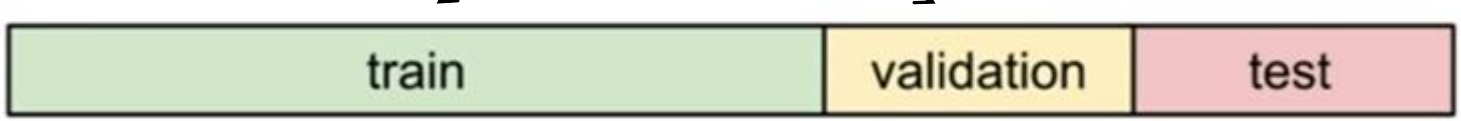
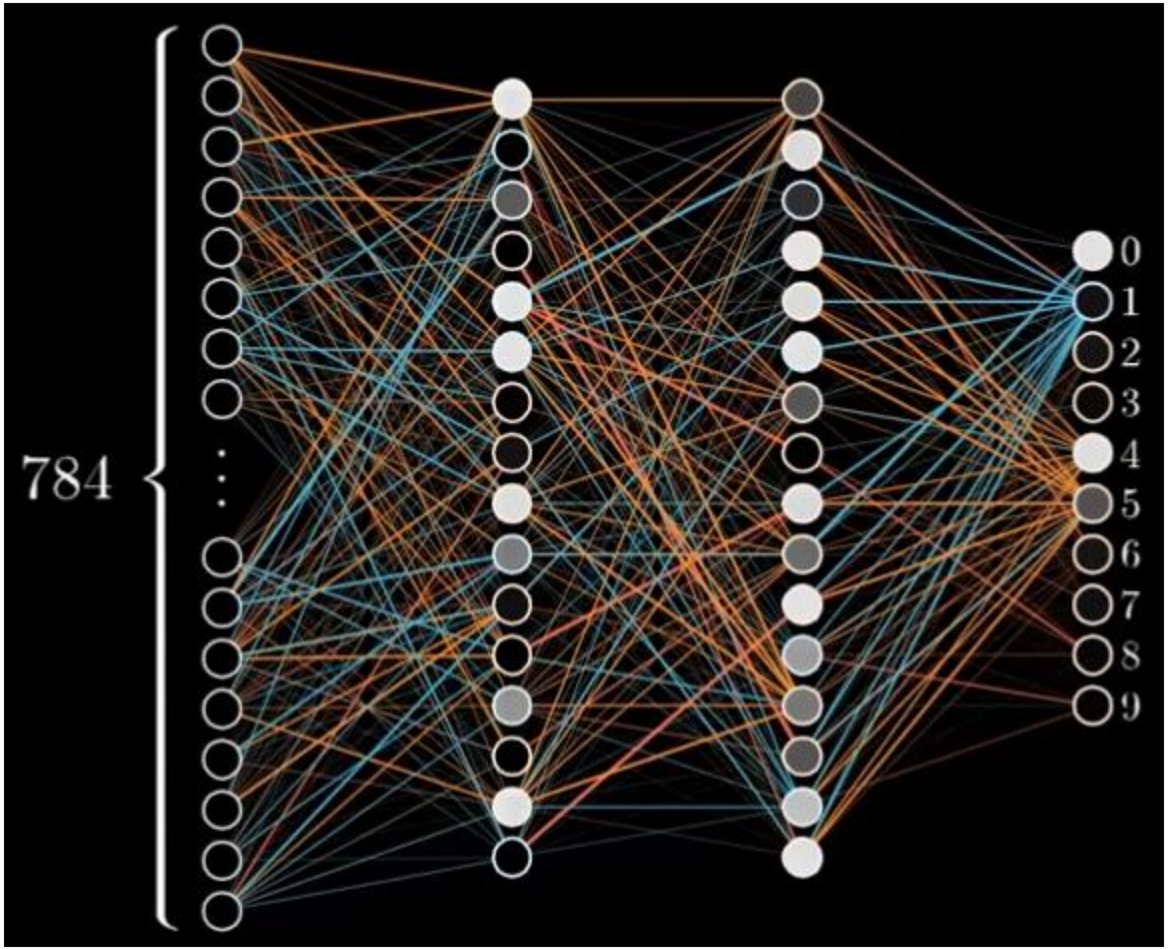
Навчання нейронних мереж

$$(\mathbf{X}_i, \mathbf{Y}_i), i = 1, \dots, N,$$



X_i

Y_i



https://www.youtube.com/watch?v=IIg3gGewQ5U

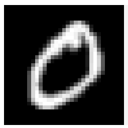
Функція помилки (Loss Function)

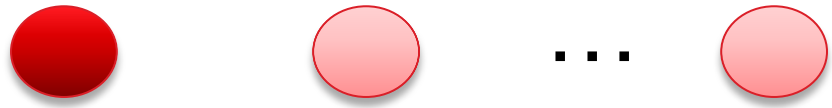
$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

Loss function for classification

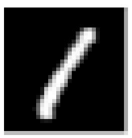
X

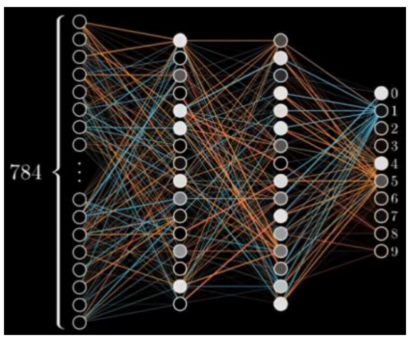
$p(c=0|x_1)$ $p(c=1|x_1)$ $p(c=9|x_1)$

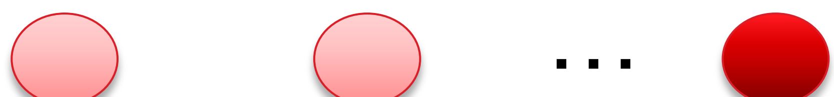




$p(c=0|x_2)$ $p(c=1|x_2)$ $p(c=9|x_2)$







Maximum likelihood

$$p(data) = \prod_i p(c = y_i \mid x_i)$$

w, b

Negative log-likelihood

w, b

$$-\ln p(data) = -\sum_i \ln p(c = y_i \mid x_i)$$

Cross-Entropy Loss Function

$$L = -\frac{1}{N}\sum_i \ln p(c = y_i \mid x_i)$$

Loss function for classification

Cross-Entropy Loss Function

$$L = -\frac{1}{N} \sum_i \ln p(c = y_i \mid x_i)$$

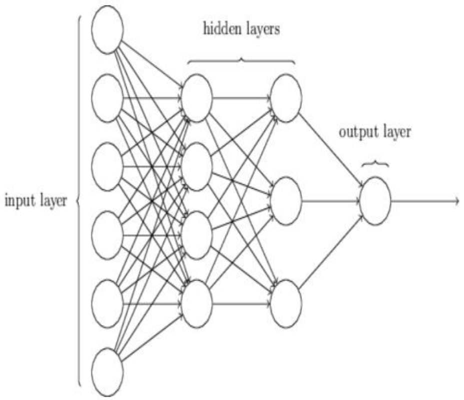
Activation function $\rightarrow$ Softmax

Loss function for binary classification

Binary cross-entropy loss

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \ln p(y_i \mid x_i) + (1 - y_i)\ln(1 - p(y_i \mid x_i))$$

