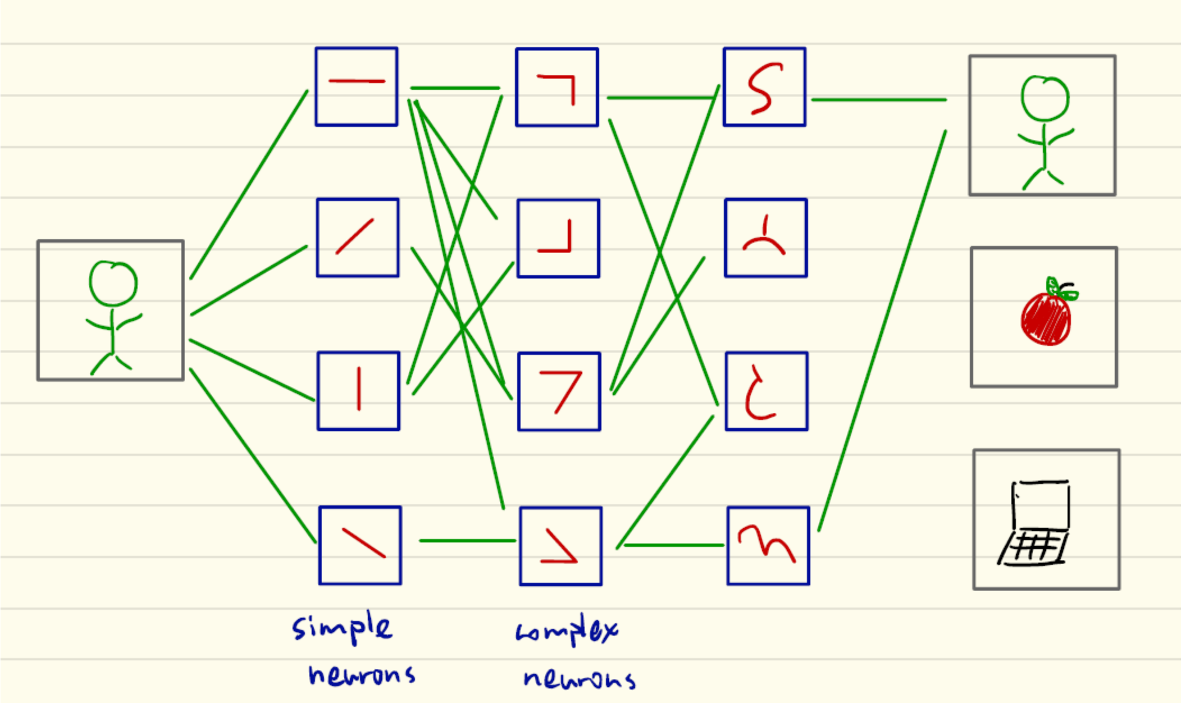
AI Lab for Wireless Communications

Week3 – Deep Learning

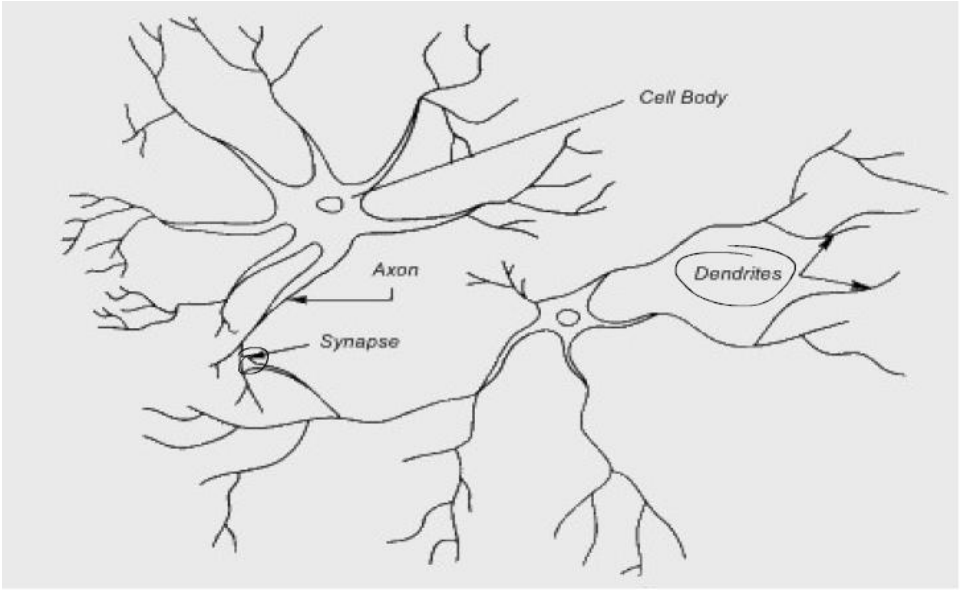
What is Deep Learning?

