**DIGIT RECOGNITION**

**Introduction:**

A common dataset for computer vision and deep learning is the MNIST handwritten digit classification issue. The dataset can be used as a starting point to create, assess, and apply convolutional deep learning neural networks for image classification. The MNIST dataset is an acronym that stands for the Modified National Institute of Standards and Technology dataset.

It is a dataset of 60,000 small square 28×28 pixels grayscale images of handwritten single digits between 0 and 9. The task is to classify a given image of a handwritten digit into one of 10 classes representing integer values from 0 to 9, inclusively.

**Problem Statement:**

we need to identify digits from the given dataset of handwritten images. Like we have different strokes such as italic and various types. These handwritten images are captured through pictures. And we need to train and test these images to predict the digits correctly or not.

**DATASET:**

The MNIST dataset contains a total of 70,000 images where 60,000 images are for training and 10,000 images for testing. Each image is of 28\*28 pixels with grayscale intensity.

We have two files TRain.csv and test.csv are grayscale images.

Suppose if any image is 28\*28=786, we have 786 pixels train data has 785 columns

The first column is called label that was drawn by the user. The rest of the columns contain the pixel-values of the associated image.

**PROJECT WORKFLOW:**

Diagram

Description automatically generated

**Techniques:**

**CNN**: A spectacular advancement that combines Artificial Neural Networks (ANN) and modern deep learning techniques is the Convolutional Neural Network (CNN). In a wide range of applications, including pattern recognition, sentence classification, speech recognition, face identification, text categorization, document analysis, scene recognition, and handwritten digit recognition, it has been used.

The goal of this project is to compare the classification accuracy of CNN across a range of hidden layers and epoch numbers, and to observe the variance in classification accuracy. We ran an experiment using the Modified National Institute of Standards and Technology (MNIST) dataset to assess how well CNN performed. Additionally, the backpropagation technique and stochastic gradient descent are used to train the network.

Each layer of the CNN has several neurons. The input of a neuron in the following layer receives the weighted total of all the neurons from that layer and adds a biased value. The layer in CNN has three dimensions. Not every neuron in this region is linked entirely.

The local receptive field is instead connected to each neuron in the layer. The network is trained using a cost function. It contrasts the network's output with the desired output. The signal returns to the system repeatedly to update the shared weights and biases across all receptive fields in order to reduce the cost function value and improve network performance.

Convolutional Neural Network (CNN) is widely used in image classification, video analysis, etc. because to its high accuracy. Sentence sentiment recognition is a goal of many researchers.

A seven-layered convolutional neural network with one input layer, five hidden layers, and one output layer is created to recognize the handwritten numbers.

Diagram

Description automatically generated

The network's input layer is made up of 28 by 28-pixel pictures, which translates to 784 neurons.

The input pixels are grayscale, where a white pixel has a value of 0 and a black pixel has a value of 1. Five hidden layers make up this CNN model. Convolution layer 1 is the first hidden layer, and it is in charge of extracting features from input data. By combining a filter from the preceding layer with a convolution operation, this layer performs convolution to small, localized areas. Additionally, it includes numerous feature maps with rectified linear units and learnable kernels (ReLU).

The location of the filters is determined by the kernel size. In ReLU, utilized to improve the performance of the model as a fully linked layer and as an activation function at the conclusion of each convolution layer. The pooling layer 1 is the following hidden layer. In addition to reducing the number of parameters and computational complexity in the model, it also reduces the output data from the convolution layer. Max pooling, Min pooling, Average pooling, and L2 pooling are the various types of pooling. The dimension of each feature map is subsampled in this case using max pooling.

With the exception of different feature maps and kernel sizes, convolution layer 2 and pooling layer 2 perform the same task as convolution layer 1 and pooling layer 1 and carry out the same operations. Following the pooling layer, a flattened layer is employed to transform the 2D feature map matrix into a 1D feature vector and enable the output to be handled by the fully connected layers. Another hidden layer known as the dense layer is a fully connected layer. Like the hidden layer in artificial neural networks (ANNs), it connects every neuron from the previous layer to the next layer in this case, but it is fully connected.

