AI VIETNAM
Free AI Course 2020

Multi-layer Perception

(Draft Version)

Quang-Vinh Dinh
Ph.D. in Computer Science

Year 2020

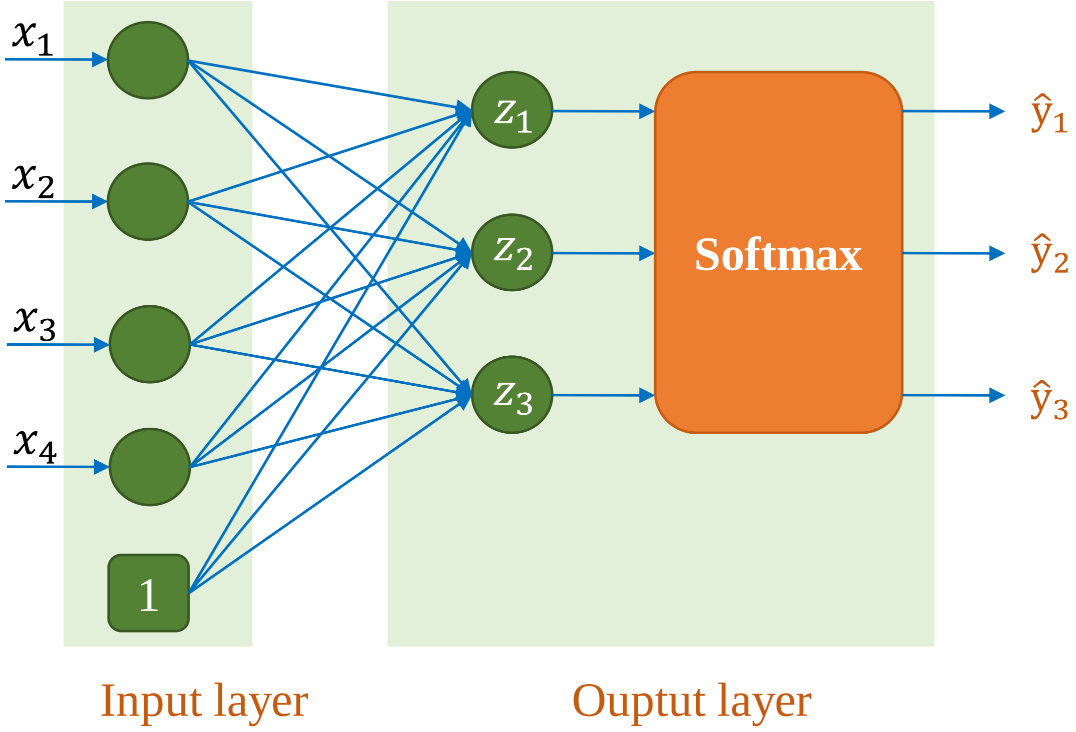
Outline

- Multi-layer Perceptron
- To-do List for Training
- Forward Computation Example
- Image Classification: Fashion-MNIST
- Image Classification: Cifar-10
- Underfitting and Overfitting

Multi-layer Perceptron

❖ Softmax regression

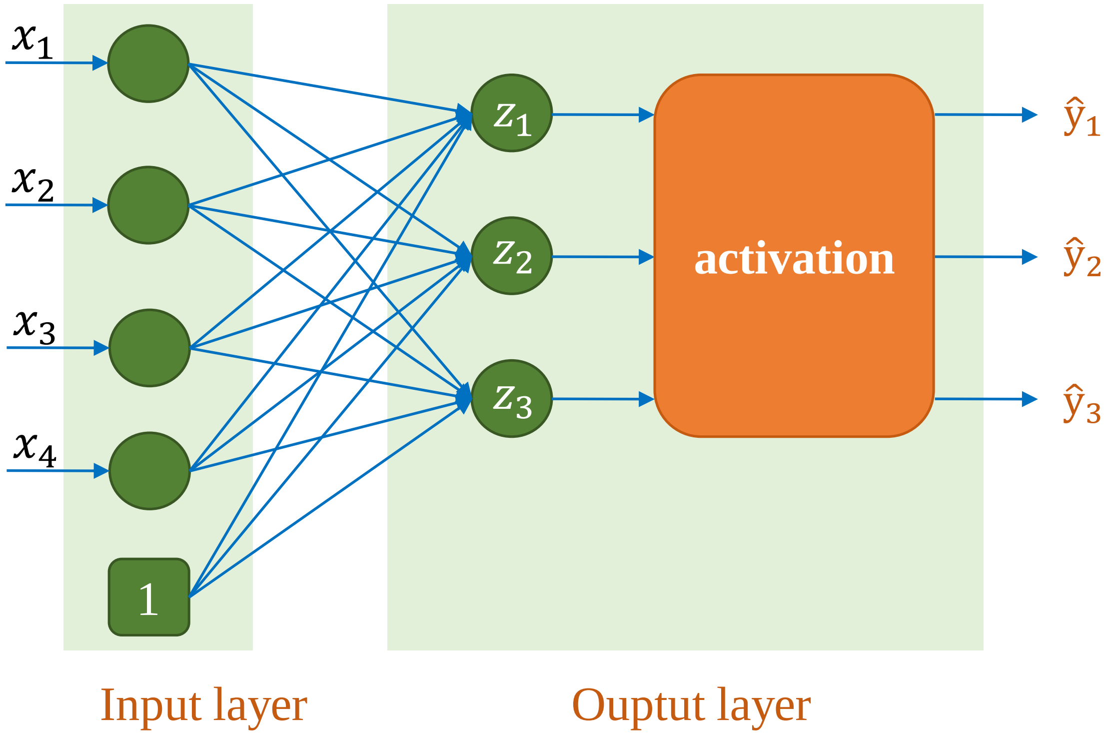
Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Label
5.2	3.5	1.5	0.2	1
5.2	3.4	1.4	0.2	1
4.7	3.2	1.6	0.2	1
6.3	3.3	4.7	1.6	2
4.9	2.4	3.3	1.1	2
6.6	2.9	4.6	1.3	2
6.4	2.8	5.6	2.2	3
6.3	2.8	5.1	1.5	3
6.1	2.6	5.6	1.4	3

