Student Name: Indhu Reddy Kottalam Raveendra Reddy
Student Id: 24096112
Github: https://github.com/indhu0204/mlp-depth-width-breast-cancer

Exploring Multilayer Perceptron Depth and Width on Breast Cancer Diagnosis

Introduction

Multilayer Perceptrons (MLPs) are among the most fundamental neural network architectures and are widely used for tabular classification tasks, including medical diagnosis. A key question for practitioners is: how do design choices—specifically the **depth** (number of hidden layers) and **width** (number of neurons per layer)—affect model performance? This tutorial investigates this question using the Breast Cancer Wisconsin (Diagnostic) dataset, a well-known benchmark for machine learning in healthcare.

The motivation is practical: on small-to-medium tabular datasets, more complex models do not always generalise better. Understanding when to use shallow vs deep networks, and narrow vs wide networks, is essential for building efficient, interpretable models in real applications.

Dataset and preprocessing

The Breast Cancer Wisconsin (Diagnostic) dataset contains 569 samples of cell nuclei measurements extracted from histopathology images, with 30 numeric features (e.g. mean radius, texture, perimeter) and a binary target: malignant (0) or benign (1). The dataset is imbalanced: 212 malignant and 357 benign cases.

We split the data into training (60%), validation (20%), and test (20%) sets using stratified random sampling to preserve class ratios. All features are scaled to zero mean and unit variance using StandardScaler, fitted only on the training set to prevent data leakage. A fixed random seed ensures reproducibility.

Multilayer Perceptron architecture

An MLP consists of an input layer, one or more hidden layers with non-linear activation functions, and an output layer. For binary classification, we use:

- **Input layer:** 30 features (the dataset dimensionality).
- **Hidden layers:** variable number and size, with ReLU activation (max(0, x)), which introduces non-linearity.

- **Output layer:** single neuron with sigmoid activation for probability output.

The network is trained using the Adam optimiser with binary cross-entropy loss and early stopping on the validation set to prevent overfitting.

Depth refers to the number of hidden layers; **width** refers to the number of neurons per hidden layer. Both parameters control the model's capacity: the ability to learn complex, non-linear decision boundaries. However, higher capacity does not always translate to better generalisation, especially on small datasets.

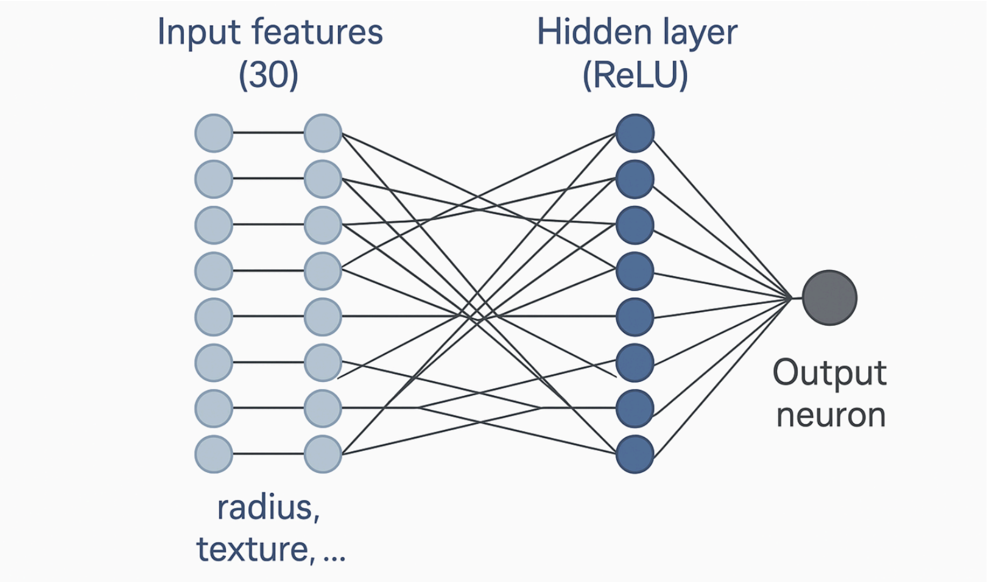


Figure 01
Multilayer Perceptron architecture used in this tutorial, with 30 input features, one hidden ReLU layer, and a single sigmoid output neuron for binary classification.

Schematic MLP diagram showing 30 input features fully connected to a hidden ReLU layer, which is fully connected to one output neuron predicting malignancy probability.

