Assignment 3 - Report

Submitted by:

- Nandakishore S Menon (IMT2019057)
- Rachna S Kedigehalli (IMT2019069)
- K Yashovardhan Reddy (IMT2019097)

3a - Play With CNNs

- We first make the given dataset compatible with the input of AlexNet model. `torchvision.transforms` is used for this. Many transforms are applied together using `transforms.Compose`.
  1. Image are resized to size (227, 227) as required by AlexNet.
  2. Images are converted to tensors.
  3. Images are also normalized.
- Data is loaded using `DataLoader`.
- A medium deep network with 3 Conv and 2 FC layers is constructed. The class also has a function `forward_activation_fn` which does forward propagation using appropriate activation function.

Training and Validation

- Optimization algorithms used:
  - SGD (Stochastic gradient descent): Can be used with or without momentum
  - Adam: For adaptive learning

Comparison

- **Metrics:** training time and classification performance

Training and validation time (in seconds):

Algorithm	ReLU	Tanh	Sigmoid
Adam	209.9232337474823	206.80912613868713	206.27637553215027
SGD - Without Momentum	202.20012021064758	206.27637553215027	203.10808515548706
SGD - With Momentum	203.1022801399231	203.6202211380005	201.28381085395813

Accuracy:

Algorithm	ReLU	Tanh	Sigmoid
Adam	70.55%	69.65%	39.95%
SGD - Without Momentum	10.54%	22.27%	10.00%
SGD - With Momentum	30.59%	33.83%	10.00%

Recommended Architecture

Adam optimization algorithm gives much better accuracy than SGD (with or without momentum). Though training time of Adam is higher, its better classification performance compensates for it. So with respect to optimization algorithm, Adam is recommended.

With Adam algorithm, ReLU activation function gives better accuracy than tanh and sigmoid, but takes more time to train. With SGD, tanh gives better accuracy, but takes more time.

Overall, Adam optimization algorithm with ReLU as the activation function is recommended.

3b - CNN as a feature extractor

A pre-trained Alexnet model is obtained, whose last layer is altered to obtain the features. This is done by making the last layer of the classifier linear. The model is **not** to be trained again on the given data.

A pre-trained AlexNet model is obtained from `torchvision.models`. The AlexNet classifier has the following layers:

```
(classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
```

The last layer is replaced with a Linear layer that outputs the features for the given image without altering the weights of the pre-trained model.

Each of the given images(train and test) is transformed into tensors of appropriate dimensions so that they are compatible with the AlexNet model input dimensions.

Both the model and the tensor corresponding to the input image are moved to CUDA(if available) for general-purpose computing on GPU.

The features for all the training and test data are computed and a Logistic Regression classifier is trained on the training data. The accuracies for the Intel Image Classification data set are given below:

```
0.6646666666666666
```

The same model used on the Bike vs Horse dataset (no training required), gives an astoundingly high accuracy:

```
0.9833333333333333
```

3c - Auto detection

YOLO an acronym for 'You only look once', is an object detection algorithm that divides images into a grid system. Each cell in the grid is responsible for detecting objects within itself. YOLO is one of the most famous object detection algorithms due to its speed and accuracy.

YOLOv1 was released as a research paper by Joseph Redmon. The paper was titled You Only Look Once: Unified, Real-Time Object Detection. Further improvements and changes introduced YOLOv2,YOLOv3,YOLOv4 and YOLOv5. For detecting auto-rickshaw YOLOv5 has been used.

The salient features of “yolo” are :

- Entire image is used as a single instance and predicts the coordinates for the bounding box for classes and also their probabilities.
- It divides the image into grids and predicts the objects in each grid and non max suppression is used to get the overall best bounding box for a specific object.
- The above method of processing images allows for it to process and identify classes with speed and accuracy. It is fast enough to be deployed in real time i.e. it can process more than 40 frames per second.

Its disadvantages are :

- It struggles to detect objects which occupy multiple grids and also objects which are very small.

To train the model to detect auto-rickshaws, a dataset was created by annotating images of autos. To obtain the dataset to train on the tool “roboflow” was used. The 80% of the dataset was used to train, 13% was used to validate and 7% was used to test the model. The annotated dataset has only one class “autos”.

The YOLOv5 model used was “yolov5s”, as it was the fastest among the other options. To train the command used was :

```
!python train.py --img 640 --cfg custom_yolo5s.yaml --batch 4 --epochs 30 --data /content/yolov5/data/data.yaml --weights yolov5s.pt -
```

The data was sent in batches of 4 and was trained for 30 epochs. It obtained a maximum mAP

value of `0.915` . The notebook where the code was executed can be found here. The output for some of the test and validation set is as follows: