

# **SIS1: Classification using Perceptron and Multilayer Perceptron (MLP)**

Student 1: Balym Sekenova

ID: 23B031433

Student 2: Nursultan Zhantuar

ID: 23B030356

Student 3: Maxim Abakarov

ID: 23B031182

Student 4: Balzhan Batyrbaeva

ID: 23B031245

February 25, 2026

# 1 Objectives

The objectives of this laboratory work are:

- To understand the architecture of a single-layer Perceptron and Multilayer Perceptron (MLP).
- To calculate the number of trainable parameters.
- To compare classification performance on non-linearly separable data.
- To analyze the effect of network depth and width.
- To evaluate models using classification metrics.

## 2 Dataset Description

In this work, the Moons dataset from `scikit-learn` was used. The dataset consists of two interleaving half-circles and is not linearly separable.

- Total samples: 1000
- Training set: 800 samples
- Test set: 200 samples
- Noise level: 0.2

## 3 Task 1: Network Architectures

### 3.1 Single-Layer Perceptron (2–1)

The Perceptron consists of two input features and one output neuron with a sigmoid activation function.

$$y = \sigma(Wx + b) \tag{1}$$

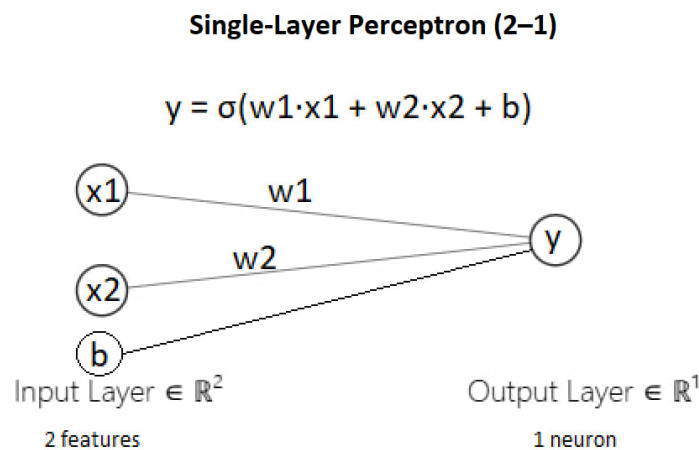


Figure 1: Single-Layer Perceptron (2–1)

The model is fully connected and includes a bias term.

### 3.2 MLP with One Hidden Layer (2–8–1)

This architecture contains one hidden layer with 8 neurons and ReLU activation.

