University of Windsor

TOPICS IN ARTIFICIAL INTELLIGENCE (COMP - 8720)

PROJECT 1 - CNN IMPLEMENTATION

Professor Name: Dr Robin Gras

Team:

Akshaya Muthuraman (Student ID: 110070509)

Ameer Asif Khader Batcha (Student ID: 110076465)

Paridhi Dilipbhai Gondalia (Student ID: 110071190)

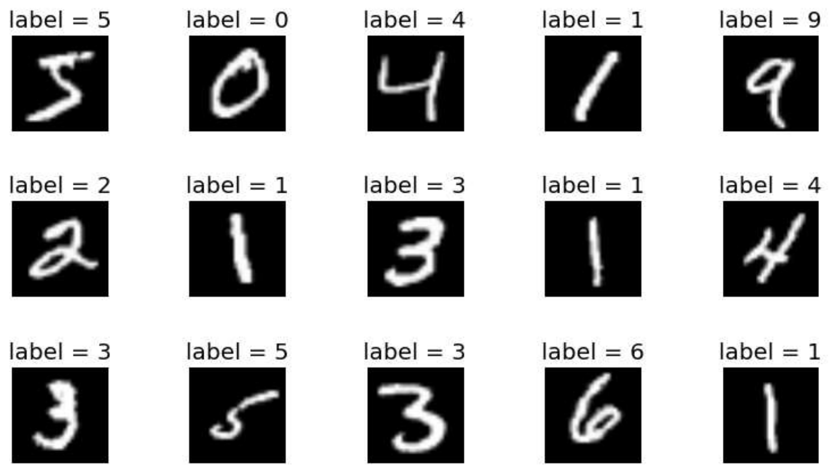
TABLE OF CONTENTS

1. ABSTRACT

This project intends to compare various simple CNN architectures for the MNIST dataset. This CNN takes in an image from the dataset as input and recognizes the handwritten digit in the image. The number of output classes of this CNN is 10 where each class is a number ranging from 0-9. The output of the CNN will be any one of these classes. The comparison is done by varying architectures and optimizers.

2. MNIST DATASET

The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.



The training set is divided into 55,000 examples for training and 5,000 examples for validation.

Operation	Number of training examples
Training	55,000
Validation	5,000
Testing	10,000

3. STRUCTURE OF CNN

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. The various components of a CNN are listed below.

3.1 INPUT IMAGE

The input image is fed from the MNIST dataset which consists of grayscale images of handwritten digits of size 28x28x1 and each of these images has an output label corresponding to the output class it belongs to (0-9).

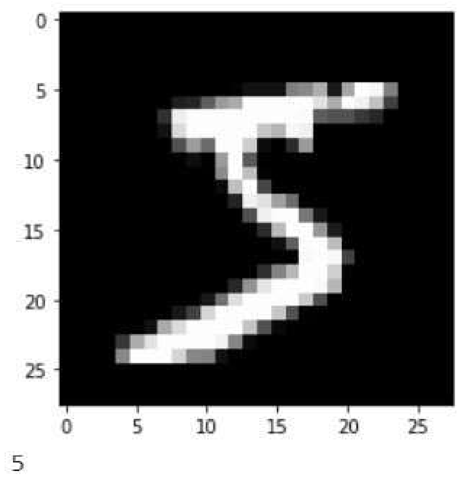


Fig 1: MNIST Dataset Input Image Example

3.2 CONVOLUTIONAL LAYER

In a CNN, the input is a tensor with a shape: (number of inputs) x (input height) x (input width) x (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape:

(number of inputs) x (feature map height) x (feature map width) x (feature map channels). Filters or kernels are the matrices representing features that are to be extracted from the image. A convolution operation means performing a dot product of the convolution kernel with the layer's input matrix. There may be multiple filters applied to an image. The rate at which the filter moves across the image is called stride.

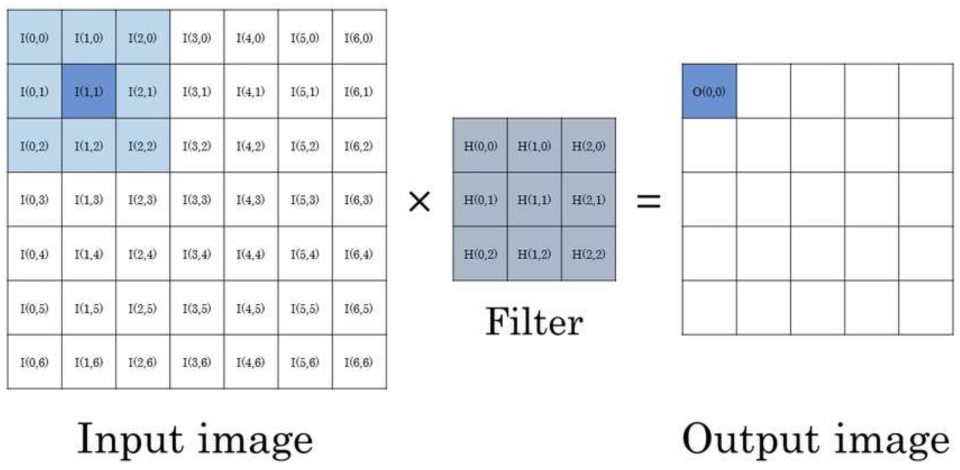


Fig 1: Convolution with stride 1 Example

3.3 POOLING LAYER

Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. There are two common types of pooling in popular use: max and average. *Max pooling* uses the maximum value of each local cluster of neurons in the feature map,while *average pooling* takes the average value. In this project we use Max Pooling for all architectures.

