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.

CHOOSE AN INPUT IMAGE

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**.

important when distinguishing dogs, whereas pointy ears are used to classify "tiger cat".

CHANNELS THAT MOST SUPPORT ... LABRADOR RE ▼ ⟷ TIGER CAT ▼

feature visualization of channel

...

hover for attribution maps →

net evidence	1.63	1.51	1.19	1.32	1.54	1.72
for "Labrador retriever"	1.22	1.24	1.32	-0.70	-1.24	-0.43
for "tiger cat"	-0.40	-0.27	0.13	0.62	0.30	1.29

REPRODUCE IN A NOTEBOOK

AUTHORS AFFILIATIONS

Chris Olah Google Brain

Arvind Satyanarayan Google Brain

Ian Johnson Google Cloud

Shan Carter Google Brain

Ludwig Schubert Google Brain

Katherine Ye CMU

Alexander Mordvintsev Google Research

PUBLISHED
March 6, 2018

DOI
10.23915/distill.00010

With the growing success of neural networks, there is a corresponding need to be able to explain their decisions — including building confidence about how they will behave in the real-world, detecting model bias, and for scientific curiosity. In order to do so, we need to both construct deep abstractions and reify (or instantiate) them in rich interfaces [1]. With a few exceptions [2, 3, 4], existing work on interpretability fails to do these in concert.

The machine learning community has primarily focused on developing powerful methods, such as feature visualization [5, 6, 7, 8, 9, 10], attribution [7, 11, 12, 13, 14, 15, 16, 17], and dimensionality reduction [18], for reasoning about neural networks. However, these techniques have been studied as isolated threads of research, and the corresponding work of reifying them has been neglected. On the other hand, the human-computer interaction community has begun to explore rich user interfaces for neural networks [19, 20, 21], but they have not yet engaged deeply with these abstractions. To the extent these abstractions have been used, it has been in fairly standard ways. As a result, we have been left with impoverished interfaces (e.g., saliency maps or correlating abstract neurons) that leave a lot of value on the table. Worse, many interpretability techniques have not been fully actualized into abstractions because there has not been pressure to make them generalizable or composable.

In this article, we treat existing interpretability methods as fundamental and composable building blocks for rich user interfaces. We find that these disparate techniques now come together in a unified grammar, fulfilling complementary roles in the resulting interfaces. Moreover, this grammar allows us to systematically explore the space of interpretability interfaces, enabling us to evaluate whether they meet particular goals. We will present interfaces that show *what* the network detects and explain *how* it develops its understanding, while keeping the amount of information *human-scale*. For example, we will see how a network looking at a labrador retriever detects floppy ears and how that influences its classification.

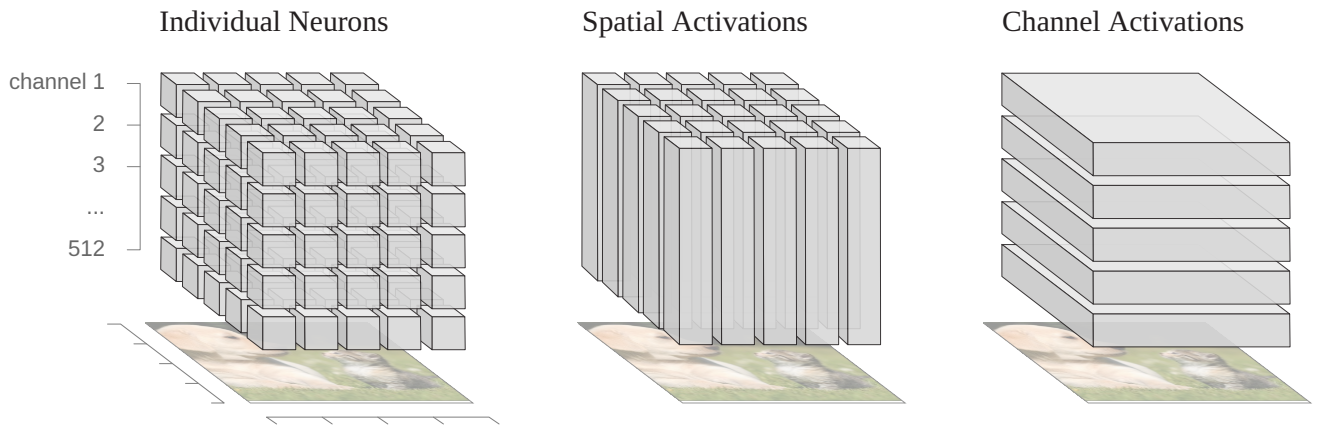
Our interfaces are speculative and one might wonder how reliable they are. Rather than address this point piecemeal, we dedicate a section to it at the end of the article.

In this article, we use GoogLeNet [22], an image classification model, to demonstrate our interface ideas because its neurons seem unusually semantically meaningful. [1] Although here we've made a specific choice of task and network, the basic abstractions and patterns for combining them that we present can be applied to neural networks in other domains.

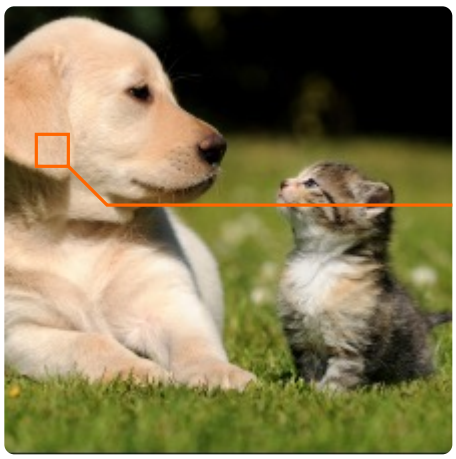
Making Sense of Hidden Layers

Much of the recent work on interpretability is concerned with a neural network's input and output layers. Arguably, this focus is due to the clear meaning these layers have: in computer vision, the input layer represents values for the red, green, and blue color channels for every pixel in the input image, while the output layer consists of class labels and their associated probabilities.

However, the power of neural networks lies in their hidden layers — at every layer, the network discovers a new representation of the input. In computer vision, we use neural networks that run the same feature detectors at every position in the image. We can think of each layer's learned representation as a three-dimensional cube. Each cell in the cube is an *activation*, or the amount a neuron fires. The x- and y-axes correspond to positions in the image, and the z-axis is the channel (or detector) being run.



The cube of activations that a neural network for computer vision develops at each hidden layer. Different slices of the cube allow us to target the activations of individual neurons, spatial positions, or channels.



Making sense of these activations is hard because we usually work with them as abstract vectors:

$a_{4,1}$ = [0, 0, 0, 25.2, 164.1, 0, 42.7, 4.51, 115.0, 51.3, 0, 0, ...]