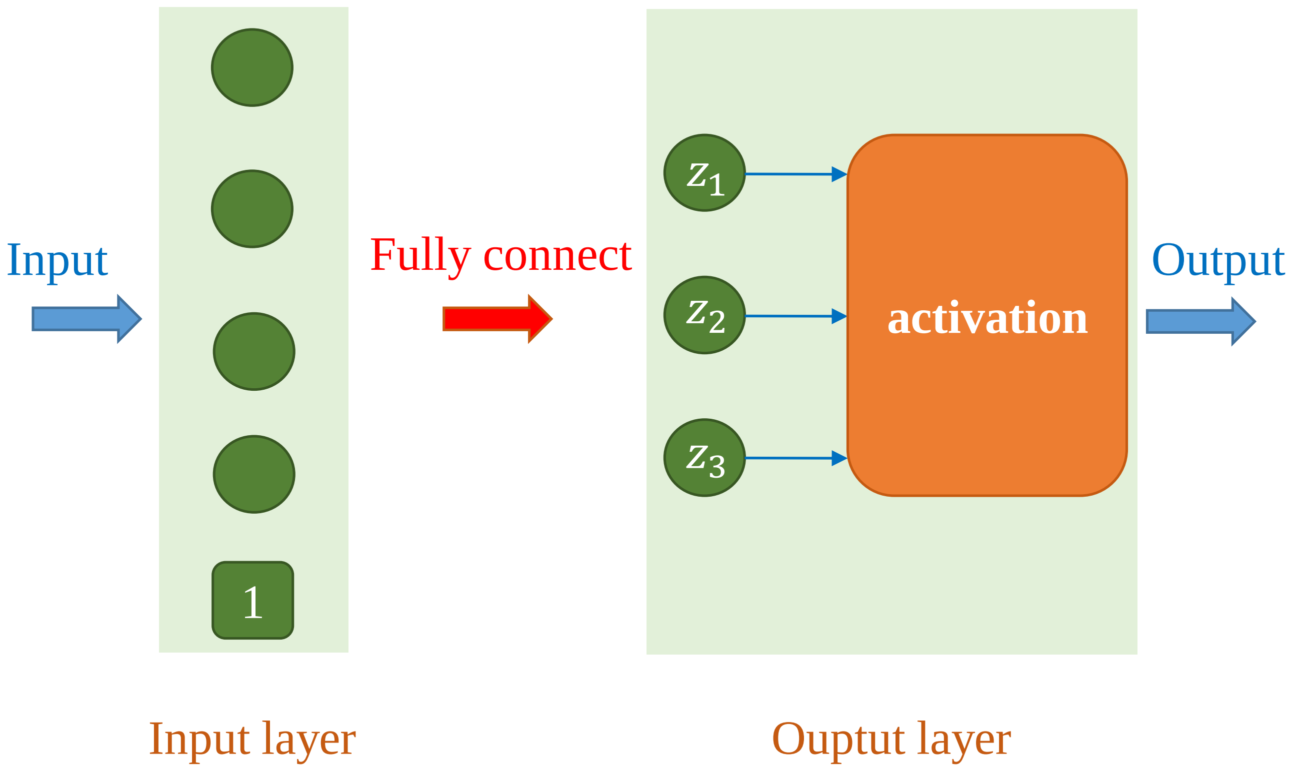


AI VIETNAM
Free AI Course

Multi-layer Perceptron

❖ Softmax regression

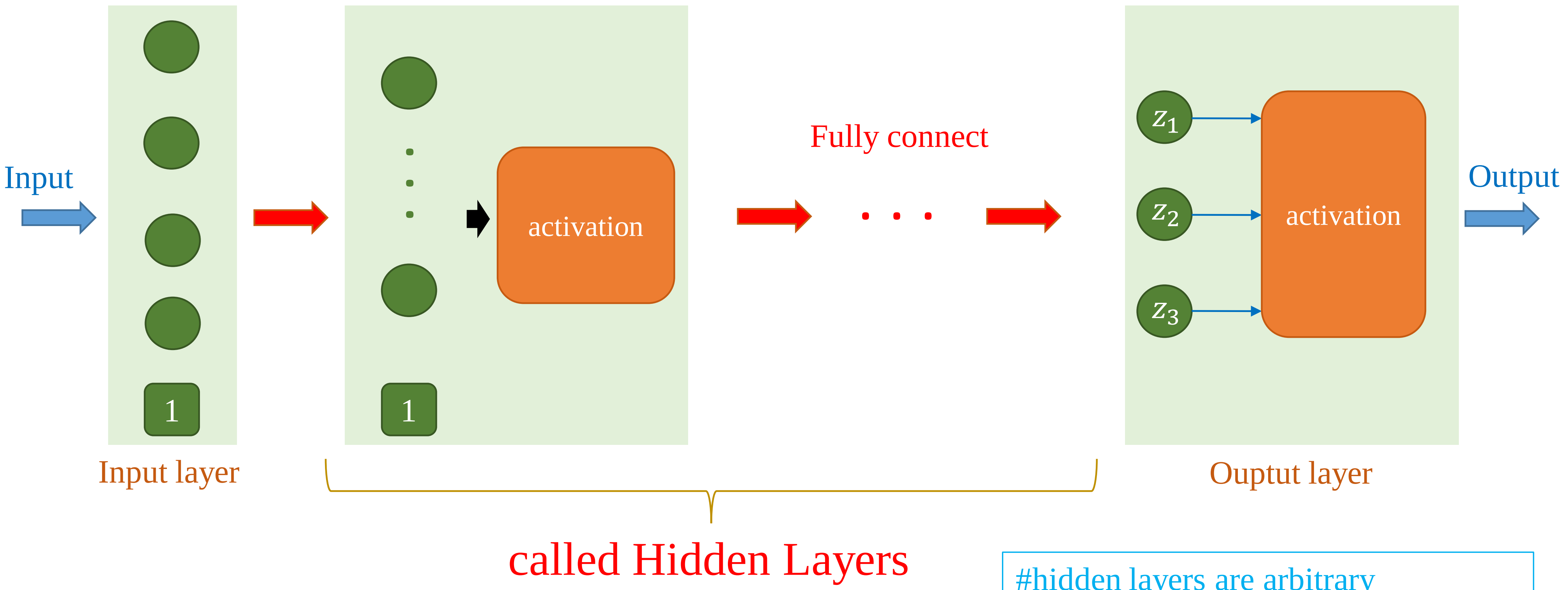




Multi-layer Perceptron

❖ **An idea: More parameters → better capacity (~stronger model)**

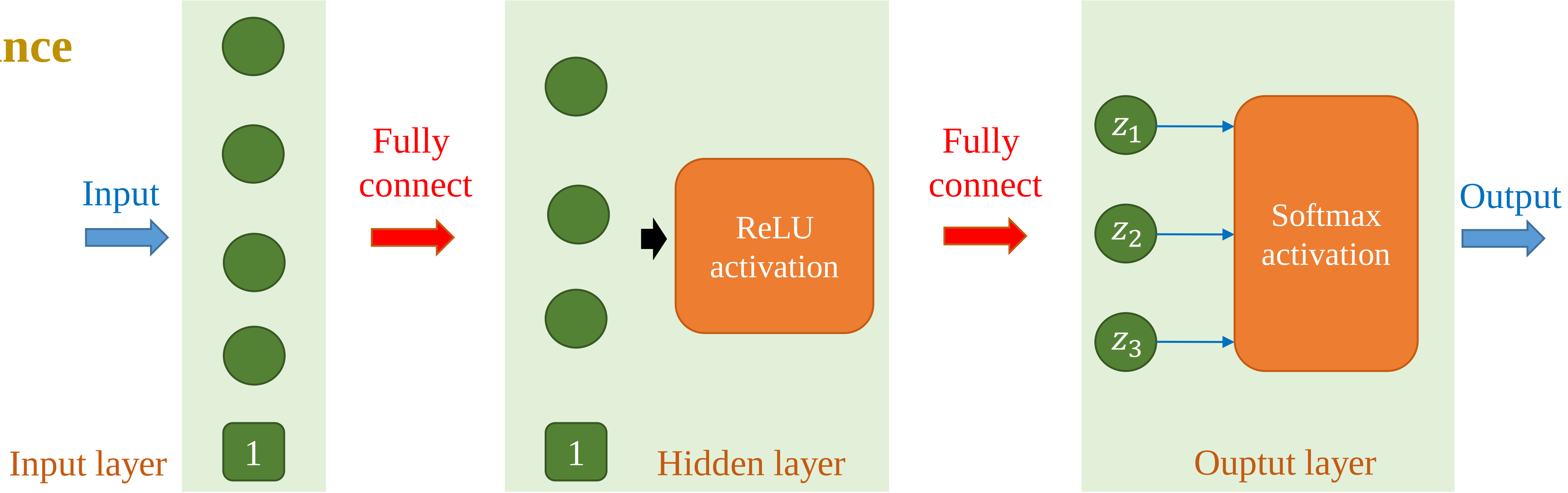
- ❖ **Adding more layers**



#hidden layers are arbitrary
#nodes in a hidden layer are arbitrary

Multi-layer Perceptron

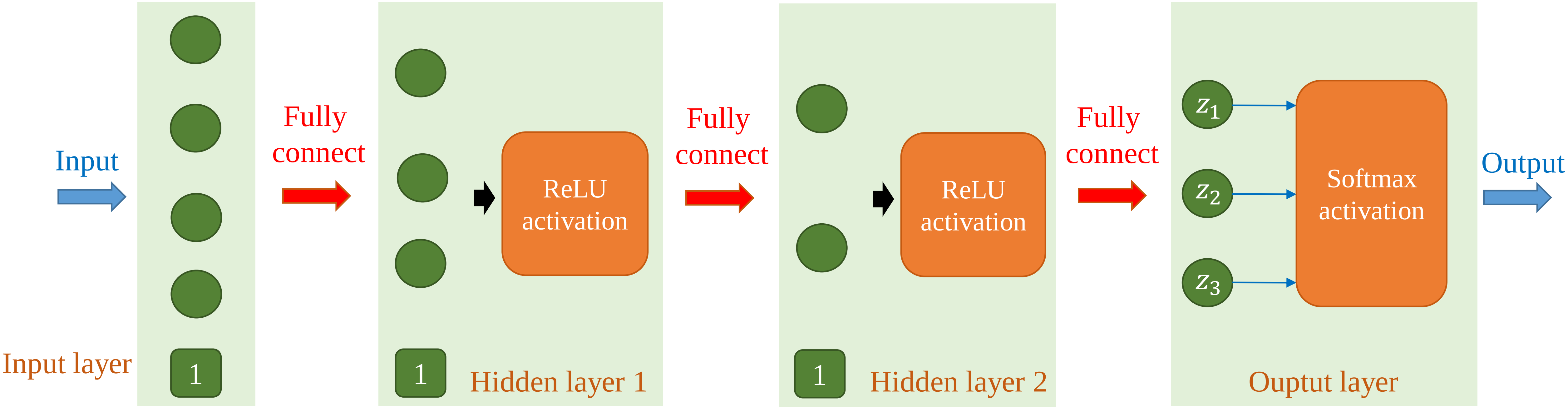
An instance




```python
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(4,)))
model.add(keras.layers.Dense(3, activation='relu'))
model.add(keras.layers.Dense(3, activation='softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 3)                 15
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 12
=================================================================
Total params: 27
Trainable params: 27
Non-trainable params: 0
```

Multi-layer Perceptron




```python
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(4,)))
model.add(keras.layers.Dense(3, activation='relu'))
model.add(keras.layers.Dense(2, activation='relu'))
model.add(keras.layers.Dense(3, activation='softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 3)                 15
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 8
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 9
=================================================================
Total params: 32
Trainable params: 32
Non-trainable params: 0
```

Outline

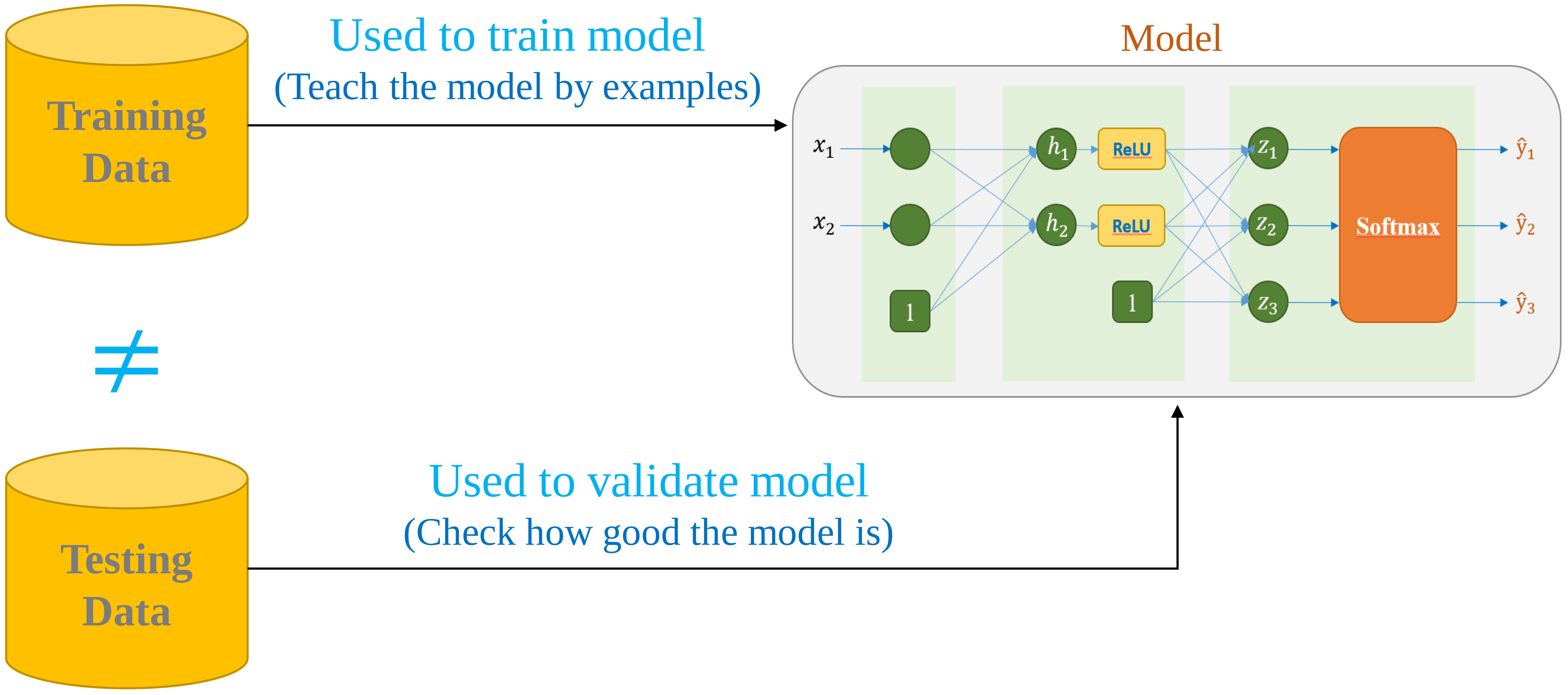
- Multi-layer Perceptron
- To-do List for Training
- Forward Computation Example
- Image Classification: Fashion-MNIST
- Image Classification: Cifar-10
- Underfitting and Overfitting

To-do List for Training

- Data Preparation
- Data Normalization
- Model (Network) Construction
- Parameter Initialization
    - Optimizer Selection
    - Loss function Selection
    - Metric Selection

