Understanding Multilayer Perceptron's: Depth vs. Width

By
VAISHNAV RAJITH KALATHIL

- A simple visual explanation

What is an MLP?

A Multilayer Perceptron (MLP) is a type of artificial neural network composed of multiple layers of neurons

It is one of the most fundamental deep learning models used for classification and regression tasks.

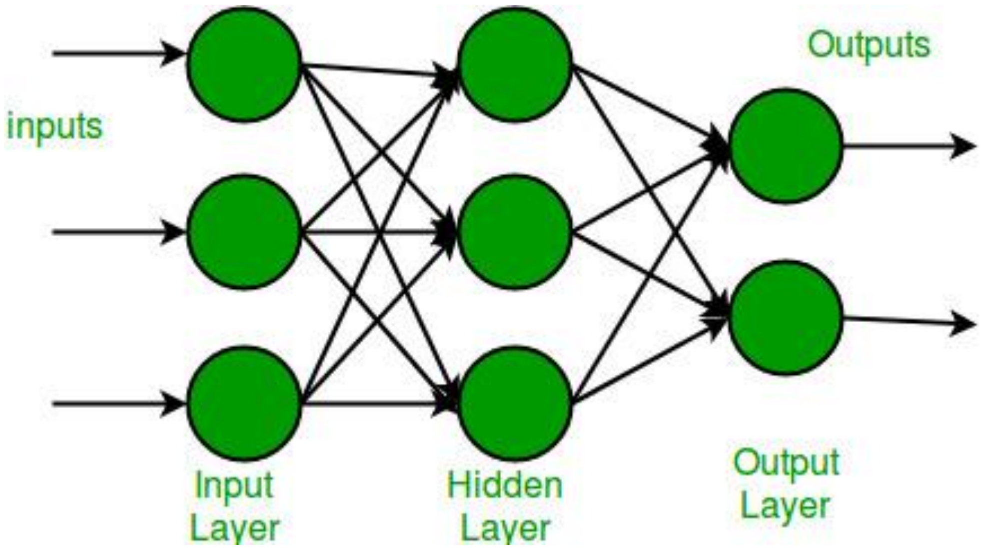
In this tutorial, we focus on.

Understanding how the depth (number of hidden layers) and width (number of neurons in each layer) affect the MLP's performance.

