Deep Learning for Computer Vision and NLP (COMP 8920-02)

Professor Name : Dr Robin Gras

Team : Sanju Atwal (Student id: 40152236)

Margaret Arulmalar Rebeka Nesaraj (Student id:110026527)

Contents

What is MNIST?

The MNIST is a database of handwritten digits, it has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalised and centred in a fixed-size image.

Fig 1 a: Mnist dataset

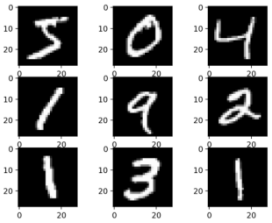


Fig 1 b: Size normalized data

Training and Validation Sets:

Out of the 60000 examples in MNIST database which are reserved for training, we keep 55000 samples for training and 5000 samples for validation. This validation set is used to get an accurate estimate of the model's performance while the model is under training.

Testing Set:

10000 MNIST samples are used for testing purpose.

Objective of this project:

This project aims at implementing a CNN architecture, which will take the image as an Input and gives us the class in which the image belongs as an Output. For instance, if an image of handwritten digit 5 is given as an input, the output should be class 5. We will also compare the different CNN architectures, to analyse the model which gives best performance based on the various metrics such as Accuracy, Precision and the like.

What is CNN ?

Convolutional Neural Networks or ConvNet or CNN basically takes the input image and assigns weights and biases to specific aspects of the image, using a Deep Learning Algorithm. This in turn would help the network to differentiate the various images from one another. Some filters are applied on the input image in order to extract the useful features from an image dataset. While, in other object detection techniques, these filters are hand engineered, whereas the CNN's have the ability to automatically learn these filters with enough training.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond

to stimuli only in a restricted region of the visual field known as the Receptive Field (refer Fig 2 below). A collection of such fields overlaps to cover the entire visual area.

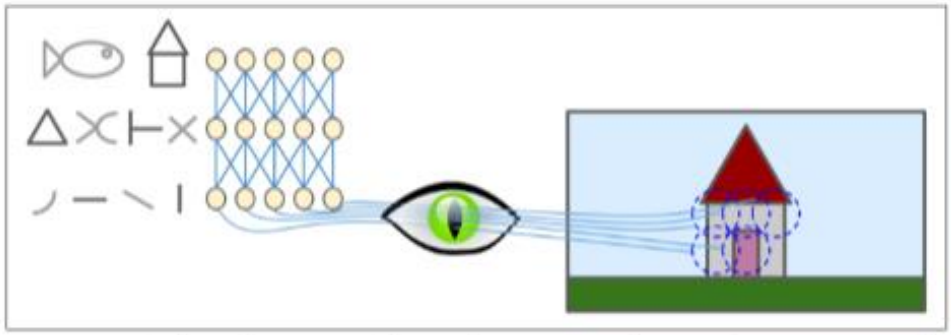


Fig : 2 Representation of neurons in a Visual cortex

Structure of CNN:

The CNN architecture implemented in our model comprises of the following components:

(i) Input Image (MNIST dataset)
(ii) Convolutional layer (Conv2D)
(iii) Max Pooling layer (MaxPool2D)
(iv) Dense layer (Fully connected Neural Network)
(v) Dropout Layer
(vi) Flatten Layer

Input Image (MNIST dataset)

The input contains the Grey scale images of handwritten digits (MNIST) which is of size 28*28*1 (784 pixels). Along with each image, a label is also provided to the model, this includes the class which the image belongs to

Convolutional layer (Conv2D)

This layer performs the convolution operation on the input image. The convolution layer implemented comprises of the following elements:

- Filter (Kernel) : It is a metrics which when applied on the image, gives the resultant feature map as the output. The feature map varies based on the filters applied. Multiple filters are applied to the image dataset in our model implementation
- Stride: It represents the number of pixels shifts over the input matrix. When the **stride** is 1 then we move the filters to 1 pixel at a time. When the **stride** is 2 then we move the filters to 2 pixels at a time and so on.

