Deep Convolutional Neural Networks

Aldo Ivan Aguilar Aldecoa

Instituto Tecnológico de Monterrey Campus Estado de México

November 26, 2019

Let's get Prepared!

Go to:

`https://github.com/robotica-cem/conv_neural_nets_seminar`

And Follow the README Instructions.

What's a Neural Network?

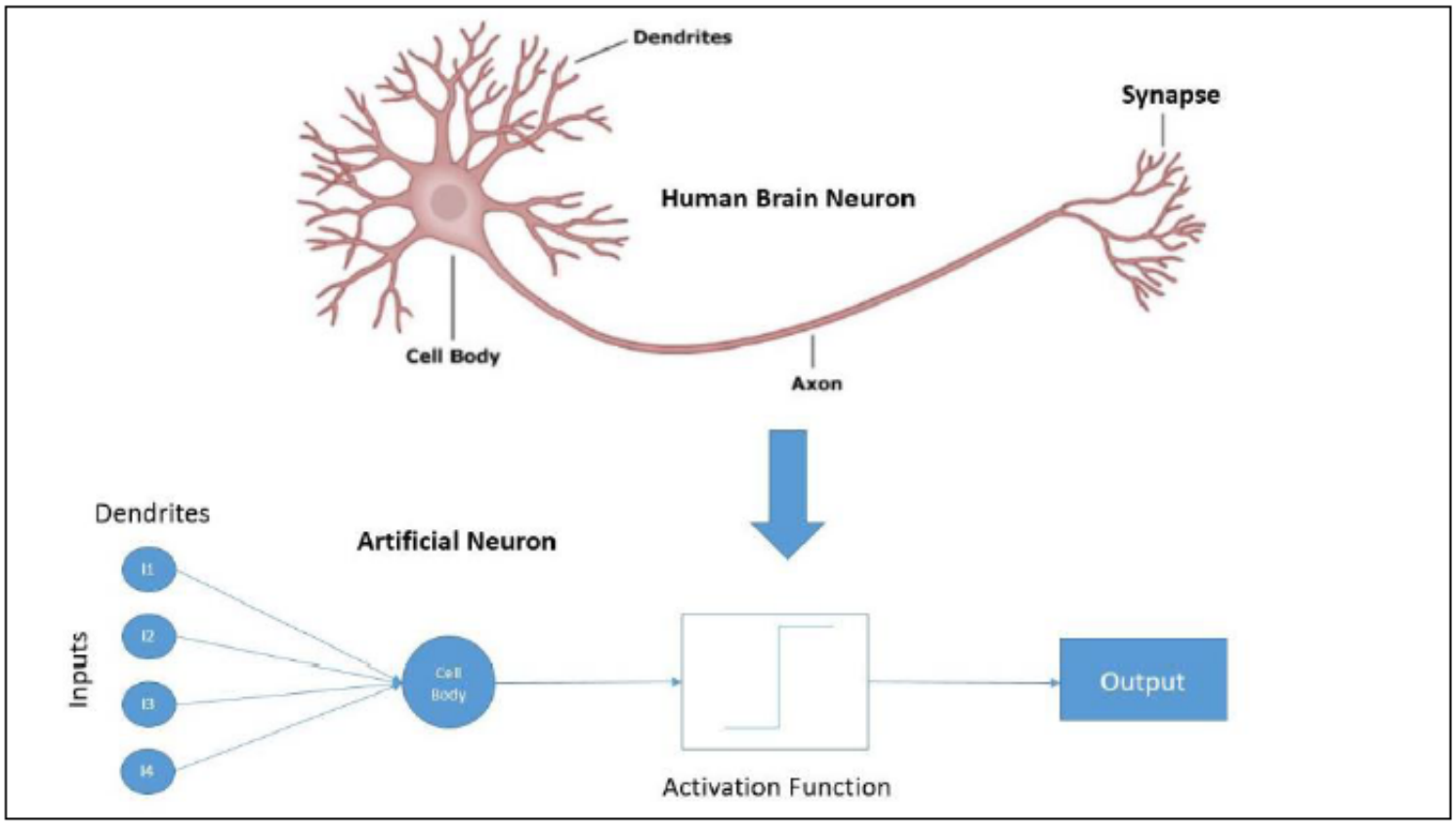


Figure: Human vs Artificial Neuron Comparisson.

What's a Neural Network?

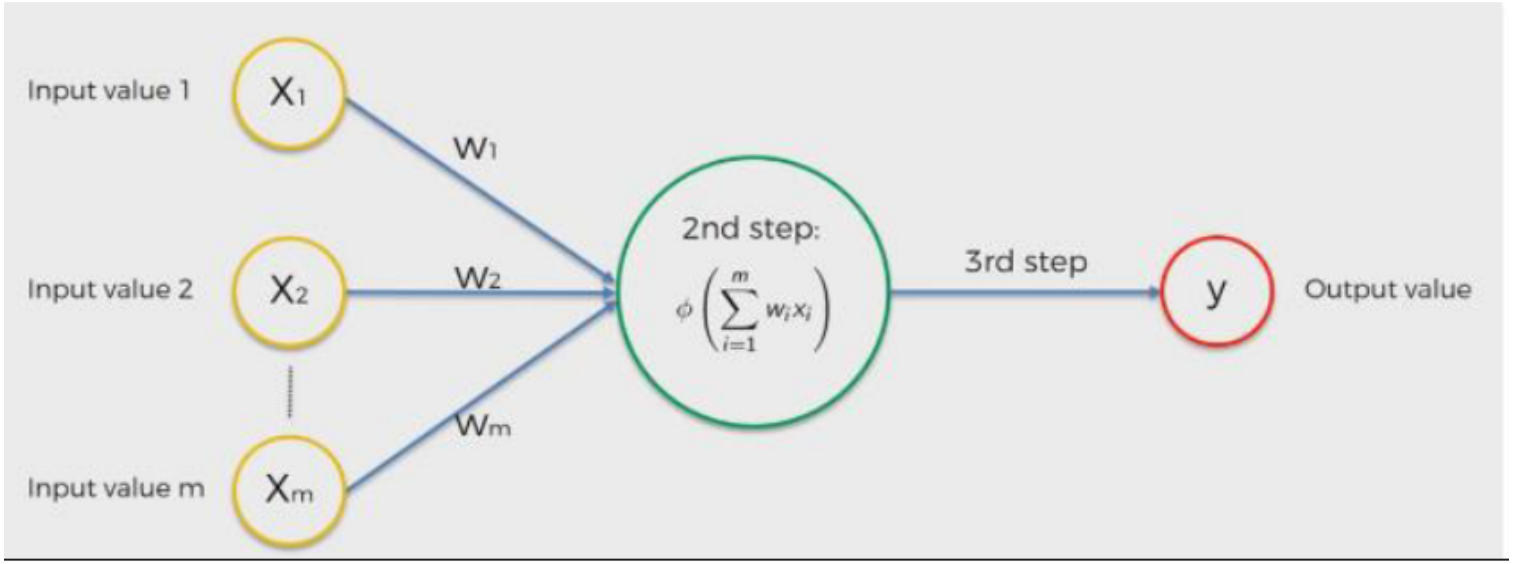


Figure: Basic Neuron Architecture.

n	:	Number of the current input value (n : 1, 2, ..., m)
x_n	:	Single input value of the network.
W_n	:	Assigned weights of the current connection.
Y	:	The predicted Output of the Neuron.
ϕ	:	Activation Function.

