Ministério da Ciência, Tecnologia e Inovações

Centro Brasileiro de Pesquisas Físicas

Redes Neurais profundas e aplicações Deep Learning

Clécio Roque De Bom – debom@cbpf.br

clearnightsrthebest.com

EXEMPLO 3

RECONHECIMENTO DE CARACTERES MANUSCRITOS

REDE NEURAL CNN

CONVOLUTION NEURAL NETWORK

The simplest example I know

```
from keras.datasets import mnist
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from keras.models import Sequential

model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

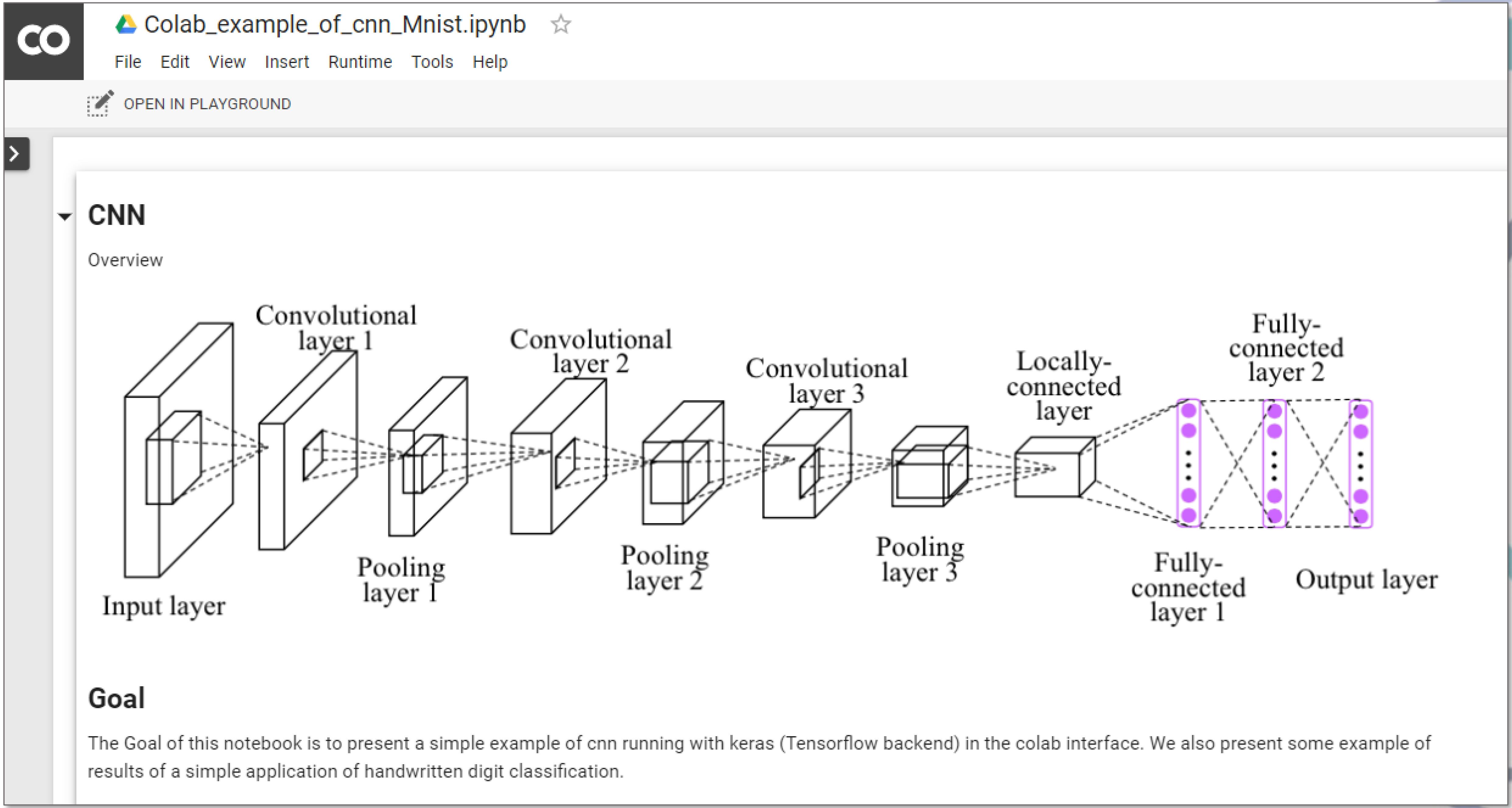
The simplest example I know

```
batch_size = 128
num_classes = 10
epochs = 10

# input image dimensions
img_x, img_y = 28, 28

# load the MNIST data set, which already splits into train and test sets
for us
(x_train, y_train), (x_test, y_test) = mnist.load_data()

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test),
          callbacks=[history])
score = model.evaluate(x_test, y_test, verbose=0)
```



