HW3 18739

Due Date: April 17th Before Class

1 Introduction

In this homework, you will be implementing explanation methods in 2 papers covered in class: influence-directed explanations[1] and integrated gradients [4]. You should revisit both papers and relevant slides before you start. You will be explaining a pretrained model, VGGNet [3] for ImageNet.

2 Setup

In this homework, it is strongly suggested for you to use **Python 2** (instead of Python 3). As a result, you can either create a new conda environment in Python 2 or install the required packages(you will be using `Keras` with `Theano` backend). You should be able to run this homework on your laptop since there will be no training involved.

2.1 Data

All the test image data are provided for you via a `npy` files in `test_data.py`. You can download the pretrained model into `models` folder from here. The Keras model architectures can be found in `models.py`.

3 Implementation Clarifications

3.1 Linear layer substitution

Notice that the activation of the final dense layers of the model are modified in to "linear" instead of "softmax". This is because a unit will receive less influence to class A if it simultaneously gives more influence to class B according to the definition of softmax, so it tends to decrease the score of the smaller value. As a result, linear substitution is often used so the classes can be reasoned about more independently.

3.2 A "top neuron"

Since each neuron is a square matrix, there can be multiple ways of defining a "top", or "most influenced" neuron. For example, if there are 20 neurons is of

size 28×28(giving an attribution size of $20 \times 28 \times 28$), the top neurons could be determined by the mean of the size 28×28 attributions of each neuron , or the max of the attributions. In this homework, please use the latter. For example, if neuron 0 has the largest "max attribution" of 784 attribution values, then neuron 0 is the most influential.

3.3 Visualization of Influence of Internal Neurons

Visualizations of influence for internal neurons can be computed using saliency map[2], which is the gradient of neuron activation wrt input pixels. However, in this homework, to make things simpler, you will be using the same attribution method for calculating the influence of individual pixels(Integrated Gradients) or internal neurons towards the output. Instead, you will be calculating the attributions of input pixels towards the specific neuron(s) you picked as the most influential. By scale pixels of original image accordingly, we will able to project the explanations of internal neurons back to the pixels.

4 PART I: Understanding of Measures of Influence (20 pts)

4.1 Influence-Directed Explanations (5pts)

After reading both papers, answer the following questions about [1]:

1. What is a quantity of interest, what is a distribution of interest? Please provide brief definition.
2. In the query "what is the influence of neuron 0 toward the cat class?", what is the distribution of interest and quantity of interest?
3. In the query "what is the influence of pixel (5,0) toward a cat image C, compared to a dog image D?", what is the distribution of interest and quantity of interest?

4.2 Aumann-Shapley Values (15 pts)

As is introduced in lectures, both integrated-gradients and influence-directed explanations define influence in terms of Aumann-Shapley Value. It is a notion borrowed from economics. You can read more about it here. There are mainly two differences between Definition 1 in [1] and equation (1) in [4]: The latter has a "difference term" to satisfy its completeness axiom, and the former has additional terms for "quantity of interest" and "distribution of interest". As result, both can be parametrized under one class using the same approximation method using equation (3) in [4].

Please complete the function `get_attributions()` in `AumannShapley.py` which returns the attributions for the quantity of interest Q.

5 PART II: Focused Explanations of Slices (25 pts)

5.1 Quantity of Interest: Absolute (5 pts)

Complete `absolute()` in `ExplanationForLayer` Class in `Explanation.py`. After your implementation, test your implemention of both `get_attributions()` and `absolute()` using the starter code to generate a integrated gradients explanation for a starfish image.

5.2 Explanations (20 pts)

Using the `Explantion.py` class, please provide explanations to answer the following queries:

1. What are the 10 most influential neurons in Conv4_1 (layer 19) for a specific instance of "starfish"? Use the first instance as the specific instance.
2. What are the 10 most influential neurons in Conv4_1 (layer 19) for a class of images "starfish"?

For each query, return the 10 neuron indices and visualize the influence of top 3 neurons on the first instance. (You can use `VisualizerTiler` to visualize images side by side.) How does the explanations of using internal neurons compared with the explanations of integrated gradients?

6 PART III: Comparative Explanations(25 + 10 pts)

6.1 Quantity of Interest: Comparative (5 pts)

Complete `comparative()` in `ExplanationForLayer` Class in `Explanation.py` which returns the comparative quantity for 2 classes.

6.2 Explanations (20 pts)

Please provide explanations to answer the following queries:

1. What is the most influential neuron in Conv4_1 (layer 19) for the sports car class?
2. What is the most influential neuron in Conv4_1 (layer 19) for the sports car class compared with convertible class?

For both queries, calculate your attributions using `sports_cars` and `convertibles` and visualize the most influential neuron on 3 test images in `test_sports_cars`. Report the neuron index for each query. What is the difference of two explanations, What concept is revealed by each explanation?

6.3 Extra Credit (10 pts)

This is an extra credit section, it is advised for you to finish Part IV first before you work on this section.

In this section, you should find at least 10 images each for 2 similar classes (other than sports car and convertibles) within the 1000 classes of imagenet and then replicate your result in 6.2. The class number and labels can be found here. In `test_data.py`, you can find functions that read image from a url or resize a image into appropriate sizes.

Is your explanations/visualizations making sense in terms of the difference between two image classes? Explain your results.

7 PART IV: Model Compression (30 pts)

In this section, you will be implementing section 3.2 of [1]. Read the section carefully and perform the following 2 experiments:

1. Replicate Figure 3 in [1] for class "chainsaw". Plot the corresponding heatmap. You can make use of heatmap function in `seaborn` http://seaborn.pydata.org/generated/seaborn.heatmap.html.

2. Replicate Column 1 (class recall for original model) and Column 3(class recall using a compressed model) of Table 1 [1]. The 5 classes can be found in `test_data.py`. You don't have to calculate the values for column 2(class recall using activation-compressed models), nor do you need to calculate the precision values (since they are all 1). For each class, the recalls of the compressed model is calculated using the best "top" and "bottom" combination using the parameters in Figure 3. Make sure your values (recalls of original model and recalls of compressed models for each class) are printed after executing the cell. This part may take a while to run, make sure your code works for one class first.

In this section, you can write a function for calculating the recall and another function for compressing a model to keep your code modular.

Submission

You have to submit the files according to the following procedures:

1. Rename all the ipynb files (hw3_explain.ipynb) as ⟨your_andrew_id⟩_hw3.ipynb.

2. Make sure that you have run all your program in all the cells before you submit so that the results/plots can be seen by simply opening the file. In other words, the grader won't have to run the program to see your results. If you completed the extra-credit section, please submit a txt file with your image urls as well.

3. Please comment your code.

4. Please cite all your references in the `ipynb` file.

You don't have to rename `py` files, however, do **put all files into ⟨your_andrew_id⟩_HW3 folder**, before you zip the folder into ⟨your_andrew_id⟩_HW3.zip and submit the zip file on Canvas.

References

[1] K. Leino, L. Li, S. Sen, A. Datta, and M. Fredrikson. Influence-directed explanations for deep convolutional networks. *arXiv preprint arXiv:1802.03788*, 2018.

[2] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[4] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017.