# Deep Learning
# CSC-Elective

Instructor : Dr. Muhammad Ismail Mangrio

Slides credit Dr. M Asif Khan

ismail@iba-suk.edu.pk

*Week 5-6*

# Contents

- What is Tensor?
- TensorFlow
- Keras
- Difference between Training, Testing, and Validation
- Dropout Layer

# What is Tensor?

- A mathematical object that generalizes **scalars, vectors, and matrices**.

- It is a **multi-dimensional array of numbers** used to represent data and relationships in a variety of applications.

# What is Tensor?

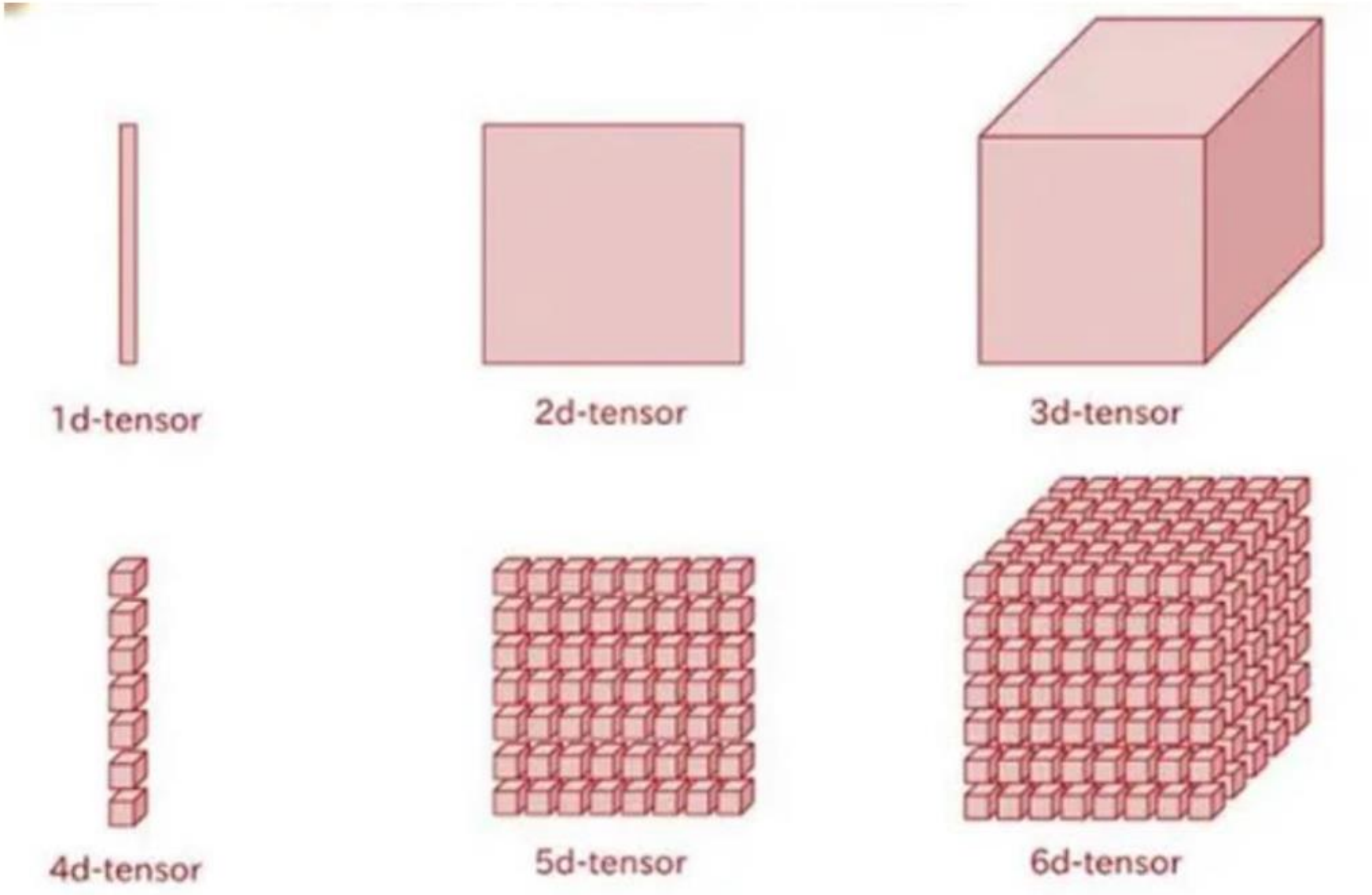
- **Types of Tensors:**
  1. Scalar (0D Tensor)
     Example: 5 (A single number)

