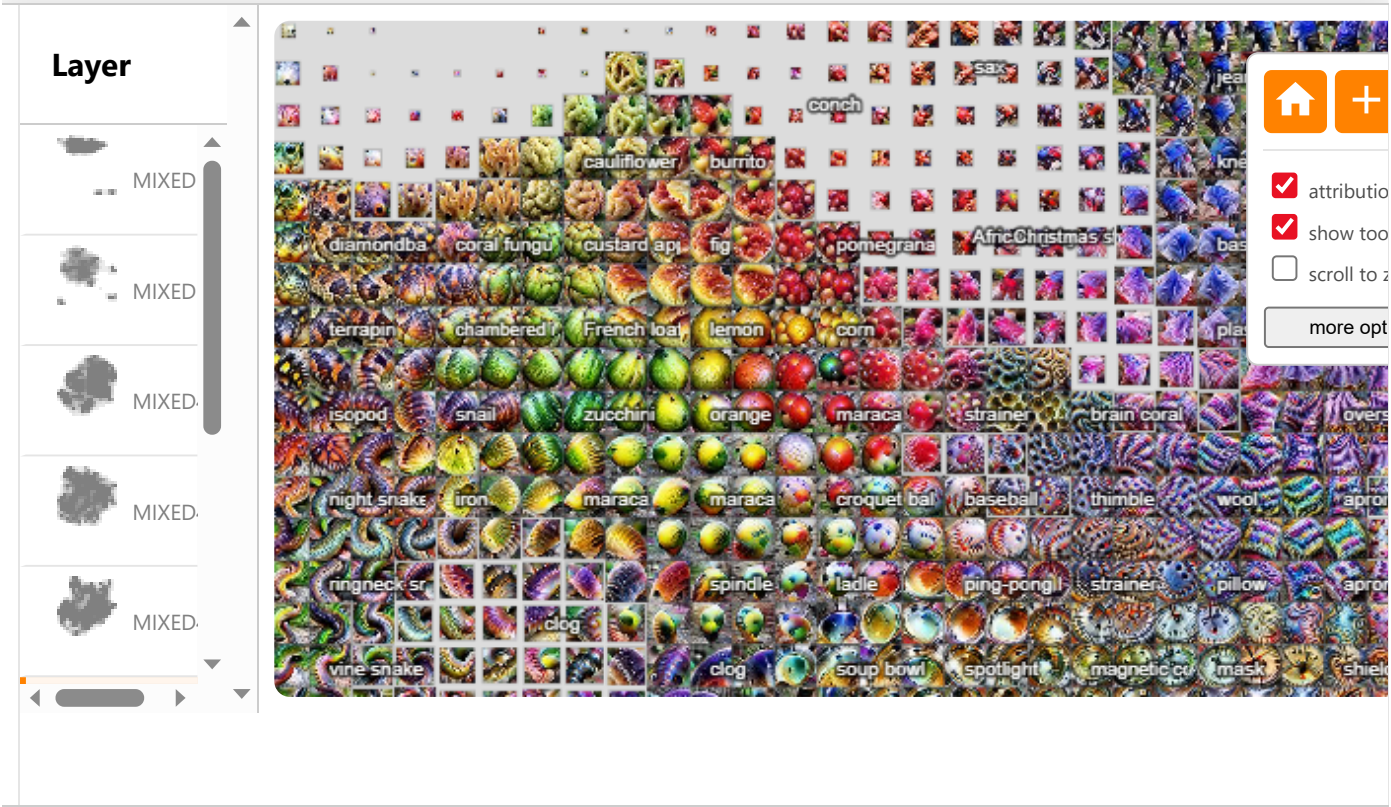
Exploring Neural Networks with Activation Atlases

By using feature inversion to visualize millions of activations from an image classification network, we create an explorable *activation atlas* of features the network has learned which can reveal how the network typically represents some concepts.



Above, an **activation atlas** of the InceptionV1 vision classification network reveals many fully realized features, such as electronics, buildings, food, animal ears, plants, and watery backgrounds. Grid cells are labeled with the classification they most support. [1] Grid cells are also sized according to the density of the number of activations that are averaged within. View full-screen version

TRY IN A NOTEBOOK

AUTHORS

Shan Carter
Zan Armstrong
Ludwig Schubert
Ian Johnson

AFFILIATIONS

Google Brain Team
Google Accelerated Science
OpenAI
Google Cloud

Chris Olah

OpenAI

PUBLISHED

March 6, 2019

DOI

10.23915/distill.00015

Introduction

Neural networks can learn to classify images more accurately than any system humans directly design. This raises a natural question: What have these networks learned that allows them to classify images so well?

Feature visualization is a thread of research that tries to answer this question by letting us "see through the eyes" of the network [1, 2, 3, 4, 5, 6]. It began with research into visualizing individual neurons and trying to determine what they respond to. Because neurons don't work in isolation, this led to applying feature visualization to simple combinations of neurons. But there was still a problem — what combinations of neurons should we be studying? A natural answer (foreshadowed by work on model inversion [7]) is to visualize activations [8], the combination of neurons firing in response to a particular input.

These approaches are exciting because they can make the hidden layers of networks comprehensible. These layers are the heart of how neural networks outperform more traditional approaches to machine learning and historically, we've had little understanding of what happens in them [2]. Feature visualization addresses this by connecting hidden layers back to the input, making them meaningful.

Unfortunately, visualizing activations has a major weakness — it is limited to seeing only how the network sees a single input. Because of this, it doesn't give us a big picture view of the network. When what we want is a map of an entire forest, inspecting one tree at a time will not suffice.

There are techniques which give a more global view, but they tend to have other downsides. For example, Karpathy's CNN codes visualization [9] gives a global view of a dataset by taking each image and organizing them by their activation values from a neural network. Showing which images the model sees as similar does help us infer some ideas about what features the network is responding to, but feature visualization makes those connections much more explicit. Nguyen, et al [10] use t-SNE to make more diverse neuron visualizations, generating diverse starting points for the optimization process by clustering images in the t-SNE map. This reveals a broader picture of what the neuron detects but is still focused on individual neurons.

