Information Visualization

eXplainable Artificial Intelligence

Dr. David Koop
eXplainable Artificial Intelligence

Training data set
- Label: Apple
- Label: Cake

Features:
- Color
- Shape
- Smell...

Explaining data

Learning Model
(Using a ML algorithm)

Prediction label: Cake

Explaining “model facts”: performance, limitations, output, etc.

XAI focus: explaining model decision

New instance

[Liao et al.]
Feature Visualization

Edges (layer conv2d0)
Textures (layer mixed3a)
Patterns (layer mixed4a)
Parts (layers mixed4b & mixed4c)
Objects (layers mixed4d & mixed4e)

[C. Olah et al., 2017]
Feature Visualization vs. Attribution

**Feature visualization** answers questions about what a network—or parts of a network—are looking for by generating examples.

**Attribution** studies what part of an example is responsible for the network activating a particular way.

[C. Olah et al., 2017]
Feature Vis by Optimization

• "[W]hat kind of input would cause a certain behavior"
• Start from random noise and iteratively tweak (using derivatives)

- What are the objectives? (Where are we going?)
  - Neuron, channel, layer (has DeepDream "interesting" objective)

[C. Olah et al., 2017]
This article focuses on feature visualization. While feature visualization is a powerful tool, actually getting it to work involves a number of details. In this article, we examine the major issues and explore common approaches to solving them. We find that remarkably simple methods can produce high-quality visualizations. Along the way we introduce a few tricks for exploring variation in what neurons react to, how they interact, and how to improve the optimization process.

Feature Visualization by Optimization

Neural networks are, generally speaking, differentiable with respect to their inputs. If we want to find out what kind of input would cause a certain behavior — whether that's an internal neuron firing or the final output behavior — we can use derivatives to iteratively tweak the input towards that goal.

While conceptually simple, there are subtle challenges in getting the optimization to work. We will explore them, as well as common approaches to tackle them in the section "The Enemy of Feature Visualization."
Why not Examples?

Dataset Examples show us what neurons respond to in practice.

Optimization isolates the causes of behavior from mere correlations. A neuron may not be detecting what you initially thought.

Baseball—or stripes? 
mixed4a, Unit 6

Animal faces—or snouts? 
mixed4a, Unit 240

Clouds—or fluffiness? 
mixed4a, Unit 453

Buildings—or sky? 
mixed4a, Unit 492

---

Why not Examples?

Optimization can give us an example input that causes the desired behavior—but why bother with that? Couldn’t we just look through the dataset for examples that cause the desired behavior?

It turns out that optimization approach can be a powerful way to understand what a model is really looking for, because it separates the things causing behavior from things that merely correlate with the causes.

For example, consider the following neurons visualized with dataset examples and optimization:

- Baseball—or stripes?
- Animal faces—or snouts?
- Clouds—or fluffiness?
- Buildings—or sky?

Optimization also has the advantage of flexibility. For example, if we want to study how neurons jointly represent information, we can easily ask how a particular example would need to be different for an additional neuron to activate. This flexibility can also be helpful in visualizing how features evolve as the network trains. If we were limited to understanding the model on the fixed examples in our dataset, topics like these ones would be much harder to explore.

On the other hand, there are also significant challenges to visualizing features with optimization. In the following sections we’ll examine techniques to get diverse visualizations, understand how neurons interact, and avoid high frequency artifacts.

---

C. Olah et al., 2017
Diversity

- Examples can be diverse
- Optimization may give us one very positive takeaway
- Add a diversity term!

[8]

Simple Optimization

Optimization with diversity reveals multiple types of balls. *Layer mixed5a, Unit 9*

Dataset examples

[C. Olah et al., 2017]
Naive Optimization Doesn't Work

Even if you carefully tune learning rate, you'll get noise.

Optimization results are enlarged to show detail and artifacts.
Regularization to Avoid Noise

- Frequency penalization: penalize high-frequency noise
- Transformation robustness: jitter/rotate/scale images and still activate
- Learned priors: learn a prior and try to enforce it

[C. Olah et al., 2017]
The Building Blocks of Interpretability

Interpretability techniques are normally studied in isolation. We explore the powerful interfaces that arise when you combine them—and the rich structure of this combinatorial space.

For instance, by combining feature visualization (what is a neuron looking for?) with attribution (how does it affect the output?), we can explore how the network decides between labels like Labrador retriever and tiger cat.

Several floppy ear detectors seem to be important when distinguishing dogs, whereas pointy ears are used to classify “tiger cat.”

Channels that most support... LABRADOR RETRIEVER TIGER CAT

<table>
<thead>
<tr>
<th>channel</th>
<th>Labrador retriever</th>
<th>Tiger cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>for &quot;Labrador retriever&quot;</td>
<td>1.32</td>
<td>1.54</td>
</tr>
<tr>
<td>for &quot;tiger cat&quot;</td>
<td>-0.70</td>
<td>-1.24</td>
</tr>
<tr>
<td>net evidence</td>
<td>1.63</td>
<td>1.51</td>
</tr>
</tbody>
</table>

[C. Olah et al.]
What Does the Network See?

Applying this technique to all the activation vectors allows us to not only see what the network detects at each position, but also what the network understands of the input image as a whole. And, by working across layers (e.g., "mixed3a", "mixed4d"), we can observe how the network's understanding evolves: from detecting edges in earlier layers, to more sophisticated shapes and object parts in the latter.

