AI 825 VISUAL RECOGNITION

MINI PROJECT REPORT

Team -

Aditya Kaka(IMT2019002)

Amitha Reddy (IMT2019023)

Samhitha Perala(IMT2019521)

March 31, 2022

Contents

1 Question 3a: Play with CNNs

1.1 Introduction

CIFAR-10 dataset is trained on CNN model which is implemented in python using pytorch. The activation functions ReLU, tanh and sigmoid are performed with different combinations of Convolution and Fully Connected Layers.

1.2 CIFAR-10 data set

The CIFAR-10 dataset consists of 60000 32x32 colour 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 batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

1.3 Convolution Neural Networks

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take an input image, assign importance (learn-able weights and biases) to various aspects/objects in the image, and distinguish between them.

Convolutional neural networks are composed of multiple layers of artificial neurons, each layer generates several activation functions that are passed on to the next layer.

The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.

The layers in CNN are mainly Convolution layers, Pooling and Fully Connected Layers.

Convolutional layers convolve the input and pass its result to the next layer. After passing through a convolutional layer, the image becomes abstracted to a feature map.

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.

1.4 Observation and Results

Hyperparameters:

- Learning Rate : 0.001
- Optimizer : SGD(Stochastic Gradient Descent)
- Batch Size : 8
- 2d Max pooling
- Loss : Cross-Entropy Loss
- Number of Epochs : 10

CNN layer structure

$Input > Conv(Activationfunction) > MaxPool > Conv(Activationfunction) > MaxPool > .....FC(Activationfunction) > FC(Activationfunction) > FC(SoftMax) > 10outputs$

Activation function can be any one of relu, tanh, sigmoid functions.

2CN,3FC

- ReLU

Parameters	Time(seconds)	Accuracy(%)
Momentum = 0.9	494.8452250957489	70
Momentum = 0.5	479.86128759384155	69
Momentum = 0.1	510.13413548469543	64
Without Momentum	531.7187728881836	65

Table 1: 2CN,3FC RELU

- Tanh

Parameters	Time(seconds)	Accuracy(%)
Momentum = 0.9	570.1902086734772	66
Without Momentum	499.4411995410919	60

Table 2: 2CN,3FC Tanh

- Sigmoid

Parameters	Time(seconds)	Accuracy(%)
Momentum = 0.9	516.6938850879669	51
Without Momentum	486.24809169769287	18

Table 3: 2CN,3FC Sigmoid

3CN, 3FC

- ReLU

Parameters	Time(seconds)	Accuracy(%)
Momentum = 0.9	1913.777	71
Without Momentum	556.545970916748	61

Table 4: 3CN,3FC ReLU

4CN, 3FC

- ReLU

Parameters	Time(seconds)	Accuracy(%)
Momentum = 0.9	1882.794	73

Table 5: 4CN,3FC ReLU

1.5 Conclusion

- Out of all the activation functions relu gives better results.
- Accuracy decreases with decrease in momentum.
- SGD with momentum gives better result compared to without momentum.
- Running time and Accuracy increases as we increase the layers in network(convolution layers).

2 Question 3b: CNN as a feature extractor

2.1 Introduction

EuroSAT dataset is trained on SGD model with AlexNet as the feature extractor. Pre trained AlexNet modelis available in tourch.hub

2.2 EuroSAT Dataset

EuroSAT dataset is based on Sentinel-2 satellite images covering 13 spectral bands and consisting of 10 classes with 27000 labeled and geo-referenced samples.

Two datasets are offered: - rgb: Contains only the optical R, G, B frequency bands encoded as JPEG image. - all: Contains all 13 bands in the original value range (float32).

We have choose rgb dataset for our training.

The 10 classes of the dataset are 'AnnualCrop', 'Forest', 'Highway', 'Industrial', 'Pasture', 'PermanentCrop', 'Residential', 'River', 'SeaLake', 'HerbaceousVegetation'.

Figure 1: Example of EuroSAT dataset images

2.3 AlexNet

- This architecture consists of 5 convolutional layers, 3 max-pooling layers, 2 normalisation layers, 2 fully connected layers and 1 softmax layer.
- The input to AlexNet is an RGB image of size 256×256. This means all images in the training set and all test images need to be of size 256×256.
- The activation function used in all layers is Relu. AlexNet uses Rectified Linear Units (ReLU) instead of the tanh function because a CNN using ReLU was able to reach a 25% error on the CIFAR-10 dataset six times faster than a CNN using tanh.
- The activation function used in the output layer is Softmax.