In this article we introduce *activation atlases* to this quiver of techniques. (An example is shown at the top of this article.) Broadly speaking, we use a technique similar to the one in CNN codes, but instead of showing input data, we show feature visualizations of averaged activations. By combining these two techniques, we can get the advantages of each in one view — a global map seen through the eyes of the network.

In theory, showing the feature visualizations of the basis neurons would give us the global view of a network that we are seeking. In practice, however, neurons are rarely used by the network in isolation, and it may be difficult to understand them that way. As an analogy, while the 26 letters in the alphabet provide a basis for English, seeing how letters are commonly combined to make words gives far more insight into the concepts that can be expressed than the letters alone. Similarly, activation atlases give us a bigger picture view by showing common combinations of neurons.

INDIVIDUAL NEURONS

Loading ... Loading ...

Visualizing individual neurons make hidden layers somewhat meaningful, but misses interactions between neurons — it only shows us one-dimensional, orthogonal probes of the high-dimensional activation space.

PAIRWISE INTERACTIONS

Loading ... Loading ...

Pairwise interactions reveal interaction effects, but they only show two-dimensional slices of a space that has hundreds of dimensions, and many of the combinations are not realistic.

SPATIAL ACTIVATIONS

Loading ... Loading ...

Spatial activations show us important combinations of many neurons by sampling the sub-manifold of likely activations, but they are limited to those that occur in the given example image.

ACTIVATION ATLAS

Activation atlases give us a bigger picture overview by sampling more of the manifold of likely activations.

Loading ... Loading ...

These atlases not only reveal visual abstractions within a model, but later in the article we will show that they can reveal high-level misunderstandings in a model that can be exploited. For example, by looking at an activation atlas we will be able to see why a picture of a baseball can switch the classification of an image from “grey whale” to “great white shark”.

Loading ...

1.	**grey whale**	**91.0%**
2.	killer whale	7.5%
3.	great white shark	0.7%
4.	gar	0.4%

Loading ...

1.	**great white shark**	**66.7%**
2.	baseball	7.4%
3.	grey whale	4.1%
4.	sombrero	3.2%

Of course, activation atlases do have limitations. In particular, they're dependent on the distribution of the data we choose to sample activations from (in our examples, we use one million images chosen at random from the ImageNet dataset [11] training data). As a result, they will only show the activations that exist within the distribution of the sample data. However, while it's important to be aware of these limitations — we'll discuss them in much more depth later! — Activation Atlases still give us a new kind of overview of what neural networks can represent.

Looking at a Single Image

Before we dive into Activation Atlases, let's briefly review how we use feature visualization to make activation vectors meaningful ("see through the network's eyes"). This technique was introduced in Building Blocks [8], and will be the foundation of Activation Atlases.

Throughout this article, we'll be focussing on a particular neural network: InceptionV1 [12] (also known as "GoogLeNet"). When it came out, it was notable for winning the classification task in the 2014 ImageNet Large Scale Visual Recognition Challenge [11, 13].

InceptionV1 consists of a number of layers, which we refer to as "mixed3a", "mixed3b", "mixed4a", etc., and sometimes shortened to just "3a". Each layer successively builds on the previous layers.

Loading ...

InceptionV1 builds up its understanding of images over several layers (see overview from [2]). It was trained on ImageNet ILSVRC [11]. Each layer actually has several component parts, but for this article we'll focus on these larger groups.

To visualize how InceptionV1 sees an image, the first step is to feed the image into the network and run it through to the layer of interest. Then we collect the activations — the numerical values of how much each neuron fired. If a neuron is excited by what it is shown, its activation value will be positive.

Unfortunately these vectors of activation values are just vectors of unitless numbers and not particularly interpretable by people. This is where feature visualization comes in. Roughly speaking, we can think of feature visualization as creating an idealized image of what the network thinks would produce a particular activation vector. Whereas we normally use a network to transform an image into an activation vector, in feature visualization we go in the opposite direction. Starting with an activation vector at a particular layer, we create an image through an iterative optimization process.

Because InceptionV1 is a convolutional network, there is not just one activation vector per layer per image. This means that the same neurons are run on each patch of the previous layer. Thus, when we pass an entire image through the network, each neuron will be evaluated hundreds of times, once for each overlapping patch of the image. We can consider the vectors of how much each neuron fired for each patch separately.

Loading ...

TRY IN A NOTEBOOK

The result is a grid of feature visualizations, one for each patch. This shows us how the network sees different parts of the input image.

Loading ...

Input image from ImageNet.

Loading ...

Activation grid from InceptionV1, layer mixed4d.

Aggregating Multiple Images

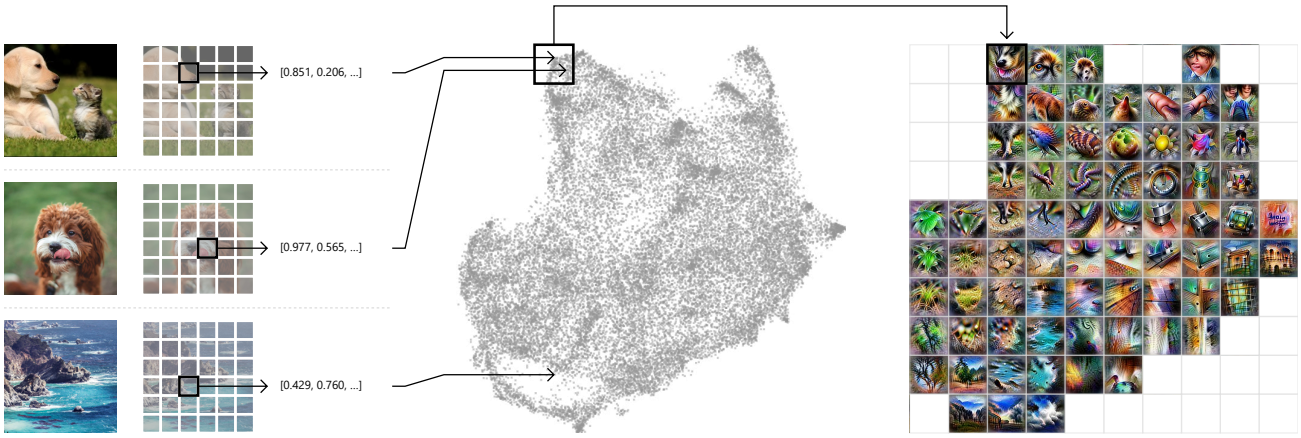
Activation grids show how the network sees a single image, but what if we want to see more? What if we

want to understand how it reacts to millions of images?

