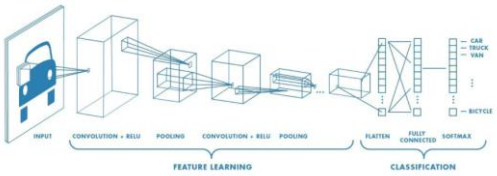
Assignment-3 Report

Team Members:

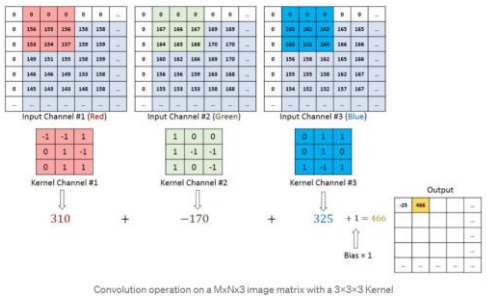
1. Maruthi Sriram (IMT2019068),
2. Tarun Reddy (IMT2019088),
3. K V V Deepesh(IMT2019508).

Question-3a:

Convolution Neural Network (CNN):



A CNN is a deep learning algorithm which takes an image as an input. It gives weight or importance to various parts of the image and be able to differentiate between different images. The pre-processing required in CNN is very less than other classification algorithms. A CNN is able to capture spatial and temporal dependencies in image through the application of different filters. The role of CNN is to reduce the given image into a form which is easy to process, without losing important or critical features required for a good prediction. This makes it easier when we have dataset with large number of images. The element involved in carrying out convolution operation in convolution layer is called Kernel/Filter. This reduces the size of the image.

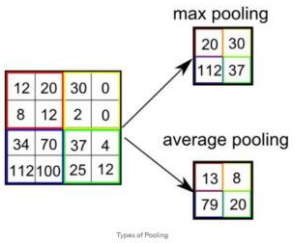


Convolution operation on a MxNx3 image matrix with a 3×3×3 Kernel

The objective of the kernel is to extract high level features such as edges from input image. CNN need not be limited to only one Convolutional layer. The first CNN layer is responsible for getting low level features such as edges, colour, etc. Other layers are there for high level features which gives us an understanding of the image.

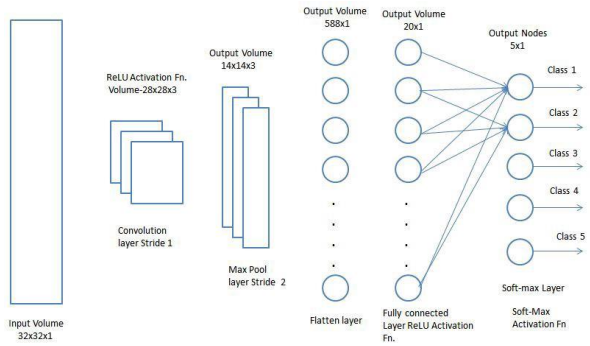
Pooling Layer:

Pooling layer is responsible for reducing even more of the spatial size of the convolved feature. There are two types of pooling: Max and Average Pooling. Max pooling returns the maximum value from the portion of image covered by the kernel. Average pooling returns average of all values from the portion of image covered by kernel. Max pooling removes noise from the images. Average pooling simply performs dimensionality reduction. So Max pooling is much better than Average Pooling.



The Convolution Layer and Pooling Layer combine and form a layer in CNN. Depending on the image we decide the number of such layers required. Now at the last CNN layer we flatten the convoluted image and give it as an input to the neural network for classification

Classification – Fully Connected Layer (FC Layer):



FC layer is a way of learning non-linear combinations of high-level features as represented by the output of the convolution layer. Now we flatten the convoluted image we obtain and give it as input to the FC layer. Over a series of epochs, the model distinguishes between dominating and certain low-level features in images and classify them using sigmoid or relu or tanh classification.

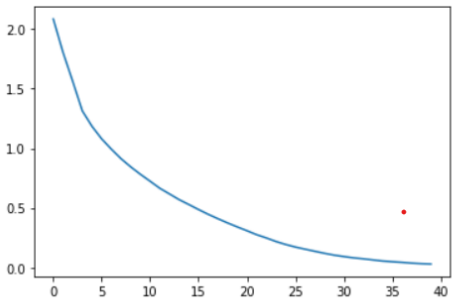
In this model I have 5 convolutional layers and two fully connected layers. The 1st, 2nd , 5th convolution layers have pooling layer after them. The pooling layer has a kernel size of 3 x 3. I used SGD optimizer and tested the models with and without momentum. Momentum is a parameter we give in SGD optimizer which increases the converging rate and gives results faster. I applied the layers of the models with sigmoid, tanh and Relu activation functions. I trained the CIFAR 10 with the above combination momentum and different activation functions. The results are listed below.

