LABSIN

An Introduction to Machine Learning and Deep Learning with R

HARPO (AKA Carlos A. Catania Ph.D)
LABSIN - Ingeniería - UNCuyo

@harpolabs
harpo@ingenieria.uncuyo.edu.ar

AGENDA Day 2: Deep Learning

- Basic concepts
- Neural networks
- Convolutional Neural Networks
- Recurrent Neural networks (??)

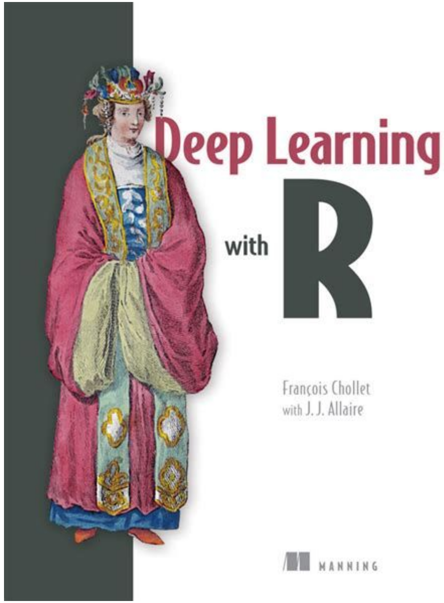
LABSIN

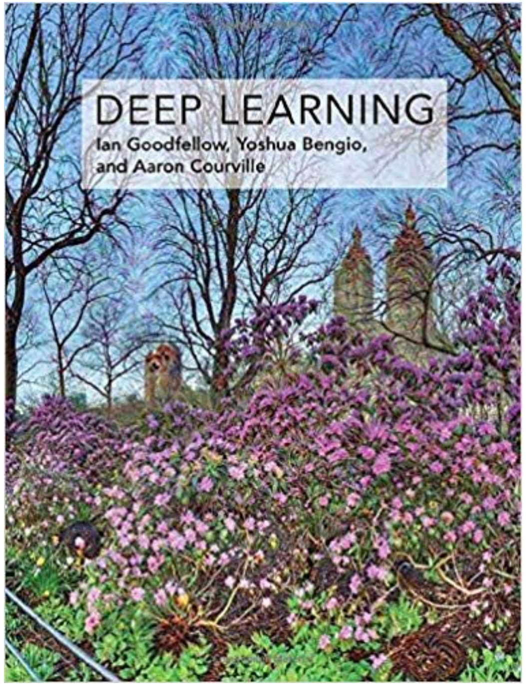
Materials Day 2: Deep Learning

- **Rstudio** desktop or server version >=1.1
- R version >= 3.5
- **Rstudio** Notebook operational
- `install.packages(“keras”)`
- `install_keras()`
- `Data for day 2 at` http://bit.ly/mllab2019

LABSIN

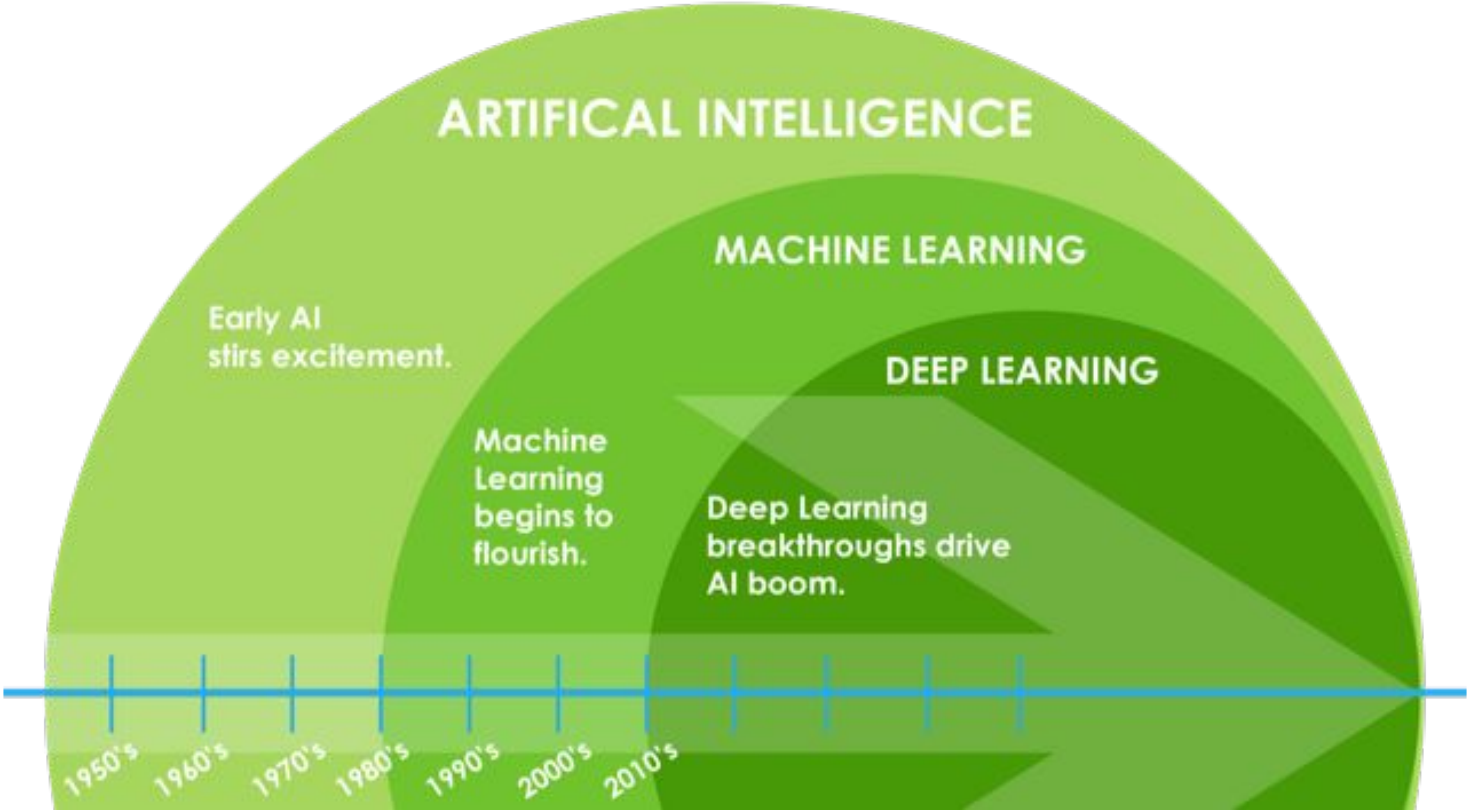
Highly recommended books





Deep Learning

Deep Learning (in 10 min??)

