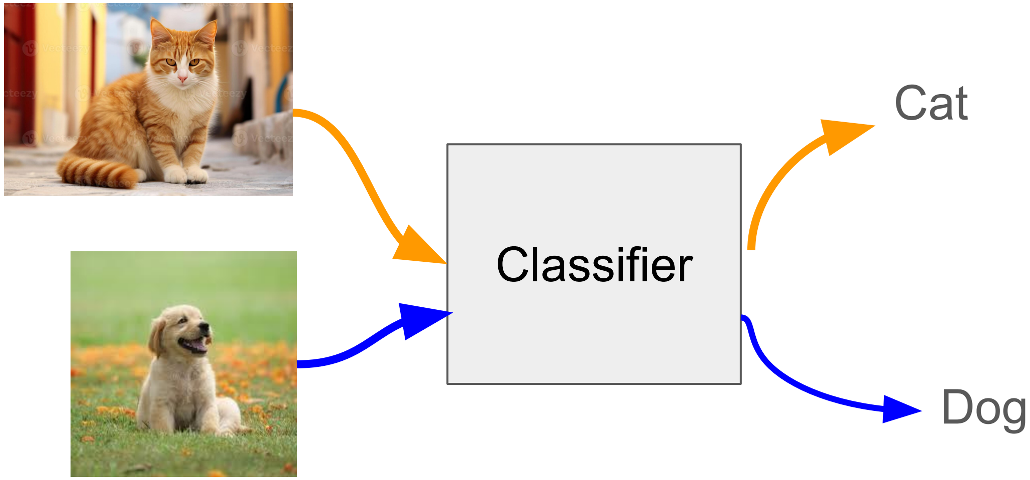
AI Lab

Lecture: 24.11.2024

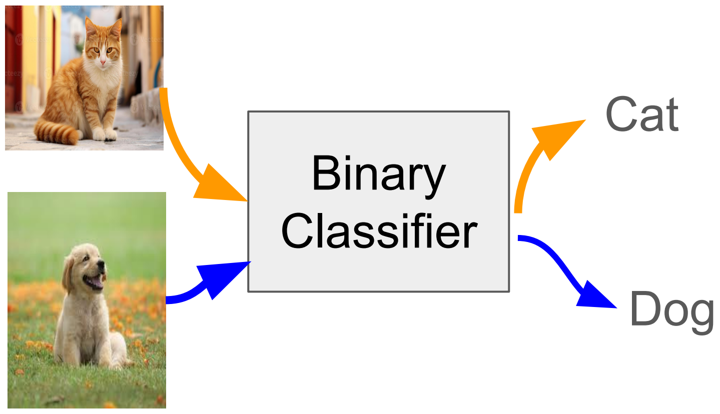
Sangeeta Biswas, Ph.D.
Associate Professor,
University of Rajshahi, Rajshahi-6205, Bangladesh

Classification

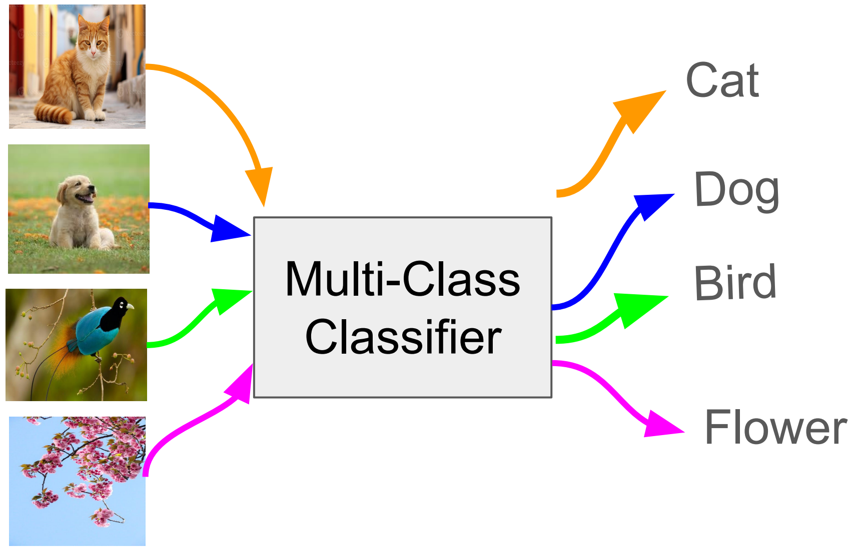
- Assigning objects to some pre-existing classes / categories / labels / groups.