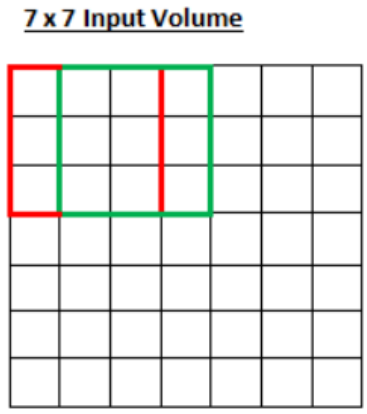


Fig 3 a: Kernel with Stride 1

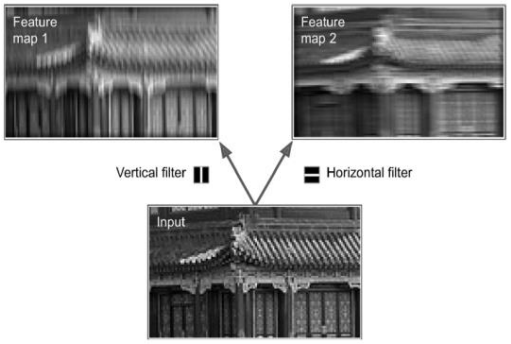


Fig 3 b: Feature map extraction

Max Pooling layer (MaxPool2D)

In this layer, the max input value in each receptive field makes it to the next layer, while the other inputs are dropped. Multiple Max pooling layers of varied sizes is implemented in our model

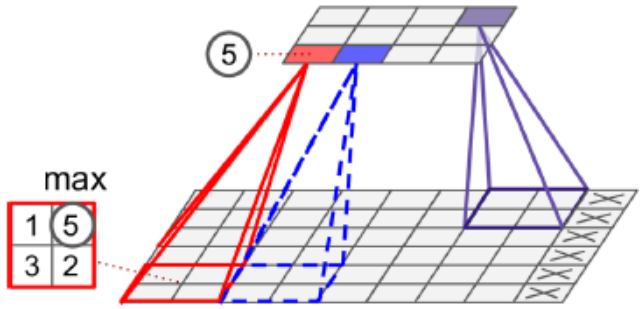


Fig 4 : Max pooling layer

Dense layer (Fully connected Neural Network)

A Dense layer consists of neurons which are connected to its input layer as well as the output layer. In a fully connected neural network, every neuron in a layer is connected to every neuron in the previous layer as well the following layer. Multiple dense layers have been used in this model

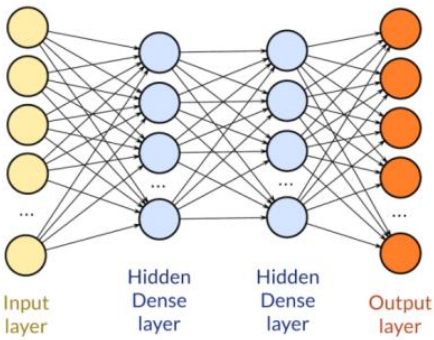


Fig 5: Fully connected CNN

Dropout Layer

A dropout layer is used for regularization. In this layer, every neuron (including the input neurons, but always excluding the output neurons) from the model is switched off randomly at different iterations, to make it more robust. This technique has been implemented neuron is able to learn more features. Neurons trained with dropout cannot co-adapt with their neighbouring neurons; they have to be as useful as possible on their own. They also do not rely excessively on just a few input neurons; they must pay attention to each of their input neurons. It is observed that they end up being less sensitive to slight changes in the inputs. Resultant outcome is a more robust network that generalizes better.

Flatten Layer

This layer in the model is responsible for flattening the array of images into one dimension.

Optimizers

Optimizers are algorithms or methods that change the attributes of our neural network such as weights and learning rate in order to reduce the losses and to provide the most accurate result possible.

There are different optimizers which are commonly used such as Gradient Descend, Stochastic Gradient Descend, Momentum, Adam, Nadam and many more. However, for this project, we have used Nadam. Nadam is a combination of Nesterov Accelerated Gradients and Adam. We have used Nadam because it outperforms other optimizers including Adam as it is more accurate.

Loss

The neural network intends to minimise the error by using an objective function referred as cost function or a loss function. The value calculated by this function is referred as simply “loss “.The sparse categorical cross entropy is used as our loss function, since we are using integer labels (not one hot encoded)in our model.