$$h = \text{ReLU}(W_1x + b_1) \quad (2)$$

$$y = \sigma(W_2h + b_2) \quad (3)$$

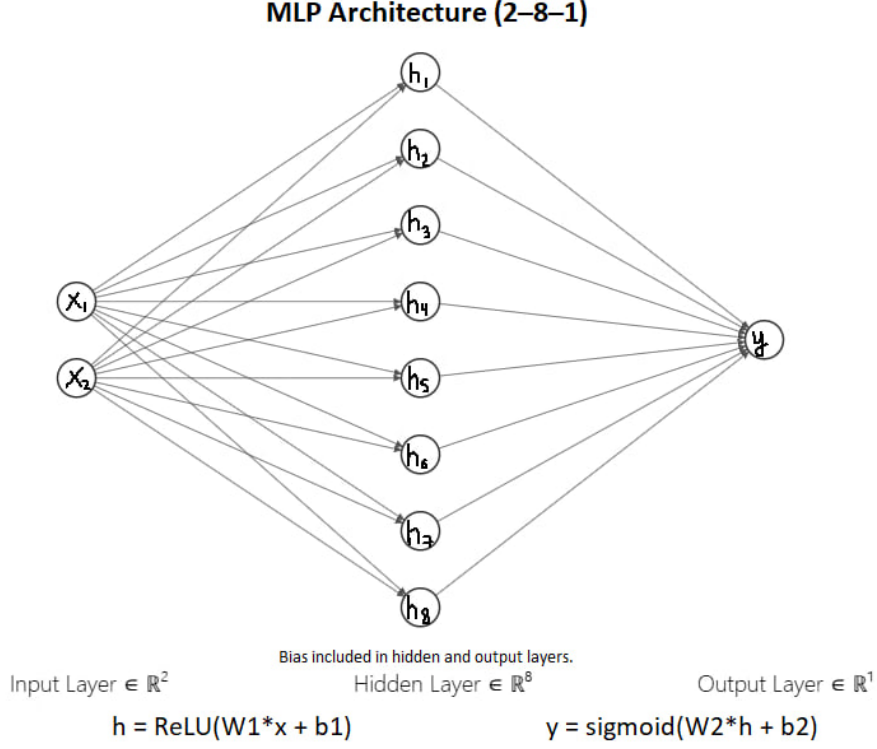


Figure 2: Multilayer Perceptron (2–8–1)

The network is fully connected. Bias terms are included in hidden and output layers.

### 3.3 Deep MLP (2–16–8–1)

This deep architecture contains two hidden layers with 16 and 8 neurons respectively.

$$h_1 = \text{ReLU}(W_1x + b_1) \quad (4)$$

$$h_2 = \text{ReLU}(W_2h_1 + b_2) \quad (5)$$

$$y = \sigma(W_3h_2 + b_3) \quad (6)$$

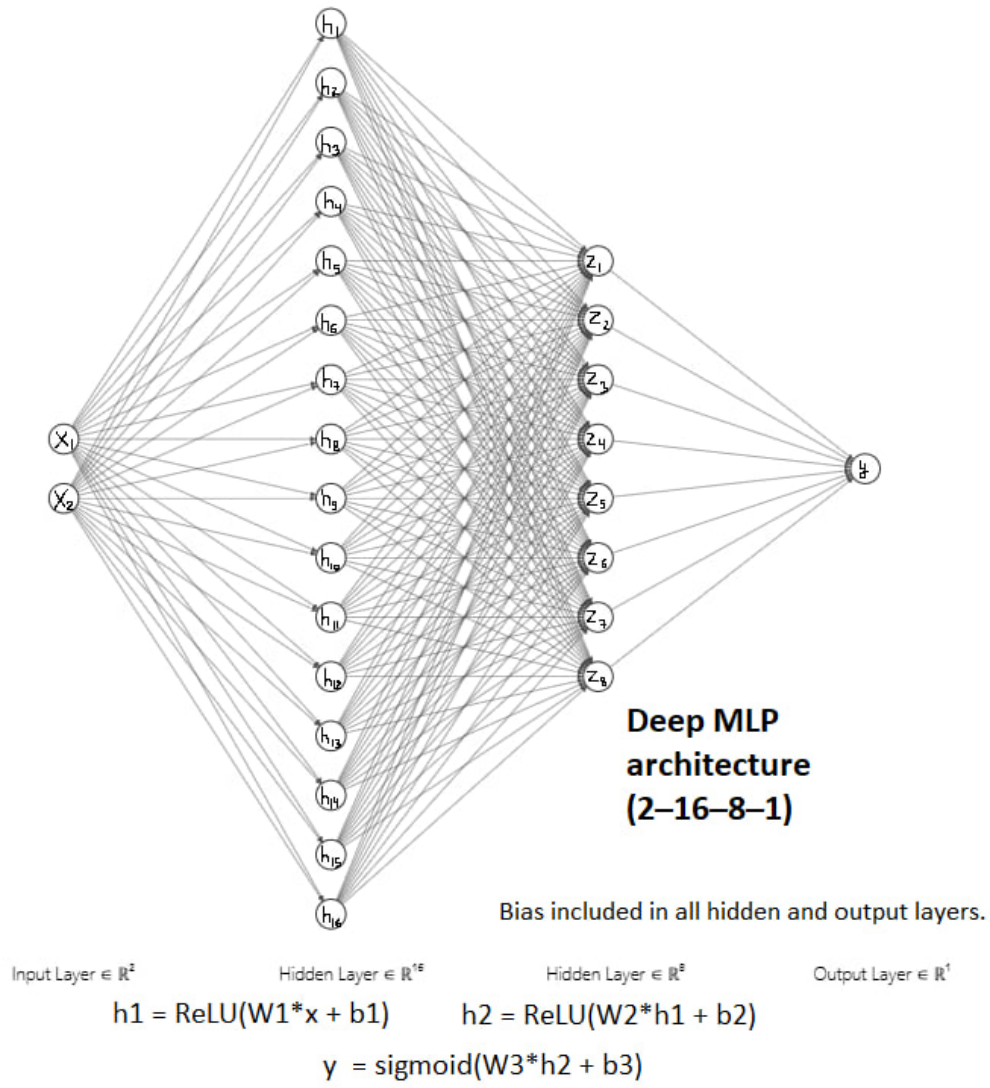


Figure 3: Deep Multilayer Perceptron (2-16-8-1)

All layers are fully connected. Bias terms are included in all hidden and output layers.

## 4 Task 2: Trainable Parameters

The number of trainable parameters in a fully connected layer is calculated as:

$$\text{Parameters} = n_{in} \times n_{out} + n_{out}$$

Architecture	Calculation (show work)	Total Parameters
Perceptron (2-1)	$2 \times 1 + 1$	3
MLP (2-4-1)	Layer 1: $2 \times 4 + 4 = 12$ Layer 2: $4 \times 1 + 1 = 5$	$12 + 5 = 17$
MLP (2-8-1)	Layer 1: $2 \times 8 + 8 = 24$ Layer 2: $8 \times 1 + 1 = 9$	$24 + 9 = 33$

Table 1: Calculation of trainable parameters

## 5 Task 3: Experimental Results

In this section, we compare the performance of Perceptron and MLP architectures on the Moons dataset. The dataset contains 1000 samples with 80% used for training and 20% for testing. The noise level was set to 0.2.

### 5.1 Classification Metrics Comparison

Model	Accuracy	Precision	Recall	F1-Score	Training Time (s)
Perceptron	0.825	0.756	0.960	0.846	0.005
MLP (2-4-1)	0.860	0.860	0.860	0.860	0.417
MLP (2-8-1)	0.890	0.875	0.910	0.892	0.619
MLP (2-16-1)	0.975	0.970	0.980	0.975	1.337
MLP (2-32-1)	0.980	0.980	0.980	0.980	1.564
MLP (2-64-1)	0.980	0.980	0.980	0.980	1.576
MLP (2-8-8-8-1)	0.985	0.990	0.980	0.985	0.987

Table 2: Classification Metrics Comparison

The Perceptron achieved moderate accuracy but showed imbalance between precision and recall. The MLP models demonstrated improved and more balanced performance, confirming the advantage of non-linear models for this dataset.

## 5.2 Effect of Network Width (1 Hidden Layer)

Hidden Neurons	Parameters	Accuracy	F1-Score
4	17	0.860	0.860
8	33	0.890	0.892
16	65	0.975	0.975
32	129	0.980	0.980
64	257	0.980	0.980

Table 3: Effect of hidden layer width on performance

Increasing the number of hidden neurons significantly improved performance up to 16 neurons. Beyond 32 neurons, accuracy improvements became marginal, indicating diminishing returns.

## 5.3 Effect of Network Depth (8 Neurons per Layer)

Hidden Layers	Architecture	Parameters	Accuracy
1	2-8-1	33	0.89
2	2-8-8-1	105	0.98
3	2-8-8-8-1	177	0.985
4	2-8-8-8-8-1	249	0.98

Table 4: Effect of network depth on performance

Increasing network depth improved accuracy up to three hidden layers. The fourth hidden layer did not improve performance further, suggesting that additional depth does not necessarily guarantee better generalization.

# 6 Task 4: Decision Boundaries

Decision boundary plots illustrate how each model separates the two classes in the 2D feature space. The background shading represents the predicted class for each region, while the scatter points show the actual dataset labels.