Of course, we could look at individual activation grids for those images one by one. But looking at millions of examples doesn't scale, and human brains aren't good at comparing lots of examples without structure. In the same way that we need a tool like a histogram in order to understand millions of numbers, we need a way to aggregate and organize activations if we want to see meaningful patterns in millions of them.

Let's start by collecting activations from one million images. We'll randomly select one spatial activation per image. [3] This gives us one million activation vectors. Each of the vectors is high-dimensional, perhaps 512 dimensions! With such a complex set of data, we need to organize and aggregate it if we want a big picture view.

Thankfully, we have modern dimensionality reduction techniques at our disposal. These algorithms, such as t-SNE [14] and UMAP [15], can project high-dimensional data like our collection of activation vectors into useful 2D layouts, preserving some of the local structure of the original space. This takes care of organizing our activation vectors, but we also need to aggregate into a more manageable number of elements — one million dots would be hard to interpret. We'll do this by drawing a grid over the 2D layout we created with dimensionality reduction. For each cell in our grid, we average all the activations that lie within the boundaries of that cell, and use feature visualization to create an iconic representation.



A randomized set of one million images is fed through the network, collecting one random spatial activation per image.

The activations are fed through UMAP to reduce them to two dimensions. They are then plotted, with similar activations placed near each other.

We then draw a grid and average the activations that fall within a cell and run feature inversion on the averaged activation. We also optionally size the grid cells according to the density of the number of activations that are averaged within.

TRY IN A NOTEBOOK

We perform feature visualization with the regularizations described in Feature Visualization [2] (in particular, transformation robustness). However, we use a slightly non-standard objective. Normally, to visualize a direction in activation space, v, one maximizes the dot product with the activation vector h at a position: $h_{x,y} \cdot v$. We find it helpful to use an objective that emphasizes angle more heavily by multiplying the dot product by cosine similarity, leading to objectives of the following form: $\frac{(h_{x,y} \cdot v)^{n+1}}{(||h_{x,y}|| \cdot ||v||)^n}$. We also find that whitening the activation space to unstretch it can help improve feature visualization. We don't yet fully understand this phenomenon. A reference implementation of this can be seen in the attached notebooks, and more general discussion can be found in this github issue.

For each activation vector, we also compute an *attribution* vector. The attribution vector has an entry for each class, and approximates the amount that the activation vector influenced the logit for each class. Attribution vectors generally depend on the surrounding context. We follow Building Blocks [8] in computing the attribution of the activation vector at a position, $h_{x,y}$, to a class logit, logit_c as $h_{x,y} \cdot \nabla_{h_{x,y}} \text{logit}_c$. That is, we estimate that the effect of a neuron on a logit is the rate at which increasing the neuron affects the logit. This is similar to Grad-CAM [16], but without the spatial averaging of the gradient. Instead, we reduce noise in the gradient by using a continuous relaxation of the gradient for max pooling in computing the gradient (as in [8]). (A detailed reference implementation can be found in this notebook.) The attribution shown for cells in the activation atlas is the average of attribution vectors for activations in that cell.

This average attribution can be thought of as showing what classes that cell tends to support, marginalizing over contexts. At early layers, the average attribution is very small and the top classes are fairly arbitrary because low-level visual features like textures tend to not be very discriminative without context.

So, how well does this work? Well, let's try applying it to InceptionV1 [12] at layer mixed4c.

Loading ...

This atlas can be a bit overwhelming at first glance — there's a lot going on! This diversity is a reflection of the variety of abstractions and concepts the model has developed. Let's take a tour to examine this atlas in more depth.

If we look at the top-left of the atlas, we see images which look like animal heads. There is some differentiation between different types of animals, but it seems to be more a collection of elements of generic mammals — eyes, fur, noses — rather than a collection of different classes of animals. We've also added labels that show which class each averaged activation most contributes to. Please note, in some areas of a layer this early in the network these attribution labels can be somewhat chaotic. In early layers the attribution vectors have a small magnitude since they don't have a consistent effect on the output.

Loading ...

As we move further down we start to see different types of fur and the backs of four-legged animals.

Loading ...

Below this, we find different animal legs and feet resting on different types of ground.

Loading ...

Below the feet we start to lose any identifiable parts of animals, and see isolated grounds and floors. We see attribution toward environments like “sandbar” and also toward things that are found on the ground, like “doormat” or “ant”.

Loading ...

These sandy, rocky backgrounds slowly blend into beaches and bodies of water. Here we see lakes and oceans, both above and below water. Though the network does have certain classes like “seashore”, we see attribution toward many sea animals, without any visual references to the animals themselves. While not unexpected, it is reassuring to see that the activations that are used to identify the sea for the class “seashore” are the same ones used when classifying “starfish” or “sea lion”. There is also no real distinction at this point between lakes and ocean — “lakeside” and “hippopotamus” attributions are intermingled with “starfish” and “stingray”.

Loading ...

Now let’s jump to the other side of the atlas, where we can see many variations of text detectors. These will be useful when identifying classes such as “menu”, “web site” or “book jacket”.

Loading ...

Moving upward, we see many variations of people. There are very few classes that specifically identify people in ImageNet, but people are present in lots of the images. We see attribution toward things people use (“hammer”, “flute”), clothes that people wear (“bow tie”, “maillot”) and activities that people participate in (“basketball”). There is a uniformity to the skin color in these visualizations which we suspect is a reflection of the distribution of the data used for training. (You can browse the ImageNet training data by category online: swimming trunks, diaper, band aid, lipstick, etc.)

Loading ...

And finally, moving back to the left, we can see round food and fruit organized mostly by colors — we see attribution toward “lemon”, “orange” and “fig”.

Loading ...

We can also trace curved paths through this manifold that we've created. Not only are regions important, but certain movements through the space seem to correspond to human interpretable qualities. With the fruit, we can trace a path that seems to correlate with the size and number of fruits in the frame.

