Assignment No. 3

Problem Statement:

Build the Image classification model by dividing the model into following 4 stages:

1. Loading and preprocessing the image data
2. Defining the model's architecture
3. Training the model
4. Estimating the model's performance

Objective:

1. To be able to apply deep learning algorithms to solve problems of moderate complexity
2. Understand how a model is trained and evaluated.
3. Classifying images from the image dataset.
4. Our main goal is to train a neural network (using Keras) to obtain > 90% accuracy on image dataset..
5. To apply the algorithms to a real-world problem, optimize the models learned and report on the expected accuracy that can be achieved by applying the models

Solution Expected:

Implement and train a Convulutional neural network (CNN) on an hand-written digits image dataset called MNIST and improve model generalisation by achieving increased accuracy and decreased loss where model gains good confidence with the prediction.

Methodology to be used:

- Deep Learning
- Convolutional Neural Network

Infrastructure:

Desktop/ laptop system with Linux /Ubuntu 16.04 or higher (64-bit)/ Windows OS/Mac OS

Software used:

LINUX/ Windows OS/ Virtual Machine/ IOS, Anaconda distribution, Jupyter notebook, python 3.9.12

Theory:

Deep Learning has been proved that its a very powerful tool due to its ability to handle huge amounts of data. The use of hidden layers exceeds traditional techniques, especially for pattern recognition. One of the most

popular Deep Neural Networks is Convolutional Neural Networks (CNN).

Convolutional Neural Networks (CNNs):

A convolutional neural network (CNN) is a type of Artificial Neural Network (ANN) used in image recognition and processing which is specially designed for processing data (pixels). The goal of a CNN is to learn higher-order features in the data via convolutions. They are well suited to object recognition with images and consistently top image classification competitions.

These attributes of the input match up to an image structure for which we have:

Image width in pixels

Image height in pixels

RGB channels as the depth

We can consider this structure to be a three-dimensional volume of neurons. A significant aspect to how CNNs evolved from previous feed-forward variants is how they achieved computational efficiency with new layer types.

CNN Architecture Overview:

CNNs transform the input data from the input layer through all connected layers into a set of class scores given by the output layer. There are many variations of the CNN architecture, but they are based on the pattern of layers, as demonstrated in Figure 1 (below).

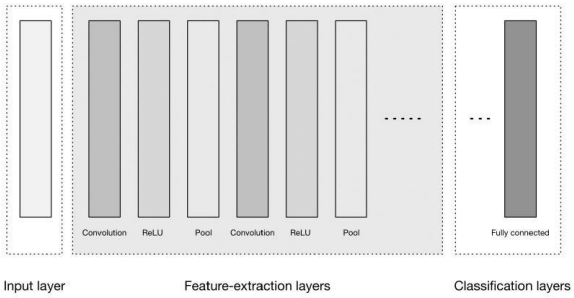


Figure 1: High-level general CNN architecture

Figure 1 depicts three major groups:

input layer

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**

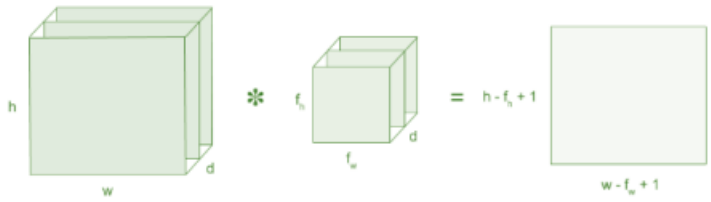


Figure 2: Image matrix multiplies kernel or filter matrix

The input layer accepts three-dimensional input generally in the form spatially of the size (width × height) of the image and has a depth representing the color channels (generally three for RGB color channels) as shown in fig 2 above.

1. Feature-extraction (learning) layers

The feature-extraction layers have a general repeating pattern of the sequence as shown in figure 1:

Convolution layer

We express the Rectified Linear Unit (ReLU) activation function as a layer in the diagram in figure 1. Convolutional layers transform the input data by using a patch of locally connecting neurons from the previous layer.

A. Pooling layer

These layers find a number of features in the images and progressively construct higher-order features. This corresponds directly to the ongoing theme in deep learning by which features are automatically learned as opposed to traditionally hand engineered.

2. Classification layers

Finally we have the classification layers in which we have one or more fully connected layers to take the higher-order features and produce class probabilities or scores. These layers are fully connected to all of the neurons in the previous layer, as their name implies. The output of these layers produces typically a two dimensional output of the dimensions [b × N], where b is the number of examples in the mini-batch and N is the number of classes we're interested in scoring.

