

Practical Work 2: Improving Deep Neural Networks

Louis Fippo Fitime
Claude Tinku
Kerolle Sonfack

Department of Computer Engineering, ENSPY
University of Yaoundé I

September 22, 2025

Abstract

This second Practical Work focuses on advanced engineering practices for deep neural networks. The goal is to go beyond basic model training by applying techniques such as performance diagnosis, regularization, normalization, and advanced optimization algorithms in order to build more robust and efficient models.

1 Part 1: Theory and Key Concepts

1.1 Performance Diagnosis: Bias vs. Variance

Performance diagnosis is a critical step in the lifecycle of a deep learning model. A model suffers from **high bias** (underfitting) when it is too simple to capture the underlying patterns in the data, resulting in poor performance even on the training set. Conversely, a model suffers from **high variance** (overfitting) when it fits the training data very well but fails to generalize to unseen data.

Data Splitting

Dividing the dataset into three distinct subsets is essential:

- **Training set:** used to learn the model parameters.
- **Validation (dev) set:** used to tune hyperparameters and diagnose bias or variance.
- **Test set:** used only once to evaluate the final performance of the model.

This separation ensures an unbiased evaluation of the model's generalization ability.

Results Analysis

By comparing training and validation errors, one can easily diagnose model issues:

- High training error and high validation error indicate **high bias**.
- Low training error but high validation error indicate **high variance**.

This simple analysis guides the choice of corrective actions such as increasing model complexity or applying regularization.

1.2 Regularization and Normalization

L2 Regularization

L2 regularization, also known as *weight decay*, adds a penalty term proportional to the square of the weights' magnitude to the loss function. This discourages large weights and forces the model to learn simpler and more generalizable representations.

Dropout

Dropout randomly disables a fraction of neurons during each training iteration. This prevents neurons from co-adapting too strongly and encourages the network to learn robust features, effectively reducing overfitting.

Batch Normalization

Batch Normalization standardizes the activations of hidden layers by maintaining a zero mean and unit variance within each mini-batch. This stabilizes training, allows higher learning rates, and significantly accelerates convergence.

1.3 Advanced Optimization Algorithms

The optimization algorithm strongly influences training speed and stability.

- **Momentum**: accelerates gradient descent by accumulating a velocity vector that smooths parameter updates.
- **RMSprop**: adapts the learning rate for each parameter based on a moving average of squared gradients.
- **Adam**: combines Momentum and RMSprop, providing fast convergence, stability, and minimal hyperparameter tuning.

Due to its robustness and efficiency, Adam is often considered the default optimizer in deep learning applications.

2 Part 2: Practical Exercises

2.1 Exercise 1: Bias and Variance Analysis

After splitting the dataset into training, validation, and test sets, the model's performance is evaluated on both the training and validation data.

Diagnosis

If the observed results show high accuracy on the training set but significantly lower accuracy on the validation set, the model suffers from high variance. This indicates overfitting and suggests the need for regularization techniques.

2.2 Exercise 2: Applying Regularization

By adding L2 regularization and Dropout layers, the model becomes less sensitive to noise and reduces overfitting.

Analysis

The regularized model typically shows slightly lower training accuracy but improved validation performance. This demonstrates better generalization, confirming the effectiveness of regularization techniques.

2.3 Exercise 3: Comparing Optimization Algorithms

Different optimizers are evaluated using MLflow to track experiments.

Comparison

Adam generally converges faster and achieves higher accuracy in fewer epochs compared to SGD with momentum and RMSprop. MLflow facilitates systematic comparison by logging metrics and parameters for each run.

2.4 Exercise 4: Batch Normalization

Adding a Batch Normalization layer improves training stability and accelerates convergence. The loss decreases more smoothly, and the model reaches optimal performance in fewer epochs.

3 Conclusion

This practical work demonstrated the importance of diagnosing bias and variance, applying regularization, and selecting appropriate optimization algorithms. These techniques are essential for building robust and high-performance deep learning models suitable for real-world deployment.