Loading ...

Similarly, with people, we can trace a path that seems to correspond to how many people are in the frame, whether it's a single person or a crowd.

Loading ...

With the ground detectors, we can trace a path from water to beach to rocky cliffs.

Loading ...

In the plants region, we can trace a path that seems to correspond to how blurry the plant is. This could possibly be used to determine relative size of objects because of the typical focal lengths of cameras. Close up photos of small insects have more opportunity for blurry background foliage than photos of larger animals, like monkeys.

It is important to note that these paths are constructed after the fact in the low-dimensional projection. They are smooth paths in this reduced projection but we don't necessarily know how the paths operate in the original higher-dimensional activation space.

Looking at Multiple Layers

In the previous section we focused on one layer of the network, mixed4c, which is in the middle of our network. Convolutional networks are generally deep, consisting of many layers that progressively build up more powerful abstractions. In order to get a holistic view, we must look at how the model's abstractions develop over several layers.

In early layers (eg. mixed3b), the network seems to represents textures and simple patterns.

By mid layers (eg. mixed4c), icons evoke individual concepts like eyes, leaves, water, that are shared among many classes.

In the final layers (eg. mixed5b), abstractions become more closely aligned with the output classes.

To start, let's compare three layers from different areas of the network to try to get a sense for the different personalities of each — one very early layer (mixed3b), one layer from the middle (mixed4c), and the final layer (mixed5b) before the logits. We'll focus on areas of each layer that contribute to the classification of "cabbage".

MIXED3B

You'll immediately notice that the early layer is very nonspecific in comparison to the others. The icons that emerge are of patterns and splotches of color. It is suggestive of the final class, but not particularly evocative.

MIXED4C

By the middle layer, icons definitely resemble leaves, but they could be any type of plant. Attributions are focused on plants, but are a little all over the board.

MIXED5B

Here we see foliage with textures that are specific to cabbage, and curved into rounded balls. There are full heads of cabbage rather than individual leaves.

As you move through the network, the later layers seem to get much more specific and complex. This is to be expected, as each layer builds its activations on top of the preceding layer's activations. The later layers also tend to have larger receptive fields than the ones that precede them (meaning they are shown larger subsets of the image) so the concepts seem to encompass more of the whole of objects.

There is another phenomenon worth noting: not only are concepts being refined, but new concepts are appearing out of combinations of old ones. Below, you can see how sand and water are distinct concepts in a middle layer, mixed4c, both with strong attributions to the classification of "sandbar". Contrast this with a later layer, mixed5b, where the two ideas seem to be fused into one activation.

MIXED4C

MIXED5B

Loading ...

Loading ...

Loading ...

Loading ...

Loading ...

Loading ...

In mixed4c we see two different regions that have high attribution toward "sandbar": an area of **sandy textures**...

...and a separate area of **watery textures**.

In a later layer we see activations that contain **both** of those concepts when detecting "sandbar".

Finally, if we zoom out a little, we can see how the broader shape of the activation space changes from layer to layer. By looking at similar regions in several consecutive layers, we can see concepts getting refined and differentiated — In mixed4a we see very vague, generic blob, which gets refined into much more specific “peninsulas” by mixed4e.

MIXED4A

In mixed4a there is a vague “mammalian” area.

Loading ...

MIXED4B

By mixed4b, animals and people have been disentangled, with some fruit and food emerging in the middle.

Loading ...

MIXED4C

All the concepts are further refined and differentiated into small “peninsulas”.

Loading ...

MIXED4D

The specialization continues in mixed4d.

Loading ...

Below you can browse many more of the layers of InceptionV1. You can compare the curved edge detectors of mixed4a with the bowls and cups of mixed5b. Mixed4b has some interesting text and pattern detectors, whereas mixed5a appears to use those to differentiate menus from crossword puzzles from rulers. In early layers, like mixed4b, you'll see things that have similar textures near each other, like fabrics. In later layers, you'll see specific types of clothing.

MIXED4E

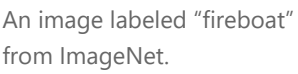
And further still in mixed4e

Loading ...

Loading ...

Focusing on a Single Classification

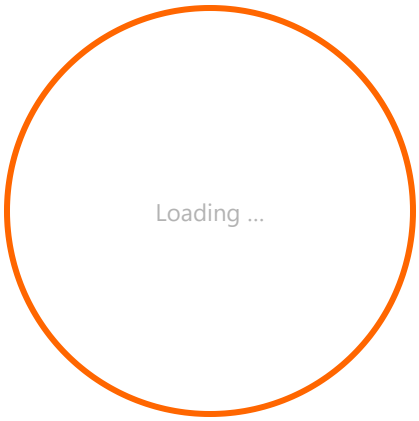
Looking at an atlas of all activations can be a little overwhelming, especially when you're trying to understand how the network goes about ranking one particular class. For instance, let's investigate how the network classifies a "fireboat".



An image labeled "fireboat" from ImageNet.

Loading ...

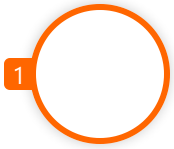
We'll start by looking at an atlas for the last layer, mixed5b. Instead of showing all the activations, however, we'll calculate the amount that each activation contributes toward a classification of "fireboat" and then map that value to the opacity of the activation icon. [4] The areas that contribute a lot toward a classification of "fireboat" will be clearly visible, whereas the areas that contribute very little (or even contribute negatively) will be completely transparent.