Activation function $\longrightarrow$ Sigmoid



Loss function for regression

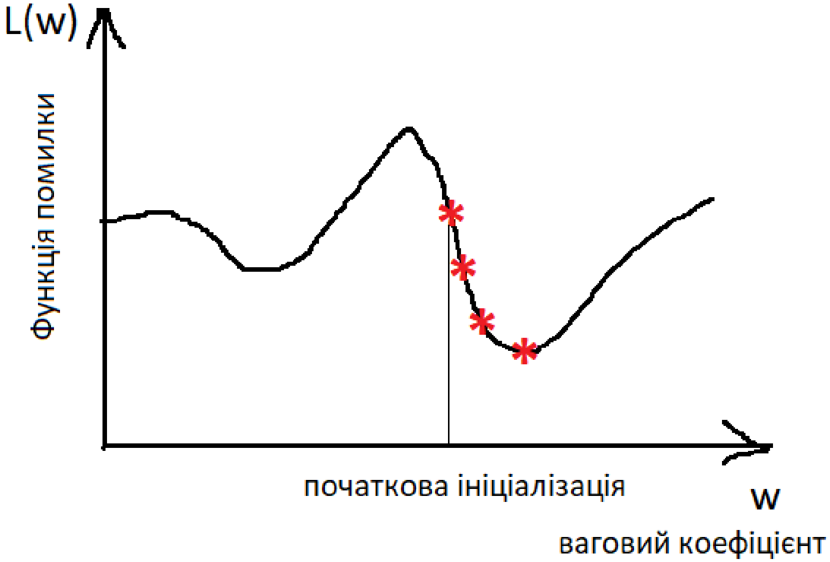
Mean Absolute Error (MAE), L1 Loss

$$L = \frac{1}{N}\sum_{i=1}^{N}\left| f_i - y_i \right|$$

Mean Square Error (MSE) , L2 Loss

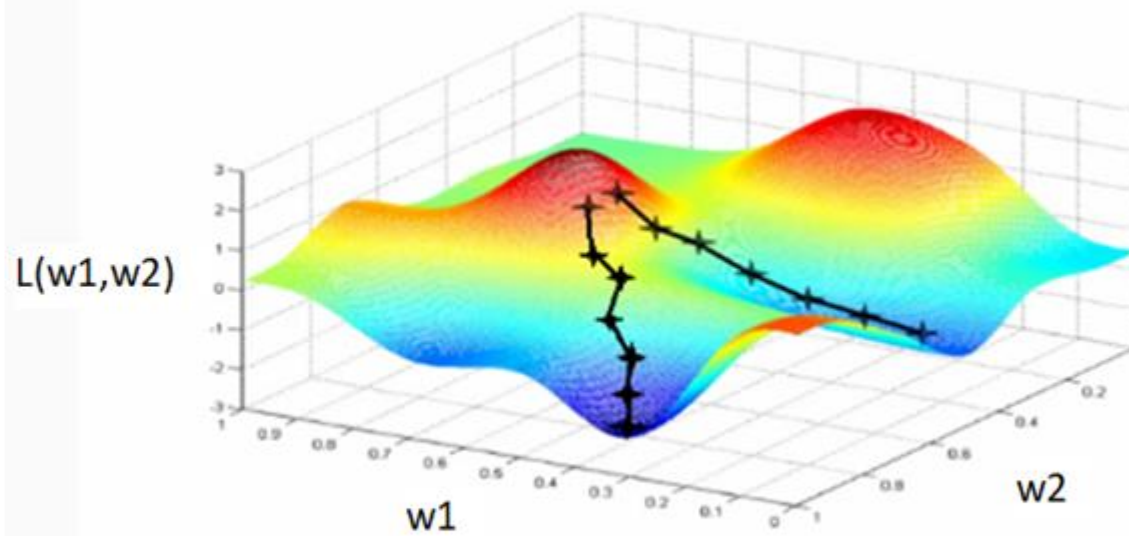
$$L = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2$$

Градієнтний спуск (Gradient descent)

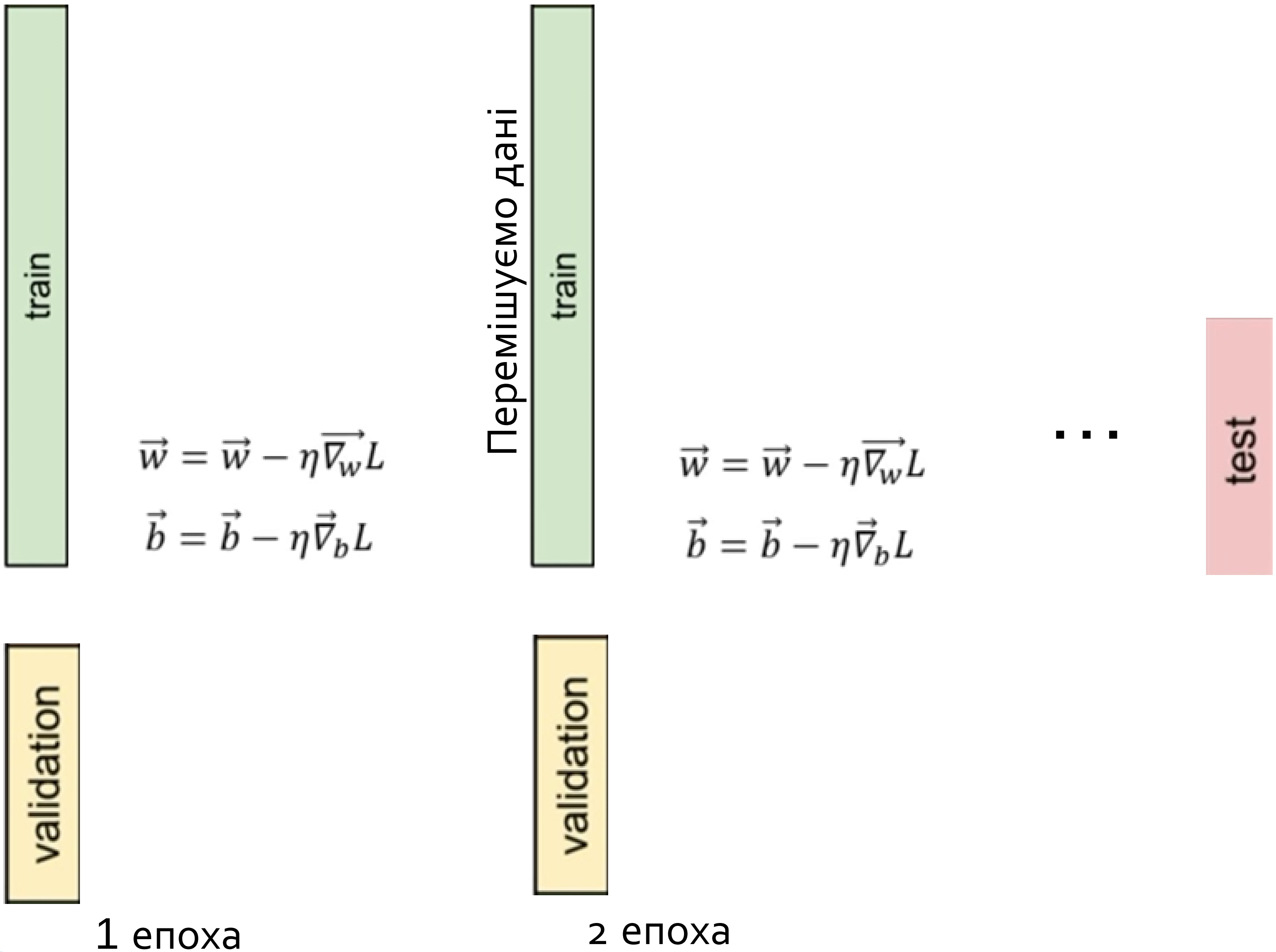


$$\vec{w} = \vec{w} - \eta \overrightarrow{\nabla_w} L$$

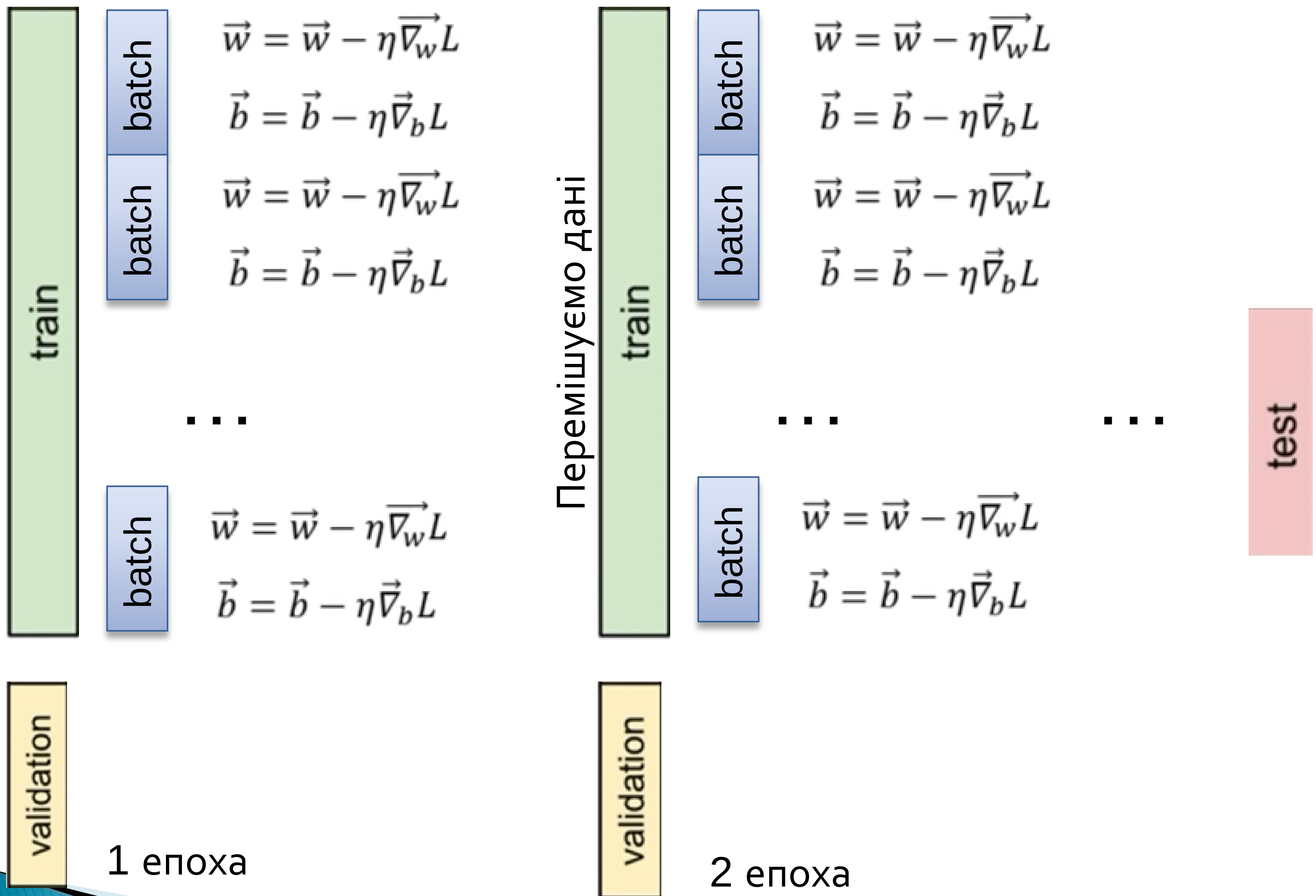
$$\vec{b} = \vec{b} - \eta \vec{\nabla}_b L$$

