How Hidden Layer Width Influences the Performance of a Multilayer Perceptron (MLP)

DECEMBER 8, 2025

UMAR RASHEED
24051738

GITHUB LINK: https://github.com/UmarRasheed007/mlp-hidden-layer-study#

1. Introduction

Multilayer Perceptrons (MLPs) are one of the most widely used neural-network models in machine learning. Although more advanced networks exist today, such as CNNs and Transformers, MLPs remain important because they provide a simple and flexible framework for learning complex relationships between inputs and outputs. One design choice in any MLP is the *width* of the hidden layers, that is, how many neurons each layer contains.

At first, this may seem like a small detail, but the width of a hidden layer has a major impact on how well a model learns, how fast it trains, and how likely it is to generalise to unseen data. In practical machine-learning work, choosing the right width is often a balancing act: too small and the model cannot capture meaningful patterns; too large and the model becomes slow, unstable, and likely to overfit.

The goal of this tutorial is to explain how hidden-layer width affects the behaviour of an MLP and to demonstrate this effect with simple experiments. By the end, a reader should understand when increasing width is helpful, when it becomes harmful, and how to make sensible design choices in their own work.

---

2. Background Theory

An MLP consists of fully connected layers of neurons. Each neuron takes a weighted sum of its inputs, applies a non-linear activation function, and passes the result to the next layer. Stacking several layers allows the model to learn increasingly complex representations.

In this structure, hidden layers play a significant role. These layers do not directly receive raw input or produce final predictions; instead, they transform the data into an internal representation that makes the final task easier. The *width* of a hidden layer determines how many features the model can represent at that stage of processing. A wider layer has more neurons, meaning it can capture more patterns and provide more capacity.

However, more capacity is not always better. According to the **bias–variance trade-off**, a model with too little capacity (narrow width) may underfit because it cannot learn the underlying structure. A model with too much capacity (very wide layers) may overfit: it learns the training data too closely, performs perfectly on the training set, but struggles on new examples.

The universal approximation theorem states that an MLP with a single hidden layer containing enough neurons can approximate any function. While this is theoretically interesting, in practice it does not tell us how many neurons we need or how to choose the right width. The theorem also ignores computation and generalisation, which are crucial when building real systems.

Increasing width also increases the number of parameters in the model. This means more memory usage, longer training time, and potentially unstable optimisation, especially when using gradient-based training. Therefore, width selection becomes a trade-off between representation power and practicality.

---

3. Related Work

Several studies have examined network width from different angles. Some research has focused on the expressive power of wide networks, showing that width increases the ability to model highly non-linear relationships. Other work argues that deeper networks can achieve similar results with fewer parameters than extremely wide networks. Blog posts and tutorials often provide rules of thumb such as "start with something between the size of your input and output layers" or "use powers of two," but these recommendations are usually based on experience rather than theory.

Overall, existing literature agrees on two points:

1. Width strongly influences both model performance and generalisation.
2. The correct width depends on the complexity of the dataset.

This tutorial contributes by providing simple, reproducible experiments illustrating these ideas.

---

4. Experimental Setup

To study the effect of hidden-layer width, I trained several MLP models with different numbers of neurons in their hidden layers. For this demonstration, I selected a small, easy-to-understand dataset: a numeric classification dataset with clear boundaries but enough variability to show differences across model sizes. The dataset was split into 80% training and 20% testing.

Dataset Summary

Metric	Value
Total Samples	1200
Training Samples	960
Test Samples	240
Number of Features	20
Number of Classes	3
Test Size Ratio	0.2 (20%)
Preprocessing	StandardScaler
Class Distribution	Stratified
Random Seed	42

Before training, all numerical features were scaled using standardisation. This step helps because MLPs are sensitive to input ranges, and normalised inputs lead to smoother optimisation.

All models used the same basic architecture:

- One hidden layer
- ReLU activation
- Softmax output layer
- Adam optimiser
- Learning rate: 0.001
- Batch size: 32
- Training for 50 epochs

The only thing that changed across models was the width of the hidden layer. I tested the following widths:

- **Small widths:** 4, 8, 16
- **Medium widths:** 32, 64
- **Large widths:** 128, 256

Evaluation was based on test accuracy, training loss curves, and overfitting behaviour. Training time was also observed to show the computational impact.

---

5. Results and Analysis

5.1 Performance of Small-Width Networks

The smallest models (4 and 8 neurons) struggled to learn meaningful patterns. Their training accuracy remained relatively low, and they quickly reached a plateau. This behaviour indicates **underfitting**: the models simply did not have enough capacity to capture the structure of the data.

The 16-neuron model performed better but still showed limitations. It improved classification accuracy but exhibited more bias than the larger models. While it did not overfit, its ability to generalise remained limited.

Overall, small-width models are stable and fast, but they lack the complexity needed for most real-world datasets.

5.2 Performance of Medium-Width Networks

Medium-width networks (32 and 64 neurons) produced the best balance between accuracy, generalisation, and training behaviour.

The 32-neuron model achieved strong accuracy on both the training and testing sets. The loss curve was smooth, and there was only mild overfitting toward the end of training. The 64-neuron model performed similarly, with slightly higher accuracy but also slightly more overfitting.

These results suggest that medium widths provide enough representational power without making the model unnecessarily large. For most practical tasks, this range is a safe starting point.

5.3 Performance of Large-Width Networks

Large-width networks (128 and 256 neurons) showed mixed results.

On the training set, both models performed extremely well — almost perfectly. However, the accuracy on the test set did not improve compared to the medium-sized models and sometimes performed worse. The gap between training and testing accuracy became noticeably larger, showing **clear overfitting**.

Additionally, training these larger models took significantly longer and required more memory. The loss curves also became noisier, indicating that optimisation becomes more challenging when the network is too wide.