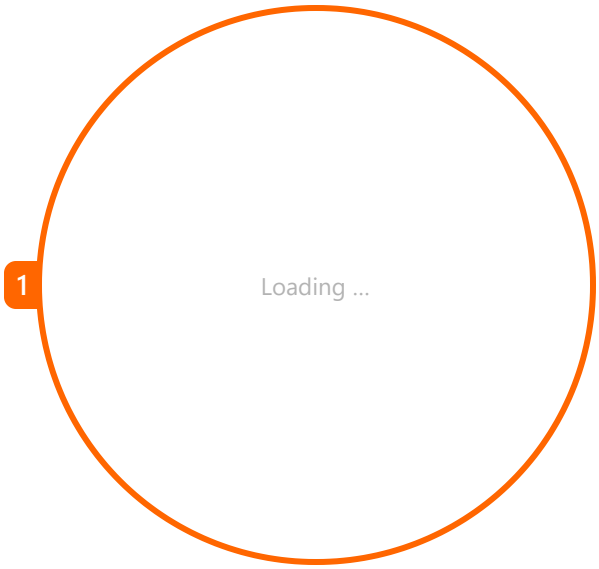


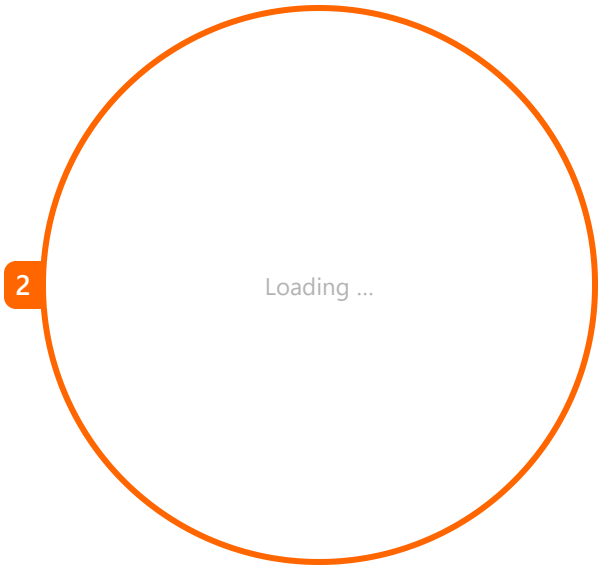
When we map opacity to the amount that each activation contributes to "fireboat" in the layer mixed5b, we see a main cluster of icons showing red boats and splashing, spraying water. While there are some stray areas elsewhere, it seems that this is region of the atlas that is dedicated specifically to classifying red boats with splashing water nearby.

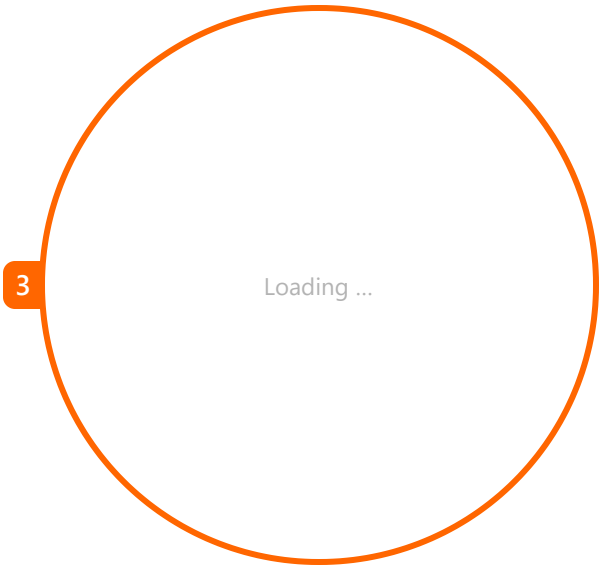
The layer we just looked at, mixed5b, is located just before the final classification layer so it seems reasonable that it would be closely aligned with the final classes. Let's look at a layer a little earlier in the network, say mixed4d, and see how it differs.

Loading ...









In mixed4d we see we see the attribution toward "fireboat" is high in several clusters located in different positions around the atlas. One is very focused on windows, another on geysers and splashing water, and yet another on crane-like objects.

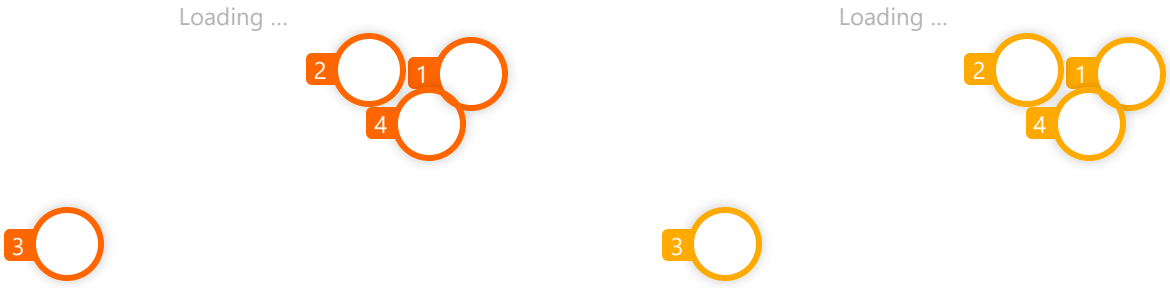
Here we see a much different pattern. If we look at some more input examples, this seems entirely reasonable. It's almost as if we can see a collection of the component concepts the network will use in later layers to classify "fireboat". Windows + crane + water = "fireboat".

Loading ... Loading ... Loading ...

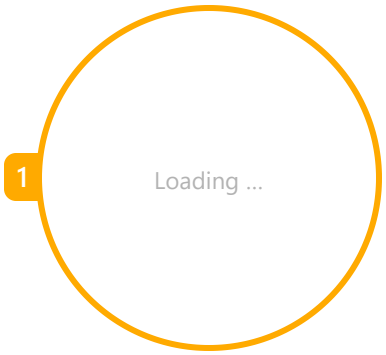
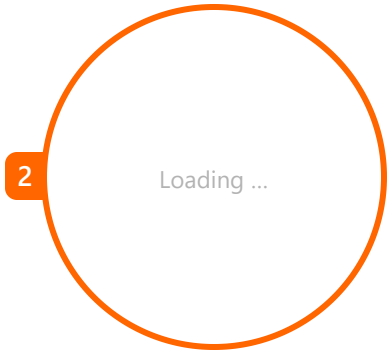
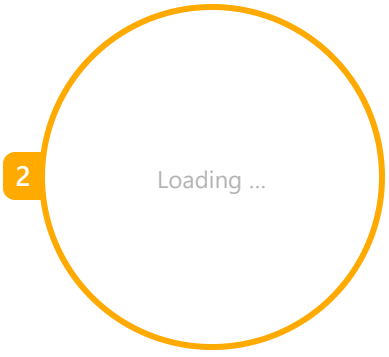
One of the clusters, the one with windows, has strong attribution to "fireboat", but taken on its own, it has an even stronger attribution toward "streetcar". So, let's go back to the atlas at mixed4d, but isolate "streetcar" and compare it to the patterns seen for "fireboat". Let's look more closely at the four highlighted areas: the three areas we highlighted for fireboat plus one additional area that is highly activated for streetcars.