What's a Neural Network?

- Multiple interconnected Neurons.
- Multilayered Architecture.
- Multiple inputs mapped to specific number of outputs.
- Input data is passed through inner conversion layers.

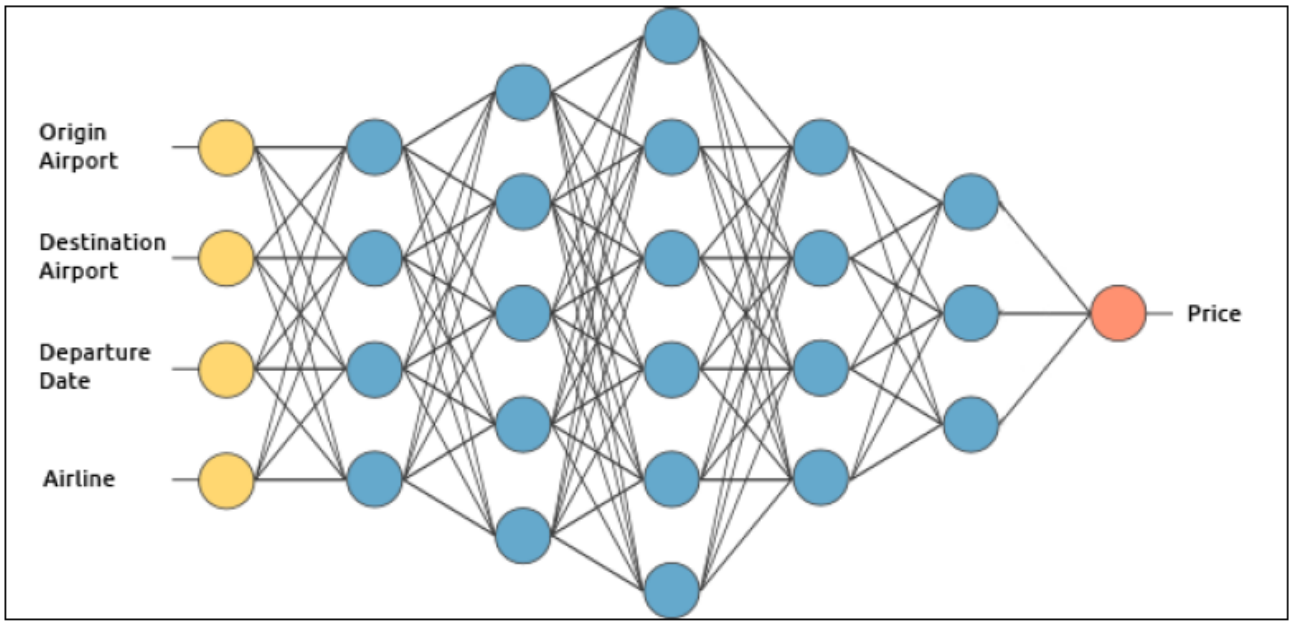


Figure: Representation of a Neural Network application for airplane tickets price prediction.

What's Machine Learning?

- Learn from Raw Data to make Predictions.
- Identify patrons from learning input data.
- Adjust parameters to generate a desired Output.
- **A change in paradigms...**

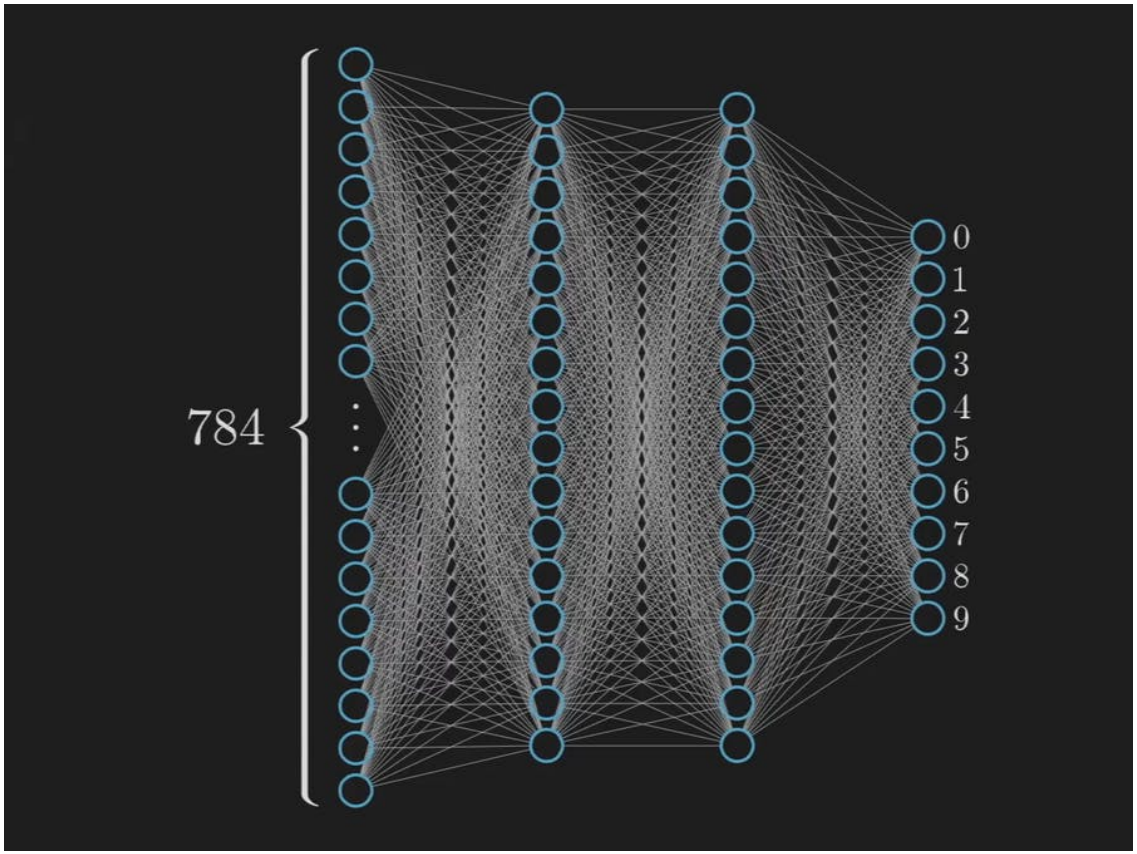
What's Machine Learning?

The Development of a Machine Learning System is normally conformed of three phases:

1. Training
2. Validation
3. Testing

What's a Deep Neural Network?