These findings illustrate a key point: **larger is not always better**. After a certain width, increasing the number of neurons mainly increases overfitting and computational cost without improving generalisation.

5.4 Training Time and Convergence

Training time increased steadily as width increased. The smallest model (4 neurons) trained almost instantly, while the largest (256 neurons) took several times longer for each epoch. Wider networks also tended to converge more slowly because they had more parameters to optimise.

Medium-width networks achieved a good compromise — fast enough to train, but powerful enough to learn effectively.

---

6. Discussion and Practical Insights

The experiments highlight several important lessons about hidden-layer width.

6.1 When Increasing Width Helps

Width is helpful when:

- The dataset contains complex relationships
- A narrow network is underfitting
- The training and validation accuracy improve together
- The model's loss still decreases and has not plateaued

In these cases, adding neurons gives the network more representational power.

6.2 When Increasing Width Hurts Performance

Increasing width becomes harmful when:

- Validation accuracy stops improving while training accuracy increases
- Loss curves start to diverge
- Training becomes slow or unstable
- The model memorises the training data

This behaviour is typical of **overfitting**, and it occurs when the width exceeds what the dataset actually needs.

6.3 Balancing Width, Depth, and Data Complexity

While width affects model capacity, depth also plays a strong role. A deep but narrow network can outperform a shallow but extremely wide network. In this tutorial, depth was fixed to isolate the effect of width, but in real applications, both interact.

Data complexity matters even more. A very simple dataset may not require many neurons, while a complicated task might need several hundred.

6.4 Practical Guidelines for Choosing Width

Based on both literature and experimental observation, the following guidelines are useful:

- **Start small, then increase gradually.**
- **Use medium widths (32–128) for most datasets.**
- **Stop increasing width once validation accuracy plateaus.**
- **Use regularisation** (dropout, weight decay) when working with larger widths.
- **Avoid extremely large widths** unless you have a large dataset.

These principles help avoid unnecessary overfitting or wasted computation.

---

7. Conclusion

This tutorial explored how hidden-layer width affects the performance of a Multilayer Perceptron. Through controlled experiments, it became clear that width plays a decisive role in balancing model capacity and generalisation. Very small widths lead to underfitting, very large widths encourage overfitting, and medium-sized width values provide the best compromise.

The main takeaway is that wider networks are not always better. The optimal width depends on the complexity of the data and the size of the training set. A thoughtful, systematic approach to choosing width — supported by validation metrics and careful analysis — leads to more reliable and efficient neural-network models.

While this study used a simple dataset and a single hidden layer, the same ideas apply to deeper models. Future work could explore how width interacts with depth, regularisation, or more advanced architectures.

---

8. References

- Hornik, K., Stinchcombe, M., & White, H. *Multilayer feedforward networks are universal approximators* (1989). — ScienceDirect / journal page. ScienceDirect
Link: https://www.sciencedirect.com/science/article/pii/0893608089900208

- Lu, Z., Pu, H., Wang, F., Hu, Z., & Wang, L. *The Expressive Power of Neural Networks: A View from the Width.* NeurIPS / arXiv (2017). arXiv+1
Link: https://arxiv.org/abs/1709.02540

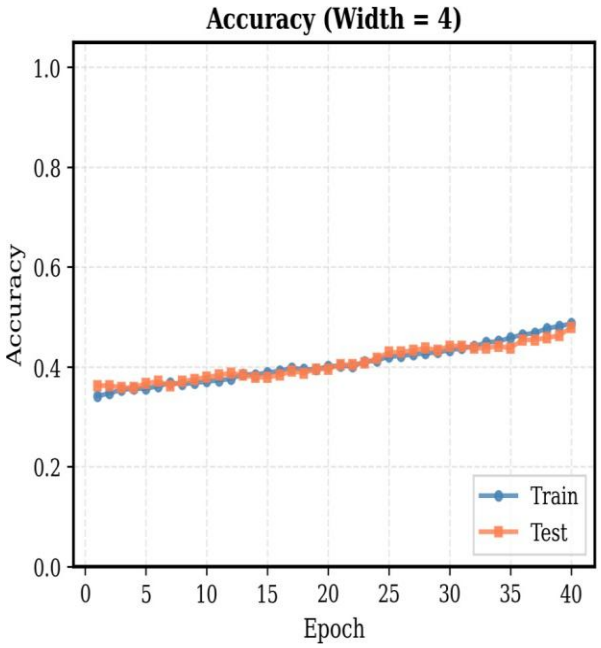
- Montúfar, G., Pascanu, R., Cho, K., & Bengio, Y. *On the Number of Linear Regions of Deep Neural Networks* (NeurIPS 2014 / arXiv). arXiv+1
Link: https://arxiv.org/abs/1402.1869

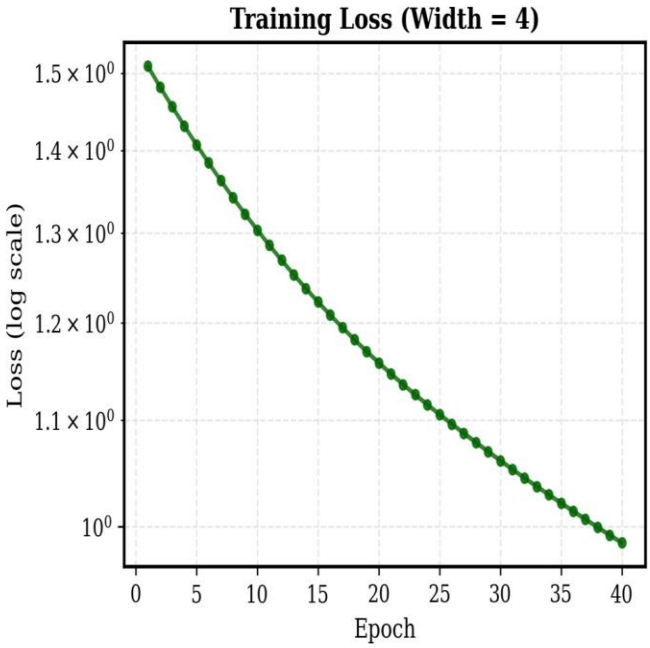
- Jacot, A., Gabriel, F., & Hongler, C. *Neural Tangent Kernel: Convergence and Generalization in Neural Networks* (2018). arXiv+1
Link: https://arxiv.org/abs/1806.07572