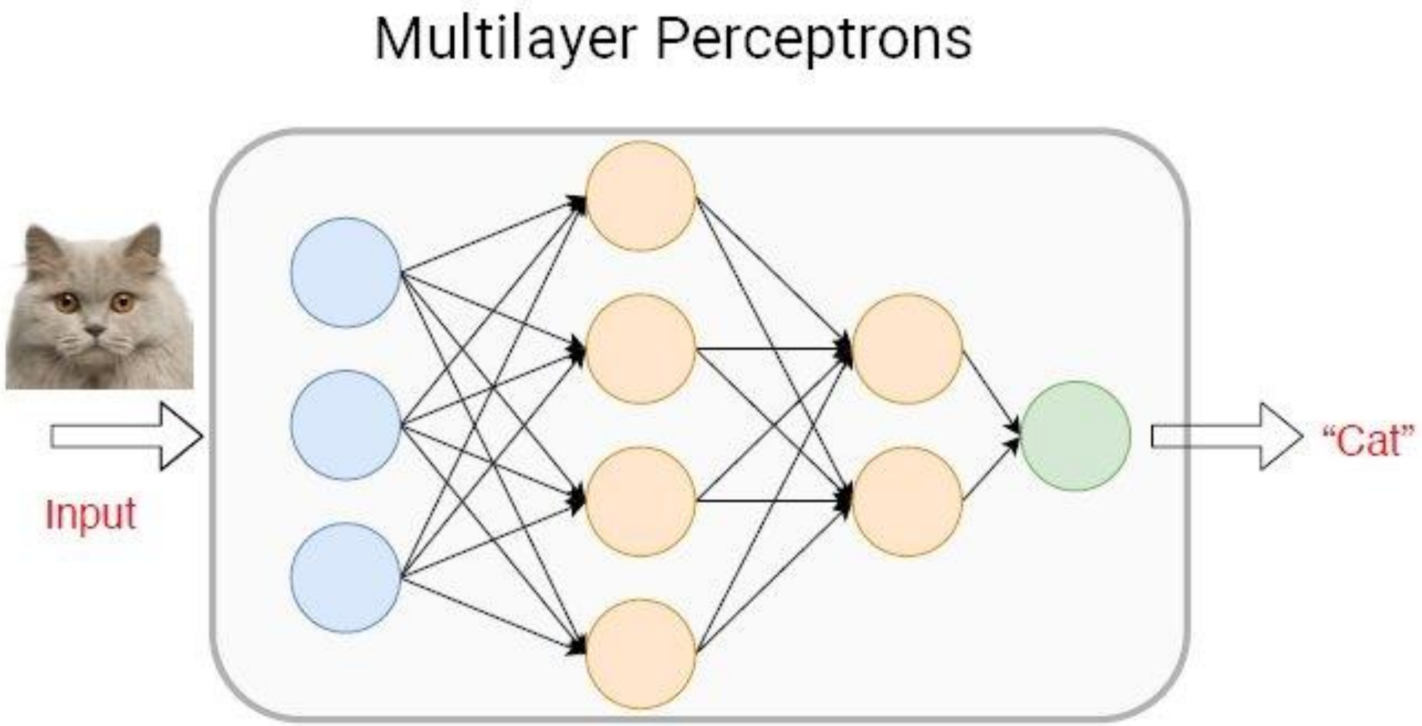
- A Multilayer Perceptron (MLP) is like a recipe: layers = steps, neurons = ingredients.
- Each layer transforms input data and passes it on.

What is an MLP?

Key Components of a Multi-Layer Perceptron (MLP):
Input Layer: This is where the data enters the network. Each neuron in the input layer represents one feature from your dataset. For example, if your data has three features, the input layer will have three neurons.