Binary-Class Vs. Multi-Class Classification



Number of classes = 2



Number of classes = 4

Artificial Neural Network

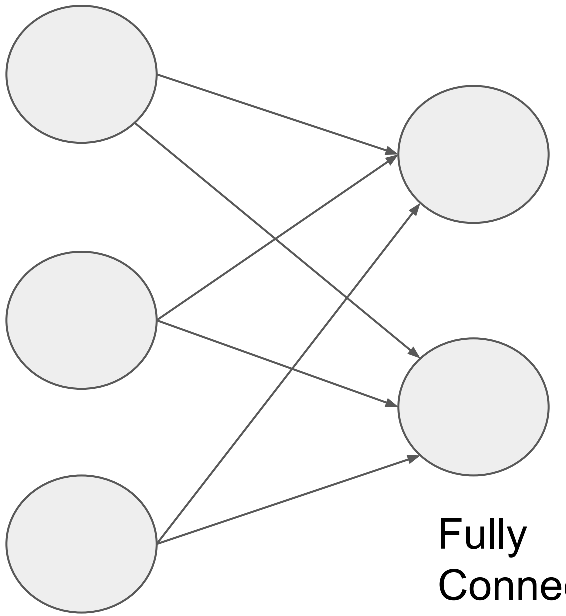
- An artificial neural network (ANN) is a machine learning algorithm that uses a network of interconnected artificial neurons to process data to imitate our human brain.
- All ANNs have three kinds of layers:
  - **One** Input Layer: to receive data.
  - **Multiple** Hidden Layers: to process data
  - **One** Output layer: to produce the output.
- Shallow NN is a type of ANN with a few hidden layers, usually one or two.
- Deep NN (DNN) is a type of ANN having multiple hidden layers to solve complex problems.
  - GPT-3: 96 hidden layers
  - EfficientNet: 5-400 hidden layers
  - ResNet-152: 152 hidden layers

Fully Connected Layer

A fully connected layer connects every neuron in one layer to every neuron in its previous layer.

An image need to be turned into a vector before feeding into an fully connected layer.

Flatten() is used in Tensorflow.keras



Fully
Connected
Layer

Fully Connected Neural Network (FCNN)

FCNN consists of a series of fully connected layers

It is also known as multi-layer perceptron (MLP).

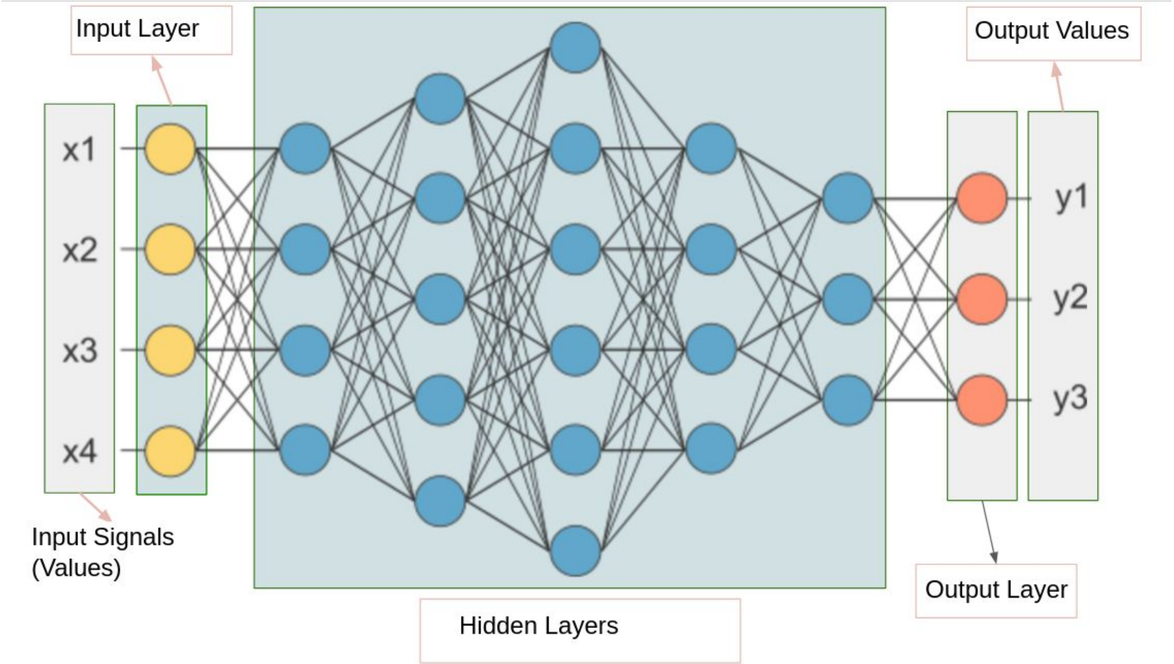


Image Source: Google Search Engine

Hidden Layers

- The number of hidden layers in a DNN depends on the complexity of the data:
    - **Linearly separable data**: No hidden layers are needed
    - **Less complex data**: 1–2 hidden layers are sufficient
    - **Large data**: 3–5 hidden layers are recommended
    - **Complex data**: Additional layers can be helpful
- Depth of an ANN = the number of hidden layers + the output layer
- Sometimes a deeper network have better performance comparing to a shallower network
- A DNN has risks of
    - experiencing vanishing gradient problem
    - high sensitivity to input data
    - having similar performance of a shallow network.