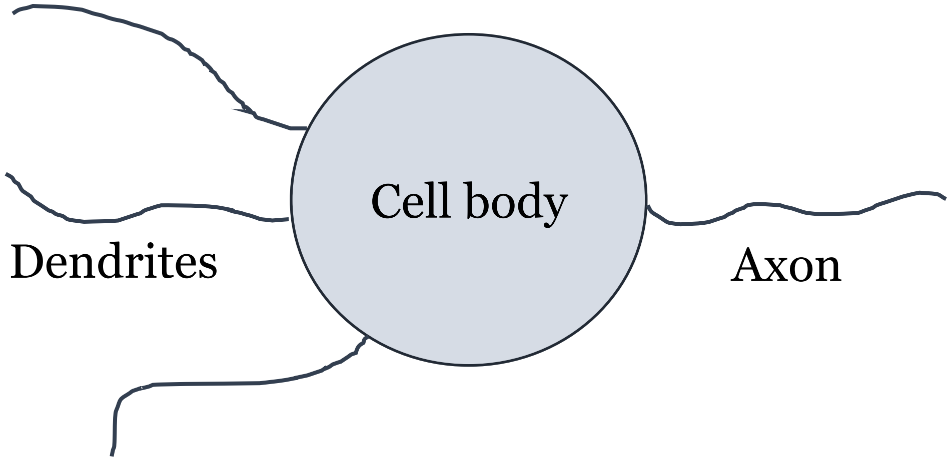
- Deep learning is a branch of ML and is based on neural network (NN)
- In recent years, deep learning is popular because of image processing.



Biological Neuron

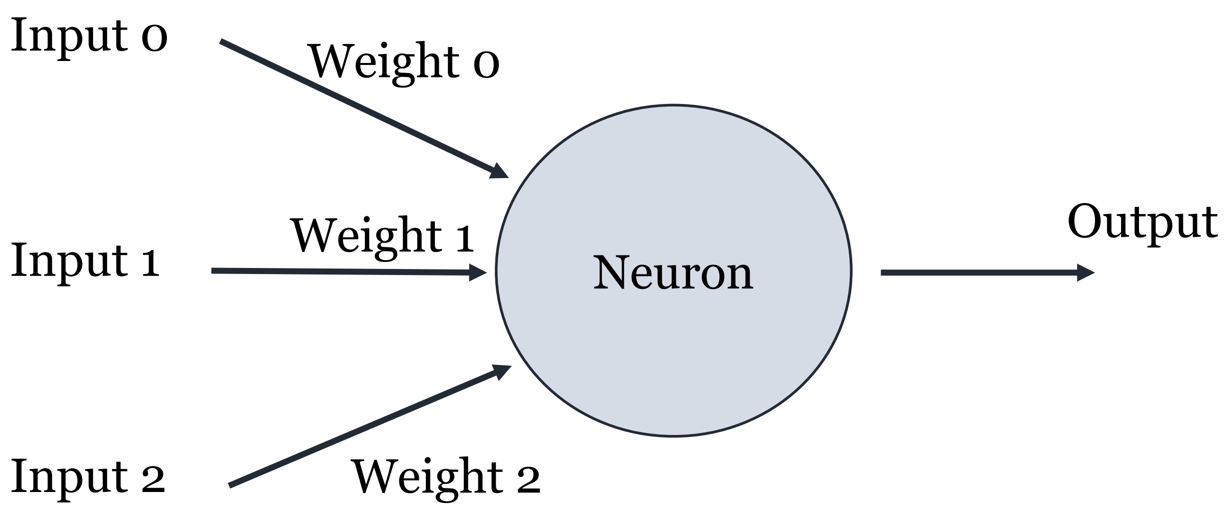
- Several neurons are connected to one another to form a neural network or a layer of a neural network