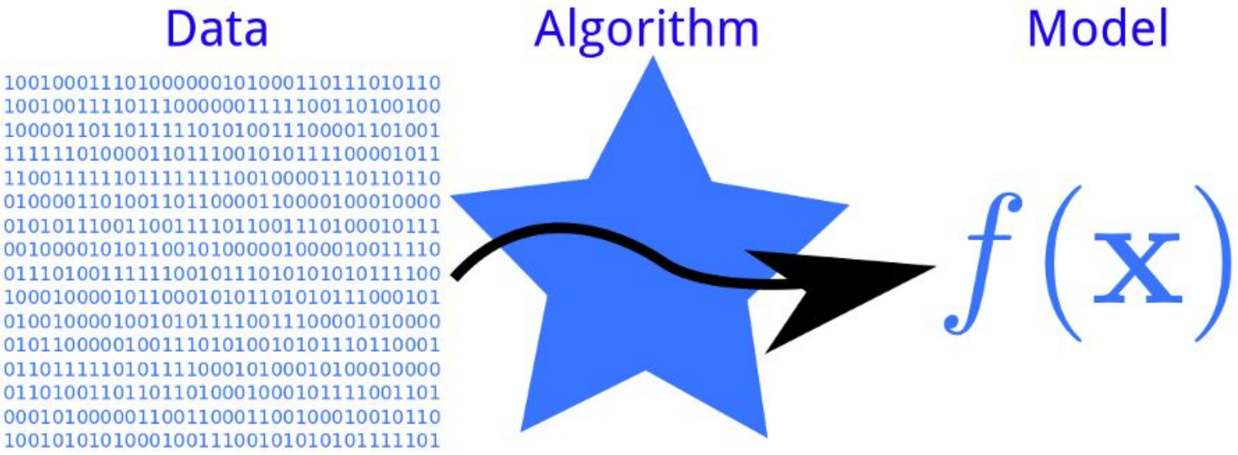

ARTIFICAL INTELLIGENCE
MACHINE LEARNING
DEEP LEARNING
Early AI
stirs excitement.
Machine
Learning
begins to
flourish.
Deep Learning
breakthroughs drive
AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

DEEP LEARNING

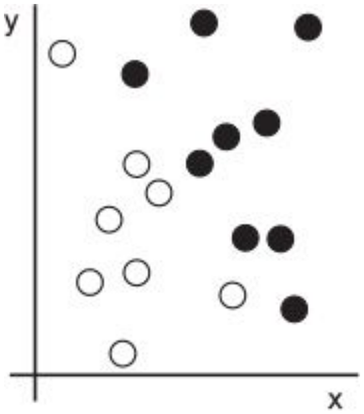
What is LEARNING?

What is DEEP?

The general idea behind ML



A simple example

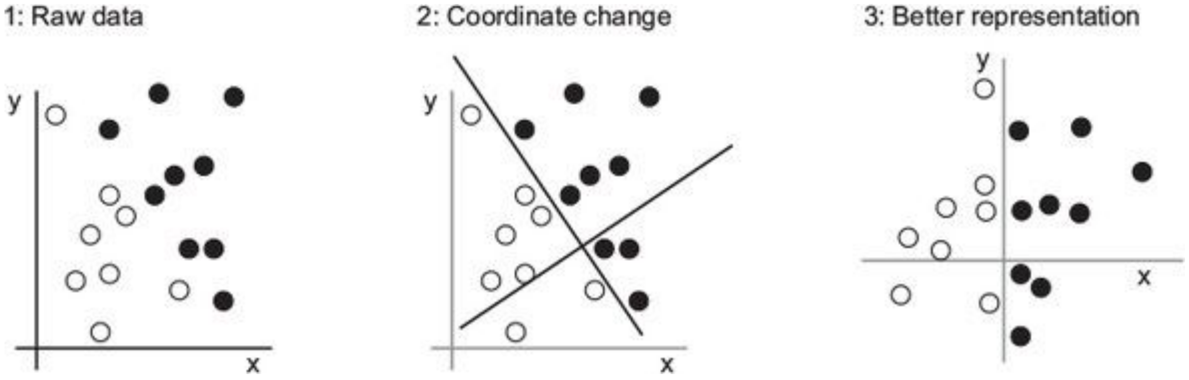


Goal: predict data point color based on its coordinate (x,y)

Method: We need a new representation of data that cleanly separates the white points from the black points.

Simple approach: Coordinates Change

The problem can be expressed as a simple rule: "**Black points** are such that **x > 0**," or "**White points** are such that **x < 0**."



LABSIN

LEARNING

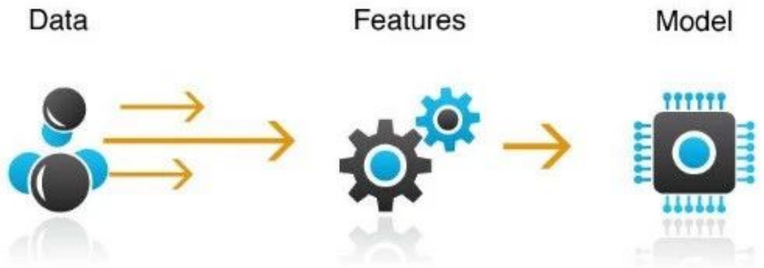
- In the context of **machine learning**, describes an automatic search process for better representations.