- Neural Network implementations that require to process big amount of input data.
- Large Architectures for the implemented model.
- High Utility for **Computer Vision Applications**.



Convolutional Neural Networks

Consider a Computer Vision Implementation where an RGB image of a high resolution, say 8K (7680x4320 px) is going to be processed using a Neural Network.

Convolutional Neural Networks

- Extracts/Assigns importance to various aspects/elements in an image.
- Capable of differentiate one element from another.
- Learn to apply different filters autonomously with enough training.
- Captures Spatial and Temporal Dependencies in an image by applying relevant filters.

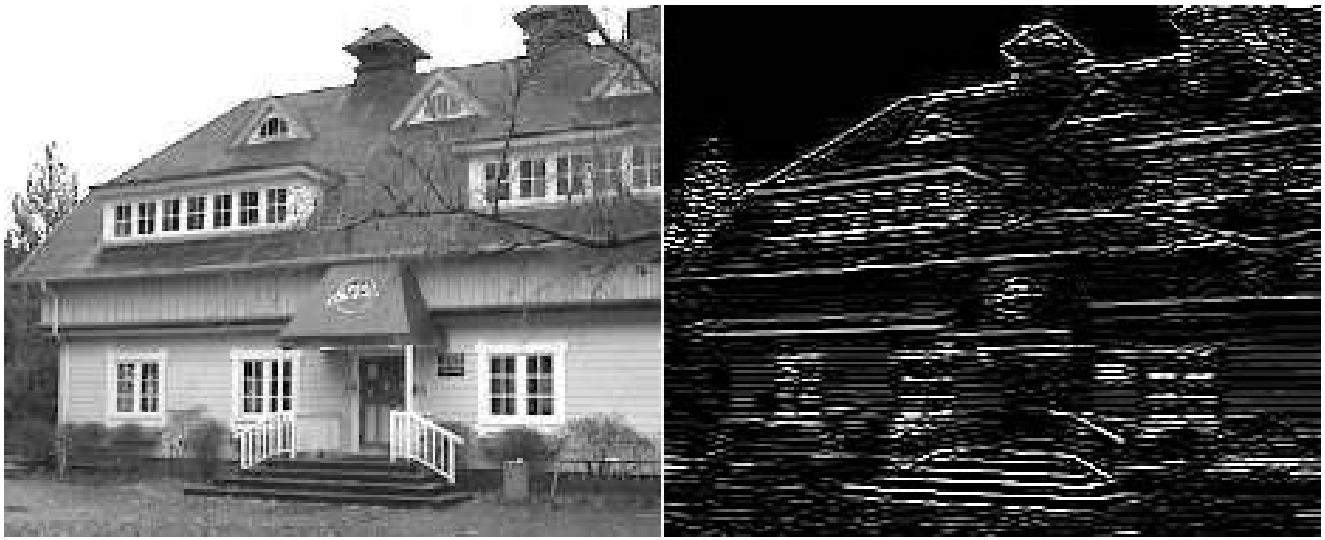
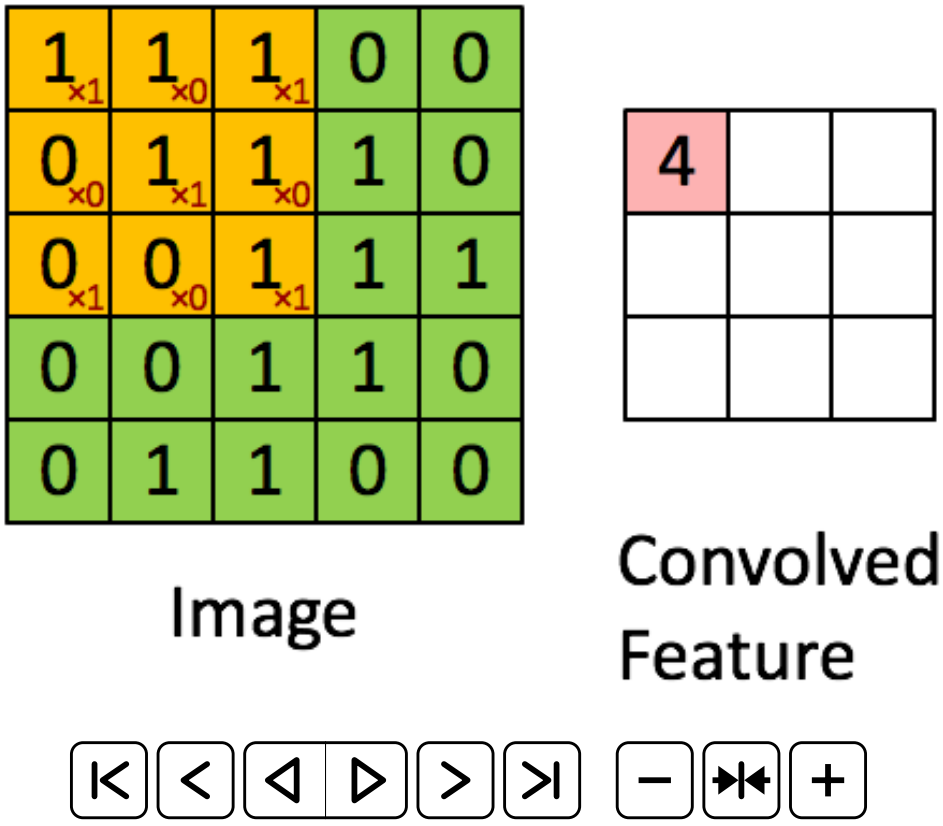


Figure: Example of a horizontal line convolutional filter Output.