To-do List for Training

Data Preparation



To-do List for Training

Data Normalization

Convert to the range [0,1]

$$Image = \frac{Image}{255}$$

Convert to the range [-1,1]

$$Image = \frac{Image}{127.5} - 1$$

Z-score normalization

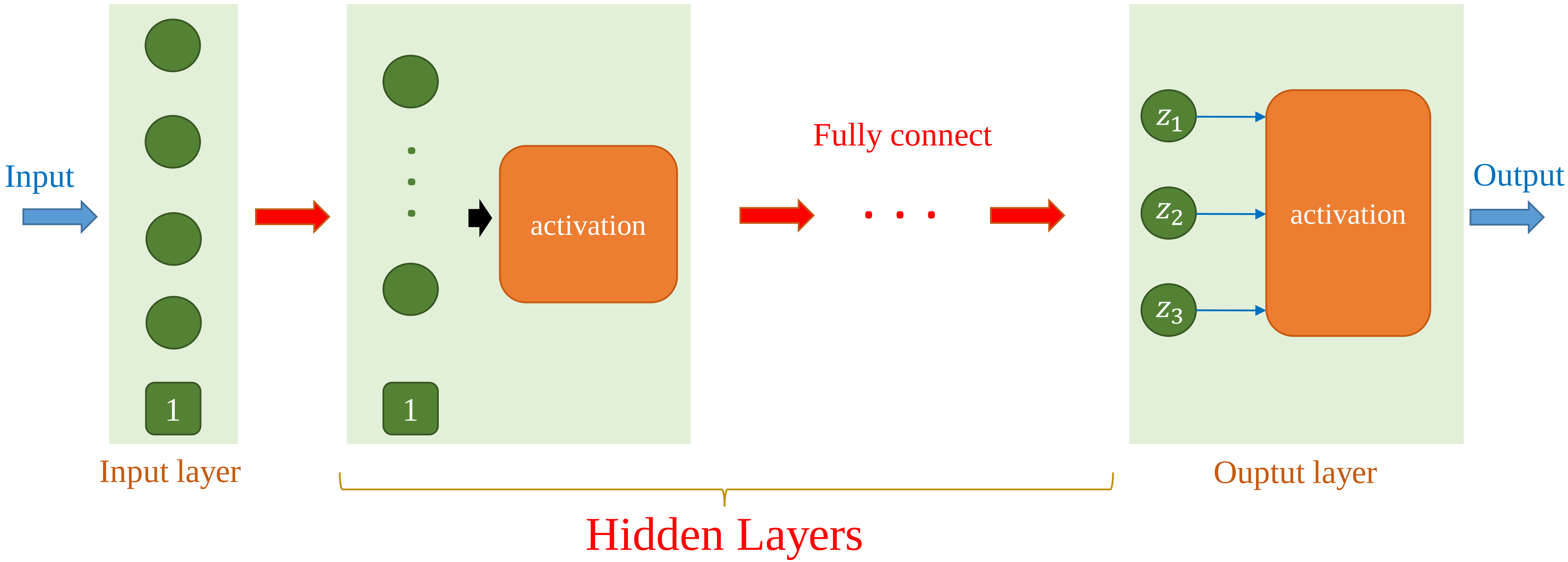
$$Image = \frac{Image - \mu}{\sigma}$$

μ is the mean of
the image (or training data)

σ is thestandard deviation
of the image (or training data)

To-do List for Training

Model (Network) Construction



How many hidden layers?
How many nodes in a hidden layer?

Which activation function?
Which network components?

To-do List for Training

Model (Network) Construction

Which activation function?

$$\mathrm{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

$$\mathrm{ELU}(x) = \begin{cases} \alpha(e^{x} - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

$$\mathrm{PReLU}(x) = \begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

$$\mathrm{softplus}(x) = \log(1 + e^{x})$$

To-do List for Training

❖ Tanh function

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

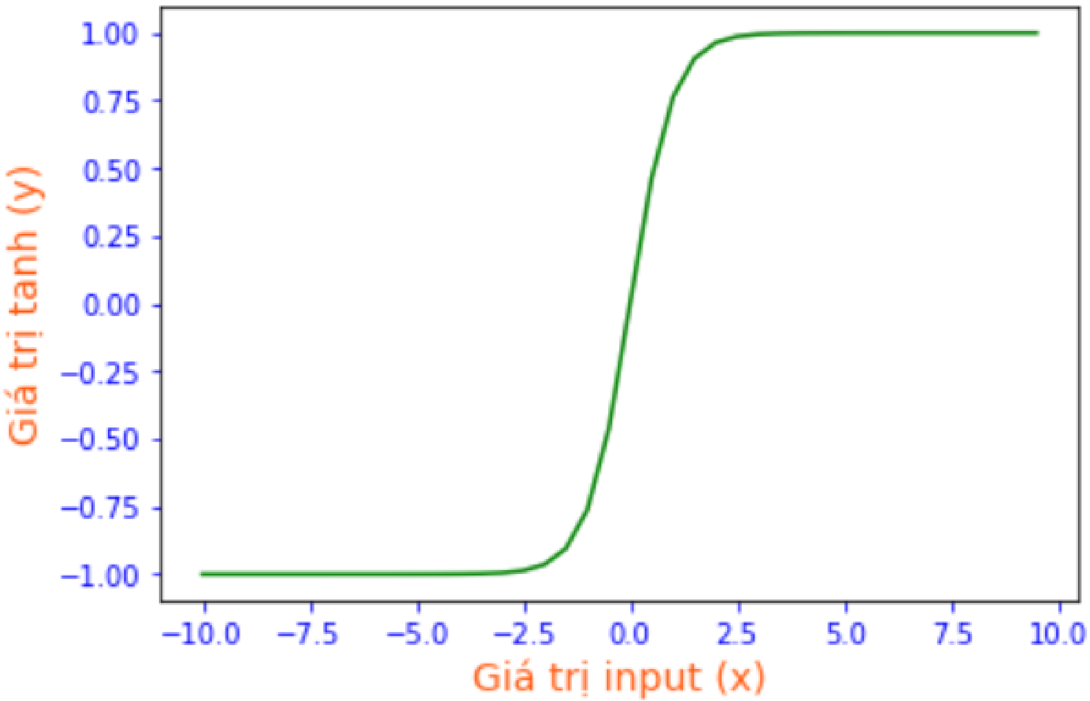
data =

1	5	-4	3	-2

data_a = tanh(data)

data_a =

0.761	0.999	-0.999	0.995	-0.964



To-do List for Training

❖ Sigmoid function

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

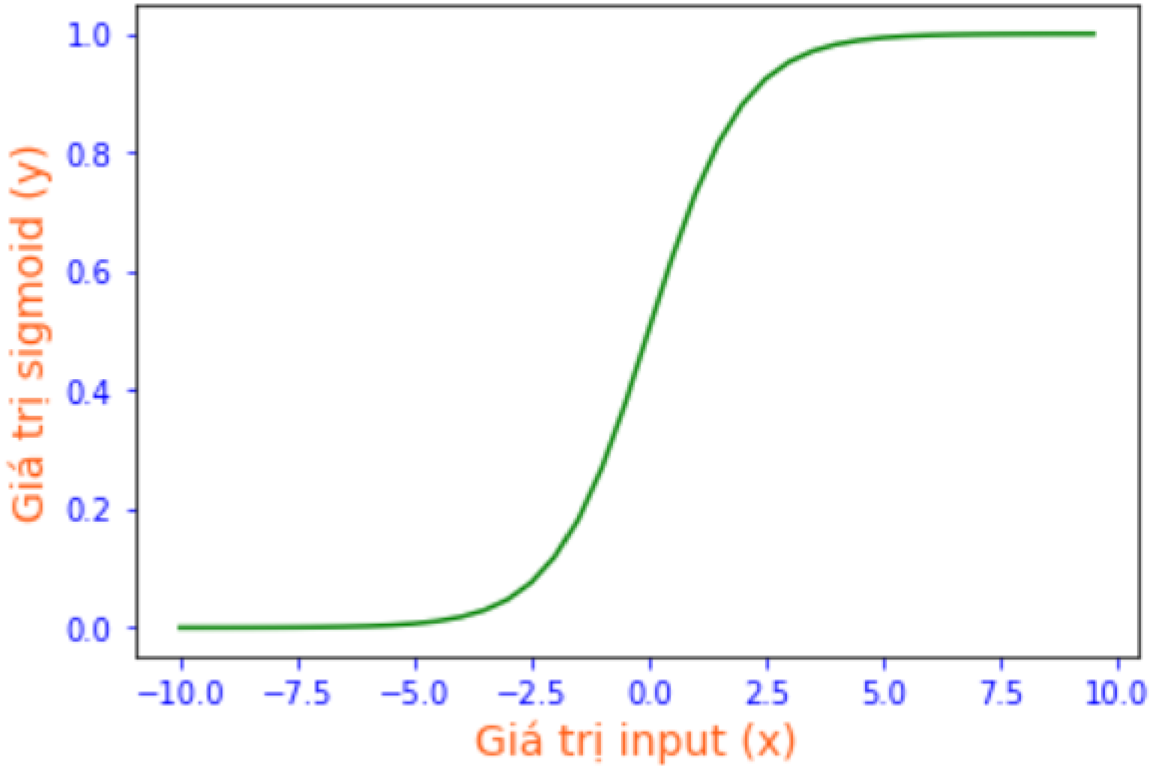
data =

1	5	-4	3	-2

data_a = sigmoid(data)

data_a =

0.731	0.993	0.017	0.95	0.119



To-do List for Training

❖ PReLU function

$$\text{PReLU}(x) = \begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

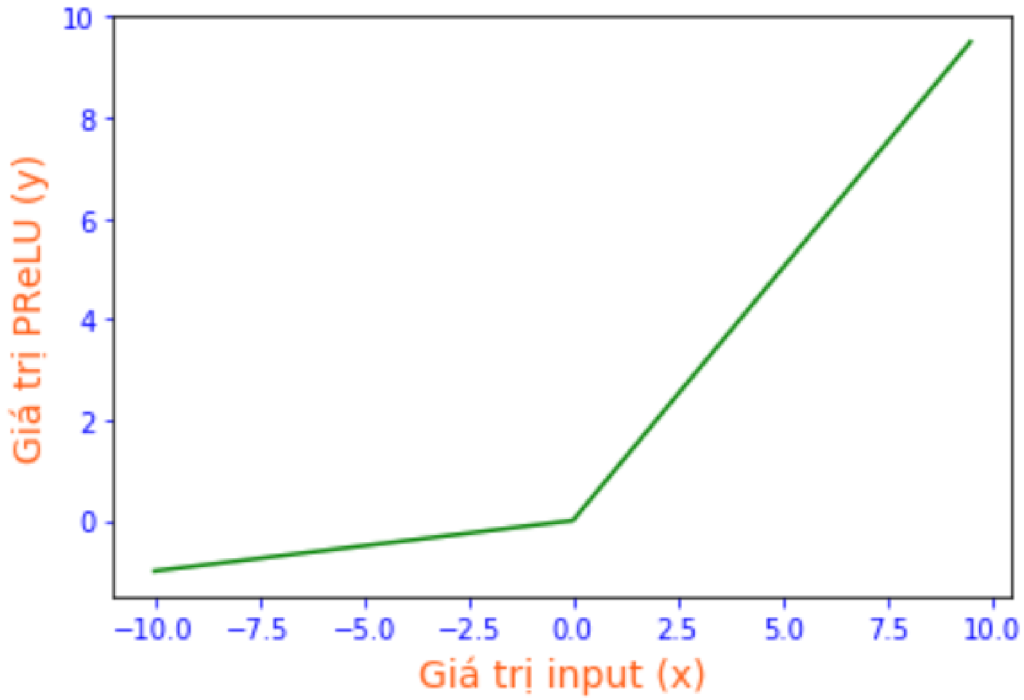
data =

1	5	-4	3	-2

data_a = PRELU(data)

