Indian Institute of Technology, Mandi
February - May 2020

CS671 - Deep Learning and its Applications

Programming Assignment 1

Course Instructor : Aditya Nigam

7 March 2019

Instructions

- Plagiarism is strictly prohibited. In case of violation of the same, a zero will be awarded for this assignment as a warning and a quick F grade if repeated later.
- Submit a README.MD file for each question as well as a full assignment which provides the instructions for running your codes in detail including the versions of programming language and all the modules that have been used.
- Students using Windows or other OS are requested to make sure that their code runs perfectly on Linux as mentioned in each problem. Your evaluation will be done on computers in the PC lab.
- Zip the three zips made one for each question as group_ID.zip e.g., **13.zip**.
- Only submit documents that are mentioned in the submission sub-section of each problem, on **Moodle**. Other stuff has to be shown separately.
- The deadline for the assignment is **Sunday, 22nd March, 2020, 2200 HRS**. Please start working on the assignment from first day on-wards. No extensions will be given.
- You are required to upload your codes on **Github** as well as Moodle for all the assignments.
- You are required to make a web-page for your progress in this course. Links to your codes, videos, project pages, should be present on this web-page. Details of this have been conveyed via mail.
- The Github and web-page requirement carry **10** marks for each assignment. So take it seriously.
- You are free to use any high level api (tensorflow, pytorch, keras, etc) for the assignment.
- Contact Daksh Thapar (9592563214) for any queries related to assignment. Contact other TA's (list provided in mail) for basic fundamental doubts.

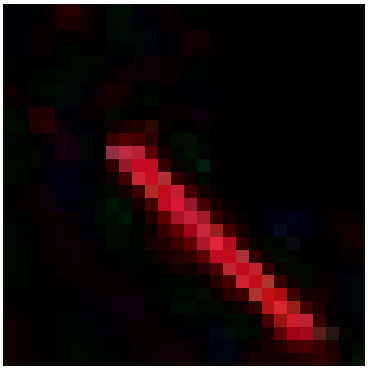

Figure 1: $1\_1\_9\_0\_3.jpg$

1 Data Management - Line Dataset Generation (10 marks)

1.1 Motivation

This problem is aimed at introducing you to the housekeeping around any Machine Learning problem, Data Management. Data that is not properly labelled becomes hard to manage. You will learn about **numpy** array manipulation and basic image handling by solving this problem.

1.2 Problem Statement

Make a dataset of $(28 \times 28 \times 3)$ images of **straight** lines on a **black background** with the following variations:

1. **Length** - 7 (short) and 15 (long) pixels.
2. **Width** - 1 (thin) and 3 (thick) pixels.
3. **Angle with X-axis** - Angle $\theta \in [0°, 180°)$ at intervals of $15°$.
4. **Color** - Red and Blue.

1.3 Notes

- The number of classes of images is fixed. You are required to make 1,000 random variations within each class. Note that only translation of the centre of the line can be used for intra-class variations.
- Any image of a particular type will not be correct if it doesn't fully confirm with all its said properties. For example, any image which has a long line can not contain any shorter or longer line than 15 times the pixel side length (small deviations due to pixels allowed). Distances are to be Euclidean i.e. diagonal of a pixel $= \sqrt{2} \times side$

1.4 Submission

- Name each image with underscore separated variation labels in the order mentioned before. Variation labels should be integers 0, 1, 2, ... mapped to variations in the order mentioned before e.g., thin and thick should be labelled as 0 and 1 respectively. Numbers after the last underscore will be IDs of images in that class. Images should be a .jpg files. For example, a sample 3^{rd} image of a long and thick red line, inclined at 135° with the positive x-axis should be named $1\_1\_9\_0\_3.jpg$ shown in Figure 1.

- Make a video of the images. Choose 90 images for each class and make a frame of 9 images (3×3). You will have 10 frames for each class. Make a video at 2 FPS. The aim of the video will be to convince us whether all the variations have been captured. In order to do that keep the class of 3 variations constant and then vary the class of the 4^{th} variations. The order in which you choose the variations to change the class in the video is the same as depicted in the problem statement.

2 A Simple Neural Network (20 marks)

2.1 Motivation

As most of this course will be wrapped around working with neural networks, a detailed understanding of the existing low-level libraries used for these tasks is incumbent. By solving this problem you will become familiar with how layer APIs work in these libraries.