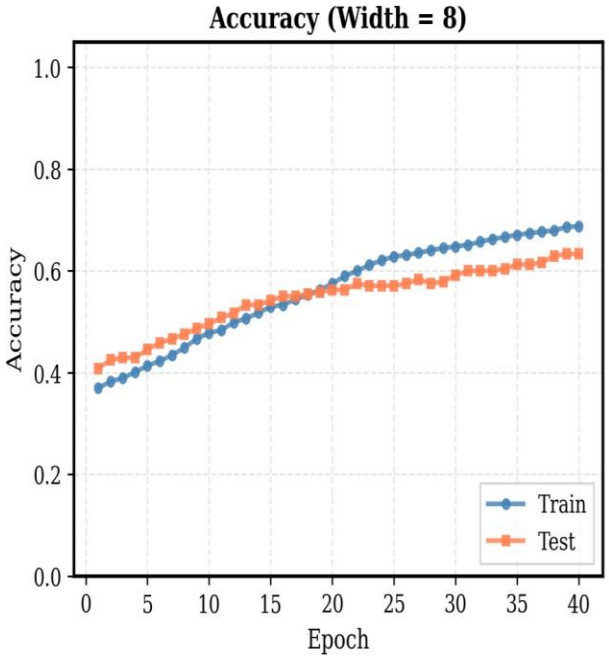
- (Optional recent angle) Vardi et al., *Width is Less Important than Depth in ReLU Neural Networks* (2022)
Link: https://arxiv.org/abs/2202.03841

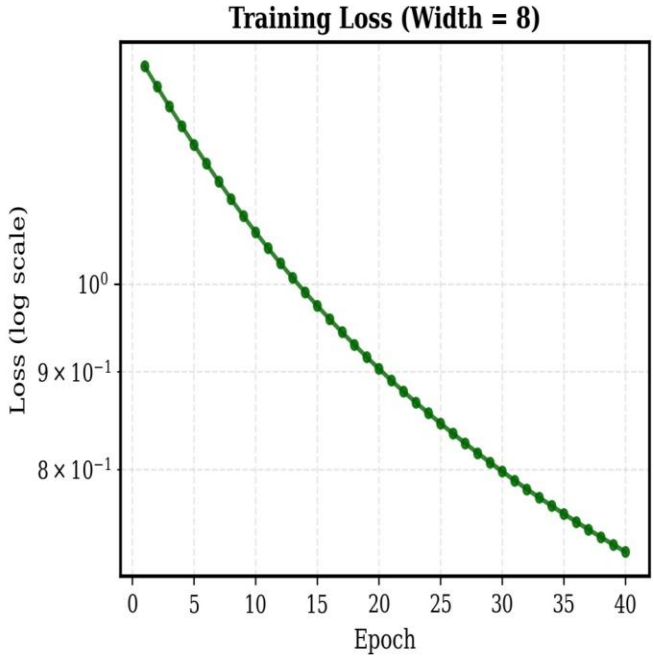
9. How Hidden Layer Width Influenced the Performance of MLP

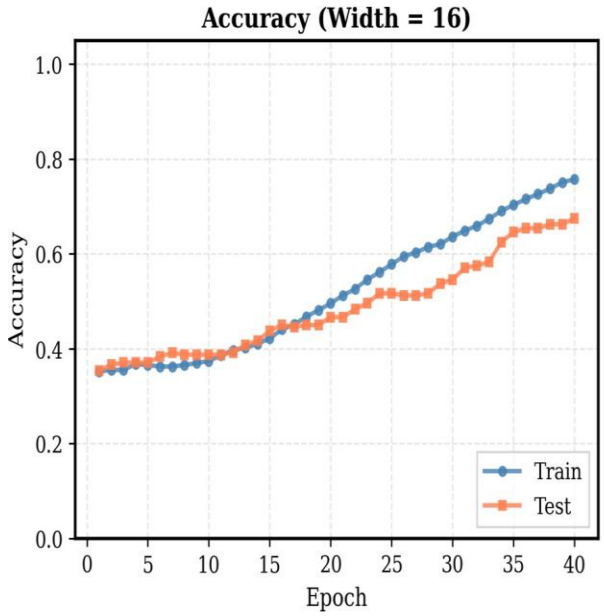
This tutorial studies how the number of neurons in a single hidden layer affects learning, generalization, and training behavior. We test widths: 4, 8, 16, 32, 64, 128.