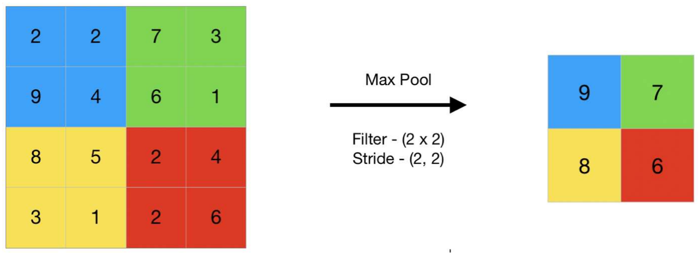


Fig 3: Max Pooling Example

3.4 FULLY CONNECTED LAYER

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

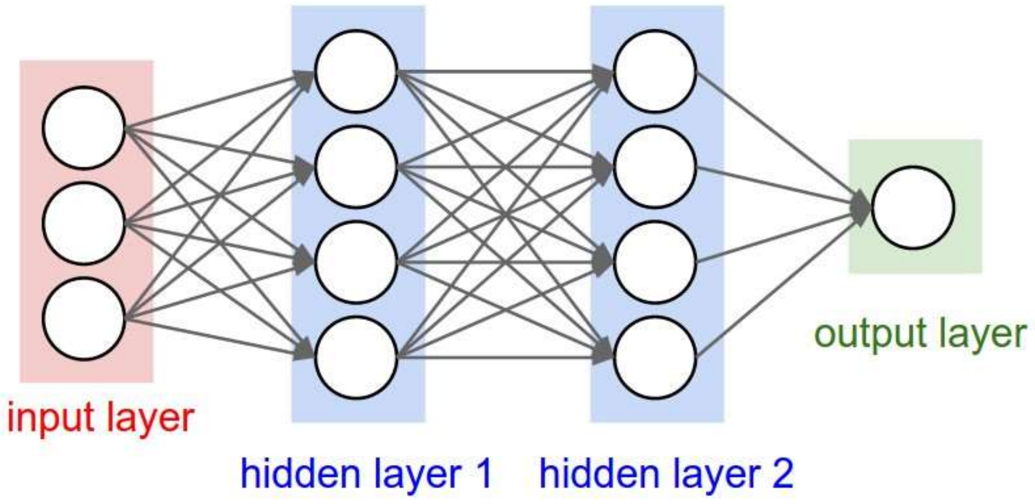


Fig 4: Fully Connected Layer Example

3.5 FLATTEN LAYER

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

4. OPTIMIZERS

Loss tells us how poorly the model is performing at that current instant. This loss is needed to train our network such that it performs better. Hence, we take the loss and try to minimize it, because a lower loss means the model will perform better. The process of minimizing (or maximizing) any mathematical expression is called optimization. Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses. Optimizers are used to solve optimization problems by minimizing the function.

In this project we use various optimizers like **Adam**, **RMSProp, Stochastic Gradient Descent (SGD)** and **Nadam** which is a combination of Nesterov Accelerated Gradient (NAG) and Adam.

5. LOSS

The loss function in a neural network quantifies the difference between the expected outcome and the outcome produced by the machine learning model. From the loss function, we can derive the gradients which are used to update the weights. The average over all losses constitutes the cost.

In this project, **Categorical Cross Entropy** is used as our loss function, since we are using one hot encoded output label. The MNIST dataset contains output labels as integer labels(0-9). If these integer labels are used directly, then Sparse Categorical Cross Entropy can be used as the loss function.

