Understanding Dropout in Neural Networks: A Practical Tutorial on Regularisation and Generalisation

Name: Srujan Gullapalli

Student ID: 24099354

Course: MSc Data Science with Adv. Research

GitHub: https://github.com/JISOOUNNIE1234/Srujan-Machine-learning.git

1. Introduction:

Modern neural networks are powerful learning systems capable of fitting extremely complex functions. While this flexibility is beneficial, it also creates a fundamental challenge: overfitting. When a network learns patterns that are too specific to the training data rather than general principles, its performance on unseen data deteriorates. Regularisation techniques help constrain this tendency, leading to models that generalise better in real-world use.

Among the various regularisation strategies, dropout has proven to be one of the simplest and most effective. Introduced by Srivastava et al. (2014), dropout involves randomly disabling a proportion of neurons during training. This seemingly destructive process forces the network to learn redundantly robust representations, reducing its reliance on any single pathway and thereby improving generalisation.

Although dropout is widely used in deep learning, many learners develop only a surface-level understanding of how it works. This tutorial aims to provide a clear, intuitive, and experimentally grounded explanation of dropout. Using a controlled experiment on a subset of the Fashion-MNIST dataset, we compare two identical neural networks one with dropout and one without and investigate how dropout affects learning dynamics and model performance.

The goal is to give readers a practical and conceptual understanding of dropout so they can confidently apply it in their own work.

2. Conceptual Foundations of Dropout.

2.1 What Is Dropout?

In a standard neural network, every neuron contributes to every forward pass during training. Dropout disrupts this behaviour by zeroing out neurons at random. Formally, for a given vector of hidden activations $\boldsymbol{h}$, dropout samples a binary mask $\boldsymbol{m}$ from a Bernoulli distribution:

$$\boldsymbol{m_i} \sim \text{Bernoulli}(\mathbf{1} - \boldsymbol{p})$$

and computes:

$$\boldsymbol{h^{drop} = m \odot h}$$

where $\boldsymbol{p}$ is the dropout rate. For example, with $\boldsymbol{p} = \mathbf{0.5}$, half of the activations are removed at random during each update.

2.2 Why Does Dropout Work?

Dropout succeeds for three major reasons:

- Prevention of Co-Adaptation
  Neurons cannot depend on specific inputs because those neurons might be dropped. This forces the network to learn more distributed and redundant features.
- Implicit Model Averaging
  Each dropout mask effectively samples a subnetwork. During inference the fully connected network approximates the average prediction of exponentially many subnetworks.
- Noise Injection and Smoothing of the Loss Landscape
  The randomness discourages the network from settling into sharp local minima that generalise poorly.

2.3 Training vs Evaluation Behaviour

A critical detail is that dropout behaves differently in training and evaluation modes:

- Training mode: Random neurons are dropped without scaling.
- Evaluation mode: No neurons are dropped, but activations are scaled by $1 - p$.

This maintains consistent expected activation magnitudes between training and inference phases.

---

3. Experimental Design

3.1 Motivation

To reveal the role of dropout, we compare two networks under controlled conditions. Both models use identical architectures, optimisation strategies, and datasets the only difference is the use of dropout. By holding all other variables constant, we can attribute differences in behaviour directly to dropout.

3.2 Dataset Description

We use the **Fashion-MNIST** dataset (Xiao et al., 2017), consisting of 70,000 grayscale images (28×28 pixels) across 10 clothing categories. Unlike MNIST digits, Fashion-MNIST offers more realistic visual complexity and variability.

To make overfitting more prominent and to introduce an original tutorial angle, we restrict the dataset to **three visually similar classes**:

- Class 0: T-shirt/Top
- Class 2: Pullover
- Class 4: Coat

This 3-class subset encourages the models to learn fine-grained distinctions while keeping training time manageable.

After filtering, we normalise the images to [0,1], add a channel dimension, and create a validation split using stratified sampling.