- In a way, all machine-learning algorithms consist of **automatically** finding such **transformations** that turn data into more-useful representations for a given task.

However...

Sometimes we need to **feed** the algorithm with some particular presentation aiming at helping the final performance.

Such process is usually call **feature engineering** and could be **extremely time consuming**



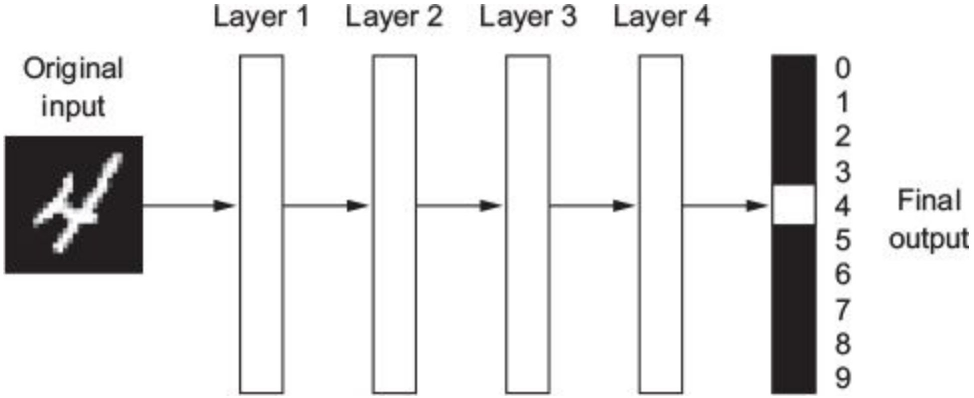
LABSIN

The “deep” in Deep Learning

Deep learning is a specific subfield of machine learning: a new take on learning representations from data that puts an emphasis on learning successive *layers* of increasingly meaningful representations.

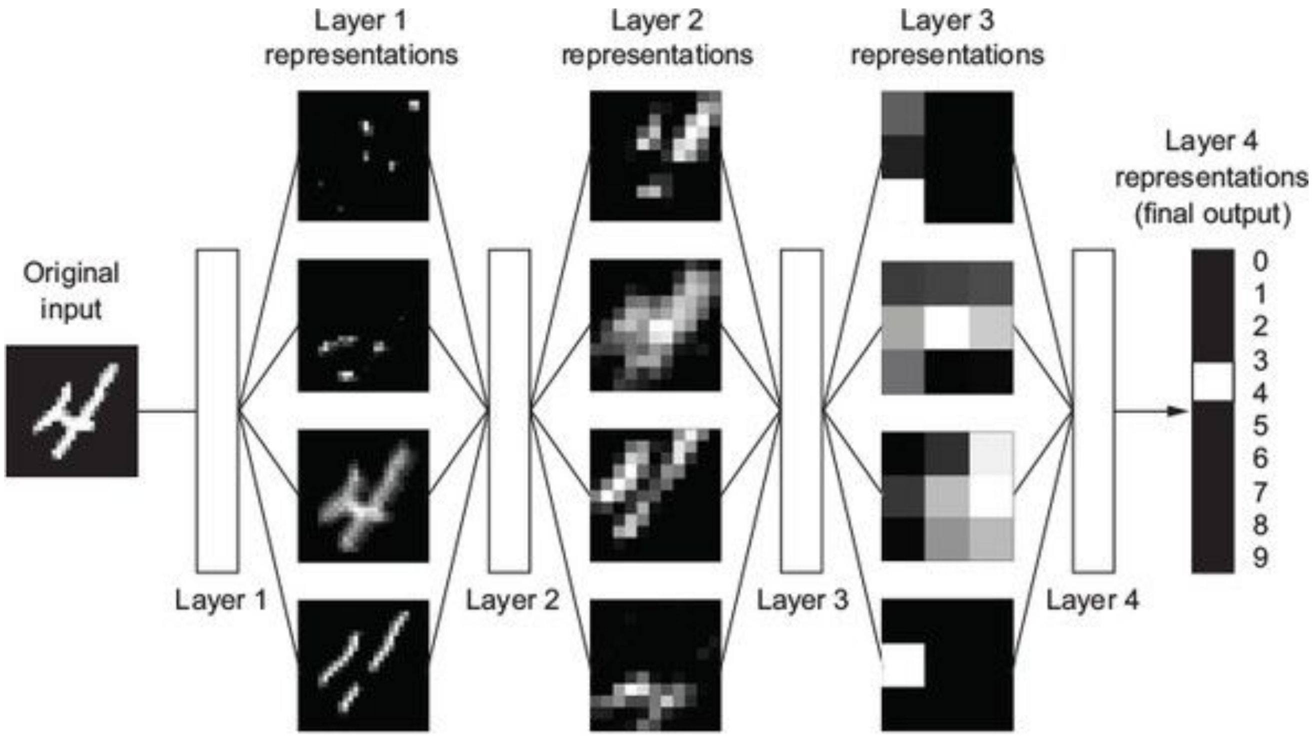
Neural networks

In deep learning, these layered representations are (almost always) learned via models called *neural networks*