With feature visualization, however, we can transform this abstract vector into a more meaningful "semantic dictionary".

{ 886. , 599. , 328. ,

There seem to be detectors for **floppy ears**, dog snouts, cat heads, furry legs, and grass. GoogLeNet has a rich variety of ear detectors which help it distinguish between 100 species of dog.

REPRODUCE IN A NOTEBOOK

To make a semantic dictionary, we pair every neuron activation with a visualization of that neuron and sort them by the magnitude of the activation. This marriage of activations and feature visualization changes our relationship with the underlying mathematical object. Activations now map to iconic representations, instead of abstract indices, with many appearing to be similar to salient human ideas, such as "floppy ear," "dog snout," or "fur."

We use optimization-based feature visualization [6] to avoid spurious correlation, but one could use other methods.

Semantic dictionaries are powerful not just because they move away from meaningless indices, but because they express a neural network's learned abstractions with canonical examples. With image classification, the neural network learns a set of visual abstractions and thus images are the most natural symbols to represent them. Were we working with audio, the more natural symbols would most likely be audio clips. This is important because when neurons appear to correspond to human ideas, it is tempting to reduce them to words. Doing so, however, is a lossy operation — even for familiar abstractions, the network may have learned a deeper nuance. For instance, GoogLeNet has multiple floppy ear detectors that appear to detect slightly different levels of droopiness, length, and surrounding context to the ears. There also may exist abstractions which are visually familiar, yet that we lack good natural language descriptions for: for example, take the particular column of shimmering light where sun hits rippling water. Moreover, the network may learn new abstractions that appear alien to us — here, natural language would fail us entirely! In general, canonical examples are a more natural way to represent the foreign abstractions that neural networks learn than native human language.

By bringing meaning to hidden layers, semantic dictionaries set the stage for our existing interpretability techniques to be composable building blocks. As we shall see, just like their underlying vectors, we can apply dimensionality reduction to them. In other cases, semantic dictionaries allow us to push these techniques further. For example, besides the one-way attribution that we currently perform with the input and output layers, semantic dictionaries allow us to attribute to-and-from specific hidden layers. In principle, this work could have been done without semantic dictionaries but it would have been unclear what the results meant.

While we introduce semantic dictionaries in terms of neurons, they can be used with any basis of activations. We will explore this more later.

What Does the Network See?

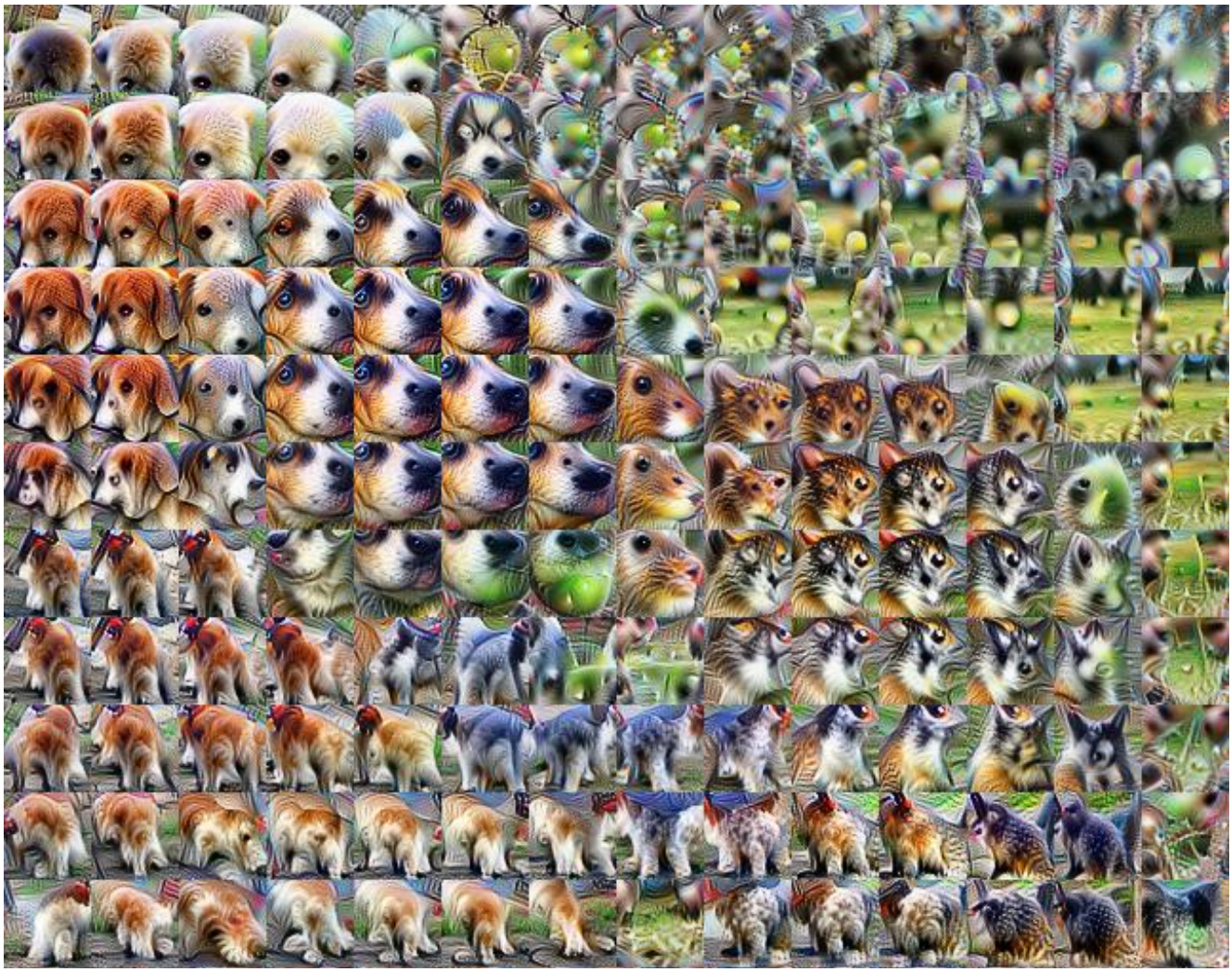
Semantic dictionaries give us a fine-grained look at an activation: what does each single neuron detect? Building off this representation, we can also consider an activation vector as a whole. Instead of visualizing individual neurons, we can instead visualize the *combination* of neurons that fire at a given spatial

location. (Concretely, we optimize the image to maximize the dot product of its activations with the original activation vector.)

886. 599. 328.

Activation Vector Channels

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.