Overview of Multilayer Perceptrons

A Multilayer Perceptron (MLP) is a feed-forward neural network composed of an input layer, one or more hidden layers, and an output layer. Each layer contains a set of neurons, and every neuron computes a weighted sum of its inputs followed by a non-linear activation function such as ReLU. By stacking layers, MLPs can learn complex non-linear decision boundaries that simpler linear models cannot represent.
In this tutorial, the MLP takes 30 numeric features describing cell nuclei as input and outputs the probability that a tumour is malignant. The input layer has 30 units (one per feature). The hidden layers are the part we experiment with: their depth (how many layers) and width (how many neurons in each) determine the model's capacity. A deeper or wider network can, in principle, represent more complex patterns, but it also becomes easier to overfit small datasets such as Breast Cancer Wisconsin

(Diagnostic).

The output layer contains a single neuron with a sigmoid activation, producing a value between 0 and 1 that can be interpreted as the probability of malignancy. During training, the model adjusts its weights to minimise binary cross-entropy loss using the Adam optimiser, while early stopping monitors validation performance to avoid overfitting.

Conceptually, the MLP used here can be visualised as a series of fully connected layers:

- A 30-node input layer receiving the scaled features.
- One or more hidden layers (e.g. 32 neurons each) with ReLU activations.
- A final sigmoid neuron producing the classification output.
  This simple architecture is sufficient to explore how changing depth and width affects performance. The rest of the tutorial focuses on systematically varying these hidden layers to study underfitting, overfitting, and the trade-off between model capacity and generalisation on a real medical dataset.

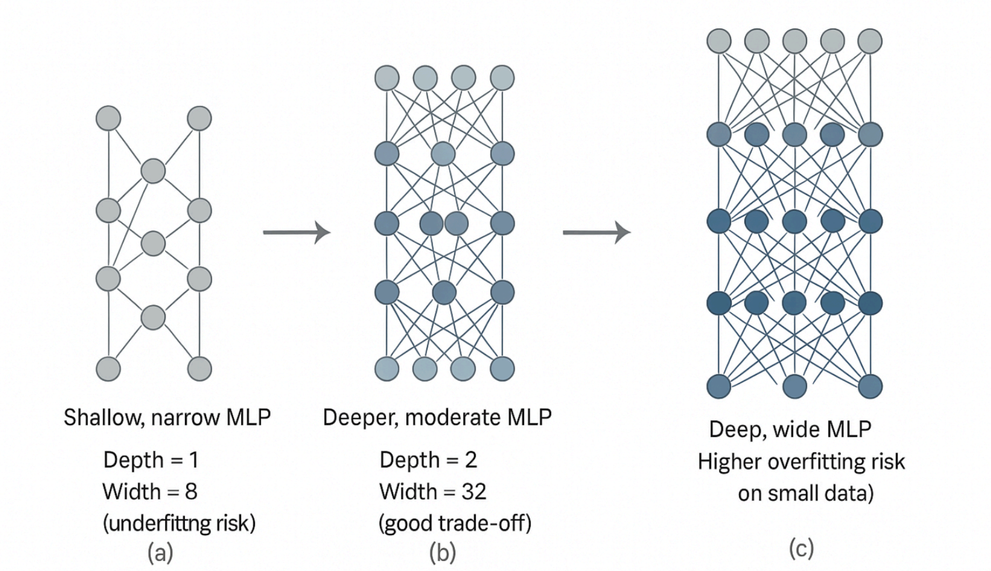


Figure 02

Illustrating depth (number of hidden layers) and width (neurons per layer) for shallow, moderate, and deep-wide MLPs. Increasing depth and width increases capacity but also overfitting risk on small datasets.

Three schematic MLP diagrams side by side: (a) shallow narrow network with one small hidden layer labelled underfitting risk; (b) two-layer network with moderate width labelled good trade-off; (c) deep wide network with many neurons labelled higher overfitting risk on small data.

Experiment 1: effect of depth

To isolate the effect of depth, we fixed the width at 32 neurons per layer and trained three architectures:

- **1 hidden layer:** (32,)
- **2 hidden layers:** (32, 32)
- **3 hidden layers:** (32, 32, 32)

