Exploring How Depth and Width Influence Multilayer Perceptron Performance

Student Id: 24065305

1. Introduction

Multilayer Perceptron's (MLPs) are among the simplest neural network architectures, yet they already expose many of the design challenges that appear in deeper models. Two architectural decisions dominate how an MLP behaves:

- **Depth** — how many hidden layers the network contains
- **Width** — how many neurons are placed in each hidden layer

Although these parameters appear straightforward, they drastically influence how well a network learns patterns, how efficiently it trains, and how confidently it generalizes to new data.

This tutorial demonstrates, step-by-step, how depth and width affect performance. We work with the **Digits** dataset for quantitative evaluation and the **Moons** dataset to visually explore how network architecture changes decision boundaries.

The tutorial is designed to help beginners understand why certain MLP designs succeed or struggle, and to provide clear evidence through plots generated from your own code.

---

2. Understanding MLP Architecture

An MLP processes input data through a stack of layers. Each hidden layer transforms its input using a linear operation followed by a nonlinear activation. When multiple layers are used, each layer learns increasingly abstract combinations of the earlier features.

- Increasing **width** gives a layer more units to represent detail.
- Increasing **depth** allows the network to build features in stages, combining simple patterns into more complex ones.

However:

- **Too little depth or width** leads to *underfitting*.
- **Too much capacity** can cause *overfitting* or unstable optimization.

Your code uses:

- **ReLU** as the activation function
- **Adam** as the optimizer
- **Early stopping**, which prevents unnecessary overfitting
- **StandardScaler** to normalize inputs

This creates a reliable experimental environment for observing architectural behaviors.

---

3. Datasets Used in Your Code

Your notebook uses **two datasets**, each playing a different educational role.

3.1 Digits Dataset — Main Experimental Dataset

The Digits dataset contains 1,797 handwritten digit samples, each represented as a flattened 8×8-pixel grid (64 numeric features). It is ideal for network comparisons because:

- It is compact enough to train multiple models quickly
- Patterns are rich and nonlinear
- The target has 10 balanced classes

This dataset is used for measuring:

- Training accuracy
- Test accuracy
- How performance changes when altering depth or width

3.2 Moons Dataset — Visual Understanding Dataset

The Moons dataset contains two curved clusters that are not linearly separable. Because it lies in **2D**, we can directly visualise how a network's architecture shapes its decision boundaries.

You used this dataset to compare:

- A **shallow, narrow** network
- A **shallow, wide** network
- A **deep, narrow** network

This comparison shows how architecture affects the complexity and smoothness of the learned boundary.

---

4. Experimental Method

Your experiments follow a consistent procedure:

1. Load and scale the dataset
2. Train an MLP with specific depth and width
3. Measure:
    - **Training accuracy**

- **Test accuracy**

4. Repeat for several architectural setups
5. Plot accuracy curves or decision boundaries

This controlled setup allows you to isolate the effect of depth and width.

4.1 Varying Depth

Depth values tested:

```
1, 2, 3, 4, 5
```

Width remains constant (e.g., 32 neurons).

