Deep Learning methods for pattern recognition

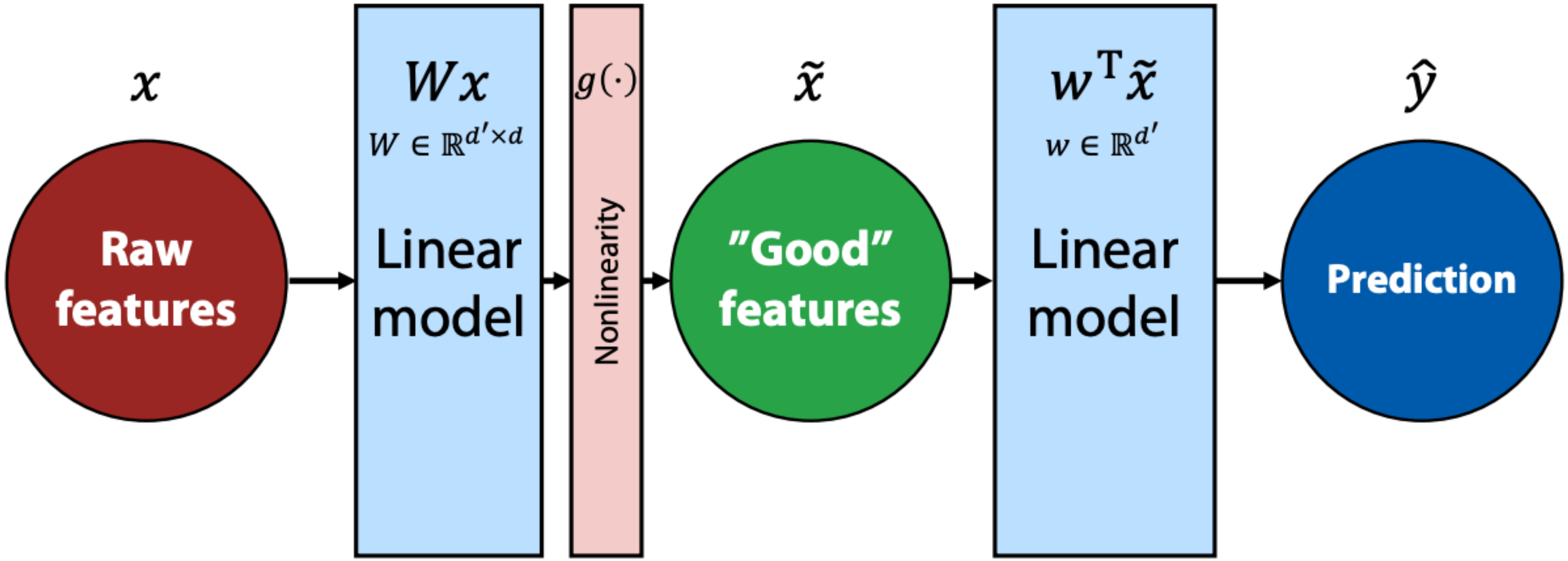
Ahmed Hammad

High Energy Accelerator Research Organization (KEK), Japan.

4^{th} summer school at CTP, BUE

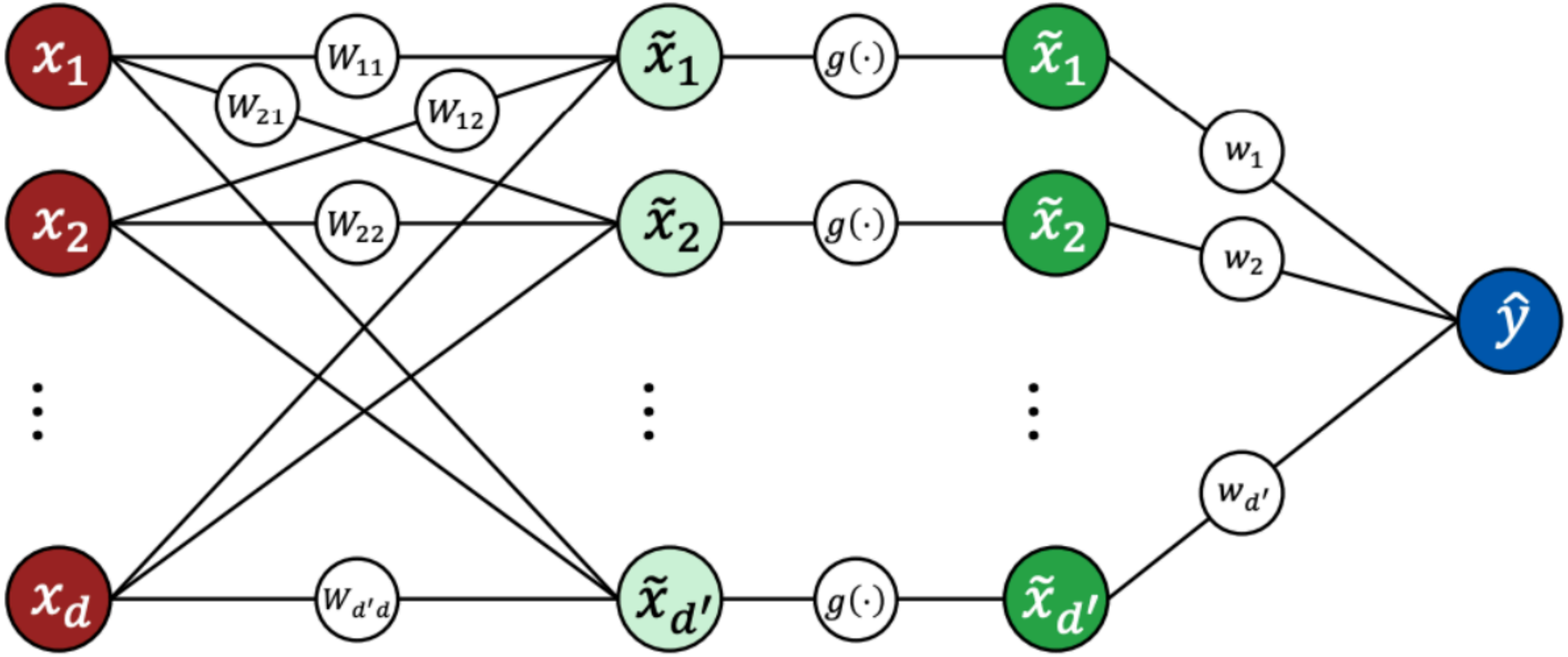
Second lecture

Fully connected deep learning



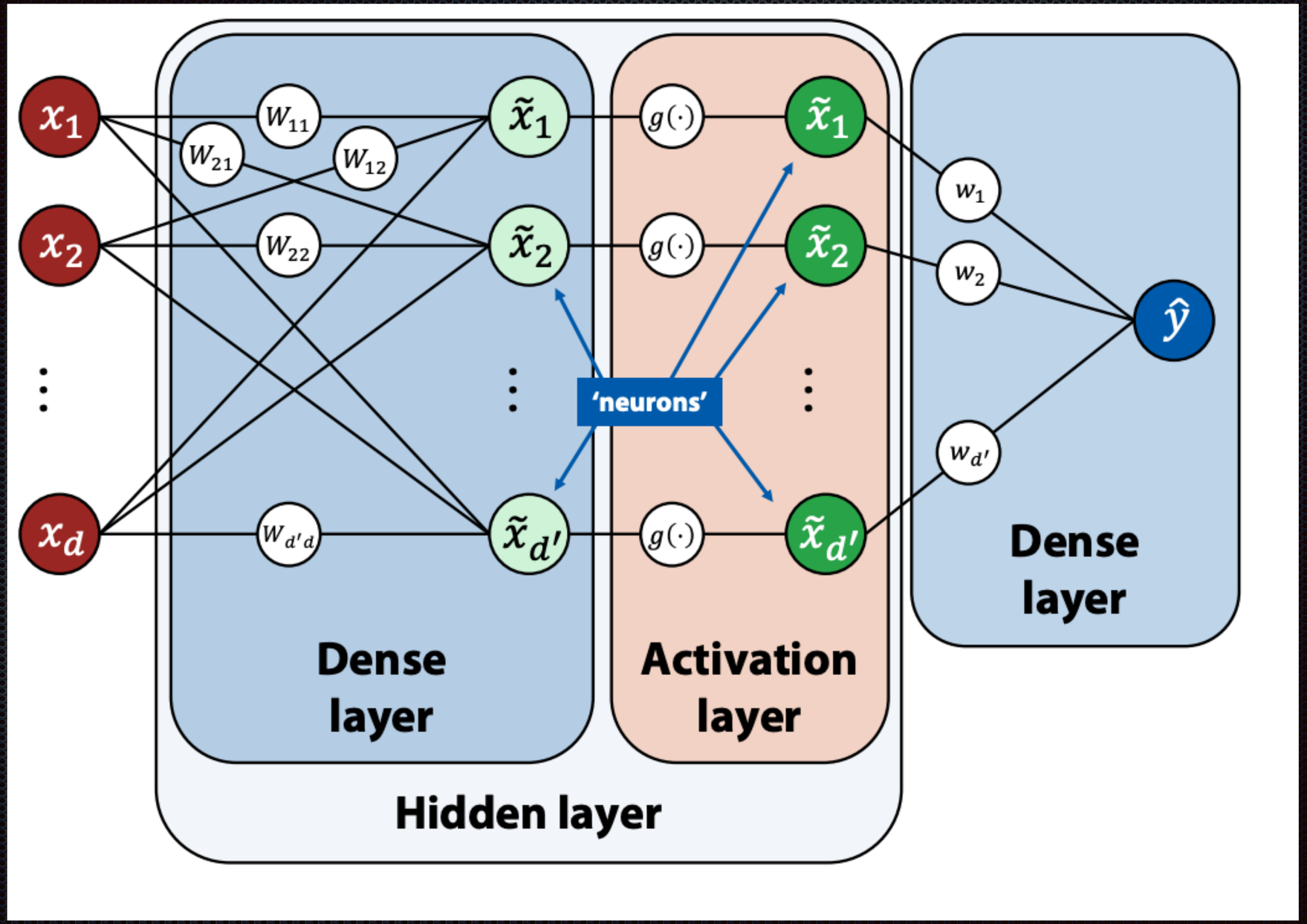
$$\hat{y} = w^T\tilde{x} = w^T g(Wx)$$

Fully connected deep learning

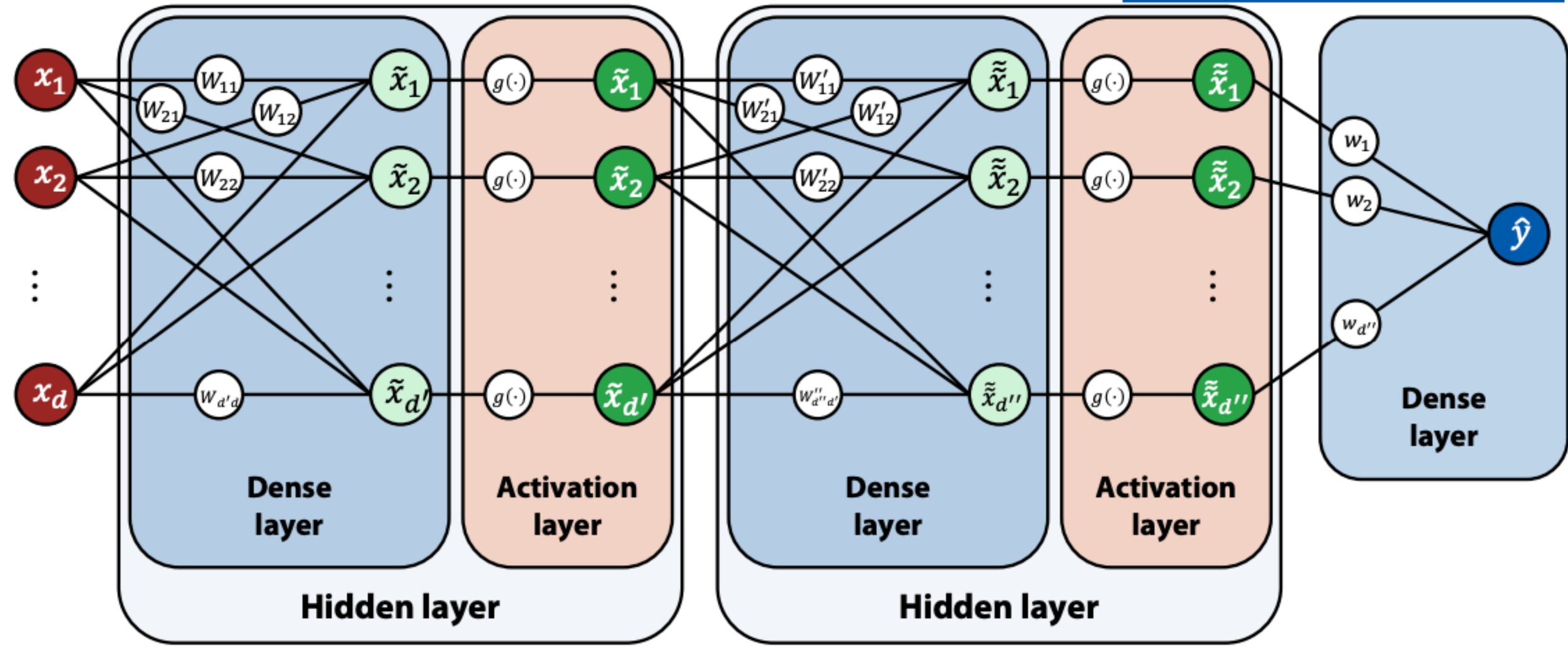


$$\hat{y} = w^{\mathrm{T}}\tilde{x} = w^{\mathrm{T}}g(Wx) = \sum_j \left[w_j g\left(\sum_i W_{ji} x_i \right) \right]$$