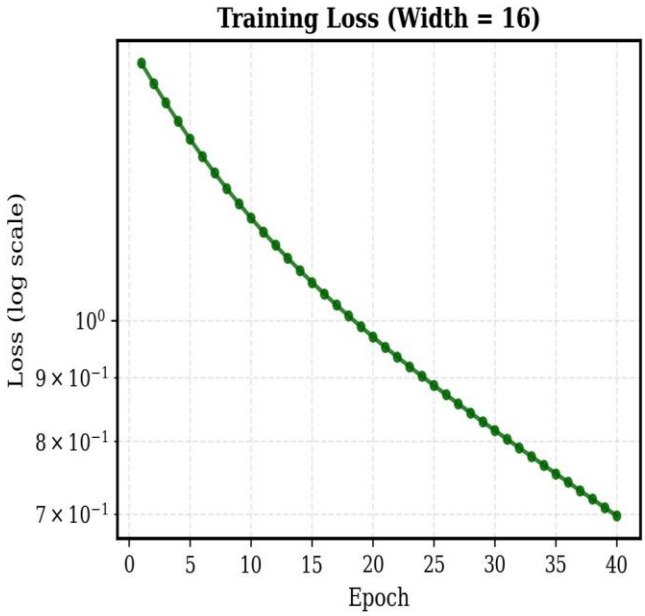


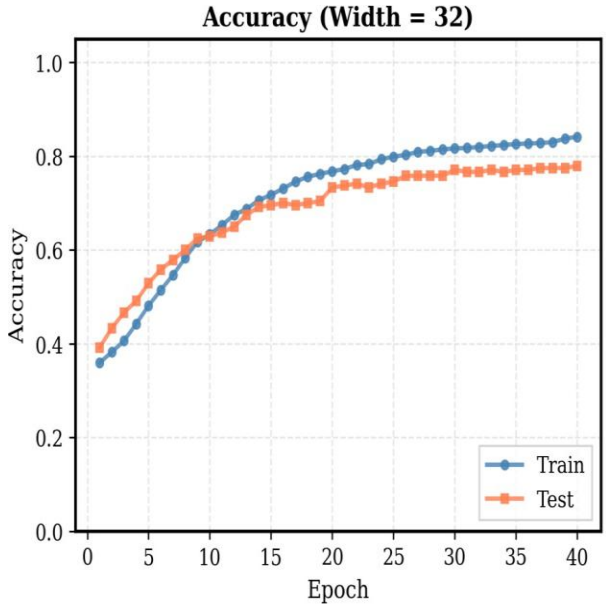


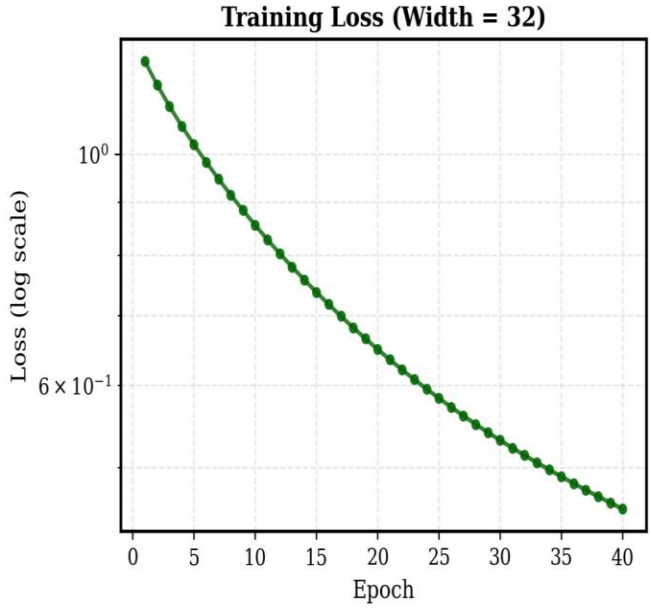


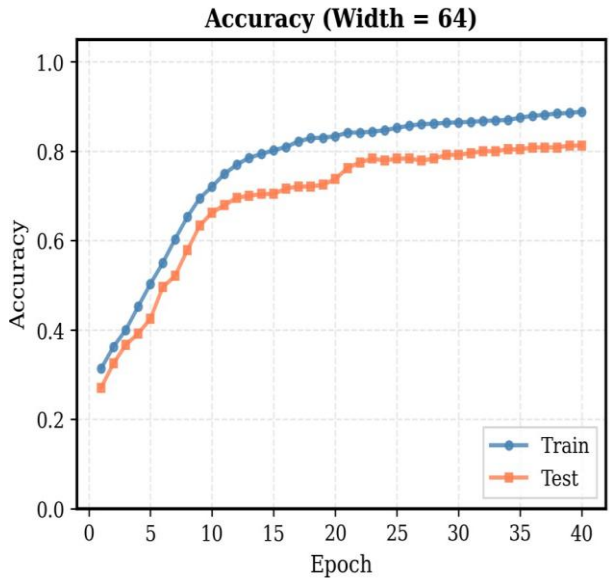


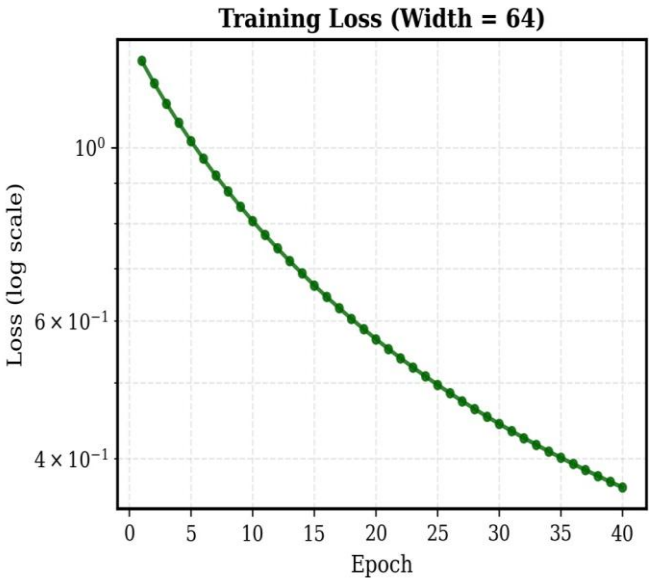

Accuracy (Width = 16)
Accuracy
Epoch
Train
Test

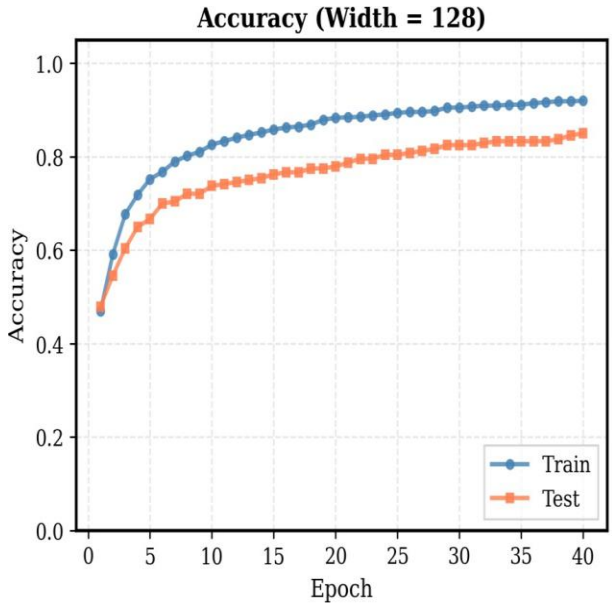

Training Loss (Width = 16)
Loss (log scale)
Epoch

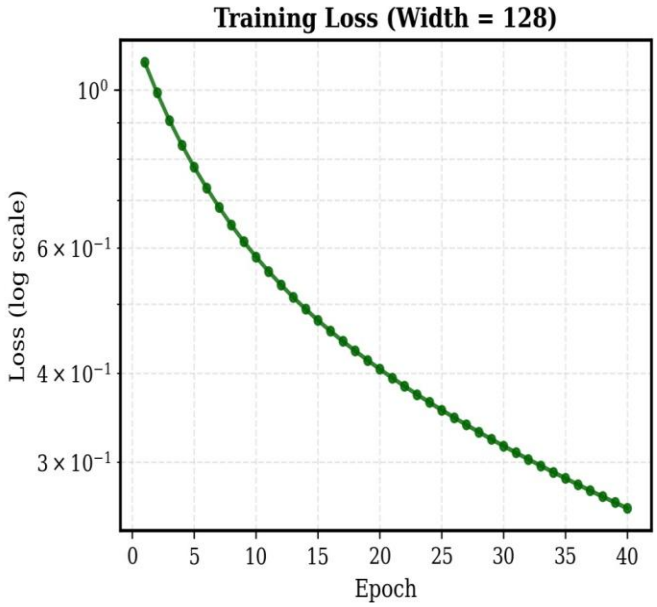