  2. Vector (1D Tensor)
     Example: [1,2,3] (An array of numbers)

  3. Matrix (2D Tensor)
     Example: [1234][ 13 24 ]
     A grid of numbers arranged in rows and columns

  4. Higher-Dimensional Tensor
     Example: A 3D tensor could represent an RGB image, with dimensions corresponding to height, width, and color channels.

# What is Tensor?



# TensorFlow

- **Definition: TensorFlow** is an open-source machine learning framework developed by Google. It provides tools and libraries for building and deploying machine learning models.


- **Key Features**:
  - **Flexible Architecture**:
    - Supports deployment on various platforms: CPUs, GPUs, and TPUs.
    - Can run on desktops, servers, and mobile devices.
  - **High-Level APIs Keras**:
    - Simplified API for building neural networks with ease.
    - tf.data: Tools for efficient data input pipelines.
  - **Automatic Differentiation**:
    - tf.GradientTape: Automatically computes gradients for training neural networks.

# TensorFlow

- **Key Features**:
  - **Comprehensive EcosystemTensorFlow Hub**:
    - Repository of pre-trained models.
    - TensorFlow Lite: For deploying models on mobile and edge devices.
    - TensorFlow Extended (TFX): End-to-end pipeline for production machine learning
- **Core Component**s:
  - **Tensors**: Multi-dimensional arrays used for data representation.
  - **Graphs**: Directed acyclic graphs (DAGs) that describe computations.
  - **Sessions**: Environments for running graphs and performing computations.

# Keras

- Keras is a **high-level API** for building and training deep learning models.
- It is designed to **be user-friendly, modular, and extensible**.
- **Originally developed as an independent** library, Keras **is now integrated into TensorFlow** as tf.keras.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

# Hidden Layers

- Transform input data into output using learned weights & activation functions.
- Capture complex patterns and relationships in the data.

# Hidden Layers

- Factors Influencing Number of Hidden Layers
  - **Complexity of the Problem:**
    - **Simple Problems:** Example: linear classification or basic regression
      - a single hidden layer with a reasonable number of neurons might suffice.
    - **Complex Problems:** Example: image recognition, NLP
      - deeper networks (more hidden layers) can capture more intricate patterns.
  - **Data Size and Quality**
    - Small Datasets:
      - Fewer layers may be needed to avoid overfitting.
    - Large Datasets:
      - More layers can be used to leverage the data effectively and capture complex features.
    - Computational Resources
      - **Limited Resources:** Fewer layers can help manage computational costs and reduce training time.
      - **High Resources**: More layers can be employed if resources permit, potentially improving model performance.

# Hidden Layers

- **Scenario 1**: Basic Classification Task
  - Task: Classify handwritten digits (MNIST dataset)
  - Network Architecture:
    - Input Layer: 784 neurons (for 28x28 pixel images)
    - Hidden Layers: 1 or 2 hidden layers. Hidden Layer Neurons: 128 or 256 neurons
    - Output Layer: 10 neurons (for 10 digit classes).
  - Example Configuration:
    - Input Layer → Hidden Layer 1 (256 neurons, ReLU) → Hidden Layer 2 (128 neurons, ReLU) → Output Layer (10 neurons, Softmax)
    - Reasoning: This architecture is sufficient for the MNIST dataset, which is relatively simple compared to more complex image recognition tasks.
- The number of neurons on each hidden layer also depends on similar facts.