A deep neural network for digit classification

Deep learning for digit classification

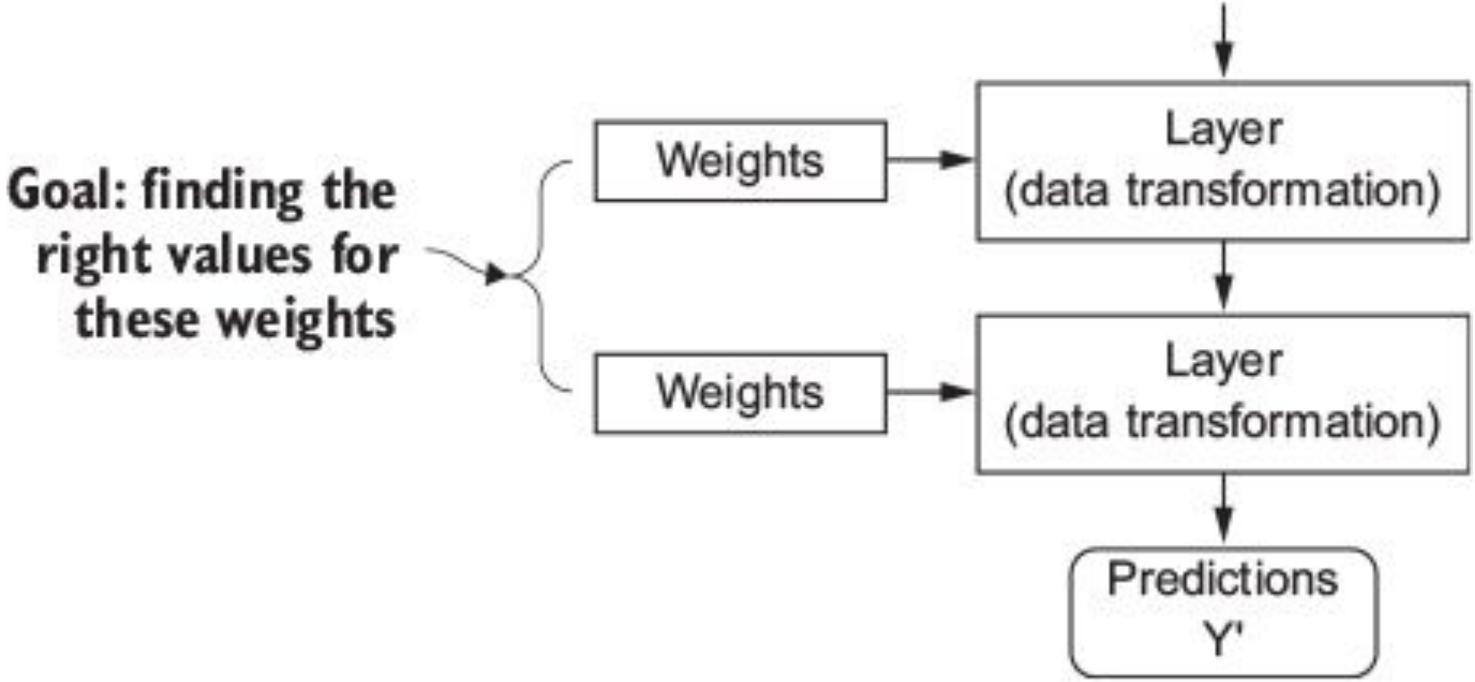


LABSIN

Understanding how deep learning works, in three figures

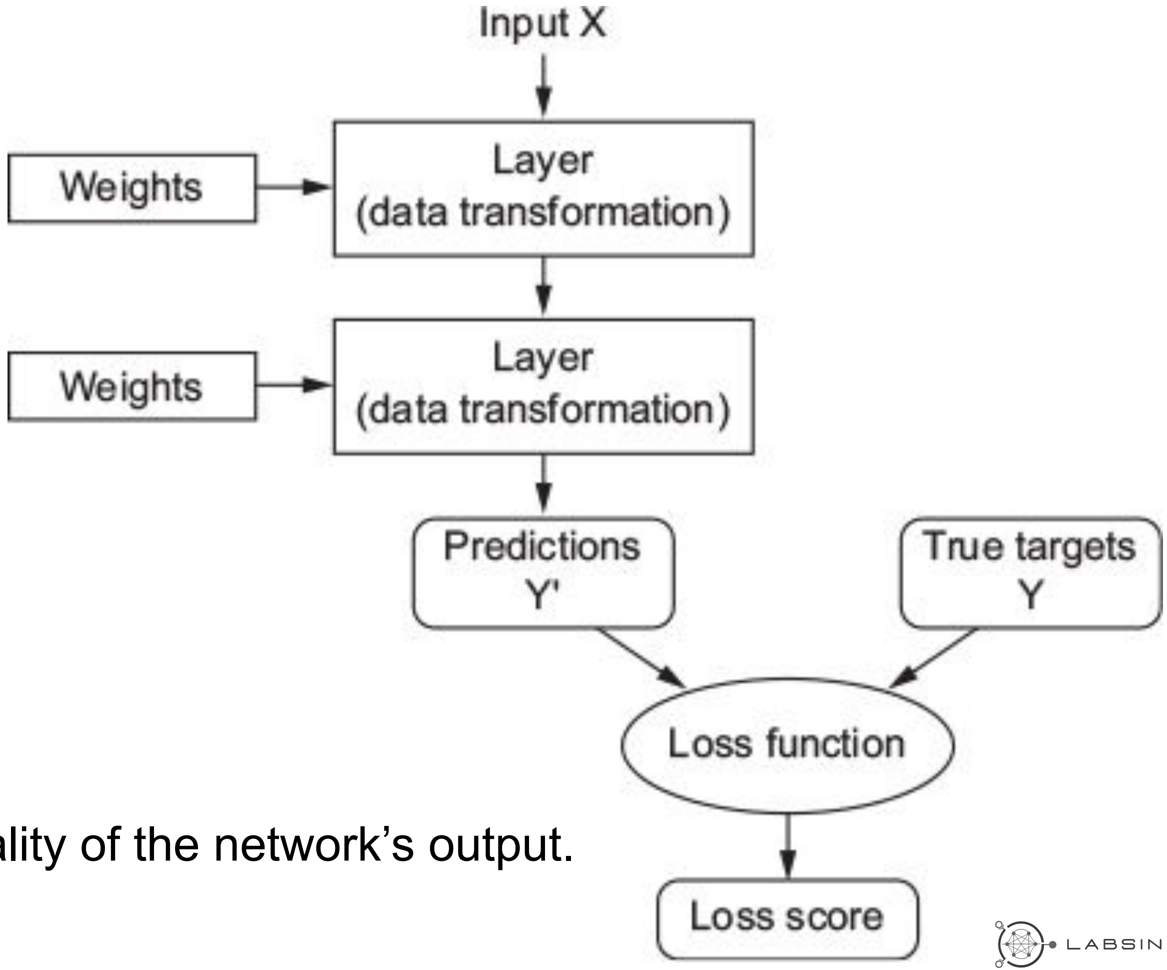
Figure 1:

A neural network is parameterized by its weights.



LABSIN

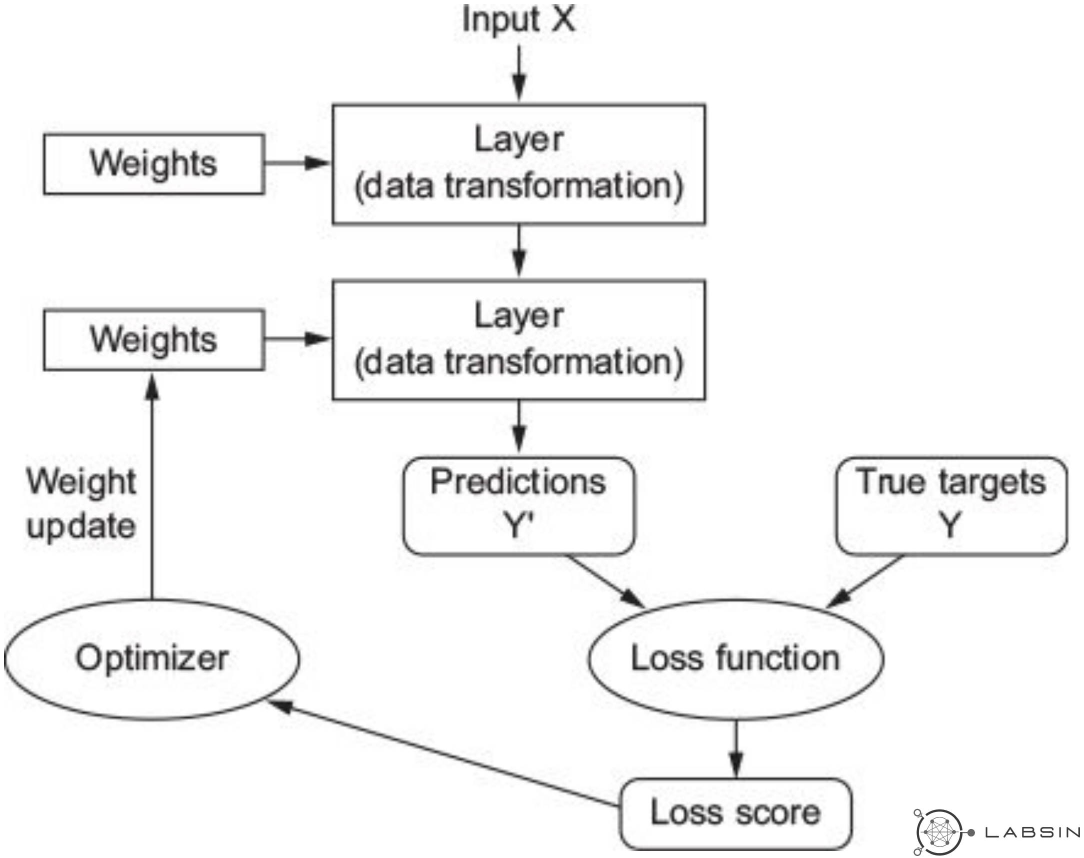
Figure 2:



A loss function measures the quality of the network's output.

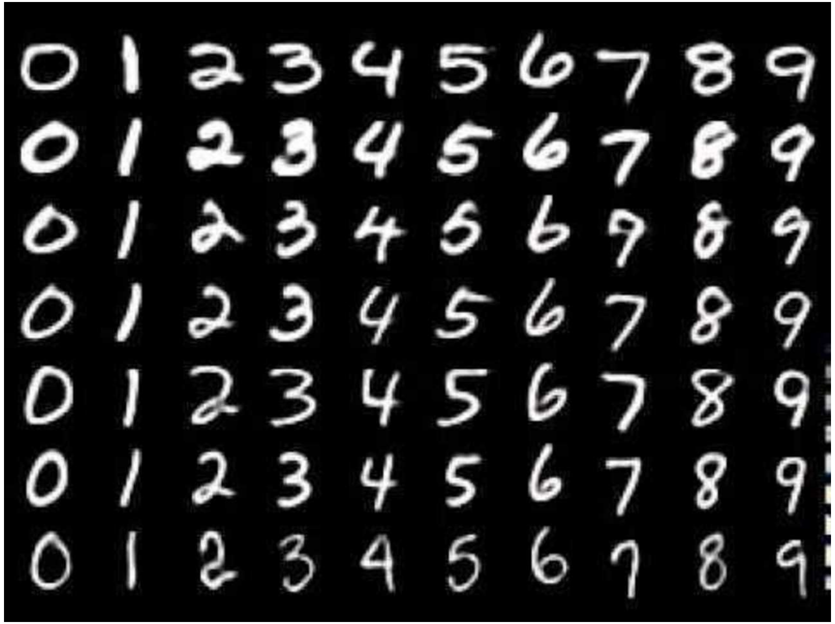
Figure 3:

The loss score is used as a feedback signal to adjust the weights.