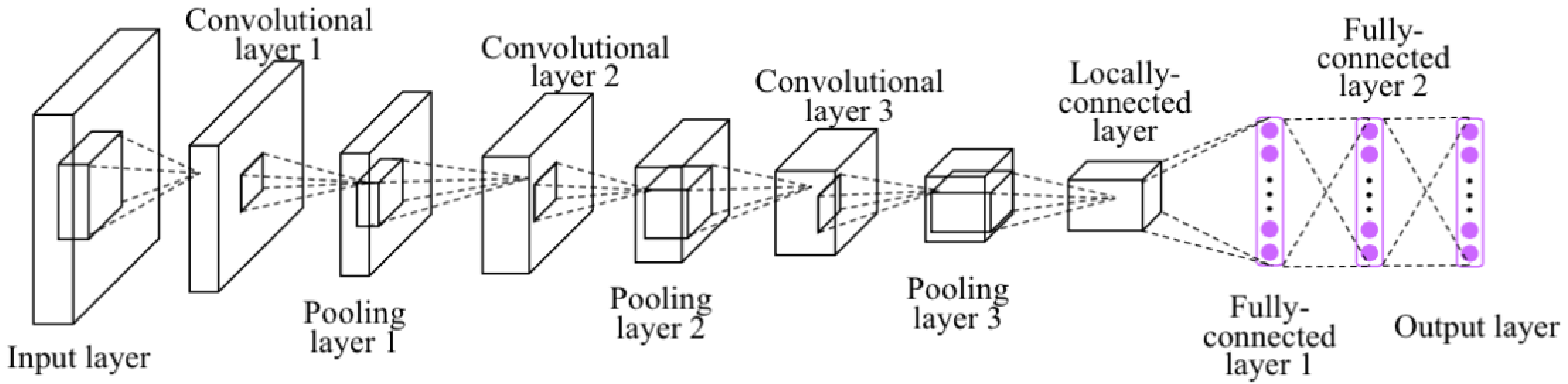
Colab_example_of_cnn_Mnist.ipynb

File Edit View Insert Runtime Tools Help

OPEN IN PLAYGROUND

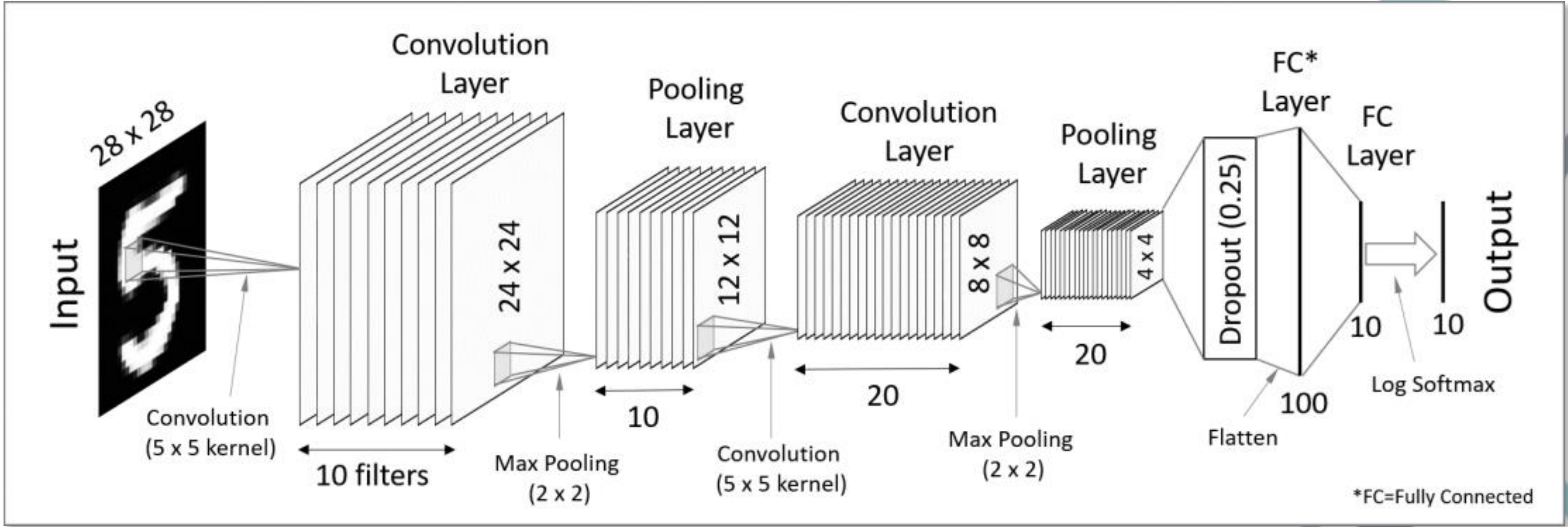
CNN

Overview

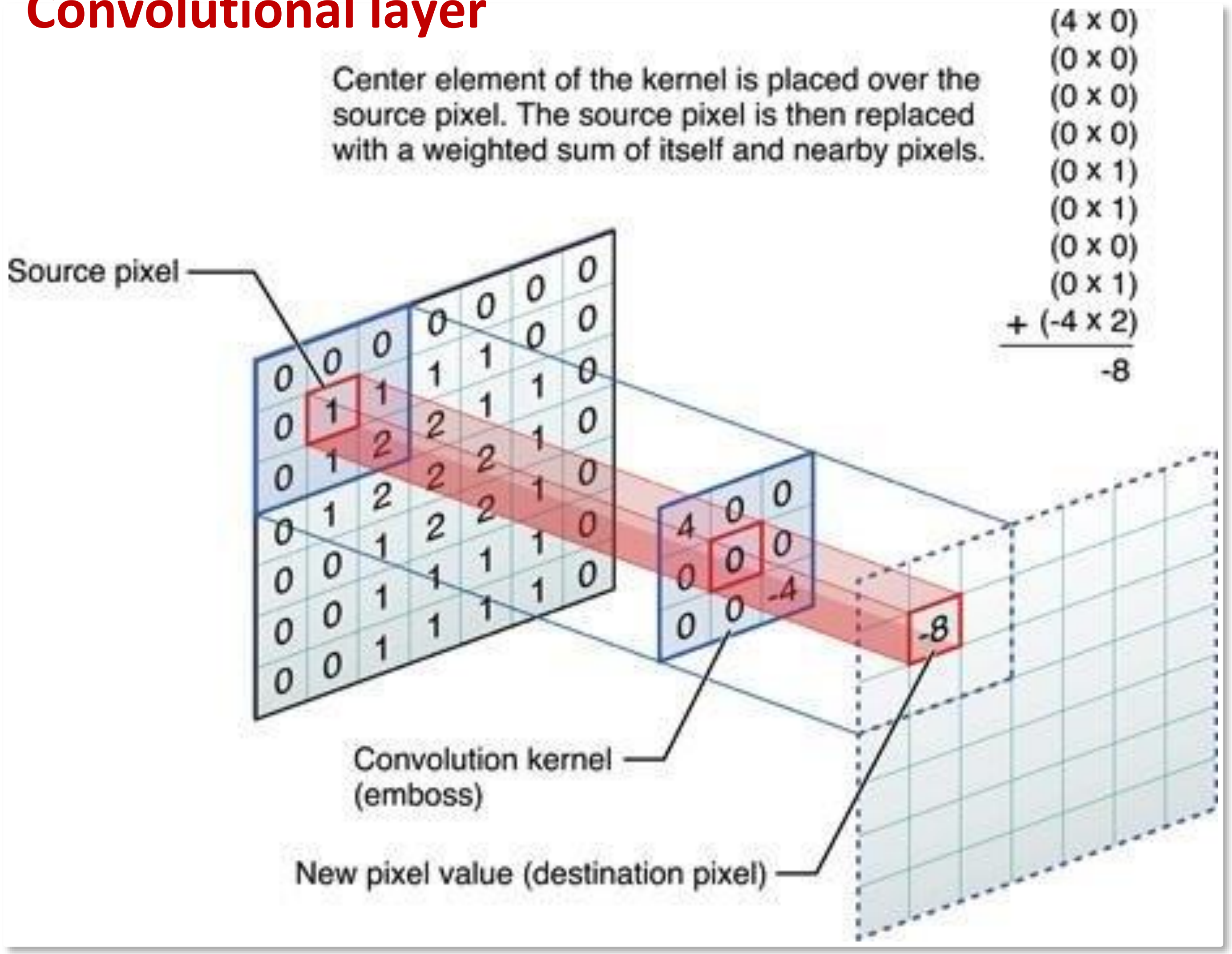


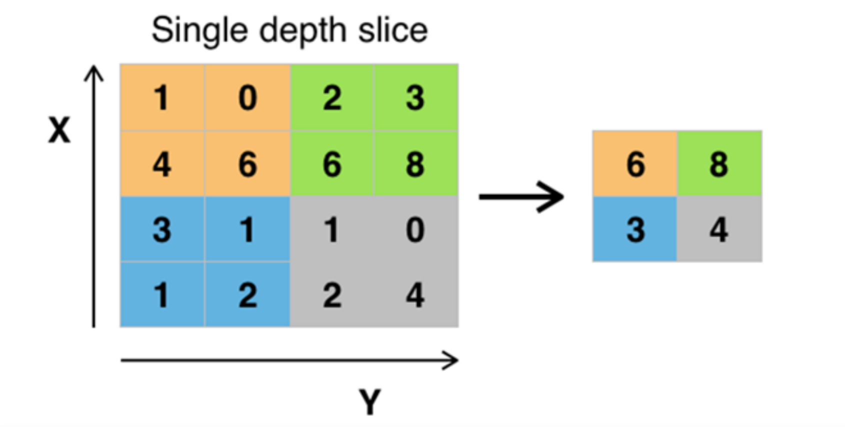
Goal

The Goal of this notebook is to present a simple example of cnn running with keras (Tensorflow backend) in the colab interface. We also present some example of results of a simple application of handwritten digit classification.