Single layer neural network



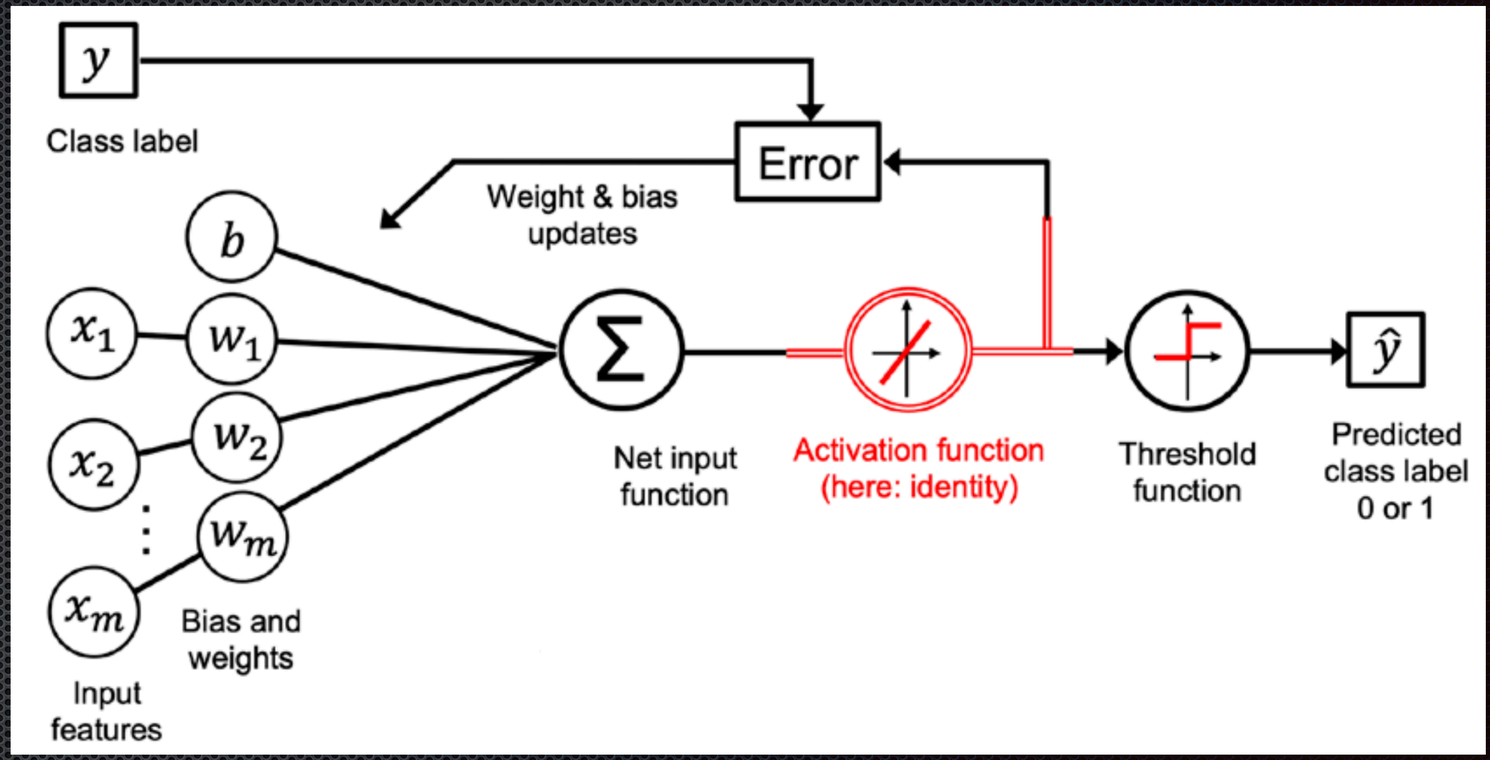
Multilayer neural network



Training deep learning models

Backward propagation

The training loop consists of forward and backward propagation

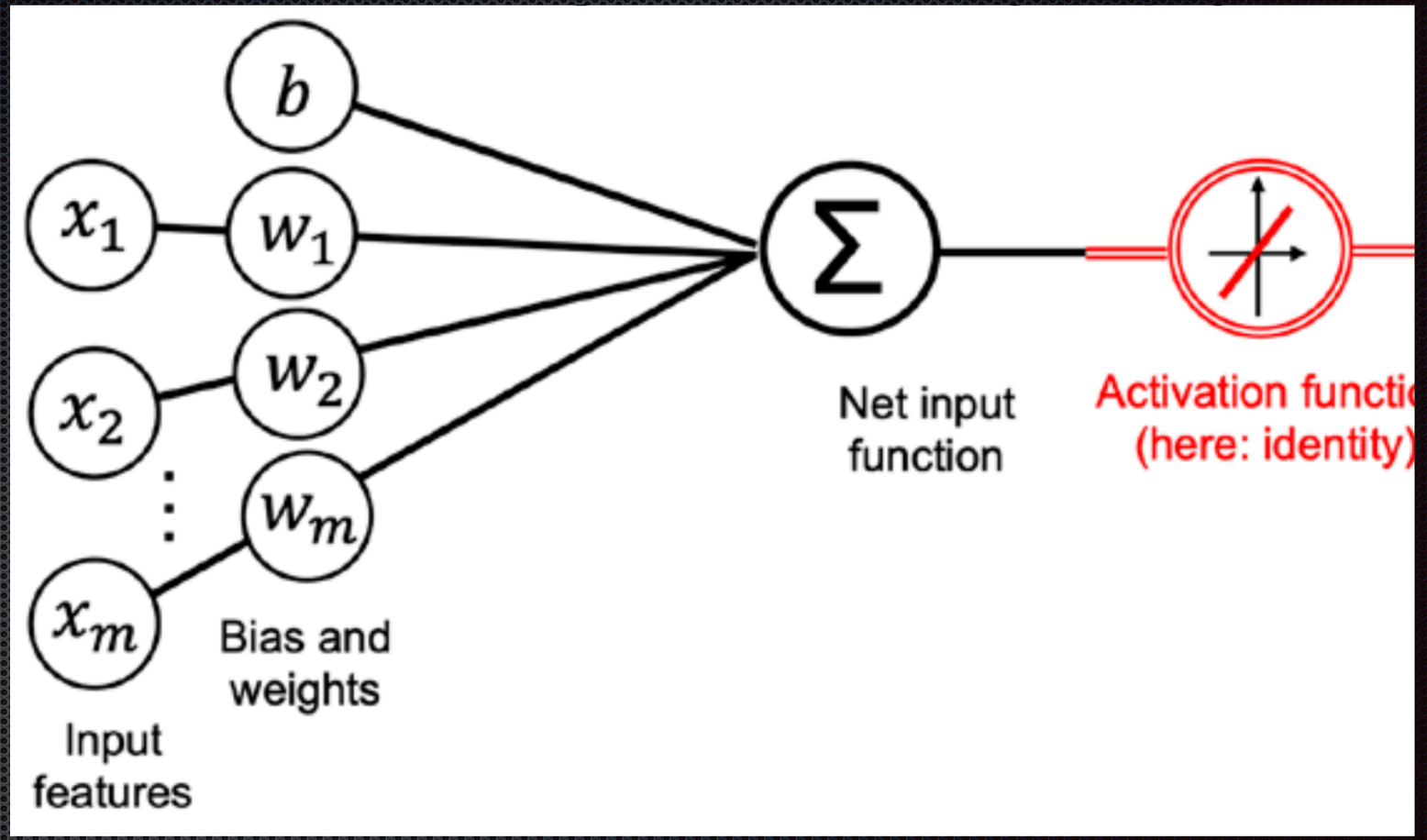


Each iteration is called “epoch”

Forward propagation

Forward propagation–single layer

Non-linear activation function is needed to add nonlinearity to the approximated learned classification boundaries



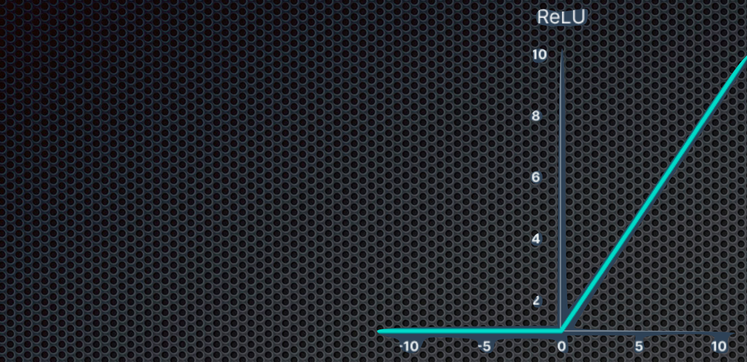
$$\text{Neuron Output} = \text{Activation function}\left(\sum_m \omega_m \, x_m + b\right)$$

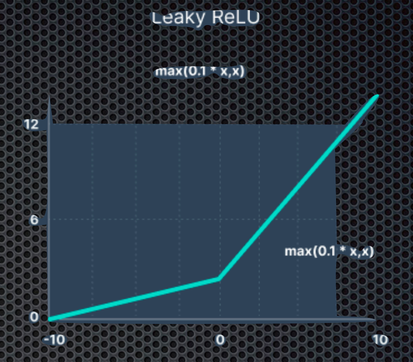
Activation functions

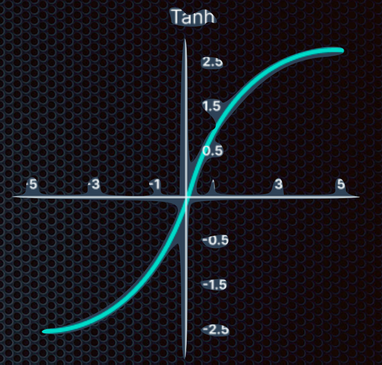
The choice of the activation function at each layer depends no the problem at hand

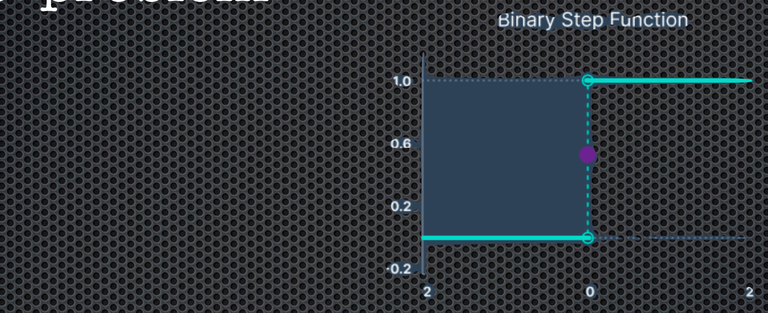
The choice of the activation functions can be optimized using the “Grid search optimization”