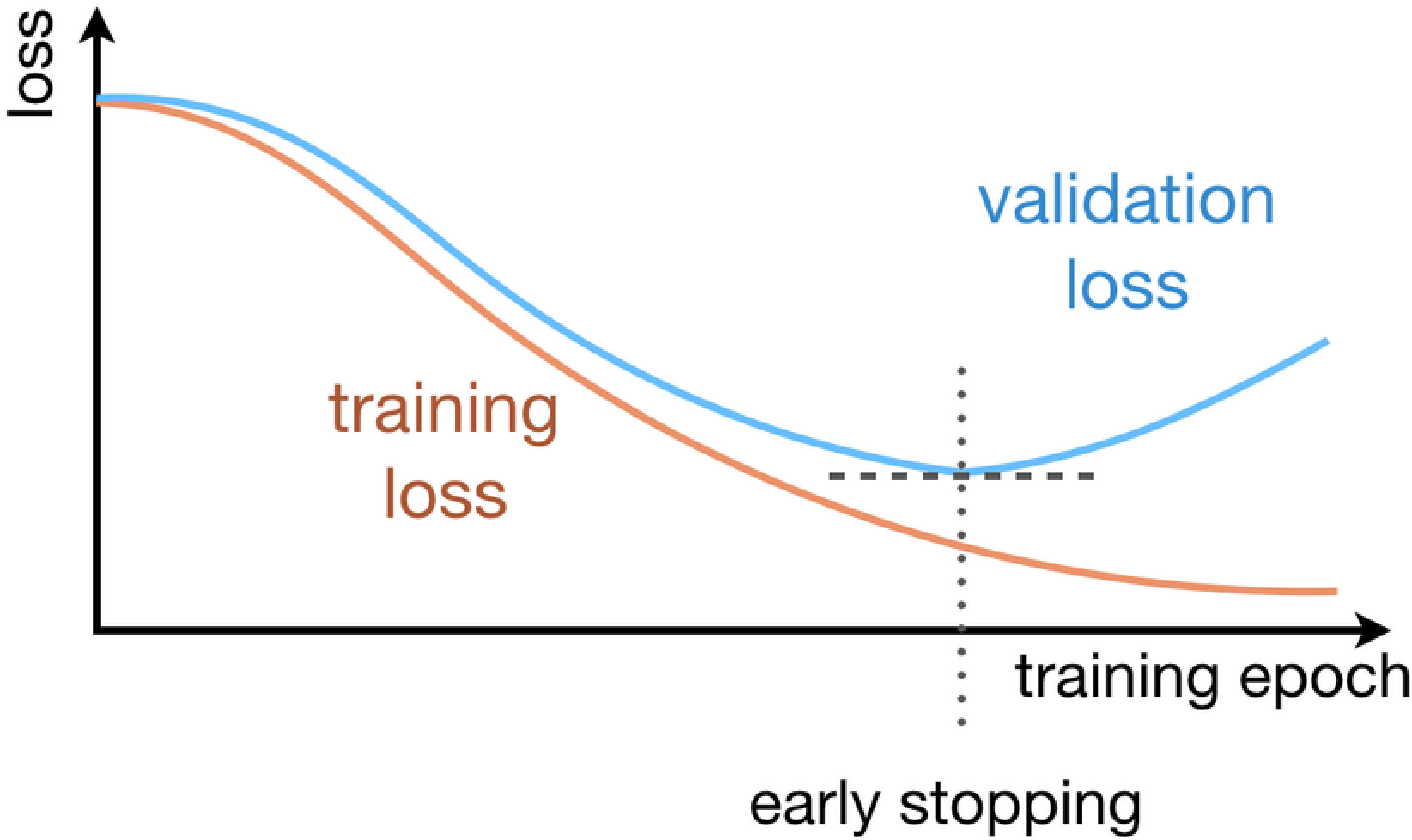


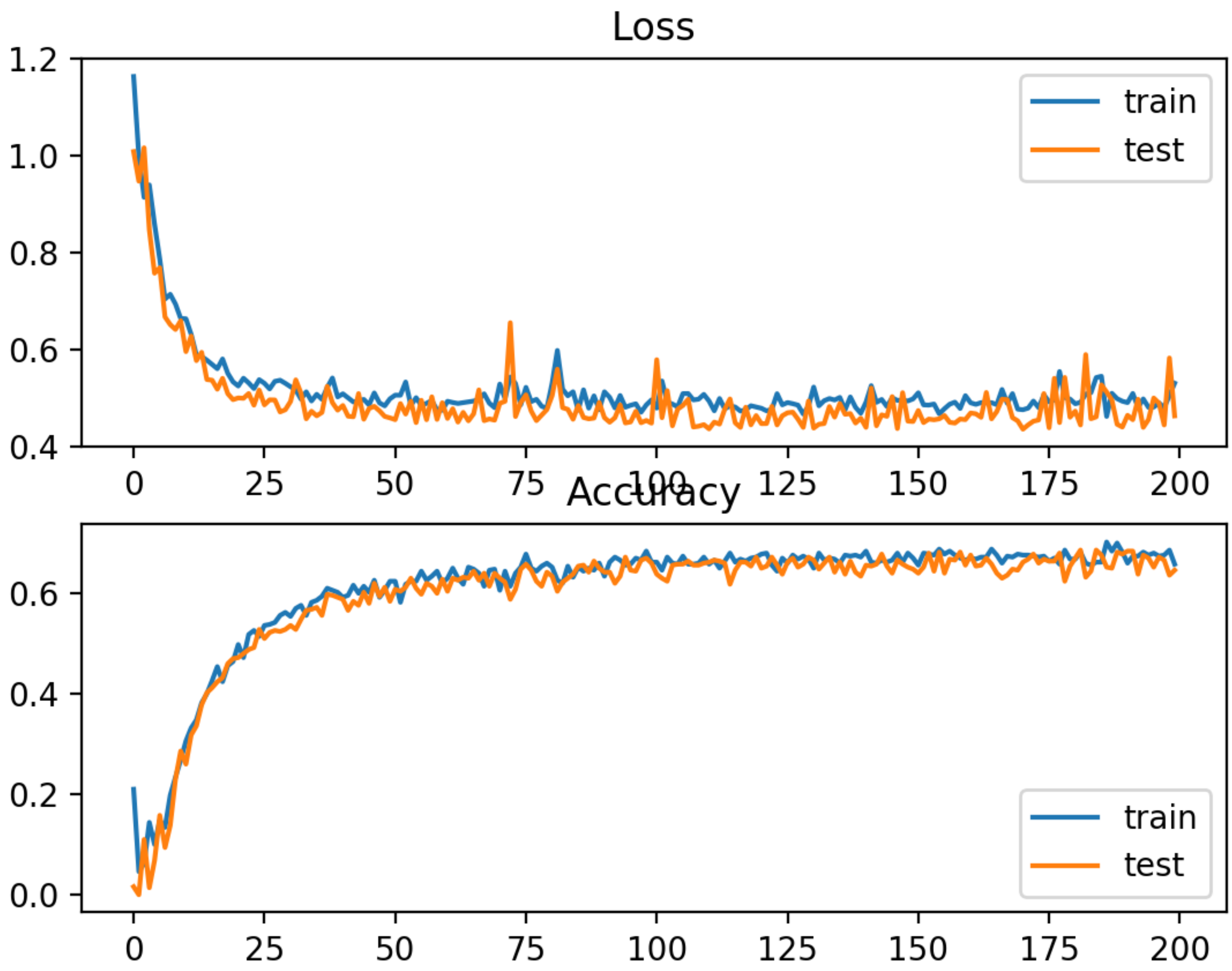
Gradient descent

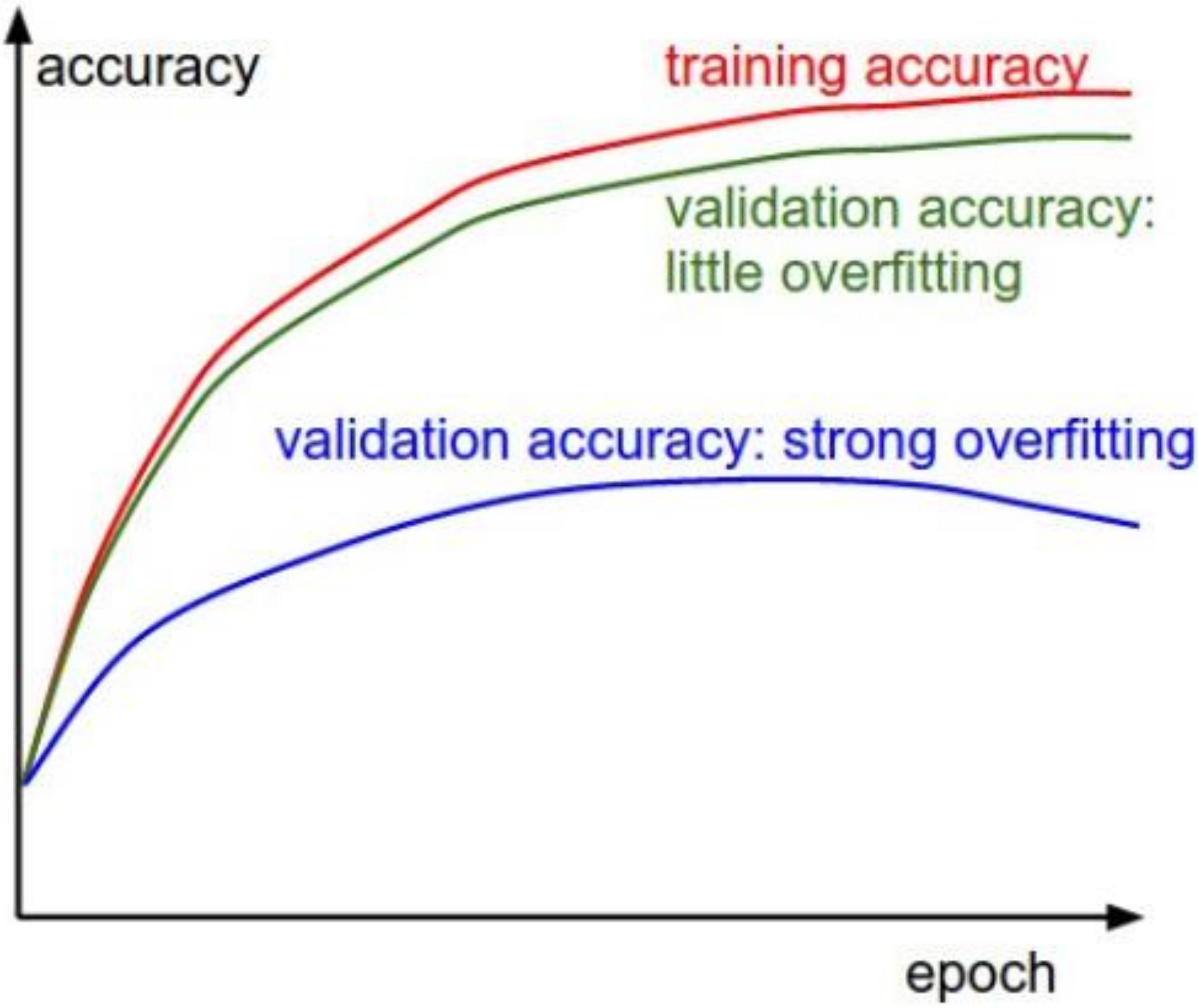


Stochastic Gradient Descent (SGD)

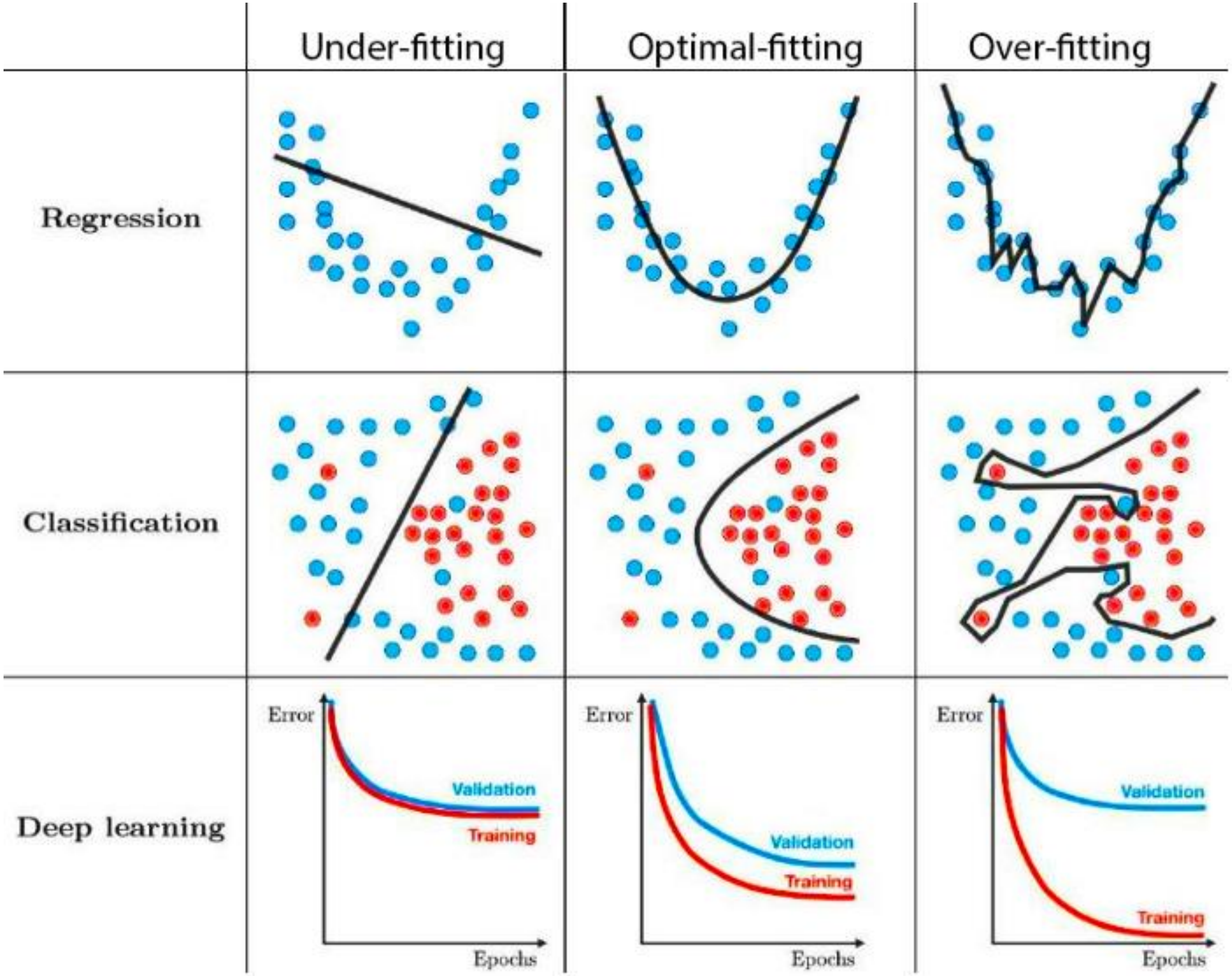



loss
validation loss
training loss
training epoch
early stopping


Loss
train
test
1.2
1.0
0.8
0.6
0.4
0
25
50
75
100
125
150
175
200
Accuracy
train
test
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200



http://cs231n.stanford.edu/



https://towardsdatascience.com/te[...]es-for-handling-underfitting-and-overfitting-in-machine-learning-348daa2380b9