Multilayer neural networks vs CNN:

In traditional multilayer neural networks, the layers are fully connected and every neuron in a layer is connected to every neuron in the next layer whereas The neurons in the layers of a CNN are arranged in three dimensions to match the input volumes. Here, depth means the third dimension of the activation volume, not the number of layers, as in a multilayer neural network.

Evolution of the connections between layers:

Another change is how we connect layers in a convolutional architecture. Neurons in a layer are connected to only a small region of neurons in the layer before it. CNNs retain a layer-oriented architecture, as in traditional multilayer networks, but have different types of layers. Each layer transforms the 3D input volume from the previous layer into a 3D output volume of neuron activations with some differentiable function that might or might not have parameters, as demonstrated in Figure 1.

Input Layers

Input layers are where we load and store the raw input data of the image for processing in the network. This input data specifies the width, height, and number of channels. Typically, the number of channels is three, for the RGB values for each pixel.

Convolutional Layers

Convolutional layers are considered the core building blocks of CNN architectures. As Figure 2 illustrates, convolutional layers transform the input data by using a patch of locally connecting neurons from the previous layer. The layer will compute a dot product between the region of the neurons in the input layer and the weights to which they are locally connected in the output layer.

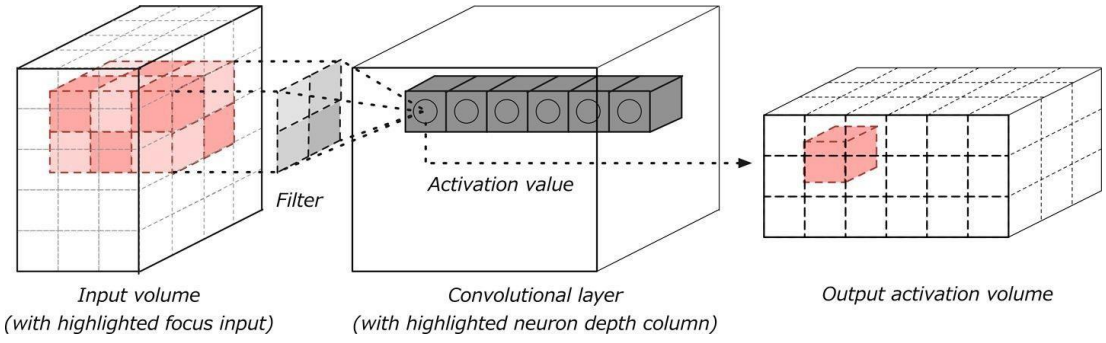


Figure 3: Convolution layer with input and output volumes

The resulting output generally has the same spatial dimensions (or smaller spatial dimensions) but sometimes increases the number of elements in the third dimension of the output (depth dimension).

TensorFlow is an open-source platform for machine learning. Keras is the high-level application programming interface (API) of TensorFlow. Using Keras, we can rapidly develop a prototype system and test it out. This is the first in a three-part series on using TensorFlow for supervised classification tasks.

STEPS: To implement Convolutional Neural Network for image classification

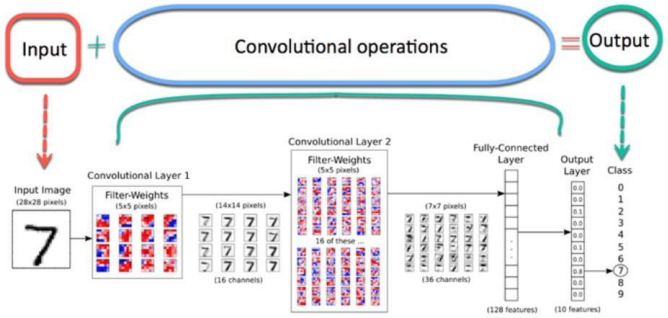


Figure 4: Image Classification model using CNN with python

CNN:

Now imagine there is an image of a bird, and you want to identify it whether it is really a bird or something other. The first thing you should do is feed the pixels of the image in the form of arrays to the input layer of the neural network (MLP networks used to classify such things). The hidden layers carry Feature Extraction by performing various calculations and operations. There are multiple hidden layers like the convolution, the ReLU, and the pooling layer that performs feature extraction from your image. So finally, there is a fully connected layer that you can see which identifies the exact object in the image. You can understand very easily from the following figure:

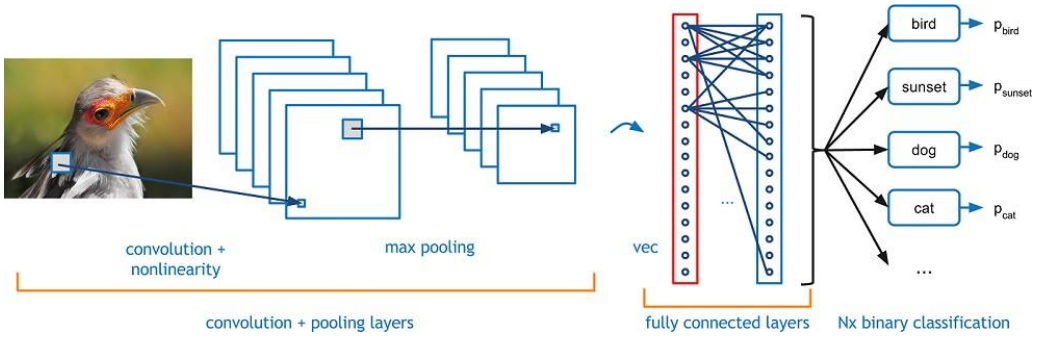


Figure 5: Convolution and pooling layers in CNN

Convolution:

Convolution Operation involves matrix arithmetic operations and every image is represented in the form of an array of values(pixels).

Let us understand example:

a = [2,5,8,4,7,9]

b = [1,2,3]

In Convolution Operation, the arrays are multiplied one by one element-wise, and the product is grouped or summed to create a new array that represents a*b.

The first three elements of matrix a are now multiplied by the elements of matrix b. The product is summed to get the result and stored in a new array of a*b.

This process remains continuous until the operation gets completed.

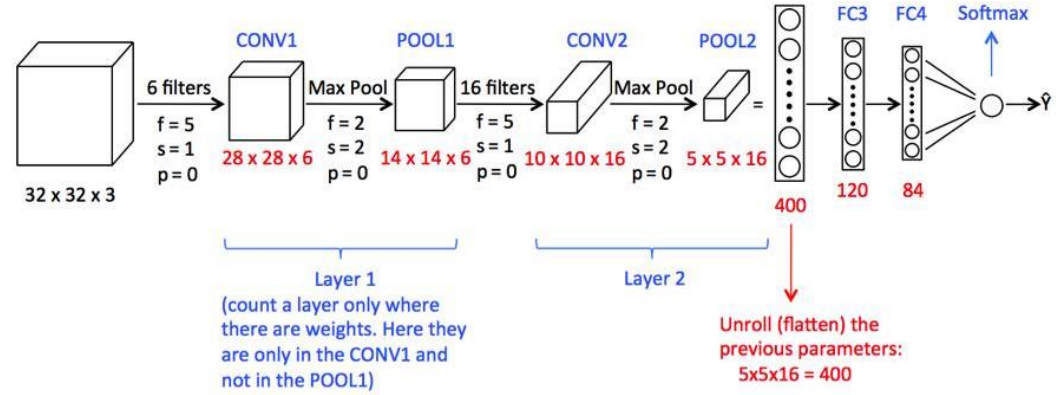
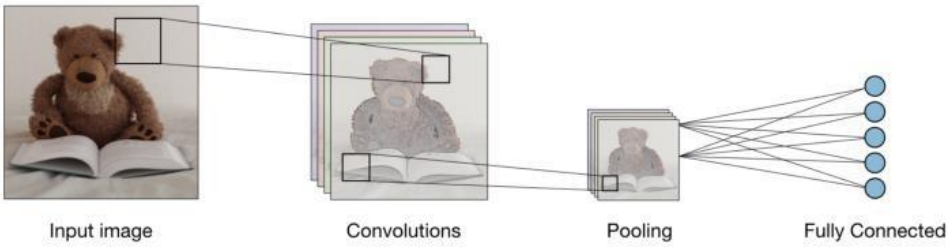


Figure 6: Sequence of Convolution and pooling layers in CNN

Pooling:

After the convolution, there is another operation called pooling. So, in the chain, convolution and pooling are applied sequentially on the data in the interest of extracting some features from the data. After the sequential convolutional and pooling layers, the data is flattened into a feed-forward neural network which is also called a Multi-Layer Perceptron.



igure 7: Data flattening into a feed-forward neural network

Conclusion:

Thus, we have implemented the Image classification model using CNN. With above code we can see that sufficient accuracy has been met. Throughout the epochs, our model accuracy increases and loss decreases that is good since our model gains confidence with our prediction

This indicates the model is trained in a good way

1. The loss is decreasing and the accuracy is increasing with every epoch.
2. The test accuracy is the measure of how good the model is predicting so, it is observed that the model is well trained after 10 epoch