Assignment 5

November 2020

Instructions

- This assignment should be completed individually.
- Do not look at solutions to this assignment or related ones on the Internet.
- The files related to the assignment are present in `lab5-rollno.zip` folder. Extract it and upload it on moodle in the same .zip format after completion and after replacing the string "rollno" with your actual roll number. For example, if you roll number is 00405036, then single zip folder that you will upload will be named "lab5-00405036.zip". Also collate all the CS337 based theory solutions into ONE pdf file named `answers.pdf`. Include `answers.pdf` inside the zip folder mentioned above and submit the zip folder.
- Answers to all subjective questions need to be placed in single pdf `answers.pdf` including all plots and figures and uploaded.
- Only add/modify code between `TODO` and `END TODO` unless specified otherwise. You must not import any additional libraries.
- Files to submit - `layers.py`, `nn.py`, `trainer.py` and `model.p`
- This Assignment carries a total of **34** marks for CS335 Lab and **12** marks for CS337 theory.

1 CS 335: Lab problems

1.1 Implementing ReLU and Softmax functions

In this part, you will complete the various activation functions namely `relu_of_X`, `softmax_of_X`, `gradient_relu_of_X`, `gradient_softmax_of_X` methods in `layers.py` file. (1+1+1+1 marks)

1.2 Implementing the blocks of a neural network

In this assignment you need to implement a CNN in numpy. For this, you need to complete code in the following three files-

(a) `layers.py` - This file contains the classes corresponding to the layers you would need to build your neural net. Complete the functions `forward_pass` and `backward_pass` for the following layers-

(i) FullyConnectedLayer - A simple feed forward layer having weights of size `in_nodes x out_nodes` and a bias parameter. The forward pass involves multiplying and adding the parameters and input appropriately and applying the activation function. The backward pass involved can be derived using some matrix calculus. (2 marks)

(ii) ConvolutionLayer - This layer has filters of size `(out_depth, in_depth, filter_rows, filter_cols)`. The sizes of input and output matrices are mentioned in the base code. The forward pass involves a patchwise cross-correlation of the input with the filters. The backward pass needs to be figured out by you. Note that both forward and backward pass need to be vectorized to have atmost 2 nested loops for efficient computation. (4 marks)

(iii) MaxPooling - This layer pools the values in a window of size `(filter_row,filter_col)` and replaces them by the maximum value in that window. The backward pass through this for each such window would be a sparse matrix having exactly 1 value equal to 1, multiplied by the incoming gradient. Note that both forward and backward pass need to be vectorized to have atmost 2 nested loops for efficient computation. (3 marks)

(iv) AvgPooling - Similar to MaxPooling this layer pools and replaces the values in a window by the average of all values in that window. Note that both forward and backward pass need to be vectorized to have atmost 2 nested loops for efficient computation. (3 marks)

(v) Flatten - This is a helper layer which converts a 3-D array into a 1-D array. (2 marks)

In order to vectorize the code properly you may find the numpy functions `tile, transpose, max, argmax, repeat, dot, matmul` to be useful,

(b) `nn.py` - This file contains the class corresponding to the neural net. Complete the functions `feedforward`, `backpropagate` and `compute_loss` by calling upon the functions defined in `layers.py` (2+2+1 marks)

(c) `trainer.py` - This file contains the code to train your network on multiple datasets. Complete the function `train` by calling upon the functions defined in `nn.py` to train the network on a particular dataset. (2 marks)

In order to validate your functions we have provided the file `check_gradients.py` which numerically computes the gradients for a single layer neural network and compares them with your computations. You can use this file to check whether your gradients are indeed correct.

1.3 Applications of Neural Network

Once you have completed the implementation for the neural network, you shall use it to train a classifier on some datasets. You need to initialize the network in `trainer.py` to be the minimal topology network for achieving the required accuracies. The datasets are given below -

1. **XOR dataset** - This dataset is a 2 dimensional dataset for the XOR function. The points in the first and third quadrant are marked as 1 and the other two as 0. You need to use a feed-forward network to achieve a test accuracy greater than 90% across multiple random seeds. (2 marks)

2. **Circle dataset** - This is a dataset where the points inside a circle of a particular radius are marked 1 and the others are 0. You need to use a feed-forward network to achieve a test accuracy greater than 90% across multiple random seeds. (2 marks)