Results:

Without Momentum:

The below are the results when Relu is used as activation function:

Epochs vs Loss:



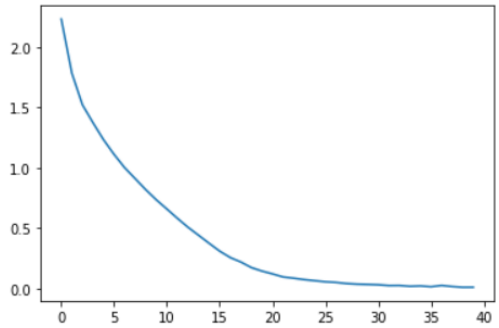
Accuracy of every class:

```
Accuracy for class plane = 81.7 %
Accuracy for class car   = 87.5 %
Accuracy for class bird  = 59.9 %
Accuracy for class cat   = 70.1 %
Accuracy for class deer  = 69.8 %
Accuracy for class dog   = 56.9 %
Accuracy for class frog  = 83.2 %
Accuracy for class horse = 82.3 %
Accuracy for class ship  = 83.9 %
Accuracy for class truck = 81.9 %
```

The final accuracy is 81 percent

Results when tanh is the activation function:

Epoch vs Loss:



Accuracy for every class:

```
Accuracy for class plane = 87.4 %
Accuracy for class car   = 89.5 %
Accuracy for class bird  = 61.4 %
Accuracy for class cat   = 65.0 %
Accuracy for class deer  = 74.4 %
Accuracy for class dog   = 72.1 %
Accuracy for class frog  = 77.3 %
Accuracy for class horse = 83.4 %
Accuracy for class ship  = 81.7 %
Accuracy for class truck = 86.9 %
```

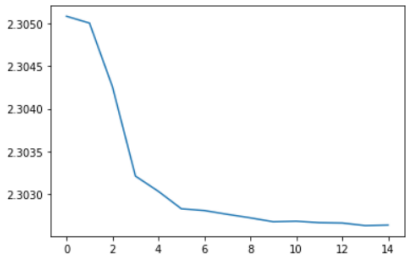
The final accuracy is 78 percent

Results when sigmoid is the activation function:

Sigmoid gives accuracy of 10 percent.

I ran sigmoid activation function for 15 iterations because it was computing results really slow and it was giving low accuracy.

Epoch vs Loss:



Accuracy for every class:

```
Accuracy for class  plane = 0.0 %
Accuracy for class  car   = 0.0 %
Accuracy for class  bird  = 0.0 %
Accuracy for class  cat   = 0.0 %
Accuracy for class  deer  = 0.0 %
Accuracy for class  dog   = 0.0 %
Accuracy for class  frog  = 0.0 %
Accuracy for class  horse = 0.0 %
Accuracy for class  ship  = 100.0 %
Accuracy for class  truck = 0.0 %
```

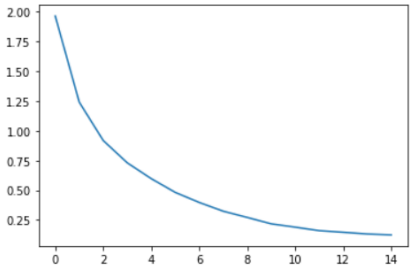
The final accuracy was 10 percent.

With momentum:

I used a momentum of 0.85 for this task

The below are the results when Relu is used as activation function:

Epoch vs Loss:



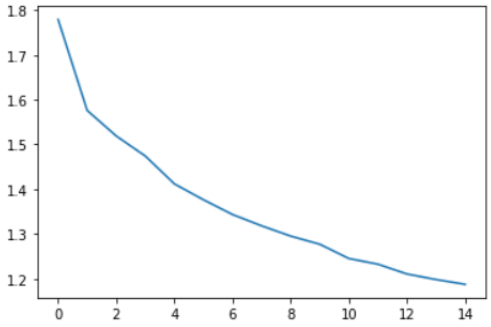
Accuracy for Every Class:

```
Accuracy for class plane = 79.8 %
Accuracy for class car   = 87.0 %
Accuracy for class bird  = 73.9 %
Accuracy for class cat   = 62.7 %
Accuracy for class deer  = 81.0 %
Accuracy for class dog   = 65.7 %
Accuracy for class frog  = 86.2 %
Accuracy for class horse = 82.6 %
Accuracy for class ship  = 92.6 %
Accuracy for class truck = 88.6 %
```

The final accuracy is 82 percent.