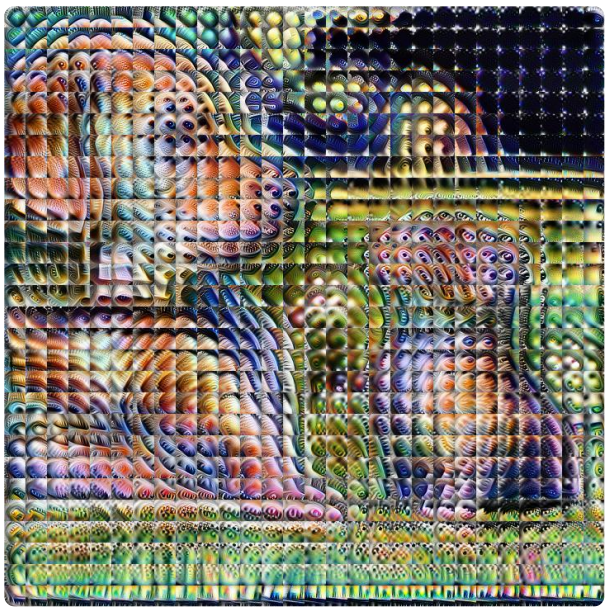


REPRODUCE IN A NOTEBOOK

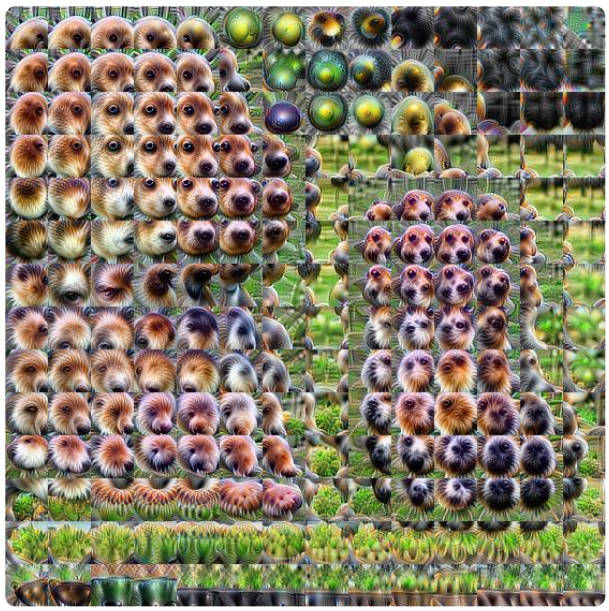
mixed4d

And, by working across layers (eg. “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.

Zoom In Zoom Out



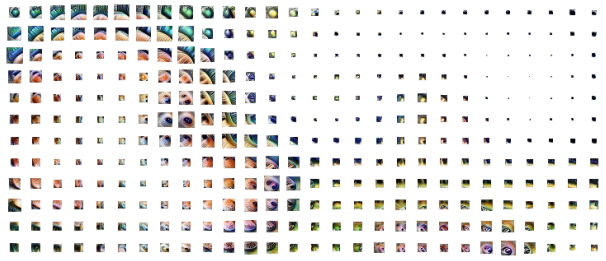
MIXED3A

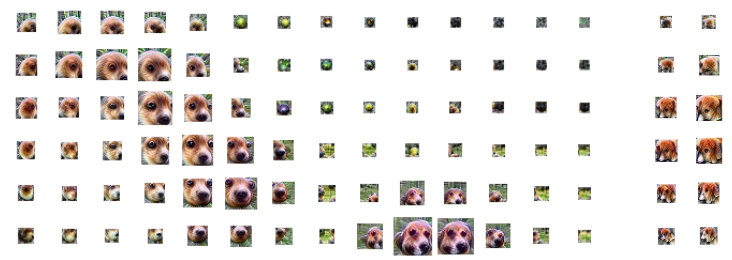


MIXED4A

MIXED4

These visualizations, however, omit a crucial piece of information: the magnitude of the activations. By scaling the area of each cell by the magnitude of the activation vector, we can indicate how strongly the network detected features at that position:





MIXED3A

MIXED4A

MIXED4

How Are Concepts Assembled?

Feature visualization helps us answer *what* the network detects, but it does not answer *how* the network assembles these individual pieces to arrive at later decisions, or *why* these decisions were made.

Attribution is a set of techniques that answers such questions by explaining the relationships between neurons. There are a wide variety of approaches to attribution [7, 11, 12, 13, 14, 15, 17] but, so far, there doesn't seem to be a clear right answer. In fact, there's reason to think that all our present answers aren't quite right [16]. We think there's a lot of important research to be done on attribution methods, but for the purposes of this article the exact approach taken to attribution doesn't matter. We use a fairly simple method, linearly approximating the relationship [2], but could easily substitute in essentially any other technique. Future improvements to attribution will, of course, correspondingly improve the interfaces built on top of them.

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

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 [24, 13] 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.

Loading

The above interface affords us a more flexible relationship with attribution. To start, we perform attribution from each spatial position of each hidden layer shown to all 1,000 output classes. In order to visualize this thousand-dimensional vector, we use dimensionality reduction to produce a multi-directional saliency map. Overlaying these saliency maps on our magnitude-sized activation grids provides an information scent [25] over attribution space. The activation grids allow us to anchor attribution to the visual vocabulary our semantic dictionaries first established. On hover, we update the legend to depict attribution to the output classes (i.e., which classes does this spatial position most contribute to?).

Perhaps most interestingly, this interface allows us to interactively perform attribution *between hidden layers*. On hover, additional saliency maps mask the hidden layers, in a sense shining a light into their black boxes. This type of layer-to-layer attribution is a prime example of how carefully considering interface design drives the generalization of our existing abstractions for interpretability.

With this diagram, we have begun to think of attribution in terms of higher-level concepts. However, at a particular position, many concepts are being detected together and this interface makes it difficult to split them apart. By continuing to focus on spatial positions, these concepts remain entangled.

Channel Attribution

Saliency maps implicitly slice our cube of activations by applying attribution to the spatial positions of a hidden layer. This aggregates over all channels and, as a result, we cannot tell which specific detectors *at each position* most contributed to the final output classification.

An alternate way to slice the cube is by channels instead of spatial locations. Doing so allows us to perform *channel attribution*: how much did each detector contribute to the final output? (This approach is similar to contemporaneous work by Kim et al. [26], who do attribution to learned combination of channels.)

Loading

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.