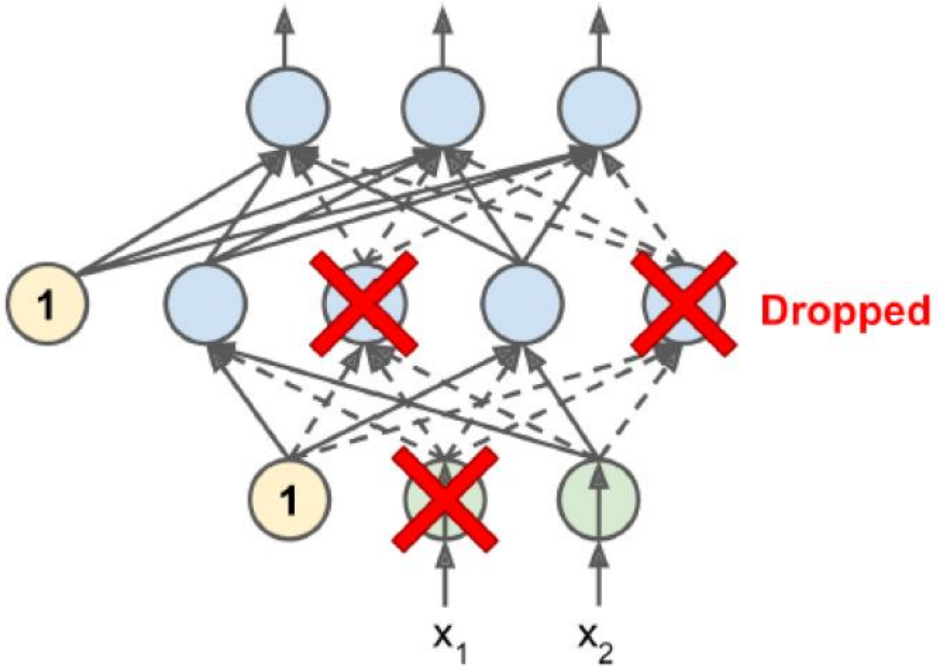
# Hidden Layers

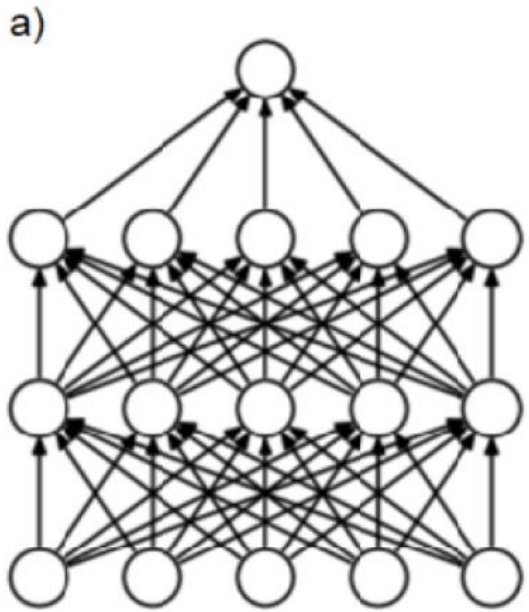
- **Scenario 2**: Complex Task: Recognize objects in images (e.g., CIFAR-10 dataset).
- Network Architecture:
  - Input Layer: 32x32x3 (for color images).
  - Hidden Layers: Several convolutional and fully connected layers.
    - Example Layers: Convolutional layers (e.g., 32, 64, 128 filters), pooling layers, dropout layers, followed by dense layers.
  - Output Layer: 10 neurons (for 10 classes).
  - Example Configuration: Input Layer → Conv Layer 1 (32 filters) → Pooling → Conv Layer 2 (64 filters) → Pooling → Dense Layer 1 (512 neurons) → Dense Layer 2 (256 neurons) → Output Layer (10 neurons, Softmax)
  - Reasoning:
    - This deeper architecture with multiple layers helps in capturing hierarchical features (edges, textures, shapes) necessary for complex image recognition tasks.