Помилка на train → велика →

<u>Underfitting</u>

Більше ресурсів для тренування

Більш складна модель

Інший підхід

↓ мала

Помилка на validation → велика →

<u>Overfitting</u>

Більше даних

Регуляризація, dropout

Інший підхід

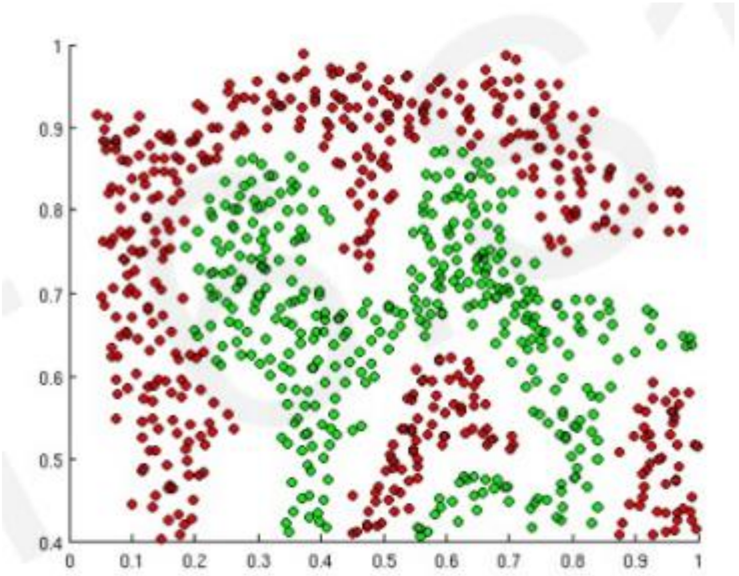
↓ мала

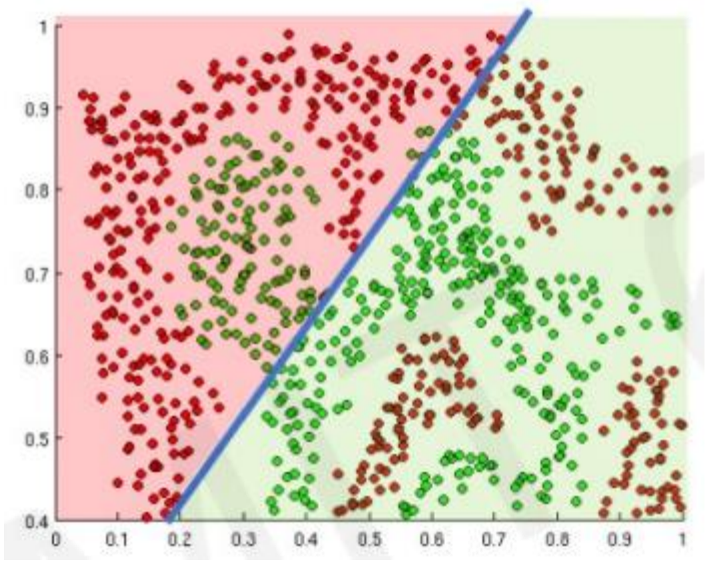
Помилка на test → велика → Відрізняються train і test