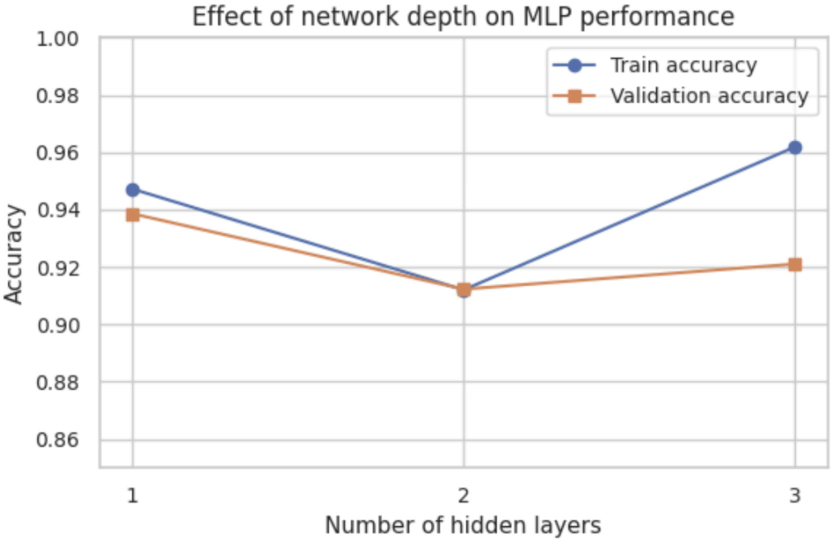


Figure 03 (depth plot)

Line plot showing train accuracy (blue circles) and validation accuracy (orange squares) vs number of hidden layers. Single layer achieves highest validation accuracy.

Figure 03 shows train and validation accuracy for each depth. The single-layer model achieves the highest validation accuracy (~0.939), indicating it generalises best. The two-layer model drops in both train and validation accuracy (~0.912), suggesting that the additional depth makes optimisation harder without providing useful capacity. The three-layer model recovers high training accuracy (~0.962) but validation accuracy (~0.921) lags, a classic sign of **overfitting**: the model memorises the training set without learning robust features.

Key insight: Adding depth does not monotonically improve generalisation on this dataset. A single hidden layer is sufficient to capture the decision boundary.

Experiment 2: effect of width

To isolate the effect of width, we fixed the architecture at two hidden layers and varied the width:

- **Narrow:** (8, 8) (64 parameters total in hidden layers)
- **Medium:** (32, 32) (1024 parameters)
- **Wide:** (128, 128) (16384 parameters)

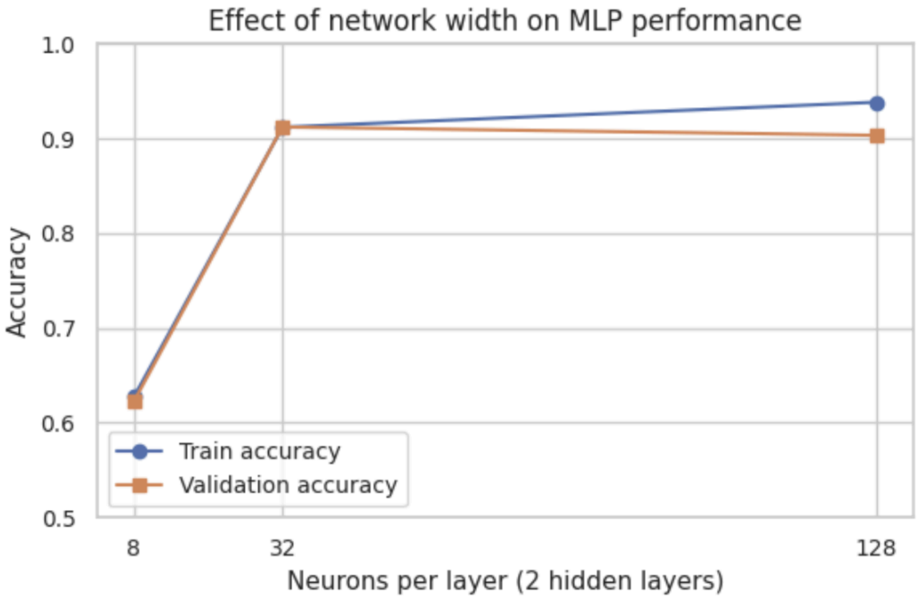


Figure 04 (width plot)

Line plot showing train accuracy (blue circles) and validation accuracy (orange squares) versus neurons per layer for a two-layer MLP; very narrow networks underfit, medium width performs best, and very wide networks begin to overfit.

Figure 04 shows train and validation accuracy for each width. The narrow network severely **underfits**: both train and validation accuracy are only ~0.62, meaning the model cannot learn the task. The medium-width network achieves good validation accuracy (~0.912), indicating a good bias–variance trade-off. The wide network further increases training accuracy (~0.938) but decreases validation accuracy (~0.904), again indicating **overfitting**.

Key insight: Increasing width helps up to a point but can hurt generalisation if the model is unnecessarily complex for the dataset size.