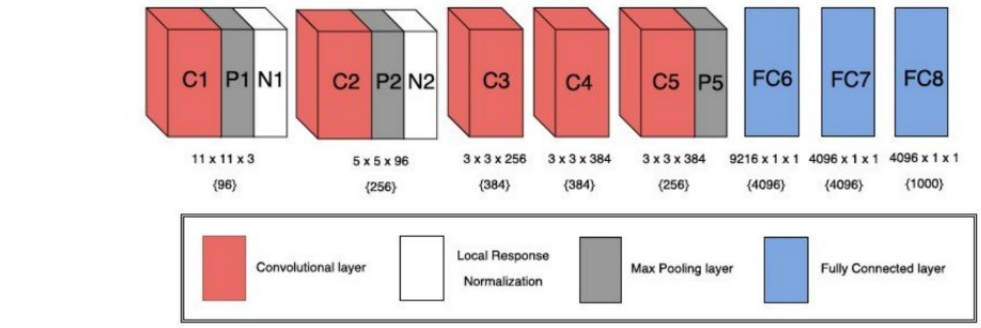

Figure 2: Layers of AlexNet

- AlexNet allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU.
- The total number of parameters in this architecture is 62.3 million.

Layer	# filters / neurons	Filter size	Stride	Padding	Size of feature map	Activation function
Input	-	-	-	-	227 x 227 x 3	-
Conv 1	96	11 x 11	4	-	55 x 55 x 96	ReLU
Max Pool 1	-	3 x 3	2	-	27 x 27 x 96	-
Conv 2	256	5 x 5	1	2	27 x 27 x 256	ReLU
Max Pool 2	-	3 x 3	2	-	13 x 13 x 256	-
Conv 3	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 4	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 5	256	3 x 3	1	1	13 x 13 x 256	ReLU
Max Pool 3	-	3 x 3	2	-	6 x 6 x 256	-
Dropout 1	rate = 0.5	-	-	-	6 x 6 x 256	-

Figure 3: Details of Layers of AlexNet

2.4 Observations and Results

Hyperparameters:

- Learning Rate : 0.001
- Optimizer : SGD(Stochastic Gradient Descent)
- Feature Extractor : AlexNet
- Batch Size : 8
- Loss : Cross-Entropy Loss
- Number of Epochs : 10

Results after running the dataset on our model are

Accuracy of individual classes are shown below -

Evaluation Metrics	
Accuracy	97%
Time taken	18644.87 sec
Cross Entropy Loss	0.08

Class	Accuracy(%)
AnnualCrop	99
Forest	95
Highway	94
Industrial	98
Pasture	99
PermanentCrop	98
Residential	97
River	99
SeaLake	96
HerbaceousVegetation	99

Table 6: Accuracy of individual classes

2.5 Horses vs Bikes

3 Question 3c: Auto detection

Custom training with YOLOv5.

3.1 YOLOv5

- YOLO is an acronym that stands for You Only Look Once.
- It is a revolution in real time object detection.

- It is a novel convolutional neural network (CNN) that detects objects in real-time with great accuracy.
- This approach uses a single neural network to process the entire picture, then separates it into parts and predicts bounding boxes and probabilities for each component.
- These bounding boxes are weighted by the expected probability.

3.2 Exporting XML files of images

- For the set of training images, we need the bounding boxes around the autos.
- So we have uploaded the image files to 'makesense.ai' and drew bounding boxes around each auto.
- We exported the annotation files in XML format to the same directory as our image files.
- We imported this directory to 'app.roboflow'.
- roboflow maps the images files to its respective XML files and draws the bounding boxes.

3.3 Augmentation

Now with the help of roboflow we split the data into 3 sets. Train,test and validation respectively in the ratio of 7:1:2. For every image in the training set, we take a flipped and rotated version of the image and save it back in the training set. Now we export this dataset in YOLOv5 mode.

3.4 Training

Now on our training dataset, we use the YOLOv5 model.Hyper parameters in YOLOv5 model-

- image size = 416
- batch = 16
- epochs = 200
- data = our YML files created earlier
- cfg = YOLOv5
- weights = custom path to weights
- name = results name
- device = runtime type GPU
- cache = cache images for faster training.