2.2 Problem Statement

You are required to make a simple fully connected network to classify the MNIST dataset and the dataset you made in part 1. The caveat here is that you can not use layer APIs already provided in your library of choice. You will need to code a dense layer using in-built operations for matrix multiplication, etc. Please note that you are free to use all other functions of the library just not the layer functions.

2.3 Submission

- Code for the task.
- Models for both datasets.
- A 3-4 page report containing:
  - Learning curves (accuracy and loss).
  - F-scores.
  - Confusion Matrices.

  - Variations tried.
  - Inferences.

- Zip the code and report as 2.zip

3 Convolutional Neural Networks (20 marks)

3.1 Motivation

The goal of this assignment is to learn the basic principles of designing deep convolutions neural networks for image classification. Use 60% of training data for training the models and 40% for testing.

3.2 Problem Statement

Making a CNN model from scratch to classify the images of the line dataset into the respective 96 classes and MNIST dataset into 10 classes.

3.2.1 Part 1

In this part, we're going to specify a model for you to construct. The goal here isn't to get good performance (that'll be next), but instead to get comfortable with understanding the TensorFlow & Pytorch documentation and configuring your own model. Implement the model with the following network architecture.

1. 7x7 Convolutional Layer with 32 filters and stride of 1.
2. ReLU Activation Layer.
3. Batch Normalization Layer
4. 2x2 Max Pooling layer with a stride of 2
5. fully connected layer with 1024 output units.
6. ReLU Activation Layer.

Create a model with the above specified network architecture. Use the Adam optimizer with categorical crossentropy loss. Once the model is trained test it using the test data.

3.2.2 Part 2

In this part you need to create your own network architecture to achieve greater accuracy. You need to experiment with architectures, hyperparameters, loss functions, and optimizers to train a model.

Things you should try:

- **Filter Size**: Above we used 7x7; this makes pretty pictures but smaller filters may be more efficient
- **Number of Filters**: Above we used 32 filters. Do more or fewer do better?
- **Pooling vs Strided Convolution**: Do you use max pooling or just stride convolutions?
- **Network architecture**: The network above has two layers of trainable parameters. Can you do better with a deep network?
- **Use Learning Rate Decay**: Decaying the learning rate might help the model converge. Feel free to decay every epoch, when loss doesn't change over an entire epoch, or any other heuristic you find appropriate.
- **Regularization**: Use dropouts or l2 weight regularization.

At the very least, you should be able to train a ConvNet that gets at least 70% accuracy on the test set. This is just a lower bound - if you are careful it should be possible to get accuracies much higher than that! Extra credit points will be awarded for particularly high-scoring models or unique approaches.

3.3 Submission

- Code for both the parts.
- Models for both datasets.
- A 3-4 page report containing:
  - Learning curves (accuracy and loss).
  - F-scores.
  - Confusion matrices.
  - Variations tried.
  - Inferences.
- Zip the code and report as **3.zip**.

4 Multi-Head Classification (35 marks)

4.1 Motivation

The objective of this problem is to design non-sequential networks.