Hidden Layers: These are the layers between the input and output. There can be one or several hidden layers, each with multiple neurons. These layers are responsible for learning patterns and processing the information coming from the input layer.
Output Layer: This layer provides the final prediction or result. The number of neurons here matches the number of outputs you need. For instance, if you're predicting a single value, there will be one neuron; if you're classifying into multiple categories, you'll have one neuron per category.



What is an MLP?



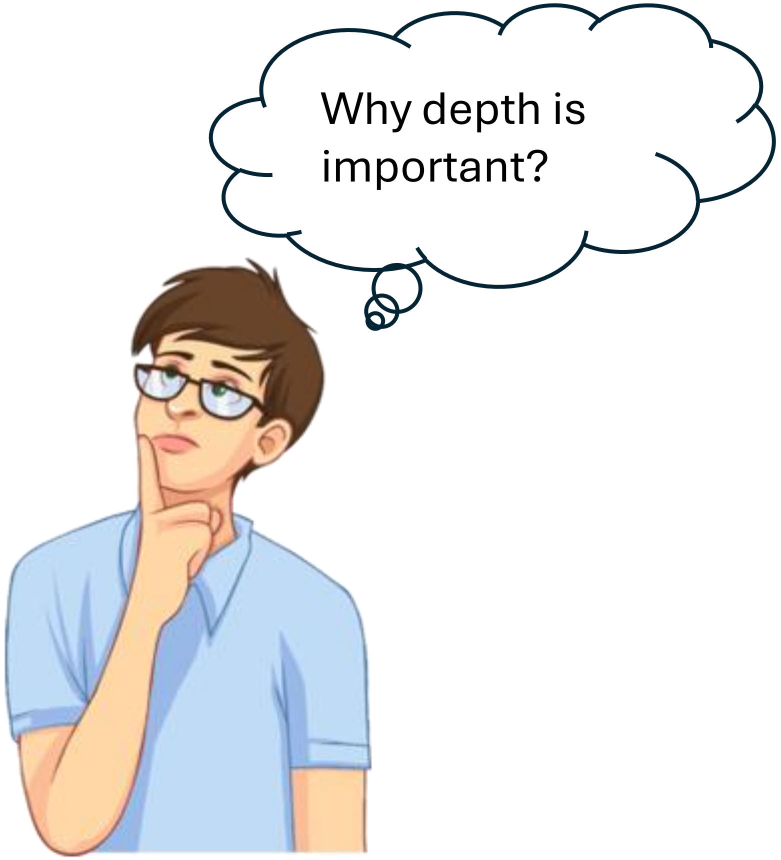
What is Depth?