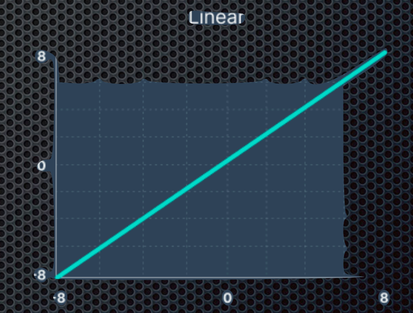
$$\max(0, x)$$

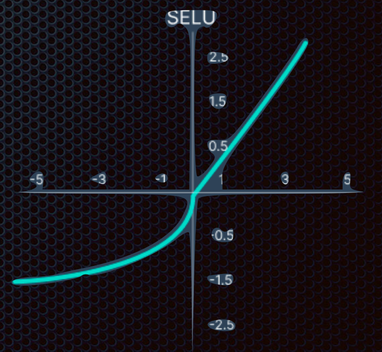




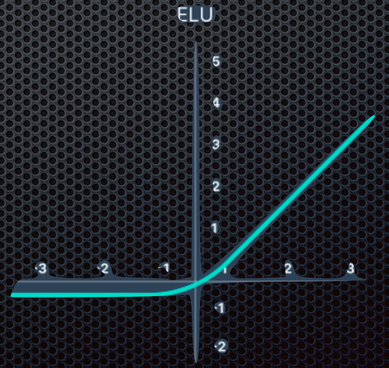




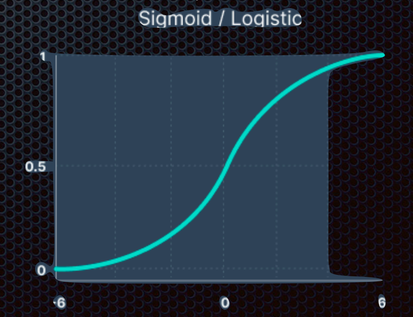


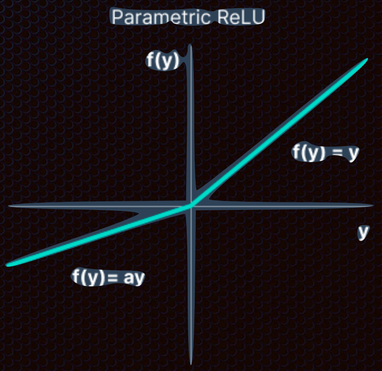


$$\max(\alpha, x)$$



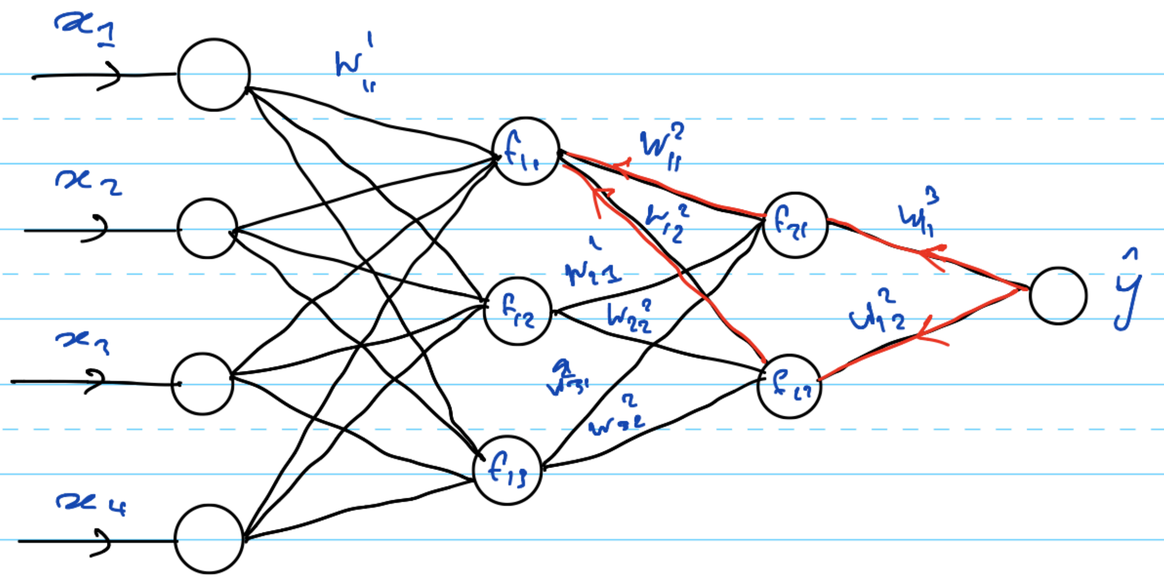
$$\frac{1}{1+e^{-x}}$$





Back propagation-single layer

As discussed before, to update the weights using the gradient descent method we need to compute the derivatives



We use the gradient descent method

$$w_{new}^i = w_{old}^i - \eta \nabla L(w_{old}^i)$$

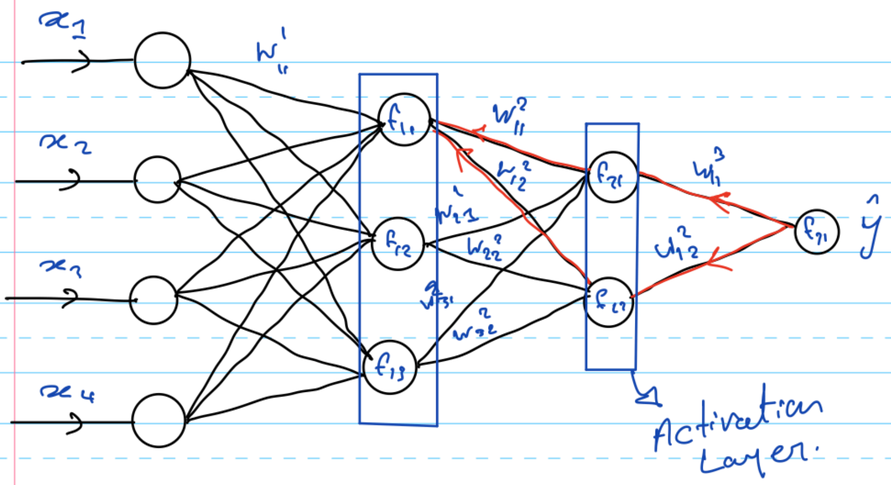
to update the weights in backward mode and minimize the loss function.

Back propagation–single layer

Very tedious work for deeper NN, can you do it yourself ?

To compute the derivatives for the tunable weights, computers use the "Automatic differentiation"

Please take a look at the auto-diff methods in TensorFlow or PyTorch



$L = \frac{1}{2}|y-\hat{y}|^2$, $w^i_{new} = w^i_{old} - \eta \nabla L(w^i_{old})$

Example:

Find the updated weight for w^3_{11}:

$$\frac{\partial L}{\partial w^3_{11}} = \frac{\partial L}{\partial f_{31}} \frac{\partial f_{31}}{\partial w^3_{11}}$$

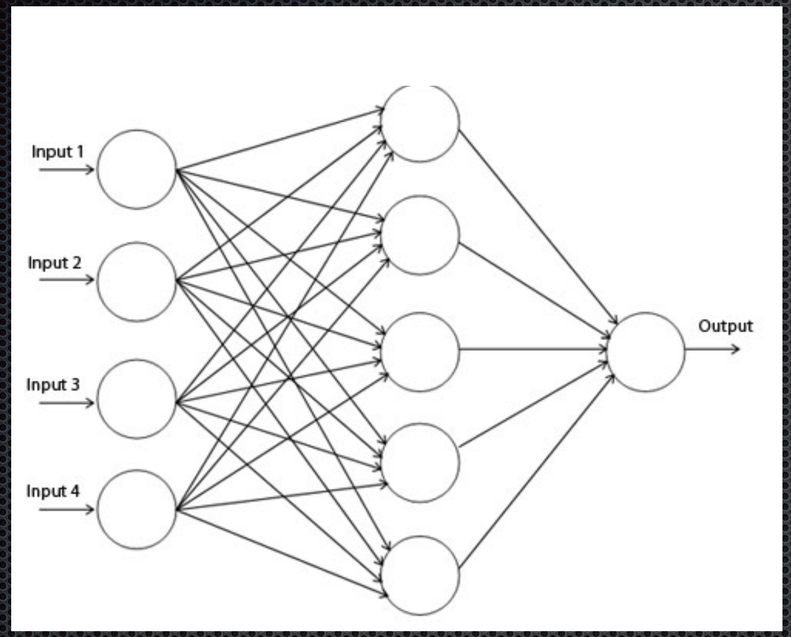
$$w^3_{11,new} = w^3_{11} - \eta * \left[\frac{\partial L}{\partial F_{31}} \frac{\partial f_{31}}{\partial w^3_{11}}\right]$$

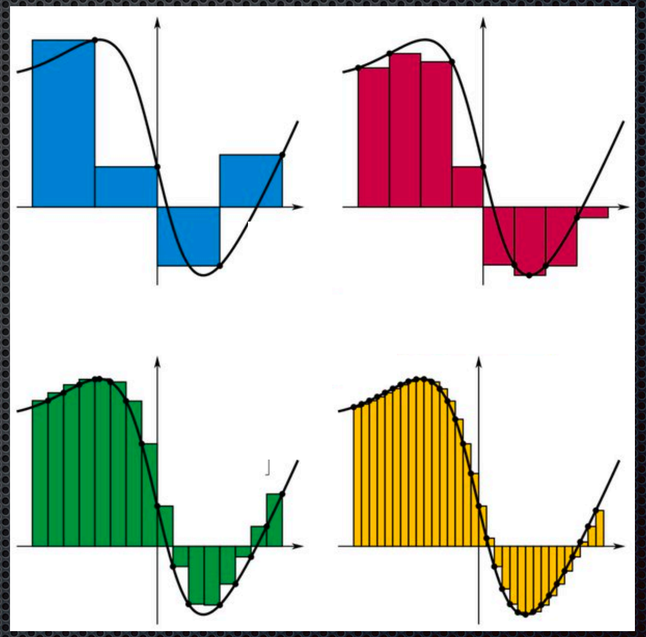
* For w^2_{11}:

$$\frac{\partial L}{\partial w^2_{11}} = \left[\frac{\partial L}{\partial f_{31}} \frac{\partial f_{31}}{\partial w^3_{11}} \frac{\partial f_{21}}{\partial w^2_{11}}\right] + \left[\frac{\partial L}{\partial f_{31}} \frac{\partial f_{31}}{\partial w^3_{12}} \frac{\partial f_{22}}{\partial w^2_{11}}\right]$$

Universal approximation theorem

Given any continuous function, no matter how complicated it is, there is always exist a network that can approximate this function



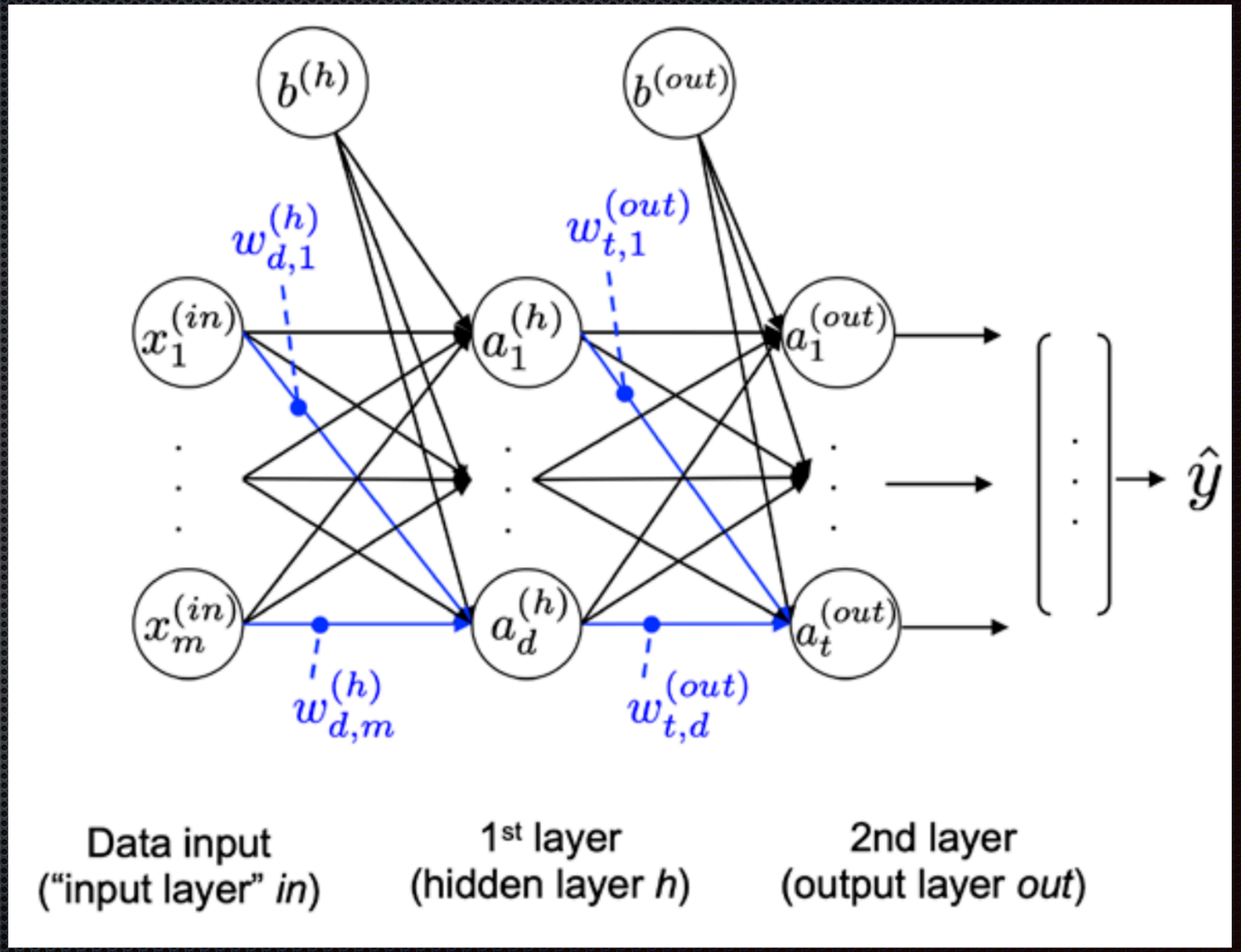


What about nonlinear discontinuous high dimensional function ?