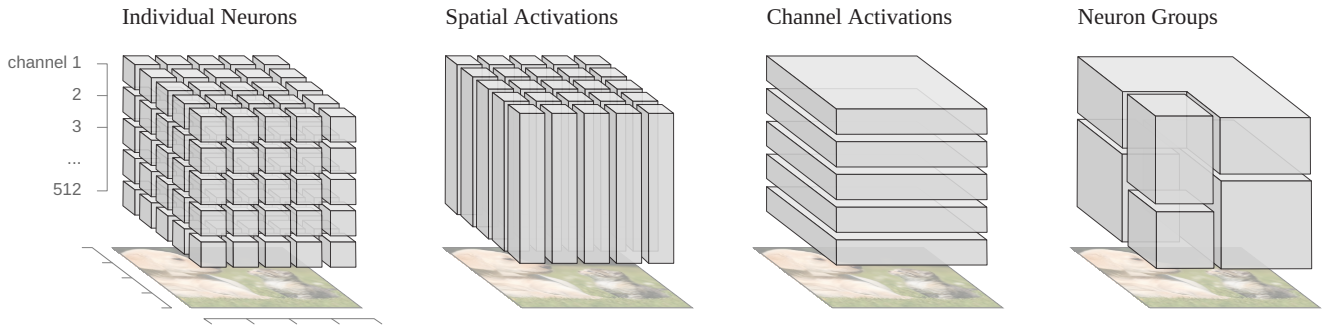
Making Things Human-Scale

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 one only uses spatial activations or channels, they miss out on very important parts of the story. For example it's interesting that the floppy ear detector helped us classify an image as a Labrador retriever, but it's much more interesting when that's combined with the locations that fired to do so. One can try to drill down to the level of neurons to tell the whole story, but the tens of thousands of neurons are simply too much information. Even the hundreds of channels, before being split into individual neurons, can be overwhelming to show users!

If we want to make useful interfaces into neural networks, it isn't enough to make things meaningful. We need to make them human scale, rather than overwhelming dumps of information. The key to doing so is finding more meaningful ways of breaking up our activations. There is good reason to believe that such decompositions exist. Often, many channels or spatial positions will work together in a highly correlated way and are most useful to think of as one unit. Other channels or positions will have very little activity, and can be ignored for a high-level overview. So, it seems like we ought to be able to find better

decompositions if we had the right tools.

There is an entire field of research, called matrix factorization, that studies optimal strategies for breaking up matrices. By flattening our cube into a matrix of spatial locations and channels, we can apply these techniques to get more meaningful groups of neurons. These groups will not align as naturally with the cube as the groupings we previously looked at. Instead, they will be combinations of spatial locations and channels. Moreover, these groups are constructed to explain the behavior of a network on a particular image. It would not be effective to reuse the same groupings on another image; each image requires calculating a unique set of groups.



In addition to naturally slicing a hidden layer's cube of activations into neurons, spatial locations, or channels, we can also consider more arbitrary groupings of locations and channels.

The groups that come out of this factorization will be the atoms of the interface a user works with. Unfortunately, any grouping is inherently a tradeoff between reducing things to human scale and, because any aggregation is lossy, preserving information. Matrix factorization lets us pick what our groupings are optimized for, giving us a better tradeoff than the natural groupings we saw earlier.

The goals of our user interface should influence what we optimize our matrix factorization to prioritize. For example, if we want to prioritize what the network detected, we would want the factorization to fully describe the activations. If we instead wanted to prioritize what would change the network's behavior, we would want the factorization to fully describe the gradient. Finally, if we want to prioritize what caused the present behavior, we would want the factorization to fully describe the attributions. Of course, we can strike a balance between these three objectives rather than optimizing one to the exclusion of the others.

In the following diagram, we've constructed groups that prioritize the activations, by factorizing the activations [3] with non-negative matrix factorization [4] . Notice how the overwhelmingly large number of neurons has been reduced to a small set of groups, concisely summarizing the story of the neural network.

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 [5] .

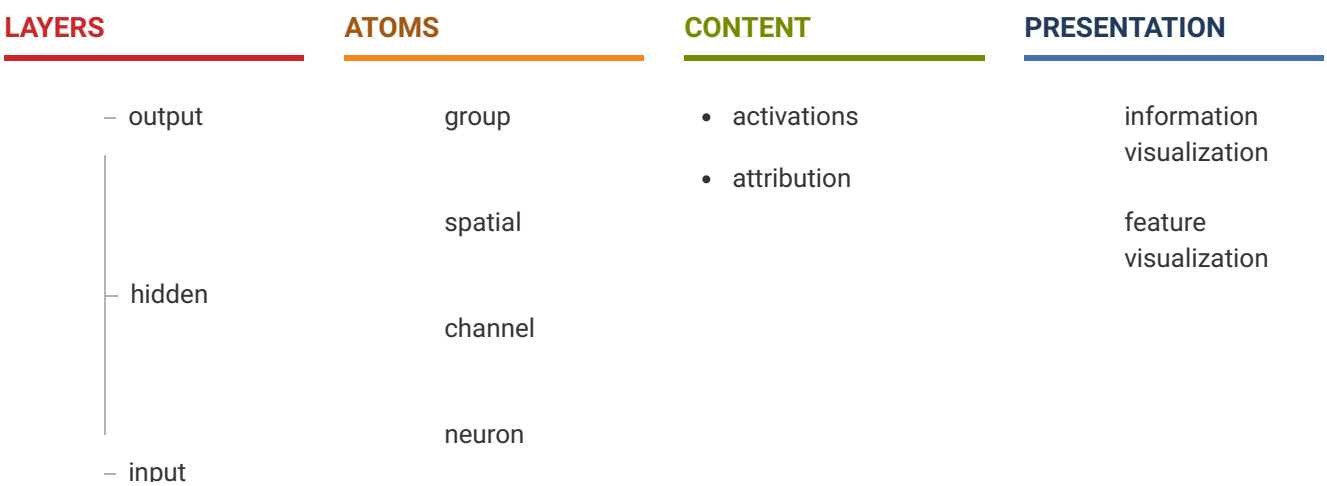
In this section, we recognize that the way in which we break apart the cube of activations is an important interface decision. Rather than resigning ourselves to the natural slices of the cube of activations, we construct more optimal groupings of neurons. These improved groupings are both more meaningful and more human-scale, making it less tedious for users to understand the behavior of the network.