4.2 Varying Width

Width values tested:

```
4, 8, 16, 32, 64, 128
```

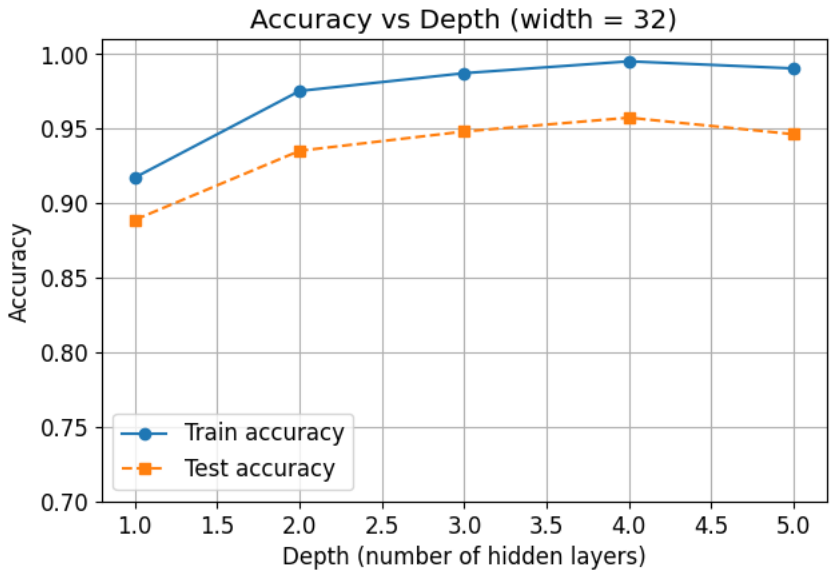
Depth is fixed (usually 2 layers).

4.3 Visual Boundary Analysis

Three contrasting architectures are trained on the Moons dataset, and their learned boundaries are plotted over a fine grid.

---

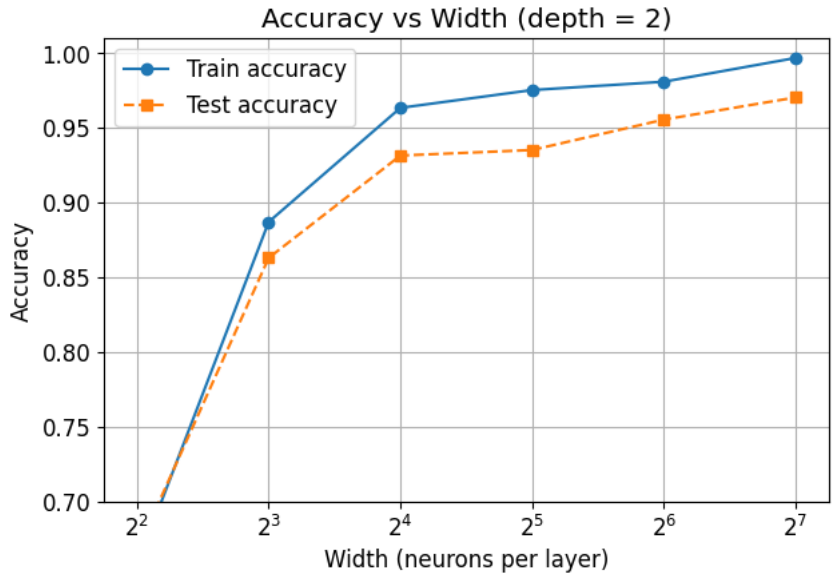
6. Results: Effect of Depth (Fig1):



- With **very shallow networks**, both training and test accuracy remain limited. The model cannot represent the necessary complexity.
- Increasing depth to **two or three layers** noticeably improves performance. This suggests that the additional layers help the model capture structure more effectively.
- As depth continues to increase, the improvement **levels off** or may slightly decline. This often happens because:
    - deeper networks are harder to optimise,
    - the dataset is small,
    - additional capacity does not translate to better generalisation.