data_a =

1	5	-0.4	3	-0.2



To-do List for Training

❖ ELU function

$$\text{ELU}(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

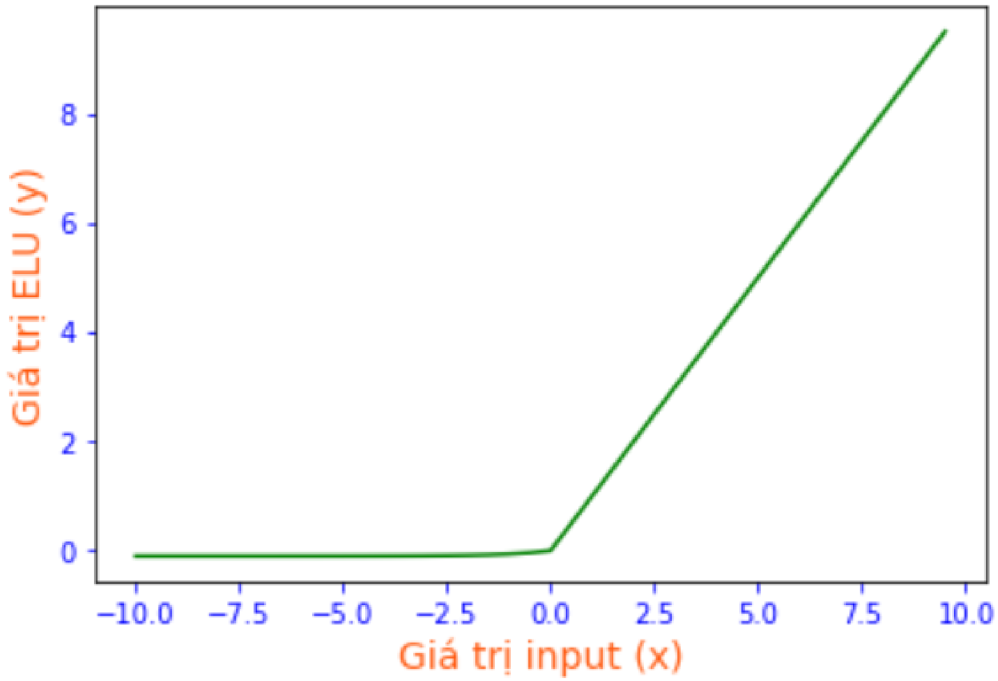
data =

1	5	-4	3	-2

data_a = ELU(data)

data_a =

1	5	-0.098	3	-0.086



To-do List for Training

❖ Softplus function

$$\text{softplus}(x) = \log(1 + e^x)$$

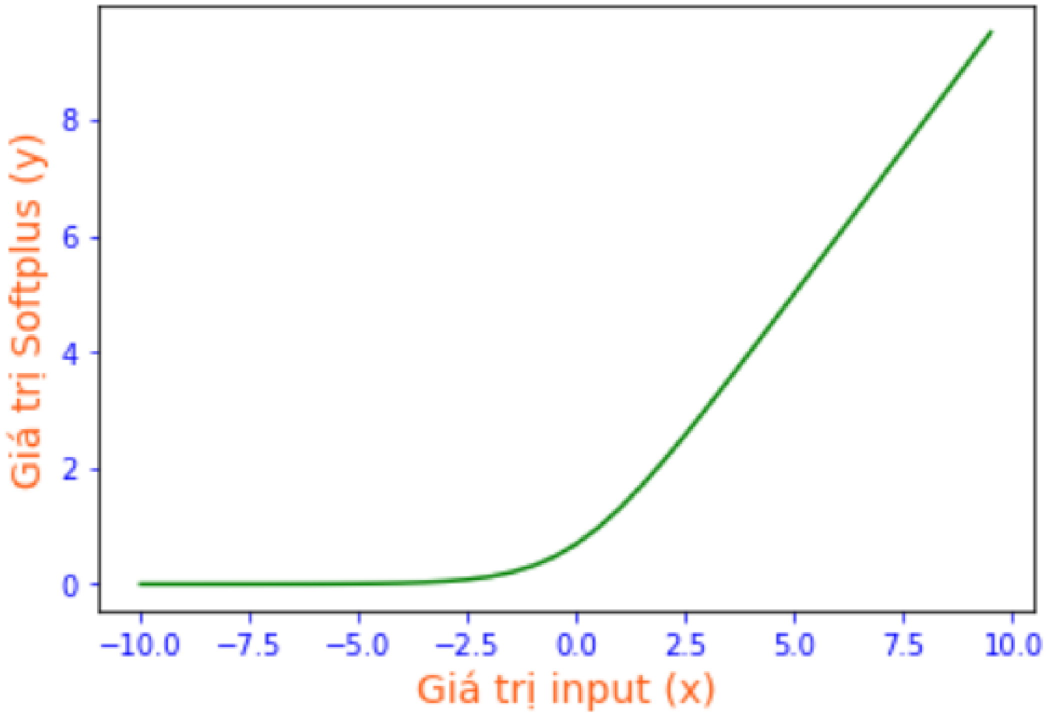
data =

1	5	-4	3	-2

data_a = softplus(data)

data_a =

1.313	5.006	0.018	3.048	0.126



To-do List for Training

❖ ReLU function

$$\mathrm{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

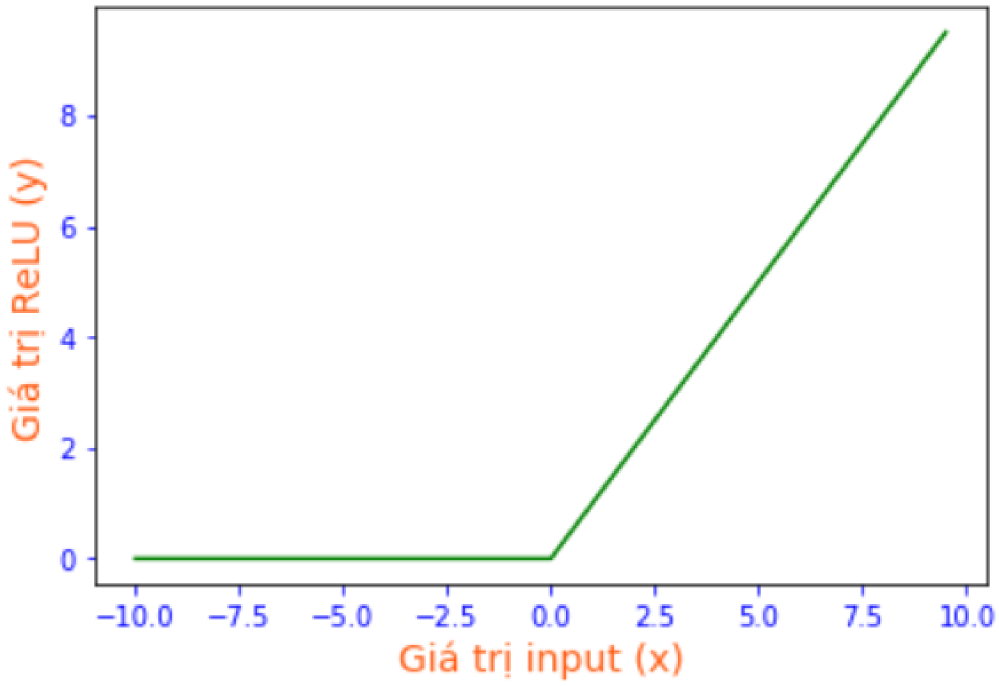
data =

1	5	-4	3	-2

data_a = ReLU(data)

data_a =

1	5	0	3	0



To-do List for Training

Model (Network) Construction

Which network components?

Dropout

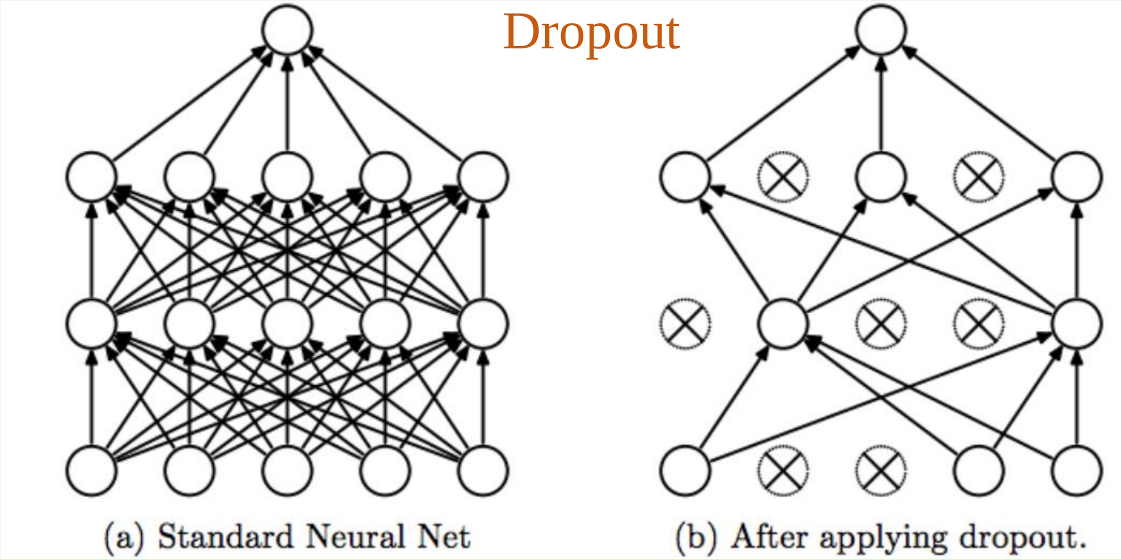


Figure is from (1)

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

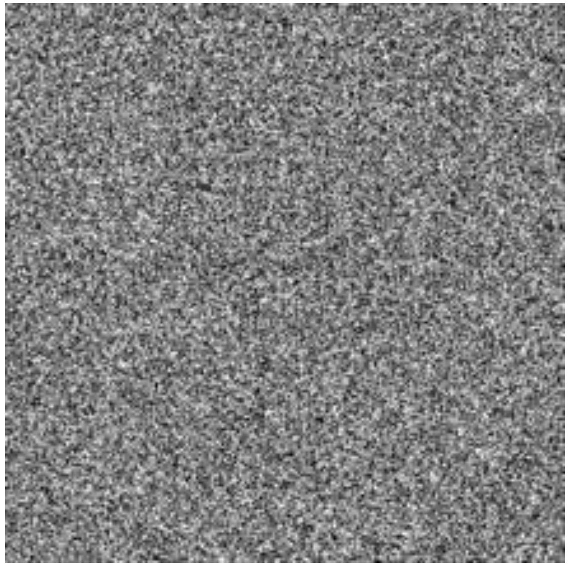
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

(1) https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

To-do List for Training

Parameter Initialization

Random Initialization



https://en.wikipedia.org/wiki/Randomness

- initializers
  - Overview
  - deserialize
  - get
  - GlorotNormal
  - GlorotUniform
  - he_normal
  - he_uniform
  - Identity
  - Initializer
  - lecun_normal
  - lecun_uniform
  - Orthogonal
  - serialize
  - TruncatedNormal
  - VarianceScaling