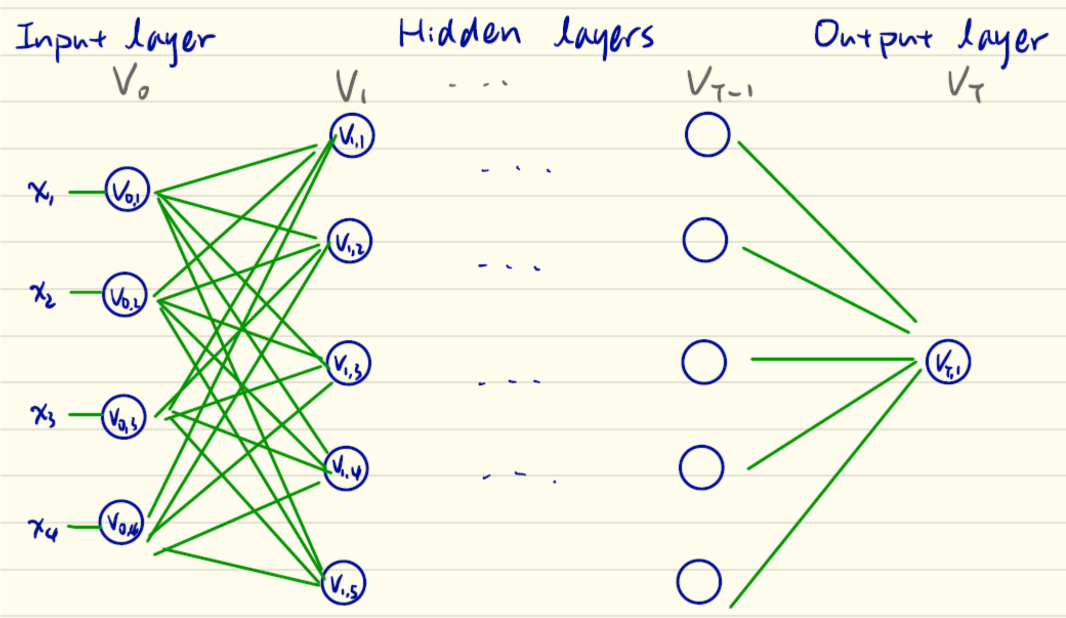
Artificial Neuron

- Artificial neuron closely mimics the characteristics of biological neuron

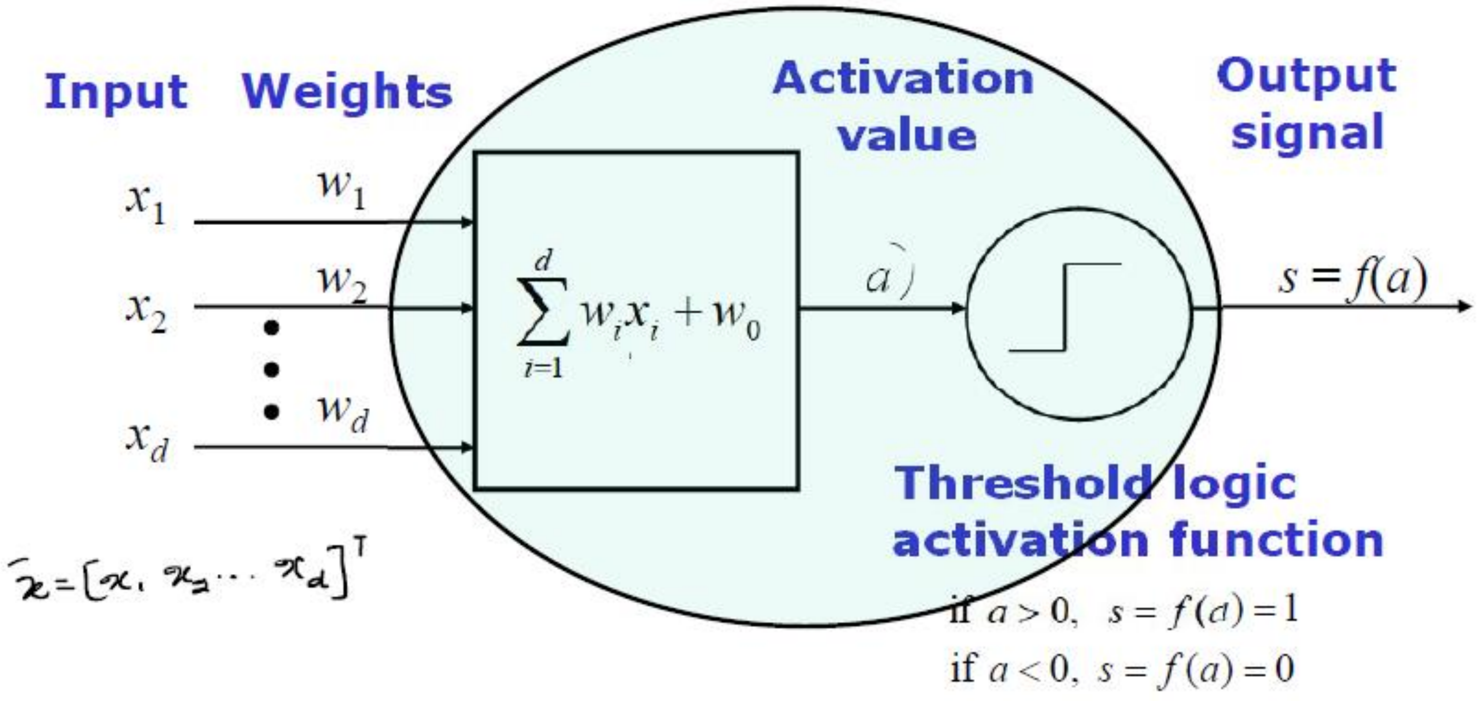


Deep Neural Network

- $x_1 \sim x_n$: Inputs, V_o: Input layer, $V_1 \sim V_{T-1}$: hidden layers, V_T: output layer