DEPTH REFERS TO THE NUMBER OF LAYERS IN THE NETWORK. MORE LAYERS ALLOW THE MODEL TO LEARN MORE COMPLEX PATTERNS.

DEPTH = NUMBER OF LAYERS.

MORE LAYERS = LEARNING MORE COMPLEX PATTERNS.

Why is Depth Is important ?

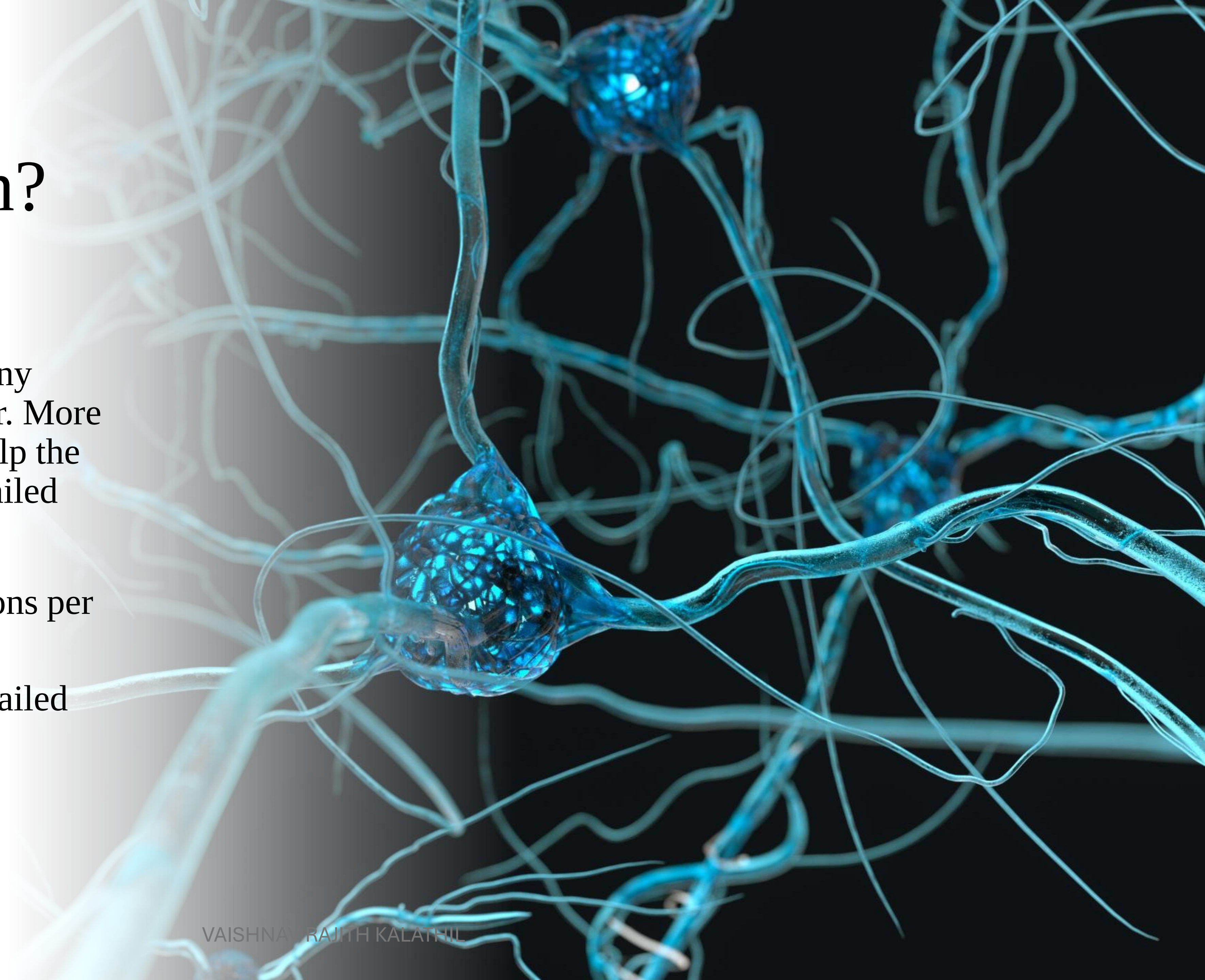


Why is Depth important?

- Depth is crucial in Multilayer Perceptron's (MLP) because it allows the network to learn more complex and hierarchical