28 x 28
Input
Convolution (5 x 5 kernel)
Convolution Layer
24 x 24
10 filters
Max Pooling (2 x 2)
Pooling Layer
12 x 12
10
Convolution (5 x 5 kernel)
Convolution Layer
8 x 8
20
Max Pooling (2 x 2)
Pooling Layer
4 x 4
20
Dropout (0.25)
Flatten
FC* Layer
100
FC Layer
10
Log Softmax
10
Output
*FC=Fully Connected

Convolutional layer


Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)



Example of MaxPool with 2x2 and a stride of 2

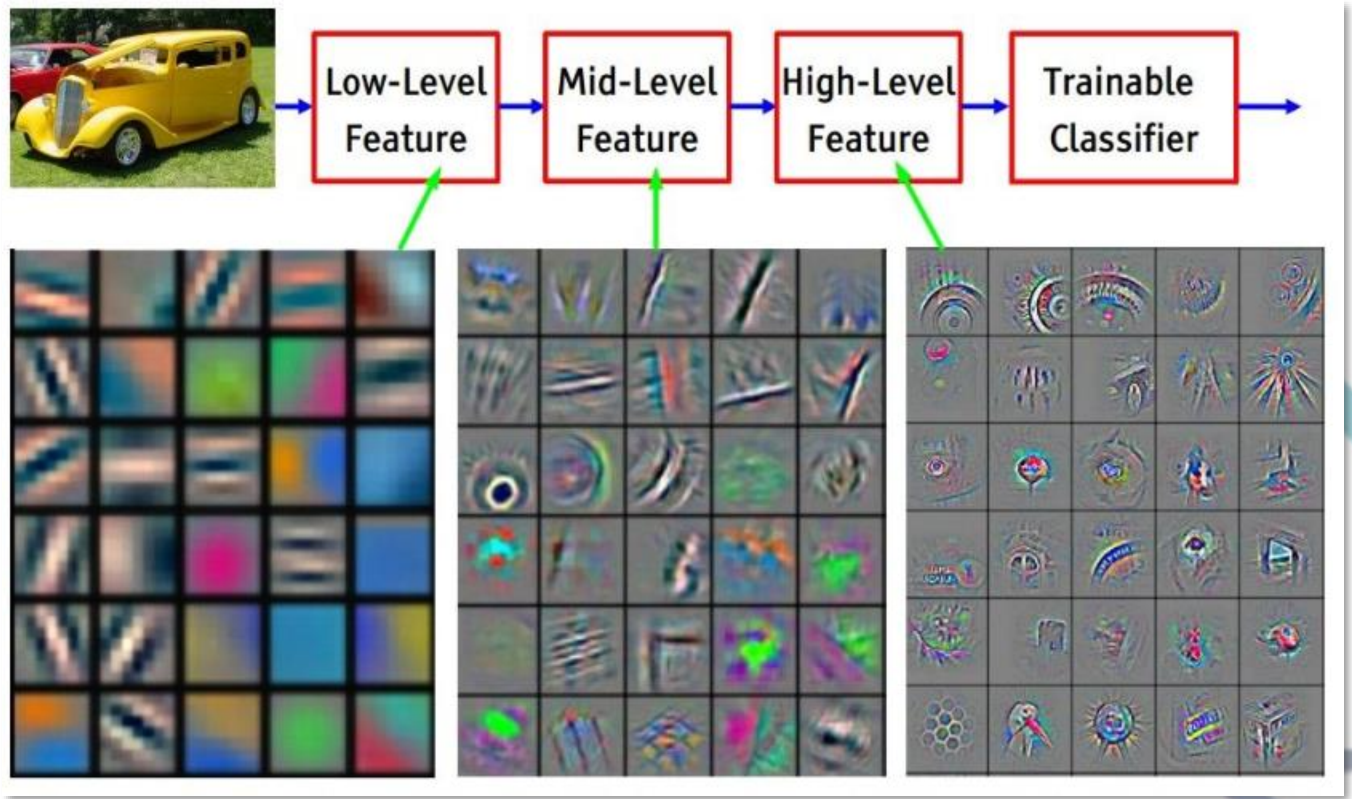
Convolutional Neural Networks

What makes CNNs so special?

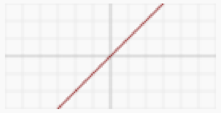
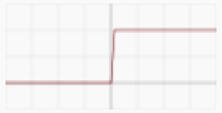
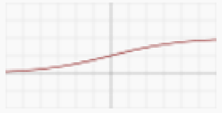
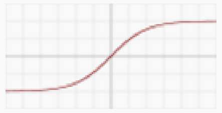
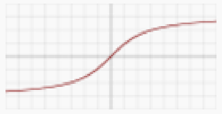
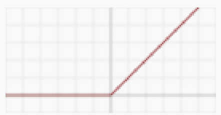
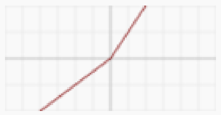
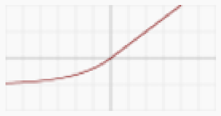
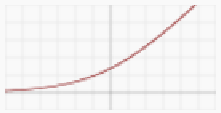
- Based on mammal visual cortex
- Extract **surrounding-depending** high-order features.