- T: Depth of the network
- Deep NN or Deep learning: if T>2
- Associate with each edge is a weight $W(V_{t,r}, V_{t+1,r}, j)$



Neuron with Threshold Logic Activation Function



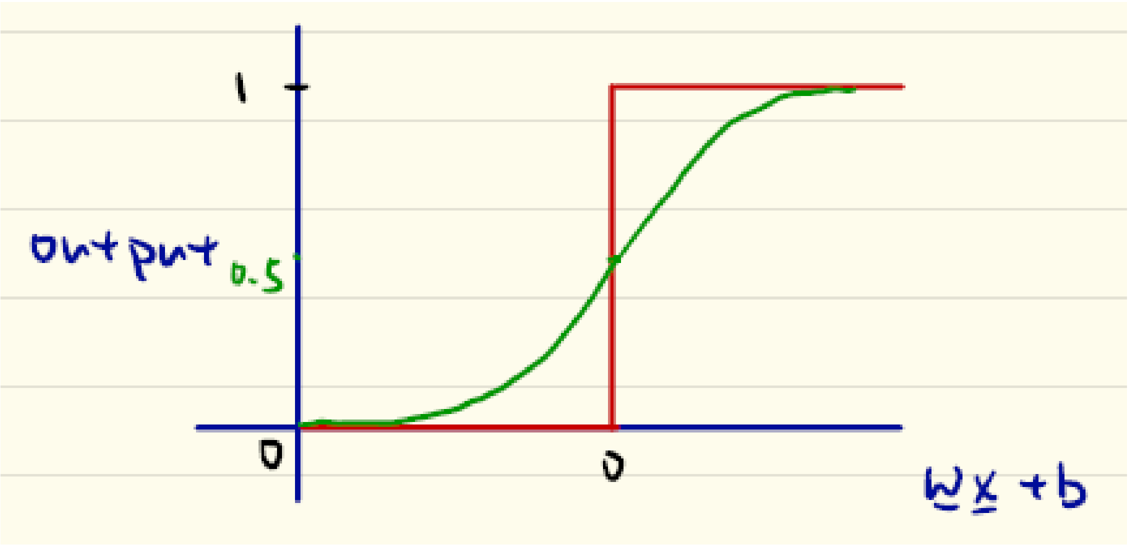
W.S.McCulloch and W.Pitts. A logival calculus of the ideas imminent in nervous activity. 1943.