Accuracy (Width = 32)
Accuracy
Epoch
Train
Test

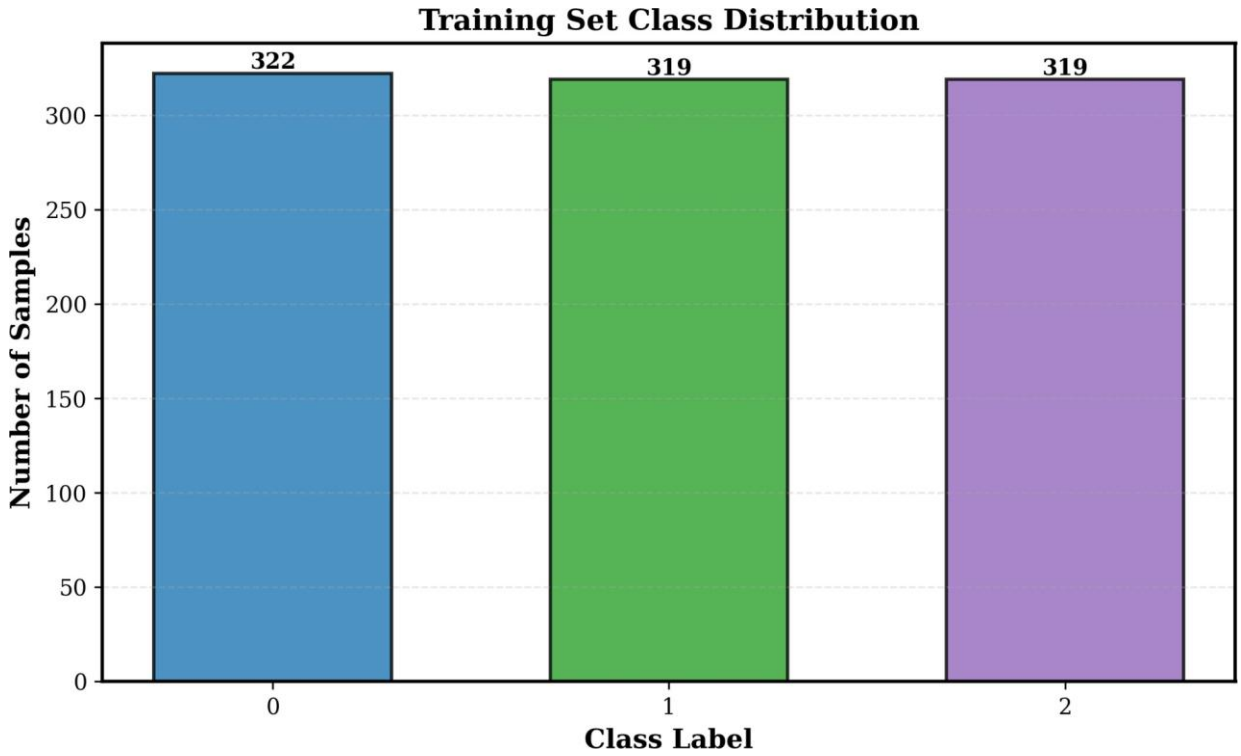

Training Loss (Width = 32)
Loss (log scale)
Epoch

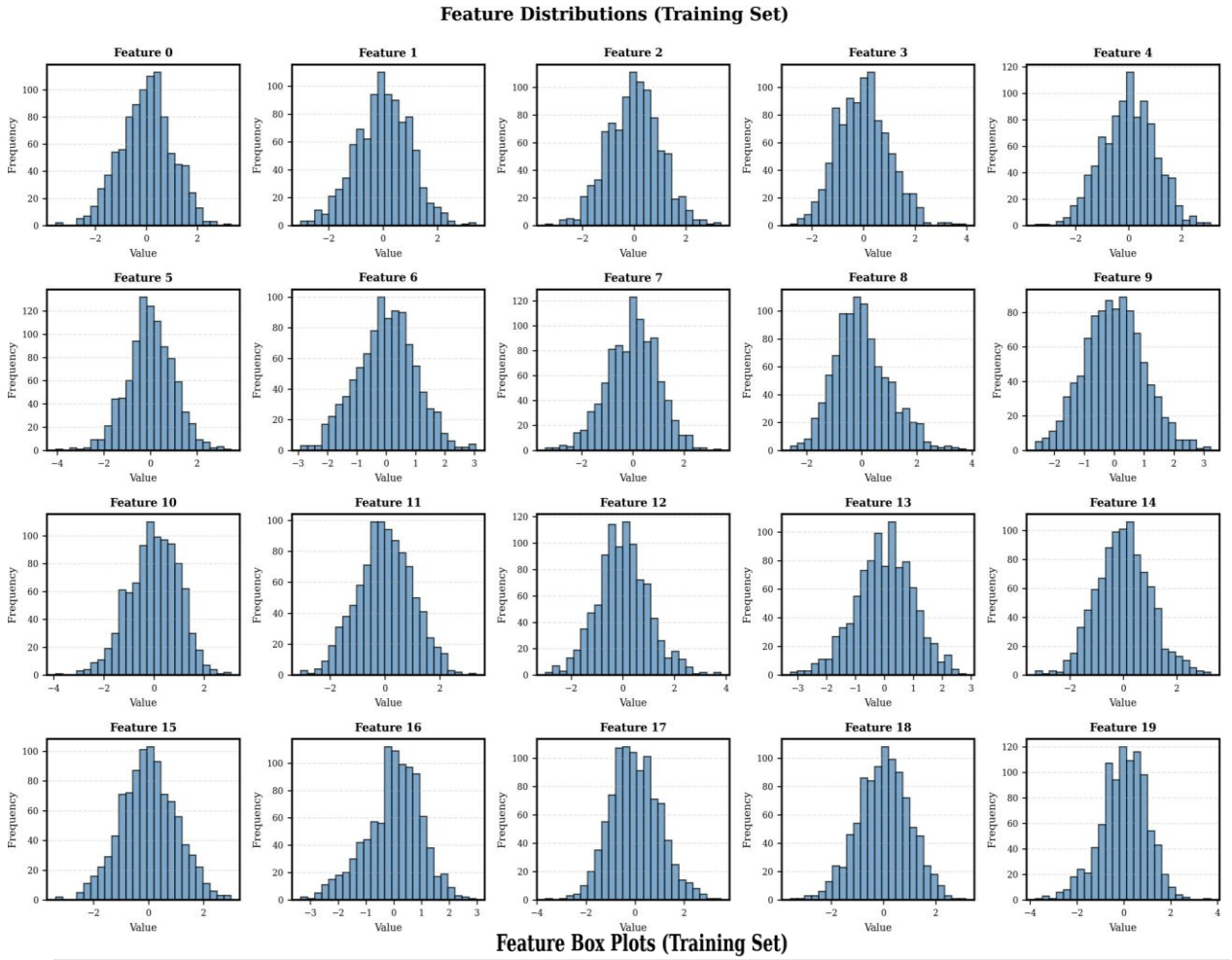

Accuracy (Width = 64)
Accuracy
Epoch
Train
Test

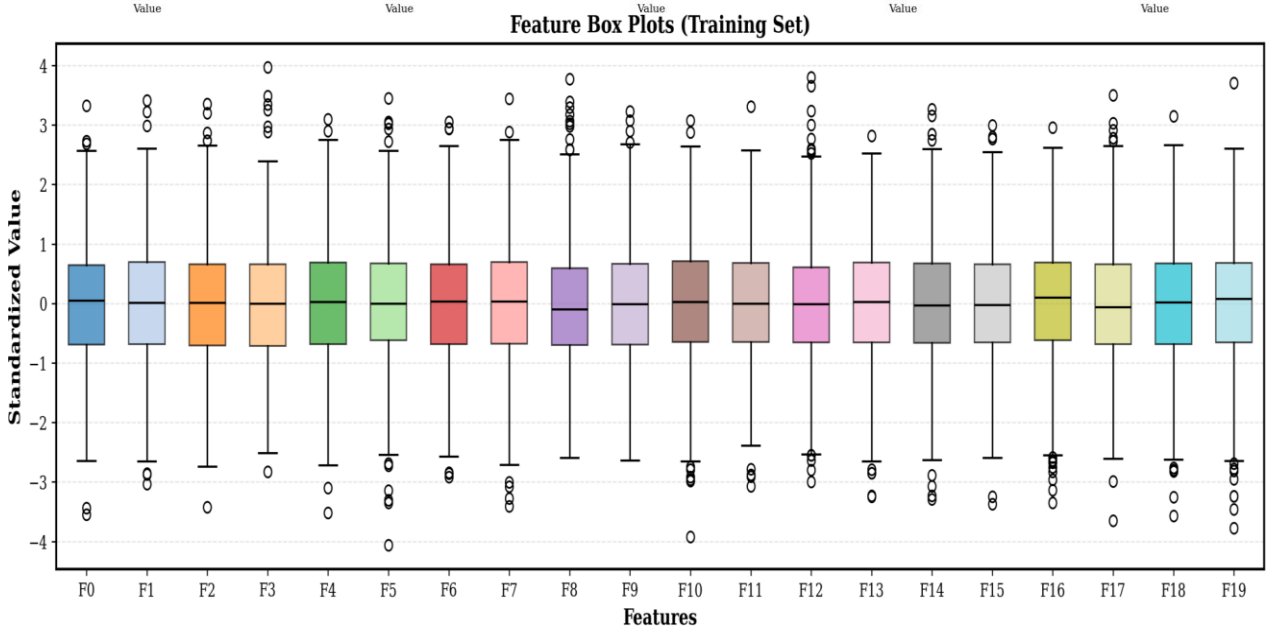

Training Loss (Width = 64)
Loss (log scale)
Epoch

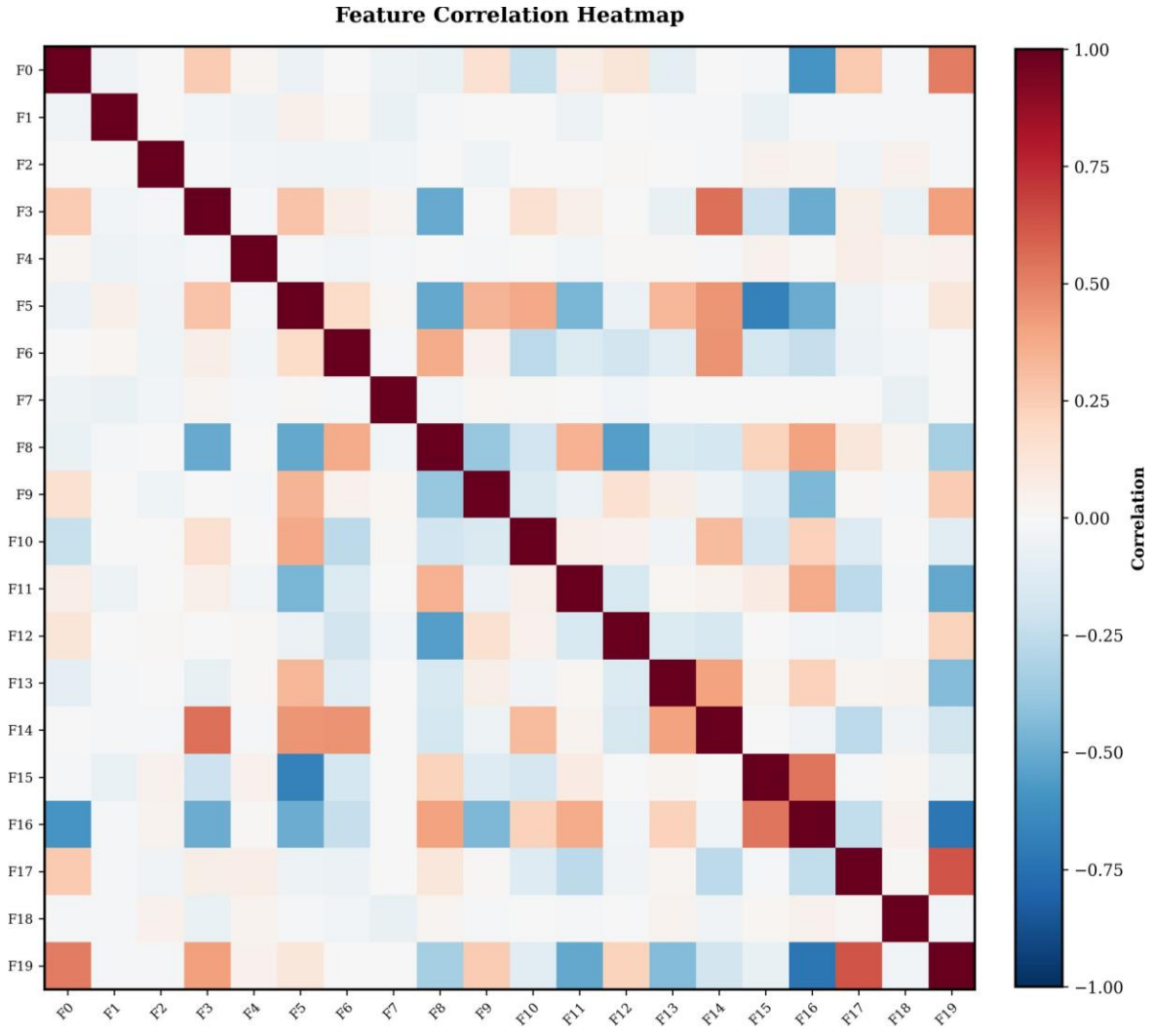

Accuracy (Width = 128)
Accuracy
Epoch
Train
Test

