Artificial Intelligence Fullstack
[Course]
Week 16 – Deep Learning –
Convolutional Neural Networks (CNN)
[See examples / code in GitHub code repository]

It is not about Theory, it is 20% Theory and 80% Practical – Technical/Development/Programming [Mostly Python based]

DL | What is a CNN- Convolutional Neural Network?

A Convolutional Neural Network (CNN or ConvNet) is a deep learning algorithm specifically designed for any task where object recognition is crucial such as image classification, detection, and segmentation. Many real-life applications, such as self-driving cars, surveillance cameras, and more, use CNNs.

The importance of CNNs

These are several reasons why CNNs are important:

- Unlike traditional machine learning models like SVM and decision trees that require manual feature extractions, CNNs can perform automatic feature extraction at scale, making them efficient.
- The convolutions layers make CNNs translation invariant, meaning they can recognize patterns from data and extract features regardless of their position, whether the image is rotated, scaled, or shifted.
- Multiple pre-trained CNN models such as VGG-16, ResNet50, Inceptionv3, and EfficientNet are proved to have reached state-of-the-art results and can be fine-tuned on news tasks using a relatively small amount of data.
- CNNs can also be used for non-image classification problems and are not limited to natural language processing, time series analysis, and speech recognition.

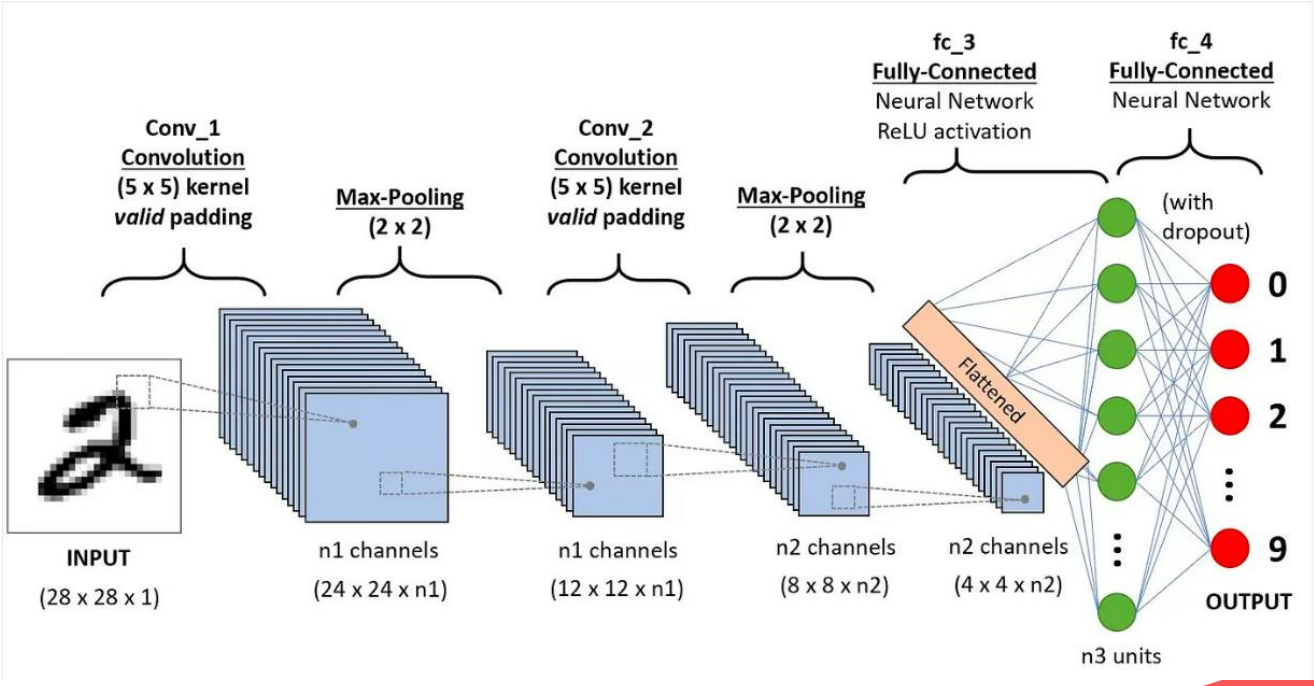
Reference:

https://www.datacamp.com/tutorial/cnn-tensorflow-python
https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns
https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-tensorflow/
https://www.tensorflow.org/tutorials/images/cnn
https://developer.ibm.com/articles/using-deep-learning-to-take-on-covid-19/