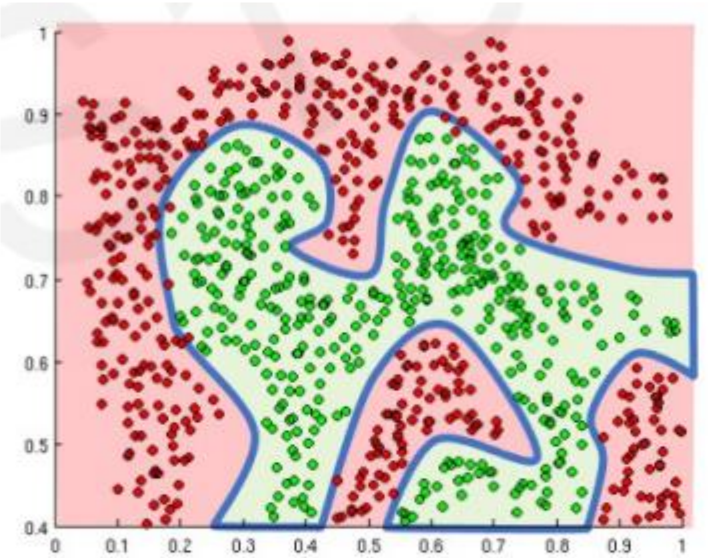
↓ мала

Застосовуємо на практиці

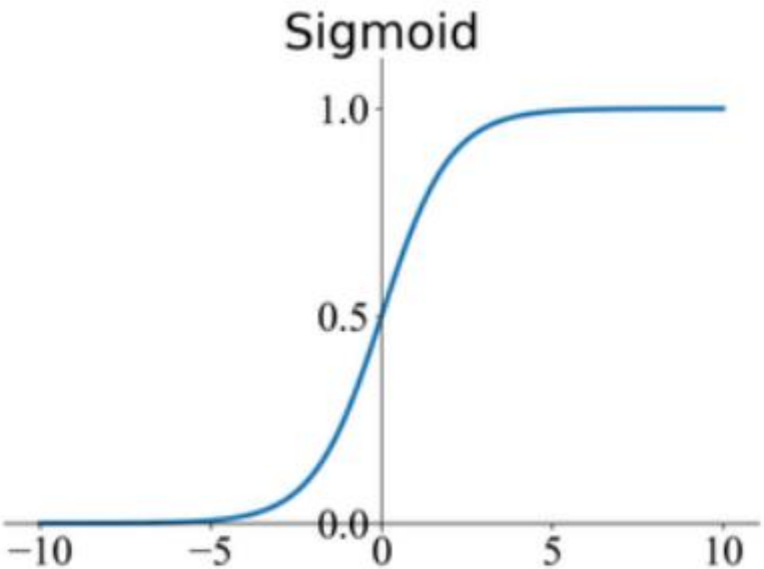
Activation functions

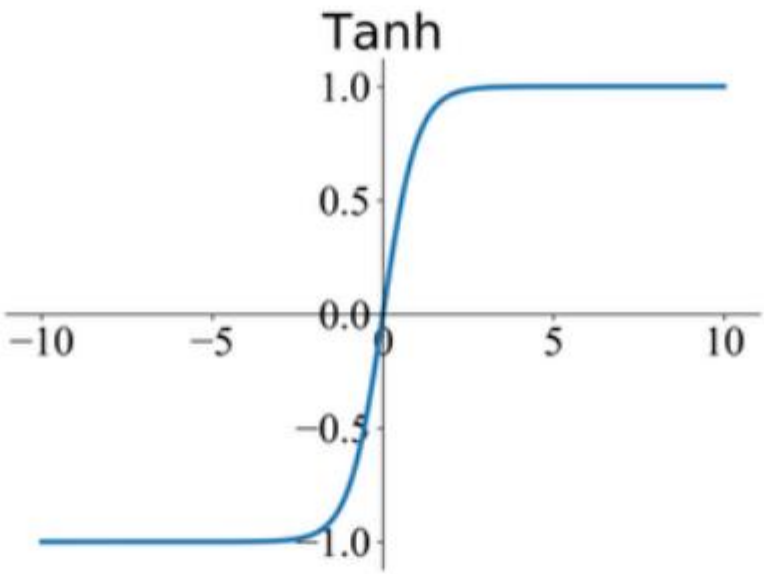






//introtodeeplearning.com/2021/slides/6S191_MIT_DeepLearning_L1.pdf





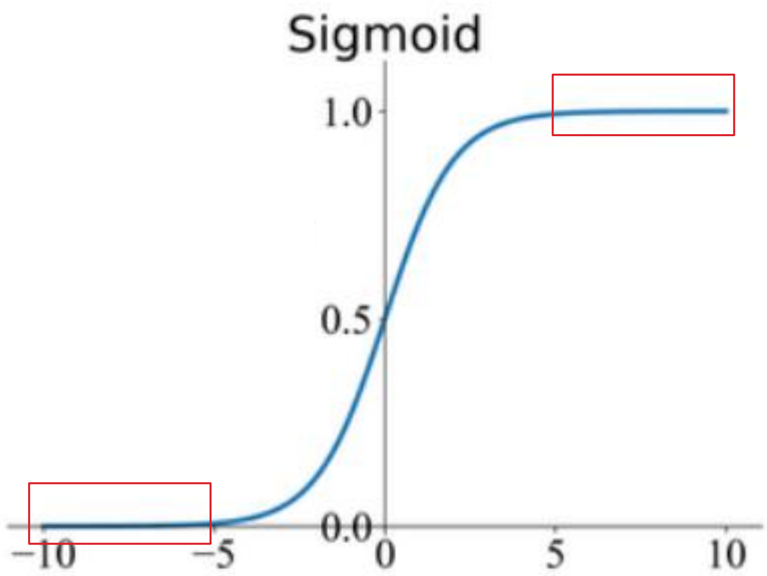
$$g(z) = \frac{1}{1 + e^{-z}}$$

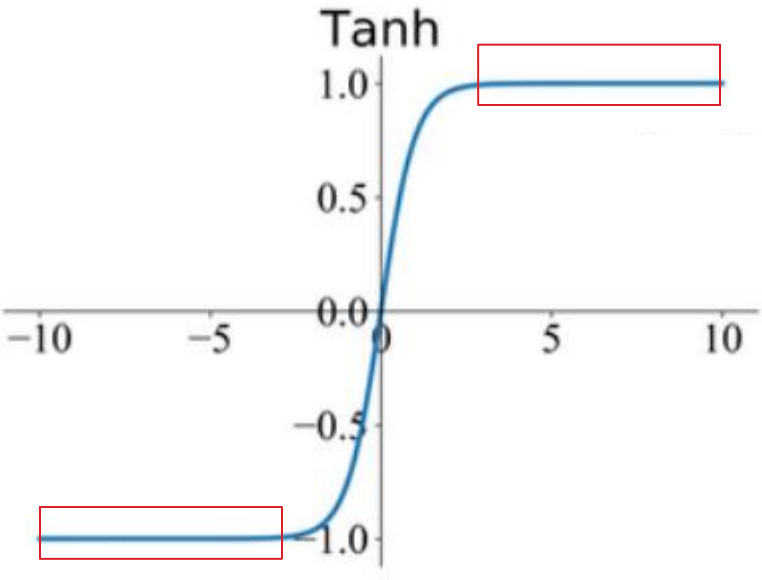
$$g'(z) = g(z)(1 - g(z))$$