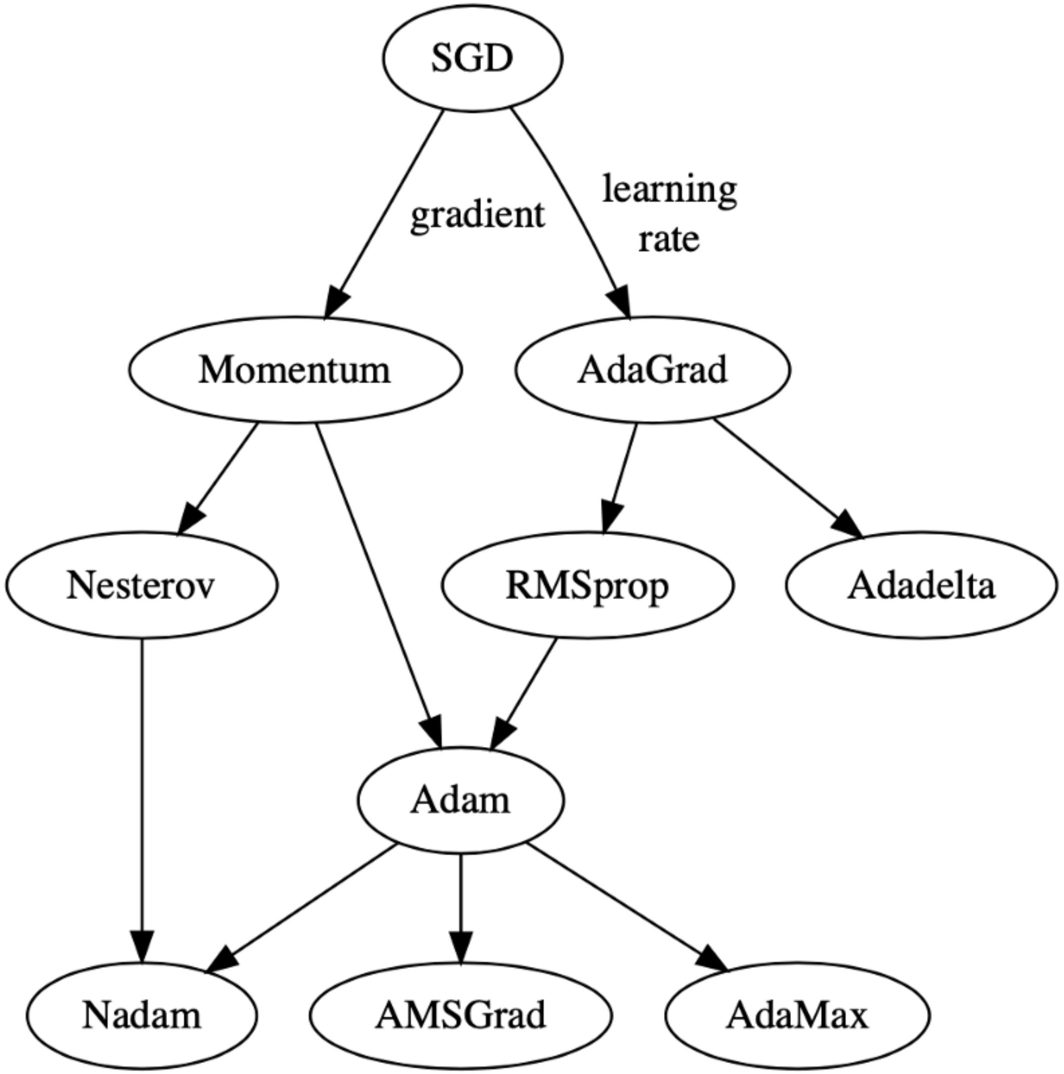
Initialization method supported in Tensorflow

https://www.tensorflow.org/api_docs/python/tf/keras/initializers

To-do List for Training

Optimizer Selection

Define a way to update parameters



https://www.kdnuggets.com/2019/06/gradient-descent-algorithms-cheat-sheet.html

To-do List for Training

Loss function Selection

Compute the goodness of the current model

Useful for training

https://www.tensorflow.org/api_docs/python/tf/keras/losses

`class BinaryCrossentropy` : Computes the cross-entropy loss between true labels and predicted labels.

`class CategoricalCrossentropy` : Computes the crossentropy loss between the labels and predictions.

`class CategoricalHinge` : Computes the categorical hinge loss between `y_true` and `y_pred`.

`class CosineSimilarity` : Computes the cosine similarity between `y_true` and `y_pred`.

`class Hinge` : Computes the hinge loss between `y_true` and `y_pred`.

`class Huber` : Computes the Huber loss between `y_true` and `y_pred`.

`class KLDivergence` : Computes Kullback-Leibler divergence loss between `y_true` and `y_pred`.

`class LogCosh` : Computes the logarithm of the hyperbolic cosine of the prediction error.

`class Loss` : Loss base class.

`class MeanAbsoluteError` : Computes the mean of absolute difference between labels and predictions.

`class MeanAbsolutePercentageError` : Computes the mean absolute percentage error between `y_true` and `y_pred`.

`class MeanSquaredError` : Computes the mean of squares of errors between labels and predictions.

`class MeanSquaredLogarithmicError` : Computes the mean squared logarithmic error between `y_true` and `y_pred`.

`class Poisson` : Computes the Poisson loss between `y_true` and `y_pred`.

`class Reduction` : Types of loss reduction.

`class SparseCategoricalCrossentropy` : Computes the crossentropy loss between the labels and predictions.

`class SquaredHinge` : Computes the squared hinge loss between `y_true` and `y_pred`.

To-do List for Training

Metric Selection

Compute the goodness of the current model

Useful for developers

Precision

True Positives

False Positives

True Negatives

Recall

False Negatives

Accuracy

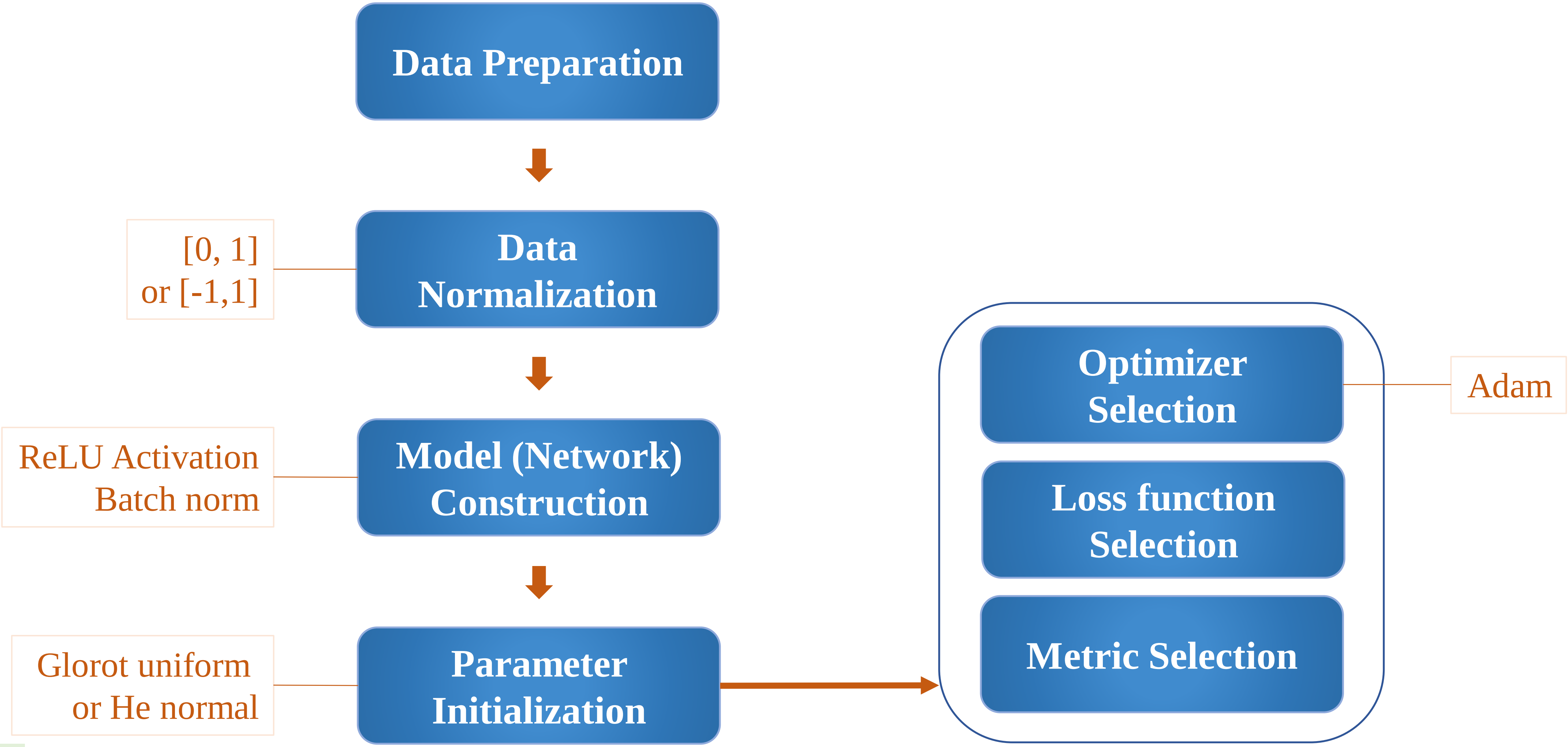
Root Mean Squared Error

Precision At Recall

Mean Absolute Error

To-do List for Training

Recommendation

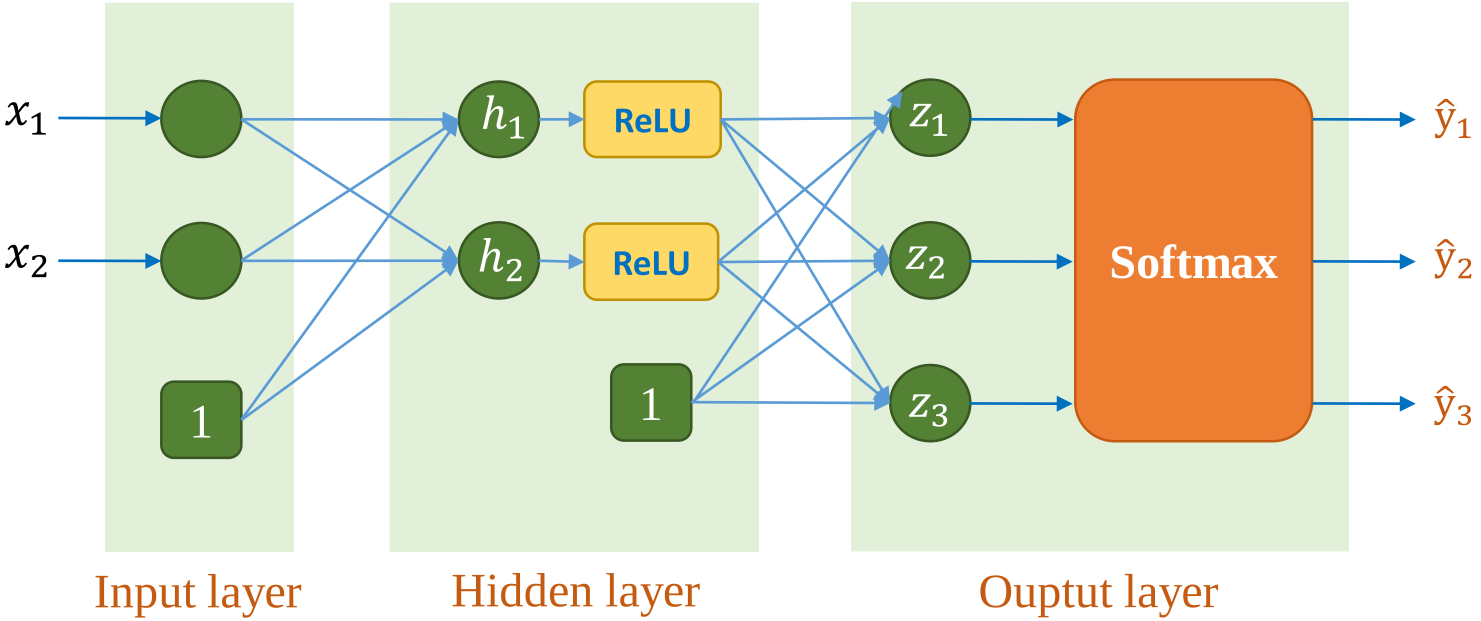


Outline

- Multi-layer Perceptron
- To-do List for Training
- Forward Computation Example
- Image Classification: Fashion-MNIST
- Image Classification: Cifar-10
- Underfitting and Overfitting