Final model, baseline comparison, and test evaluation

Based on validation accuracy, we selected the single-layer, 32-neuron MLP as the final model. Retraining on the combined train+validation data and evaluating on the test set:

- **MLP (32,) test accuracy:** 0.947
- **Logistic regression baseline test accuracy:** 0.983

Surprisingly, logistic regression—a simple linear model—achieves higher test accuracy than the MLP. This demonstrates that **on small tabular datasets, simpler baselines can outperform more complex models**. The confusion matrix for the MLP shows 5 false negatives (malignant misclassified as benign) out of 42 malignant cases, a clinically important error type.

This comparison illustrates a central principle of machine learning: **higher model capacity does not guarantee better generalisation**. Practitioners should always evaluate simple baselines before deploying deep models.

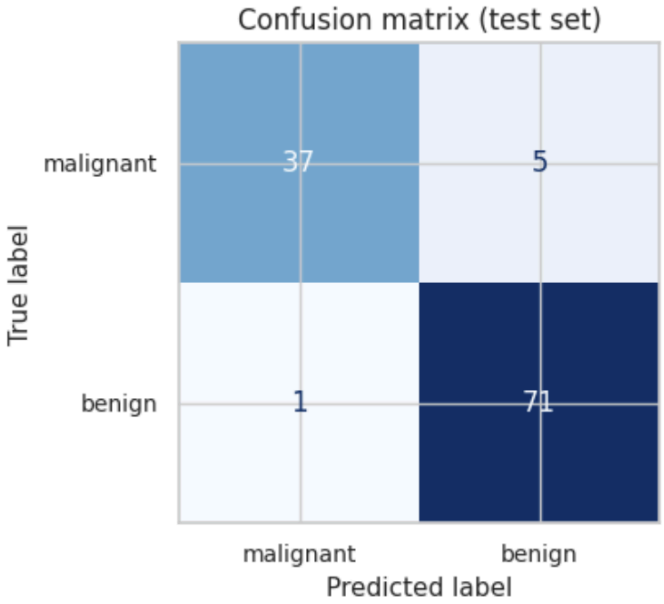
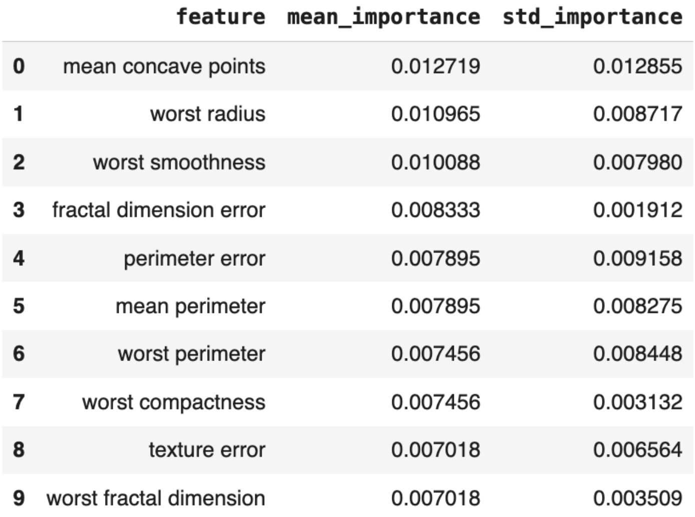


Figure 05 (feature importance table)

Feature importance and interpretability

To understand which features the MLP relies on most, we computed permutation feature importance on the test set. Figure 07 shows the top 10 most important features. The leading features are:



	feature	mean_importance	std_importance
0	mean concave points	0.012719	0.012855
1	worst radius	0.010965	0.008717
2	worst smoothness	0.010088	0.007980
3	fractal dimension error	0.008333	0.001912
4	perimeter error	0.007895	0.009158
5	mean perimeter	0.007895	0.008275
6	worst perimeter	0.007456	0.008448
7	worst compactness	0.007456	0.003132
8	texture error	0.007018	0.006564
9	worst fractal dimension	0.007018	0.003509

Figure 06 (feature importance table)

- **Mean concave points** (importance ~0.013)
- **Worst radius** (importance ~0.011)
- **Worst smoothness** (importance ~0.010)
- **Fractal dimension error, perimeter error, mean perimeter**