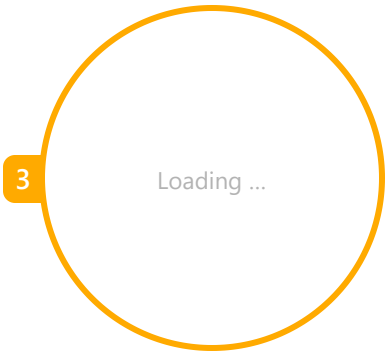
ACTIVATIONS FOR FIREBOAT

ACTIVATIONS FOR STREETCAR

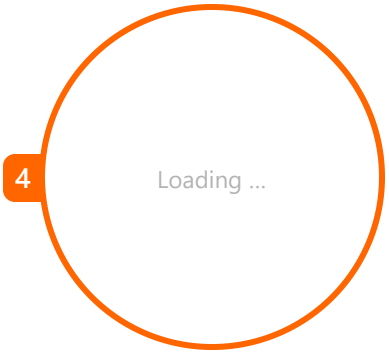


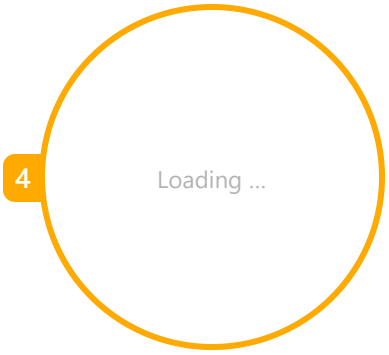
If we zoom in, we can get a better look at what distinguishes the two classifications at this layer. (We've cherry-picked these examples for brevity, but you can explore all the layers and activations in detail in a explorable playground below.)

FIREBOAT	STREETCAR	
		At mixed4d, Both "streetcar" and "fireboat" contain activations for what look like windows.
		Both classes contain activations for crane-like apparatuses, though they are less prominent than the window activations.



"Fireboat" activations have much stronger attributions from water than "streetcar", where there is virtually no positive evidence.





The activations for "streetcar" have much stronger attributions from buildings than does "fireboat".

If we look at a couple of input examples, we can see how buildings and water backgrounds are an easy way to differentiate between a “fireboat” and a “streetcar”.

Images from ImageNet

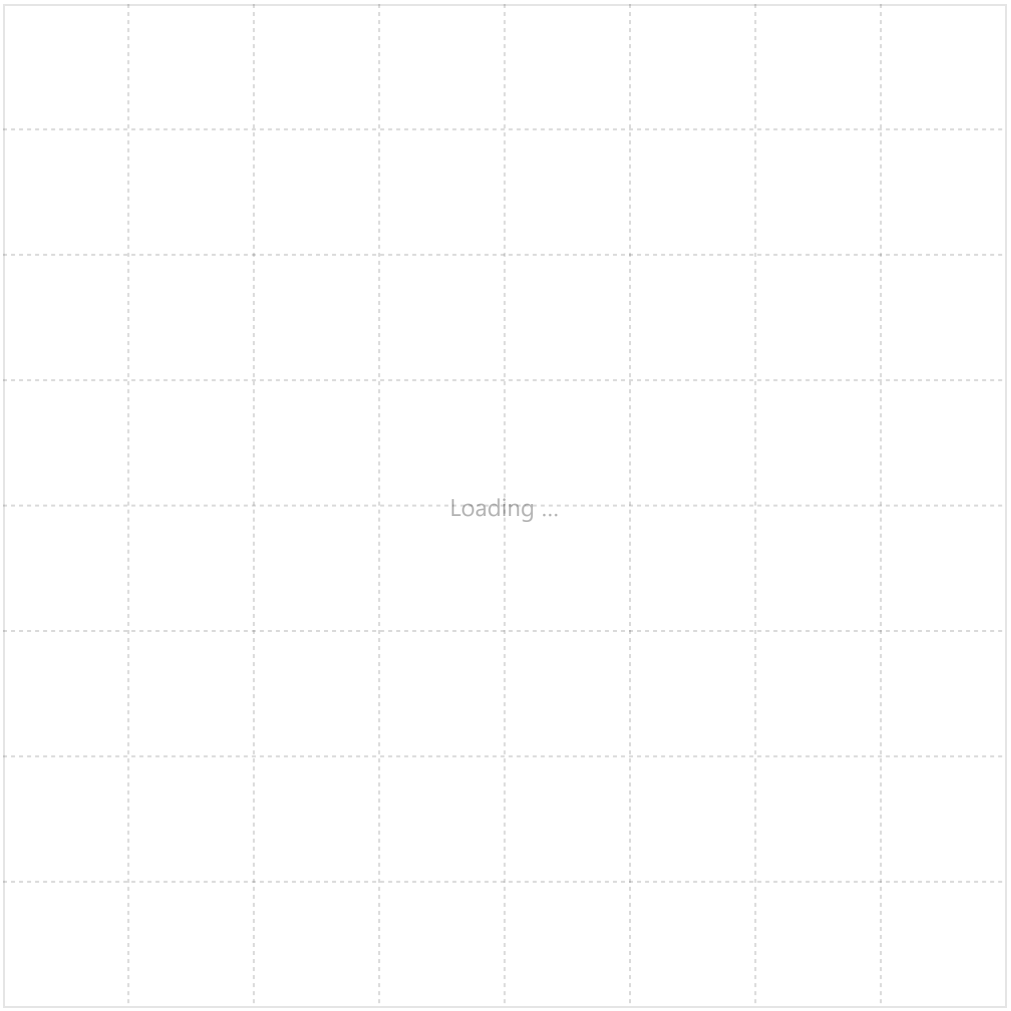
By isolating the activations that contribute strongly to one class and comparing it to other class activations, we can see which activations are conserved among classes and which are recombined to form more complex activations in later layers. Below you can explore the activation patterns of many classes in ImageNet through several layers of InceptionV1. You can even explore negative attributions, which we ignored in this discussion.