Example

Feature		Label
Petal_Length	Petal_Width	Label
1.5	0.2	0
1.4	0.2	0
1.6	0.2	0
4.7	1.6	1
3.3	1.1	1
4.6	1.3	1
5.6	2.2	2
5.1	1.5	2
5.6	1.4	2



$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{(1)} & \boldsymbol{x}^{(2)} & \boldsymbol{x}^{(3)} \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} 1.5 & 4.7 & 5.6 \\ 0.2 & 1.6 & 2.2 \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

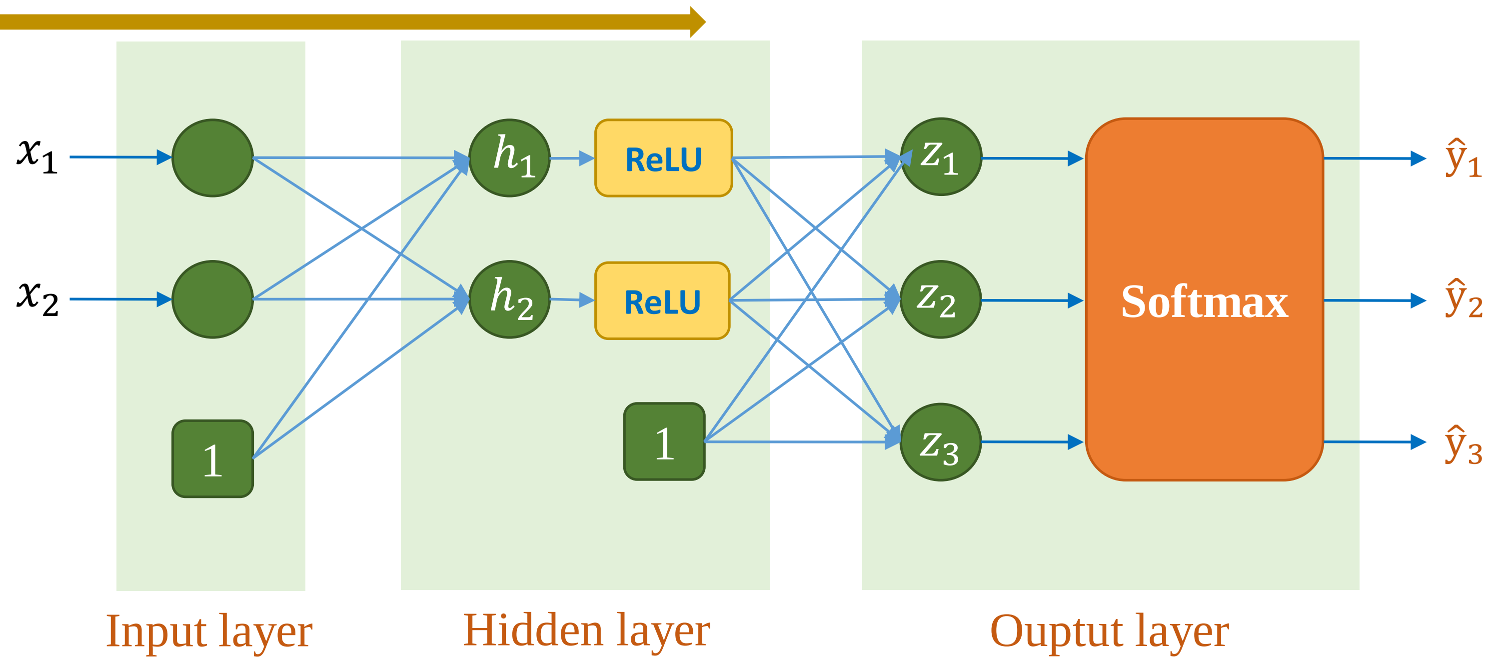
$$\boldsymbol{W}_h = [\boldsymbol{W}_{h1} \quad \boldsymbol{W}_{h2}] = \begin{bmatrix} 0.0 & 0.0 \\ 0.86 & -1.04 \\ 0.41 & -0.65 \end{bmatrix}$$

$$\boldsymbol{W}_z = [\boldsymbol{W}_{z1} \quad \boldsymbol{W}_{z2} \quad \boldsymbol{W}_{z3}] = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.32 & 0.25 & 0.14 \\ -0.47 & -1.06 & 0.063 \end{bmatrix}$$

$$\boldsymbol{h} = \boldsymbol{W}_h^T \boldsymbol{x} = \begin{bmatrix} 0.0 & 0.86 & 0.41 \\ 0.0 & -1.04 & -0.65 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1.5 & 4.7 & 5.6 \\ 0.2 & 1.6 & 2.2 \end{bmatrix} = \begin{bmatrix} 1.373 & 4.708 & 5.731 \\ -1.696 & -5.951 & -7.281 \end{bmatrix}$$

$$\mathrm{ReLU}(\boldsymbol{h}) = \begin{bmatrix} 1.373 & 4.708 & 5.731 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

Feature		Label
Petal Length	**Petal Width**	**Label**
1.5	0.2	0
1.4	0.2	0
1.6	0.2	0
4.7	1.6	1
3.3	1.1	1
4.6	1.3	1
5.6	2.2	2
5.1	1.5	2
5.6	1.4	2



$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{(1)} & \boldsymbol{x}^{(2)} & \boldsymbol{x}^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1.5 & 4.7 & 5.6 \\ 0.2 & 1.6 & 2.2 \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

$$\boldsymbol{W}_h = \begin{bmatrix} \boldsymbol{W}_{h1} & \boldsymbol{W}_{h2} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.0 \\ 0.86 & -1.04 \\ 0.41 & -0.65 \end{bmatrix} \qquad \boldsymbol{W}_z = \begin{bmatrix} \boldsymbol{W}_{z1} & \boldsymbol{W}_{z2} & \boldsymbol{W}_{z3} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.32 & 0.25 & 0.14 \\ -0.47 & -1.06 & 0.063 \end{bmatrix}$$

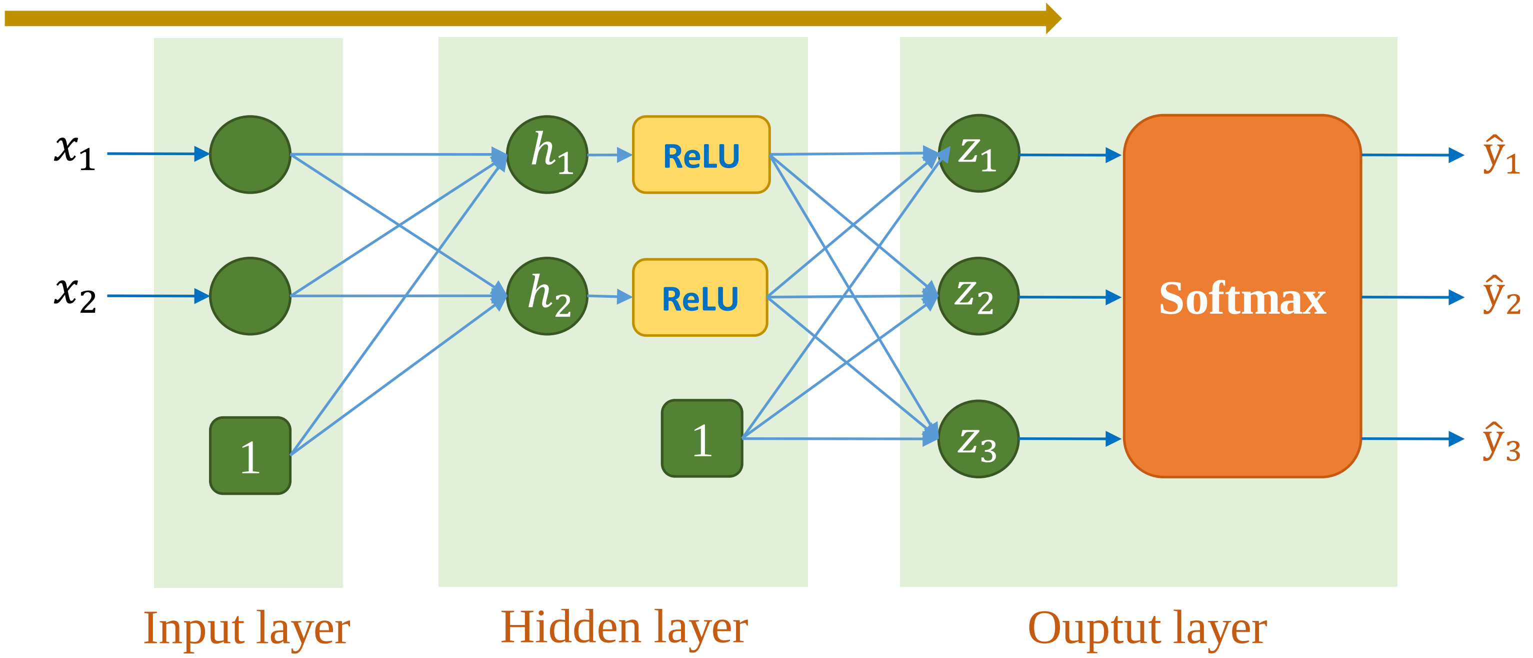
$$\text{ReLU}(\boldsymbol{h}) = \begin{bmatrix} 1.373 & 4.708 & 5.731 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{1} \\ \text{ReLU}(\boldsymbol{h}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1.373 & 4.708 & 5.731 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

$$\boldsymbol{z} = \boldsymbol{W}_z^T \begin{bmatrix} \mathbf{1} \\ \text{ReLU}(\boldsymbol{h}) \end{bmatrix} = \begin{bmatrix} 0.0 & 0.32 & -0.47 \\ 0.0 & 0.25 & -1.06 \\ 0.0 & 0.14 & 0.063 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1.373 & 4.708 & 5.731 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.439 & 1.507 & 1.835 \\ 0.356 & 1.220 & 1.485 \\ 0.195 & 0.670 & 0.816 \end{bmatrix}$$

Feature		Label
Petal Length	Petal Width	Label
1.5	0.2	0
1.4	0.2	0
1.6	0.2	0
4.7	1.6	1
3.3	1.1	1
4.6	1.3	1
5.6	2.2	2
5.1	1.5	2
5.6	1.4	2



$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{(1)} & \boldsymbol{x}^{(2)} & \boldsymbol{x}^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1.5 & 4.7 & 5.6 \\ 0.2 & 1.6 & 2.2 \end{bmatrix}$$

$$\boldsymbol{y} = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

$$\boldsymbol{W}_h = \begin{bmatrix} \boldsymbol{W}_{h1} & \boldsymbol{W}_{h2} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.0 \\ 0.86 & -1.04 \\ 0.41 & -0.65 \end{bmatrix}$$

$$\boldsymbol{W}_z = \begin{bmatrix} \boldsymbol{W}_{z1} & \boldsymbol{W}_{z2} & \boldsymbol{W}_{z3} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.32 & 0.25 & 0.14 \\ -0.47 & -1.06 & 0.063 \end{bmatrix}$$