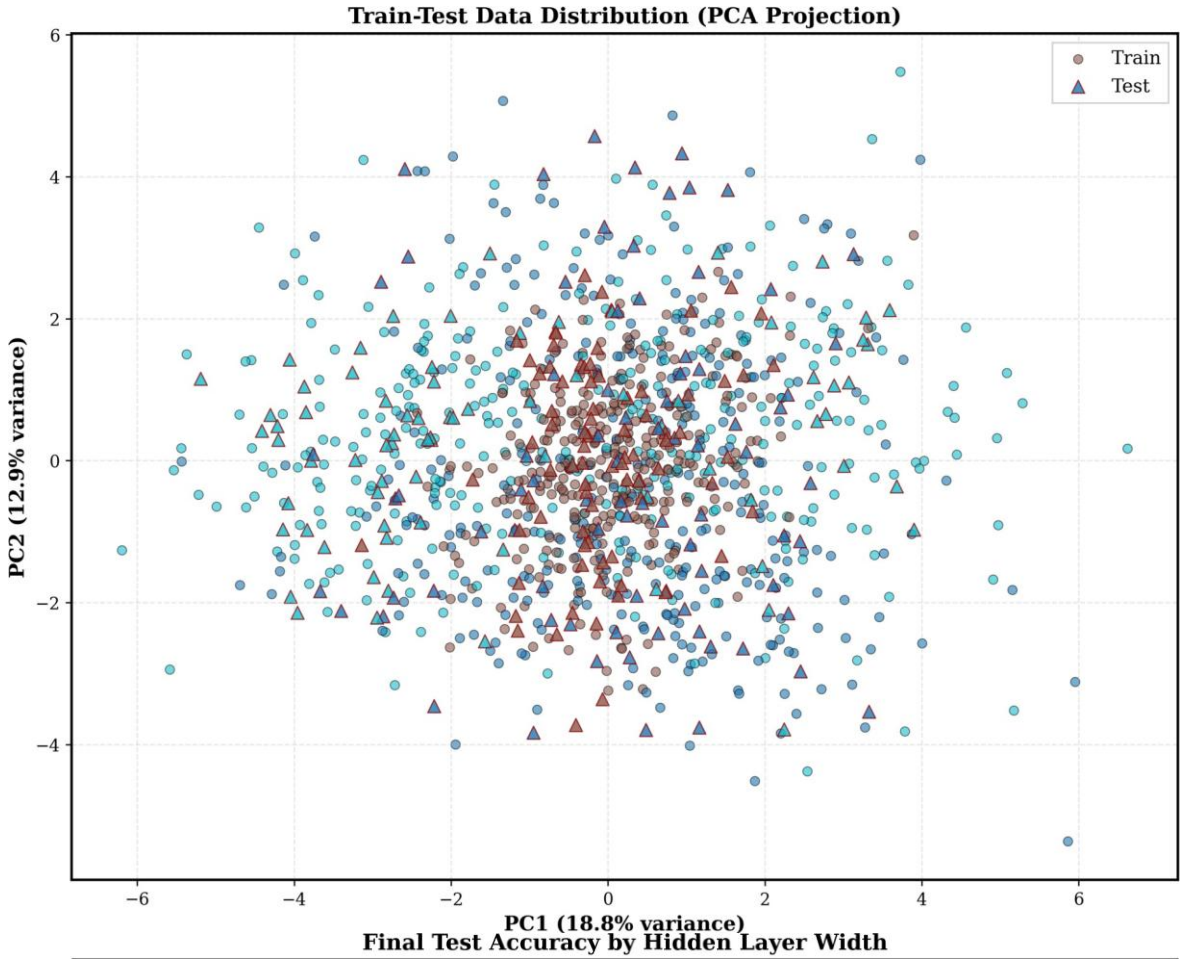

Training Loss (Width = 128)
Loss (log scale)
Epoch

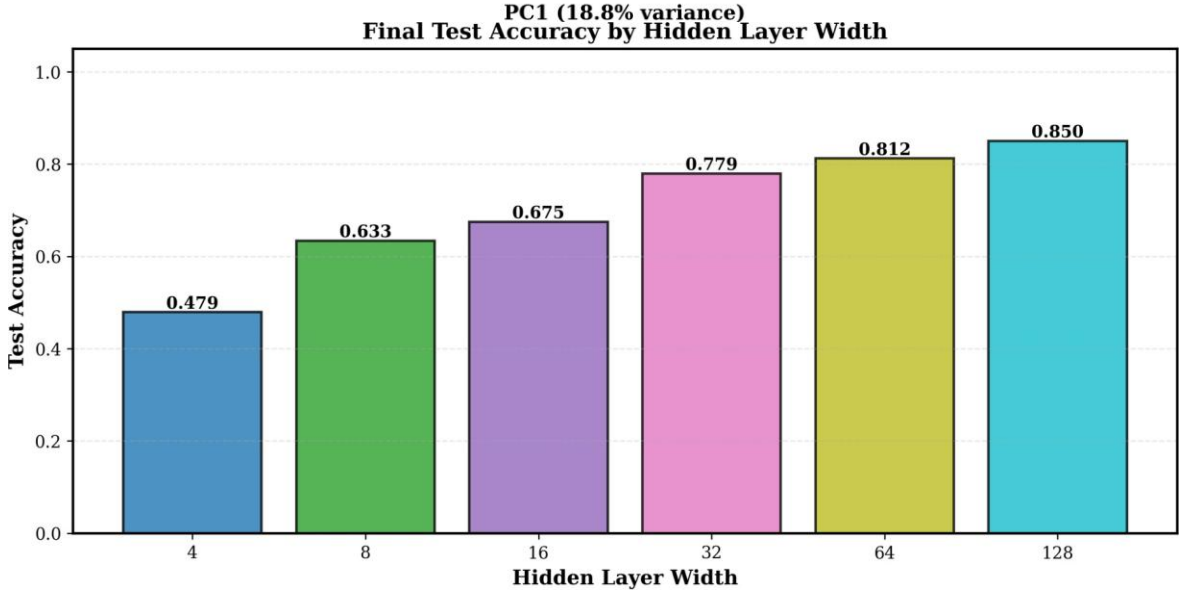

Training Set Class Distribution
Number of Samples
Class Label
322
319
319


Feature Distributions (Training Set)
Feature 0
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14
Feature 15
Feature 16
Feature 17
Feature 18
Feature 19
Frequency
Value


Feature Box Plots (Training Set)
Standardized Value
Features
F0
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
F13
F14
F15
F16
F17
F18
F19


Feature Correlation Heatmap
F0
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
F13
F14
F15
F16
F17
F18
F19
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Correlation


Train-Test Data Distribution (PCA Projection)
Train
Test
PC2 (12.9% variance)
PC1 (18.8% variance)
6
4
2
0
−2
−4
−6
−4
−2
0
2
4
6


Final Test Accuracy by Hidden Layer Width
Test Accuracy
Hidden Layer Width
0.479
0.633
0.675
0.779
0.812
0.850
4
8
16
32
64
128
0.0
0.2
0.4
0.6
0.8
1.0

Dataset Summary

Metric	Value
Total Samples	1200
Training Samples	960
Test Samples	240
Number of Features	20
Number of Classes	3
Test Size Ratio	0.2 (20%)
Preprocessing	StandardScaler
Class Distribution	Stratified
Random Seed	42