Understanding Overfitting in Neural Networks and Techniques to Prevent It

A TensorFlow CNN case study using Fashion-MNIST

Name: Ogundero Stephen Abimbola

Student ID: 24086166

Dev Article: https://dev.to/stephen_ogundero_7c2f6842/understanding-overfitting-in-neural-networks-tensorflow-cnn-353b

GitHub repository: https://github.com/Stephen-OG/Neural-Network-Overfitting-Tutorial.git

---

1. Introduction

Neural networks, especially Convolutional Neural Networks (CNNs), have become the default models for many computer vision tasks [1]. They can learn rich, hierarchical features directly from image data and achieve state-of-the-art results on benchmarks such as MNIST, CIFAR and ImageNet. However, this expressive power comes with a cost: CNNs are extremely prone to **overfitting**. With millions of parameters, they can easily memorise the training set instead of learning patterns that generalise to unseen data.

In this tutorial, I focus on how CNNs behave under **overfitting** and how several regularisation techniques can be used to reduce it:

- **Dropout**
- **L2 weight decay (weight regularisation)**
- **Early stopping**

As an application, I train small CNNs on the **Fashion-MNIST** dataset using TensorFlow/Keras [5]. I deliberately restrict the training set to 10,000 images to make overfitting more visible. I then compare a baseline model with three regularised variants and discuss how the training and validation curves change.

The goal of the tutorial is that a reader who already knows basic neural networks will understand what overfitting looks like in practice, and how to apply simple but powerful regularisation tricks in their own work.

2. Underfitting and Overfitting

When training neural networks we usually care about performance on unseen data, not just on the training set. Three typical regimes are:

Concept	**Description**
Underfitting	The model is too simple or trained for too few epochs. It fails to capture the structure in the data and has **high error on both training and validation** sets.
Overfitting	The model is too flexible or trained for too long. It **memorises noise** in the training set and achieves low training error but high validation error.
Good generalisation	The model has enough capacity and regularisation. Training and validation performance are **similar**, and both errors are reasonably low.

In CNNs, overfitting is common because convolutional and dense layers together can represent very complex functions. Without regularisation, the optimisation process will happily keep reducing training loss even after validation performance has stopped improving. This behaviour is best seen in plots of training vs. validation loss and accuracy over epochs.

3. Experimental Setup

I use the Fashion-MNIST dataset, which contains 60,000 training and 10,000 test grayscale images of clothing items, each of size 28×28 pixels [5]. To encourage overfitting, only **10,000** training examples are used, with 20% of those reserved for validation.

All models share the same basic CNN architecture:

- Conv2D(32 filters, 3×3, ReLU) + MaxPooling
- Conv2D(64 filters, 3×3, ReLU) + MaxPooling
- Flatten
- Dense(128, ReLU)
- Dense(10, softmax output)

I train four variants in TensorFlow/Keras:

Model	Regularisation applied
Baseline	No regularisation
Model B	Dropout with rate 0.5 before the final dense layer
Model C	L2 weight decay ($\lambda = 0.001$) on convolution and dense layers
Model D	Dropout (0.5) + L2 ($\lambda = 0.001$) + EarlyStopping on validation loss

All models are trained using the Adam optimiser with batch size 128. For the baseline, Dropout and L2 models I train for 30 epochs. For the early-stopping model I allow up to 50 epochs but use an `EarlyStopping` callback that stops training if validation loss does not improve for 5 epochs, restoring the best weights.

The full TensorFlow code is in the Jupyter notebook in the GitHub repository [6] [7].

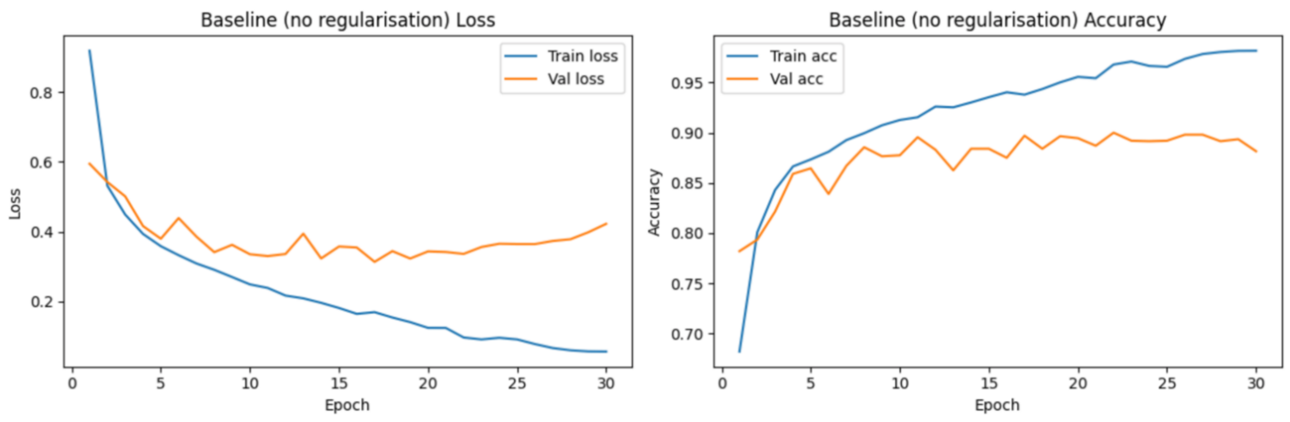
4. Results and Graph Interpretations

4.1 Baseline CNN – clear overfitting

The baseline model, trained without any regularisation, shows classic overfitting. In the loss plot (**Figure 1**), the training loss steadily decreases over all 30 epochs, while the validation loss decreases initially but starts to increase again after about epoch 10. In the accuracy plot, training accuracy keeps climbing towards about 97%, whereas validation accuracy plateaus around 88% and even fluctuates down slightly.

