**Part I: A Dataset of Images or Audio Files (20 Points)**

**Name of the database: Microsoft COCO (Common Objects in Context)**

This is a database that aims to enable future research for object detection, instance segregation, captioning, and personal key points localization. This dataset has various features:

Object Segmentation

Recognition in context

Superpixel stuff segmentation

330K images (>200K labeled)

1.5 million object instances

80 object categories

91 stuff categories

5 captions per image

250,000 people with 17 different key points

A picture containing text, different, various, items

Description automatically generated

**Size of the dataset: approximately 25 GB (compressed)**

Number of records: 330K images, 80 object categories, 5 captions per image, 250,000 people with key points.



Keypoint detection for Pose Estimation in the COCO dataset.

This dataset is achieved by gathering images of complex everyday scenes containing common objects and their natural context. Objects are labeled using per-instance segmentations to aid in precisely object localization. This dataset contains photos of 91 object types that would be easily recognizable by a 4-year-old.

The dataset has 123,287 images and 886,284 instances. On the website, particular images can be explored by selecting them and the dataset shows all the images of that kind in the dataset.

The dataset has an active research paper that talks about this dataset in detail. The dataset also has an active API through which it can be fetched.

The download link for the dataset: <http://cocodataset.org/#download>

The link to the research paper: <http://arxiv.org/abs/1405.0312>

**Part II: Obtain CIFAR-10 Dataset (5 Points)**

First, start the GCP instance, activate the SSH browser window. After opening that point to the directory, we made earlier (JPTR\_NTBK) and make a new sub directory with the name CIFAR\_10\_DATA. After that ls-all for showing al the directories. Now point to the sub- directory and upload the data.

Graphical user interface, application

Description automatically generated

Now upload the data in it.

Graphical user interface, text

Description automatically generated

Upload all the files like this one by one. Then copy them from main home directory to the specific directory.

Text

Description automatically generated

**Part III: Build, Train, and Test CNN on CIFAR Dataset (25 Points)**

**Section III.I: Design of Model**

The CIFAR-10 dataset contains of 60000 images of 32 x 32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training sub-datasets and one test sub-dataset, each with 1000 images.

32, 5x5 convolutional filters/channels plus ReLU (rectified linear unit) node activations were created. After those 28 nodes are still present in height and width. We then perform down-sampling by applying a 2x2 max pooling operation with a stride of 2. Layer 2 consists of the same structure, but now with 64 filters/channels and another stride-2 max-pooling down-sample. We then flatten the output to get a fully connected layer with 3164 nodes, followed by another hidden layer of 1000 nodes. These layers will use ReLU node activations. Finally, SoftMax is used for classification on the layer to output the classes.

Diagram

Description automatically generated

the network architecture consists of two convolutional layers and fully connected neural networks to input layer as a flattened layer (8\*8\*64 nodes) which is connected to the dense layer having 1024 nodes. To predict the probability of the 10classes of the image, the dense layer is connected to the output later with 10 nodes. The highest probability node is considered as the label for the image. For the output layer, the SoftMax function is used to classify the probabilities of 10 classes.

**Convolution Layer 1:**

Filter for convolution: 4x4 +ReLu Layer

For max pooling: 2x2

Channels: 32

**Convolution Layer 2:**

Filter for convolution: 4x4 +ReLu Layer

For max pooling: 2x2

Channels: 64

The number of training steps is 5000. The network will be tested after 100 steps of training.

**Section III.II: Results of Testing Model**

**Graphical user interface, application, table

Description automatically generated**

**Section III.III: Report of Testing Model**

|  |  |
| --- | --- |
|  | CIFAR-10 |
| Learning\_rate  epochs  batch\_size  training\_step  Number of Convolution Layers | 0.0001  1  100  5000  2 |
| Accuracy\_Evaluation | 62.23% |

As mentioned in the above table, these were the hyperparameters and layers that were used for evaluating the test data accuracy. As we can see that the accuracy of the model is about 63.61% after 1 epoch. This result has been achieved without extensive optimization of the network.