Our visualizations have only begun to explore the potential of alternate bases in providing better atoms for understanding neural networks. For example, while we focus on creating smaller numbers of directions to explain individual examples, there's recently been exciting work finding "globally" meaningful directions [27, 26, 28] — such bases could be especially helpful when trying to understand multiple examples at a time, or in comparing models. [6]

The Space of Interpretability Interfaces

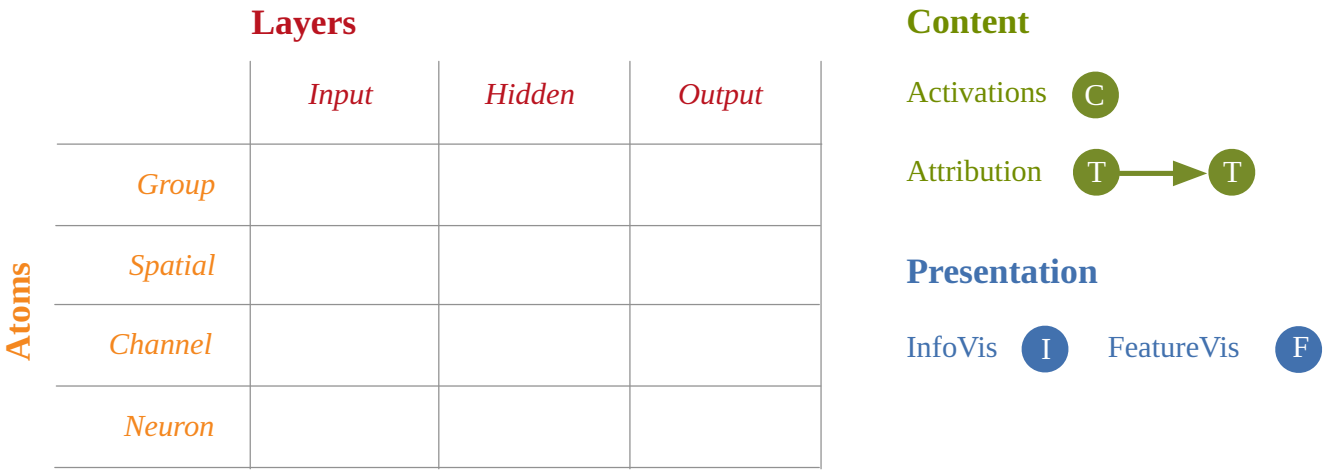
The interface ideas presented in this article combine building blocks such as feature visualization and attribution. Composing these pieces is not an arbitrary process, but rather follows a structure based on the goals of the interface. For example, should the interface emphasize *what* the network recognizes, prioritize *how* its understanding develops, or focus on making things *human-scale*. To evaluate such goals, and understand the tradeoffs, we need to be able to *systematically* consider possible alternatives.

We can think of an interface as a union of individual elements.

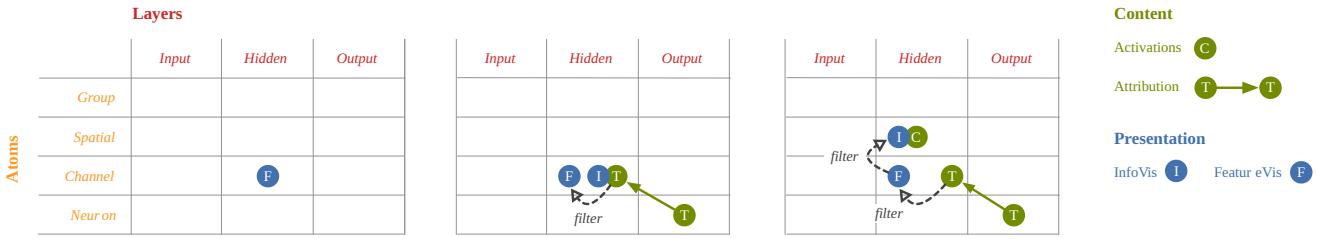


Each element displays a specific type of *content* (e.g., activations or attribution) using a particular style of *presentation* (e.g., feature visualization or traditional information visualization). This content lives on substrates defined by how given *layers* of the network are broken apart into *atoms*, and may be *transformed* by a series of operations (e.g., to filter it or project it onto another substrate). For example, our semantic dictionaries use feature visualization to display the activations of a hidden layer's neurons.

One way to represent this way of thinking is with a formal grammar [7], but we find it helpful to think about the space visually. We can represent the network's substrate (which layers we display, and how we break them apart) as a grid, with the content and style of presentation plotted on this grid as points and connections.



This setup gives us a framework to begin exploring the space of interpretability interfaces step by step. For instance, let us consider our teaser figure again. Its goal is to help us compare two potential classifications for an input image.



1. Feature visualization

2. Filter by output attribution

3. Drill down on hover

To understand a classification, we focus on the channels of the `mixed4d` layer. Feature visualization makes these channels meaningful.

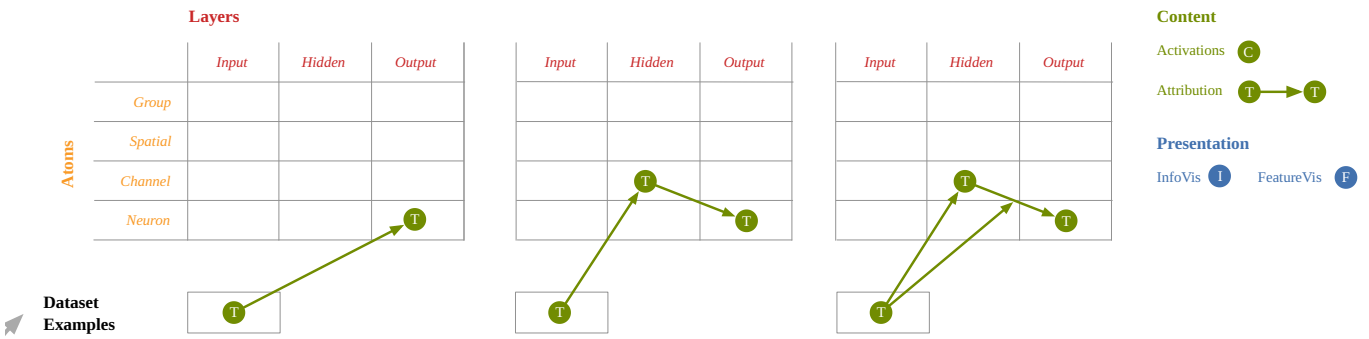
Next, we filter for specific classes by calculating the output attribution.

Hovering over channels, we get a heatmap of spatial activations.

In this article, we have only scratched the surface of possibilities. There are lots of combinations of our building blocks left to explore, and the design space gives us a way to do so systematically.

Moreover, each building block represents a broad class of techniques. Our interfaces take only one approach but, as we saw in each section, there are a number of alternatives for feature visualization, attribution, and matrix factorization. An immediate next step would be to try using these alternate techniques, and research ways to improve them.