Multi-layer perceptron (MLP)

Adding more hidden layers increases the model ability to approximate more complicated functions

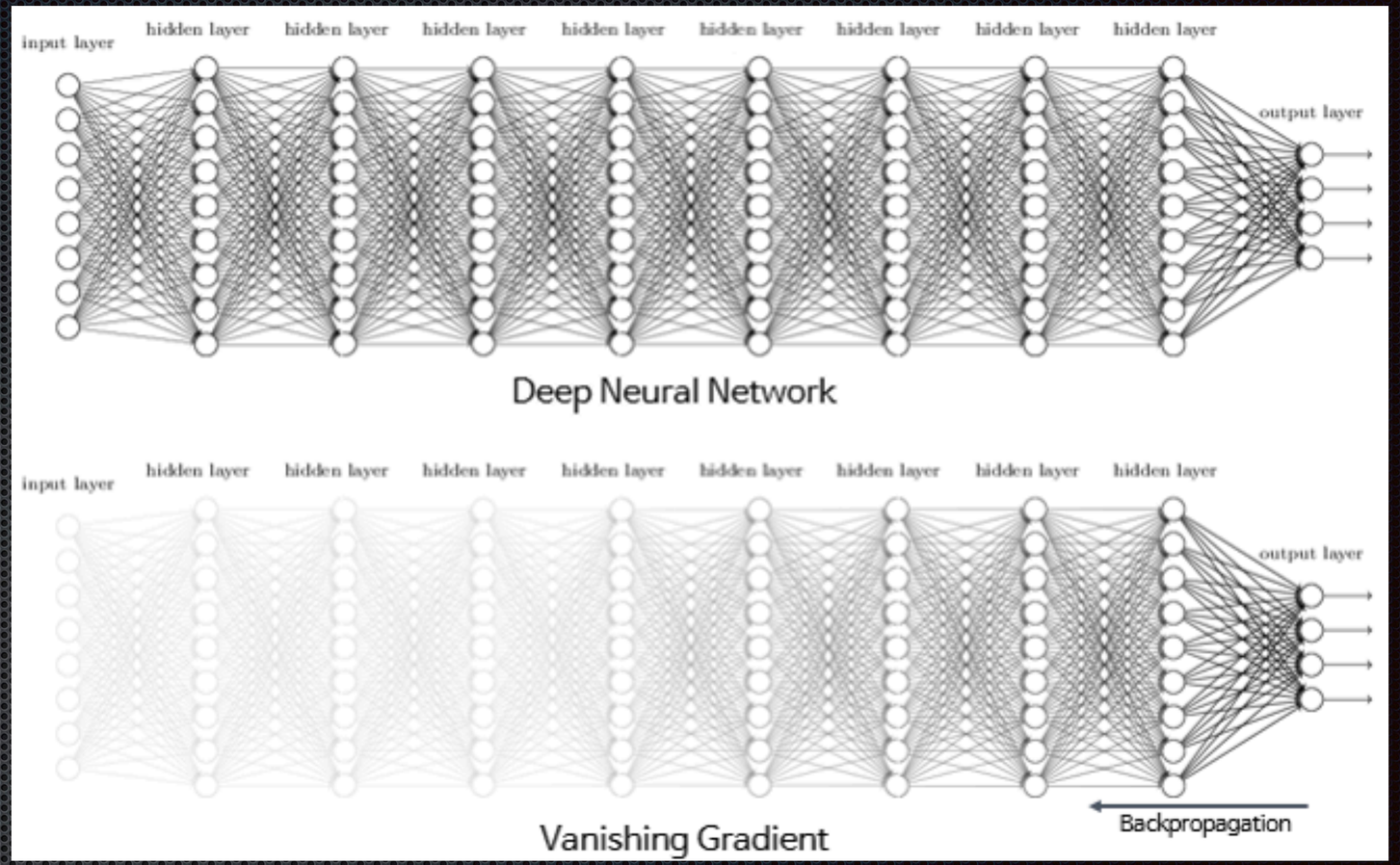
Are we free to use any number of hidden layers?



Vanishing gradient

For very deep models, during the backpropagation the gradient of the initial layers vanishes

$$\omega_i^{\text{new}} = \omega_i^{\text{old}} - \eta \nabla L(\omega_i^{\text{old}}) \approx \omega_i^{\text{old}}$$



Why vanishing gradients takes place?

Exploratory data analysis (EDA)

Prepare the data for deep learning models

- Data Cleaning
- Handling text and categorical attributes
- Features scaling

Data cleaning

- Handel the missing data

Working on real life data, most of the time the data are not standard.

Most of the collected real data suffer from missing information

As ML models cannot process with missing values

Easy to be handled in Pandas

Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
3	Braund, Mr. Owen Harris	male	22.0	1	0.0	A/5 21171	7.2500		S
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1			71.2833	C85	C
3	Heikkinen, Miss. Laina	female		0	0.0	STON/O2. 3101282	7.9250		S
1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0.0	113803	53.1000	C123	S
3	Allen, Mr. William Henry	male	35.0	0	0.0	373450	8.0500		S
...	...	...	...	...	...	...	...	...	...
2	Montvila, Rev. Juozas	male	27.0	0	0.0	211536	13.0000		S
1	Graham, Miss. Margaret Edith	female	19.0	0	0.0	112053	30.0000	B42	S
3	Johnston, Miss. Catherine Helen "Carrie"	female		1	2.0	W./C. 6607	23.4500		S
1	Behr, Mr. Karl Howell	male	26.0	0	0.0	111369	30.0000	C148	C
3	Dooley, Mr. Patrick	male	32.0	0	0.0	370376	7.7500		Q

Pandas.isnull() ⇒ Find the missing values

Pandas.fillna() ⇒ Fill the missing values by extrapolation between the last values

Data cleaning

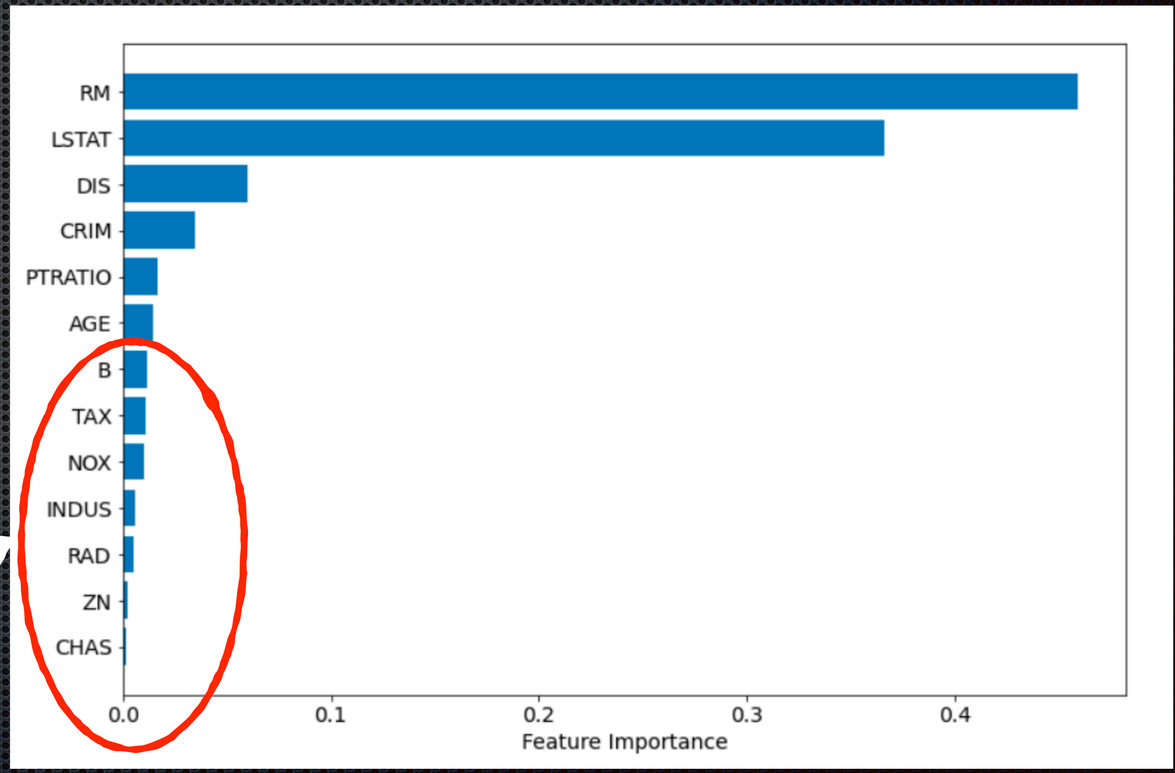
- Feature importance

Deep models learn from large data and keeping more redundant information into the data consume the computational resources and may hinders the model accuracy

Two ways for the selection

Forward selection

Reverse selection



These features can be removed

Handling text data

- One hot encoding

Real life data mostly contains, texts, numbers, characters, etc

ML models deal with numbers only.

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

For multi-class classification problem with cross entropy loss Labels (targets) need to be one hot encoded

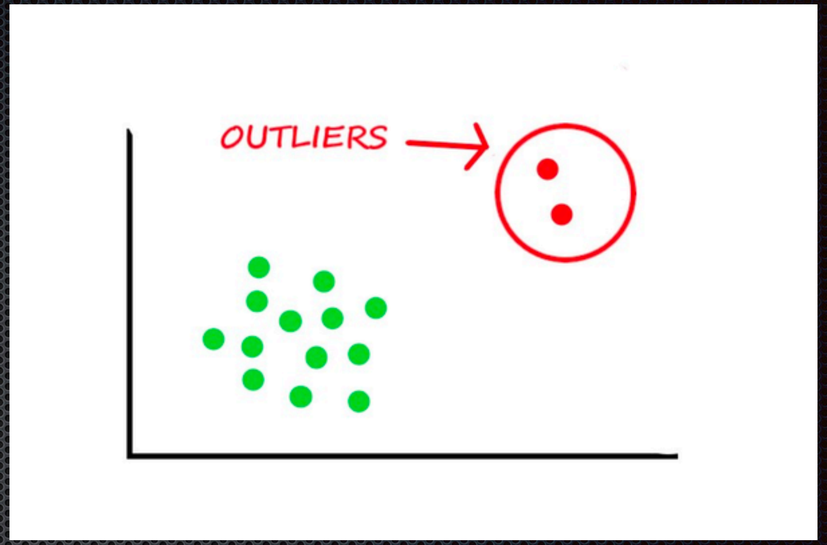
Labels

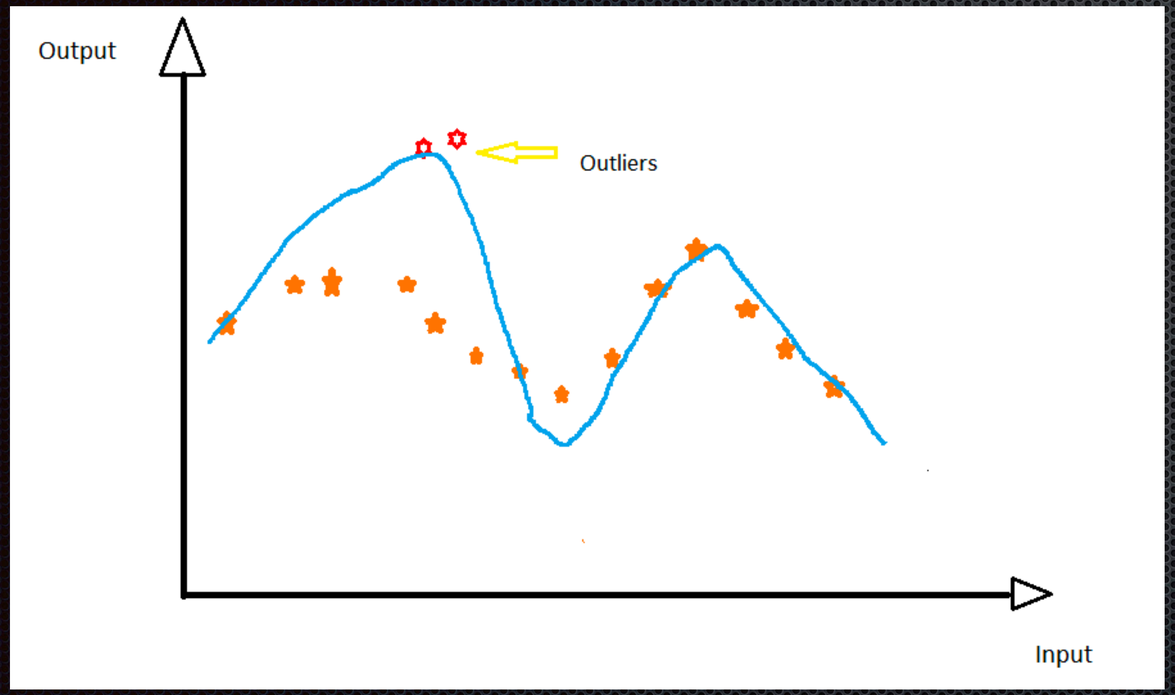
	0	1	2	3	4	5	6	7	8	9	label
0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	5
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
2	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	4
3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	9
5	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2
6	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
7	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	3
8	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
9	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	4