**Part IV: Compare Convolutional Neural Network Performance (10 Points)**

**Section IV.I: 50 Data Points of accuracy level produced by CNN with MNIST dataset**

**Text

Description automatically generated with medium confidence**

**Section IV.II: 50 Data Points of accuracy level produced by CNN with CIFAR dataset**

**Graphical user interface, application, table

Description automatically generated**

**Section IV.III: Comparison of MNIST and CIFAR-10 Dataset Accuracy levels**

As we can see from the figure, MNIST clearly has a higher accuracy right from step 100. It says close to constant as soon it touches step 1400. In the case of CIFAR-10, we see a slow increase in the accuracy right from step 0. Although, the accuracy keeps on increasing and there is no dip in that.

|  |  |  |
| --- | --- | --- |
|  | MNIST | CIFAR-10 |
| Learning\_rate  batch\_size  training\_step  Number of Convolution Layers | 0.0001  100  5000  2 | 0.0001  100  5000  2 |
| Accuracy\_Evaluation | 98.88% | 62.23% |

As mentioned in the above table, there were the hyperparameters and layers that were used for evaluating the test data accuracy. As we can see that the accuracy of the model is about 98.88%. One of the reasons for the dip of the accuracy could be because MNIST dataset was normalized, and the CIFAR-10 isn’t.

**Section IV.IV: Explanation of gap in the performance**

This gap inaccuracy might be due to different reasons. While we set the same parameters and layers, the CNN model works differently with different databases. When training the model, the model’s learning ability depends on different parameters of optimization such as batch size, learning speed, epochs, layers of convolution, etc. It seems that for different values of hyperparameters, CIFAR-10 models need to be set or need more flexibility for better accuracy. Another reason could be that the image data in the MNIST dataset was normalized which was not the case in CIFAR-10. Another reason could be the type of dataset, the MNIST data is comparatively simpler and the features to recognize airplanes, birds like objects need more filters and different size of convolution filters as well.

**Part V: Improve Convolutional Neural Network Performance (20 Points)**

**Section V.I: Proposal to improve CNN performance**

The network architecture consists of two convolutional layers and fully connected neural networks to the input layer as a flattened layer (8\*8\*64 nodes) which is connected to the dense layers having 1024 node. To predict the probability of the 10 classes of the image, the dense layer is connected to the output layer with 10 nodes. The highest probability node is considered as the label for the image. For the output layer, the SoftMax function is used to classify the probabilities of 10 digits.

**Convolution Layer 1:**

Filter for convolution: 4x4 +ReLu Layer

For max pooling: 2x2

Channels: 32

**Convolution Layer 2:**

Filter for convolution: 4x4 +ReLu Layer

For max pooling: 2x2

Channels: 64

There are three cases that were designed to improve the accuracy of the CIFAR-10 dataset.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Case 1 | Case 2 | Case 3 |
| Learning\_rate  batch\_size  training\_step  Number of Convolution Layers | 0.0001  100  10000  2 | 0.0001  100  15000  2 | 0.001  100  5000  2 |
| Accuracy\_Evaluation | 67.73% | 69.62% | 69.60% |

As we can see that there is an improvement from the older model (62.23%) to 67.73% in both the cases above. Case 1 takes lesser time to execute than case 2 due to lesser number of training steps.

The original part 3 dataset has 62.23% where the training step is 5000 and case 3 has the same model as of the original part 3 and the accuracy is better just by increasing the learning rate.

I personally will prefer case 3 since it takes lesser time to execute and still gives great accuracy. This model goes about 8% of the accuracy gain.

The minor changes to this model will be due to the increase in the epochs and higher training step. From the analysis it clearly looks like that the model needs to learn more.

**Section V.II: Results of Testing Model**

Table

Description automatically generated

**Section V.III: Report of Testing Model**

|  |  |  |
| --- | --- | --- |
|  | Old CIFAR-10 Model | New CIFAR-10 Model |
| Learning\_rate  batch\_size  training\_step  Number of Convolution Layers | 0.0001  100  5000  2 | 0.001  100  10000  2 |
| Accuracy\_Evaluation | 62.23% | 69.60% |

As mentioned in the above table, these were the hyperparameters and layers that were used for evaluating the test data accuracy. As we can see that the accuracy of the model is about 67.73%. But the training step here for the training data is 10000. One of the reasons for the increase in the accuracy can be due to a greater number of the run for the training data so that it can learn more. The model gets to learn more

while reading through the training data repeatedly.