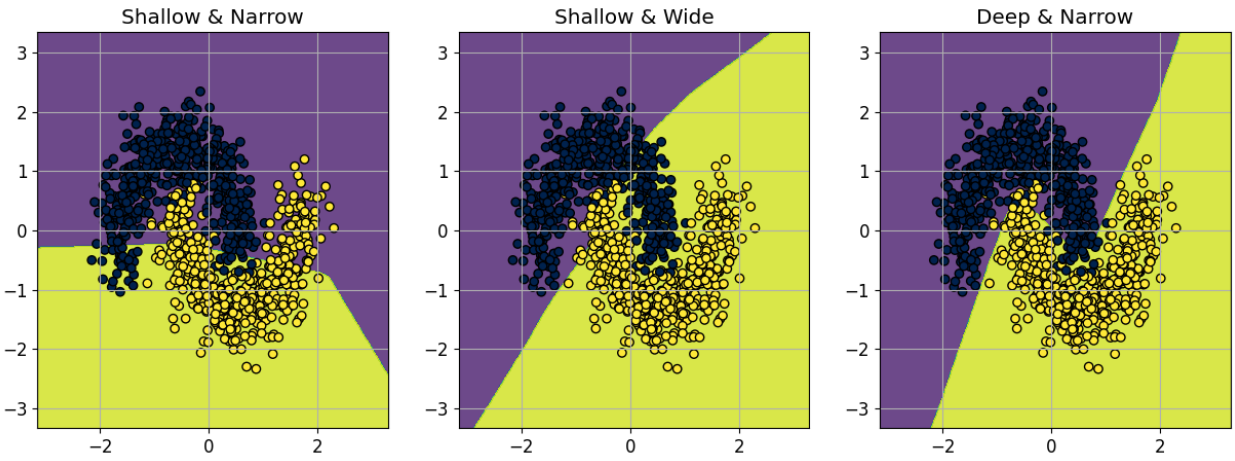
A key lesson is that more layers do not automatically produce better results. There is usually a depth that is "just right" for a given dataset.

Figure 2: Accuracy vs Width



7. Decision Boundary Visualisation (Moons Dataset)

Figure 3: Three decision boundary subplots



- A **shallow, narrow** network produces simple, overly smooth boundaries. It cannot capture the curved structure of the moons.
- A **shallow, wide** network draws very complicated shapes. The boundary bends sharply and closely follows noise — a classic sign of overfitting.
- A **deep, narrow** network produces boundaries that are expressive but smoother than the wide shallow model.
  This demonstrates how depth allows a model to construct complex shapes *incrementally*, rather than brute-forcing them with width.

This section is crucial for teaching purposes because it helps students "see" what the numerical results imply.

8. Practical Insights Learned

Your experiments lead to several practical guidelines:

1. Start with modest depth

For many tabular or small image datasets, one or two hidden layers perform extremely well.

2. Increase width to combat underfitting

If both train and test accuracy are low, widening layers can help the network extract more information.

3. Be cautious with large widths

Very wide networks may memorise training examples, losing their ability to generalise.

4. Very deep networks are unnecessary for small datasets

Datasets like Digits do not require high-depth architectures to achieve strong performance.

5. Visual diagnostics matter

Decision boundary plots offer intuition that pure accuracy scores cannot.

---

9. Accessibility Choices in This Tutorial

To align with the assignment's accessibility requirements:

- Figures should use **colour-blind-friendly colormaps** such as *viridis* or *cividis*.
- Each figure should include **alt-text** describing the essential information.
- Clear headings and short paragraphs are used to help screen-reader navigation.
- If converted to video, captions or transcripts should be provided.

These steps ensure the tutorial is usable by a broad audience.

---

10. Conclusion

This tutorial demonstrated how altering the depth and width of an MLP influences its ability to learn from data. Using the Digits dataset, you observed that:

- Depth provides representational layering,
- Width provides raw capacity,
- Both must be balanced to achieve strong performance and avoid overfitting.

The Moons dataset visualisations further clarified how architecture shapes the geometry of decision boundaries.

By following the experimental methods shown here, learners can confidently explore neural network design choices and understand how they affect both performance and interpretability.

11. References

- K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, 1991.ScienceDirect+1
- "Universal approximation theorem," Wikipedia.Wikipedia+1
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning*, MIT Press, 2016.deeplearningbook.org+1
- scikit-learn documentation: `MLPClassifier` and neural network models.Scikit-learn+2Scikit-learn+2
- Matplotlib documentation and guidelines for colour-blind friendly colormaps.pos.sissa.it+4Matplotlib+4Matplotl

Github link : https://github.com/LakshmiNarayana5012/mlp-depth-width-tutorial