Activation Function

- Sigmoid function
- Hyperbolic tangent function
- ReLu (Rectified Linear Units)

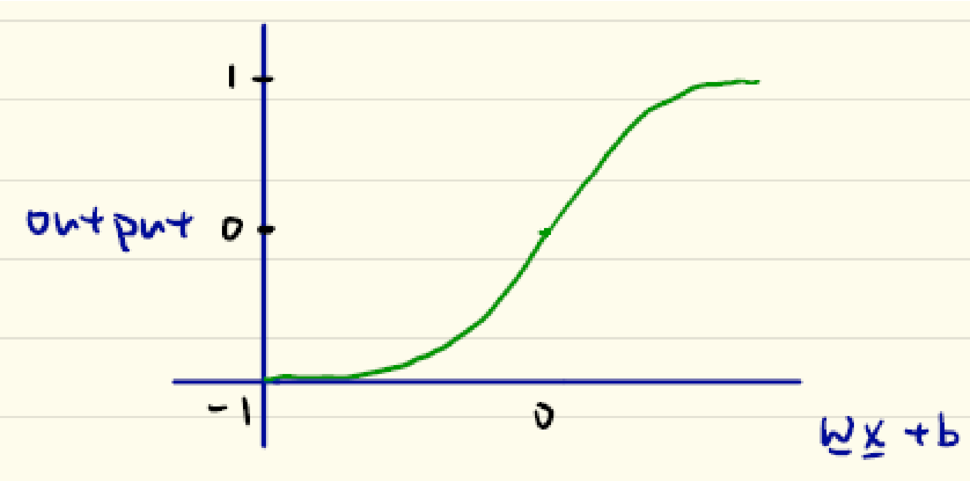
Sigmoid Function

- $\sigma(z) = \frac{1}{1+e^{-z}}$
- It outputs soft value in (0,1)
- $\sigma(z) \to 0 \; as \; z \to -\infty$
- $\sigma(z) = \frac{1}{2}$ if $z = 0$
- $\sigma(z) \to 1 \; as \; z \to \infty$



Hyperbolic Tangent Function

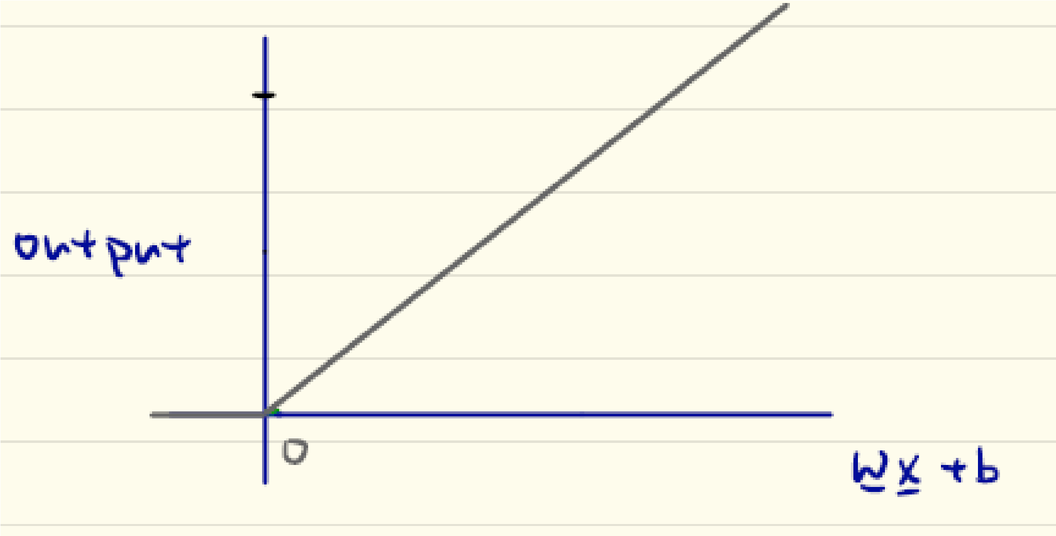
- $\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- Very similar to sigmoid, but its range is (-1,1)



- Issue in sigmoid and Tanh: They saturate!