Features scaling

- Outliers

Input data contains different scales and mostly contains outliers



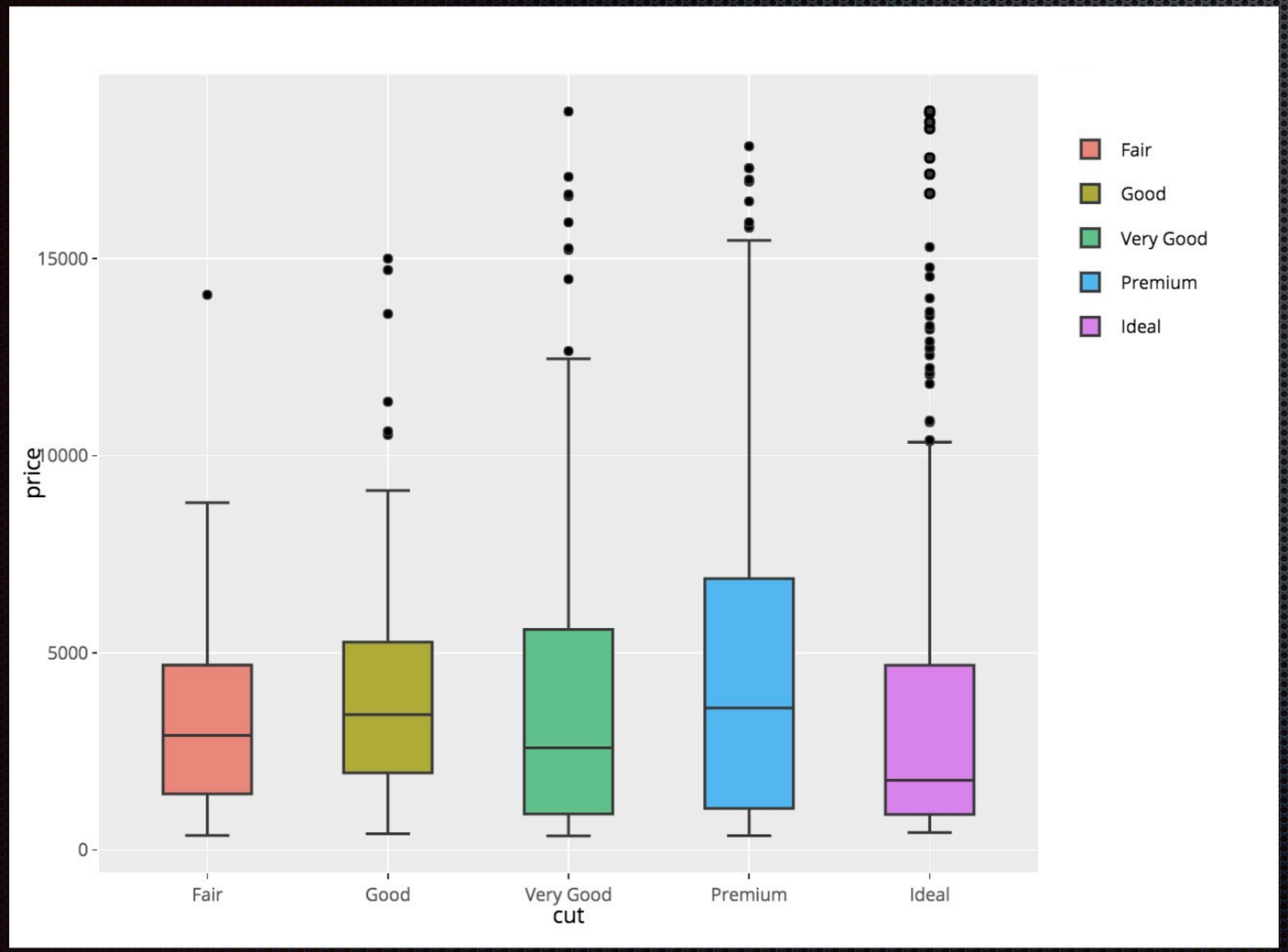


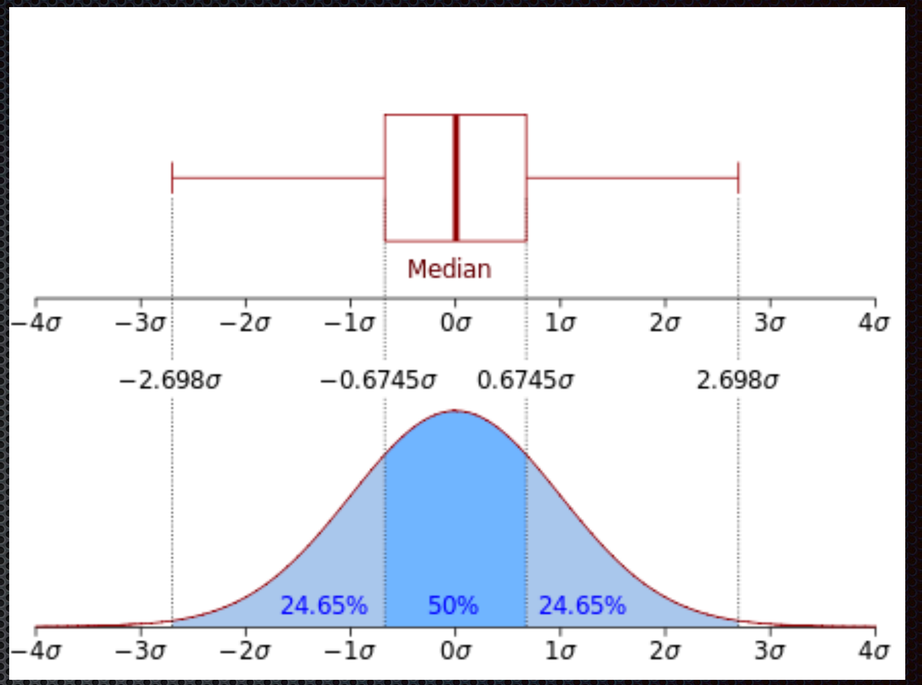
Data with outliers the ML model focuses
in learning the features of the outlier points
Not the features of the whole dataset

Features scaling

- Handling the Outliers

If the size of the outliers is small one can easily remove them





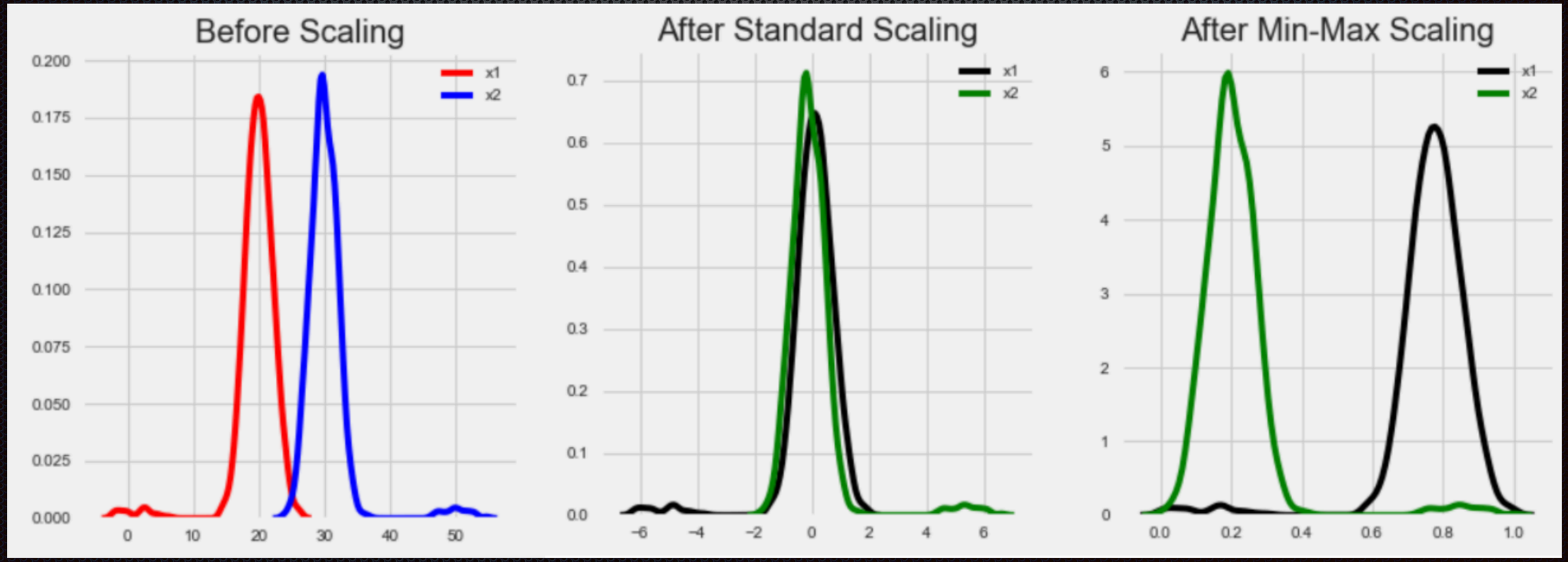
Box diagram for outliers detection

Features scaling

- Handling the Outliers

For large amount of outliers, scaling is needed

$$x^{\text{stand}} = \frac{x - \mu}{\sigma} \qquad x^{\text{minmax}} = \frac{x - x^{\min}}{x^{\max} - x^{\min}}$$