Optimum Number of Hidden Layers

- **Increasing the number of hidden layers**
  - Can increase the performance of the network.
- **Adding too many hidden layers**
  - Can lead to overfitting, where the model memorizes the training data but doesn't generalize well to new data.
- **Reducing the number of hidden layers**
  - Can directly impact the accuracy of the network. For complex problems, the network might not be trained properly with fewer hidden layers.
- **Optimal number of layers depends on several factors including:**
  - the data
  - the optimizer, and
  - the network's architecture

Number of Neurons

- Number of neurons in the:
  - input layer depends on the size of the input data
    - for 28x28 input image, 784 neurons in the input layer
  - output layer depends on the size of the output data
    - in a binary classifier, one or two neurons in the output layer
    - in a 10 class classifier, four or ten neurons in the output layer
  - hidden layers depends on us, i.e., who design the architecture of the NN
- More neurons means:
  - more parameters for tuning
  - more time for tuning parameters
  - more data for avoiding overfitting
  - more hardware support, i.e., computational power & memory during parameter tuning, and storage space for saving model.

Parameter Vs Hyperparameter

Parameters: which NN will learn from data

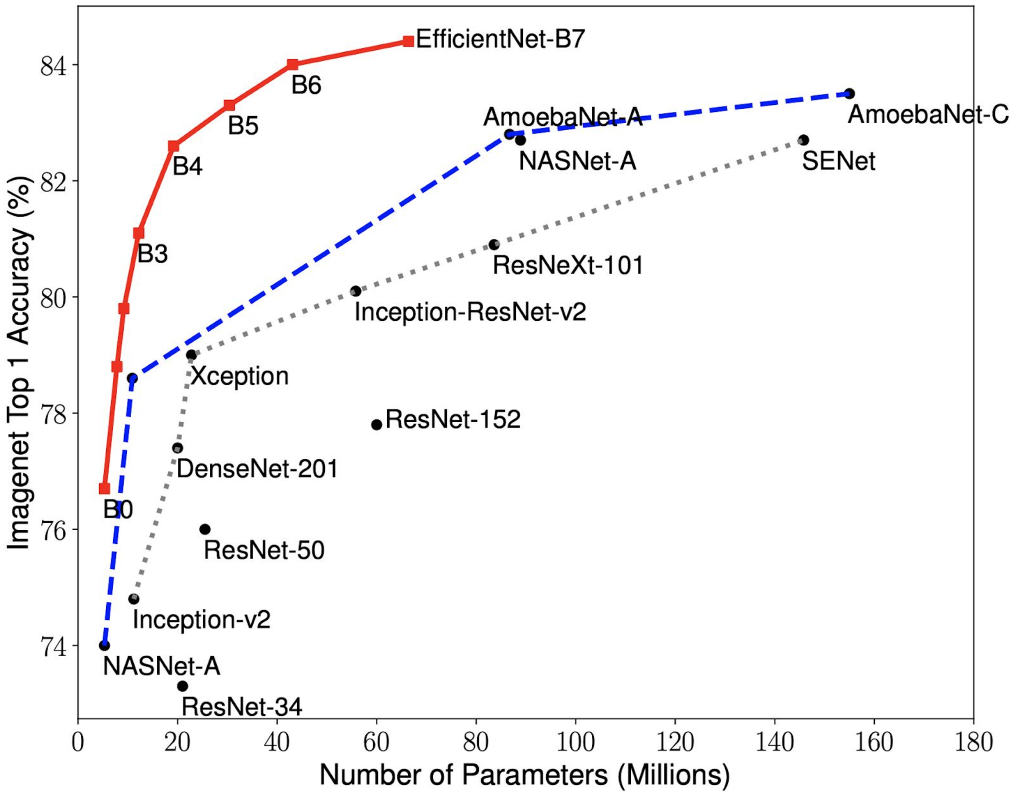
- $\boldsymbol{w}$, $\boldsymbol{b}$, any variables in $\boldsymbol{g}(.)$

Hyperparameters: which we need to decide based on our experience, intuition or error-trial such as

- number of layers
- number of neurons in each layer
- activation function
- epoch number
- learning rate

Performance Vs Parameters

In general, a model having more parameters performs better.



Keep in Mind

- Designing a suitable DNN automatically is not fully explored yet.
  - Recent Works: Neural Architecture Search—Finding the Best Model Design Automatically
- Designing a suitable network architecture is still a error-and-trial process.
- Not an architecture optimum for a data will be optimum for all datasets.
- It is upto us what will be the number of hidden layers and number of neurons in each hidden layer. However, we need to be careful.
- We should not design or work with any network which:
  - cannot be run in our available hardware support.
  - can be easily overfitted on our available training data.