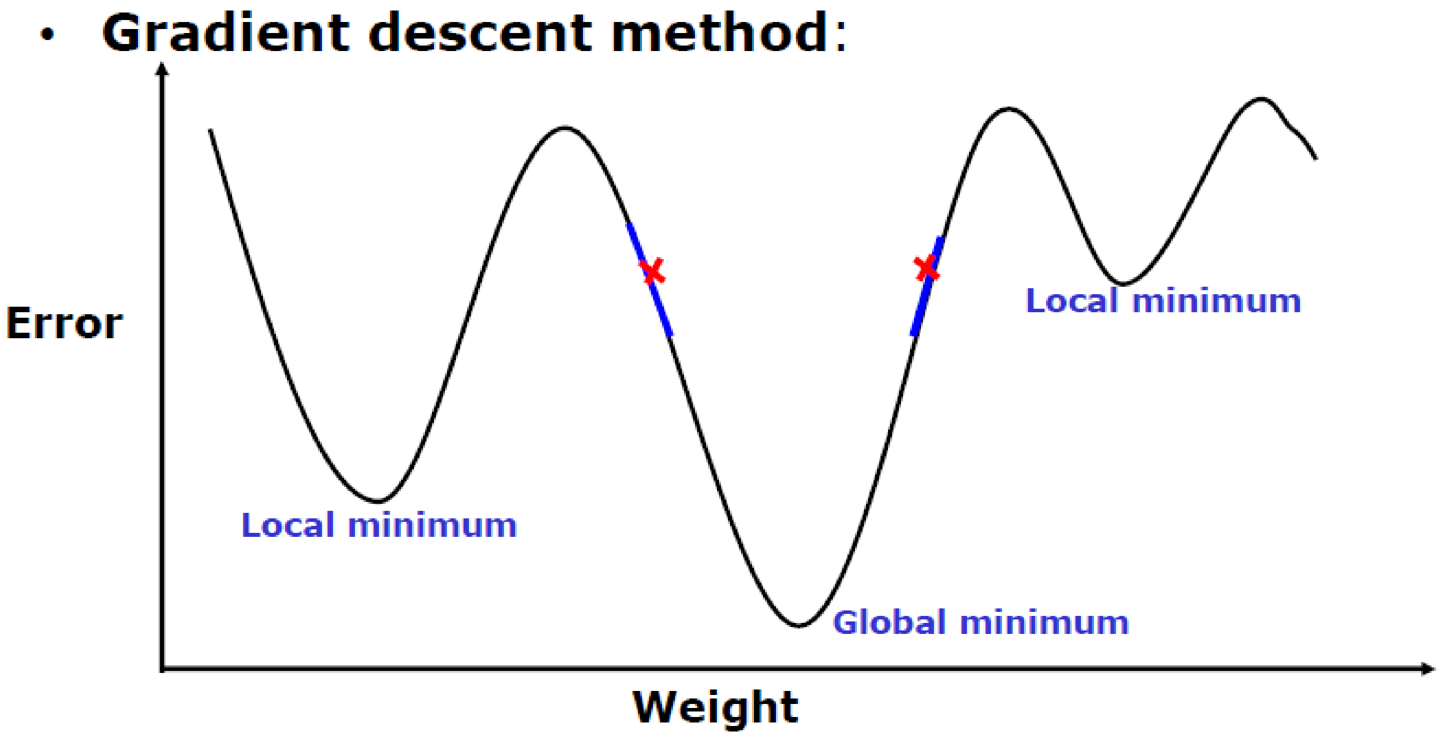
ReLu (Rectified Linear Units)

- $\sigma(z) = \max(0, \mathrm{z})$
- Super simple, do not saturate
- Most widely used for hidden layers