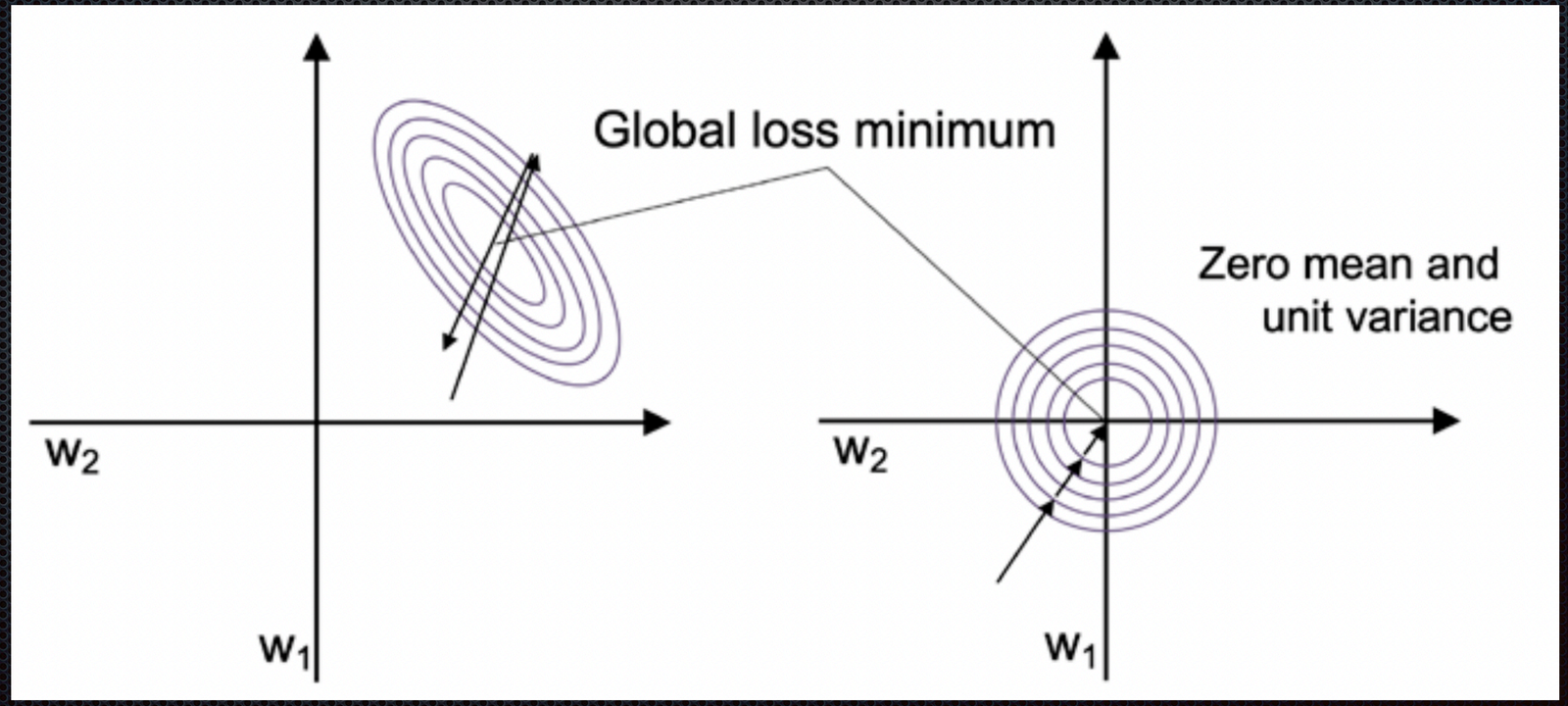


Features scaling

- Handling the Outliers

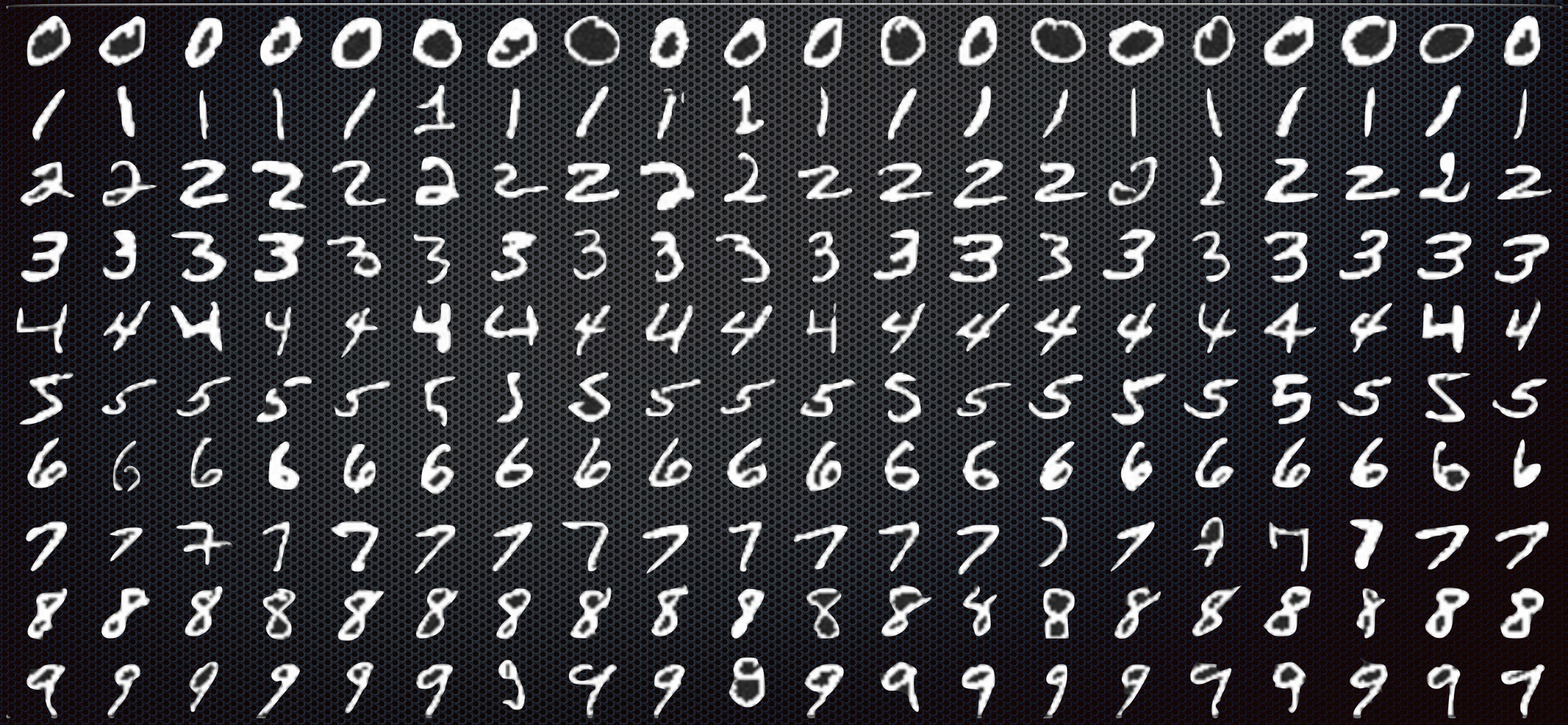
Data normalization is mandatory with gradient descent method.

If the features are on vastly different scales, a learning rate that works well for updating one weight might be too large or too small to update the other weights equally well.