Convolutional Neural Networks



Convolutional Neural Networks

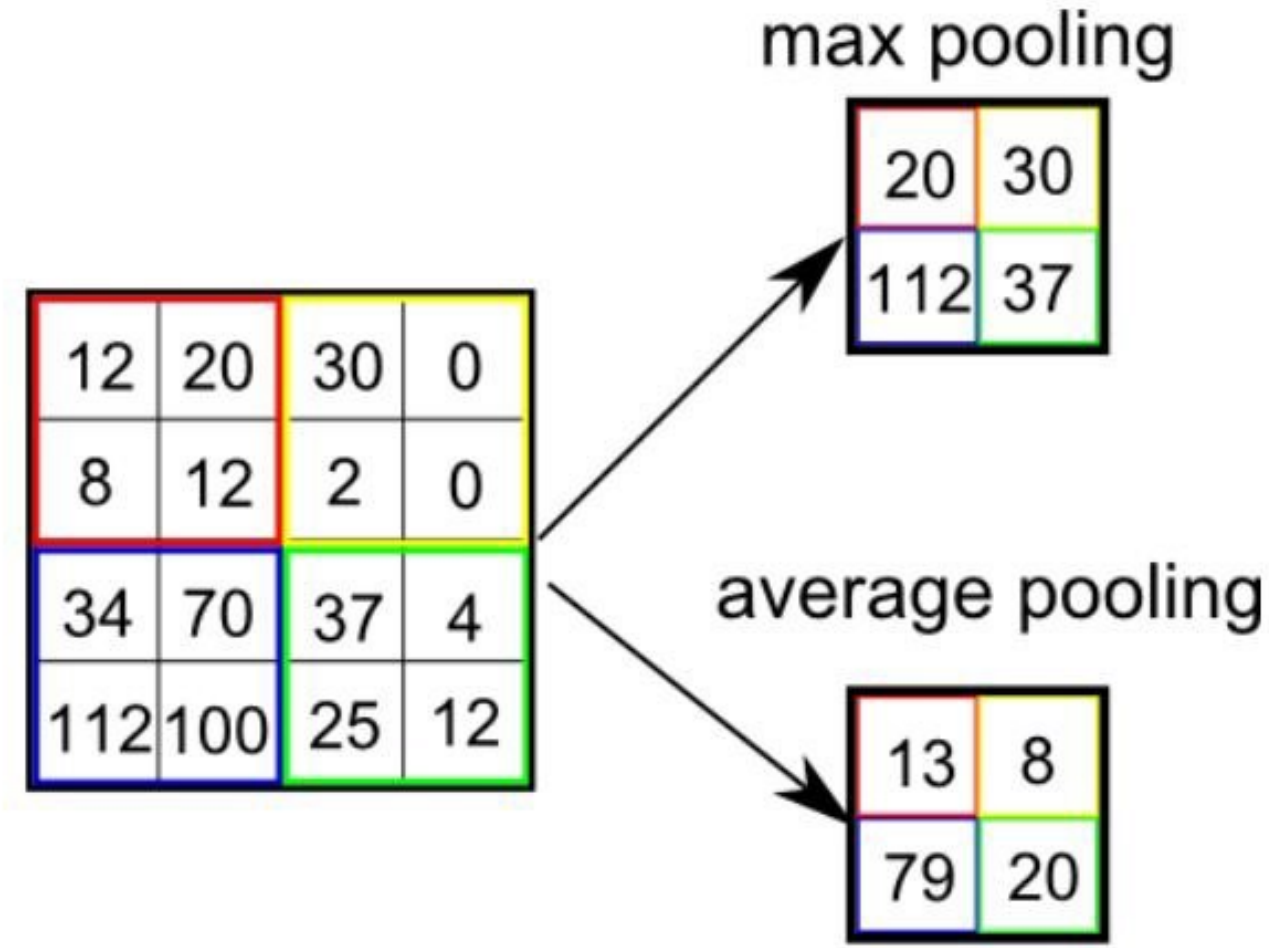


Figure: Pooling Operation Example.

Convolutional Neural Networks

This Convolutional operation allows to extract the most important features of an image in a reduced size. In other words, we can express a large amount of pixels in an image more compactly.

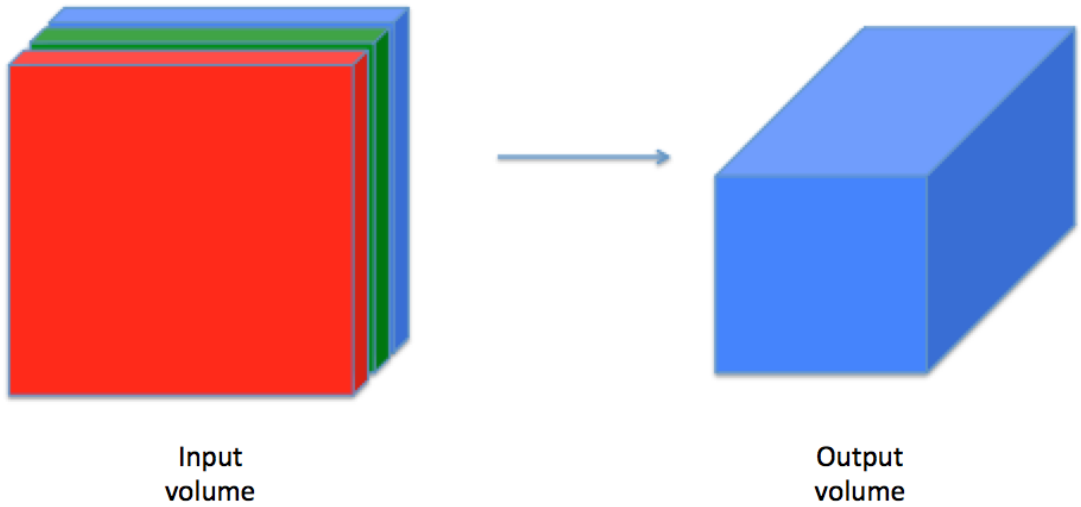
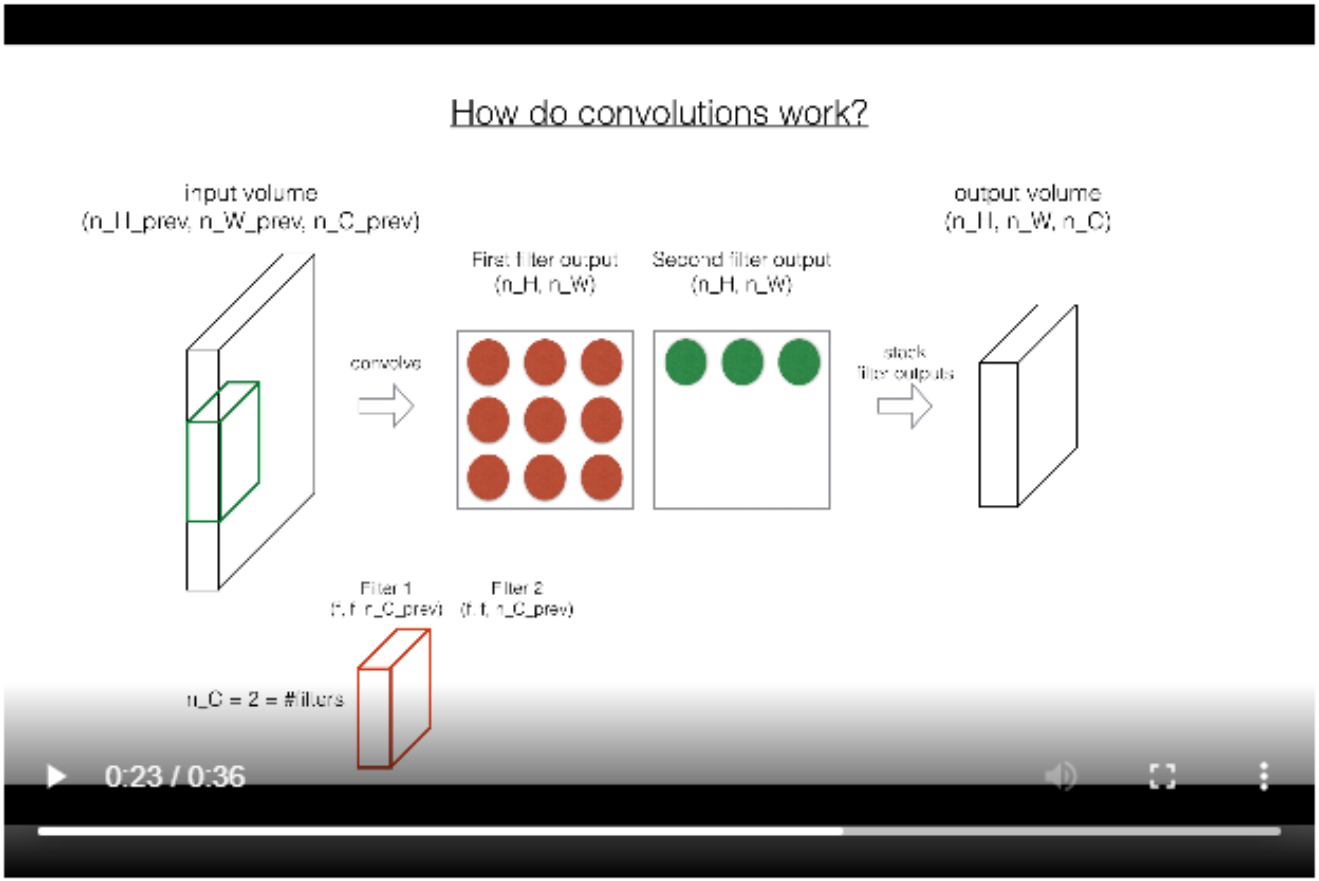


Figure: Convolution Process Example.

Convolutional Neural Networks

To achieve this volume reduction task, we apply multiple convolution operations to obtain fundamental elements of the input volume, for example edge detection, color filtering, horizontal lines, etc.



Convolutional Neural Networks

The deeper we go in our network the more complex features our model is capable of detecting in a smaller volume, thus it's easier to extract the fundamental features of the input and feed them to a fully connected set of layers.

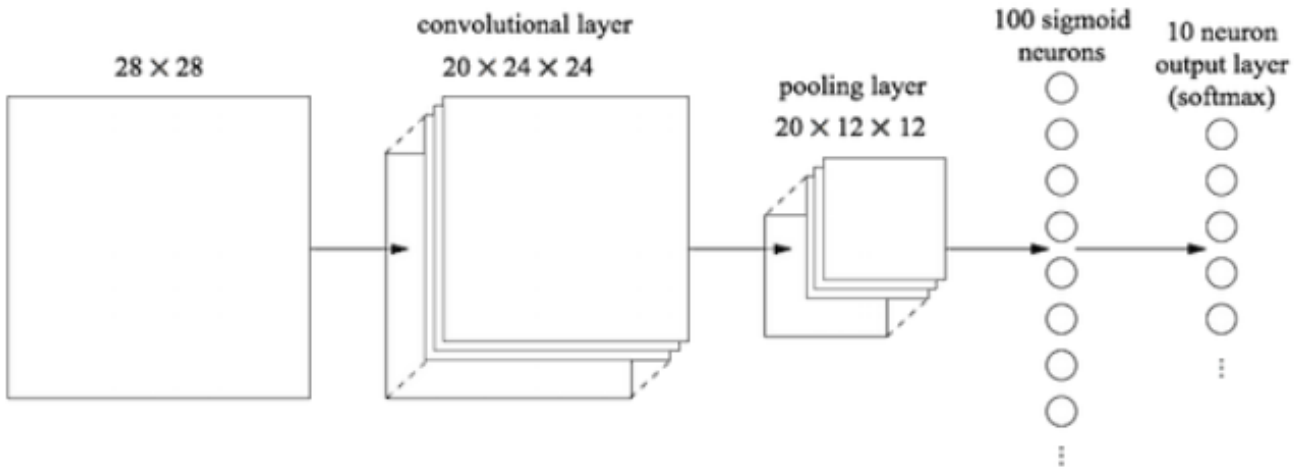


Figure: Convolutional Neural Network Example.

Convolutional Neural Networks

Figure: DNN Layer 1 Activation Outputs.

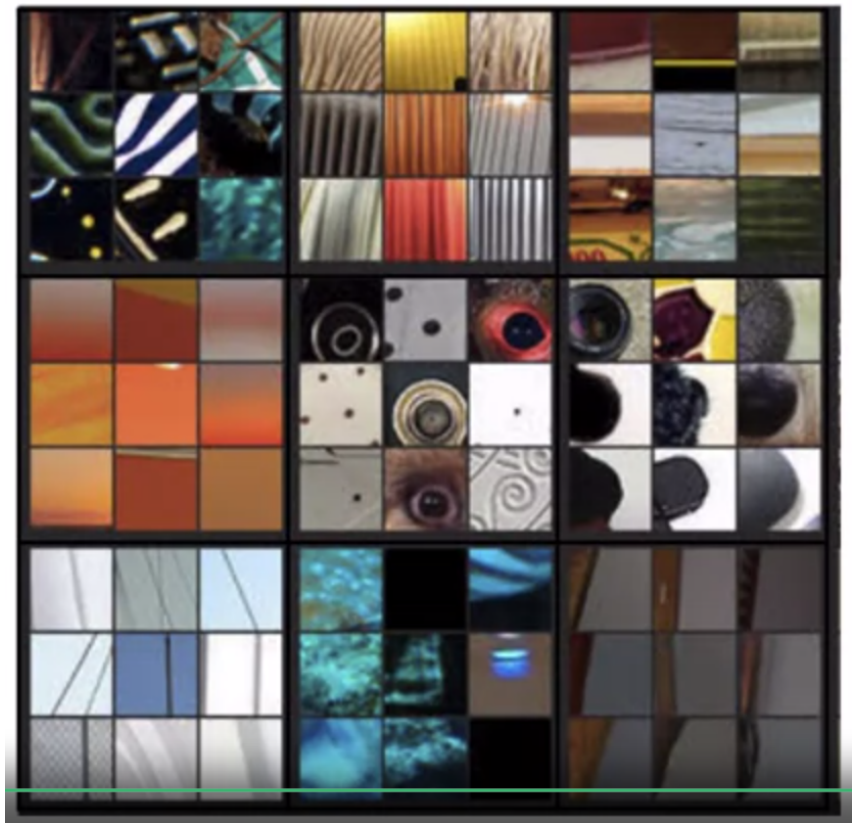


Figure: DNN Layer 2 Activation Outputs.

Convolutional Neural Networks

Figure: DNN Layer 3 Activation Outputs.

Figure: DNN Layer 4 Activation Outputs.