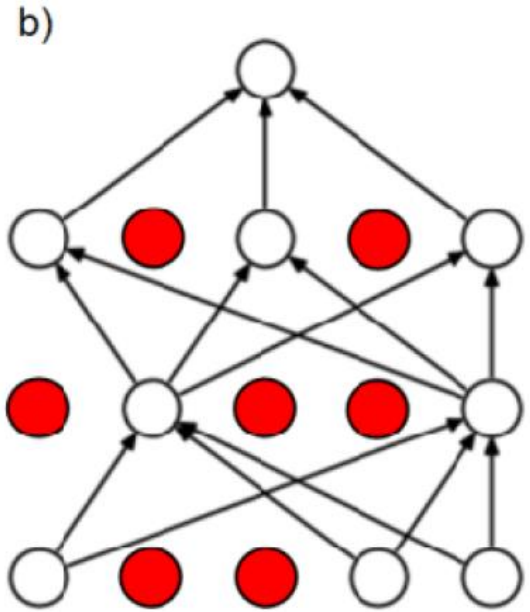
# Dropout Layers

- Dropout is a regularization technique where a randomly selected subset of neurons is ignored (dropped out) during each training iteration.
- This prevents the model from overfitting by ensuring that neurons do not co-adapt too much.
- How Dropout Works:
  - **Randomly Drop Neurons:** At each training step, a fraction of neurons (specified by the dropout rate) are randomly set to zero. This fraction is controlled by a parameter called the dropout rate ($p$).
  - **Scale the Remaining Neurons:** During training, remaining neurons are scaled up by a factor of $1/(1-p)$ to maintain the expected output level.
  - **During Inference:** Dropout is not applied during inference (testing or prediction). The full network is used without dropping any neurons.

# Dropout Layers



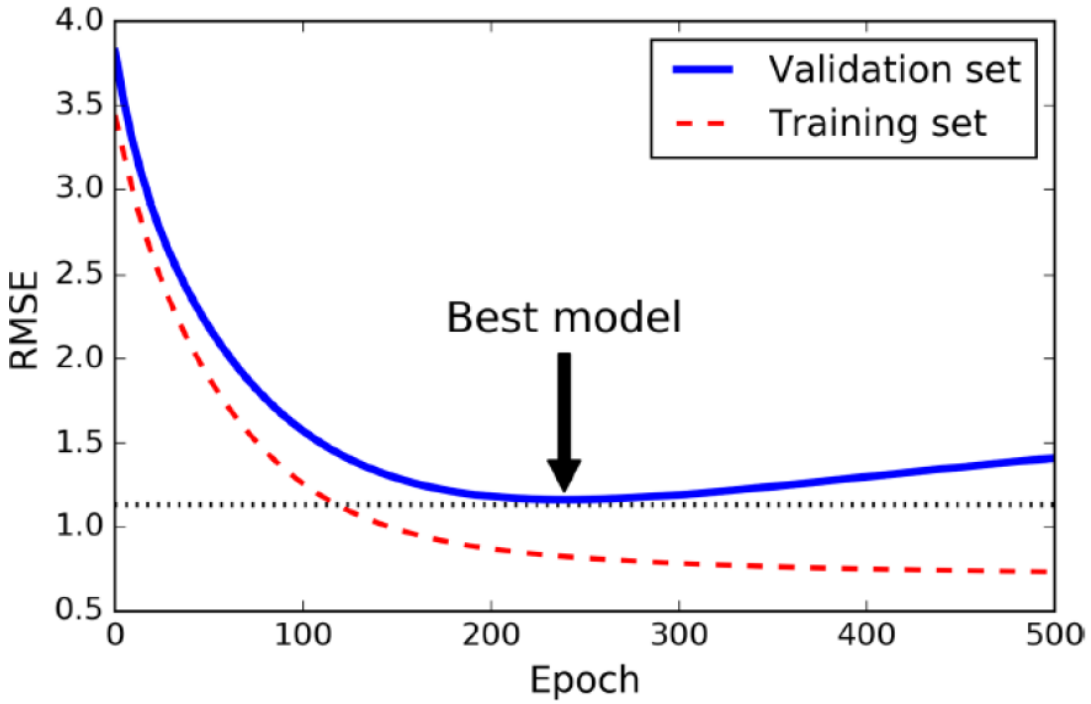
# Dropout Layers

- To implement dropout using TensorFlow, apply **dropout()** function to input layer and to output of every hidden layer (The code may vary from time to time).
- During training, this function randomly drops some items (setting them to 0) and divides the remaining items by the keep probability. After training, this function does nothing at all.