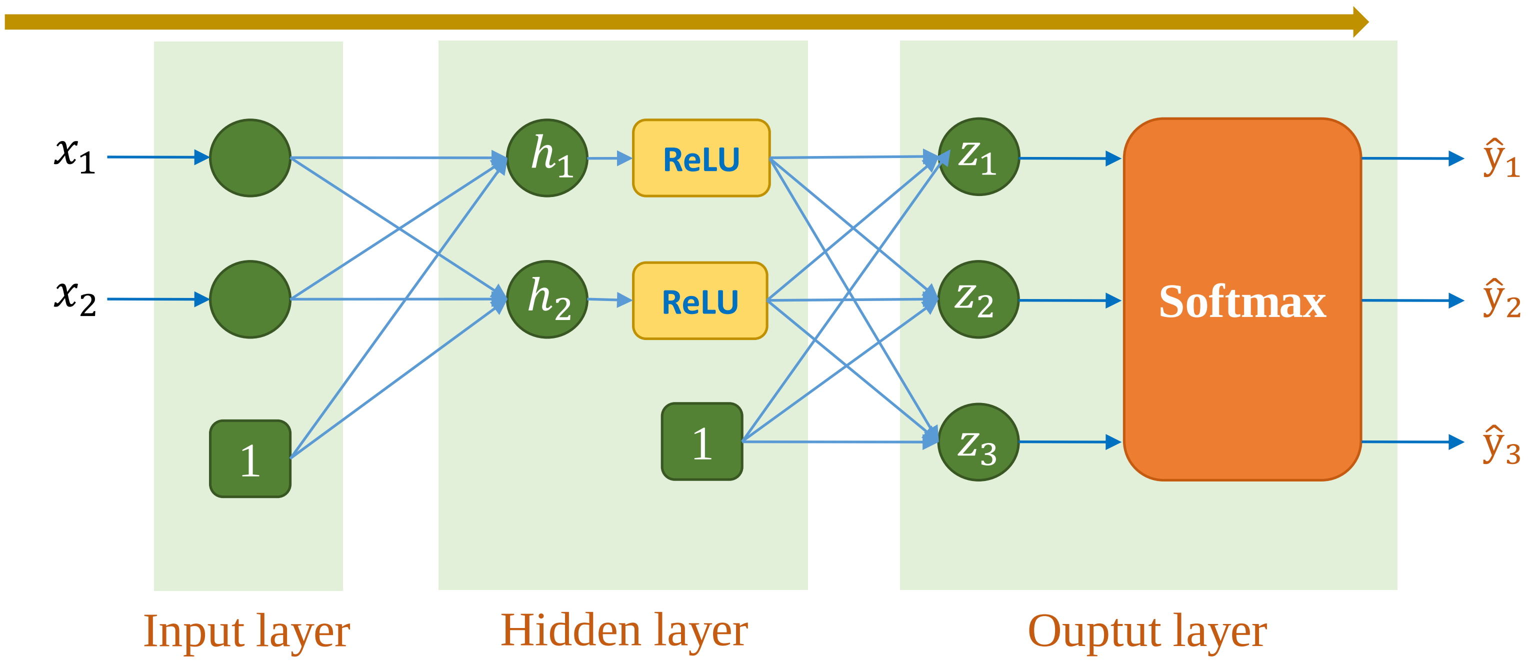
$$\boldsymbol{z} = \begin{bmatrix} 0.439 & 1.507 & 1.835 \\ 0.356 & 1.220 & 1.485 \\ 0.195 & 0.670 & 0.816 \end{bmatrix}$$

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{z}) = \begin{bmatrix} \hat{\boldsymbol{y}}^{(1)} & \hat{\boldsymbol{y}}^{(2)} & \hat{\boldsymbol{y}}^{(3)} \end{bmatrix}$$

$$= \begin{bmatrix} 0.369 & 0.458 & 0.484 \\ 0.340 & 0.343 & 0.341 \\ 0.289 & 0.198 & 0.174 \end{bmatrix}$$

$$\text{loss} = 1.269$$

Feature		Label
Petal_Length	Petal_Width	Label
1.5	0.2	0
1.4	0.2	0
1.6	0.2	0
4.7	1.6	1
3.3	1.1	1
4.6	1.3	1
5.6	2.2	2
5.1	1.5	2
5.6	1.4	2



$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{(1)} & \boldsymbol{x}^{(2)} & \boldsymbol{x}^{(3)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1.5 & 4.7 & 5.6 \\ 0.2 & 1.6 & 2.2 \end{bmatrix}$$

$$\boldsymbol{y} = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

$$\boldsymbol{W_h} = \begin{bmatrix} \boldsymbol{W_{h1}} & \boldsymbol{W_{h2}} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.0 \\ 0.86 & -1.04 \\ 0.41 & -0.65 \end{bmatrix}$$

$$\boldsymbol{W_z} = \begin{bmatrix} \boldsymbol{W_{z1}} & \boldsymbol{W_{z2}} & \boldsymbol{W_{z3}} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.32 & 0.25 & 0.14 \\ -0.47 & -1.06 & 0.063 \end{bmatrix}$$

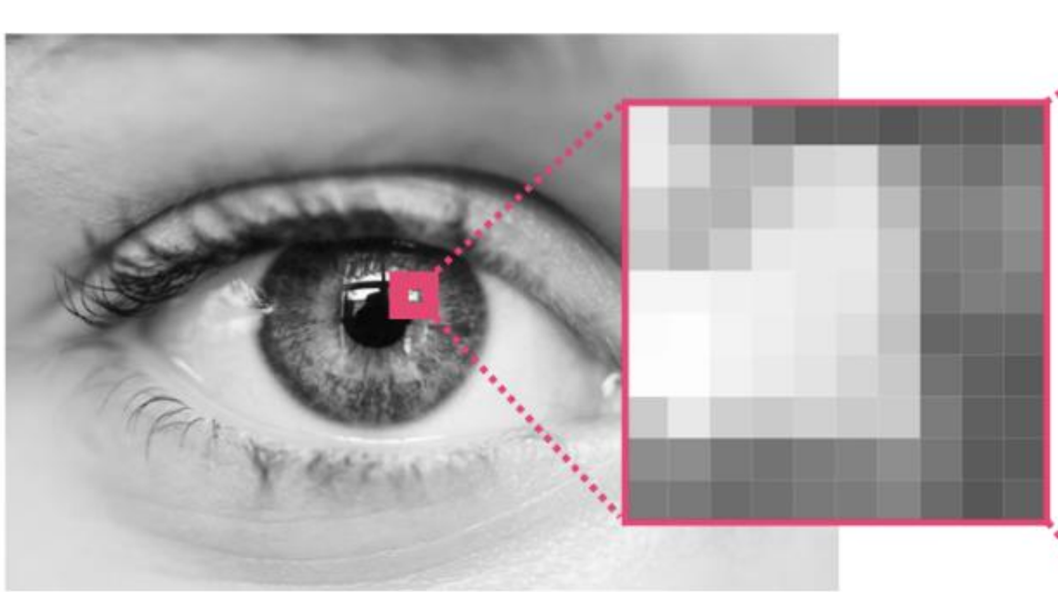
Outline

- Multi-layer Perceptron
- To-do List for Training
- Forward Computation Example
- Image Classification: Fashion-MNIST
- Image Classification: Cifar-10
- Underfitting and Overfitting

AI VIETNAM
Free AI Course

Image Classification: Image Data

❖ Grayscale images



230	194	147	108	90	98	84	96	91	101
237	206	188	195	207	213	163	123	116	128
210	183	180	205	224	234	188	122	134	147
198	189	201	227	229	232	200	125	127	135
249	241	237	244	232	226	202	116	125	126
251	254	241	239	230	217	196	102	103	99
243	255	240	231	227	214	203	116	95	91
204	231	208	200	207	201	200	121	95	95
144	140	120	115	125	127	143	118	92	91
121	121	108	109	122	121	134	106	86	97

(Height, Width)

Pixel p = scalar

$0 \leq p \leq 255$

Resolution: #pixels
Resolution = Height x Width

Image Classification: Image Data

❖ Color images

(Height, Width, channel)