Finally, this is not the complete set of building blocks; as new ones are discovered, they expand the space. For example, Koh & Liang. suggest ways of understanding the influence of dataset examples on model behavior [32]. We can think of dataset examples as another substrate in our design space, thus becoming another building block that fully composes with the others. In doing so, we can now imagine interfaces that not only allow us to inspect the influence of dataset examples on the final output classification (as Koh & Liang proposed), but also how examples influence the features of hidden layers, and how they influence the relationship between these features and the output. For example, if we consider our "Labrador retriever" image, we can not only see which dataset examples most influenced the model to arrive at this classification, but also which dataset examples most caused the "floppy ear" detectors to fire, and which dataset examples most caused these detectors to increase the "Labrador retriever" classification.



A new substrate.

An interface to understand how dataset examples influence the output classification, as presented by Koh & Liang [32]

An interface showing how examples influence the channels of hidden layers.

An interface for identifying which dataset examples most caused particular detectors to increase the output classification.

Beyond interfaces for analyzing model behavior, if we add model *parameters* as a substrate, the design space now allows us to consider interfaces for *taking action* on neural networks. [8] While most models today are trained to optimize simple objective functions that one can easily describe, many of the things

we'd like models to do in the real world are subtle, nuanced, and hard to describe mathematically. [9] One very promising approach to training models for these subtle objectives is learning from human feedback [33]. However, even with human feedback, it may still be hard to train models to behave the way we want if the problematic aspect of the model doesn't surface strongly in the training regime where humans are giving feedback. [10] Human feedback on the model's decision making process, facilitated by interpretability interfaces, could be a powerful solution to these problems. It might allow us to train models not just to make the *right decisions*, but to make them *for the right reasons*. (There is however a danger here: we are optimizing our model to look the way we want in our interface — if we aren't careful, this may lead to the model fooling us! [11])

Another exciting possibility is interfaces for comparing multiple models. For instance, we might want to see how a model evolves during training, or how it changes when you transfer it to a new task. Or, we might want to understand how a whole family of models compares to each other. Existing work has primarily focused on comparing the output behavior of models [34, 35, 36] but more recent work is starting to explore comparing their internal representations as well [37]. One of the unique challenges of this work is that we may want to align the atoms of each model; if we have completely different models, can we find the most analogous neurons between them? Zooming out, can we develop interfaces that allow us to evaluate large spaces of models at once [2]?

How Trustworthy Are These Interfaces?

In order for interpretability interfaces to be effective, we must trust the story they are telling us. We perceive two concerns with the set of building blocks we currently use. First, do neurons have a relatively consistent meaning across different input images, and is that meaning accurately reified by feature visualization? Semantic dictionaries, and the interfaces that build on top of them, are premised off this question being true. Second, does attribution make sense and do we trust any of the attribution methods we presently have?

Much prior research has found that directions in neural networks are semantically meaningful [38]. One particularly striking example of this is "semantic arithmetic" (eg. "king" - "man" + "woman" = "queen") [39, 40]. We explored this question, in depth, for GoogLeNet in our previous article [6] and found that many of its neurons seem to correspond to meaningful ideas. [12] Besides these neurons, however, we also found many neurons that do not have as clean a meaning including "poly-semantic" neurons that respond to a mixture of salient ideas (e.g., "cat" and "car"). There are natural ways that interfaces could respond to this: we could use diversity visualizations to reveal the variety of meanings the neuron can take, or rotate our semantic dictionaries so their components are more disentangled. Of course, just like our models can be fooled, the features that make them up can be too — including with adversarial examples [41]. In our view, features do not need to be flawless detectors for it to be useful for us to think about them as such. In fact, it can be interesting to identify when a detector misfires.

With regards to attribution, recent work suggests that many of our current techniques are unreliable [16]. One might even wonder if the idea is fundamentally flawed, since a function's output could be the result of non-linear interactions between its inputs. One way these interactions can pan out is as attribution being "path-dependent" [17]. A natural response to this would be for interfaces to explicitly surface this

information: how path-dependent is the attribution? A deeper concern, however, would be whether this path-dependency dominates the attribution. Clearly, this is not a concern for attribution between adjacent layers because of the simple (essentially linear) mapping between them. While there may be technicalities about correlated inputs, we believe that attribution is on firm grounding here. And even with layers further apart, our experience has been that attribution between high-level features at the output is much more consistent than attribution to the input — we believe that path-dependence is not a dominating concern here.

Model behavior is extremely complex, and our current building blocks force us to show only specific aspects of it. An important direction for future interpretability research will be developing techniques that achieve broader coverage of model behavior. But, even with such improvements, we anticipate that a key marker of trustworthiness will be interfaces that do not mislead. Interacting with the explicit information displayed should not cause users to implicitly draw incorrect assessments about the model (we see a similar principle articulated by Mackinlay for data visualization [30]). Undoubtedly, the interfaces we present in this article have room to improve in this regard. Fundamental research, at the intersection of machine learning and human-computer interaction, is necessary to resolve these issues.

Trusting our interfaces is essential for many of the ways we want to use interpretability. This is both because the stakes can be high (as in safety and fairness) and also because ideas like training models with interpretability feedback put our interpretability techniques in the middle of an adversarial setting.

Conclusion & Future Work

There is a rich design space for interacting with enumerative algorithms, and we believe an equally rich space exists for interacting with neural networks. We have a lot of work left ahead of us to build powerful and trustworthy interfaces for interpretability. But, if we succeed, interpretability promises to be a powerful tool in enabling meaningful human oversight and in building fair, safe, and aligned AI systems.

Acknowledgments

Our article was greatly strengthened thanks to the detailed feedback by Ben Poole, Emma Pierson, Jason Yosinski, Jeff Heer, John Backus, Martin Wattenberg, Matt Johnson, and Tim Hwang.

We also really appreciated the conversations we've had with Tom Brown, Catherine Olsson, Daniel Dewey, Ajeya Cotra, Dario Amodei, Paul Christiano on the relationship of interpretability to safety; the thoughtful comments of Michael Nielsen, Zak Stone, Zan Armstrong and Anjuli Kannan; the support of Wolff Dobson, Jack Clark, Charina Chou, Jason Freidenfelds, Christian Howard, Karoly Zsolnai-Feher in communicating our work to a broader audience; and the supportive environment fostered by Greg Corrado and Jeff Dean at Google Brain.

Finally, we really appreciate Justin Gilmer stepping in as acting Distill editor of this article, and Qiqi Yan, Guillaume Alain, and

anonymous reviewer C for taking the time to review our article.

Author Contributions

Interface Design & Prototyping. This work began with a number of exciting interface demonstrations by Alex in 2015, combining feature visualizations, activations, and dimensionality reduction. In early 2017, Chris generalized this line of research and combined it with attribution. Katherine prototyped early interfaces for spatial attribution. Ian built the neuron group Sankey diagram interface. Arvind created most of the final set of interfaces in the diagram, significantly improving them, with extensive design input and polish from Shan.

