Exploring the Impact of Depth and Width in Multilayer Perceptron Using Fashion-MNIST

By NANA BROWNY
ID:23037587

Github Link: ***https://github.com/23037587/mlp-fashion-mnist-tutorial***

1. Introduction

Neural networks are used in a wide range of machine learning applications, from image recognition to natural language processing. One of the most fundamental neural network architectures is the **Multilayer Perceptron (MLP)**. Although simple compared to modern deep learning models, MLPs are powerful enough to learn complex relationships in data and are an excellent starting point for understanding how neural networks behave.
In this tutorial, we investigate how the **depth** (number of hidden layers) and **width** (number of neurons per layer) of an MLP affect its ability to learn from data. We use the **Fashion-MNIST** dataset, a widely used benchmark that contains grayscale images of clothing items. Our goal is to help you understand how model architecture influences accuracy, overfitting, and generalisation, knowledge that transfers to more complex architectures such as CNNs and transformers.
All experiments are implemented in **PyTorch**, and the full code is available in the accompanying Jupyter Notebook.

2. Background: What Are MLPs, and Why Structure Matters?

A **Multilayer Perceptron** is a *feed-forward neural network* consisting of:

- an **input layer** (flattened pixels in this case),
- one or more **hidden layers**, and

- an **output layer** (10 classes for Fashion-MNIST).

Each hidden layer applies a linear transformation followed by a nonlinear activation function, commonly
ReLU:

$$Output = \sigma(Wx + b)$$

Depth

Depth refers to the number of hidden layers.

- A **shallow** model has one hidden layer.
- A **deep** model has many hidden layers.

Increasing depth allows the network to learn hierarchical features, but can also increase:

- training difficulty,
- risk of vanishing gradients,
- computational cost.

Width

Width refers to how many neurons each hidden layer contains.
Wider layers give the network more representational capacity, allowing it to fit training data more closely.
However, excessive width may lead to:

- overfitting,
- slower training,
- diminishing performance improvements.

Understanding these trade-offs is essential for designing effective neural networks.

3. Dataset: Fashion-MNIST

Fashion-MNIST consists of **70,000 grayscale images (28×28 pixels)** divided into 10 clothing classes, such as:

- T-shirt/top
- Trouser
- Pullover
- Dress
- Sneaker
- Bag

Each image contains one centered item of clothing on a simple background. Compared to the original MNIST digit dataset, Fashion-MNIST is more visually complex and better represents real-world classification challenges, making it ideal for studying model capacity.

4. Implementation Overview

All experiments were implemented in PyTorch, following this structure:

1. **Load and normalize the dataset**
2. **Define a generic MLP class** that accepts a list of hidden layer sizes
3. **Train each model** using Adam optimizer and CrossEntropy loss
4. **Track accuracy** on both training and test sets
5. **Plot curves** (training vs test accuracy)
6. **Generate final evaluation metrics**, including a confusion matrix

The notebook includes fully runnable code so that others may reproduce the results.

5. Experiment 0: Baseline Model

Before comparing architectures, we train a simple baseline model:

- **1 hidden layer**
- **128 neurons**
- **ReLU activation**

Results

The baseline model achieves:

- **Training accuracy:** rises smoothly
- **Final test accuracy:** ~86–88%
- Minimal overfitting

Accuracy curves (train vs test) show the model learning steadily without divergence.

This gives us a reference point for evaluating larger or deeper networks.

6. Experiment 1: How Depth Affects Performance

In this experiment, we fix the width at **128 neurons per layer** and vary depth:

- Model A: 1 layer (128)
- Model B: 2 layers (128, 128)
- Model C: 3 layers (128, 128, 128)
- Model D: 4 layers (128, 128, 128, 128)

Observations

1. **Adding a second layer improves accuracy**
   Test accuracy increases slightly, showing deeper models extract more useful features.
2. **Three or four layers show diminishing returns**
   Accuracy improves only a little, and sometimes becomes unstable depending on initialization.
3. **Training takes longer**
   Deeper networks require more computation.
4. **Slight overfitting appears** in deeper models
   Training accuracy rises faster than test accuracy.

Conclusion on Depth

Increasing depth helps *up to a point*, but very deep MLPs offer limited benefit for small images like Fashion-MNIST.

7. Experiment 2: How Width Affects Performance

Here, depth is fixed at **one hidden layer**, and we vary width:

- 64 neurons
- 128 neurons
- 256 neurons
- 512 neurons

Observations

1. **Going from 64 → 128 → 256 neurons increases accuracy steadily.**
   Wider models learn more features and fit the data better.

2. **512 neurons gives the highest score**, but with more overfitting.
   The training accuracy rises rapidly, while test accuracy increases only slightly.
3. **Computation increases significantly**
   Wider layers multiply the number of parameters.

Conclusion on Width

Width increases capacity, but extremely wide layers lead to diminishing returns and more risk of overfitting.

8. Confusion Matrix of the Best Model

We selected the model with the highest test accuracy (typically 256 or 512 neurons).

The confusion matrix reveals:

- The model performs well on distinct classes such as *Sneakers* and *Bags*.
- Misclassifications occur mostly between visually similar items, such as *Pullover* vs *Coat* or *Shirt* vs *T-shirt/top*.

This confirms that structural model improvements help, but the inherent difficulty of certain classes remains.

9. Summary of Findings

Depth

- Improves performance up to ~2–3 layers
- Beyond that, improvements are small
- More layers increase training time and overfitting risk

Width

- Moderate widening significantly improves accuracy
- Very wide layers provide limited gains
- Wider models overfit more easily

Overall

The ideal MLP for Fashion-MNIST strikes a balance:

1–2 hidden layers with 128–256 neurons each.

Modern architectures (CNNs) would perform better on image data, but MLPs remain a valuable teaching tool for understanding network capacity.

10. How You Can Apply This

Understanding the effect of depth and width helps when designing:

- neural networks for tabular data
- small image classification projects
- rapid prototypes
- educational models for coursework
- experiments exploring model capacity or overfitting

When building your own models:

- Start small (shallow & narrow).
- Increase width before depth.
- Increase complexity only when the baseline is insufficient.
- Track train vs test accuracy to detect overfitting.

These principles generalize well to more complex deep-learning models.

11. References

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.
2. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning representations by back-propagating errors.* Nature.
3. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-Based Learning Applied to Document Recognition.* Proceedings of the IEEE.
4. Xiao, H., Rasul, K., & Vollgraf, R. (2017). *Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms.*
5. PyTorch (2024). *torch.nn Documentation*.
6. Chollet, F. (2018). *Deep Learning with Python.* Manning.