representations of the input data, leading to better **generalization** and **performance**, especially for complex tasks.

What is Width?

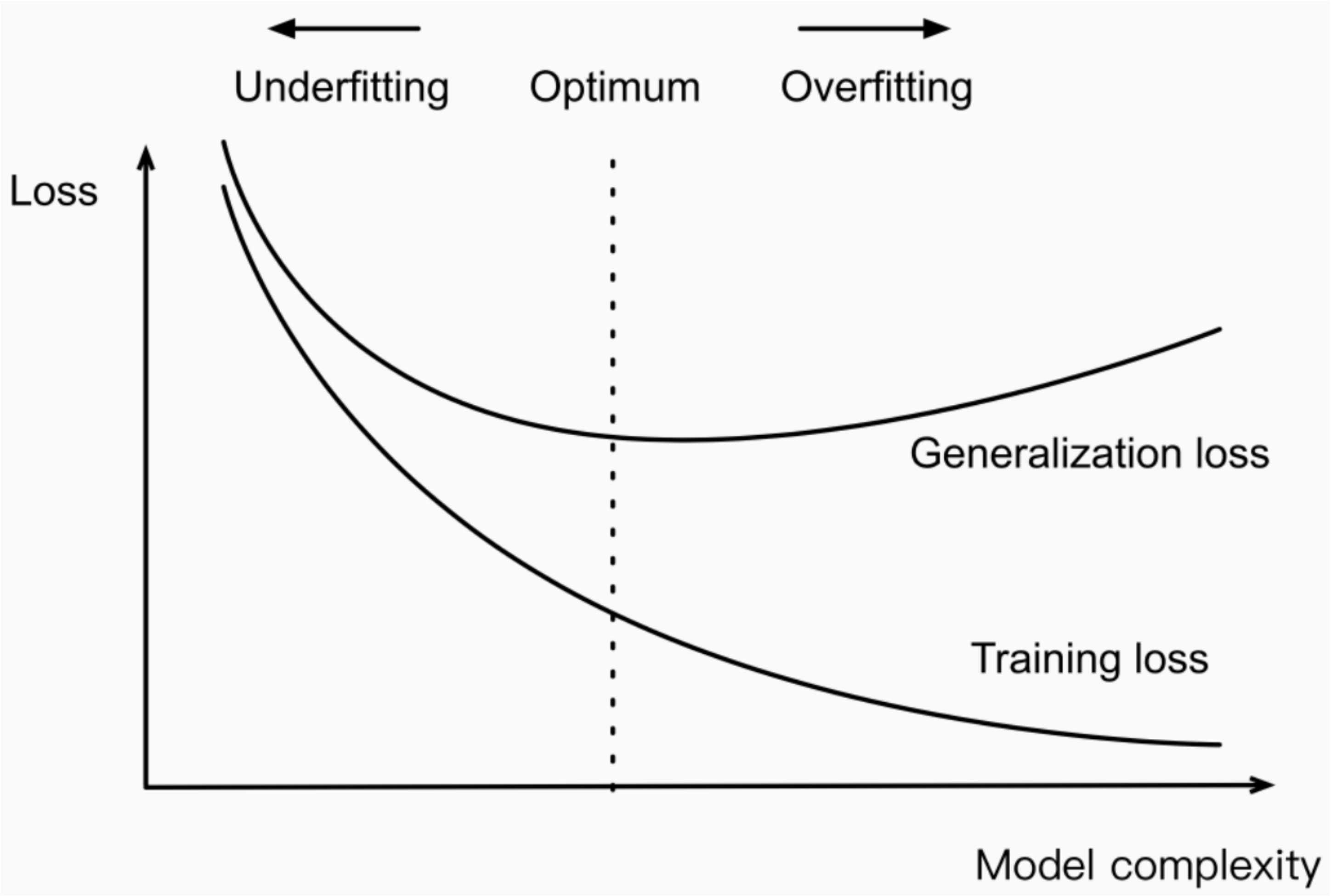
- **Width** refers to how many neurons are in each layer. More neurons per layer can help the model capture more detailed information.
- Width = number of neurons per layer.
- More neurons = more detailed information.