CNN Architecture: Final model

Our Sequential CNN model comprises of 8 layers and includes the following components as mentioned below

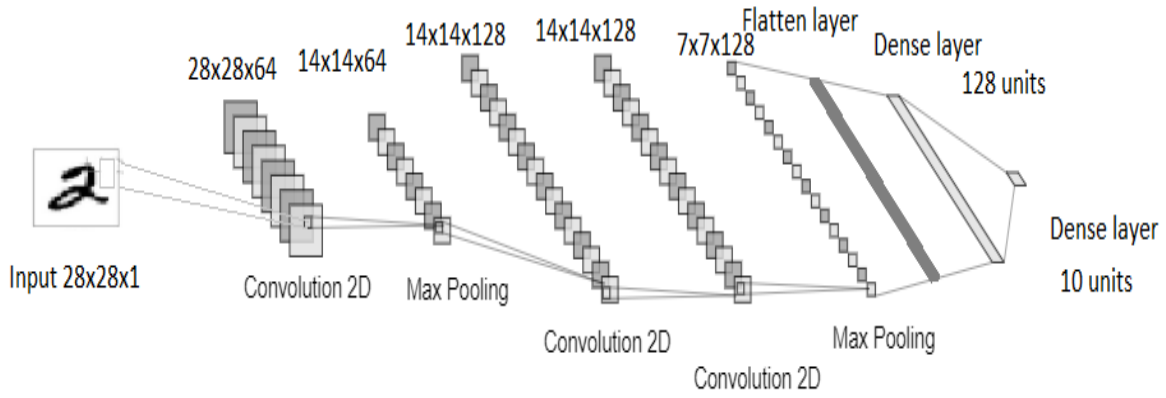


Fig 6 : Sequential CNN architecture

INPUT Layer: An MNIST dataset of 60000 samples and size 28*28*1

CNN Layer 1 (Conv2D): A Conv2D layer with 64 filters, Kernel size 7 , 'SAME' padding , 'relu' activation ,input 28*28*1

CNN Layer 2(Max Pooling): A Max Polling layer with pool size of 2.

CNN Layer 3(Conv2D): A Conv2D layer with 128 filters, Kernel size 3 , 'SAME' padding , 'relu' activation

CNN Layer 4(Conv2D): A Conv2D layer with 128 filters, Kernel size 3 , 'SAME' padding , 'relu' activation

CNN Layer 5(Conv2D): A Max Pooling layer with pool size of 2.

CNN Layer 6(Flatten): A layer to flatten the array and feed it to fully connected network

CNN Layer 7(Dense):A dense layer with 128 units and 'relu' activation function.

CNN Layer 8(Dense): A dense layer with 10 units and 'softmax' activation, this is the final output layer.

Transformation of CNN layers:

CNN Layer 1 transforms the input layer of size 28*28*1 into an output of shape 28*28*64 (ie 64 feature maps) by applying 64 filters of Kernel size 7 with same padding and relu activation.

CNN Layer 2 Produces 64, 14x 14 feature map, which highlights the most present feature in the patch by applying Max Pooling of size 2.

CNN Layer 3 transforms the input from the previous layer into 14*14*128 output after the application of 128 filters of Kernel size 3 with same padding and relu activation.

CNN Layer 4 transforms the input from the layer ahead into 14*14*128 output after the application of 128 filters of Kernel size 3 with relu activation.

CNN Layer 5 works with the input from the preceding layer to produce 128, 7x 7 feature maps. This highlights the most present feature in the patch by applying Max Pooling of size 2.

CNN Layer 6 receives the 7*7*128 input from the previous layer and flattens it into a single dimensional array of size 6272 (ie 7x7x128=6272)

CNN Layer 7 this fully connected neural network layer (dense layer) containing 128 units and a relu activation, processes the input of the previous layer and transforms into and output of shape 128.

CNN Layer 8 this final CNN layer (dense layer) comprising 10 units and a Softmax classifier , extracts values from the above layer and converts it into an output layer of size 128.The resultant provides the is the class label to which the MNIST image belongs to .

Significance of Activation functions applied on the model:

Relu activation: This activation function allows the models to learn faster and perform better as it overcomes the vanishing gradient problem. The significance of Relu function over other activation functions is that it does not activate all the neurons at the same time

Softmax activation: Produces an output which lies in the range [0, 1] and the total value from all the output neurons sums up to 1. This function is used to classify the probability to which the image belongs

Model Compilation:

The following components are used in compilation of the model

- **Nadam optimizer:**
  Nadam is an optimization algorithm with improved convergence speed. Our model utilises the momentum technique used in this algorithm to get the information from recent past and apply it on determining the current step.

