**Train a CNN on the SVHN Dataset for Classification**

A blue circle with text and a book

Description automatically generated

Name: Nouman Ali

Section:05

Roll No: DSAI\_GIL\_248

**Summary:**

In this project, we are to do a multi-class classification of digits. The data set used in this project is the Google Street View housing number (SVHN). The project flow will start with data loading then we will do data preprocessing. We will use techniques like normalization, one-hot encoding, and data augmentation. Then, we will build our model. The model which we will be using is CNN. We will define convolutional layers, pooling layers, dropout layers, dense layers, and flatten layers in the model. We will use activation functions based on our requirements. Then, we will compile and train the model using Adam optimizer, batch size of 64 and 30 epochs. Finally we will access the performance of our model on the test set.

**Overview:**

In this project, we will be training a model to perform a multi-class classification of digits from the SVHM (Street view house number). To do this we will be using CNN, the reason for using CNN is that it performs well on image datasets. We will try to come up with different architecture of CNN and see how we can improve the accuracy of the model. To evaluate the model’s performance, we use metrics like accuracy, recall, f1 score, and precision.

**Literature review (2 articles from 2022-23):**

In the paper “Theoretical Understanding of Convolutional Neural Network:

Concepts, Architectures, Applications, Future Directions” author Mohammad Mustafa Taye discusses theories used in CNN. He discusses how CNN have dominated the computer vision and pattern recognition tasks in recent years. The author talks about the basics of CNN such as inputs, convolutional layers, pooling layers, activation functions, fully connected layers and output layers. Further, the author discusses some of the popular CNN architectures starting from LeNet (1998) to DenseNet-121 (2017) and also discusses the strengths and gaps of these architectures. He also mentions different types of CNN architectures used for classification, detection and segmentation.

In another article “Convolutional Neural Networks for Image Classification” author Jasmin Bharadiya discusses the importance of CNN in daily life. She mentions the key reasons for the significance of CNN. She discusses in about translation invariance, data efficiency in CNN, transfer learning and scalability. She discusses the benefits of transfer learning and how we can take advantage of using pre-defined models.

**Model architecture:**

As we are using a CNN the model architecture will consist of an input layer which will accept the same dimension of our image which is 32\*32\*32. Then it will have different combinations of, Conv2D layers, Batch Normalizations layers, Dropout layers, pooing layers and Flatten layers. The activation function for the Conv2d layers will be relu. The last layer (output layer) will be a dense layer which will have 10 neurons, and each will represent the probity of the class label. We will use the softmax function as activation and after that, it will classify the output to label with the highest probability. The whole model’s architecture is shown in the figure1.1 below.

A screen shot of a computer code

Description automatically generated

**Figure:1.1**

**Dataset Details:**

The dataset used for this project is SVHN which consists of real-world images. We will be using format 2 of the dataset where the images have been resized to 32x32 pixels and cropped to fit the format with minimal loss of information. There are a total of 600000 images, and they are divided into training and testing sets. The training set contains 531131 images, and the testing set consists of 26032 images. These are stored in mat format which we can easily read in Python. This set contains 2 variables ‘X’ and ‘y’. X contains a 4-D matrix containing the images and y contains the labels for the images. There are a total of 10 labels. Label 1-9 corresponds to the digits 1-9 respectively, whereas the label 10 represents digit 0.

**Hyperparameters:**

The hyperparameters for this project as listed below:

1. **Convolutional Layer Hyperparameters (Conv2d):**

In the CNN we use different filters. The number of filters used can be considered as hyperparameters. Also, the kernel size is another hyperparameter. In our model, we used a total of 4 Conv2d layers and we also changed the depth of these layers. Another hyperparameter here is the activation function. We used relu activation function for all the convolutional layers.

1. **Pooling Layer Hyperparameters**

We used two types of pooling layers in our model. First is the max pooling layer and the next is the average pooling layer. We can also set the pool sizes here we used only (2,2).

1. **Dense Layer Hyperparameters**

In the dense layer, we set the size of the layer to 10 which sets the output layer to have 10 neurons. The activation function used here was softmax which scales the output layer into probabilities.

1. **Dropout Later Hyperparameters**

The dropout layer was taken in between the convolutional layers and pooling layers which drops the fraction of neurons during training. It takes a value between 0-1 and drops the neurons based on it.

1. **Optimizer and Learning rate**

We have used the Adam optimizer and the learning rate was set as default which is 0.001 in case of Adam optimizer.

1. **Batch Size**

The batch size used for training is set to 128 for our model. Typical values for bath size ranges from 16 to 256.

1. **Number of Epochs**

Number of times our model trains with the training data is called epochs. We set the epoch to 30 for this model.

**Evaluation metrics:**

To evaluate the model, we first generated a confusion matrix to see all the true positives, true negatives, false positives and false negatives. Then we used the metrics accuracy, precision, recall, and F1 score. Accuracy is the measure of the proportion of correctly predicted instances (true positives + true negatives) out of the total instances. Precision represents how many of the predicted true positives are actually positive (true positives/ (true positive + false positive)). Recall is the measure of how many of the actual positive instances were correctly predicted (true positives/ (true positive + false negatives)). F1 score is the harmonic mean of precision and recall (2 x (precision x recall/precision + recall)). These metrics will give a clear understanding of the model’s performance.

**Analysis of results:**

The accuracy of the model on the testing data was 91.77%.

The precision of the model on the testing data was 91.82%.

The recall of the model on the testing data was 91.77%.

The f1 score of the model on the testing data was 91.78%.

Overall, the model performed pretty well with minimal difference between the precision and recall, which is ideal when both the false positives and false negatives needs to be minimized. The model is well calibrated as the differences between recall, precision, and f1 score is very less.

**Possible improvements:**

The overall performance of the model is satisfactory but there is still room for improvements. One thing we can do to improve the model performance is to add more filters/layers in the CNN model. We can try different sizes for the filters and add more filters or remove some filters and see the performance of the model. Also, we can change the pooling layers to see the results of the model. Another thing which might improve the results of the model is to increase the epoch and change the batch size. We can also change the values in datagen see the results and alter the values accordingly.

These are some methods which we can apply and check the results of the models and see what effect changing these factors might have.

**References:**

Taye, M. M. (2023). Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions. Computation, 11(3), 52. <https://doi.org/10.3390/computation11030052>

Bharadiya, J. (2023). Convolutional neural networks for image classification. International Journal of Innovative Research in Science, Engineering and Technology, 8, 673. <https://doi.org/10.5281/zenodo.7952031>