Convolutional Neural Networks

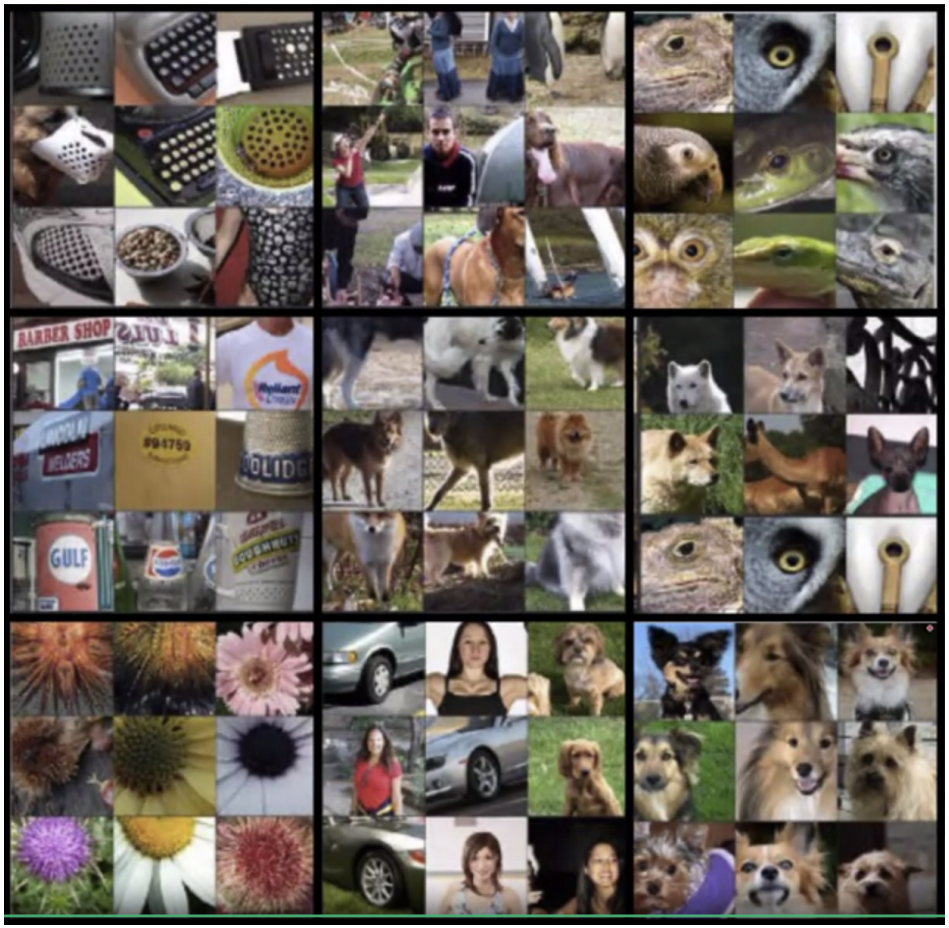


Figure: DNN Layer 5 Activation Outputs.

Convolutional Neural Networks

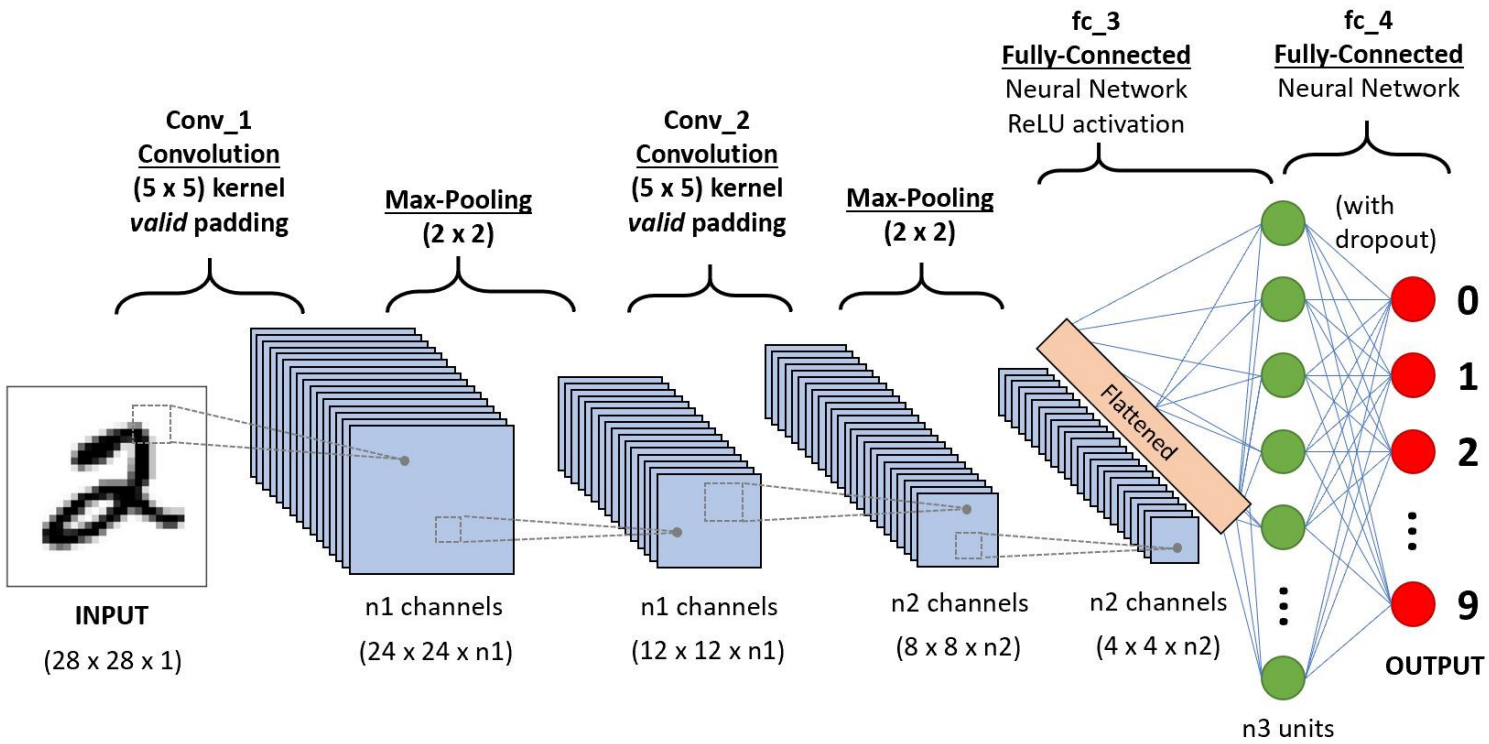


Figure: DCNN Architecture for Hand Written Number Classificaction.

How to code this?

There are many different open source tools for Machine Learning development, **Tensor Flow** is one of the mostly used for Neural Network architecture developments.

DCNN Example in TensorFlow

Handwritten number classification using TensorFlow's MNIST dataset

Figure: MNIST Dataset Elements examples.