LAB 3: The MNIST dataset

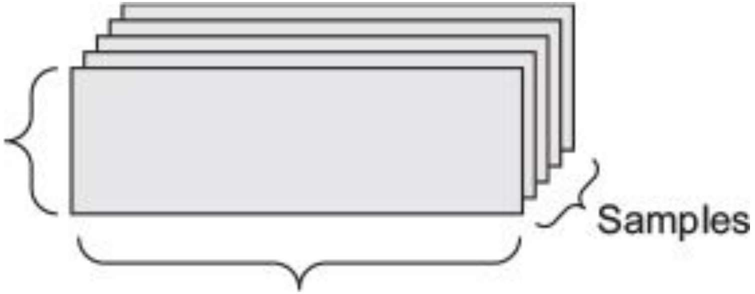
The problem we're trying to solve here is to classify grayscale images of handwritten digits (28 pixels by 28 pixels) into their 10 categories (0 to 9).



Loading the Mnist Dataset

````
```{r, results='hide'}
library(keras)

mnist <- dataset_mnist()
train_images <- mnist$train$x
train_labels <- mnist$train$y
test_images <- mnist$test$x
test_labels <- mnist$test$y
```
````




````
```{r}
str(train_images)
```
````

```
int [1:60000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0
```

The images are encoded as as 3D arrays, and the labels are a 1D array of digits, ranging from 0 to 9.

Let’s build our first model

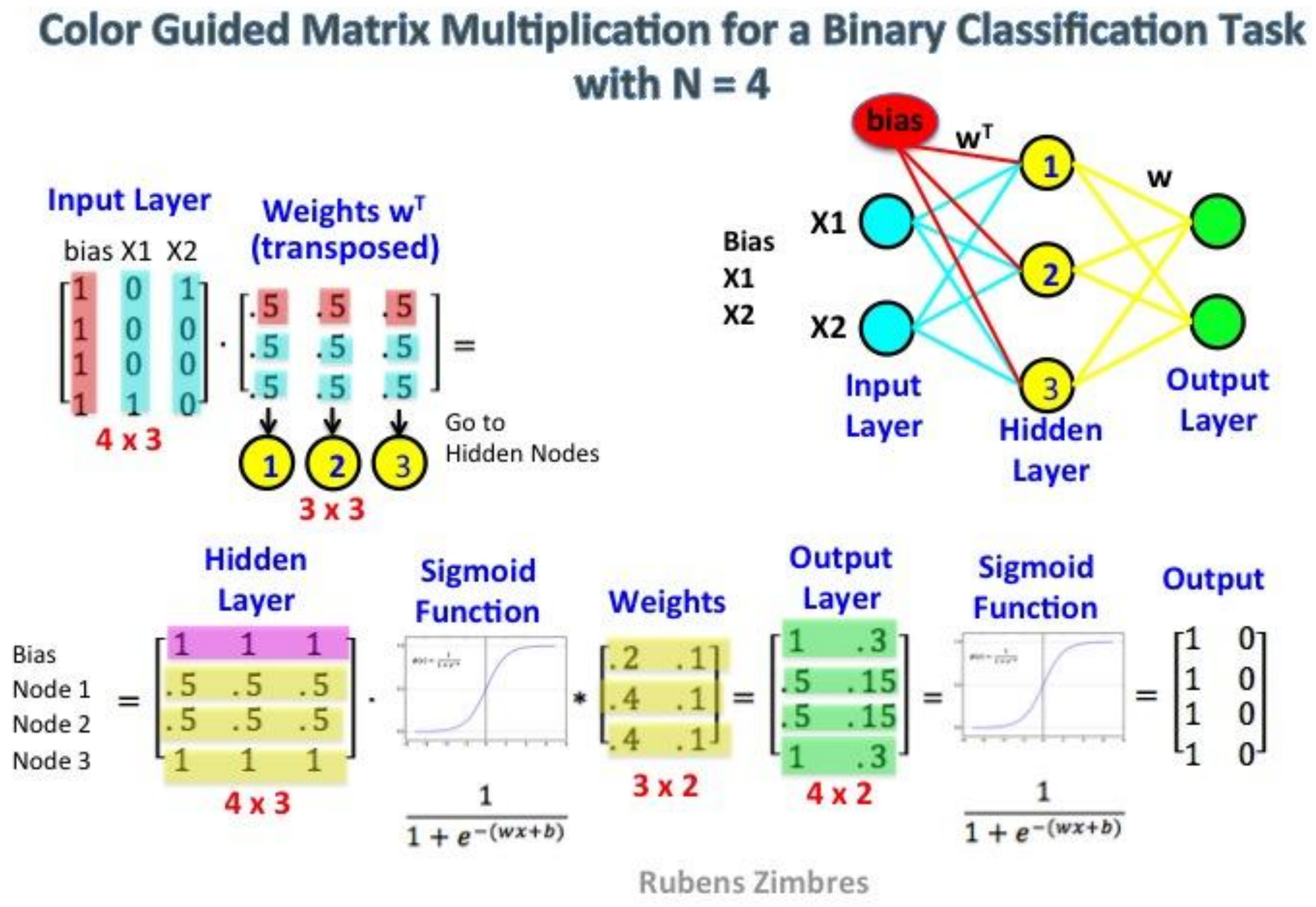
```
```{r}
network <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 10, activation = "softmax")
```
```

- The core building block of neural networks is the **_layer_**, a data-processing module that you can think of as a filter for data. Some data comes in, and it comes out in a more useful form.

- Specifically, layers extract **_representations_** out of the data fed into them—hopefully representations that are more meaningful for the problem at hand.