Loading ...

Further Isolating Classes

Highlighting the class-specific activations in situ of a full atlas is helpful for seeing how that class relates to the full space of what a network "can see." However, if we want to really isolate the activations that contribute to a specific class we can remove all the other activations rather than just dimming them, creating what we'll call a *class activation atlas*. Similar to the general atlas, we run dimensionality reduction [5] over the class-specific activation vectors in order to arrange the feature visualizations shown in the class activation atlas.

CLASS ACTIVATION ATLAS FOR "SNORKEL" FROM MIXED5B



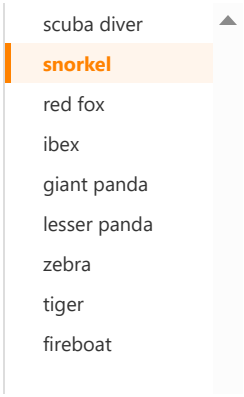
TRY IN A NOTEBOOK

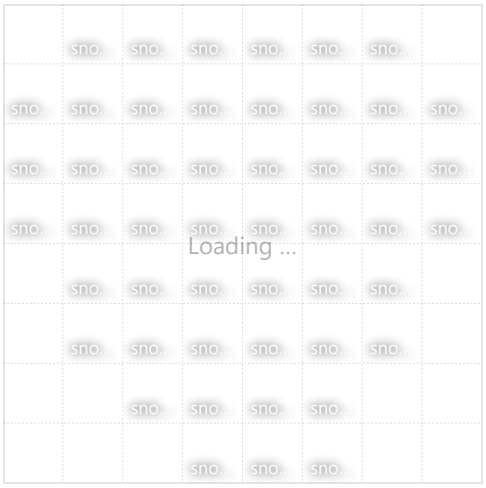
A class activation atlas gives us a much clearer view of which detectors the network is using to rank a specific class. In the “snorkel” example we can clearly see ocean, underwater, and colorful masks.

In the previous example, we are only showing those activations whose strongest attribution is toward the class in question. This will show us activations that contribute mostly to our class in question, even if their overall strength is low (like in background detectors). In some cases, though, there are strong correlations that we’d like to see (like fish with snorkelers). These activations on their own might contribute more strongly to a different class than the one we’re interested in, but their existence can also contribute strongly to our class of interest. For these we need to choose a different filtering method.

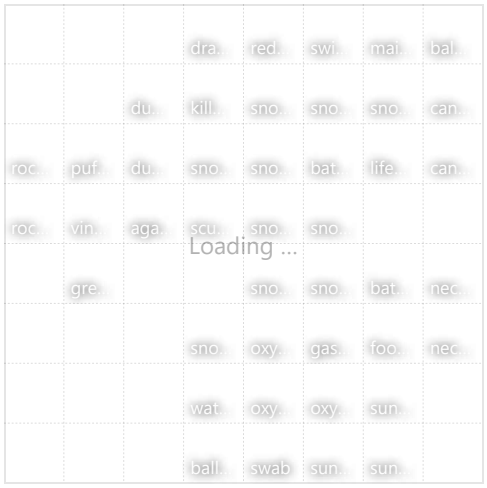
“SNORKEL”
FILTERED BY TOP RANK

“SNORKEL”
FILTERED BY OVERALL MAGNITUDE





We pluck only those activations whose top attribution is toward the class in question. The results are often much more focused and isolated, exclusive to the class. Some are low magnitude, like backgrounds, and we miss correlations or concepts that are shared among many classes.

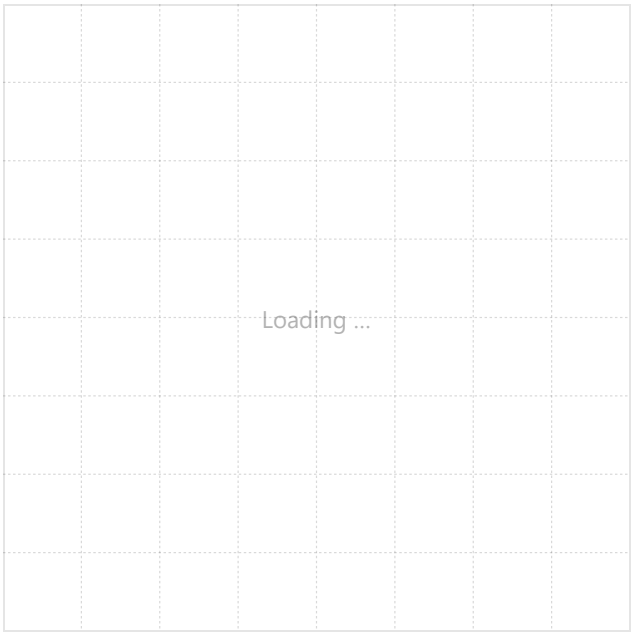


Here we sort all the activations by the magnitude toward the class in question (independent of other classes) and take the top 2,000 activations. We see more correlated activations that could, on their own, contribute to another classification (We label each activation with the class that it contributes toward most). Notice that certain fish that are commonly seen while snorkeling now show up with that class, and a “coffee mug” now shows up with “crossword puzzle”. Some of them appear spurious, however.