$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

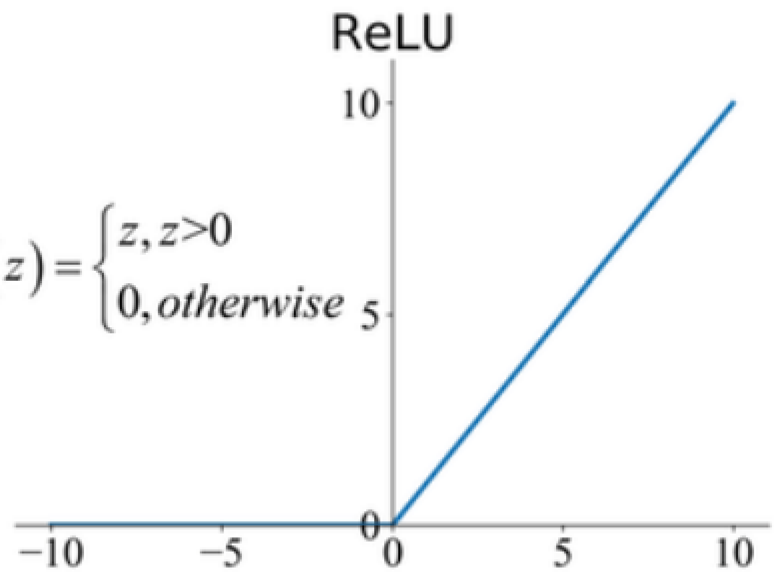
Vanishing gradient problem





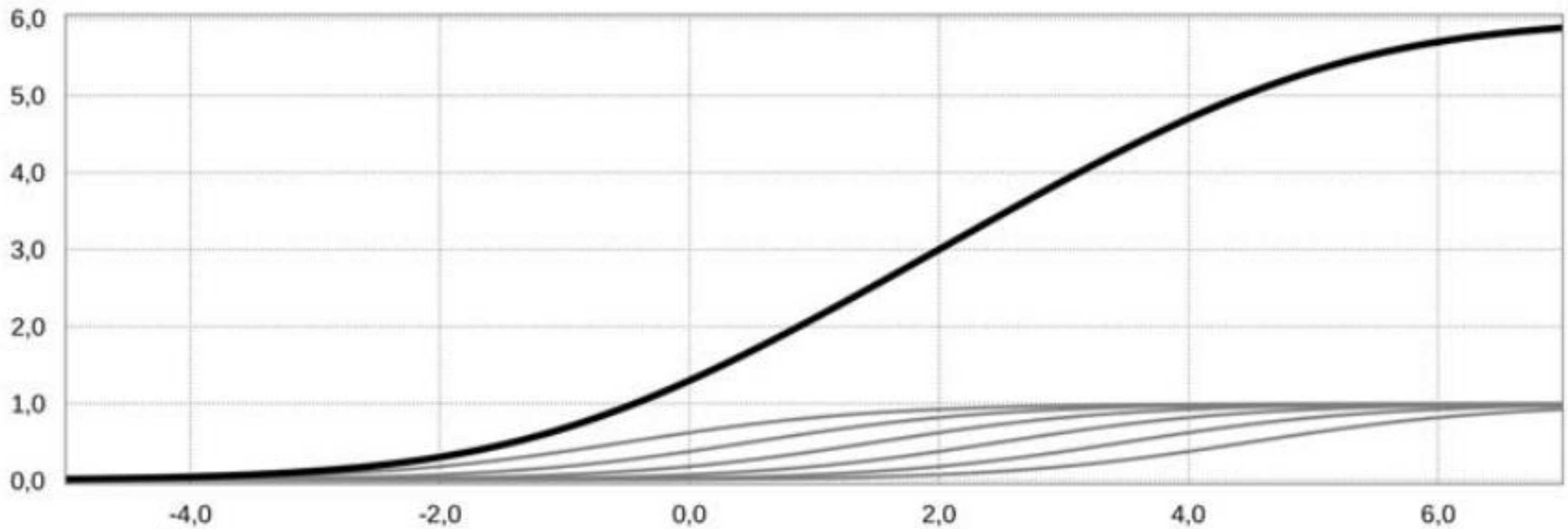
Rectified Linear Unit

$$\mathrm{ReLU}(z) = \begin{cases} z, & z > 0 \\ 0, & \textit{otherwise} \end{cases}$$



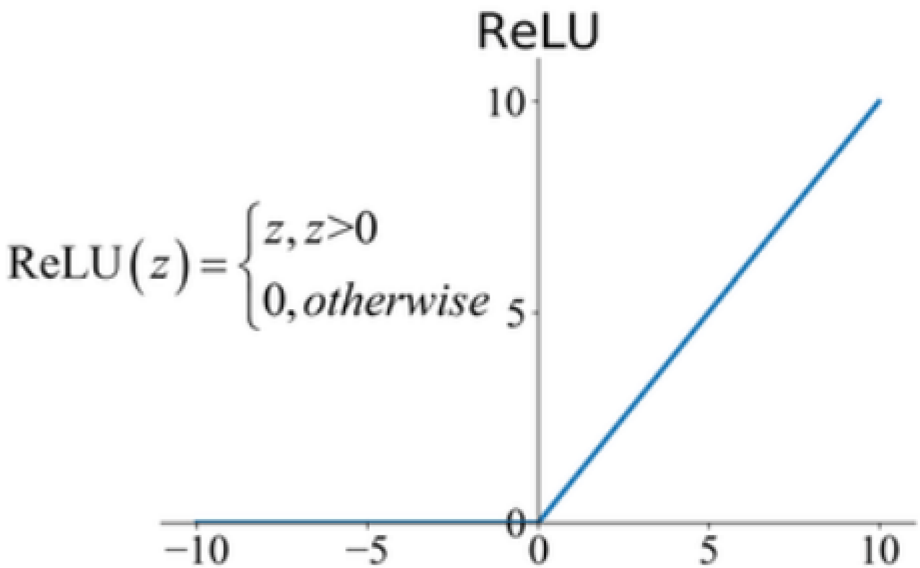
$$g(z) = \max(0, z)$$

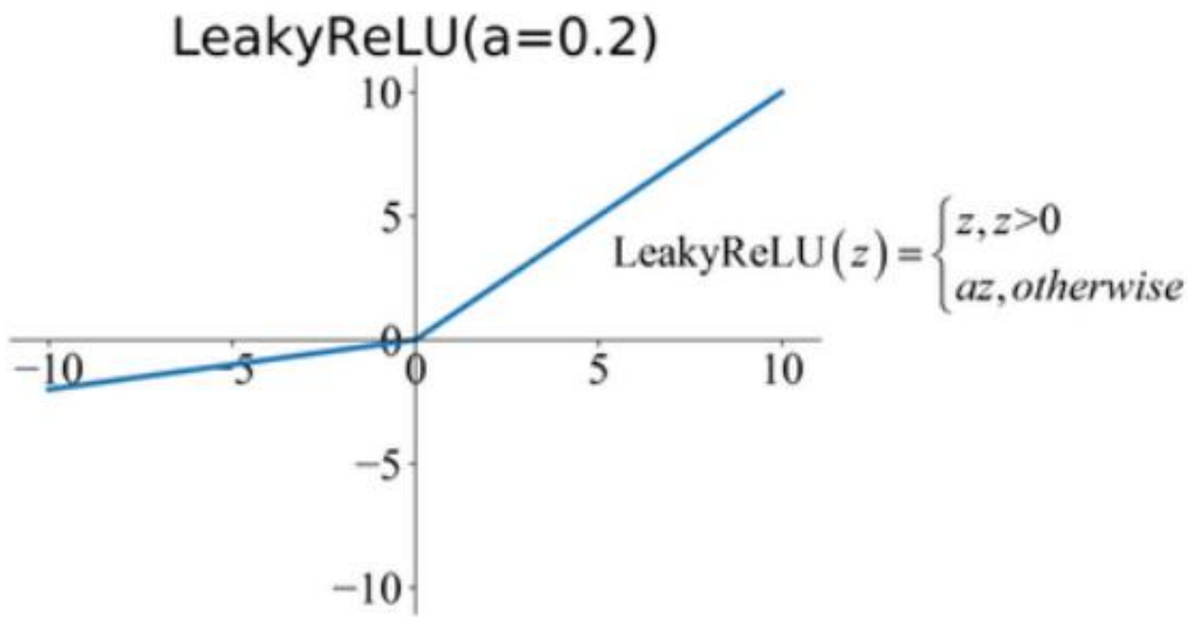
$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