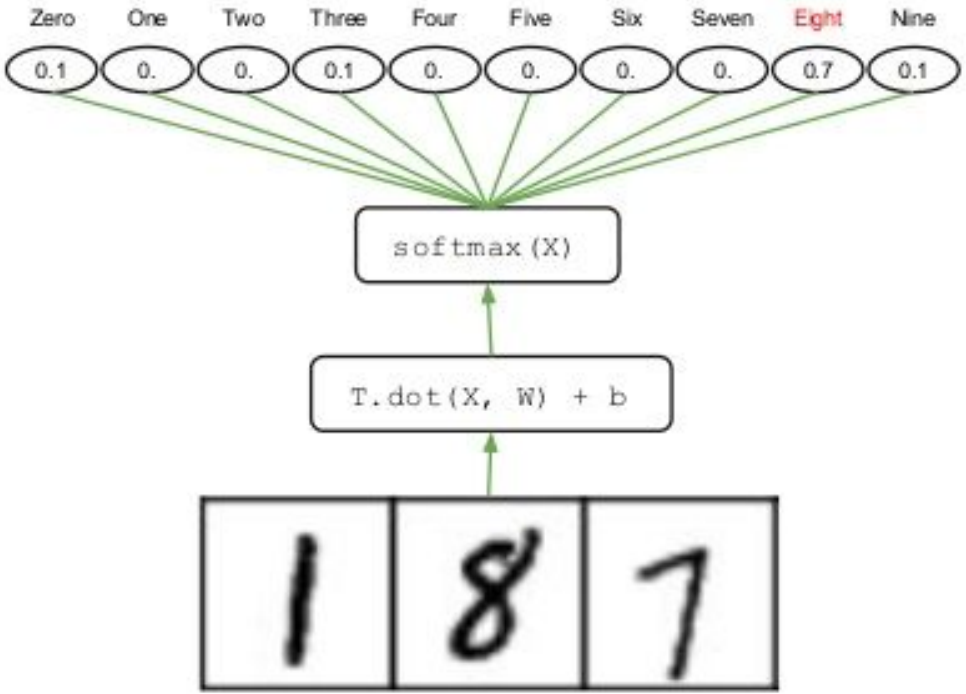
LABSIN

Neural Networks Fully Connected



LABSIN

Softmax activation function



Let’s build our first model

To make the network ready for training, we need to pick three more things:

- **A loss function:** How the network will be able to measure how good a job it's doing on its training data, and thus how it will be able to steer itself in the right direction.
- **An optimizer:** The mechanism through which the network will update itself based on the data it sees and its loss function.
- **Metrics** to monitor during training and testing: i.e. accuracy (the fraction of the images that were correctly classified).

```
network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)
```

LABSIN

But before.. One more thing..

Before training, we'll preprocess the data by reshaping it into the shape the network expect

```
train_images <- array_reshape(train_images, c(60000, 28 * 28))
train_images <- train_images / 255

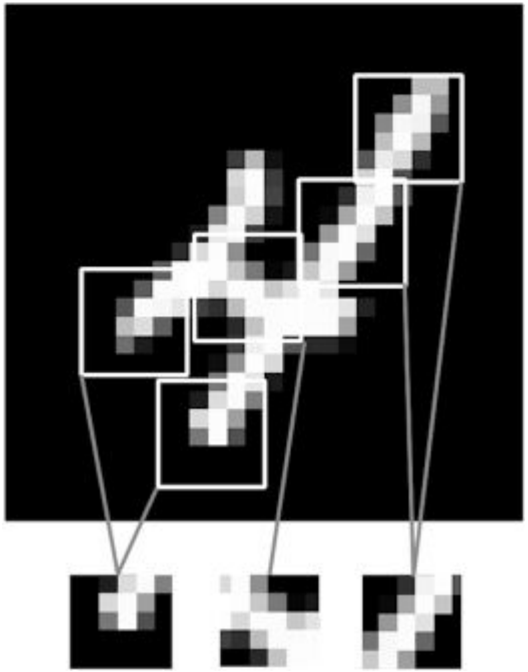
test_images <- array_reshape(test_images, c(10000, 28 * 28))
test_images <- test_images / 255
```

LAB 3: Regression, the Boston Dataset.

1. What is the output of the Keras model when using regression?
2. Callbacks
3. Scaling vs. no Scaling.

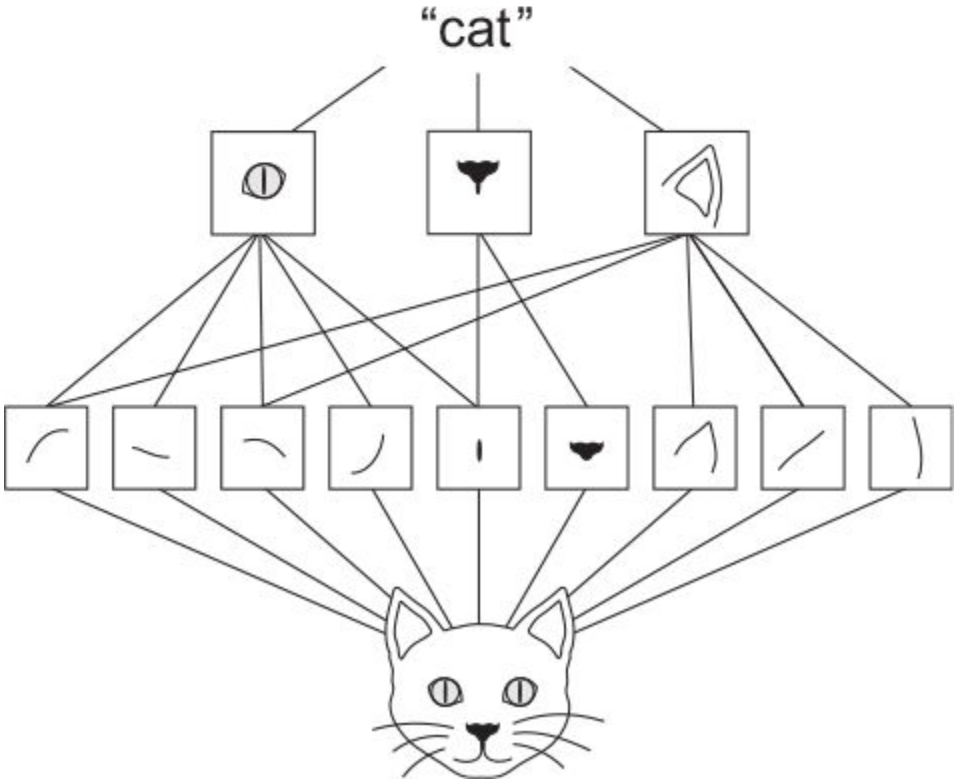
Convolutional Neural Networks: (CNN)