Why Balance Matters ?

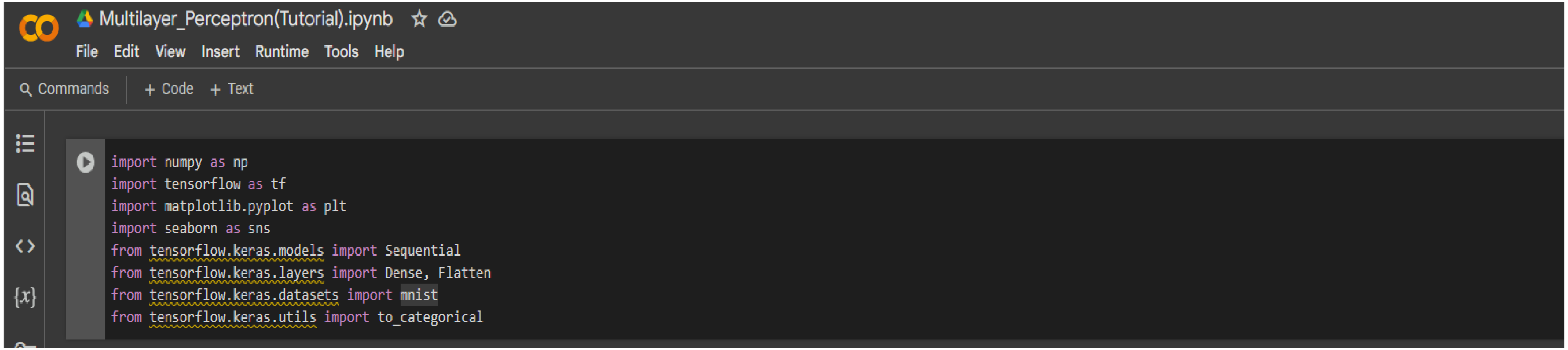
Why Balance Matters ?

- Too little depth or width = underfitting.
- Too much = overfitting.
- Just right = best learning.



Understanding Multilayer Perceptron's: How Depth and Width Impact Performance

Step 1: Import Libraries



Multilayer_Perceptron(Tutorial).ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```
# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize pixel values (0 to 1 range)
x_train, x_test = x_train / 255.0, x_test / 255.0

# Convert labels to one-hot encoding
y_train, y_test = to_categorical(y_train), to_categorical(y_test)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

Step 2: Load & Preprocess the Dataset

- MNIST (handwritten digits)- Small but useful for demonstrating neural network performance. Easy to train and allows clear visualization of results.
- Data Processing : Normalization , train- test split

Step 3: Define a Function to Build MLP Models

- \- Input Layer: Accepts raw data (such as pixel values from images).
- \- Hidden Layers: Perform computations and learn patterns with activation functions like Relu or sigmoid.
- \- Output Layer: Provides the final prediction result, typically with SoftMax or sigmoid activation.