At fully linked layer 1, the dropout regularization method is employed to lessen overfitting. In order to strengthen the network and increase performance, it randomly turns off some neurons during training. As a result, the network can generalize better and is less likely to overfit the training set of data. Ten neurons make up the network's output layer, which determines the digits 0 through 9. Classifies the output digit from 0 to 9 with the highest activation value because the output layer utilizes an activation function like SoftMax to improve the performance of the model.

**BATCH NORMALIZATION:**

Another approach that can rapidly accelerate the learning of a model and can result in large performance improvements is batch normalization. We will evaluate the effect that batch normalization has on our baseline model.

Batch normalization can be used after convolutional and fully connected layers. It has the effect of changing the distribution of the output of the layer, specifically by standardizing the outputs. This has the effect of stabilizing and accelerating the learning process.

BN helps to fine tune hyperparameters better and train really deep neural networks.

**CON2D:**

The first is the convolutional (Conv2D) layer. It is like a set of learnable filters. I chose to set 32 filters for the two firsts conv2D layers and 64 filters for the two last ones. Each filter transforms a part of the image (defined by the kernel size) using the kernel filter. The kernel filter matrix is applied on the whole image. Filters can be seen as a transformation of the image.

CNN can isolate features that are useful everywhere from these transformed images (feature maps).

**MAXPOOLING:**

The second important layer in CNN is the pooling (MaxPool2D) layer. This layer simply acts as a down sampling filter. It looks at the 2 neighboring pixels and picks the maximal value. These are used to reduce computational cost, and to some extent also reduce overfitting. We have to choose the pooling size (i.e the area size pooled each time) more the pooling dimension is high, more the down sampling is important.

Combining convolutional and pooling layers, CNN are able to combine local features and learn more global features of the image.

DROPOUT:

Dropout is a regularization method, where a proportion of nodes in the layer are randomly ignored (setting their weights to zero) for each training sample. This drops randomly a proportion of the network and forces the network to learn features in a distributed way. This technique also improves generalization and reduces overfitting.

'relu' is the rectifier activation function max.The rectifier activation function is used to add nonlinearity to the network.

The Flatten layer is used to convert the final feature maps into single 1D vector. This flattening step is needed so that you can make use of fully connected layers after some convolutional/Maxpooling layers. It combines all the found local features of the previous convolutional layers.

DATA AUGMENTATION: It is a technique of showing slightly different or new images to neural network to avoid overfitting. And to achieve better generalization. In case when we have very small dataset, we can use different kinds of data augmentation techniques to increase your data size. Neural networks perform better if we provide them with more data. One of the benefits of data augmentation is it acts as a regularizer and helps to reduce overfitting when training a model. This is because with more artificially generated images, the model is unable to overfit to specific examples and is forced to generalize, thus the model becomes more robust. This generally leads to a better overall performance.

Different data augmentation techniques are as follows:

1. Cropping
2. Rotating
3. Scaling
4. Translating
5. Flipping
6. Adding Gaussian noise to input images etc.

PCA (principal component analysis)- PCA is **used to lower the dimensionality.**Higher dimensional value to lower value (dimensionality Reduction ) . Here we reduce 786 dimensions to 2 dimensions.

PCA is first applied to the two datasets to achieve dimensionality reduction. The compressed datasets are used to train the 2D-CNN and 3D-CNN models. The trained models are then used to classify the test samples.

**DROPOUTS:**

Even though the test data is taken from the same distribution as the training set, many of these complex associations will exist in the training set but not in the real data because of sampling noise when there is insufficient training data. Overfitting results from this, and numerous techniques have been devised to lessen it. These include putting an end to the training as soon as performance on a validation set begins to deteriorate, adding various weight penalties like L1 and L2 regularization, and soft weight sharing.