**Part VI: Project Report (20 Points)**

**Section VI.I: Introduction**

This project is primarily based on the CIFAR dataset. CIFAR is ana acronym that stands for the Canadian Institute for Advanced Research and the CIFAR-10 dataset was developed along with the CIFAR-100 dataset by researchers at the CIFAR institute.

**About the dataset:** The dataset is comprised of 60,000 32x32 pixel color photographs of objects from 10 classes, such as frogs, birds, cats, ships etc. The class labels and their standard associated integer values are listed below:

* 0: airplane
* 1: automobile
* 2: bird
* 3: cat
* 4: deer
* 5: dog
* 6: frog
* 7: horse
* 8: ship
* 9: truck

These are very small images, much smaller than a typical photograph, and the dataset was intended for computer vision research. This project aims at building a convolutional neural network that can classify images. Along with this, another important section of this project is to compare the performance of the CIFAR-10 dataset with the MNIST dataset.

**Section VI.II: What has been done in Part III, IV, and V**

In part III, on the original MNIST dataset, 32, 5x5 convolutional filters/channels plus ReLU node activations were created. After those 28 nodes are still present in height and width. We then perform down sampling by applying a 2x2 max pooling operation with a stride 2. Layer 2 consists of the same structure, but now with the 64 filters/channels and another stride-2 max pooling down sample. We then flatten the output to get a fully connected layer with 3164 nodes, followed by another hidden layer of 1000 nodes. These layers will use ReLU node activations. Finally, SoftMax is used for classification on the layer to output the 10-digit probabilities.

The network architecture consists of two convolution layer and fully connected neural networks to the input layer as a flattened layer (8\*8\*64 nodes) which is connected to the dense layer having 1024 nodes. To predict the probability of the 10 classes of the images, the dense layer is connected to the output layer with 10 nodes. The highest probability node is considering as the label for the image. For the output layer, the SoftMax function is used to classify the probabilities of 10 digits.

In part IV, the data points that were generated from MNIST data and CIFAR-10 dataset were added to the file and then the performance of them was compared. Some possible reasons for the difference in accuracy between the two models were also evaluated.

In part V, a proposal was given to improve the performance of the CIFAR-10 dataset. Changes were made to the original model to improve its accuracy.

**Section VI.III: Conclusion of Part III, IV and V.**

In part 3, after running the model, 62.23% accuracy was achieved. The epochs were 1 and the step size was 5000 for the model. This result was achieved without extensive optimization of the network. But the accuracy of the model needs to be improved.

In part 4, the accuracy of the MNIST data as well as CIFAR data, was then compared in which it was found the MNIST has a significant test accuracy of 98.53% whereas the CIFAR-10 dataset has 63.61%. A graph is drawn to compare the accuracy of both the dataset models on each 100th step.

This gap inaccuracy might be due to different reasons. Although we set the same parameters and layers, with different datasets, the CNN model works differently which is called the stochastic nature of the algorithm. It appears that CIFAR-10 models need to be configured for different values of hyperparameters or need more flexibility for better accuracy. Another explanation might be the normalization of the image data in the MNIST data set, which was not the case in CIFAR-10.

In part V, a proposal was introduced with 3 cases to improve the accuracy of the CIFAR-10 data. Amongst, the three cases, the model that was chosen was case 3 which has a test accuracy of about 69.60 %. But the learning rate here for the training data is 0.001. One of the reasons for the increase in accuracy can be due to the small learning the model was not able to converge smoothly and was slowing down the whole process.

**Section VI.IV: Conclusion of the Project.**

This project gave good experience in creating a CNN for a very popular dataset of CIFAR-10. I learned to build CNN and create my models by tuning the hyperparameters. I also learned good knowledge of how to effectively use cloud servers (such as Google Cloud Platform) to use resources efficiently, as these models require high computing power. I tried to run this on my own computer's Jupyter Notebook, but the kernel was dead. This obviously means that there is a need for cloud platforms to run these models.

One of the most interesting parts about this project was to tweak the parameters. I was getting similar accuracy by increasing the training step, but since we want a model to run optimal, more training step was a more logical approach than increasing more epochs in the comparison of the cases.

**Part VII: Final Presentation Videos: YouTube Links**

I made one 30 mins video, the link to that is:

<https://youtu.be/jDEoBeQN-2Q>