Any NN including a Convolutional Layer



1. **Dense layers learn global patterns** (for example, for an MNIST digit, patterns involving all pixels)
2. **Convolution layers** learn local patterns

The hierarchical approach of CNN



Convolutional Neural Networks in KERAS

```
layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = 'relu')
```

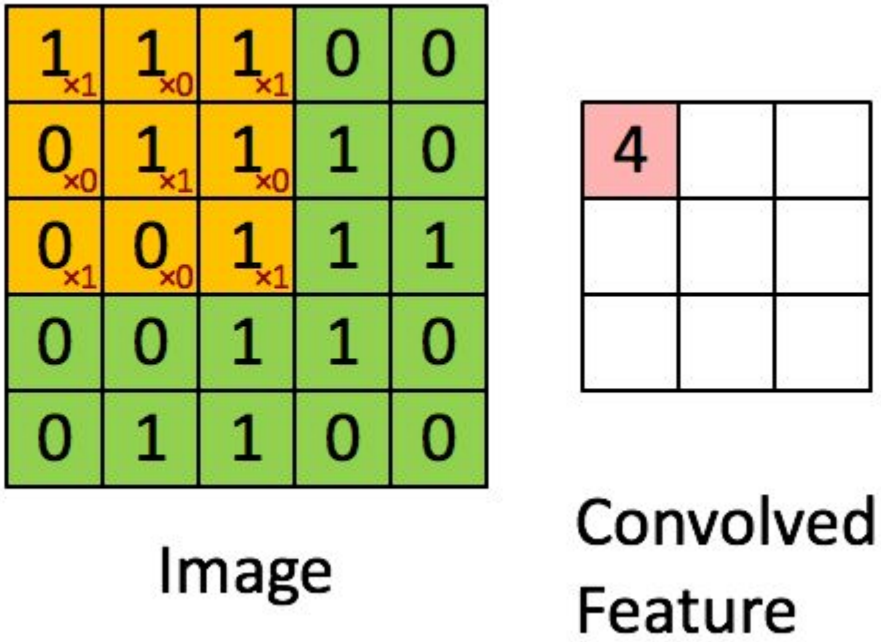
Convolutions operate over 3D tensors, called ***feature maps***, with two spatial axes (*height* and *width*) as well as a *depth* axis (also called the *channels* axis).

- For an **RGB image, the dimension of the depth axis is 3**, because the image has three color channels: red, green, and blue.
- For a **black-and-white picture, like the MNIST digits, the depth is 1** (levels of gray)

Convolutional Neural Networks

- This output feature map is still a 3D tensor: it has a width and a height.
- The output depth is a parameter of the layer, **the so called filters.**
- Filters encode specific aspects of the input data: at a high level, a single filter could encode the concept "presence of a face in the input," for instance.

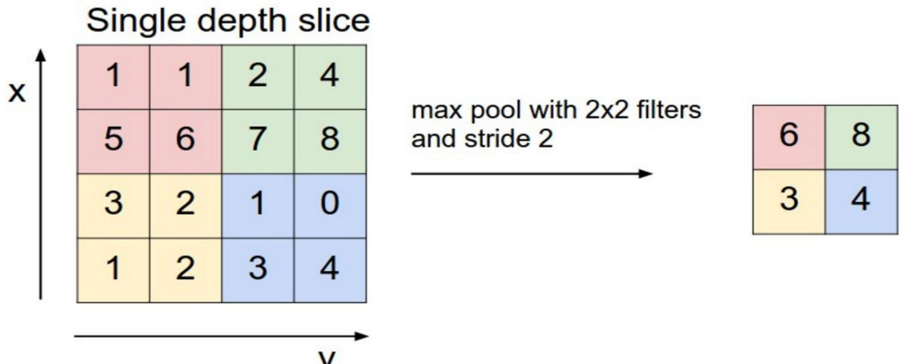
The Convolution Operation



The (Max) Pooling Operation

```
layer_max_pooling_2d(pool_size = c(2, 2))
```

- Used in order to downsample the feature maps.
- Max pooling consists of extracting windows from the input feature maps and outputting the max value of each channel.
- Similar to convolution, except they're transformed via a hardcoded max/average/min tensor operation instead of kernel.



LAB 4: The MNIST DATASET with CNN

1. Compare the performance with Dense Layers
2. The input format has changed (check NN)
3. The Flatten() “Layer”
4. Adding Layers

LAB5: The IMDB Dataset with RNN

- Dataset of 25,000 movies reviews from IMDB, labeled by sentiment (positive/negative).
- Reviews have been preprocessed, and each review is encoded as a sequence of word indexes (integers).
- For convenience, words are indexed by overall frequency in the dataset, so that for instance the integer "3" encodes the 3rd most frequent word in the data.
- This allows for quick filtering operations such as: "only consider the top 10,000 most common words, but eliminate the top 20 most common words".

More examples at

https://keras.rstudio.com/articles/examples/index.html

Carlos A. Catania (PhD)
(AKA **Harpo**)
LABSIN - Ingeniería - UNCuyo
@harpolabs
harpo@ingenieria.uncuyo.edu.ar

LABSIN

http://labsin.org