The final test accuracy for this model is **0.8796**. Although this is not terrible, the gap between training and validation curves indicates that the model is learning training-specific noise rather than robust features.

Figure 1. Baseline Overfitting Behaviour



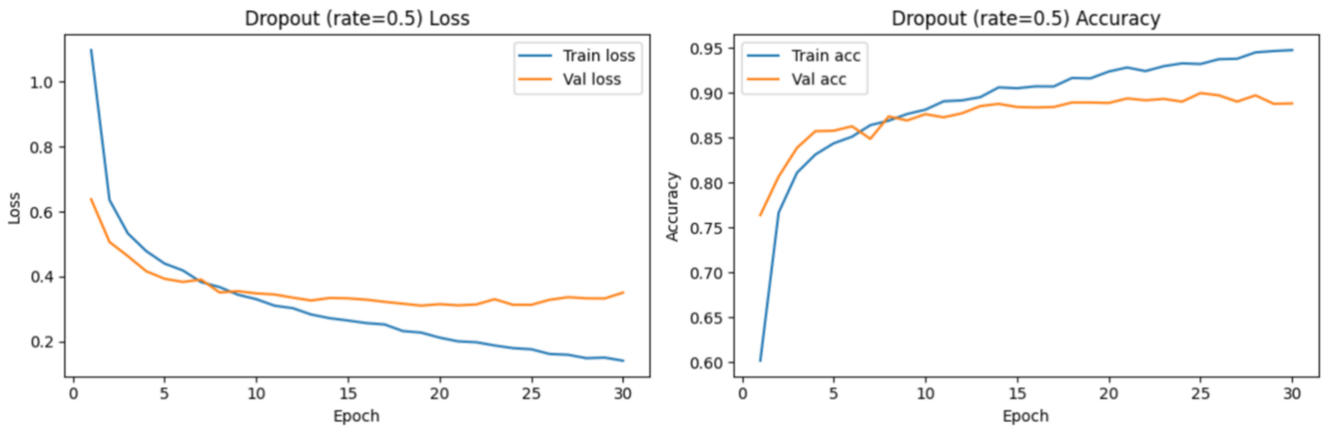
The CNN has too much effective capacity for the small subset of data and no mechanism to restrict it, so it overfits strongly.

4.2 CNN with Dropout (rate = 0.5)

Dropout randomly sets a proportion of activations to zero during training [2]. This prevents units from co-adapting too strongly and acts like averaging over many thinned networks. In this experiment I apply dropout with rate 0.5 before the final dense layer.

The training and validation curves in **Figure 2** are noticeably closer together than in the baseline case. Training accuracy is slightly lower, but validation accuracy improves and becomes more stable. The final test accuracy rises to **0.8865**, the best of all models.

Figure 2. Dropout Effect



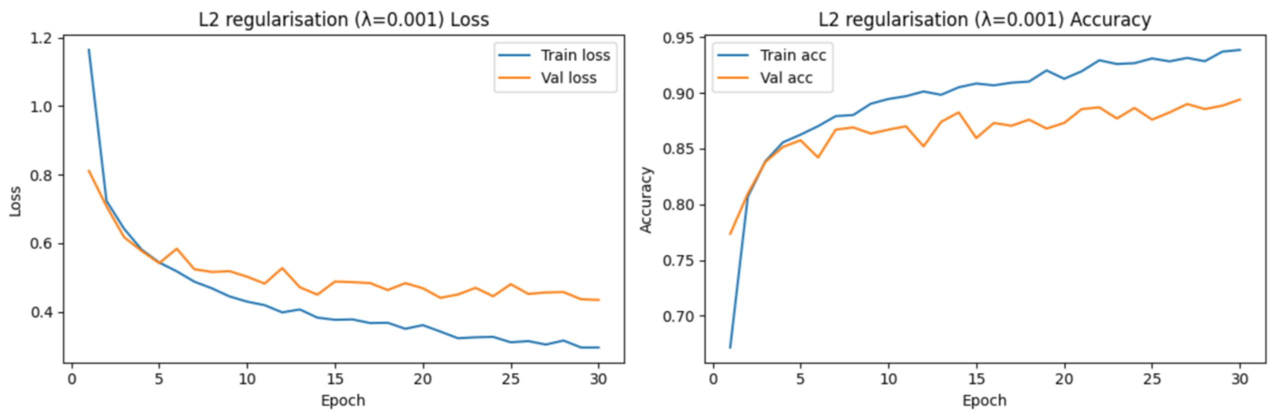
By forcing the network to rely on multiple redundant representations, dropout reduces overfitting and improves generalisation performance.

