Advanced Machine Learning: Final Project

Aman Gulati
ag3743

Joaquim Lyrio
jc4637

Ricardo Pommer
rap2194

Henrique Saboya
hs2923

December 11, 2017

Abstract

Neural networks, although empirically validated, remain a theoretically opaque tool. Once we have trained a particular architecture, what within its layers, neurons, or channels, drive its classification performance? How do we interpret the results beyond out-of-sample loss measures? Feature visualization attempts to elucidate this question by, generally speaking, projecting intermediate layers to pixel-space. In this report, we review and consolidate different methods of feature visualization, their definitions, theoretical approaches and results on three neural network architectures and datasets.

0.1 Definitions

As any young field, neural networks and their interpretability has yet to agree on definitions for a vast number of methods and parameters. Here, we provide our own that may at times conflict with other authors' but will hopefully clarify the differences and similarities between each.

0.1.1 Feature Visualization:

We refer to **feature visualization** as any projection onto pixel-space that relays information about a (generally hidden) subset of the network's architecture. The distinction will become increasingly important as we delve into

the details of each technique. Furthermore, some authors (Olah et al., 2017) distinguish between **feature visualization** and **attribution**. The former defined as "generating" maximum activation images, while the latter looks for specific inputs that generate layer (or node)-specific activation. We draw no such distinction.

0.1.2 Weight Projection:

The simplest and canonical form of feature visualization: projecting **trained** weights of a hidden node/layer directly onto pixel-space, regardless of dimension. The only restriction we impose is that they be rectified via reLu as to avoid negative values.

0.1.3 Filtered-Input Projection:

The next-simplest approach is to pass a given image through a **trained** node/layer and project the filtered activation onto pixel-space. Again, we only require reLu to guarantee a positive signal but make no restriction on the dimension of the projection. The analogy with the physical world is quite immediate, we shine our input image through a trained filter, and see what comes out on the other side. Since we know what our original image looks like, seeing the difference between input-output should give us some insight about the filter. Comparing with **filtered-input projection** looks at the image *through* the filter, while **weight projection** looks at the filter directly.

0.1.4 Deconvolutional Projection:

A slight change on filetered-input projection, **deconvolutional projection**

1 A Simple (But Useful) Example

To motivate our survey, we begin with a toy model and dataset of the letters "E", "F" and "L". The network and dataset are based on (Orbanz, 2017) as presented in lecture notes. We generate 2,000 images of each category and add noise distributed as $\mathcal{N} \sim (0, 0.2)$, with all negative noise samples regularized to 0. Following, we train a 1-hidden layer neural network classifier, as shown in *Figure 1.1.* Note that we do not include biases or convolutions– the hidden neurones are only sigmoid functions. This allows us to project

the weights directly as:

[ADD FIGURES]

[FIGURE ONE WILL INCLUDE DETAILS ABOUT VALIDATION AND WHATNOT]

This example informs of what would be the "ideal" of feature visualization: knowing with some certainty what the network has learned and how it performs classification through an image. In our weights it is self-evident that the network learns the two discriminant features needed to sort the three categories. Node h_1 has positive weights highly concentrated on the bottom leg of the letter L and E appear. If this node is NOT activated by the input image, our classifier will know that it has seen an F. Node h_2, on the other hand, concentrates its weights on the appearance of the two upper legs of the letter "E". A positive activation for a given input lets

[ADD FORMULAS FOR BOTH NODE EXAMPLES]