- Specially useful for:
  - Images
  - Time-dependent parameters

    Speech recognition

    Signal analysis



Activation Functions

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Do not take advantage of Neural Nets for non linearities estimation. Useful in regression problems since is unbounded.

Hard to train, derivative vanishes.

Easily differentiable. In the last layers can be associated with probability.

Similar results as in sigmoid activations in intermediate layers. However, is numerically faster.

Tentative to avoid the vanishing of the ReLu derivative.

Adapted from
https://towardsdatascience.com/
activation-functions-neural-networks-1cbd9f8d91d6

Why Sigmoid for classification?

Consider two classes, $y \in \{0, 1\}$.

The conditional probability of class $P(y|z(x))$ where $z = \omega^T h(x) + b$ the output of a set of neurons with x inputs.

Why Sigmoid for classification?

the unnormalized log probability can be written as

$$log\ \widehat{P}(y=1|z)=z \quad \text{(neurons “on”)}$$
$$log\ \widehat{P}(y=0|z)=0 \quad \text{(neurons “off”)}$$

$$\widehat{P}(y=1|z)=exp(z)$$

$$\widehat{P}(y=0|z)=exp(0)=1$$

Why Sigmoid for classification?

The Normalized version:

$$P(y=1|z) = \frac{exp(z)}{1+exp(z)}$$

$$P(y=0|z) = \frac{1}{1+exp(z)} \quad .$$

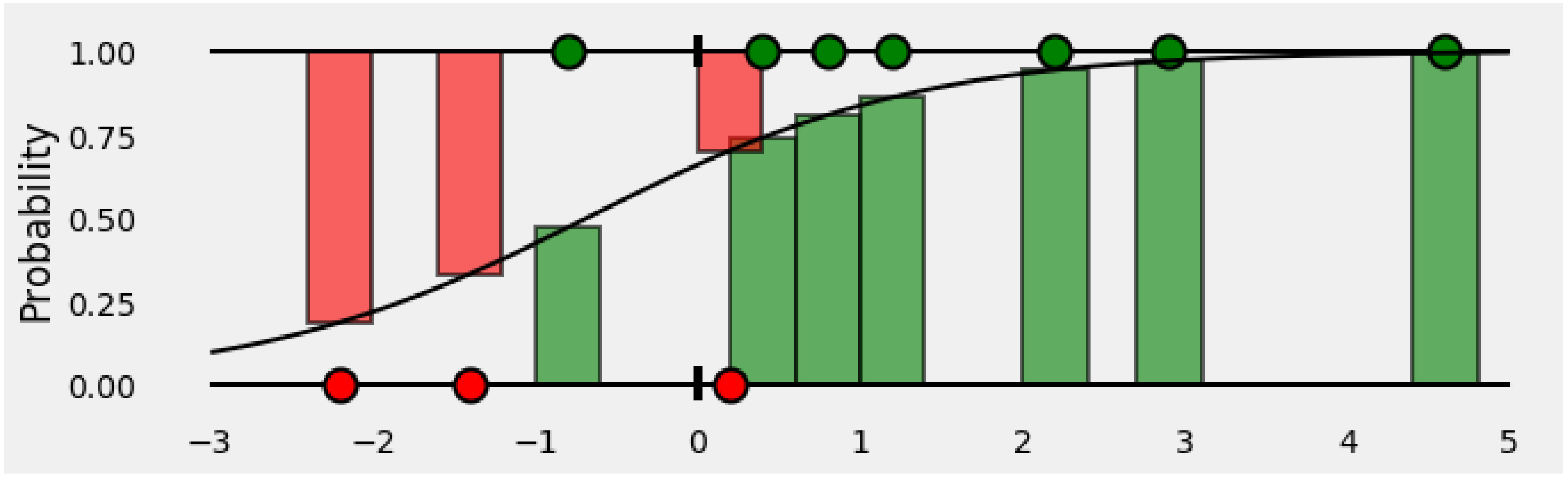
This is

$$P(y=1|z) = \frac{exp(z)}{1+exp(z)} = \frac{1}{\frac{exp(z)+1}{exp(z)}} = \frac{1}{1+exp(-z)} = \sigma(z)$$

$$P(y=0|z) = \sigma(-z) .$$

Some Loss intuition...