## Perceptron

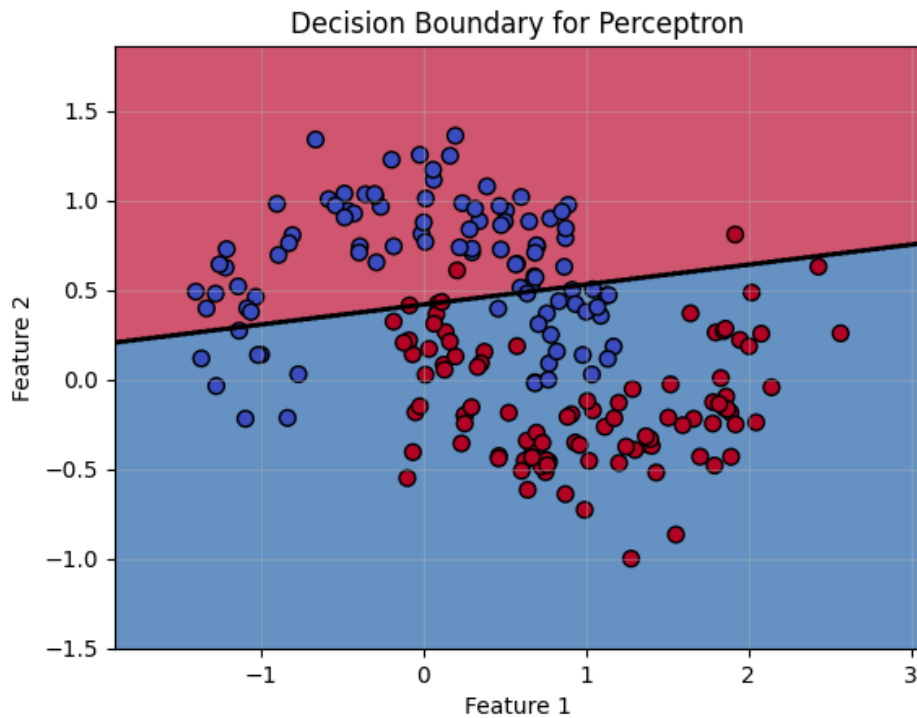


Figure 4: Decision boundary of the Single-Layer Perceptron (Accuracy: 0.825).

The Perceptron produces a **linear** decision boundary, which is a single straight line dividing the feature space into two half-planes. Since the Moons dataset is not linearly separable, this boundary cannot correctly classify the interleaved regions, leading to systematic misclassification near the overlap zone.

## MLP with Best Accuracy (2-8-8-8-1)

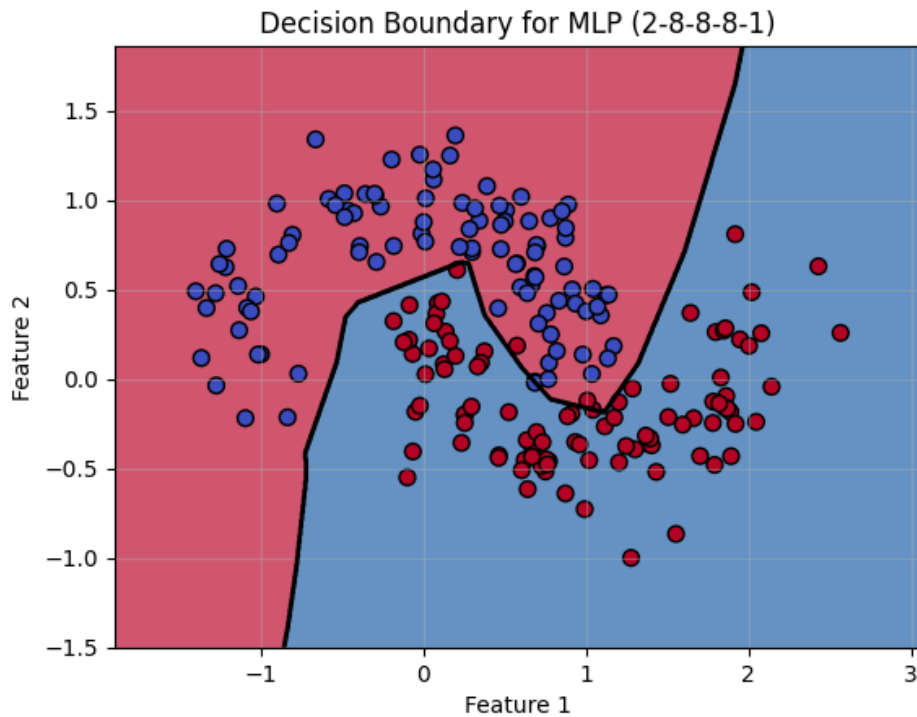


Figure 5: Decision boundary of the best MLP (2-8-8-8-1), Accuracy: 0.985.