Semantic dictionaries give us a fine-grained look at an activation: what does each single neuron detect? Building off this representation, we can also consider an activation vector as a whole. Instead of visualizing individual neurons, we can instead visualize the combination of neurons that fire at a given spatial location. (Concretely, we optimize the image to maximize the dot product of its activations with the original activation vector.)

Activation Vector

Channels

Activation Vector = 886. + 599. + 328. + 303. + ...

Combine Feature Vis and Activation

[C. Olah et al.]
What Does the Network See?

Applying this technique to all the activation vectors allows us to not only see what the network detects at each position, but also what the network understands of the input image as a whole. And, by working across layers (e.g., "mixed3a", "mixed4d"), we can observe how the network’s understanding evolves: from detecting edges in earlier layers, to more sophisticated shapes and object parts in the latter.

Semantic dictionaries give us a fine-grained look at an activation: what does each single neuron detect?

Building off this representation, we can also consider an activation vector as a whole. Instead of visualizing individual neurons, we can instead visualize the combination of neurons that fire at a given spatial location. (Concretely, we optimize the image to maximize the dot product of its activations with the original activation vector.)

Activation Vector = 886. + 599. + 328. + 303. + ...
Spatial Attribution with Saliency Maps

The most common interface for attribution is called a saliency map—a simple heatmap that highlights pixels of the input image that most caused the output classification. We see two weaknesses with this current approach. First, it is not clear that individual pixels should be the primary unit of attribution. The meaning of each pixel is extremely entangled with other pixels, is not robust to simple visual transforms (e.g., brightness, contrast, etc.), and is far-removed from high-level concepts like the output class. Second, traditional saliency maps are a very limited type of interface—they only display the attribution for a single class at a time, and do not allow you to probe into individual points more deeply. As they do not explicitly deal with hidden layers, it has been difficult to fully explore their design space.

We instead treat attribution as another user interface building block, and apply it to the hidden layers of a neural network. In doing so, we change the questions we can pose. Rather than asking whether the color of a particular pixel was important for the “labrador retriever” classification, we instead ask whether the high-level idea detected at that position (such as “floppy ear”) was important. This approach is similar to what Class Activation Mapping (CAM) methods do but, because they interpret their results back onto the input image, they miss the opportunity to communicate in terms of the rich behavior of a network’s hidden layers.

Attribution tends to be more meaningful in later layers. The ‘floppy ear’, ‘dog snout’, ‘cat head’, etc, do mostly what you expect. Surprisingly, the lower snout at mixed4d seems entangled with the idea of a tennis ball and supports “tennis ball” and “granny smith apple.”
This diagram is analogous to the previous one we saw: we conduct layer-to-layer attribution but this time over channels rather than spatial positions. Once again, we use the icons from our semantic dictionary to represent the channels that most contribute to the final output classification. Hovering over an individual channel displays a heatmap of its activations overlaid on the input image. The legend also updates to show its attribution to the output classes (i.e., what are the top classes this channel supports?). Clicking a channel allows us to drill into the layer-to-layer attributions, identifying the channels at lower layers that most contributed as well as the channels at higher layers that are most supported. While these diagrams focus on layer-to-layer attribution, it can still be valuable to focus on a single hidden layer. For example, the teaser figure allows us to evaluate hypotheses for why one class succeeded over the other. Attribution to spatial locations and channels can reveal powerful things about a model, especially when we combine them together. Unfortunately, this family of approaches is burdened by two significant problems. On the one hand, it is very easy to end up with an overwhelming amount of information: it would take hours of human auditing to understand the long-tail of channels that slightly impact the output. On the other hand, both the aggregations we have explored are extremely lossy and can miss important parts of the story. And, while we could avoid lossy aggregation by working with individual neurons, and not aggregating at all, this explodes the first problem combinatorially.

Making Things Human-Scale

In previous sections, we’ve considered three ways of slicing the cube of activations: into spatial activations, channels, and individual neurons. Each of these has major downsides. If...
By using non-negative matrix factorization we can reduce the large number of neurons to a small set of groups that concisely summarize the story of the network.

By using non-negative matrix factorization we can reduce the large number of neurons to a small set of groups that concisely summarize the story of the network.

Factoring into Neuron Groups

INPUT IMAGE

ACTIVATIONS of neuron groups

NEURON GROUPS based on matrix factorization of mixed4d layer

EFFECT of neuron groups on output classes

C. Olah et al.
EFFECT of neuron groups on output classes

This figure only focuses at a single layer but, as we saw earlier, it can be useful to look across multiple layers to understand how a neural network assembles together lower-level detectors into higher-level concepts.

The groups we constructed before were optimized to understand a single layer independent of the others. To understand multiple layers together, we would like each layer's factorization to be "compatible"—to have the groups of earlier layers naturally compose into the groups of later layers. This is also something we can optimize the factorization for.

To understand multiple layers together, we would like each layer’s factorization to be “compatible”—to have the groups of earlier layers naturally compose into the groups of later layers. This is also something we can optimize the factorization for.

Adding InfoVis

INPUT IMAGE

ATTIBUTION BY FACTORIZED GROUPS

MIXED4A

MIXED4D

OUTPUT CLASS

8 groups

6 groups

Align layer factors

tiger cat

positive influence

negative influence

Labrador retriever

beagle

lynx

C. Olah et al.
Progress Reports