3. **MNIST** - This dataset is an image dataset consisting of handwritten digits between 0-10. You need to use a feed-forward network to achieve a test accuracy greater than 90% across multiple random seeds. (2 marks)

4. **CIFAR10** - This dataset is an image dataset having 10 classes. You need to train a CNN having atleast 1 convolution layer for this task, achieving a test accuracy $> 35\%$. Save the trained model in a file called `model.npy` and include it in your submission. **This training can potentially require a few hours, so get started on your assignment early to have ample time for hyperparameter tuning etc**. (4 marks)

For all the datasets report the topology, as well as all hyperparameters and seed values you have used to train the networks in the report. For XOR and Circle datasets, you have to report minimal topology required to achieve high accuracy. For example, for a linearly separable binary classification dataset the minimal topology giving a high accuracy will have only a single node.
Note that due to large sizes, we have included the script `download_data.sh` which when run shall download the datasets into the correct directory.

This task can be checked by running the command -

```
$ python3 main.py --dataset <dataset> --verbose --seed <seed>
```

Here dataset is one of `MNIST, CIFAR-10, XOR, circle`.

2 CS337: Theory Problem

In this problem, assume that every input image has the same size: 1024×1024**.**

1. **Task 1:** In the class we discussed the Convolutional and pooling layers of a CNN and motivated the kernel (sparse interaction, patches, strides) for classification of images. Suppose our input image contains exactly one vehicle (either a bicycle or a car or a motorbike or any one of N vehicles) and we have trained a CNN-based network to distinguish between images based on the kind of vehicle that the image contains. Specifically, the output layer of our network consists of a soft-max layer that can classify an image into one of N vehicular categories. Thus, our CNN-based network is trained to distinguish that the image in the first row, first column of Figure 1 is a car and that the image in the first row, second column of Figure 1 is a motor-bike.

   Roughly depict or describe for each of the images of the 3 rows of Figure 2 what the different components/operators of our trained CNN-based network would extract in the intermediate layers to help distinguish between images containing different vehicles, and how the properties of CNNs would help handle differences in the objects' location and size.[1]Here all the images are actually reproduced from the first row of Figure 1. (5 marks)

[1]that is, give correct results even when the location or size of an object in the image changes