Multilayer_Perceptron(Tutorial).ipynb

```
def create_mlp(hidden_layers=1, neurons_per_layer=64):
    model = Sequential()
    model.add(Flatten(input_shape=(28, 28)))  # Flatten input images
    # Add hidden layers dynamically
    for _ in range(hidden_layers):
        model.add(Dense(neurons_per_layer, activation='relu'))

    model.add(Dense(10, activation='softmax'))  # Output layer (10 classes)

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

Step 4: Train Models with Different Depths & Widths

- To demonstrate the effect of varying depth and width, we trained MLP models using the MNIST
- dataset. Models with different numbers of hidden layers (1, 3, and 5) and varying neurons per layer
- (32, 64, 128) were compared based on their validation accuracy.

Multilayer_Perceptron(Tutorial).ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```python
depths = [1, 3, 5]  # Number of hidden layers
widths = [32, 64, 128]  # Neurons per layer

history_dict = {}

for d in depths:
    for w in widths:
        print(f"\nTraining MLP with {d} layers and {w} neurons per layer...")
        model = create_mlp(hidden_layers=d, neurons_per_layer=w)
        history = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10, batch_size=128, verbose=1)
        history_dict[f"{d} layers, {w} neurons"] = history
```

Step 5: Plot Training Results

Multilayer_Perceptron(Tutorial).ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

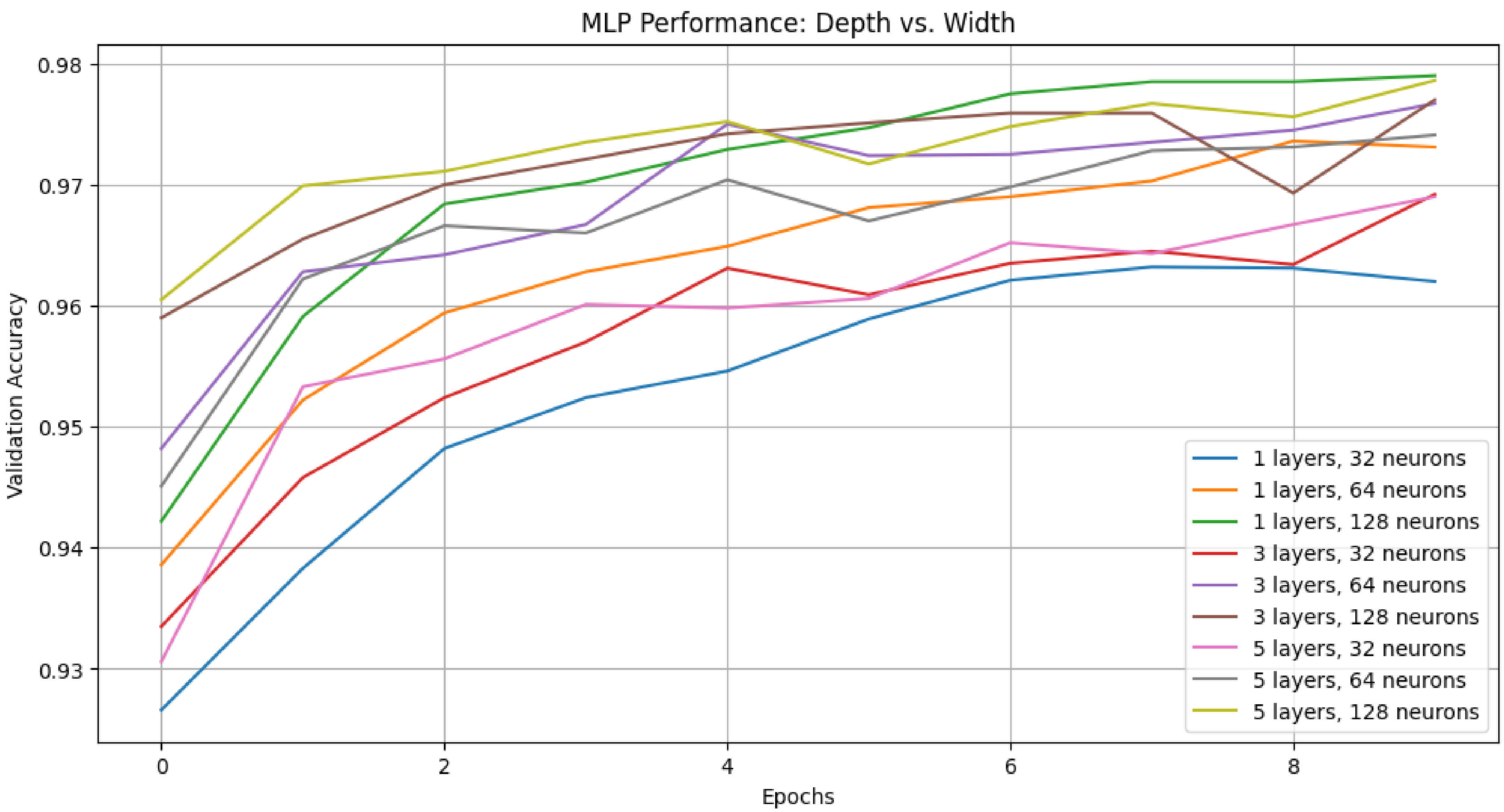
```
469/469 ━━━━━━━━━━━━━━━━━━━━ 6s 9ms/step - accuracy: 0.9920 - loss: 0.0242 - val_accuracy: 0.9756 - val_loss: 0.0879
Epoch 10/10
469/469 ━━━━━━━━━━━━━━━━━━━━ 4s 6ms/step - accuracy: 0.9930 - loss: 0.0221 - val_accuracy: 0.9786 - val_loss: 0.0796
```

```python
plt.figure(figsize=(12, 6))

for label, history in history_dict.items():
    plt.plot(history.history['val_accuracy'], label=f"{label}")

plt.xlabel("Epochs")
plt.ylabel("Validation Accuracy")
plt.title("MLP Performance: Depth vs. Width")
plt.legend()
plt.grid()
plt.show()
```