Conceptual Contributions. Many of these ideas have their earliest provenance with Alex. Chris generalized and refined them, and integrated attribution. Chris, Arvind, and Ian developed the building blocks framing. Ian and Chris coined the term "semantic dictionaries." Arvind and Chris crystallized this thinking into a grammar, and contextualized it with respect to both the machine learning and human-computer interaction communities.

Writing. Arvind and Chris wrote the text of the article, with significant input from Ian and Shan.

Infrastructure. The core library we use to visualize neural networks, Lucid, was primarily written by Chris, Alex, and Ludwig. Chris wrote most of the code used for attribution and matrix factorization. Ludwig created distributed implementations and workflows for generating our diagrams.

Discussion and Review

Review 1 - Qiqi Yan
Review 2 - Guillaume Alain
Review 3 - Anonymous

Footnotes

1. We're actively investigating why this is, and hope to uncover principles for designing interpretable models. In the meantime, while we demonstrate our techniques on GoogLeNet, we provide code for you to try them on other models. [↩]

2. We do attribution by linear approximation in all of our interfaces. That is, we estimate the effect of a neuron on the output is its activation times the rate at which increasing its activation increases the output. When we talk about a linear combination of activations, the attribution can be thought of as the linear combination of the attributions of the units, or equivalently as the dot product between the activation of that combination and the gradient.

   For spatial attribution, we do an additional trick. GoogLeNet's strided max pooling introduces a lot of noise and checkerboard patterns to it's gradients [23] . To avoid our interface demonstrations being dominated by this noise, we (a) do a relaxation of the gradient of max pooling, distributing gradient to inputs proportional to their activation instead of winner takes all and (b) cancel out the checkerboard patterns.

   The notebooks attached to diagrams provide reference implementations. [↩]

3. Most matrix factorization algorithms and libraries are set up to minimize the mean squared error of the reconstruction of a matrix you give them. There are ways to hack such libraries to achieve more general objectives through clever manipulations of the provided matrix, as we will see below. More broadly, matrix factorization is an optimization problem, and with custom tools you can achieve all sorts of custom factorizations. [↩]

4. As the name suggests, non-negative matrix factorization (NMF) constrains its factors to be positive. This is fine for the activations of a ReLU network, which must be positive as well. Our experience is that the groups we get from NMF seem more independent and semantically meaningful than those without this constraint. Because of this constraint, groups from NMF are a less efficient

and semantically meaningful than those without this constraint. Because of this constraints, groups from NMF are a less efficient at representing the activations than they would be without, but our experience is that they seem more independent and semantically meaningful. [↩]

5. We formalize this "compatibility" in a manner described below, although we're not confident it's the best formalization and won't be surprised if it is superseded in future work.

   Consider the attribution from every neuron in the layer to the set of *N* groups we want it to be compatible with. The basic idea is to split each entry in the activation matrix into *N* entries on the channel dimension, spreading the values proportional to the absolute value of its attribution to the corresponding group. Any factorization of this matrix induces a factorization of the original matrix by collapsing the duplicated entries in the column factors. However, the resulting factorization tries to create separate factors when the activation of the same channel has different attributions in different places. [↩]

6. The recent NIPS disentangling workshop provides other promising directions. We're excited to see a venue for this developing area of research. [↩]

7. Expressing design spaces as formal grammars is a technique used in human-computer interaction [29] and data visualization [30, 31] . We think this is a powerful technique and present this grammar as an initial exploration of how it might apply to interpretability interfaces.

```
Id = Int

Atom  = Neuron | Spatial | Channel | Group | Whole
Layer = Input | Hidden Int | Out

Substrate = Network Id Atom Layer
          | Dataset Id
          | Parameters Id

Content = Substrate
        | Activation Substrate
        | Attribution Substrate Substrate
        | Transform Content Content?

Element = InfoVis Content | FeatureVis Content

Interface = [ Element ] [↩]
```

8. Note that essentially all our interpretability techniques are differentiable, so you can backprop through them. [↩]

9. An extreme example of the subtle objective problem is something like "creating interesting art", but much more mundane examples arise more or less whenever humans are involved. [↩]

10. There are lots of reasons why problematic behavior may not surface or may be hard for an evaluator to give feedback on. For example, discrimination and bias may be subtly present throughout the model's behavior, such that it's hard for a human evaluator to critique. Or the model may be making a decision in a way that has problematic consequences, but those consequences never play out in the problems we're training it on. [↩]

11. Related ideas have occasionally been discussed under the term "cognitive steganography." [↩]

12. We validated this in a number of ways: we visualized them without a generative model prior, so that the content of the visualizations was causally linked to the neuron firing; we inspected the spectrum of examples that cause the neuron to fire; and used diversity visualizations to try to create different inputs that cause the neuron to fire.

For more details, see the article's appendix and the guided tour in @ch402's Twitter thread. We're actively investigating why GoogLeNet's neurons seem more meaningful. [↩]

References

1. **Thought as a Technology** [HTML]
   Nielsen, M., 2016.

2. **Visualizing Representations: Deep Learning and Human Beings** [link]
   Olah, C., 2015.

3. **Understanding neural networks through deep visualization** [PDF]
   Yosinski, J., Clune, J., Nguyen, A., Fuchs, T. and Lipson, H., 2015. arXiv preprint arXiv:1506.06579.

4. **Using Artificial Intelligence to Augment Human Intelligence** [link]
   Carter, S. and Nielsen, M., 2017. Distill. DOI: 10.23915/distill.00009

5. **Visualizing higher-layer features of a deep network** [PDF]
   Erhan, D., Bengio, Y., Courville, A. and Vincent, P., 2009. University of Montreal, Vol 1341, pp. 3.

6. **Feature Visualization** [link]
   Olah, C., Mordvintsev, A. and Schubert, L., 2017. Distill. DOI: 10.23915/distill.00007

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

8. **Deep neural networks are easily fooled: High confidence predictions for unrecognizable images** [PDF]
   Nguyen, A., Yosinski, J. and Clune, J., 2015. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 427--436. DOI: 10.1109/cvpr.2015.7298640

9. **Inceptionism: Going deeper into neural networks** [HTML]
   Mordvintsev, A., Olah, C. and Tyka, M., 2015. Google Research Blog.

10. **Plug & play generative networks: Conditional iterative generation of images in latent space** [PDF]
    Nguyen, A., Clune, J., Bengio, Y., Dosovitskiy, A. and Yosinski, J., 2016. arXiv preprint arXiv:1612.00005.

11. **Visualizing and understanding convolutional networks** [PDF]
    Zeiler, M.D. and Fergus, R., 2014. European conference on computer vision, pp. 818--833.

