizer, sparse categorical cross-entropy loss, and accuracy as the evaluation metric.

Four training configurations were tested:

1. **Model 1**: Early stopping with a patience of 3 epochs, monitoring validation loss.
2. **Model 2**: ReduceLROnPlateau scheduler, reducing the learning rate by a factor of 0.5 if validation loss does not improve for 2 epochs.
3. **Model 3**: Step decay scheduler, halving the learning rate every 5 epochs.
4. **Model 4**: Exponential Decay scheduler.

We implemented these models with tensorflow and keras, then train each model up to 20 epochs with a validation split of 0.2. We used the callbacks to track performance and adjust hyperparameters automatically.

## Analysis

### Experimental Results

- **Model 1 (Early Stopping)**: The training stopped after 10 epochs as there is no improvement in validation loss for 3 consecutive epochs, the test accuracy is 0.955. The model has the highest accuracy.
- **Model 2 (ReduceLROnPlateau)**: The training stopped after 3 epochs, it has a test accuracy of 0.9105, the model shows faster convergence but lower accuracy compared with model 1.

- **Model 3 (Step Decay)**: The training also stopped after 3 epochs, it has a test accuracy of 0.9090, the model shows faster convergence but lower accuracy compared with model 1.
- **Model 4 (Cosine Annealing)**: The training also stopped after 3 epochs, it has a test accuracy of 0.9090, it has the same performance as model 3.

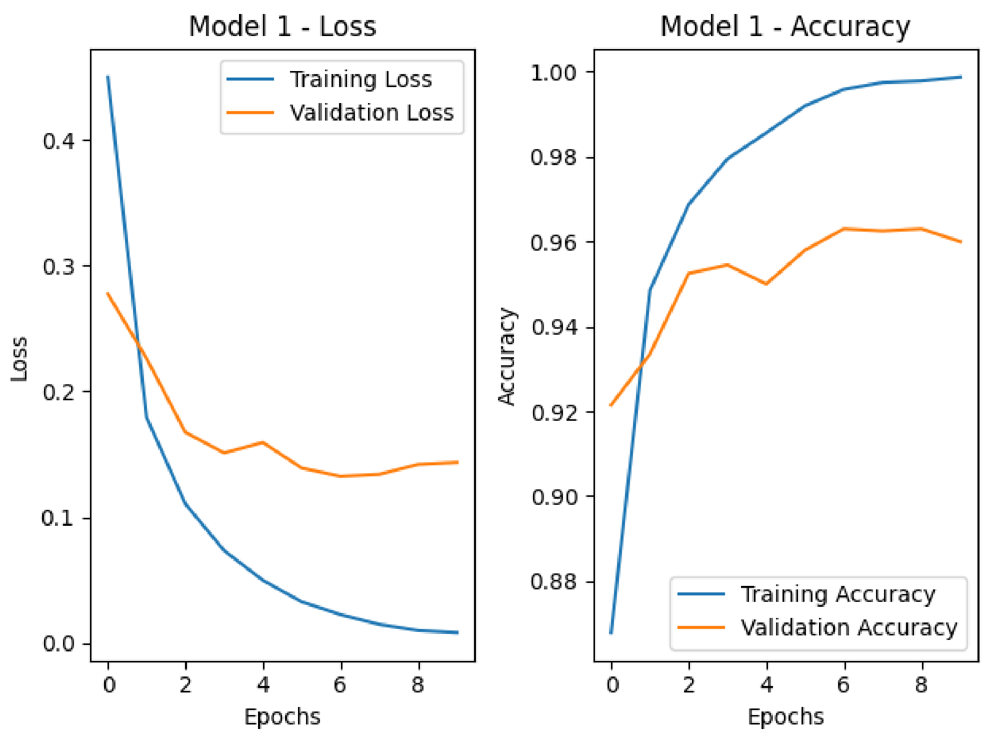

Figure 1: Training and validation loss/accuracy for Model 1 (Early Stopping). The plots show that validation loss stabilized after epoch 7, triggering early stopping at epoch 10.

### Key Findings

The model 1 achieved the highest test accuracy (0.955) with only 10 epochs, it reduces the training time by 50% compared with the maximum 20 epochs. The result aligns with (Prechelt 1998), it confirms that early stopping effectively reduce the running time. Models 2, 3, and 4 show lower test accuray and stopped early(in 3 epochs), it suggests that the learning rate schedulers did not provide significant benefits for the MNIST subset.

For model 2, the adaptive reduction in learning rate was too conservative, leading to slower convergence and lower accuracy compared to Model 1. The fixed schedule in model 3 had minimal effect as the early stopping.

## Summary and Conclusions

Based on the analysis above, we found that the application of early stopping reduces training time considerably while maintaining considerable accuracy, as shown through the performance of Model 1, which recorded a test accuracy of 0.9550 in 10 epochs on a subset of the MNIST dataset. Learning rate schedulers yielded modest improvements, with Models 2–4 having lower accuracies (0.9090–0.9105) and converging prematurely owing to the simplicity of the dataset. These results underscore the need for optimization methods to be specialized to the task and dataset. In further work, we can explore hybrid methods integrating early stopping with more intense scheduler settings or assess these methods on larger, more challenging data sets.

## References

Bengio, Y. 2012. “Practical Recommendations for Gradient-Based Training of Deep Architectures.” In *Neural Networks: Tricks of the Trade*, 437–78. Springer. https://doi.org/10.1007/978-3-642-35289-8_25.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. “Gradient-Based Learning Applied to Document Recognition.” *Proceedings of the IEEE* 86 (11): 2278–2324. https://doi.org/10.1109/5.726791.

Loshchilov, I., and F. Hutter. 2017. “SGDR: Stochastic Gradient Descent with Warm Restarts.” In *International Conference on Learning Representations*. https://arxiv.org/abs/1608.03983.

Prechelt, L. 1998. “Early Stopping—but When?” *Neural Networks: Tricks of the Trade*, 55–69. https://doi.org/10.1007/3-540-49430-8_3.

Smith, L. N. 2017. “Cyclical Learning Rates for Training Neural Networks.” In *IEEE Winter Conference on Applications of Computer Vision*, 464–72. IEEE. https://doi.org/10.1109/WACV.2017.58.