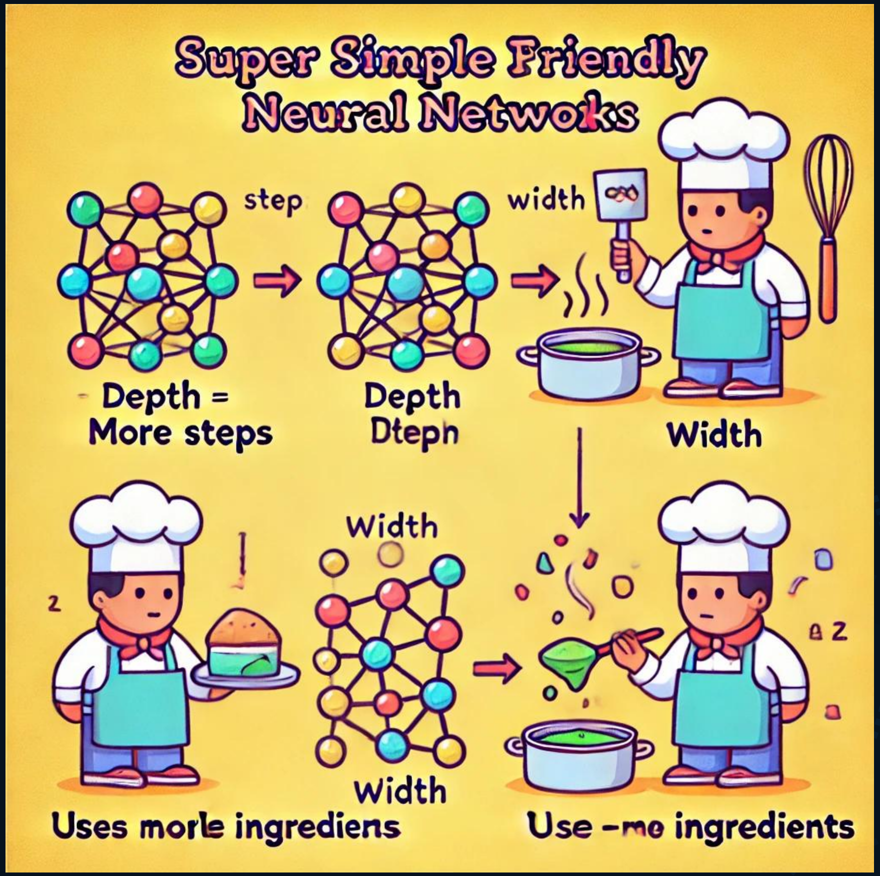
Final Result



Observation

- This Plot shows how the performance of a multilayer perceptron (MLP) changes with variations in both depth (number of layers) and width (number of neurons per layer), using validation accuracy over training epochs as the measure.
- Observations :-
    - X-axis : Number of epochs
    - Y-axis :Validation accuracy(how well the model performs on unseen data)
    - Curves: Each line represents a different MLP configuration
        - Depth : 1,3,5 layers
        - Width : 32,64,128 neurons per layer

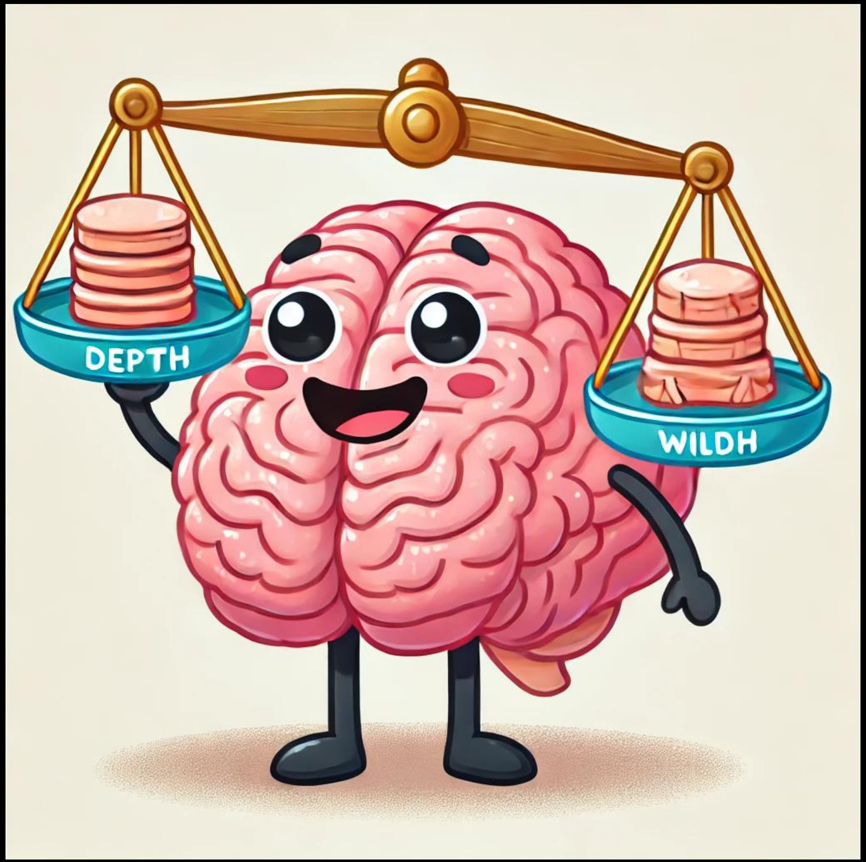
Real-Life Analogy



- Depth: Number of steps in cooking.
- Width: Number of ingredients per step.
- A balance makes the perfect dish!

Summary

- Balance depth and width.
- Deeper and wider networks can learn better but need careful tuning.



References

TensorFlow Documentation: https://www.tensorflow.org/

Research on MLP Performance: https://arxiv.org/pdf/1802.08551.pdf

Scikit-Learn: https://scikit-learn.org/stable/modules/neural_networks_supervised.html

Thank YOU !

VAISHNAV RAJITH KALATHIL