3.5 Observations

Using TensorFlow we can plot the epoch vs evaluation metrics like precision, recall etc.

epoch = 50

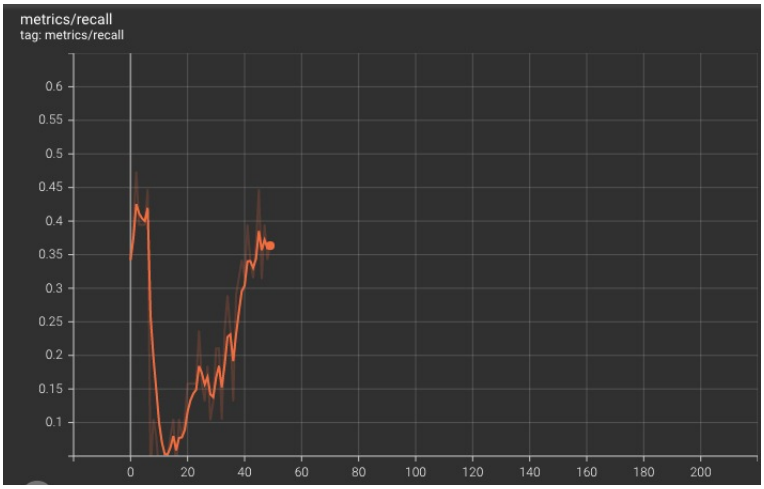

Figure 4: Recall vs epoch=50

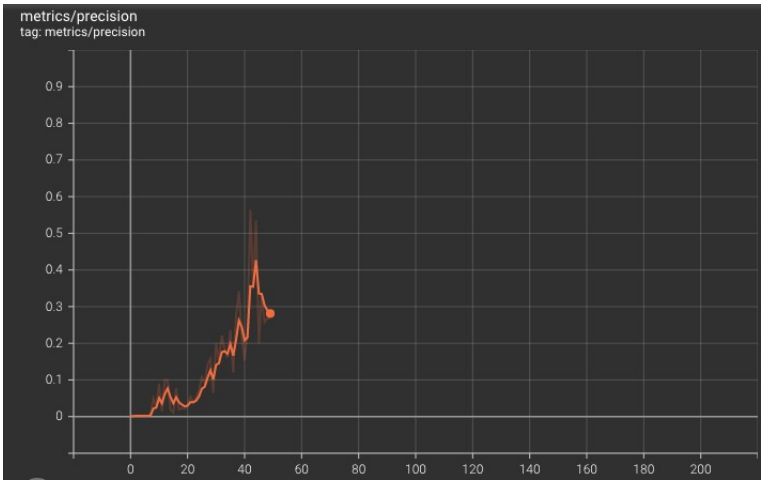

Figure 5: Precision vs epoch=50

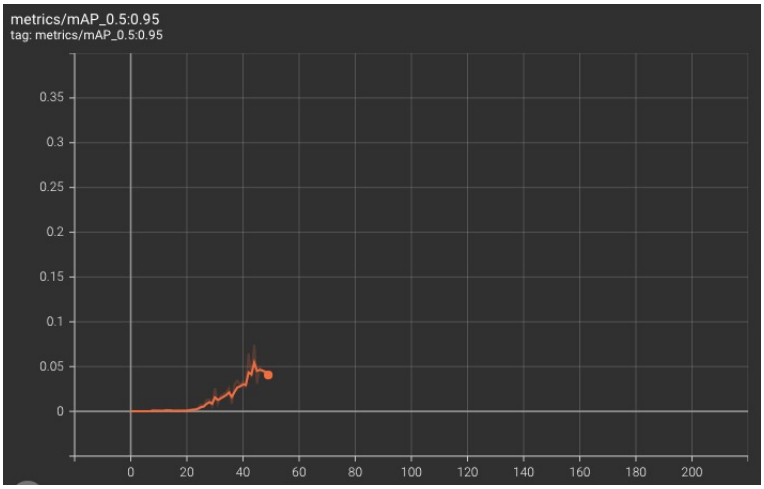

Figure 6: MAP vs epoch=50

epoch = 200

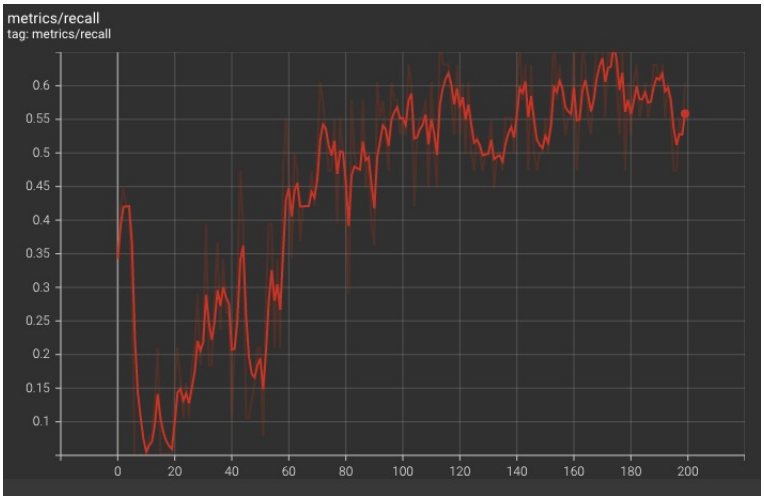

Figure 7: Recall vs epoch=200

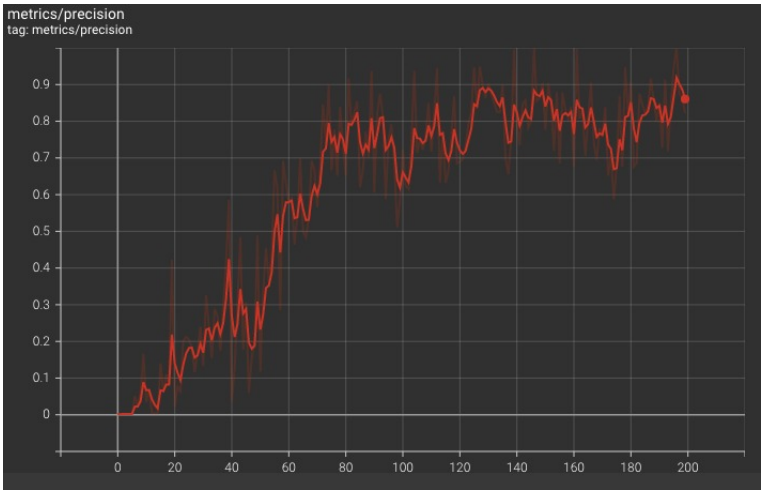

Figure 8: Precision vs epoch=200

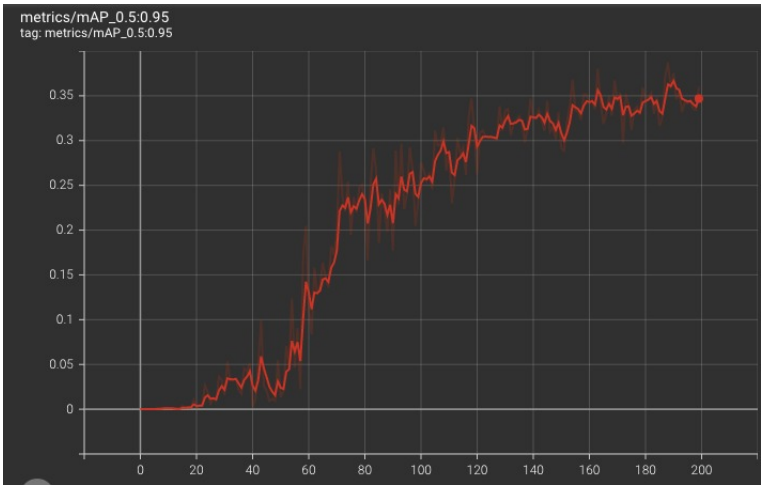

Figure 9: MAP vs epoch=200

We can see that the precision increases with increased epochs.

3.6 Testing

We apply this model on the testing images, and see the following results.

Figure 10: Testing image

Figure 11: Testing image

Figure 12: Testing image

Figure 13: Testing image

Figure 14: Testing image

3.7 Failed Cases

Figure 15: Failed image

We see that auto is not detected in dim/dark light.

Figure 16: Failed image 2

Here parts of the bike like the tyre and tail light were detected as they are common with features of an auto.

4 Citations

- https://analyticsindiamag.com/implementing-alexnet-using-pytorch-as-a-transfer-learning-m
- https://github.com/phelber/EuroSAT
- https://medium.com/analytics-vidhya/alexnet-a-simple-implementation-using-pytorch-30c14e8
- https://www.youtube.com/watch?v=MdF6x6ZmLAY&ab_channel=Roboflow