The deep MLP with three hidden layers of 8 neurons each produces a smooth, **highly non-linear** boundary that closely follows the curvature of both moon-shaped clusters. This architecture achieved the highest test accuracy of 98.5%.

## MLP with Worst Accuracy (2-4-1)

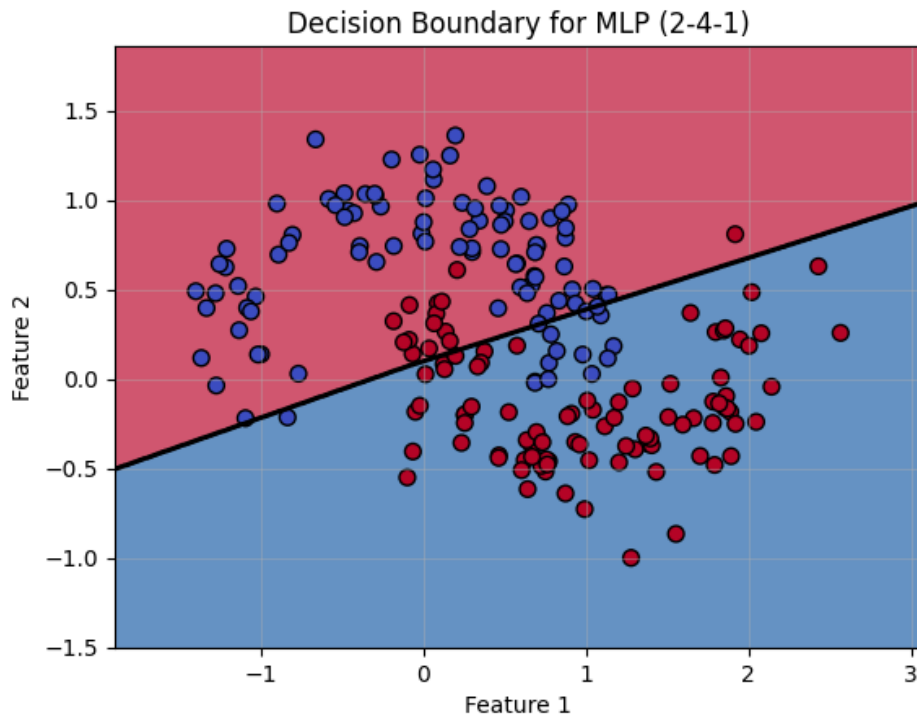


Figure 6: Decision boundary of the weakest MLP (2-4-1), Accuracy: 0.860.

The smallest MLP (4 hidden neurons) learns a mildly curved boundary that is better than the Perceptron but still fails to capture the full curvature of the moon shapes, resulting in the lowest accuracy among all MLP models at 86.0%.

## 7 Analysis Questions

**Q1: Why does the Perceptron perform poorly on the Moons dataset? Explain with reference to the decision boundary plot.**

The Perceptron is a linear classifier whose decision boundary is always a hyperplane (a straight line in 2D). The Moons dataset consists of two interleaving half-circles that are not linearly separable, meaning no single straight line can correctly divide the two classes. As seen in Figure 4, the linear boundary inevitably misclassifies a large portion of points in the overlapping region. This is a fundamental architectural limitation, not a training issue.

**Q2: How does increasing the width (number of neurons) of a single hidden layer affect accuracy? Is there a point of diminishing returns?**

Increasing the number of neurons in a single hidden layer consistently improved both accuracy and F1-score up to 16 neurons (accuracy: 97.5%). Going from 16 to 32 neurons provided a modest further gain (98.0%), while increasing from 32 to 64 neurons yielded no additional improvement. This clearly demonstrates **diminishing returns**: beyond a certain width the model has sufficient capacity to represent the data, and adding more neurons only increases computation time without improving generalization.