- *Refer to Google Colab Files on LMS for the example task on dropout*

# Early Stopping

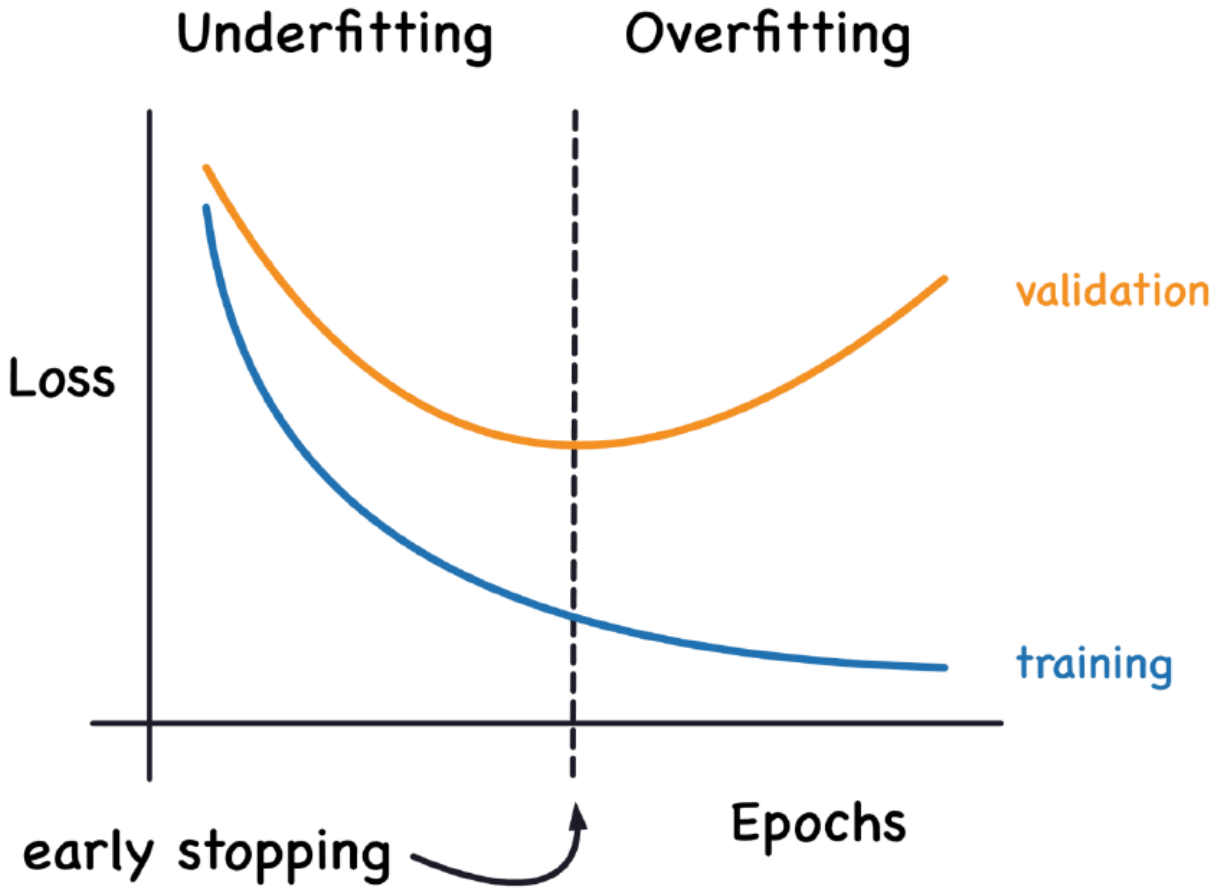
- A very different way to **regularize** iterative learning algorithms, such as Gradient Descent, is to **stop training** as soon as the validation **error reaches a minimum**. This is called **early stopping**.
- Figure 4-20 shows a complex model (in this case, a high-degree Polynomial Regression model) being trained using Batch Gradient Descent.



# Early Stopping

- In example, as the epochs go by, the algorithm learns and its prediction error (RMSE) on the training set naturally goes down, and so does its prediction error on the validation set.
- However, **after a while, the validation error stops decreasing** and actually starts to **go back up**.
- This indicates that the model has started to **overfit the training data**.
- With **early stopping**, you just **stop training as soon as the validation error reaches a minimum**.
- It is such a simple and efficient regularization technique that Geoffrey Hinton called it a “beautiful free lunch.”