Consider the binary classification of red and greens. The True class probability of a set of z points is:

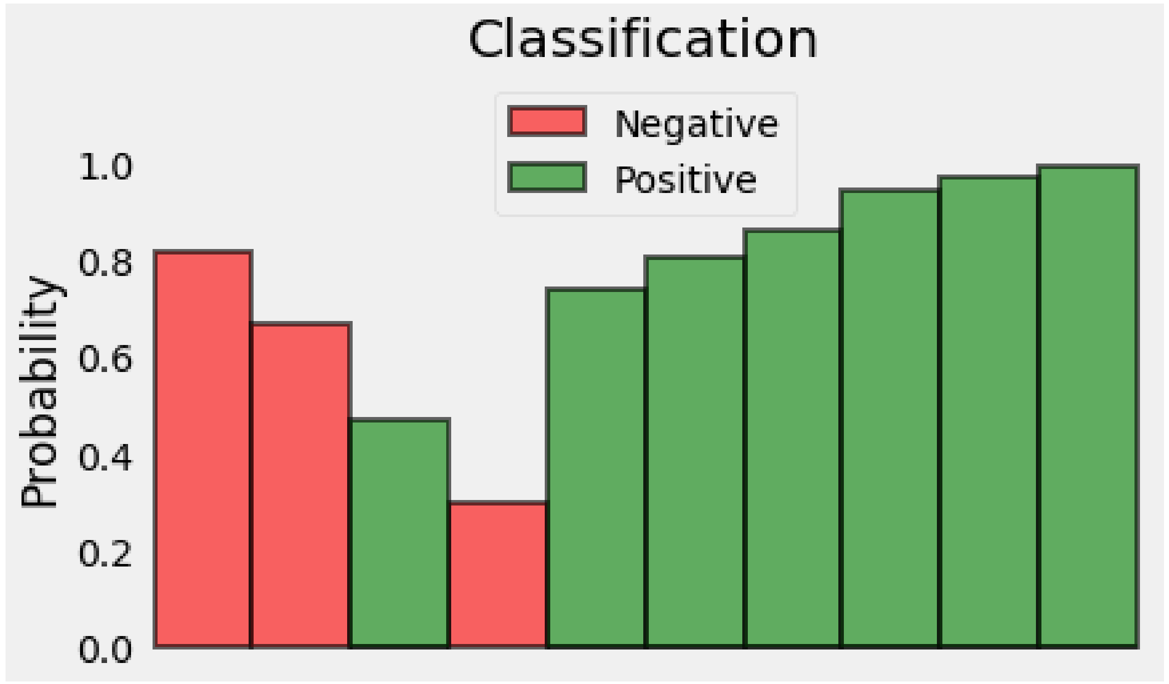


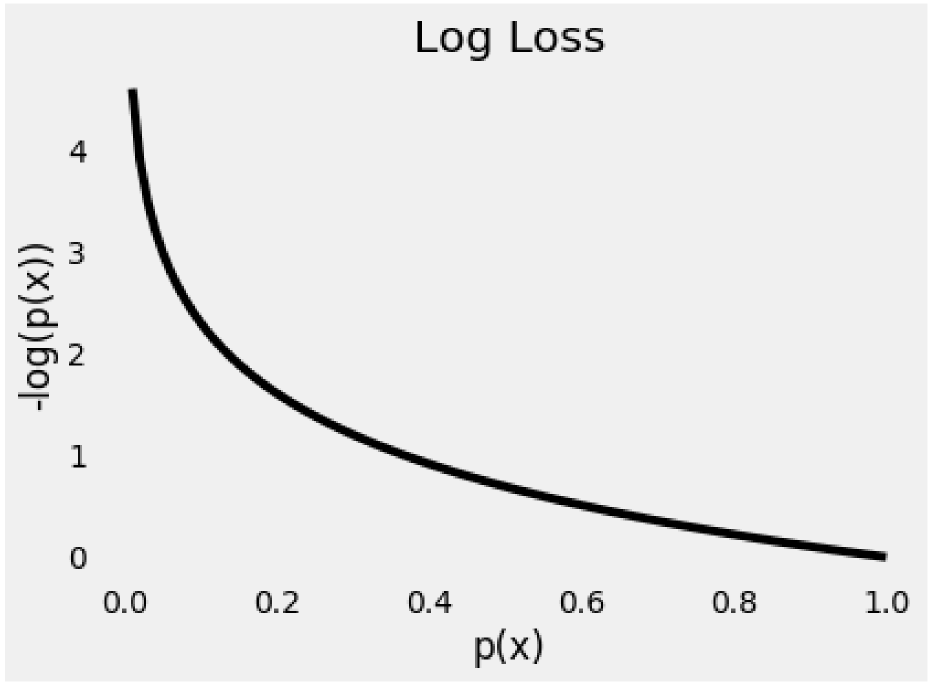
$$P(y = 0|z) = \sigma(-z)$$

$$P(y = 1|z) = \sigma(z)$$

Example adapted from : https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a

Some Loss intuition...