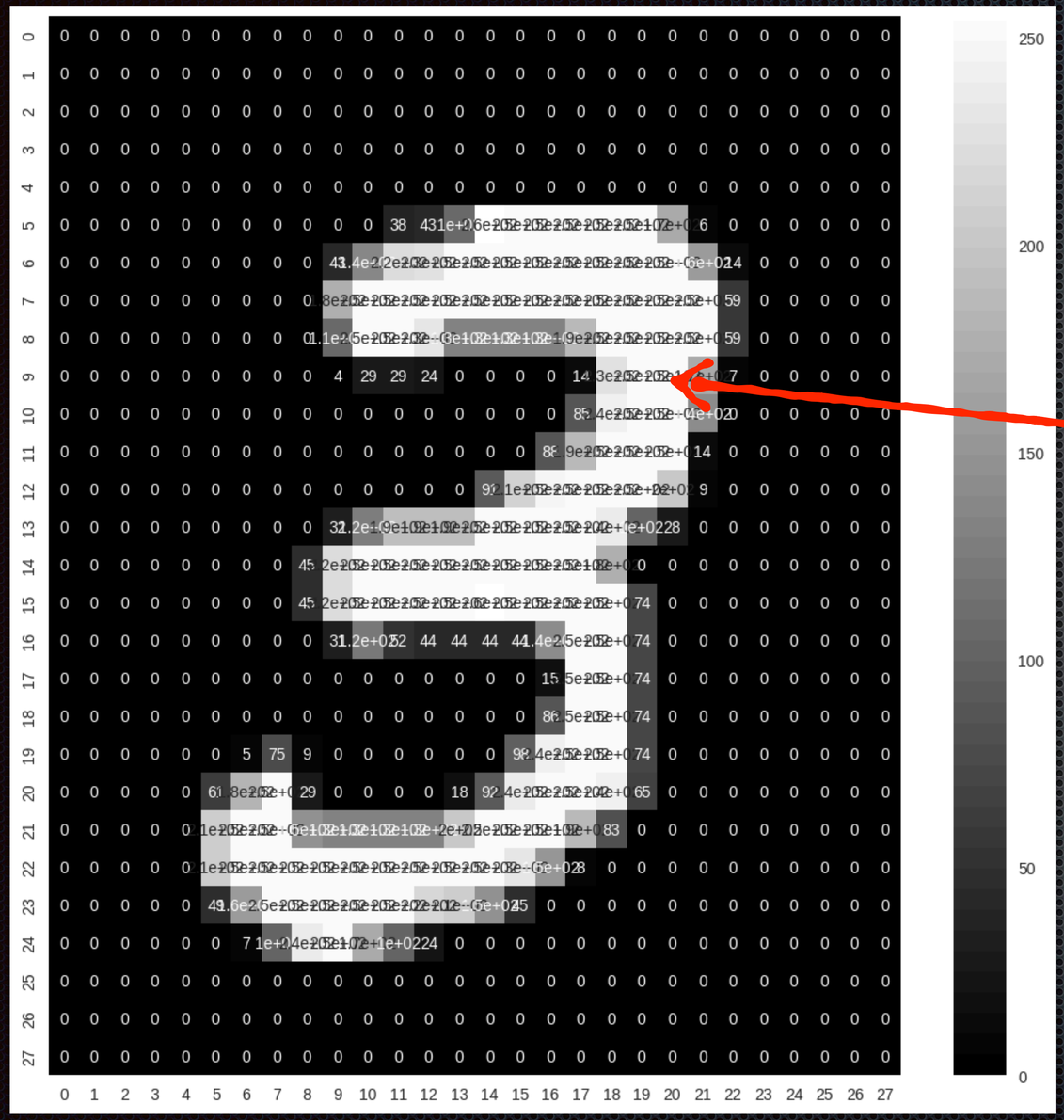


Example – MNIST data

Task: Construct DNN model to classify the hand written digits



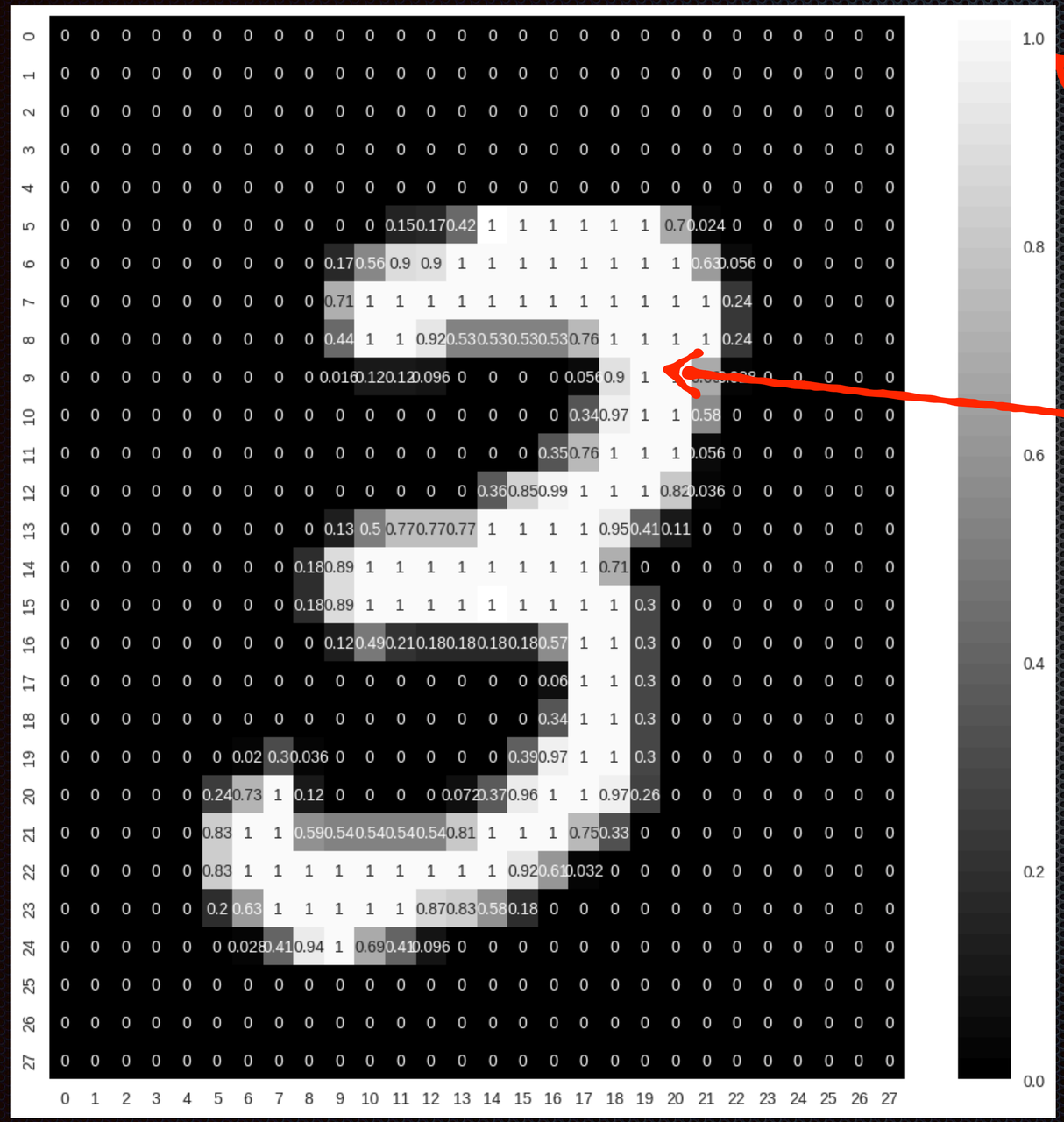
Example – MNIST data



Different scale

Of the pixels

Example – MNIST data

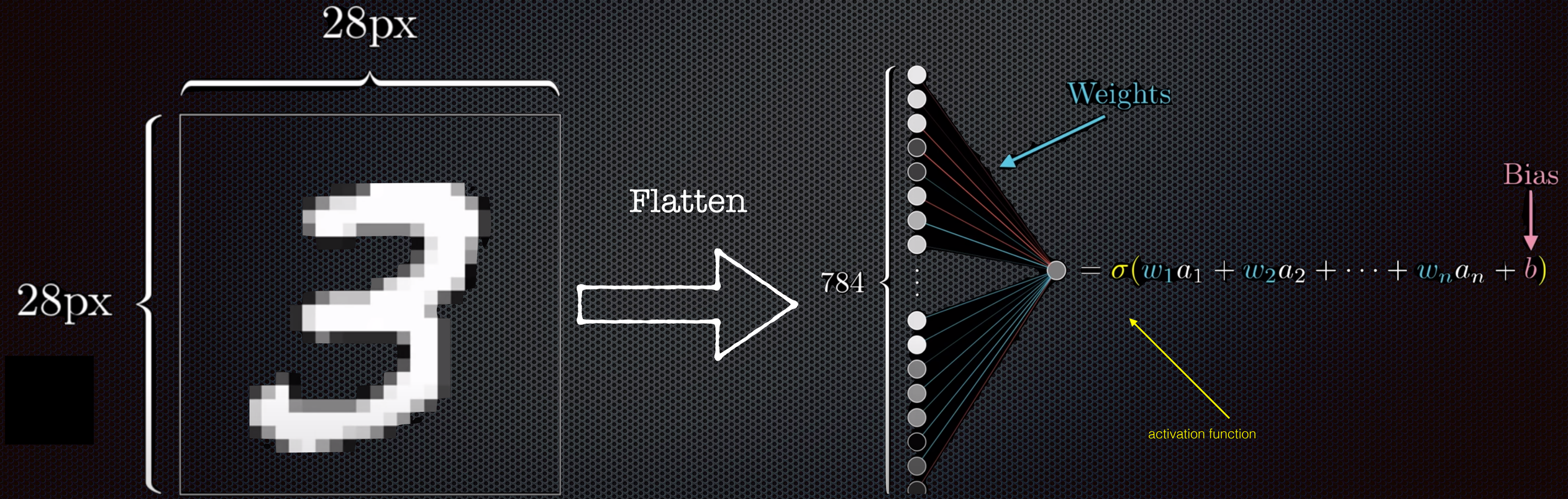


After min-max

Normalization

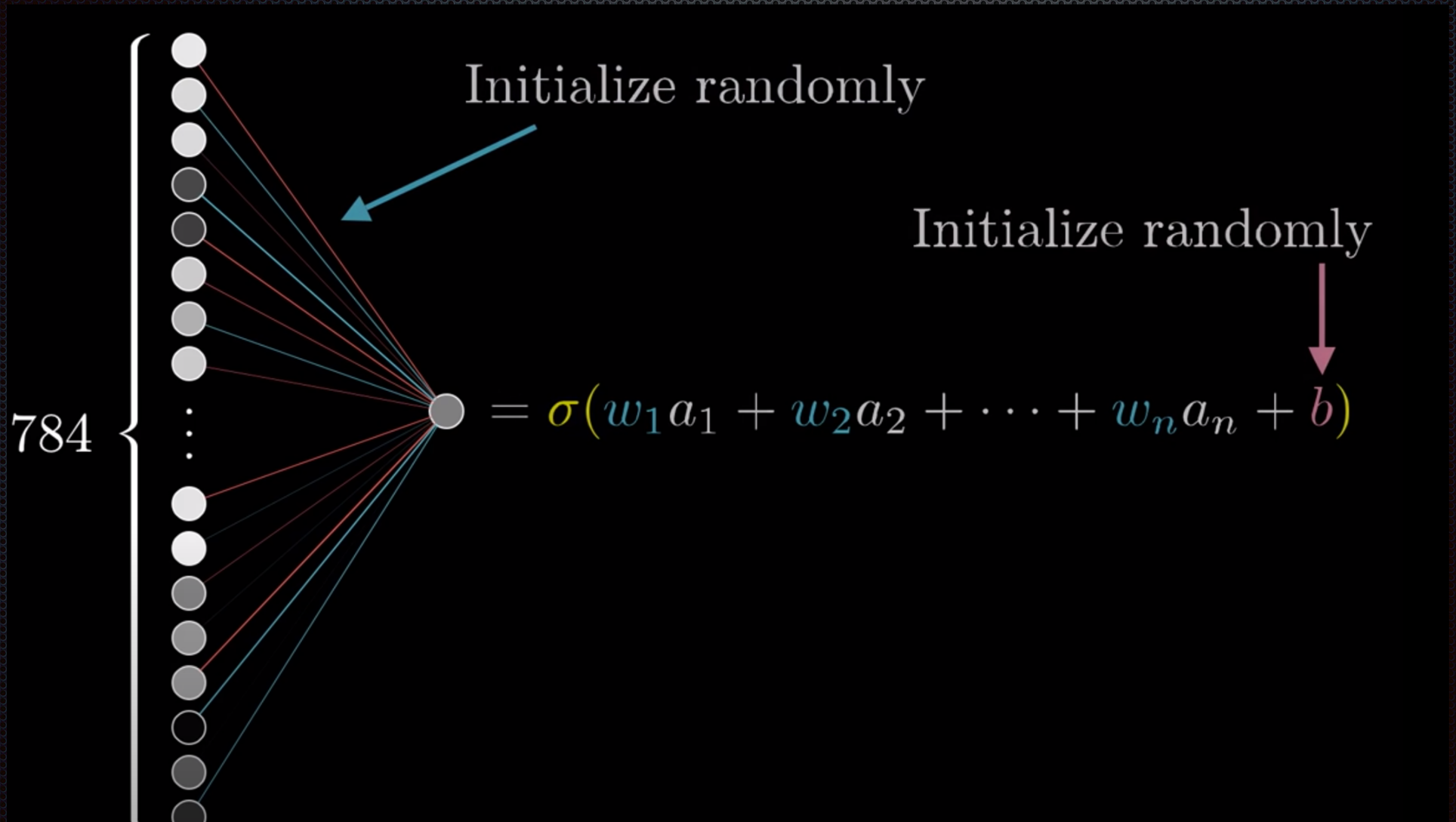
Example – MNIST data

Construct the first neuron in the first layer



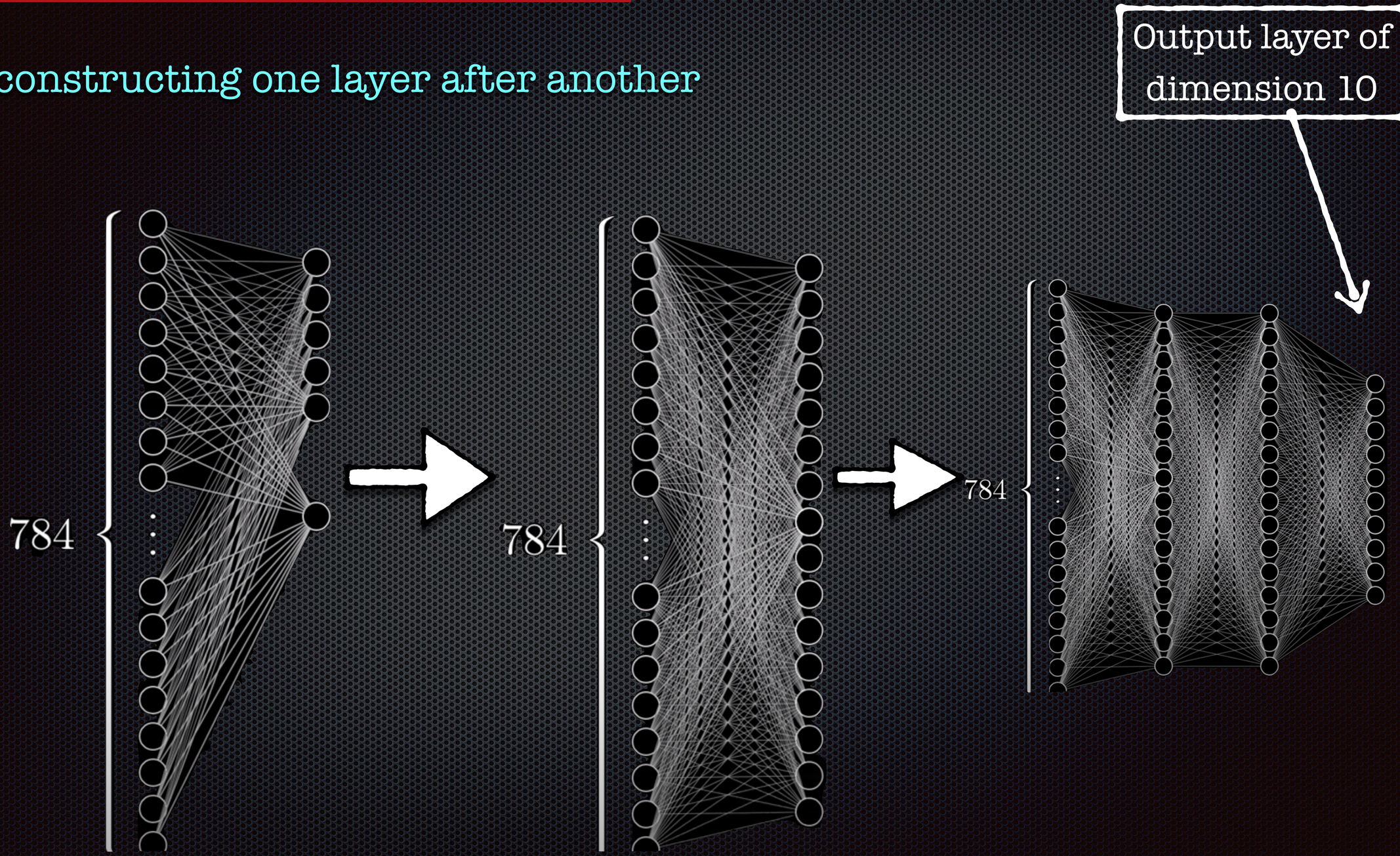
Example – MNIST data

For the first iteration only: Initialize the weights and bias randomly



Example – MNIST data

Keep constructing one layer after another

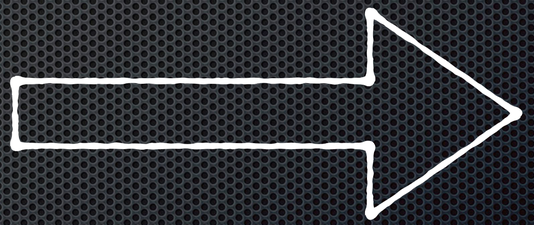


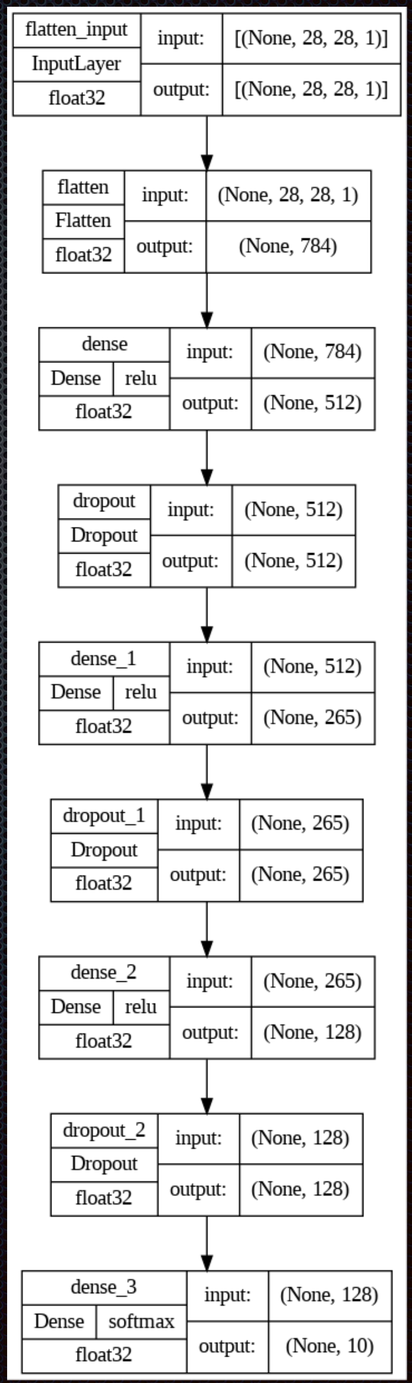
Example – MNIST data

Model summary

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 512)               401920

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 265)               135945

 dropout_1 (Dropout)         (None, 265)               0

 dense_2 (Dense)             (None, 128)               34048

 dropout_2 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 10)                1290