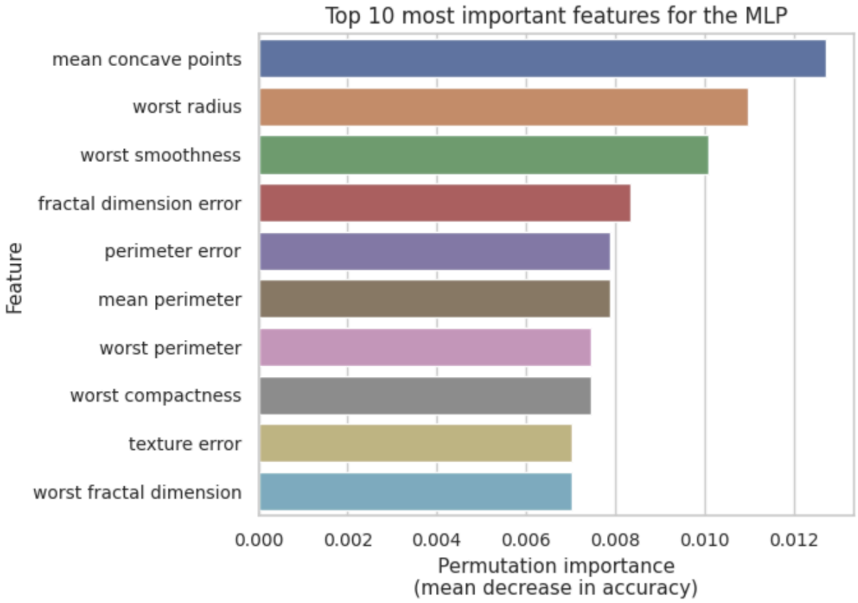


Figure 07 (feature importance plot)

Horizontal bar chart of the top 10 permutation feature importances for the final MLP, highlighting mean concave points, worst radius, and worst smoothness as the most influential features for predicting malignancy.

These features describe the size, shape regularity, and boundary characteristics of the cell nuclei. Irregular, concave boundaries with large variance (high "worst" values) are strongly associated with malignancy, aligning with clinical knowledge. Notably, the MLP's decisions are driven by only a small subset of the 30 features, suggesting that deeper or wider networks mainly change how the model combines these key variables.

Practical guidelines and limitations

Based on these experiments, this tutorial recommends

1. **Start with simple baselines** (logistic regression, decision trees) before investing in deep MLPs.
2. **Use shallow, moderately wide networks** on small tabular datasets (<10,000 samples). A single or two hidden layers with 32–64 neurons is a good starting point.
3. **Monitor train vs validation accuracy curves** to detect overfitting and apply early stopping.
4. **Leverage feature importance** to build trust in model decisions and connect them to domain knowledge.
5. **Limitations:** This study uses a single, relatively small dataset. Findings may not generalise to other medical datasets or domains. Additionally, we did not systematically search over learning rate, regularisation (L2 / dropout), or batch size; such a search might reveal different optimal architectures.

6. **Extensions:** Future work could compare MLPs against tree-based models (random forests, gradient boosting), explore regularisation techniques to combat overfitting, or apply the same depth/width analysis to other tabular datasets.

Conclusion

This tutorial showed how the depth and width of a Multilayer Perceptron influence performance on the Breast Cancer Wisconsin (Diagnostic) dataset. A shallow network with a single hidden layer of 32 neurons achieved the best validation and strong test accuracy, while deeper or very wide architectures mainly increased training accuracy without improving generalisation, indicating overfitting on this small tabular dataset.

Comparing against a logistic regression baseline revealed that a simple linear model can outperform the MLP, reinforcing the importance of starting with low-capacity models before adding complexity. Permutation feature importance further showed that the network relies heavily on a small subset, indicating over geometrical features—such as mean concave points and worst radius—that align with established clinical indicators of malignancy. Overall, the experiments highlight that careful control of model capacity, baseline comparisons, and interpretability tools are crucial when applying neural networks to real medical data.

References

- Scikit-learn developers. (2023). *Scikit-learn: Machine Learning in Python*. Retrieved from https://scikit-learn.orgscikit-learn UC Irvine Machine Learning Repository. (1995). *Breast Cancer Wisconsin (Diagnostic) Data Set*. Retrieved from https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic
- Brownlee, J. (2023). *Your First Machine Learning Project in Python Step-By-Step*. Machine Learning Mastery. Retrieved from https://www.machinelearningmastery.com/machine-learning-in-python-step-by-step/
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Zhang, C., Bengio, S., Hardt, M., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3), 107–115.
- Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (pp. 4765–4774).
  Khan, S. et al. (2022). Predicting breast cancer with supervised machine learning: techniques and performance evaluation. *Healthcare*, 10(8). Retrieved from https://pmc.ncbi.nlm.nih.gov/articles/PMC9398810/pmc.ncbi.nlm.nih
- Scikit-learn developers. (2017). *Neural network models (supervised)*. Retrieved from https://scikit-learn.org/stable/modules/neural_networks_supervised.html