CS 8803DL - Deep Learning For Perception

Assignment 3

Exercise 1: Class Model visualization

In this exercise, we will use the method suggested in the "Deep inside convolutional networks: Visualising image classification models and saliency maps" to visualize the class model learnt by a convolutional network. We will use caffe for this exercise and visualize the class model learnt by the "bvlc_reference_caffenet.caffemodel".

In order to generate the class model image, we need to optimize the unnormalized class score w.r.t to the image.

$$arg\ max_I\ S_c(I)\ -\ \lambda\left|\left|I\right|\right|^2$$

This is done by the standard back-propagation as done during the training the training of the network. The key difference is that instead of updating the parameters of the network, we update the image to maximize the score. Another aspect pointed out by the paper is that, the unnormalized Image score needs to be maximized instead of the probability. For this reason, we will be drop the final softmax layer(as the output here is the probability) and maximize the score at the inner product layer "fc8".

The following steps is a guide to get you started:

Step 0: Installation and setup:
Follow the instructions on the course website, to install caffe. Run the classify_test.py to test your installation. Copy the visualize.py into the example folder.

Step 1: Preparing the prototxt:
As the paper suggests, softmax is not a good candidate for optimizing the image for a class. So we will delete the final layer from the deploy.prototxt.

- Copy the deploy.prototxt as deploy_fc8.prototxt present in $CAFFE_ROOT/models/bvlc_reference_caffenet/

- Delete the final layer.
- Add the line "force_backward: true" . This is required for running the backward pass.
- Change the Number dimension from 4 to 1.

Step 2: Forward Pass:

In the visualize.py, the *caffe_data* has been initialized with a randomly generated image and stored as a 4D blob.
Use this as an input to the *net*, to perform the forward pass.
The forward pass is performed using the function
net.forward()

Step 3; Backward Pass:
Perform the backward pass using net.backward().
Use *caffeLabel*, which is formatted as 1 x 1000 x 1 x 1 four dimensional blob.
All values are initialized with zero except for the index corresponding to the CAT class.

Step 4: Gradient Ascent
Use the gradient at the bottom most layer to update *caffe_data.*

Step 5: Visualization
Use the visSquare function to visualize *caffe_data.*

Deliverable:
Submit the completed visualize.py along with the generated image. Write up observations on the results.

Exercise 2: Class Saliency extraction
In this exercise, we will implement the class saliency extraction described in section 3.1 in the paper. The core idea behind this approach is to use the gradients at the image layer for a given image and class, to find the pixels which need to be changed the least i.e, the pixels for which the gradients have the smallest values. Also since our image is a 3 channel image, for each pixel, there will three different gradients. The maximum of these three will be considered the class saliency extraction.

Following are the steps to get started on this exercise:

Step 1: Setup:
Complete step 0 and 1 of excercise 1. Copy class_saliency_extraction.py into the examples

Step 2: Forward Pass:
In the class_saliency_extraction.py, the *input_image* has been initialized with the image of a cat.
Use this as an input to the *net*, to perform the forward pass.

You could either preprocess the input_image and store it as a blob and perform forward pass as done in the previous exercise, or use net.prediction() to do the same directly on the input_image

Step 3; Backward Pass:
Perform the backward pass using net.backward().
Use *caffeLabel*, which is formatted as 1X1000X1X1 four dimensional blob.
All values are initialized with zero except for the index corresponding to the CAT class.

Step 4: Class Saliency extraction
As described in section 3.1, extract the class saliency and store it in the variable saliency.

Step 5: Visualization
Display the image.

Deliverable:
Submit the completed class_saliency_extraction.py along with the generated image.
Write up observations on the results.

Exercise 3: Understanding backpropagation:

In this exercise, we will visualize the gradients at different levels and analyse the data. Use the file backprop_analysis.py for this. Plot each layer's gradient.

Deliverable:
Completed python file, along with a short write up on your observations that you can make.

Final deliverable:
The three completed python file.
Report containing a images from each exercise and the short write-up for all of the exercises.
We shall be running the using an automated script. For this please follow these steps:

1. Use relative paths.
2. Submit the python files as <last_name>_<first_name>_<orginal_filename>.py

3. In your script, save all the generated figures with <last_name>_<first_name>_<figure>. The script should terminate on its own.

Grading Policy:

1. Late submission: -10% per day
2. Each exercise is ⅓ of the total score.
3. If code doesn't run, -50% for that particular exercise
4. Report will be evaluated appropriately