6. CNN ARCHITECTURES

	Architecture 1	Architecture 2	Architecture 3
Layer 1	**Conv2D**	**Conv2D**	**Conv2D**
	Filters - 5	Filters - 32	Filters - 64
	Kernel Size - 7	Kernel Size - 3	Kernel Size - 7
Layer 2	**MaxPool**	**MaxPool**	**MaxPool**
	Pool Size - 2	Pool Size - 2	Pool Size - 2
Layer 3	**Conv2D**	**Conv2D**	**Conv2D**
	Filters - 16	Filters - 64	Filters - 128
	Kernel Size - 5	Kernel Size - 3	Kernel Size - 3
Layer 4	**Flatten**	**MaxPool**	**Conv2D**
		Pool Size - 2	Filters - 128
			Kernel Size - 3

Layer 5	**Dense**	**Conv2D**	**MaxPool**
	Units - 120	Filters - 64	Pool Size - 2
		Kernel Size - 3	
Layer 6	**Dense**	**Flatten**	**Flatten**
	Units - 10		
Layer 7	-	**Dense**	**Dense**
		Units - 64	Units - 128
Layer 8	-	**Dense**	**Dense**
		Units - 10	Units - 10
Optimizer	Adam	SGD & RMSProp	Nadam

All Conv2D operations use **‘SAME’** padding and **‘Relu’** activation functions. Relu is used to solve the vanishing gradient problem of tanh activation function.

For the models, the accuracy metric is calculated. This is the ratio of correct predictions on the balanced dataset. As the MNIST dataset has an equal number of samples for each of its classes, this metric is highly beneficial in determining the training versus validation accuracy and loss of the model.

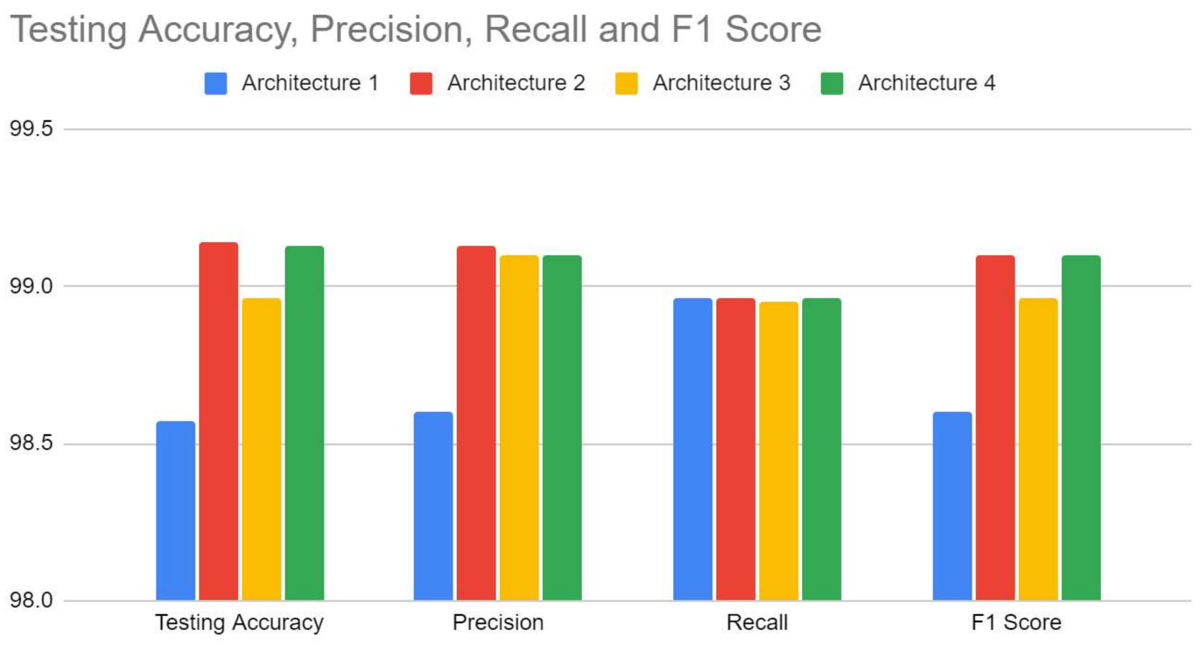
7. PERFORMANCE ANALYSIS FOR CNN ARCHITECTURES

	Architecture 1	Architecture 2	Architecture 3	Architecture 4
Testing Accuracy	98.57	99.14	98.96	99.13
Precision	0.986	0.9913	0.9896	0.991
Recall	0.986	0.991	0.9895	0.991
F1 Score	0.986	0.991	0.9896	0.991

From these results we can see that architecture 2 & 4 performs the best. The architectures are similar with only a change in the optimization function. Architecture 2 uses Stochastic Gradient Descent (SGD) whereas Architecture 4 uses RMSProp. Although RMSProp performed slightly lower than SGD over the validation set, the results are quite similar over the test set.

Architecture 3 performed better over the training and validation set than the test set. This may be a problem of overfitting which leads to higher accuracy over the training set than the test set.

Architecture 1 is inspired by Lenet Architecture. It has fewer layers as compared to other architectures due to which it has lower accuracy than others.



Performance of CNN architectures

The bar graph above represents a statistical analysis of different CNN architectures which are used to classify the MNIST samples. Various performance metrics like testing accuracy, precision, recall and F1 score is calculated for these architectures. From the graph we can see that Architecture 2 & 4 performs the best. Both architectures are the same but Architecture 2 uses SGD as its optimizer and Architecture 4 uses RMSProp as its optimizer.