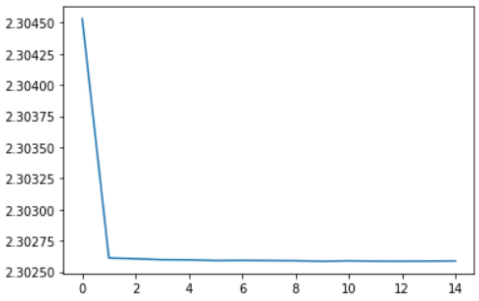
The below results are when tanh is used as activation function:

Epoch vs Loss:



The below results are when sigmoid is used as activation function:

Epoch vs Loss:



Sigmoid is a bad activation function because the loss doesn't decrease after a value.

Accuracy per class:

```
Accuracy for class  plane = 0.0 %
Accuracy for class  car   = 0.0 %
Accuracy for class  bird  = 0.0 %
Accuracy for class  cat   = 0.0 %
Accuracy for class  deer  = 0.0 %
Accuracy for class  dog   = 0.0 %
Accuracy for class  frog  = 0.0 %
Accuracy for class  horse = 0.0 %
Accuracy for class  ship  = 100.0 %
Accuracy for class  truck = 0.0 %
```

The final accuracy is 10 percent

Final Model recommended:

From the results I observed, Finally I recommend using Relu activation function for every layer and with momentum of 0.9 because it gives us the best accuracy with the given model. Sigmoid gives the least accurate model when used as activation function and computation time for it is also very slow. So it is not recommended. Relu is recommended because it takes less time to train the model to get good accuracy and when momentum is there it becomes even more fast to train. After Relu, tanh is the best model to use.

Question-3b:

Using Alexnet or any deep NN as a feature extractor for an object recognition dataset and using classification models on top to check the classification accuracies.

Dataset:

For this assignment, I used an object classification dataset called YIKES! which contains images of 15 species of spiders for classification.

Link to the dataset: https://www.kaggle.com/datasets/gpiosenka/yikes-spiders-15-species

Procedure:

First, we import the dataset and then read the images. After reading the images we transform the image to the form where it can be used in Alexnet for feature extraction.

In the transform function, we resize the image to (224, 224) which is the required size for Alexnet and also convert it to a tensor. After this we also normalise the so that they can be trained easily on Alexnet.

After the images have been transformed, we put them into Alexnet after which output the features. After we get the features, we put them into our classification models and check the accuracy.

Models used and accuracy:

Accuracy for Logistic Regression – 0.89

Accuracy for SVM – 0.86

Accuracy for Gaussian Naïve Bayes – 0.76

Models used and accuracy for Bikes vs Horses Classification:

I have run the same models for Bikes vs Horses Classification after extracting features through Alexnet.

Accuracy for Logistic Regression – 1.0

Accuracy for SVM – 1.0

Accuracy for Gaussian Naïve Bayes – 1.0

Question-3c:

YOLO

YOLO (You Only Look Once) is an algorithm which is used to provide real-time object detection. YOLO algorithm uses ”Convolutional Neural Network (CNN)” to detect objects. YOLO algorithm works by first dividing the image into equal sized grids and then it predicts probabilities for each region. YOLO makes predictions with a single neural network which makes it cost efficient compared to R-CNN, which uses multiple neural networks for a single image.

Observations

In this part of the assignment, we used, “YOLO” method to detect autos. We took an already trained vehicle model and trained it with our custom data. For lower number of, “epochs” it was also detecting vehicles, so we increased the number of “epochs” and also increased the ”–conf” parameter which is the confidence level of the model. It was detecting less number of vehicles and only autos. The model was not working well for autos which are very far away. The model was giving good results for autos which are behind objects and also for different orientations of the image.

Code Explanation

Before starting, we changed the notebook setting ”Hardware accelerator”, we set it to ”GPU” for faster processing. First we downloaded the YOLO package from ”ultralytics” yolov5 for detecting autos. We can import the folder from GitHub. Once we get the ”yolov5” folder, we have to create an image and labels folder inside yolov5 folder. Images contains the auto images, labels contain the annotated autos in the form of ”txt”. To annotate the data, I used an online website, the link to the website is given below in the resources. Once we have all the data, we have to divide the data into training set and test set. Next we trained on the training set with about ”50” epochs. We first tried for 5 epochs but it was not giving good results so we kept on increasing the number of epochs and we found out that it is giving good results for 50 epochs. Later we used this trained model on the test set.

Results

In this section we are going to show the results we got:

References

1) https://www.makesense.ai/ - For annotating the data.

2) https://github.com/ultralytics/yolov5 - Yolo package