DCNN Example in TensorFlow

```
1  import matplotlib.pyplot as plt
2  import tensorflow as tf
3  import numpy as np
4  import random
5
6  # Variable used to stablish the training desired accuracy.
7  DESIRED_ACCURACY = 0.998
8
9  # Callback function usded to check the current epoch accuracy and end the training process
10 # if the desired accuracy is achieved.
11 class myCallback(tf.keras.callbacks.Callback):
12         def on_epoch_end(self, epoch, logs={}):
13                 if(logs.get('accuracy') > DESIRED_ACCURACY):
14                         print("\nReached " + str(DESIRED_ACCURACY * 100) + "% accuracy so cancelling training!")
15                         self.model.stop_training = True
16
17
18 callbacks = myCallback()
```

DCNN Example in TensorFlow

```
19
20  # Dataset load from TensorFlow MNIST Dataset
21  mnist = tf.keras.datasets.mnist
22  (orig_training_images, training_labels) , (orig_test_images, test_labels) = mnist.load_data()
23
24  # Fromating of the obtained training and testing datasets for its further usage
25  training_images_shape = orig_training_images.shape
26  testing_images_shape = orig_test_images.shape
27  print("Training images dataset shape: " + str(training_images_shape))
28  print("Testing images dataset shape: " + str(testing_images_shape))
29
30  training_images = orig_training_images.reshape(training_images_shape[0], training_images_shape[1], training_images_shape[2], 1)
31  training_images = training_images / 255.0
32
33  test_images = orig_test_images.reshape(testing_images_shape[0], testing_images_shape[1], testing_images_shape[2], 1)
34  test_images = test_images / 255.0
35
```

DCNN Example in TensorFlow

```
36
37  # Display examples of the dataset images
38  image_size = training_images_shape[1]
39  display_grid = np.zeros((image_size, image_size * 10))
40
41  for i in range(10):
42          idx = np.where(test_labels == i)[0][0]
43          orig_image = orig_test_images[idx]
44          display_grid[:, i * image_size : (i + 1) * image_size] = orig_image
45
46  print("Press any key to continue...")
47  fig = plt.figure()
48  plt.axis('off')
49  plt.grid(False)
50  plt.imshow(display_grid, cmap='gray')
51  plt.draw()
52  plt.waitforbuttonpress(0)
53  plt.close(fig)
54
```

DCNN Example in TensorFlow

Dataset Examples Plotting:

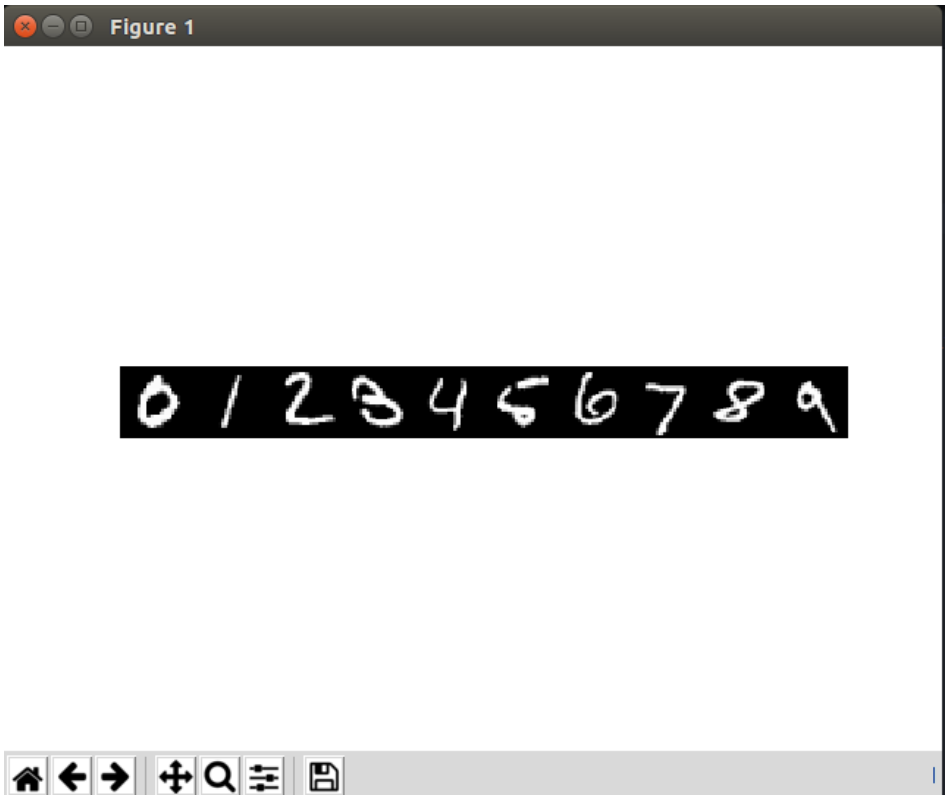


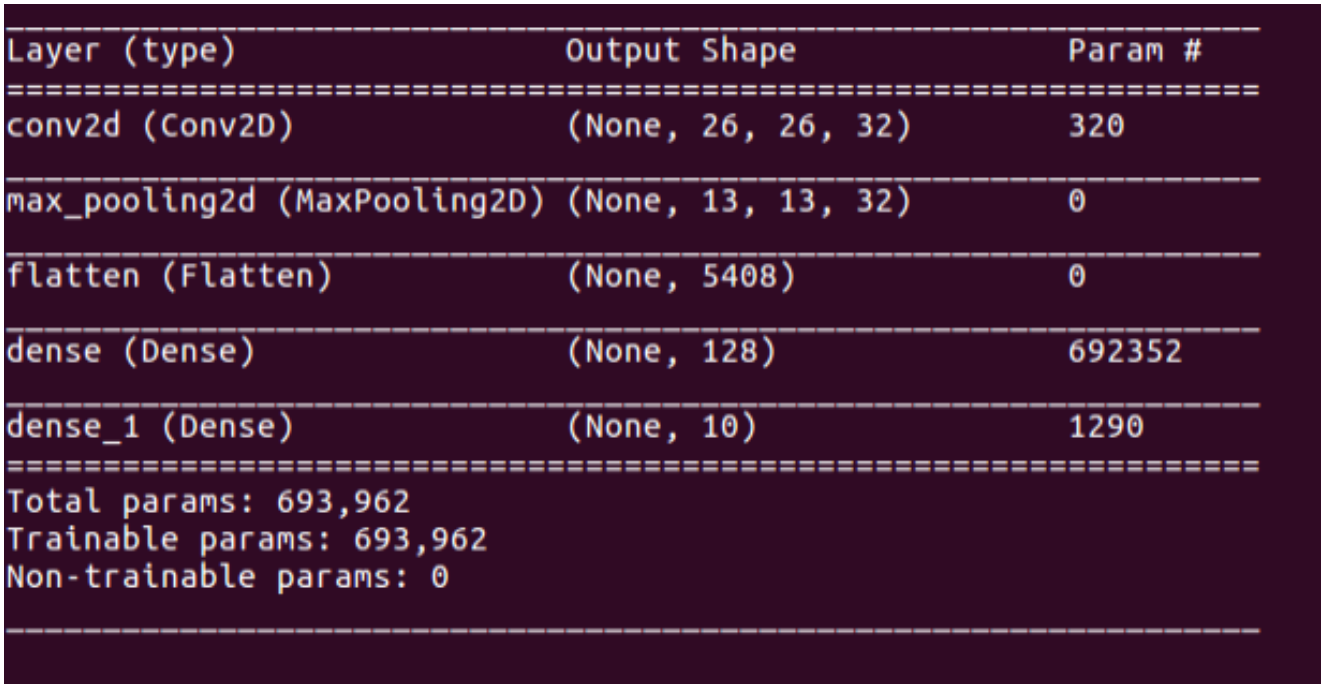
Figure: MNIST Dataset Elements examples.