RGB color image

Pixel $p = \begin{bmatrix} r \\ g \\ b \end{bmatrix}$

$0 \le r,g,b \le 255$

Resolution: #pixels

Resolution = HeightxWidth

Image Data

Fashion-MNIST dataset

Grayscale images

Resolution=28x28

Training set: 60000 samples

Testing set: 10000 samples

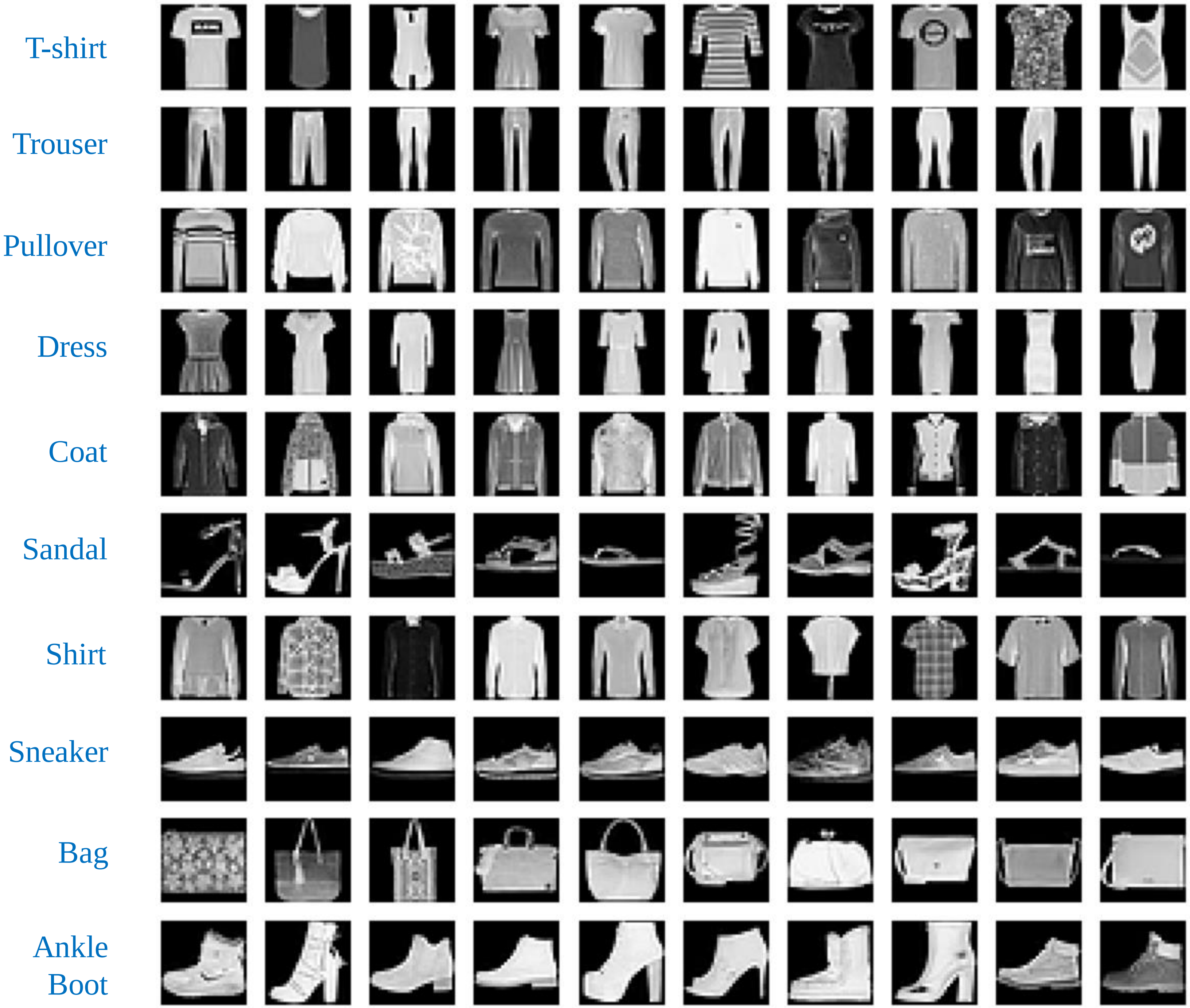


Image Classification

❖ Fashion-MNIST data

Download data

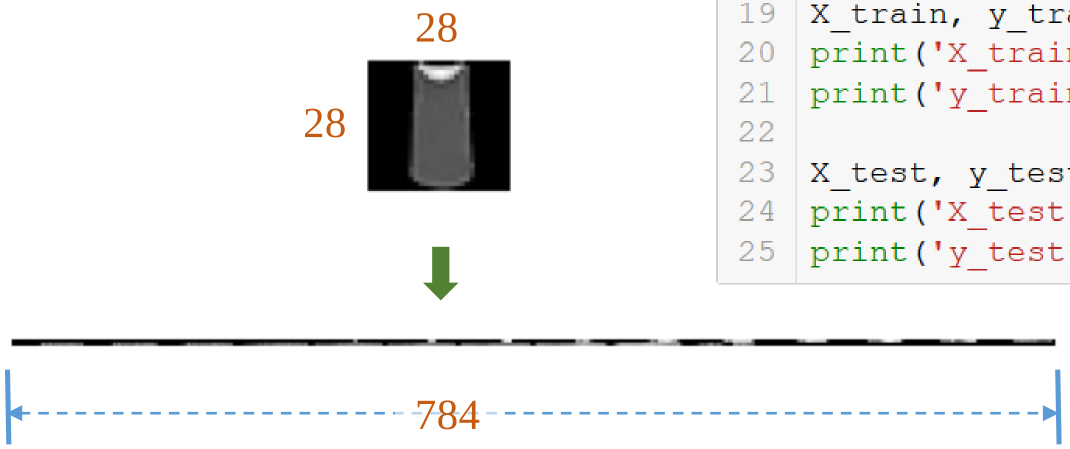
Name	Size
t10k-images-idx3-ubyte.gz	4.4 MB
t10k-labels-idx1-ubyte.gz	5.1 kB
train-images-idx3-ubyte.gz	26.4 MB
train-labels-idx1-ubyte.gz	29.5 kB

```python
import numpy as np
from urllib import request
import gzip
import pickle

filename = [["training_images","train-images-idx3-ubyte.gz"],
            ["test_images","train-labels-idx1-ubyte.gz"],
            ["training_labels","t10k-images-idx3-ubyte.gz"],
            ["test_labels","t10k-labels-idx1-ubyte.gz"]]

# function to download data
def download_fashion_mnist(folder):
    base_url = "http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/"
    for name in filename:
        print("Downloading " + name[1] + "...")

        # lưu vào folder data_fashion_mnist
        request.urlretrieve(base_url + name[1], folder + name[1])
    print("Download complete.")

# download dataset và save to folder 'data_fashion_mnist/'
folder = 'data_fashion_mnist/'
download_fashion_mnist(folder)
```

Image Classification

Fashion-MNIST data

```
X_train: (60000, 784)
y_train: (60000,)
X_test: (10000, 784)
y_test: (10000,)
```



Read data

```python
import os
import gzip
import numpy as np

def load_fashion_mnist(path, kind='train'):
    """Load fashion_MNIST data from `path`"""
    labels_path = os.path.join(path, '%s-labels-idx1-ubyte.gz' % kind)
    images_path = os.path.join(path, '%s-images-idx3-ubyte.gz' % kind)

    with gzip.open(labels_path, 'rb') as lbpath:
        labels = np.frombuffer(lbpath.read(), dtype=np.uint8, offset=8)
    with gzip.open(images_path, 'rb') as imgpath:
        images = np.frombuffer(imgpath.read(),
                               dtype=np.uint8, offset=16).reshape(len(labels), 784)

    return images, labels


X_train, y_train = load_fashion_mnist('C:/Data/data_fashion_mnist/')
print('X_train:', X_train.shape)
print('y_train:', y_train.shape)

X_test, y_test = load_fashion_mnist('C:/Data/data_fashion_mnist/', kind='t10k')
print('X_test:', X_test.shape)
print('y_test:', y_test.shape)
```

Image Classification

Fashion-MNIST data

```
X_train: (60000, 784)
y_train: (60000,)
X_test: (10000, 784)
y_test: (10000,)
```

```python
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(784,)))
model.add(keras.layers.Dense(128, activation='sigmoid'))
model.add(keras.layers.Dense(10, activation='softmax'))

# optimizer and loss
model.compile(optimizer='sgd',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# training
model.fit(X_train, y_train, epochs=10)

# testing
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print('Test accuracy:', test_acc)
```

Image Classification: Another Model

Fashion-MNIST data

```python
import tensorflow as tf
from tensorflow import keras

# Data Preparation - Use built-in function for Fashion_MNIST in Tensorflow
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

# Data Normalization [0,1]
train_images = train_images / 255.0
test_images = test_images / 255.0

# model: Use relu activation
# Glorot uniform is used by default in Tensorflow
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

# Use Adam optimizer, cross-entropy loss and accuracy metric
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])

# training
model.fit(train_images, train_labels, epochs=20)

# testing
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
print('Test accuracy:', test_acc)
```

Outline

- Multi-layer Perceptron
- To-do List for Training
- Forward Computation Example
- Image Classification: Fashion-MNIST
- Image Classification: Cifar-10
- Underfitting and Overfitting

Image Classification

Cifar-10 dataset

Color images

Resolution=32x32

Training set: 50000 samples

Testing set: 10000 samples

Image Classification

Cifar-10 dataset

Color images

Resolution=32x32

Training set: 50000 samples

Testing set: 10000 samples

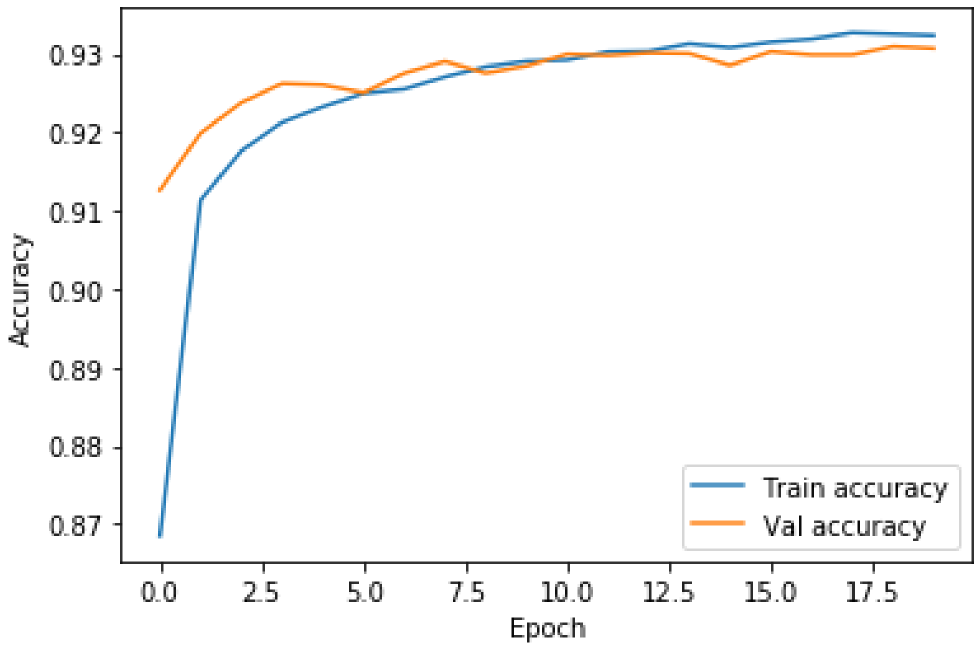
```python
import tensorflow as tf
from tensorflow import keras

# Data Preparation - Use built-in function for Fashion_MNIST in Tensorflow
cifar10 = keras.datasets.cifar10
(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()

# Data Normalization [0,1]
train_images = train_images / 255.0
test_images = test_images / 255.0

# model: Use relu activation
# Glorot uniform is used by default in Tensorflow
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(32, 32, 3)),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

# Use Adam optimizer, cross-entropy loss and accuracy metric
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])

# training
model.fit(train_images, train_labels, epochs=20)

# testing
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
print('Test accuracy:', test_acc)
```

Outline

- Multi-layer Perceptron
- To-do List for Training
- Forward Computation Example
- Image Classification: Fashion-MNIST
- Image Classification: Cifar-10
- Underfitting and Overfitting

Underfitting

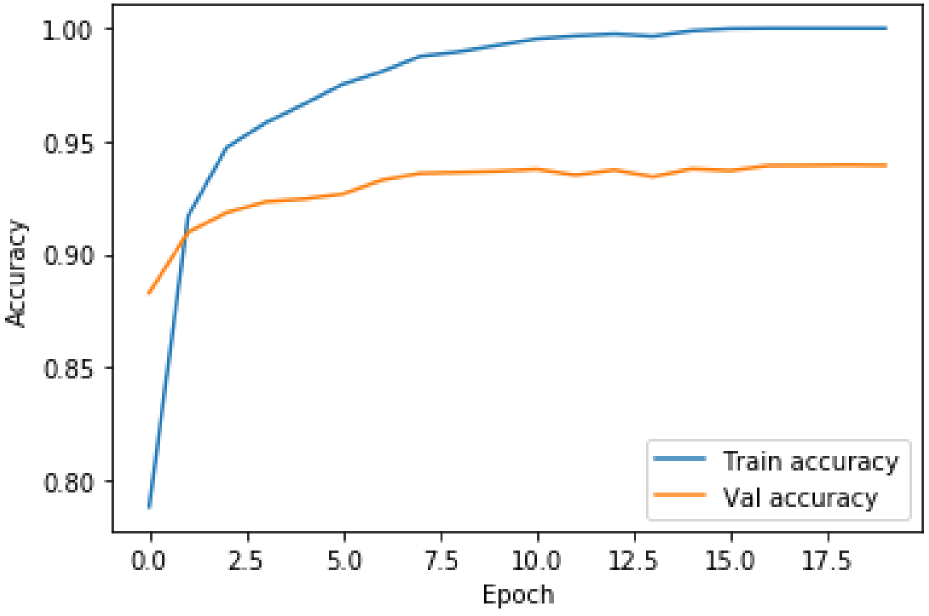
Happen when model is not strong enough




```python
import tensorflow as tf
from tensorflow import keras

# load data
mnist = keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize
x_train, x_test = x_train / 255.0, x_test / 255.0
m_train = x_train.shape[0]

# model construction
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(10, activation='softmax')
])

# compile and train
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
history = model.fit(x_train, y_train,
                    validation_split=0.2, epochs=20, verbose=0)
```

Overfitting

Model performance is 'quite' different between training and test sets




```python
import tensorflow as tf
from tensorflow import keras

# load data
mnist = keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize
x_train, x_test = x_train / 255.0, x_test / 255.0
m_train = x_train.shape[0]

# model construction
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

# model compile and train
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
history = model.fit(x_train, y_train,
                    validation_split=0.9, epochs=20, verbose=0)
```

Multi-layer Perceptron

❖ Demo

```
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] ::
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>> for epoch in range(n_epochs):
...     sum_of_losses = 0
...     gradients = np.zeros((2,1))
...
...     for index in range(4):
...         xi = X_b[index:index+1]
...         yi = y[index:index+1]
```

Thank you!