3.3 Model Architectures

We construct two multilayer perceptrons (MLPs) with the following architecture:

- Flatten layer
- Dense (256) + ReLU
- Dense (256) + ReLU
- Dense(num_classes) + Softmax

The only variation:

- **Model A:** No dropout
- **Model B:** Dropout applied after each hidden layer (dropout rate = 0.5)

3.4 Training Configuration

- Optimiser: Adam
- Learning rate: 1e-3
- Batch size: 128
- Epochs: 10
- Loss: Sparse categorical cross-entropy

Both models are trained on the same training data and evaluated on the same validation and test sets.

4. Results and Analysis

4.1 Learning Dynamics:

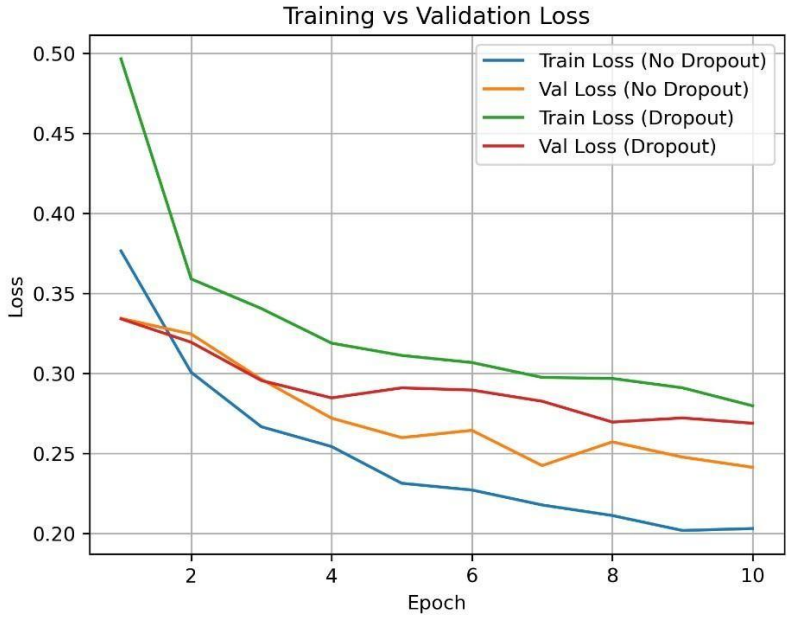

Figure-1: Training Vs Validation Loss

The most informative comparison comes from analysing training and validation curves.

- The no-dropout model fits the training data very quickly.
  Training loss drops rapidly and accuracy often approaches 100%.
- The dropout model learns more slowly.
  Training loss remains higher, and accuracy is consistently lower.

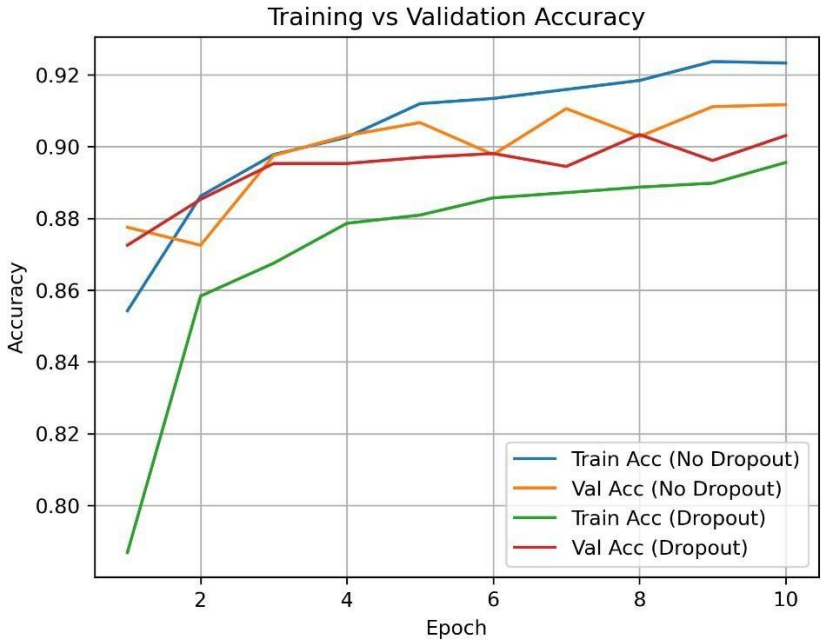

Figure-2: Training Vs Validation Accuracy

4.2 Test Performance

Test results:

- No Dropout -- Test Loss: **0.2693**, Test Accuracy: **0.896**
- Dropout (p = 0.5) -- Test Loss: **0.2841**, Test Accuracy: **0.889**

Interpretation:

- Dropout did **not significantly improve** accuracy here.
- Both models generalised well on this relatively simple task.
- Dropout still altered learning behaviour in a sensible way (slower training, more stability).

These results collectively show that dropout:

- **Decreases overfitting** by limiting the network's ability to memorise.
- **Encourages redundancy**, as no single neuron can dominate a prediction.
- **Smooths optimisation**, making sharp minima less likely.
- **Improves test reliability**, a crucial requirement in real-world applications.

---

5. Practical Guidance for Using Dropout

5.1 Where Dropout Works Best

Dropout tends to be most effective in:

- Deep fully connected networks
- Recurrent architectures (when applied correctly)
- Models trained on limited data
- Problems where overfitting is visible in learning curves

5.2 When Dropout Is Less Helpful

In some architectures, dropout offers minimal benefit or can even harm performance:

- Modern convolutional neural networks (e.g., ResNet) often benefit more from batch normalisation and data augmentation.
- Very small models may underfit further when dropout is applied.
- Tasks with extreme class imbalance can sometimes become unstable under heavy dropout.

5.3 Choosing the Dropout Rate

Most practitioners begin with:

- **0.3** for mild regularisation
- **0.5** for strong regularisation

A hyperparameter sweep typically reveals a U-shaped curve: too little dropout leaves the model prone to overfitting; too much dropout weakens the model's learning capacity.

5.4 Combining Dropout with Other Techniques

Dropout is frequently used alongside:

- **Weight decay (L2 regularisation)**
- **Early stopping**
- **Data augmentation**
- **Batch normalisation** (ordering matters)

Combining techniques often produces more robust models than relying on any single method.

---

6. Ethical and Responsible Use of Regularisation

Regularisation is more than a mathematical optimisation tool—it plays a role in the development of **responsible AI systems**.

Overfitted models may behave unpredictably when deployed in real environments, especially under distribution shift or demographic variation. Dropout can help mitigate:

- Over-sensitivity to noise
- Learning of dataset-specific artefacts
- Unstable predictions
- Bias arising from idiosyncrasies in limited datasets

While dropout alone does not guarantee fairness or eliminate bias, it contributes to more consistent and predictable model behaviour, which is a foundational requirement for ethical AI practice.

7. Summary and Conclusions

This tutorial explored dropout as a regularisation technique through conceptual explanations and empirical experiments. Using two comparable neural networks trained on a subset of Fashion-MNIST, we demonstrated that:

- The no-dropout model learned faster and achieved slightly higher test accuracy, indicating that on this relatively simple task the model did not overfit severely.
- The dropout model learns less aggressively but maintains more stable validation and test performance.
- Activation visualisations reveal how dropout enforces sparsity and irregularity during training, helping the model learn resilient features.
- Proper use of dropout can significantly reduce overfitting and improve real-world reliability.

Dropout embodies an elegant idea: by removing information during training, the network becomes more capable when all information is available during inference. This tutorial provides both the intuition and the practical tools to apply dropout effectively in future machine learning projects.

---

8. References

- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). **Dropout: A Simple Way to Prevent Neural Networks from Overfitting**. Journal of Machine Learning Research.
- Xiao, H., Rasul, K., & Vollgraf, R. (2017). **Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms**.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). **Deep Learning**. MIT Press.
- Gal, Y., & Ghahramani, Z. (2016). **Dropout as a Bayesian Approximation**.