Diagram, schematic

Description automatically generated

**Optimizers**

\*\*It is used to avoid loss and it’s a PCA we used stochastic gradient decent

\*\* we are trying to minimize the loss and performance using stochastic gradient descent (Using weight updating rule to achieve good performance in the model)

\*\*we tried RMSPROP -that uses a decaying average of partial gradients in the adaptation of the step size for each parameter

**Code:**

Loading all the required Libraries

Graphical user interface, text, application

Description automatically generated

Data Preparation:

We load the train and test.csv files

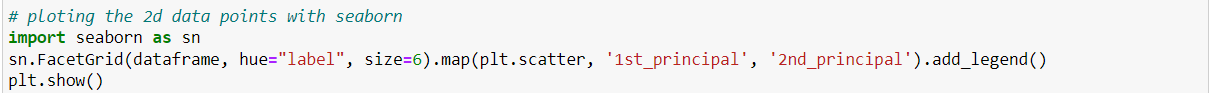
A picture containing table

Description automatically generated

A picture containing text

Description automatically generated

* WE are plotting the 2D Data points



Chart, scatter chart

Description automatically generated

## PCA using scikit-learn

**Performing PCA using Scikit-Learn is a two-step process:**

1. Initialize the PCA class by passing the number of components to the constructor.
2. Call the fit and then transform methods by passing the feature set to these methods. The transform method returns the specified number of principal components.

Graphical user interface, text, application

Description automatically generated

Chart, scatter chart

Description automatically generated

**PCA For Dimensionality Reduction:**

Graphical user interface

Description automatically generated with low confidence

\*\*If we take 200-dimensions, approx. 90% of variance is explained.

PCA With NORMALIZATION:

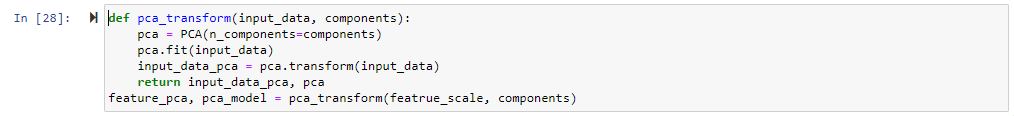
Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, chart, line chart

Description automatically generated

To reduce the dimension of given dataset:



DEFINING THE MODEL:

CNN:

# Set the CNN model

# my CNN architechture is In -> [[Conv2D->relu]\*2 -> MaxPool2D -> Dropout]\*2 -> Flatten -> Dense -> Dropout -> Out

Text

Description automatically generated

**Set the Optimizer:**

I used RMSProp as optimizer that uses a decaying average of partial gradients in the adaptation of the step size for each parameter



Text

Description automatically generated

**Data Augmentation:**

\*\*We can prevent overfitting through data Augmentation and can get higher accuracy of 0.99286.

Text

Description automatically generated

# Evaluate the model

# Plot training and validation curves

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

Confusion Matrix:

Text

Description automatically generated

Chart, scatter chart

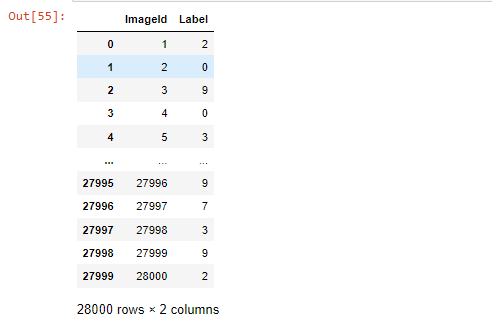
Description automatically generated

**Results:**

|  |  |  |
| --- | --- | --- |
| **MODEL** | **TRAIN ACCURACY** | **VALIDATION ACCURACY** |
| **RANDOM FOREST CLASSIFIER** | 0.1 | 0.9497 |
| **CONVOLUTION NEURAL NETWORK** | 0.9715 | 0.99 |

Graphical user interface, text, application

Description automatically generated



Graphical user interface, application

Description automatically generated

\*\* When using the Random Forest classifier, we have an accuracy of 0.94% and we can also get more accuracy by increasing the number of estimators and trees.

\*\*When using CNN, we have an accuracy of 99% which is best when compared to Random forest classifier.

**Contributions:**

\*\* We can implement this in Machine Learning but there is one big disadvantage of ML i.e., loss of 2D information due to image flattening -when we stretch any image we lose some information on surrounding values.

\*\*Here Convolution Neural Networks comes into picture which is to prevent any loss of information.

**Conclusions:**

\*\*Using the above techniques Train score: 97% constant accuracy for Validation score: 99%

\*\*when we keep increasing layers with different optimization techniques, we can see that increase in validation accuracy.

\*\*When we apply the regularization model with dropout technique there is much added advantage in validation accuracy.