**Q3: How does increasing the depth (number of hidden layers) affect accuracy and training time?**

Adding hidden layers progressively improved accuracy from 89.0% (1 layer) to 98.5% (3 layers), showing that depth helps the network learn more abstract, hierarchical representations. However, the 4-layer network slightly regressed to 98.0%, suggesting diminishing returns with depth as well. Regarding training time, deeper networks with 8 neurons per layer trained faster than wide single-layer alternatives (e.g., 2–8–8–8–1 at 0.987 s vs. 2–64–1 at 1.576 s) because their total parameter count is lower.

**Q4: Which model achieved the best accuracy? Was it also the model with the most parameters?**

The MLP (2–8–8–8–1) achieved the highest accuracy of 98.5% with 177 trainable parameters. Notably, it was *not* the model with the most parameters: MLP (2–64–1) has 257 parameters yet reached only 98.0%. This demonstrates that a deeper architecture with fewer total parameters can outperform a wide but shallow network.

**Q5: What is the relationship between the number of parameters and the risk of overfitting?**

In general, a higher parameter count gives a model greater capacity to memorize training data, increasing the risk of overfitting - i.e., learning patterns specific to the training set that do not generalize to unseen data. However, the relationship is not strictly linear: a well-regularized deep model with many parameters may generalize better than a poorly designed model with fewer parameters. In our experiments, models with more parameters (e.g., 2–64–1 with 257 params) did not overfit noticeably, likely because the Moons dataset is relatively simple and the training set is large enough relative to model complexity. Common mitigation techniques include L2 regularization, dropout, and early stopping.

**Q6: Based on your results, what architecture would you recommend for this dataset and why?**

We recommend the **MLP (2–8–8–8–1)** architecture. It achieves the highest accuracy (98.5%) with only 177 parameters and trains in under 1 second, which is faster than wider single-layer alternatives. The three hidden layers allow the network to learn increasingly abstract non-linear features that match the curved structure of the Moons data. It offers the best balance between accuracy, model complexity, and training efficiency.

**Q7: How would the results change if we used a linearly separable dataset (e.g., generated with `make_blobs`)?**

On a linearly separable dataset such as `make_blobs`, the Perceptron would achieve near-perfect accuracy since its linear decision boundary is sufficient to separate the classes. The performance gap between the Perceptron and MLP models would essentially disappear. MLPs would still reach high accuracy but would provide little advantage over the simpler Perceptron. This underscores the principle that model complexity should match data complexity: for linearly separable problems, a Perceptron is the most efficient and interpretable choice.

## 8 Conclusion

This laboratory work investigated the classification capabilities of the Single-Layer Perceptron and Multilayer Perceptron architectures on the non-linearly separable Moons dataset. The experiments confirmed that the Perceptron, constrained to a linear decision boundary, is fundamentally unable to separate the two interleaving half-circles, achieving only 82.5% accuracy with a notable imbalance between precision (75.6%) and recall (96.0%).

MLP models with ReLU-activated hidden layers demonstrated clear advantages by learning curved, non-linear decision boundaries. Experiments on network width showed that performance improves significantly up to 16 hidden neurons and plateaus beyond 32, revealing diminishing returns from excessive width. Experiments on network depth showed that three hidden layers of 8 neurons each was optimal, achieving 98.5% accuracy — the best result across all tested architectures — while using fewer parameters (177) than the widest single-layer model (257 parameters).

The best model, MLP (2–8–8–8–1), demonstrated that depth can be more parameter-efficient than width when learning complex geometric structures such as the moon-shaped clusters in this dataset. The decision boundary visualizations confirmed these findings: the Perceptron drew a straight line through the data while the deep MLP formed smooth, curved boundaries tightly aligned with the true class distributions.

Overall, the results establish that for non-linearly separable data, MLPs with appropriate depth and moderate width substantially outperform linear classifiers. Model selection should balance accuracy, parameter count, training time, and generalization risk, with regularization strategies considered when scaling to larger and more complex datasets.