Activation Function

- An activation function is a mathematical equation that determines how much data should be passed from a neuron to the next neuron.
- It is the function that is applied on the weighted sum of the input of the neuron, i.e., it is $g(u)$.
- Generally, it is a nonlinear function for a neuron in the hidden layer and linear/non-linear function for neurons in the output layer.
- A linear activation function, also known as "no activation" or "identity function," is directly proportional to the input.
- Popular non-linear activation functions are:
  - Sigmoid, Tanh, ReLU, ELU, Softmax

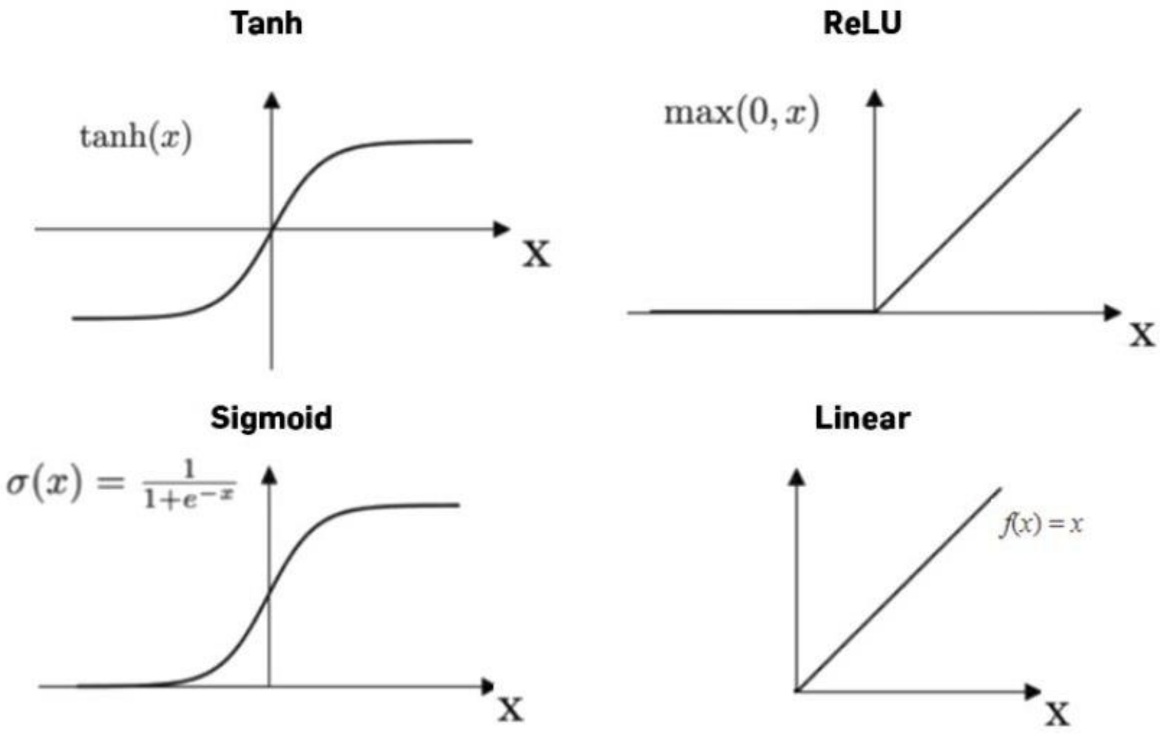
Some Popular Activation Functions

- **Sigmoid**
  a. Also known as the logistic activation function
  b. Often used for models that predict probability as an output.
  c. Its curve looks like an S-shape and exists between 0 and 1.
  d. Suffers from saturating gradients problem.
- **Hyperbolic tangent (Tanh)**
  a. Has stronger gradients than the sigmoid function, and its output ranges from -1 to +1.
  b. Helps the learning algorithm converge faster.
- **Rectified linear unit (ReLU)**
  a. A non-saturating function, meaning it doesn't become flat at the extremes of the input range.
  b. It is faster than sigmoid and tanh.
- **Softmax**
  a. Converts vectors of real numbers into a probability distribution.
  b. Each output value represents the probability that the input belongs to a specific class.

Activation Functions

- Linear activation function in the output layer generally used for regression problem.

- Depending on the range of output values, we need to choose activation function.



Activation Functions for Classification

In output layer, generally:

- Sigmoid is used for binary classification
- Softmax is used for multi-class classification

Both generate values in the range of 0-1.

Summation of softmax values is 1.

Summation of sigmoid values in a classifier does not need to be 1.

$$sigmoid,\ y_i = \frac{e^{x_i}}{1 + e^{x_i}}$$

$$softmax,\ y_i = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}$$