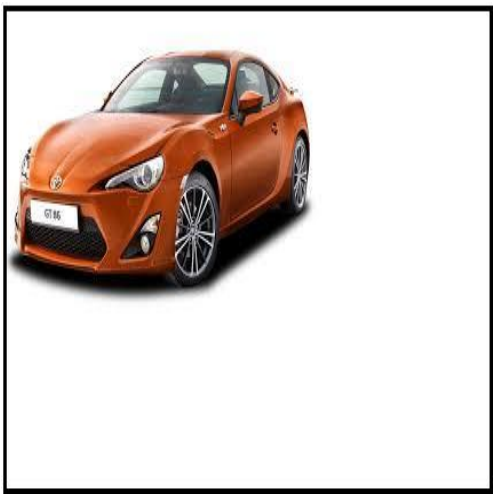
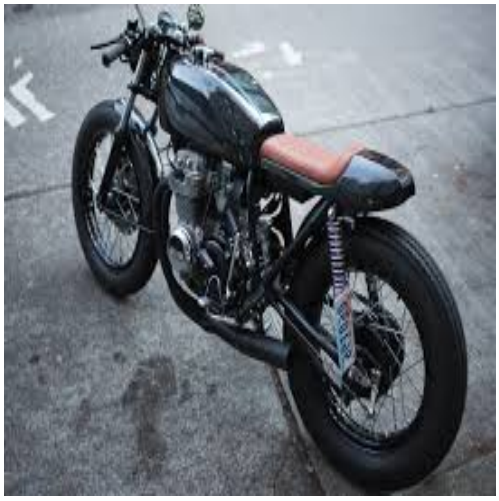
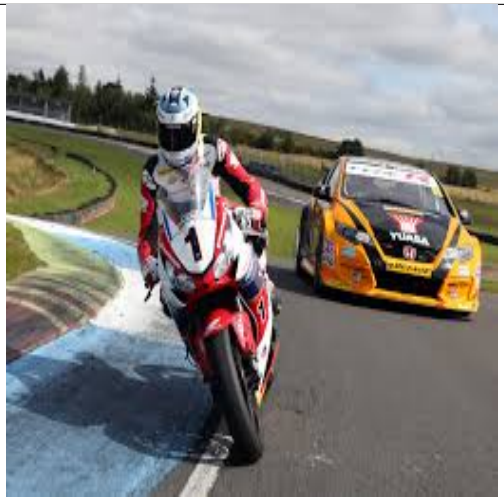
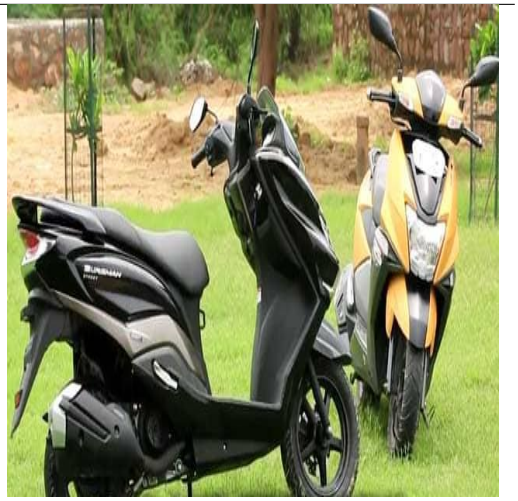
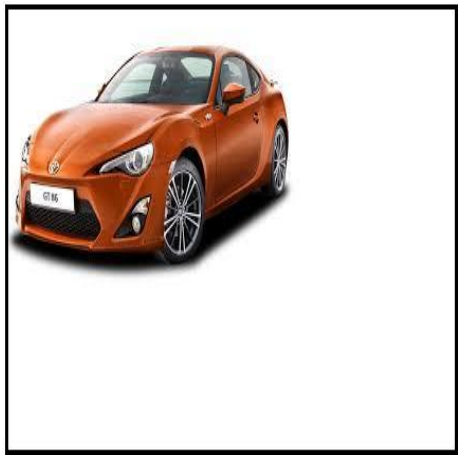
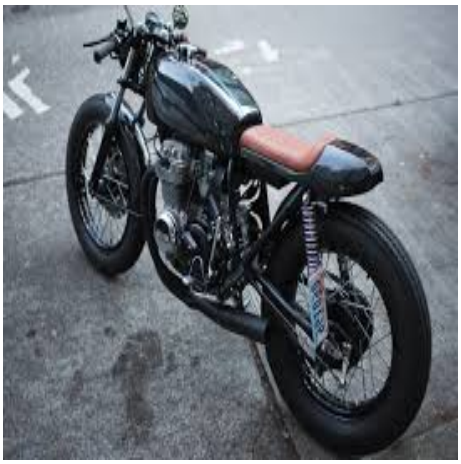
Task 1: Detecting Image Containing Single Object		
Task 2: Detecting Separated Objects		
Task 3: Detecting Overlapping Objects		

Figure 1: List of 1024×1024 images for the three tasks in this problem.





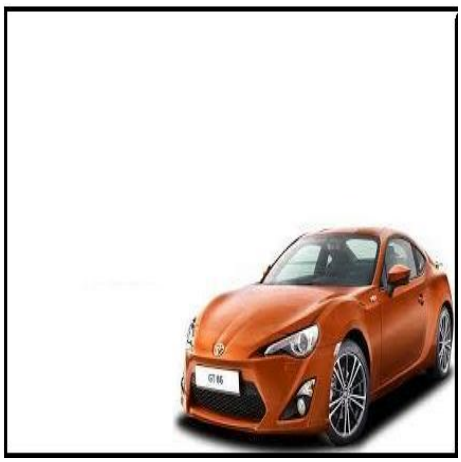


Figure 2: Explain roughly how the CNN-based network helps address/correctly classify the images here. Recall that each image is of the same size, *viz.*, 1024×1024

2. **Task 2:** Suppose we now have images that contain a combination of multiple vehicles but well separated from each other within the same image, as illustrated in the images in the second row of Figure 1. How would you modify your CNN-based network from Task 1 to detect different objects in the same image? Identify any important limitation in your solution. (4 marks)

3. **Task 3:** Now suppose we need to detect overlapping vehicles within the same image. That is, we now have images that contain a combination of multiple vehicles that are NOT well separated from each other within the same image, as illustrated in the images in the third row of Figure 1. This kind of phenomenon is called occlusion. Suggest how you will modify the CNN-based network to handle Task 3. Identify any important limitation in your solution. (3 marks)