DL | Architecture of a CNN

CNNs' architecture tries to mimic the structure of neurons in the human visual system composed of multiple layers, where each one is responsible for detecting a specific feature in the data. As illustrated in the image below, the typical CNN is made of a combination of four main layers:

- ❑ Convolutional layers
- ❑ Rectified Linear Unit (ReLU for short)
- ❑ Pooling layers
- ❑ Fully connected layers



Reference:

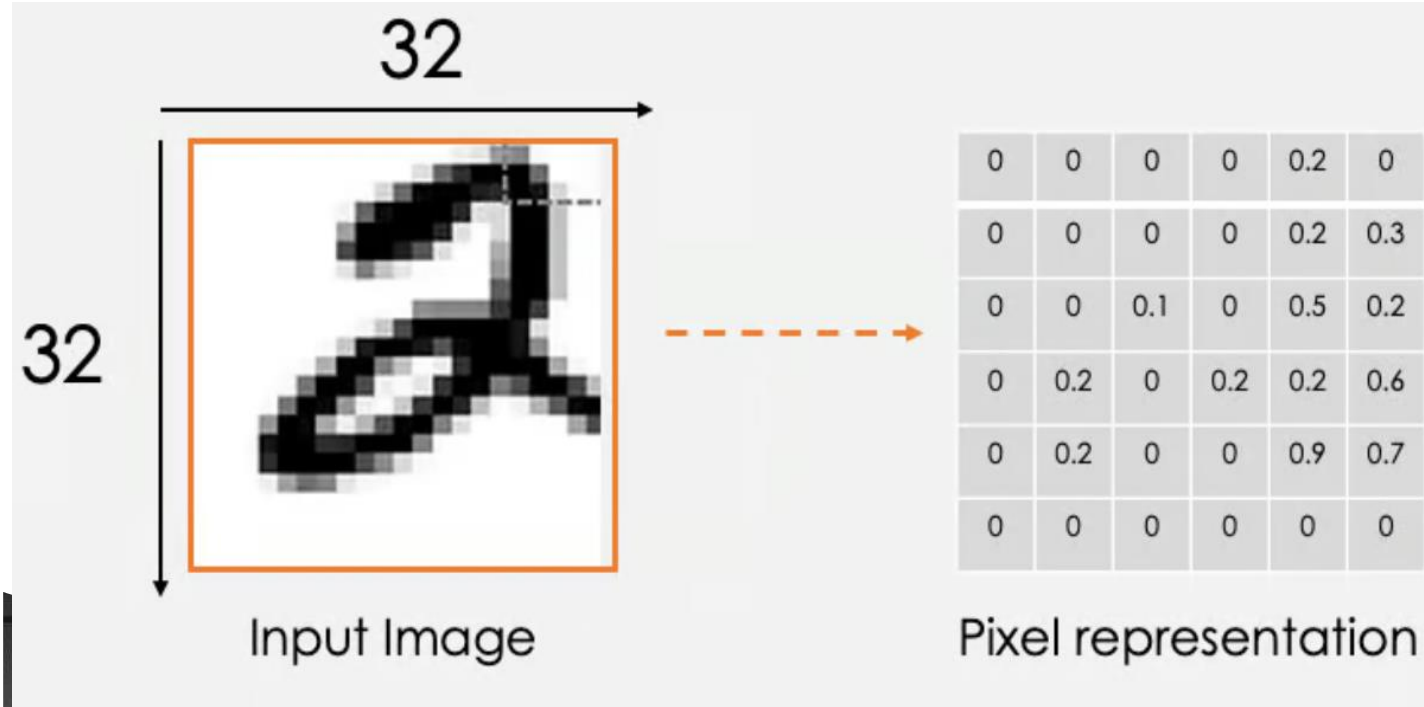
https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning
https://aiml.com/what-is-a-multilayer-perceptron-mlp/
https://builtin.com/data-science/feedforward-neural-network-intro
www.analyticsvidhya.com/blog/2020/07/neural-networks-from-scratch-in-python-and-r