4.3 CNN with L2 weight regularisation (λ = 0.001)

L2 regularisation, or weight decay, adds a penalty term proportional to the squared magnitude of the weights [3]. This biases the optimisation towards smaller weights and smoother functions.

In **Figure 3**, the L2-regularised model shows smoother loss and accuracy curves than the baseline. The gap between training and validation accuracy is smaller, and validation loss does not rise as sharply. The final test accuracy is **0.8828**, an improvement over the baseline but slightly below the dropout model.

Figure 3. L2 Regularization



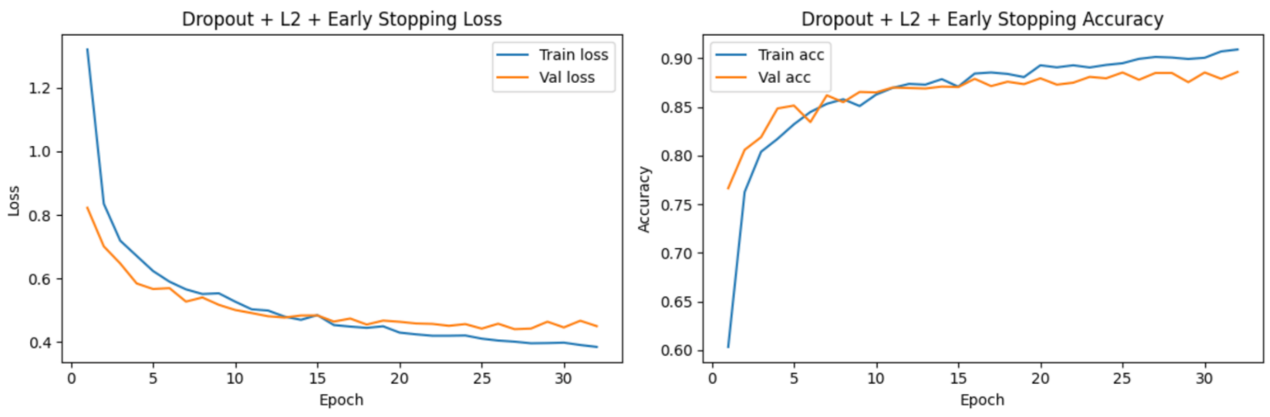
L2 reduces the effective complexity of the model, but on this task a simple dropout layer is slightly more effective than L2 alone.

4.4 Dropout + L2 + Early Stopping

The final model combines dropout and L2 with **early stopping**, which monitors validation loss and stops training when there is no improvement for several epochs. This prevents the model from entering the late-epoch regime where overfitting becomes worse [4].

The curves in **Figure 4** are the most stable: training and validation loss track each other closely and do not diverge. Validation accuracy remains high throughout training. The final test accuracy is **0.8832**, close to the dropout-only model but with better-behaved training dynamics.

Figure 4. Early Stopping



Combining several regularisation methods does not dramatically increase raw accuracy here, but it produces a more reliable training process and reduces the risk of over-training.

4.5 Final comparison

A summary of the models and their test accuracies is shown below:

Model	Test accuracy
Baseline (no regularisation)	0.8796
Dropout (0.5)	0.8865
L2 regularisation ($\lambda = 0.001$)	0.8828
Dropout + L2 + EarlyStopping	0.8832

All three regularised models outperform the baseline. Dropout alone yields the highest accuracy, while the combined model offers the best stability. This demonstrates that even simple regularisation techniques can significantly improve the generalisation of a CNN trained on a relatively small dataset.

5. Conclusion

This tutorial has shown how overfitting appears in practice when training a CNN on Fashion-MNIST and how common regularisation methods can mitigate it [5]. On a reduced training set of 10,000 images, the unregularised model overfits strongly: training loss continues to decrease while validation loss rises, and there is a clear gap between training and validation accuracy.

Adding **dropout** reduces the gap and improves test accuracy the most. **L2 weight regularisation** smooths the learnt weights and yields a modest improvement over the baseline. **Early stopping** complements these techniques by automatically stopping training when validation performance stops improving.

The main lesson is that regularisation is not optional for neural networks: a model that generalises well is more useful than a model that merely memorises the training data. In practical work, I would recommend always monitoring training and validation curves, starting with simple techniques like dropout and early stopping, and only increasing model complexity once generalisation is under control.

6. References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014.

[3] A. Krogh and J. A. Hertz, "A Simple Weight Decay Can Improve Generalization," in Advances in Neural Information Processing Systems (NIPS), 1992, pp. 950–957.

[4] L. Prechelt, “Early Stopping — But When?,” in Neural Networks: Tricks of the Trade, Springer, 1998, pp. 55–69.

[5] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms,” arXiv preprint arXiv:1708.07747, 2017.

[6] F. Chollet et al., “Keras Documentation,” 2015–2024. [Online]. Available: https://keras.io

[7] Implementing a convolutional neural network using eras (Based on Victor Zhou keras tutorial)