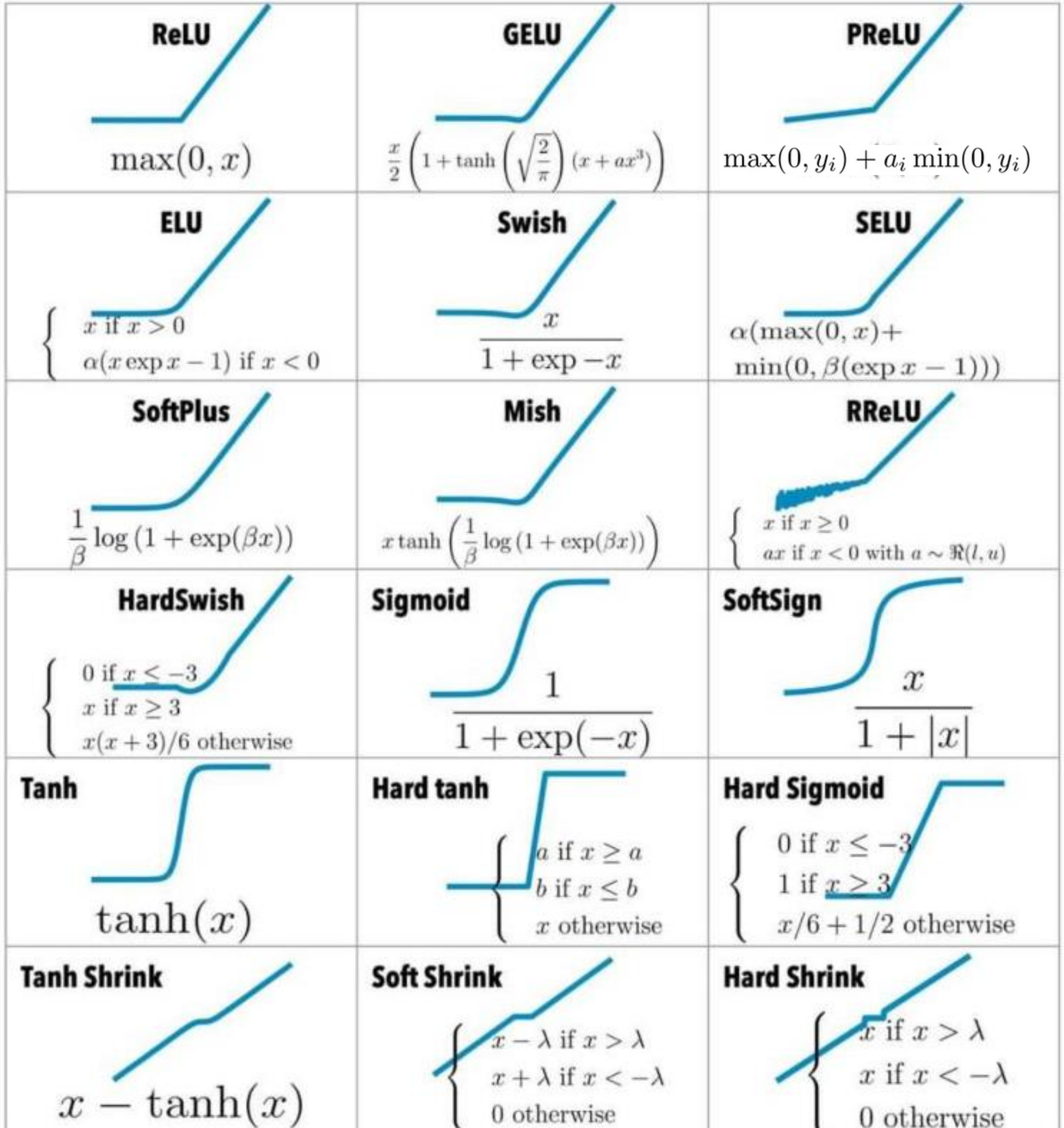


Сумма шести сигмоидов

$$f(x) = \sigma\left(x + \frac{1}{2}\right) + \sigma\left(x - \frac{1}{2}\right) + \sigma\left(x - \frac{3}{2}\right) + \sigma\left(x - \frac{5}{2}\right) + \ldots.$$

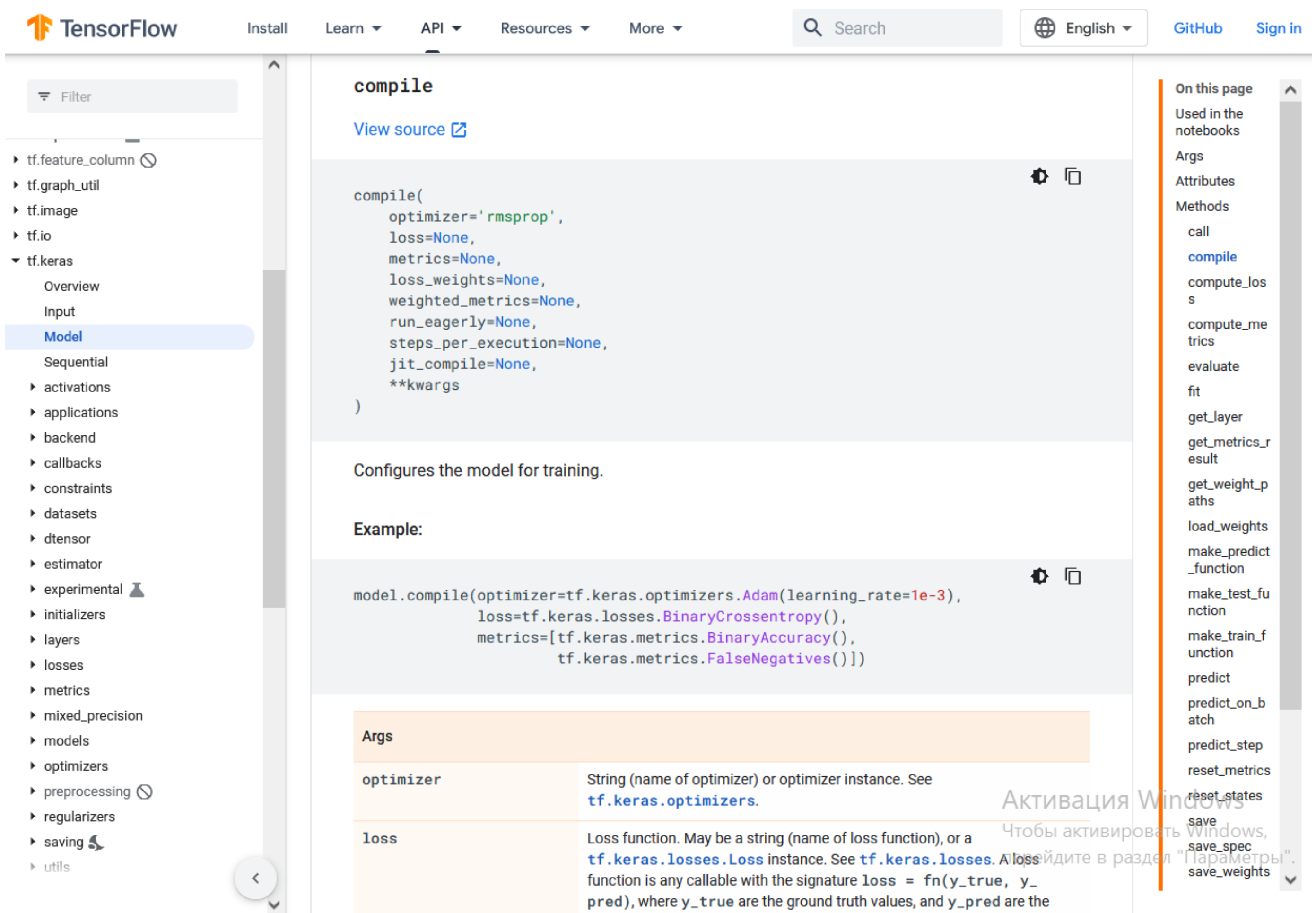

ReLU
10
5
0
−10
−5
0
5
10
$\mathrm{ReLU}(z)=\begin{cases} z, z>0 \\ 0, otherwise \end{cases}$