DL | Convolution layers

This is the first building block of a CNN. As the name suggests, the main mathematical task performed is called convolution, which is the application of a sliding window function to a matrix of pixels representing an image. The sliding function applied to the matrix is called kernel or filter, and both can be used interchangeably.

In the convolution layer, several filters of equal size are applied, and each filter is used to recognize a specific pattern from the image, such as the curving of the digits, the edges, the whole shape of the digits, and more.

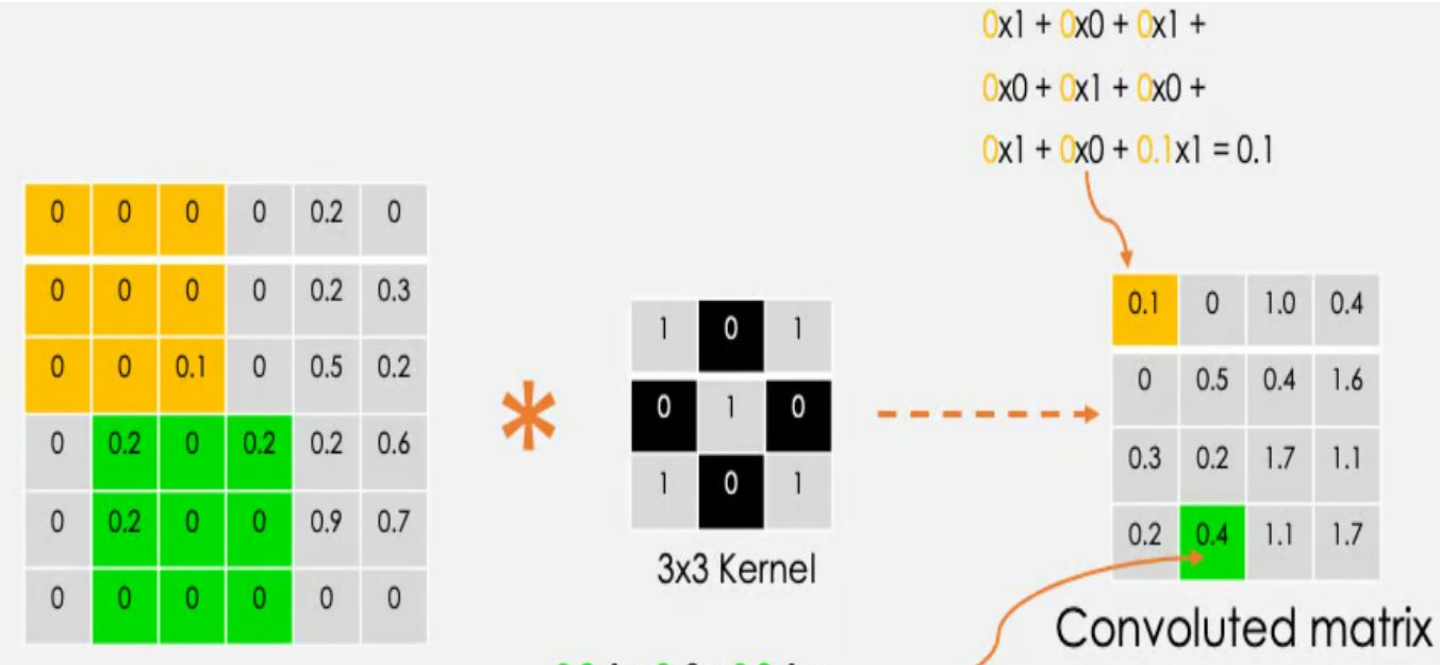
Let's consider this 32x32 grayscale image of a handwritten digit. The values in the matrix are given for illustration purposes.