# Early Stopping

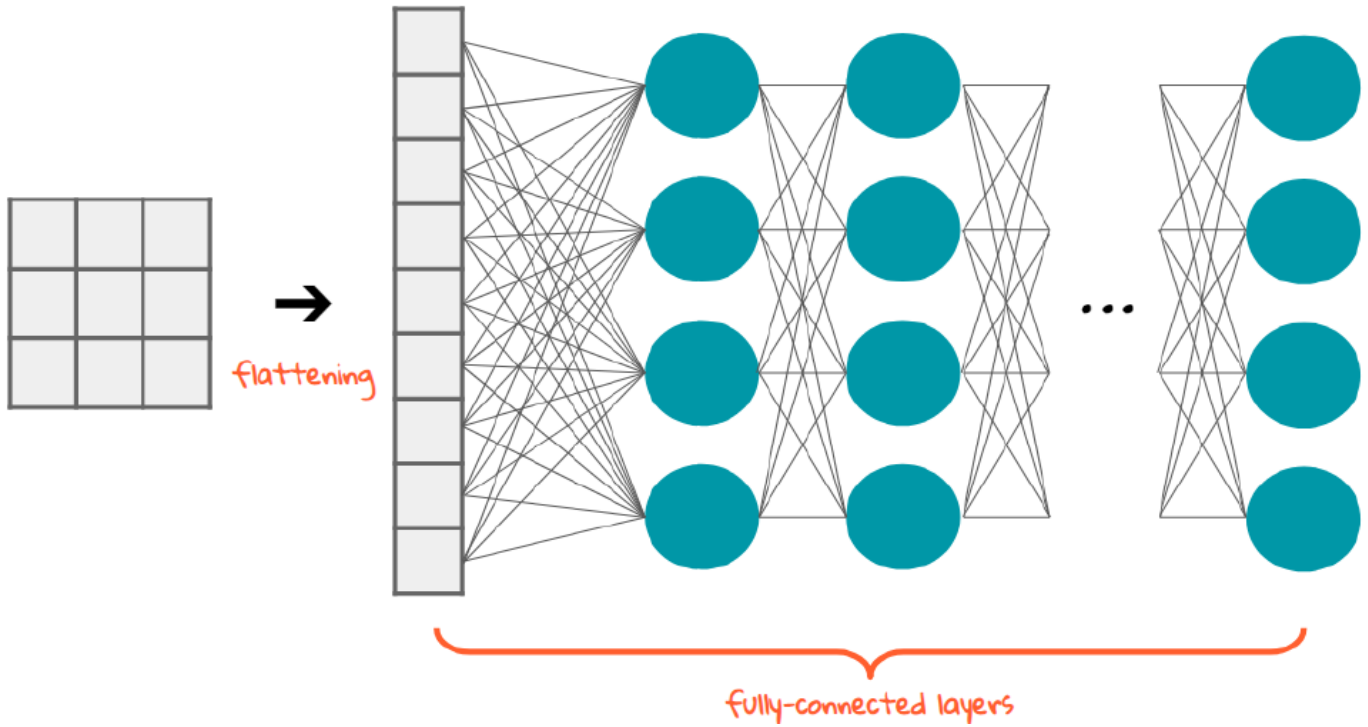


# Training, Testing, validation and k-fold cross validation

- **Training set**: Used to train the model and adjust its internal parameters (weights and biases).
- **Test set**: dataset used to assess the model's generalization ability after training is complete.
- **Validation set**: Used to evaluate the model's performance during training, adjust **hyperparameters** (Learning rate, Number of layers, network.Batch size: The number of training examples used in one forward/backward pass, Dopout rate, Number of epochs).
- **Cross-validation** is a technique where the dataset is split into multiple parts (folds) to perform training and validation multiple times across different splits of the data. Provides a more reliable estimate of model performance by reducing variability and ensuring the model is validated on different data portions.
- **Note**: K-fold cross-validation is **rarely used in deep neural networks** due to the massive computational cost, as it requires training the model k times. It **still can be used** in cases where the dataset is **small or the highest possible accuracy** is required.

# Flattening layer

- Flattening layer **converts a multi-dimensional** input (e.g., 2D images) **into a 1D vector**.
- After convolutional or pooling layers in image processing models, flattening is **used to transition to fully connected layers** (dense layers).



# Summary

- Discussed DNN implementation topics
- Implemented following
  - Binary classification
  - Multiclassification
  - Regression
- Discussed and implemented
  - Regularization with early stopping and dropout layer