LeakyReLU(a=0.2)
10
5
0
−5
−10
−10
−5
0
5
10
$\mathrm{LeakyReLU}(z)=\begin{cases} z, z>0 \\ az, otherwise \end{cases}$


ReLU
$\max(0, x)$
GELU
$\frac{x}{2}\left(1+\tanh\left(\sqrt{\frac{2}{\pi}}\right)(x+ax^3)\right)$
PReLU
$\max(0, y_i) + a_i \min(0, y_i)$
ELU
$\begin{cases} x \text{ if } x > 0 \\ \alpha(x \exp x - 1) \text{ if } x < 0 \end{cases}$
Swish
$\frac{x}{1+\exp -x}$
SELU
$\alpha(\max(0, x) +$
$\min(0, \beta(\exp x - 1)))$
SoftPlus
$\frac{1}{\beta}\log(1+\exp(\beta x))$
Mish
$x \tanh\left(\frac{1}{\beta}\log(1+\exp(\beta x))\right)$
RReLU
$\begin{cases} x \text{ if } x \geq 0 \\ ax \text{ if } x < 0 \text{ with } a \sim \mathfrak{R}(l, u) \end{cases}$
HardSwish
$\begin{cases} 0 \text{ if } x \leq -3 \\ x \text{ if } x \geq 3 \\ x(x+3)/6 \text{ otherwise} \end{cases}$
Sigmoid
$\frac{1}{1+\exp(-x)}$
SoftSign
$\frac{x}{1+|x|}$
Tanh
$\tanh(x)$
Hard tanh
$\begin{cases} a \text{ if } x \geq a \\ b \text{ if } x \leq b \\ x \text{ otherwise} \end{cases}$
Hard Sigmoid
$\begin{cases} 0 \text{ if } x \leq -3 \\ 1 \text{ if } x \geq 3 \\ x/6 + 1/2 \text{ otherwise} \end{cases}$
Tanh Shrink
$x - \tanh(x)$
Soft Shrink
$\begin{cases} x - \lambda \text{ if } x > \lambda \\ x + \lambda \text{ if } x < -\lambda \\ 0 \text{ otherwise} \end{cases}$
Hard Shrink
$\begin{cases} x \text{ if } x > \lambda \\ x \text{ if } x < -\lambda \\ 0 \text{ otherwise} \end{cases}$

Model training with TensorFlow

```python
model.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])
```



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

Вбудовані Loss функції

Module: tf.keras.losses

TensorFlow 1 version

Public API for tf.keras.losses namespace.

Classes

`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 labels and predictions.

`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` .

Реалізація власної Loss функції

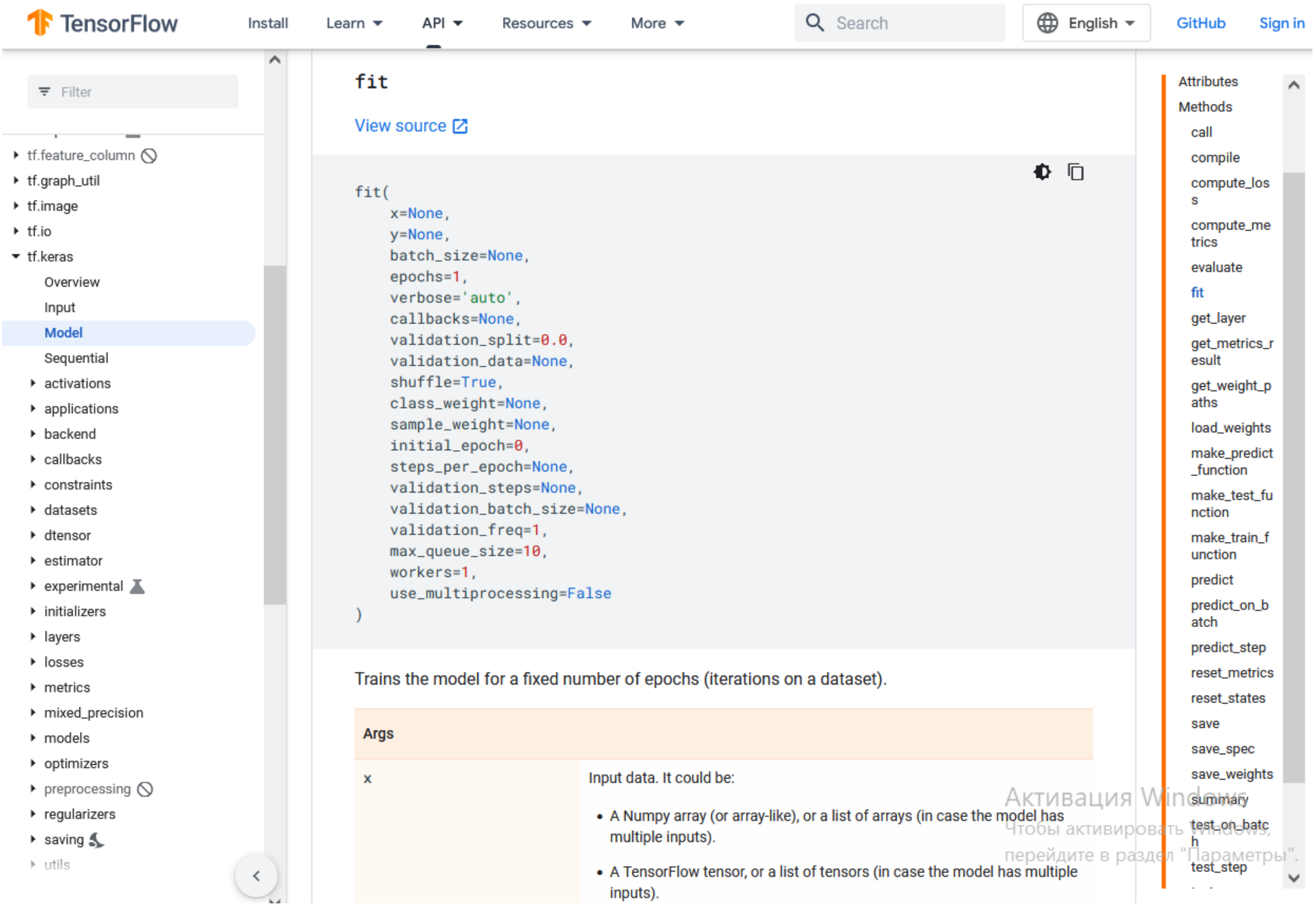
```
def my_loss_function(y_true, y_pred):
    return losses
```

```
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='my_huber_loss')




def my_huber_loss(y_true, y_pred):
    threshold = 1
    error = y_true - y_pred
    is_small_error = tf.abs(error) <= threshold
    small_error_loss = tf.square(error) / 2
    big_error_loss = threshold * (tf.abs(error) - (0.5 * threshold))
    return tf.where(is_small_error, small_error_loss, big_error_loss)
```

$$L_\delta(a) \begin{cases} \frac{1}{2}a^2 & for|a| \le \delta \\ \delta \times (|a| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$$

https://www.coursera.org/learn/custom-models-layers-loss-functions-with-tensorflow/home/info

```python
history = model.fit(x_train, y_train, validation_split = 0.2, epochs=10, batch_size = 32)
```



fit

View source

```python
fit(
    x=None,
    y=None,
    batch_size=None,
    epochs=1,
    verbose='auto',
    callbacks=None,
    validation_split=0.0,
    validation_data=None,
    shuffle=True,
    class_weight=None,
    sample_weight=None,
    initial_epoch=0,
    steps_per_epoch=None,
    validation_steps=None,
    validation_batch_size=None,
    validation_freq=1,
    max_queue_size=10,
    workers=1,
    use_multiprocessing=False
)
```

Trains the model for a fixed number of epochs (iterations on a dataset).

Args	
x	Input data. It could be: • A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs). • A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).

Реалізація побудови графіків функцій

```
import matplotlib.pyplot as plt
```

```
history = model.fit(train_ds,
          validation_data=val_ds,
          epochs=10)
```

```
acc      = history.history[     'accuracy' ]
val_acc  = history.history[ 'val_accuracy' ]
loss     = history.history[    'loss' ]
val_loss = history.history['val_loss' ]

epochs   = range(len(acc)) # Get number of epochs

plt.plot  ( epochs,     acc )
plt.plot  ( epochs, val_acc )
plt.title ('Training and validation accuracy')
plt.figure()

plt.plot  ( epochs,     loss )
plt.plot  ( epochs, val_loss )
plt.title ('Training and validation loss'   )
```