Assignment Deep Learning

1 Introduction

Convolutional neural networks are one of the most successful applications of deep learning. In their application as image processors, CNNs offer an interesting opportunity to study how neural networks work and how they find solutions. The key to achieving this is *visualization*.

Understanding the inner workings of neural networks is often referred to as *interpretability* research. There are two main tracks; feature *attribution* and feature *visualization*. The former aims at discovering what parts of a network are activated for a particular example, while the latter directly inspects the outputs of filters and layers. The former is typically performed using an approach known as CAM - Class Attribution Map. An example of an available library for CAM is torch-cam. There is a `keras` example in the git repository for visualization, under `Resources/keras/visualizing_what_convnets_learn.py`. Several libraries also exist, but are rather involved to use. You *may* use libraries, even for the distinguished grade. However, it is also almost certainly more work and more difficult to understand than implementing your own version. For the distinguished grade (VG) you *must* implement *gradient ascent*.

In this assignment, you will investigate interpretability in a neural network – either one you have trained yourself or a pretrained model from the internet (e.g. from huggingface, pytorch itself, or any of many other sources).

For the passing grade (G) a few layers should be visualized for some examples and an attempt be made to reason about what the layers do. It's enough to investigate attribution for the passing grade.

For the distinguished grade (VG) you should instead provide an implementation of an *activation maximization* approach. The feature maps should be visually interpreted with the same speculative framework as in the distill.pub papers, in other words if they look similar we assume they are similar.

2 Choose a model

There are a large number of pretrained CNN models, to name a few: `AlexNet`, `vgg16`, `Inception`, `Xception` and `ResNet`. Some of these models are gigantic and can be difficult to understand, but their performance is highly tuned and optimized. You can also choose to build your own CNN, but make sure to use several convolutional layers in order to inspect the difference between shallow and deep layers. Depending on what hardware you have available, training your own CNN may be prohibitively expensive. Choose a pretrained model in that case.

3 Research

Lectures, examples in the git-repository and links are available. There are numerous tutorials online, but all have their own quirks and many are too old to be directly relevant (i.e. you can't use the code directly and it may even confuse you since things have changed in the frameworks). From my experiments, LLMs can help but also don't seem to give good code – they look like the previously mentioned outdated tutorials. While the assignment is to be completed individually, you are encouraged to discuss the problem and code with each other and me!

For G, I recommend looking at torch-cam.

For VG, I recommend porting the `keras` example in the git-repository to `pytorch`. This is in of itself an excellent exercise that will teach you a lot about tensorflow, keras and pytorch. It does pose a programming challenge and requires reading of manuals, scouring the web for clues and trying things out. Judicial use of LLMs can help, but be careful! Make sure you understand the code.

4 Results

The results should be a mix of text, explaining what your thoughts are and introducing your method, and code/graphs that show the results and supports your arguments. Use all we have learned in previous courses to produce a report– as a webapp, a jupyter notebook or an interactive program. Whatever form you choose, remember to make it obvious what I should be looking at and consider the readability of your work.

The report will be graded with a few requirements in mind:

G

At least two layers are visualized with respect to their activations. At least two images were tested and attribution for those layers chosen is motivated. The report should be reasonably formatted and accessible (ie as a local app, a jupyter-notebook or a locally hosted webapp – I should be able to run/view it on my computer).

VG

Multiple activation maps for different layers/filters. The solution must contain an implementation of *gradient ascent* and optimize the input image for *activation maximization*. The report should be well formatted and accessible, regardless of what form you choose for the report. Try to motivate what the filters do in the same vein as the `distill.pub` papers.

DON'T HESITATE TO ASK FOR GUIDANCE AND TUTORING!