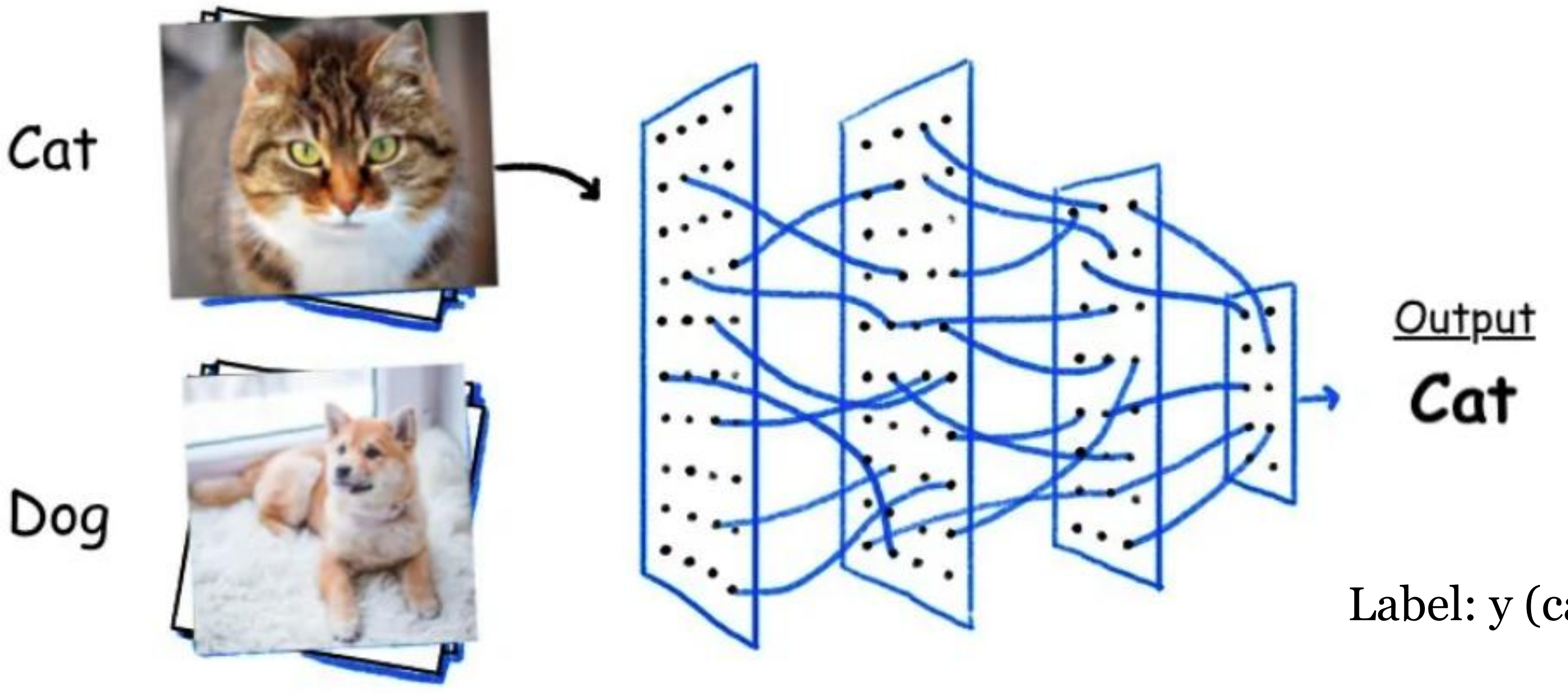


Parameter (Weights) Learning

- Gradient Descent Method, Stochastic Gradient Descent is often be used.



Example – DNN for Image Classification



Label: y (cat or dog)

Image data: x

Build Model

- Define model structure

```
# Define the four-layer model
model = keras.Sequential([
    layers.Dense(256, activation='relu', input_shape=(784,)),  # First hidden layer
    layers.Dense(128, activation='relu'),  # Second hidden layer
    layers.Dense(64, activation='relu'),   # Third hidden layer
    layers.Dense(10, activation='softmax')  # Output layer (10 classes)
])
```

- Check your model by

model.summary()

https://keras-zh.readthedocs.io/getting-started/sequential-model-guide/

Compile Model

- Compile the model

```
# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

- Different optimizer and loss function may provide different performance

Training and Inference the Model

- Given the training data, testing data and epochs. We start to train the model which we defined.
- We calculate the validation error at the same time, but it does nothing in our training progress.

```
# Train the model
model.fit(x_train, y_train, epochs=5, batch_size=64, validation_data=(x_test, y_test))
```

- Apply the trained model do the prediction

test_y = model.predict(test_x)

Implement DNN Decoding System

- Formulate decoding problem as a classification problem
- Implement procedure
  - ➢Build model
  - ➢Compile model
  - ➢Training/Fitting the model with training data
  - ➢Predict with test data
  - ➢Calculate the error

Lab for Today

- Formulate decoding problem as a classification problem
- Implement a DNN decoder for SNR=0~7
- Demo (SNR=7)
  - Model requirement
    - Epoch ≥ 5
    - # of layers ≥ 3
    - Input layer – 7 nodes
    - Output layer – 16 nodes

Top 1/3:	100
Top 2/3:	95
BLER $\leq 10^{-3}$	90