  Significance: Increases the efficiency of learning rate of the model

- **Sparse categorical cross entropy loss:**
  In Sparse categorical cross entropy in which the labels are not one-hot encoded, instead they use integer labels .This model takes advantage of the integer labels as the MNIST data set contains the classes with integer labels that are mutually exclusive

  Significance: Uses a single integer as class label rather than a vector, hence it saves time and computational memory.

- **Accuracy metrics :**
  The Accuracy metrics is beneficial in determining the ratio of correct predictions on the balanced dataset. As the MNIST dataset has equal number of sample for each of its class, this metrics is highly beneficial in determining the training versus validation accuracy and loss of the model

  Significance: Helps in identifying the true positive and true negative rates with greater accuracy on a balanced dataset

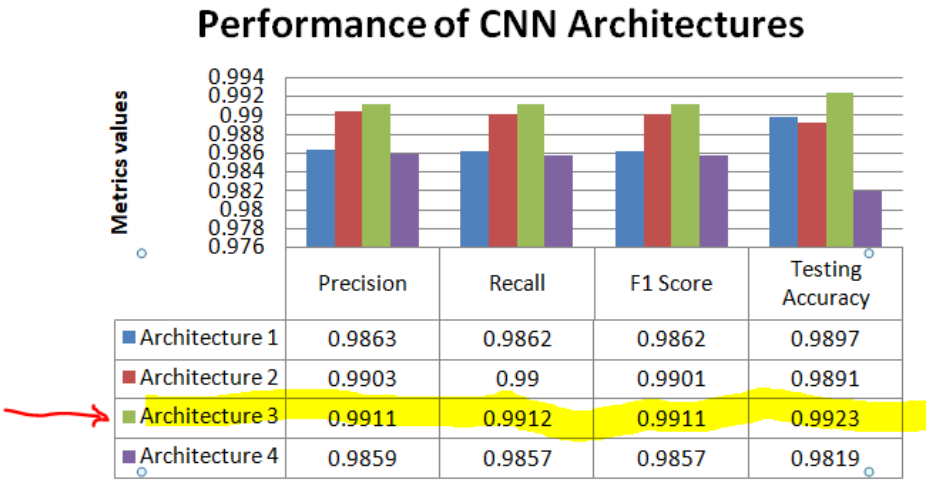
Performance analysis for CNN architectures:

Architecture 3 outperforms Architecture 1 and 2, as in the latter the model is over fitting as MNIST is a fairly simple task for CNN. This leads to a high accuracy on the training samples; however the accuracy on testing set is approximately 1 % lower than that of the training samples. Over fitting leads to false positive results which leads to detrimental effects.

To prevent over fitting issues which arose in Architecture 1 and 2, we have to simplify the model by removing two convolution layers, max pooling layer and a dense layer .It is also observed that the drop out layer doesn't add much value to the model and so it has not been implemented in this model

Architecture 4 is not of much value add as it scores the least with respect to its accuracy and performance .The Tanh activation function leads to vanishing gradient problem , hence when it is incorporated in this architecture it is accounts to the poor performance of this model . The problem of vanishing gradient is resolved in Architecture 3 by the usage of relu activation function which gives a positive output and hence Architecture 3 outperforms all the other architectures in terms of efficiency and accuracy

The bar graph below represents a statistical analysis of different CNN architectures which are used to classify the MNIST samples . From the below data ,it is evident that architecture 3 has the best case performance for all the metrics variants namely Precision,Recall,F1 Score and Testing Accuracy as opposed to the same metrics values of all the other architectures(ie Architeture 1,2,4). The precision , recall , F1 score and Testing accuracy of Architecture 3 are 0.9911,0.9912 ,0.9911 and 0.9923 respectively . While the metrics of the least performed Architecture 4 ranges between 0.9819 to 0.9857. On the other hand architecutre 1 and 2 has an average metrics range which is lesser than that of the Architecture 3.



	Precision	Recall	F1 Score	Testing Accuracy
Architecture 1	0.9863	0.9862	0.9862	0.9897
Architecture 2	0.9903	0.99	0.9901	0.9891
Architecture 3	0.9911	0.9912	0.9911	0.9923
Architecture 4	0.9859	0.9857	0.9857	0.9819

Fig 7: Performance analysis of different CNN architectures based on various metrics