Advanced Machine Learning: Final Project

Aman Gulati
ag3743

Joaquim Lyrio
jc4637

Ricardo Pommer
rap2194

Henrique Saboya
hs2923

December 12, 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. Feature visualization often produces compelling images, but their contribution to the interpretation of a neural network depends on clear definitions of each procedure. Here, we provide our own. Although they may at times conflict with other authors', we hope these will add clarity to our discussion

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 ar-

chitecture. This is a particular approach to the general problem of **neural network interpretability**. Some authors (Olah et al., 2017) distinguish between **feature visualization** and **attribution**, defining the former as the creation of 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 Features:

Subsets of any data set. The heuristic behind feature visualization is that we are visualizing the subsets of our input that drive the network's classification performance.

0.1.3 Feature Map:

The output of a single kernel applied to the previous data. This term can be used interchangeably with **activation**.

0.1.4 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.

0.1.5 Feature Map Projection:

The next-simplest approach is to pass a given image through a *trained* node or layer and project the filtered activation onto pixel-space. This is, from our above definition, projecting **feature maps** of intermediate layers. Again, we make no restriction on the dimension of the projection, since we expect feature maps to have different dimensions depending on its position in the layer's architecture.

A useful 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. Contrastingly, **weight projection** looks at the "filter" directly, while **feature map projection** looks at the image *through* the "filter" (using filter loosely and

in reference to a physical filter and not necessarily a node in a convolutional layer).

0.1.6 Deconvolutional Projection:

Unlike feature-map 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 nodes are simply performing matrix multiplication between the weights and the input image, before passing through a reLu activation function. This simple structure allows us to project the weights directly as:

[ADD FIGURES]
[FIGURE ONE WILL INCLUDE DETAILS ABOUT VALIDATION AND WHATNOT]

This would be the "ideal" of feature visualization: knowing with some degree certainty what the network has learned and how it performs classification through an image. In our weight-projection image, 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 where the bottom leg of the letter L and E appear. If this node outputs a signal near-zero, the proceeding logits layer 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" and "F". If this node outputs a near-zero signal, the proceeding layer will know the image is an L. If both h_1 and h_2 pass a positive signal, the proceeding layer will know the image is an "E".

[ADD FORMULAS FOR BOTH NODE EXAMPLES]

2 MNIST: Deconvolutions and Feature Map

We now take a further step in dataset and network complexity. The MNIST dataset is still grayscale, but of significantly higher in than EFL. Similarly, our network architecture now has two convolutional hidden layers, (with reLu and max-pooling), followed by a fully connected layer and a sigmoid function. After training, we attempt weight projection for randomly selected filters, seen in [FIGURE].

[INSERT FIGURE]

Weight projection, for this dataset, is not nearly as informative as it was for EFL data. [ELABORATE ON WHAT WE SEE]

We then consider feature map projection.

[COMMENTS ON WHAT WE SEE].

3 Dog, Muffin or Fried Chicken?

4 Dreams

5 Application to CNN design

6 Discussion

7 Further Work