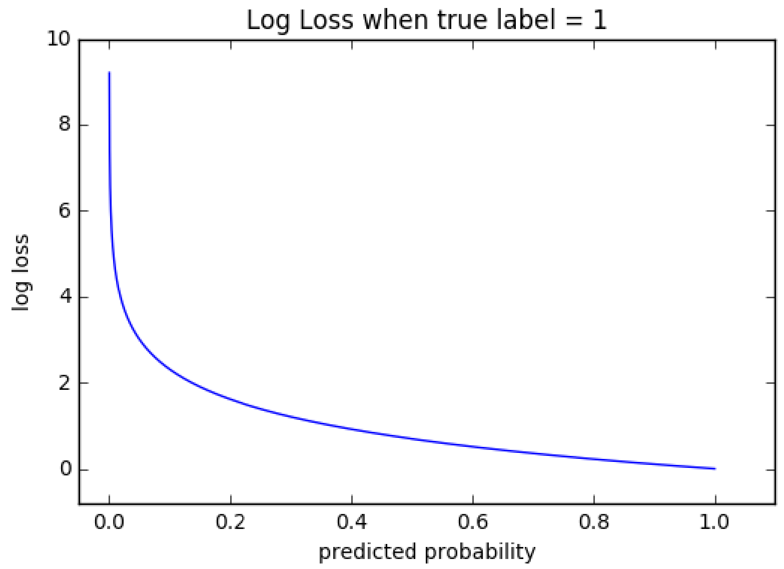


If the predicted probability of the **true class** gets **closer to zero**, the **-log(p(x)) increases exponentially**.

So... Cross entropy Loss

Consider two classes: 1 and 0s. The predicted probability is given by

$$q_{y=1} = \hat{y} \equiv g(\mathbf{w} \cdot \mathbf{x}) = 1/(1 + e^{-\mathbf{w} \cdot \mathbf{x}}),$$



$$H(p,q) = -\sum_i p_i \log q_i = -y \log \hat{y} - (1-y)\log(1-\hat{y})$$

$$J(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)\right],$$

How to Choose?

The Mean Squared Error loss is the default loss to use for regression problems.

It represents loss function under the inference framework of maximum likelihood.

This assumes the distribution of the target variable is Gaussian.

Change it Carefully.

$$P(x;\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$\ln(P(x;\mu,\sigma))=\ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right)-\frac{(x-\mu)^2}{2\sigma^2}$$

How to Choose?

The Mean Squared Error loss is the default loss to use for regression problems.

It represents loss function under the inference framework of maximum likelihood.

This assumes the distribution of the target variable is Gaussian.

Change it Carefully.