DL | Do we have to manually find these weights?

In real life, the weights of the kernels are determined during the training process of the neural network. Using these two matrices, we can perform the convolution operation by taking applying the dot product, and work as follows:

1. Apply the kernel matrix from the top-left corner to the right.
2. Perform element-wise multiplication.
3. Sum the values of the products.
4. The resulting value corresponds to the first value (top-left corner) in the convoluted matrix.
5. Move the kernel down with respect to the size of the sliding window.
6. Repeat from step 1 to 5 until the image matrix is fully covered.
7. The dimension of the convoluted matrix depends on the size of the sliding window. The higher the sliding window, the smaller the dimension.



DL | Few more things

Another name associated with the kernel in the literature is feature detector because the weights can be fine-tuned to detect specific features in the input image.

For instance:

- Averaging neighboring pixels kernel can be used to blur the input image.
- Subtracting neighboring kernel is used to perform edge detection.

The more convolution layers the network has, the better the layer is at detecting more abstract features.

Activation function

A ReLU activation function is applied after each convolution operation. This function helps the network learn non-linear relationships between the features in the image, hence making the network more robust for identifying different patterns. It also helps to mitigate the vanishing gradient problems.

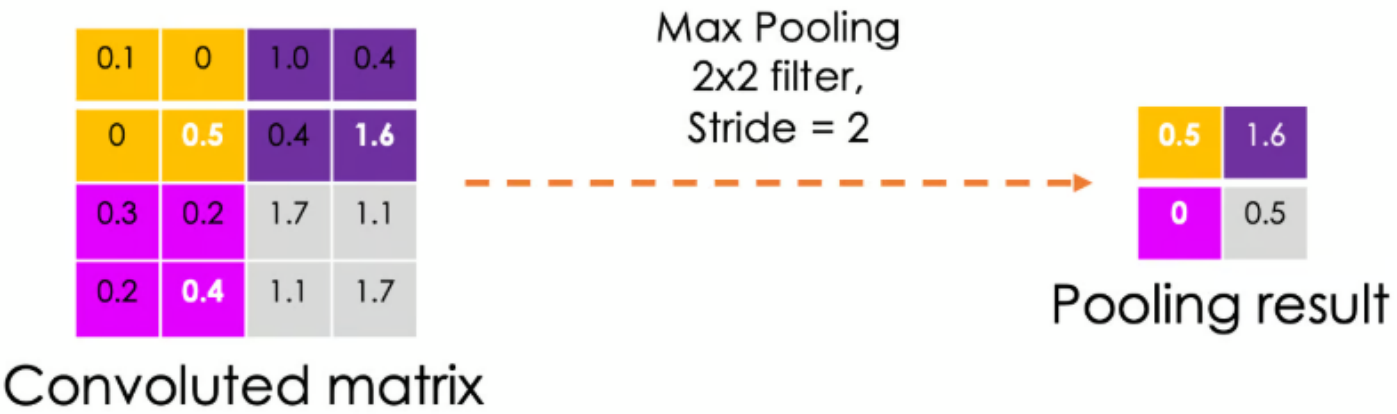
DL | Pooling layer

The goal of the pooling layer is to pull the most significant features from the convoluted matrix. This is done by applying some aggregation operations, which reduces the dimension of the feature map (convoluted matrix), hence reducing the memory used while training the network. Pooling is also relevant for mitigating overfitting.

The most common aggregation functions that can be applied are:

- Max pooling which is the maximum value of the feature map
- Sum pooling corresponds to the sum of all the values of the feature map
- Average pooling is the average of all the values.

Below is an illustration of each of the previous example:



DL | Few Next steps

Also, the dimension of the feature map becomes smaller as the polling function is applied.
The last pooling layer flattens its feature map so that it can be processed by the fully connected layer.

Fully connected layers

These layers are in the last layer of the convolutional neural network, and their inputs correspond to the flattened one-dimensional matrix generated by the last pooling layer. ReLU activations functions are applied to them for non-linearity. Finally, a softmax prediction layer is used to generate probability values for each of the possible output labels, and the final label predicted is the one with the highest probability score.