12. **Striving for simplicity: The all convolutional net** [PDF]
    Springenberg, J.T., Dosovitskiy, A., Brox, T. and Riedmiller, M., 2014. arXiv preprint arXiv:1412.6806.

13. **Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization** [PDF]
    Selvaraju, R.R., Das, A., Vedantam, R., Cogswell, M., Parikh, D. and Batra, D., 2016. arXiv preprint arXiv:1610.02391.

14. **Interpretable Explanations of Black Boxes by Meaningful Perturbation** [PDF]
    Fong, R. and Vedaldi, A., 2017. arXiv preprint arXiv:1704.03296.

15. **PatternNet and PatternLRP--Improving the interpretability of neural networks** [PDF]
    Kindermans, P., Schutt, K.T., Alber, M., Muller, K. and Dahne, S., 2017. arXiv preprint arXiv:1705.05598. DOI: 10.1007/978-3-319-10590-1_53

16. **The (Un)reliability of saliency methods** [PDF]
    Kindermans, P., Hooker, S., Adebayo, J., Alber, M., Schutt, K.T., Dahne, S., Erhan, D. and Kim, B., 2017. arXiv preprint arXiv:1711.00867.

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

18. **Visualizing data using t-SNE** [PDF]
Maaten, L.v.d. and Hinton, G., 2008. Journal of Machine Learning Research, Vol 9(Nov), pp. 2579--2605.

19. **LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks** [PDF]
Strobelt, H., Gehrmann, S., Pfister, H. and Rush, A.M., 2018. IEEE Transactions on Visualization and Computer Graphics, Vol 24(1), pp. 667--676. IEEE. DOI: 10.1109/tvcg.2017.2744158

20. **ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models** [PDF]
Kahng, M., Andrews, P.Y., Kalro, A. and Chau, D.H.P., 2018. IEEE Transactions on Visualization and Computer Graphics, Vol 24(1), pp. 88--97. IEEE. DOI: 10.1109/tvcg.2017.2744718

21. **Do convolutional neural networks learn class hierarchy?** [PDF]
Bilal, A., Jourabloo, A., Ye, M., Liu, X. and Ren, L., 2018. IEEE Transactions on Visualization and Computer Graphics, Vol 24(1), pp. 152--162. IEEE. DOI: 10.1109/tvcg.2017.2744683

22. **Going deeper with convolutions** [PDF]
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. and others,, 2015. DOI: 10.1109/cvpr.2015.7298594

23. **Deconvolution and Checkerboard Artifacts** [link]
Odena, A., Dumoulin, V. and Olah, C., 2016. Distill. DOI: 10.23915/distill.00003

24. **Learning deep features for discriminative localization** [PDF]
Zhou, B., Khosla, A., Lapedriza, A., Oliva, A. and Torralba, A., 2016. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2921--2929. DOI: 10.1109/cvpr.2016.319

25. **Information foraging** [link]
Pirolli, P. and Card, S., 1999. Psychological review, Vol 106(4), pp. 643. American Psychological Association. DOI: 10.1037//0033-295x.106.4.643

26. **TCAV: Relative concept importance testing with Linear Concept Activation Vectors** [PDF]
Kim, B., Gilmer, J., Viegas, F., Erlingsson, U. and Wattenberg, M., 2017. arXiv preprint arXiv:1711.11279.

27. **SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability** [PDF]
Raghu, M., Gilmer, J., Yosinski, J. and Sohl-Dickstein, J., 2017. Advances in Neural Information Processing Systems 30, pp. 6078--6087. Curran Associates, Inc.

28. **Net2Vec: Quantifying and Explaining how Concepts are Encoded by Filters in Deep Neural Networks**
Fong, R. and Vedaldi, A., 2018. arXiv preprint arXiv:1801.03454.

29. **Understanding Black-box Predictions via Influence Functions** [PDF]
Koh, P.W. and Liang, P., 2017. International Conference on Machine Learning (ICML).

30. **Deep reinforcement learning from human preferences**
Christiano, P.F., Leike, J., Brown, T., Martic, M., Legg, S. and Amodei, D., 2017. Advances in Neural Information Processing Systems, pp. 4302--4310.

31. **Interactive optimization for steering machine classification** [PDF]
Kapoor, A., Lee, B., Tan, D. and Horvitz, E., 2010. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1343--1352. DOI: 10.1145/1753326.1753529

32. **Interacting with predictions: Visual inspection of black-box machine learning models** [PDF]
Krause, J., Perer, A. and Ng, K., 2016. Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, pp. 5686--5697. DOI: 10.1145/2858036.2858529

33. **Modeltracker: Redesigning performance analysis tools for machine learning** [PDF]
Amershi, S., Chickering, M., Drucker, S.M., Lee, B., Simard, P. and Suh, J., 2015. Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, pp. 337--346. DOI: 10.1145/2702123.2702509

34. **Network dissection: Quantifying interpretability of deep visual representations** [PDF]
Bau, D., Zhou, B., Khosla, A., Oliva, A. and Torralba, A., 2017. Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, pp. 3319--3327. DOI: 10.1109/cvpr.2017.354

35. **Intriguing properties of neural networks** [PDF]
Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R., 2013. arXiv preprint arXiv:1312.6199.

36. **Efficient estimation of word representations in vector space** [PDF]
Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. arXiv preprint arXiv:1301.3781.

37. **Unsupervised representation learning with deep convolutional generative adversarial networks** [PDF]
Radford, A., Metz, L. and Chintala, S., 2015. arXiv preprint arXiv:1511.06434.

38. **Adversarial manipulation of deep representations** [PDF]
Sabour, S., Cao, Y., Faghri, F. and Fleet, D.J., 2015. arXiv preprint arXiv:1511.05122.

39. **Automating the design of graphical presentations of relational information** [PDF]
Mackinlay, J., 1986. Acm Transactions On Graphics (Tog), Vol 5(2), pp. 110--141. ACM. DOI: 10.1145/22949.22950

Updates and Corrections

If you see mistakes or want to suggest changes, please create an issue on GitHub.

Reuse

Citation

For attribution in academic contexts, please cite this work as

```
Olah, et al., "The Building Blocks of Interpretability", Distill, 2018.
```

BibTeX citation

```
@article{olah2018the,
  author = {Olah, Chris and Satyanarayan, Arvind and Johnson, Ian and Carter, Shan and
Schubert, Ludwig and Ye, Katherine and Mordvintsev, Alexander},
  title = {The Building Blocks of Interpretability},
  journal = {Distill},
  year = {2018},
  note = {https://distill.pub/2018/building-blocks},
  doi = {10.23915/distill.00010}
}
```