$$P(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$\ln(P(x;\mu,\sigma)) = \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(x-\mu)^2}{2\sigma^2}$$

Root Mean Squared Log Error

Regression problems in which the target value has a spread of values
When predicting a large value, you may not want to punish a model as heavily as mean squared error, that is your values are small.

Root Mean Squared Error (RMSE)

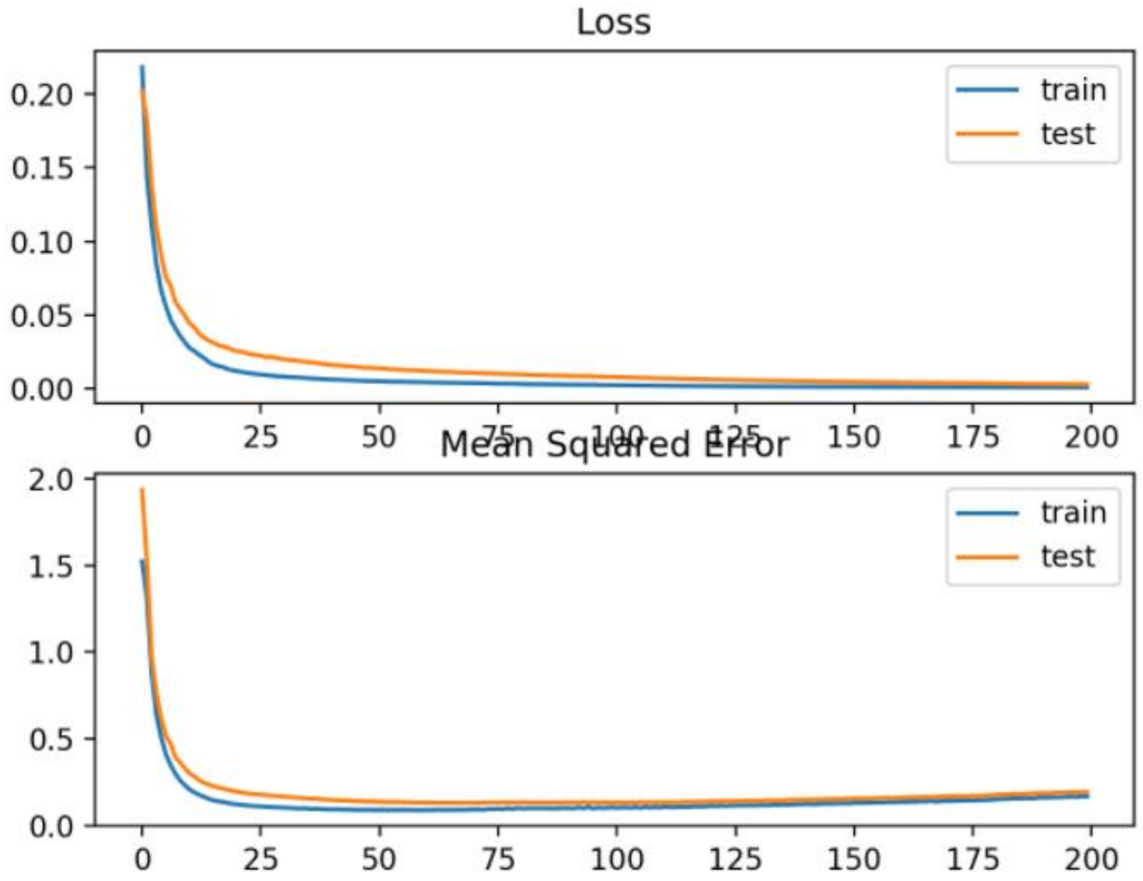
$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Root Mean Squared Log Error (RMSLE)

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

prediction

actual

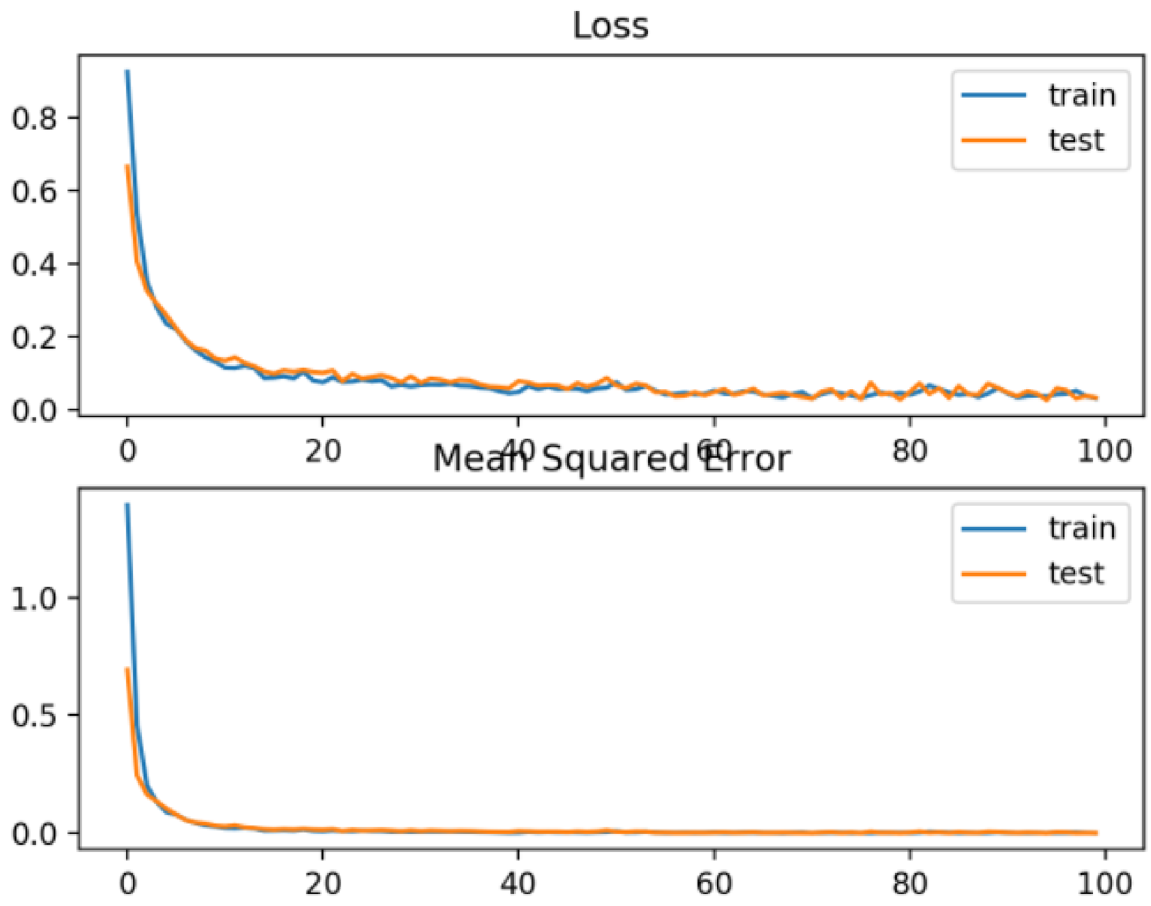


Root Absolute Squared Error

The Mean Absolute Error loss is an appropriate loss function in this case as it is more robust to outliers.

In case outliers matters!

$$\text{MAE}= \sum_{i=1}^{n}(Y_i - \hat{Y_i})^2.$$



What Metrics is for ?

Common Classification Metrics

Binary Accuracy: binary_accuracy, acc

Categorical Accuracy: categorical_accuracy,

Common Regression Metrics

Mean Squared Error: mean_squared_error, MSE or mse
Mean Absolute Error: mean_absolute_error, MAE, mae

Ministério da Ciência, Tecnologia e Inovações

Centro Brasileiro de Pesquisas Físicas

Redes Neurais profundas e aplicações Deep Learning

Clécio Roque De Bom – debom@cbpf.br

clearnightsrthebest.com