DCNN Example in TensorFlow

```
55
56  # Definition of the CNN model
57  model = tf.keras.models.Sequential([
58          tf.keras.layers.Conv2D(32, (3, 3), activation = 'relu', input_shape = (image_size, image_size, 1)),
59          tf.keras.layers.MaxPooling2D(2,2),
60          tf.keras.layers.Flatten(),
61          tf.keras.layers.Dense(128, activation = 'relu'),
62          tf.keras.layers.Dense(10, activation = 'softmax')
63  ])
64
65  model.compile(optimizer='adam', loss = 'sparse_categorical_crossentropy', metrics =['accuracy'])
66  model.summary()
```

DCNN Example in TensorFlow

Model Summary:



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
flatten (Flatten)	(None, 5408)	0
dense (Dense)	(None, 128)	692352
dense_1 (Dense)	(None, 10)	1290

Total params: 693,962
Trainable params: 693,962
Non-trainable params: 0

Figure: DCNN Model Summary

DCNN Example in TensorFlow

```
68  # NOTICE: If you want to train the model comment the second block of code contained between the asterisks line [*] and leave the first one
69  # However if you don't want to train the model and only desire to load the pretrained weights for the model comment the first block of code
70
71  # Training of the CNN model with the selected training set, the obtained weights are stored in the given directory
72  # # ************************************************
73  # model.fit(training_images, training_labels, epochs = 10, callbacks = [callbacks])
74  # model.save_weights('./checkpoints/numbers_model')
75  # ************************************************
76
77  # Load the pŕetrained model weights from the given directory
78  # ************************************************
79  model.load_weights('./checkpoints/numbers_model')
80  # ************************************************
```

DCNN Example in TensorFlow

```
82  # Testing of the trained model with 10 randome testing image_size
83  for i in range(10):
84          idx = random.randint(0, orig_test_images.shape[0])
85
86          orig_image = orig_test_images[idx]
87          image = orig_image.reshape(1, image_size, image_size, 1)
88          image = image / 255.0
89          image = tf.cast(image, tf.float32)
90
91          prediction = list(list(model.predict(image))[0])
92
93          print('\n')
94          print("Press any key to continue...")
95          print('Predicted: ' + str(prediction.index(max(prediction))))
96          print('Real: ' + str(test_labels[idx]))
97
98          fig = plt.figure()
99          plt.axis('off')
100         plt.grid(False)
101         plt.imshow(orig_image, cmap='gray')
102         plt.draw()
103         plt.waitforbuttonpress(0)
104         plt.close(fig)
```

DCNN Example in TensorFlow

Model Testing Predictions:

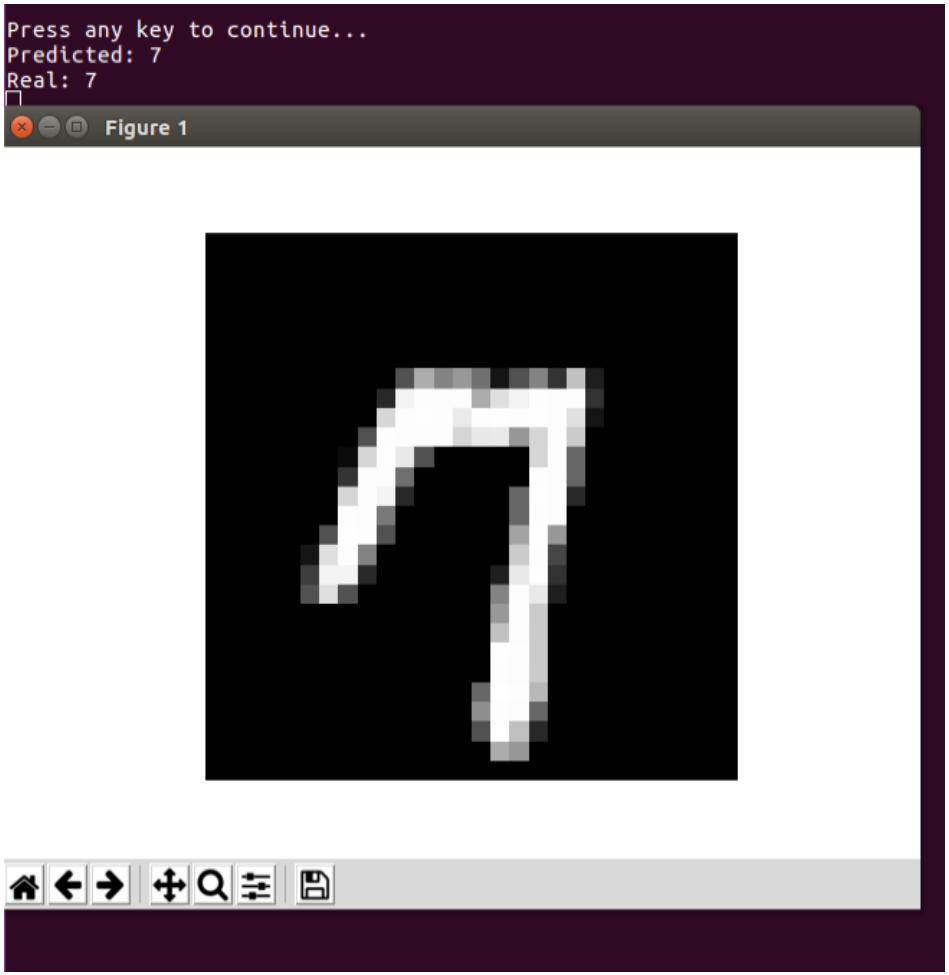


Figure: DCNN Model Prediction Example

DCNN Example in TensorFlow

1. Open the Pinta Image Editor App.
2. Go to Image $\rightarrow$ Resize Canvas.
3. Select By absolute size.
4. Deselect the Maintain aspect ratio option.
5. Set Canvas width and height to 28 pixels.
6. Zoom in to a 2,400%.
7. Select the Paint Bucket and set the canvas background to black.
8. Select the Paintbrush and draw any number from 0 to 9 in white.
9. Save the generated image inside the img folder of the repository.
10. Execute the tf_test.py script to check the model predicted number.

References

- Sumit Saha, A Comprenhensive Guide to Convolutional Neural Networks, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
- Joshua Wiley, 2016. What is Deep Learning? In: R Deep Learning Essentials. Packt Publishing, Birmingham.
- Sah Eshant, 2018. Deep Learning — In simple words... [WWW Document]. A Medium Corporation. URL https://medium.com/@eshant.sah/deep-learning-in-simple-words-d6e027468836 (accessed 5.7.19).