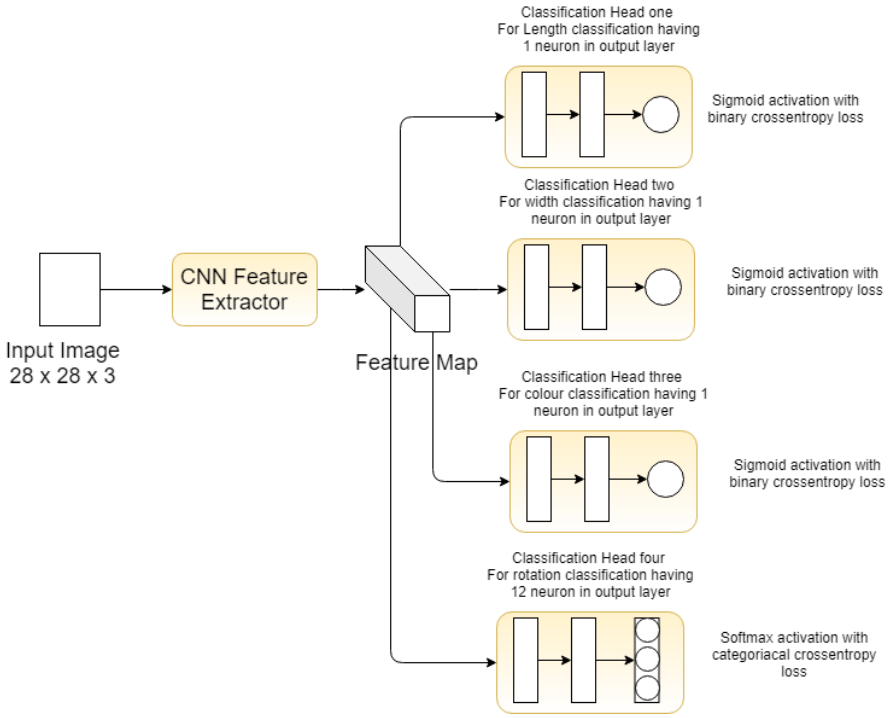

Figure 2: Network Diagram

4.2 Problem Statement

Design a non-sequential convolutional neural network for classifying the line dataset. This network will have 4 outputs based on the 4 kind of variations(length, width, color, angle). You are required to divide your network architecture into two parts a) Feature network and b) Classification heads. The feature network will be responsible for extracting the required features from the input and attached to it would be the four classification heads one for each variation. The network is shown in figure 2.

The first 3 classification heads are for 2 class problems namely length, width and color classification. In all these the final layer contains a single neuron with a sigmoid activation followed by binary crossentropy loss.

The last classification head is a 12 class problem for each 12 angles of variation. In this the final layer contains 12 neurons with softmax activation and Categorical Cross entropy loss.

4.3 Requirements and Hints

1. The number of neurons, activations and loss functions at the end of each classification head should be the same as directed above.

2. You are free to design the feature extractor and the classification heads.

3. The whole network, the feature extractor and the 4 classification heads, should be end to end trainable.

4. You may have to formalize a strategy for calculating the accuracy of the network.

5. You are required to calculate the accuracy of each of the 4 classification heads separately and also an aggregated accuracy of the whole network.

At the very least, you should be able to train a ConvNet that gets at least 70% accuracy on the test set. This is just a lower bound - if you are careful it should be possible to get accuracies much higher than that! Extra credit points will be awarded for particularly high-scoring models or unique approaches.

4.4 Submission

- Code and Model for the network.
- A 3-4 page report containing:
  - Learning curves (accuracy and loss).
  - F-scores.
  - Confusion matrices.
  - Variations tried.
  - Inferences.
- Zip the code and report as **4.zip**.

5 Network Visualization (35 marks)

5.1 Motivation

In this section you will learn a few ways visualization of the outputs in a neural network. These visualizations will help you understand the working of these networks and can help you debug and optimize the performance.

5.2 Problem Statement

Implement the below three network visualization techniques:

5.2.1 Part 1: Visualizing Intermediate Layer Activations

Intermediate layer activations are the outputs of intermediate layers of the neural network. You are required to plot the intermediate activations of the layers of the neural networks made in section 1&2 for atleast 6 images.

5.2.2 Part 2: Visualizing Convnet Filters

In this part you have to visualize the filters in the convolutional neural network. This can be done by running Gradient Descent on the value of a convnet so as to maximize the response of a specific filter, starting from a blank input image. You are required to plot the filters of the layers of the neural networks made in section 3&4.

5.2.3 Part 3: Visualizing Heatmaps of class activations

In this part you are required to plot heatmaps of class activations over input images. This will help you visualize the regions of the input image the convnet is looking at. A class activation heatmap is a 2D grid of scores associated with a particular output class, computed for every location for an input image, indicating how important is each location is with respect to that output class. You are required to plot the heatmaps for atleast 6 images for networks made in section 3&4.

5.3 Requirements

1. Your code should be able to take an image or layer name or both and generate the visualizations.

2. For each part make a report containing inference describing your understanding of what the network has learned in the initial layers till the last layer.

3. For more informations about the 3 parts, please read **Understanding your Convolution network with Visualizations** article.

5.4 Submission

- Code for the task.

- A 3-4 page report containing network visualisation and the inferences about the learning of the networks.

- Zip the code and report as **5.zip**