Dropout

Dropout is a regularization technic applied to improve the generalization capability of the neural networks with a large number of parameters. It consists of randomly dropping some neurons during the training process, which forces the remaining neurons to learn new features from the input data.

DL | CIFAR-10 dataset-Architecture of the network-Example

- The input of the model is a 32x32x3 tensor, respectively, for the width, height, and channels.
- We will have two convolutional layers. The first layer applies 32 filters of size 3x3 each and a ReLU activation function. And the second one applies 64 filters of size 3x3
- The first pooling layer will apply a 2x2 max pooling
- The second pooling layer will apply a 2x2 max pooling as well
- The fully connected layer will have 128 units and a ReLU activation function
- Finally, the output will be 10 units corresponding to the 10 classes, and the activation function is a softmax to generate the probability distributions.

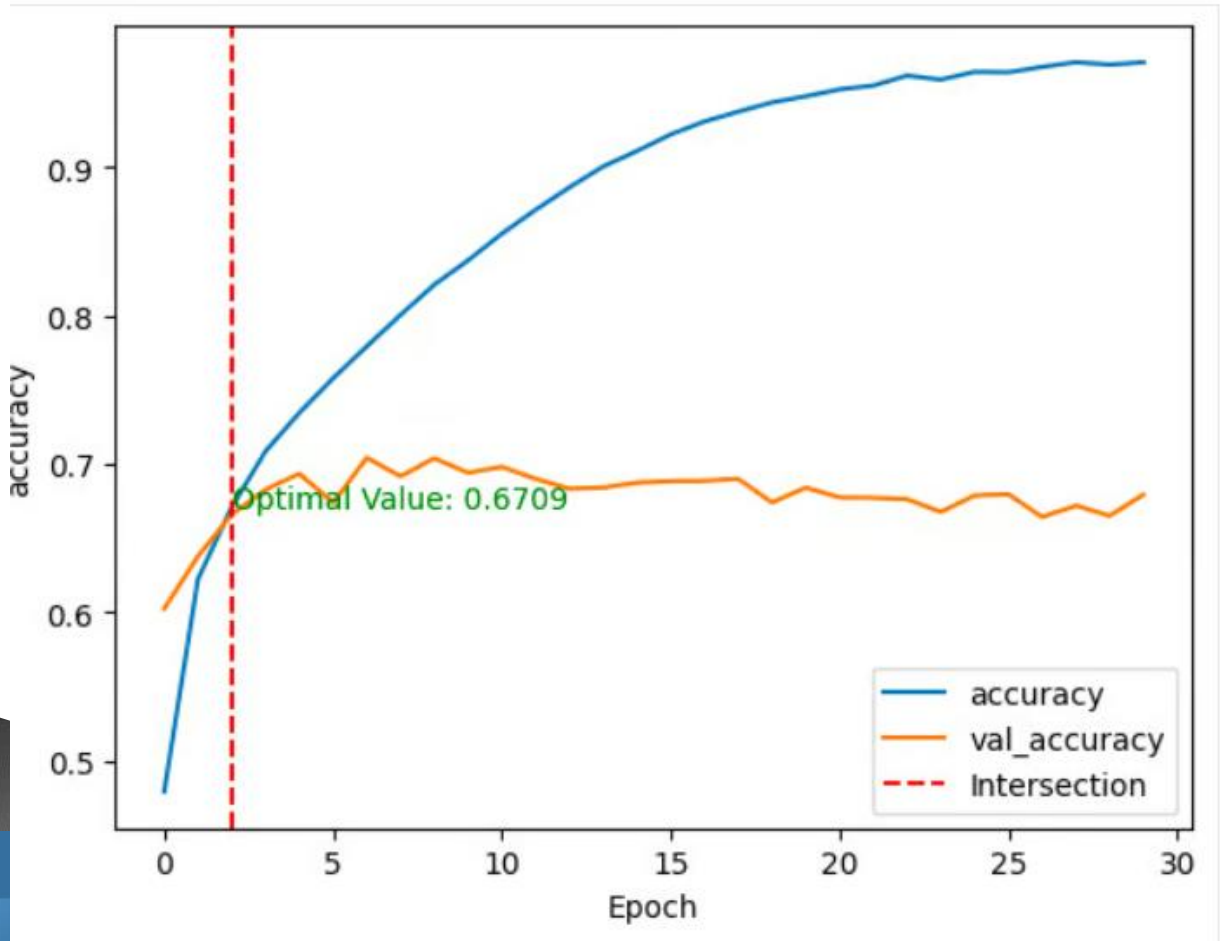
CNN - Code Example

DL | Model evaluation

After training the model without any fine-tuning and pre-processing, we end up with:

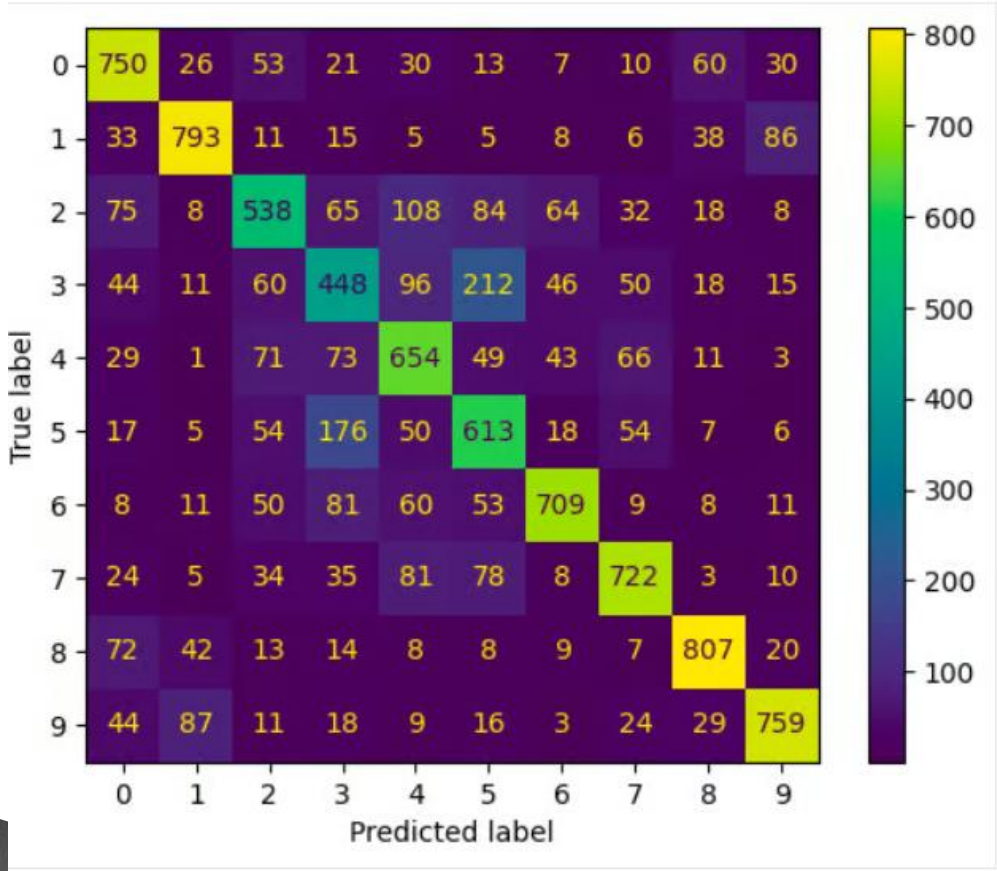
- An accuracy score of 67.09%, meaning that the model correctly classifies 67% of the samples out of every 100 samples.
- And, a precision of 76.55%, meaning that out of each 100 positive predictions, almost 77 of them are true positives, and the remaining 23 are false positives.
- These scores are achieved respectively at the third and second epochs for accuracy and precision.