=================================================================
Total params: 573,203
Trainable params: 573,203
Non-trainable params: 0
_________________________________________________________________
```

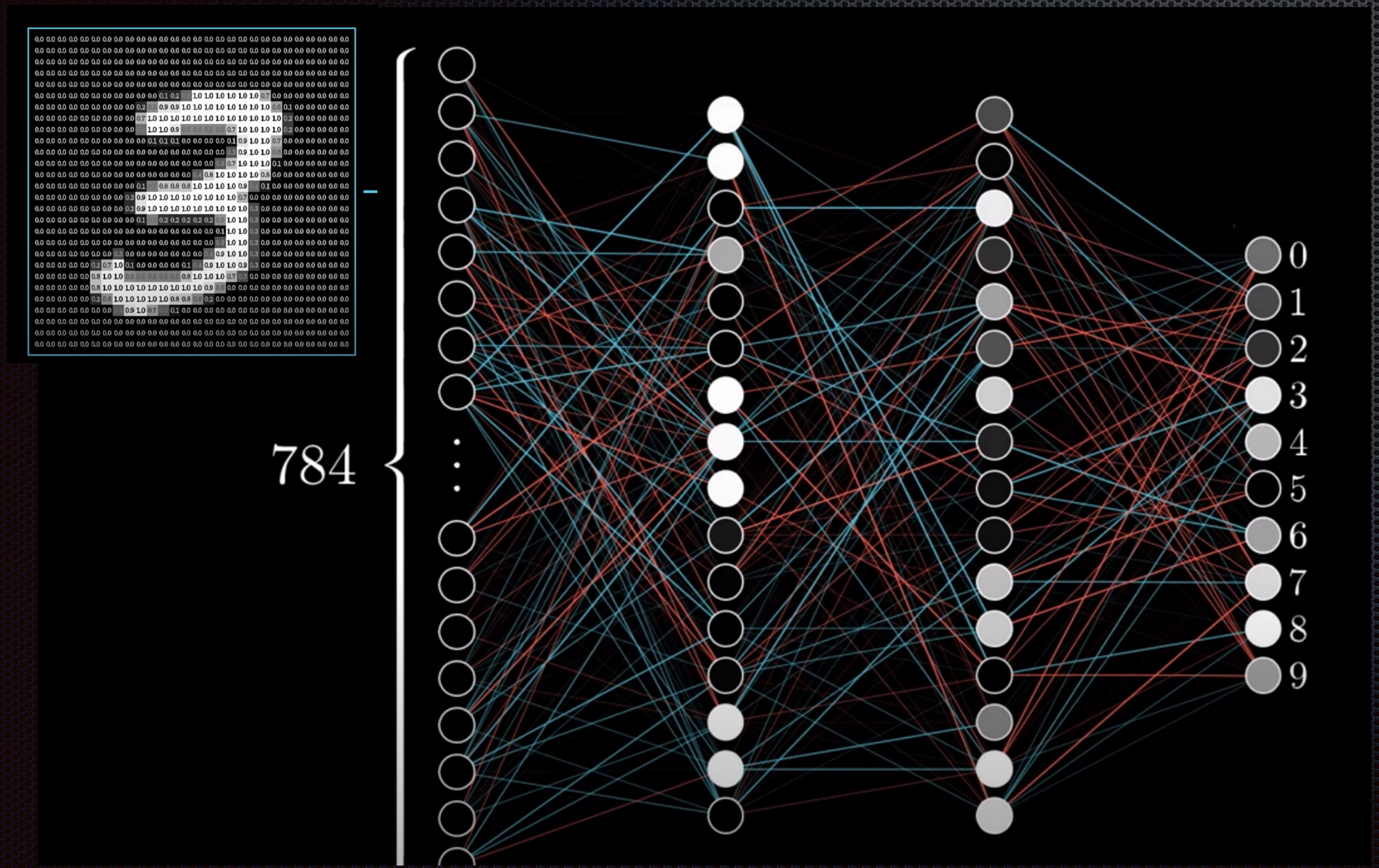
More detailed





Example – MNIST data

Forward propagation of the inputs



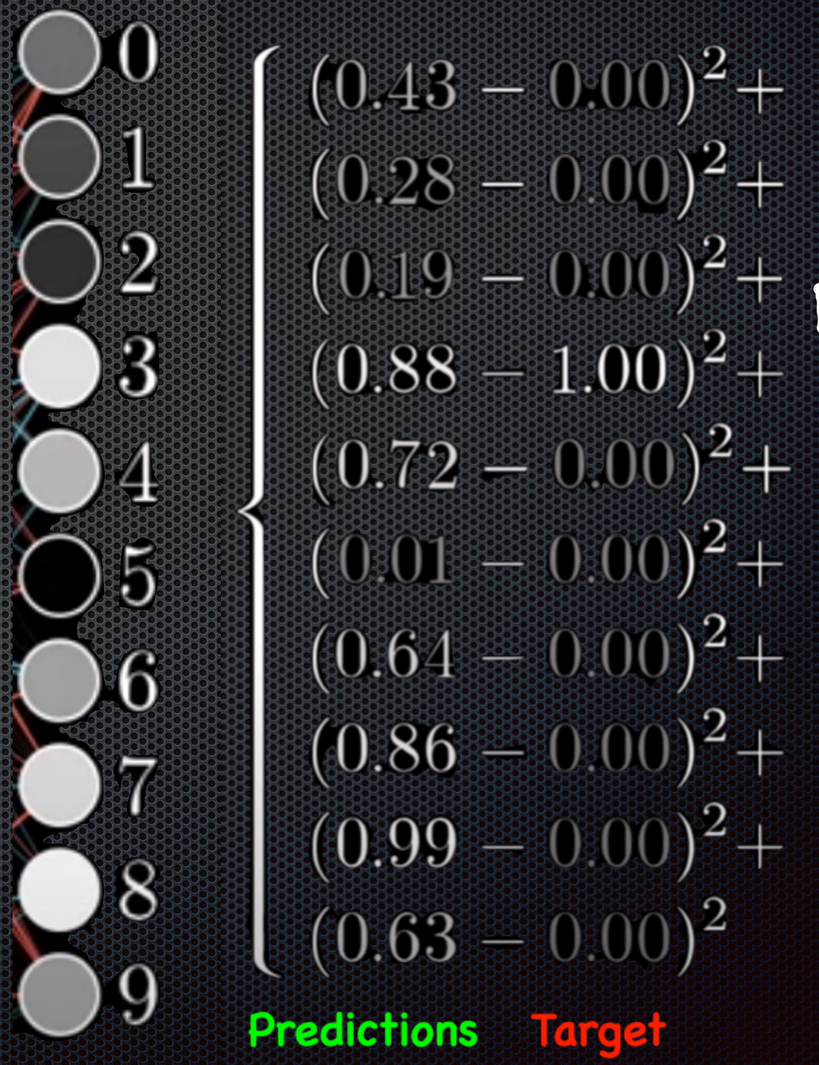
Model output

Prediction probability

$$P \in [0, 1]$$

Example – MNIST data

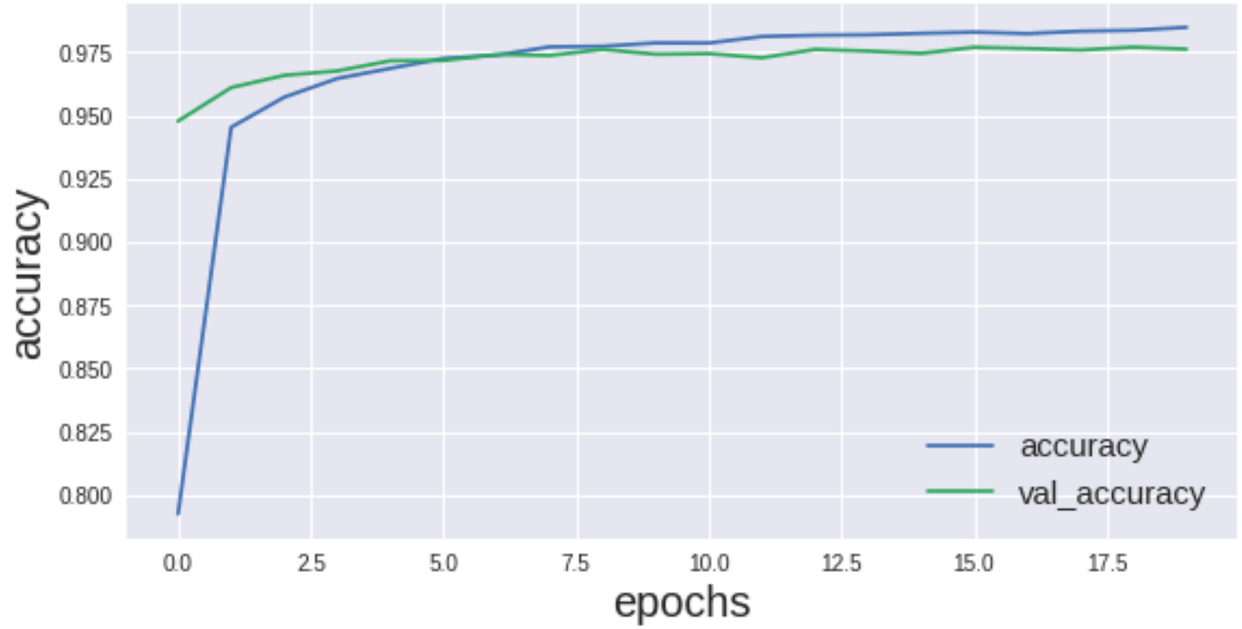
- Compute the loss function
- Back propagation to update the weights
- Repeat the iteration until to reach the desires accuracy

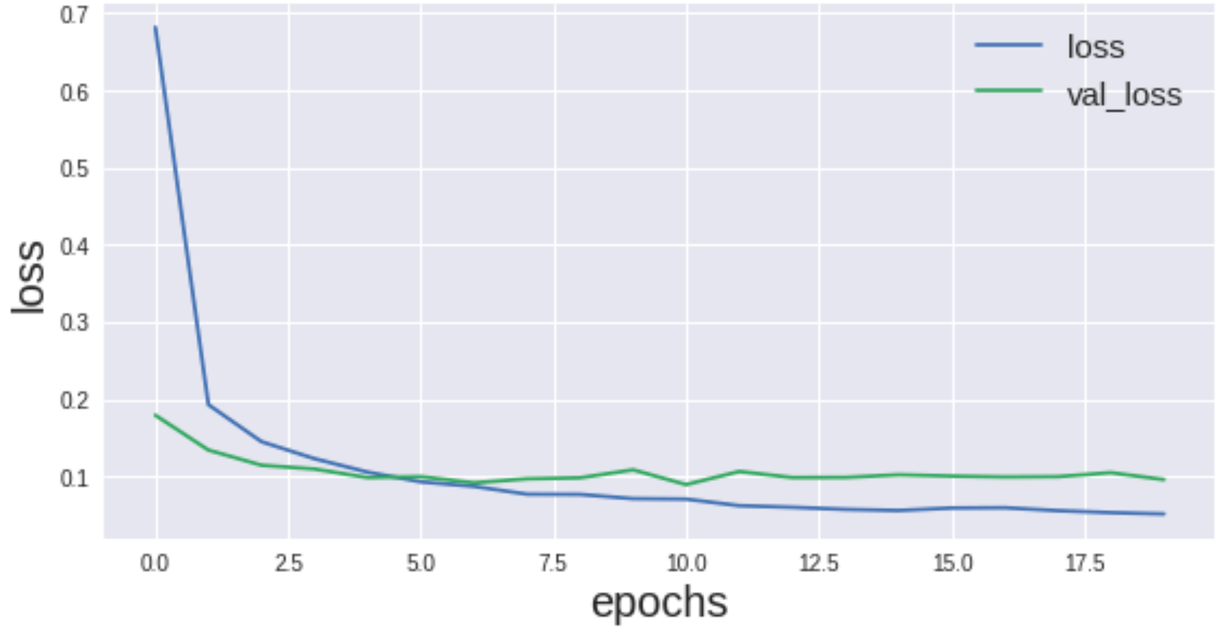


On-hot encoded Labels

Example – MNIST data

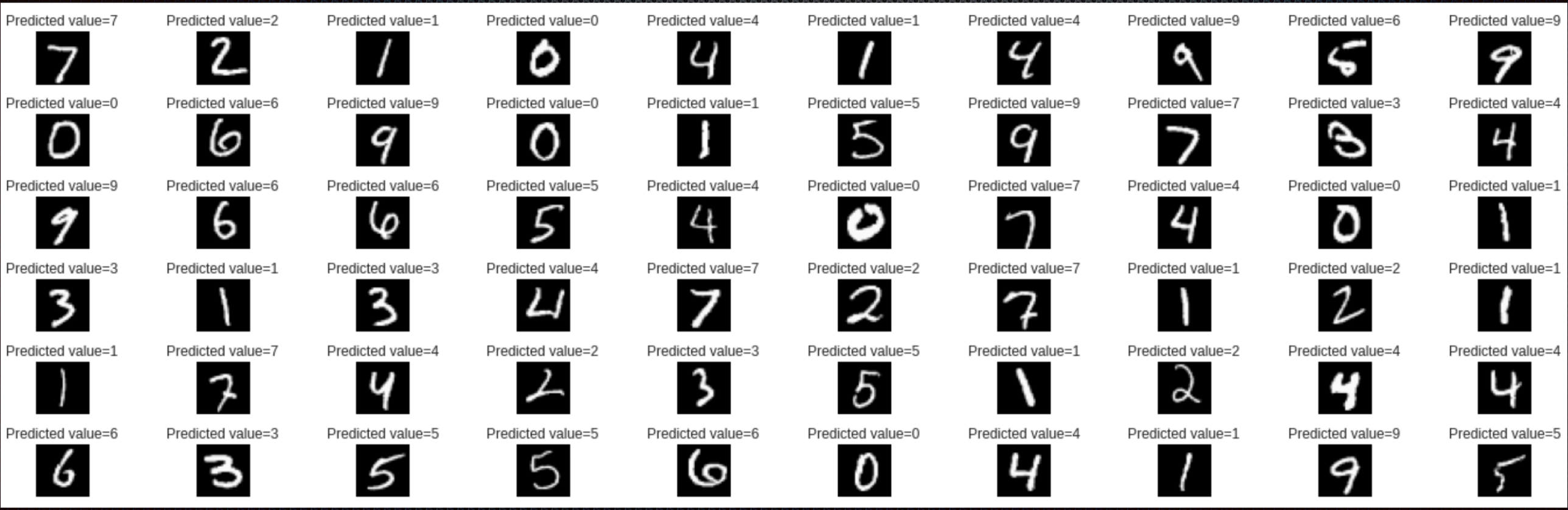
Test the model on the hold out validation sample





Example – MNIST data

Do some predictions



To be continued…