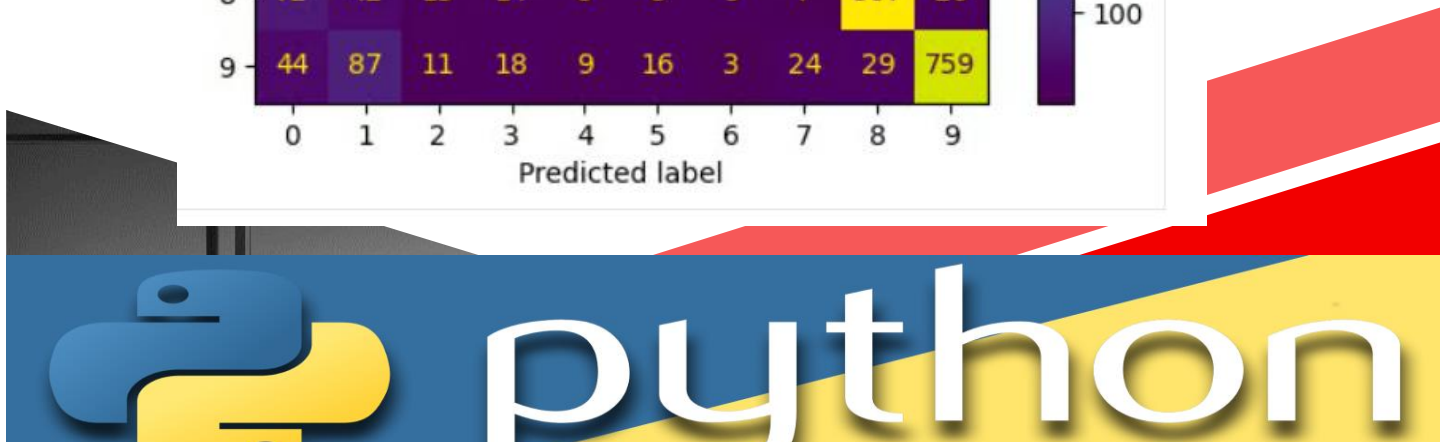
These two metrics give a global understanding of the model behavior.



DL | Confusion Matrix

- Classes 0, 1, 6, 7, 8, 9, respectively, for **airplane, automobile, frog, horse, ship,** and **truck** have the highest values at the diagonal. This means that the model is better at predicting those classes.
- On the other hand, it seems to struggle with the remaining classes:
- The classes with the highest off-diagonal values are those the model confuses the good classes with. For instance, it confuses **birds (class 2)** with an **airplane (Class 9)**, and **automobile ((Class 1)** with **trucks** (class 9).

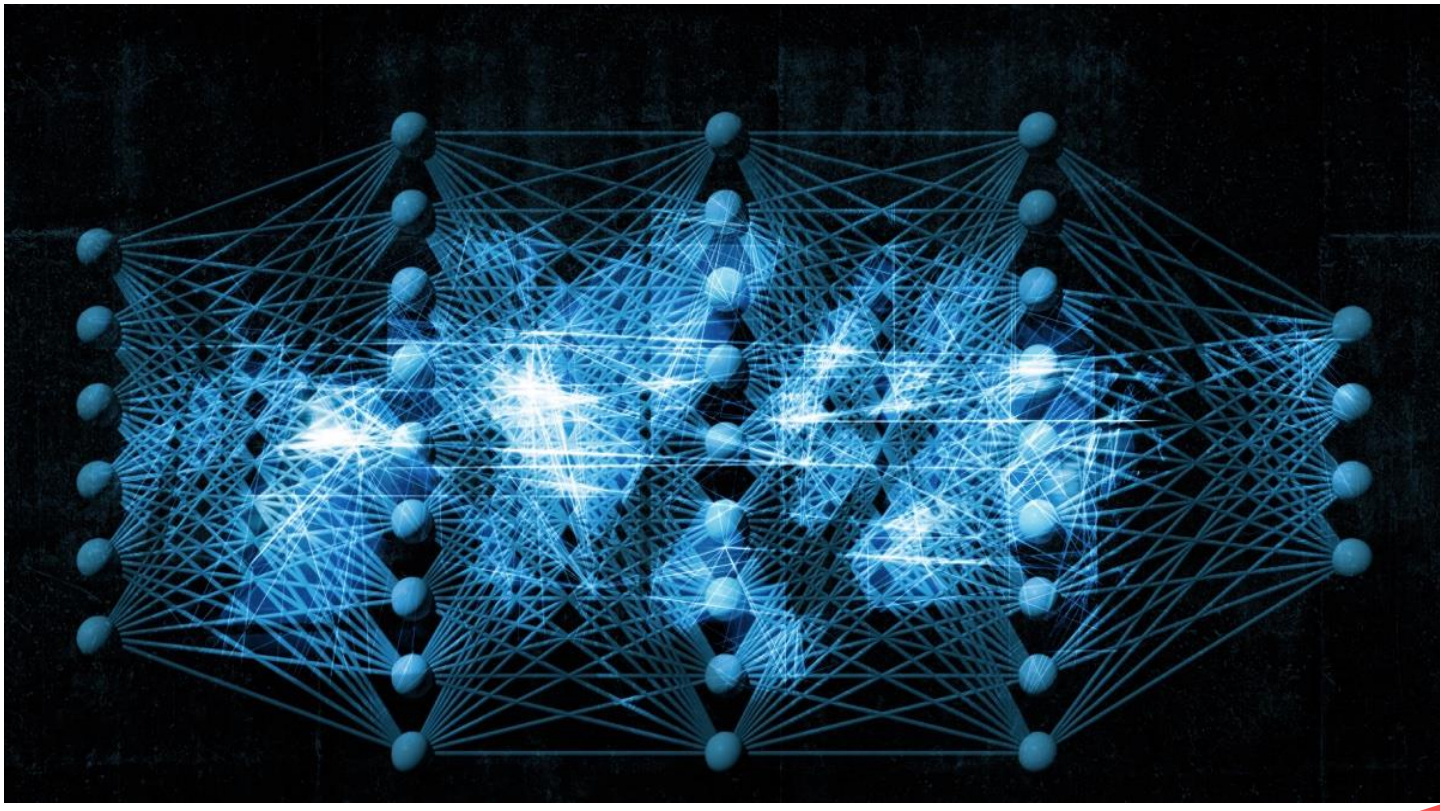




DL | Concluding Notes

This model can be improved with additional tasks such as:
Image augmentation

- ❑ Transfer learning using pre-trained models such as ResNet, MobileNet, or VGG.
- ❑ Applying different regularization technics such as L1, L2 or dropout.
- ❑ Fine-tuning different hyperparameters such as learning rate, the batch size, number of layers in the network.



DL| CNN - CIFAR-10 - IMDB – Dataset Core - Exercise

See code here: https://github.com/ShahzadSarwar10/FullStackAI-B-4-SAT-SUN-10AM-TO-12PM/blob/main/Week16/Case16-3-TF-SimpleFeedforwardNeuralNetworksExampleCode.py

https://github.com/ShahzadSarwar10/FullStackAI-B-4-SAT-SUN-10AM-TO-12PM/blob/main/Week16/Case16-4-TF-2DConvolutionalNeuralNetworks-CNN-ForImageClassification.py

https://github.com/ShahzadSarwar10/FullStackAI-B-4-SAT-SUN-10AM-TO-12PM/blob/main/Week16/Case16-5-TF-1DConvolutionalNeuralNetworks-CNN-ForTextClassification.py

You should be able to analyze – each code statement, you should be able to see trace information – at each step of debugging. "DEBUGGING IS BEST STRATEGY TO LEARN A LANAGUAGE." So debug code files, line by line, analyze the values of variable – changing at each code statement. BEST STRATEGY TO LEARN DEEP.

Let's put best efforts.
Thanks.
Shahzad – Your AI – ML Instructor

Exercises

Thank you - for listening and participating

- ❑**Questions / Queries**
- ❑**Suggestions/Recommendation**
- ❑**Ideas…..?**

Shahzad Sarwar
Cognitive Convergence
https://cognitiveconvergence.com
shahzad@cognitiveconvergence.com
voice: +1 4242530744 (USA) +92-3004762901 (Pak)