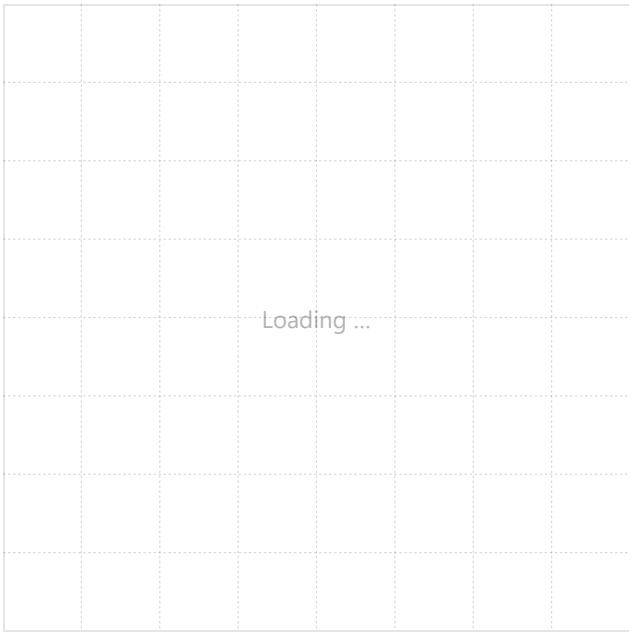
dalmatian
English setter
great white shark
goldfish
lionfish
comic book
plunger
knot
tile roof
pill bottle
measuring cup
crossword puzzle
cauliflower
head cabbage
artichoke
green mamba
goldfinch
house finch
vulture
chainlink fence
wok
grey whale
waffle iron
conch

Using the magnitude filtering method, let’s try to compare two related classes and see if we can more easily see what distinguishes them. (We could have instead used rank, or a combination of the two, but magnitude will suffice to show us a good variety of concepts).

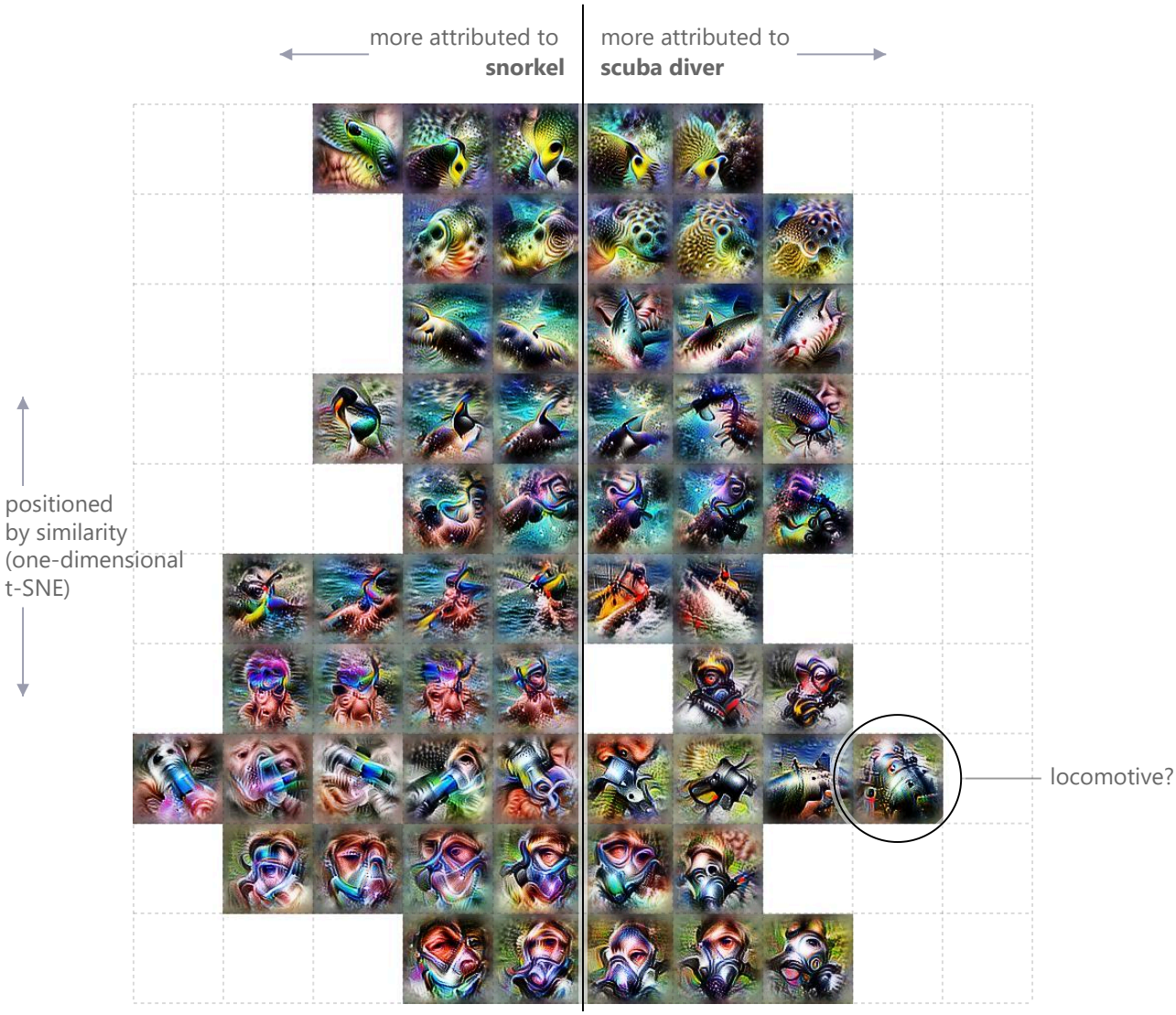
“SNORKEL”



“SCUBA DIVER”



It can be a little hard to immediately understand all the differences between classes. To help make the comparison easier, we can combine the two views into one. We'll plot the difference between the attributions of the "snorkel" and "scuba diver" horizontally, and use t-SNE [14] to cluster similar activations vertically.



In this comparison we can see some bird-like creatures and clear tubes on the left, implying a correlation with "snorkel", and some shark-like creatures and something round, shiny, and metallic on the right, implying correlation with "scuba diver" (This activation has a strong attribution toward the class "steam locomotive"). Let's take an image from the ImageNet dataset labeled as "snorkel" and add something that resembles this icon to see how it affects the classification scores.

Loading ...

1.	**snorkel**	**55.6%**
2.	coral reef	18.6%
3.	scuba diver	13.5%
4.	loggerhead	5.5%
5.	lionfish	1.7%
6.	sea snake	1.4%

Image from ImageNet labeled as “snorkel” with the classification confidences from Inceptionv1 for the top six classes.

Loading ...

1.	**scuba diver**	**71.3%**
2.	coral reef	14.7%
3.	snorkel	4.5%
4.	lionfish	3.2%
5.	sea snake	2.4%
6.	loggerhead	0.9%

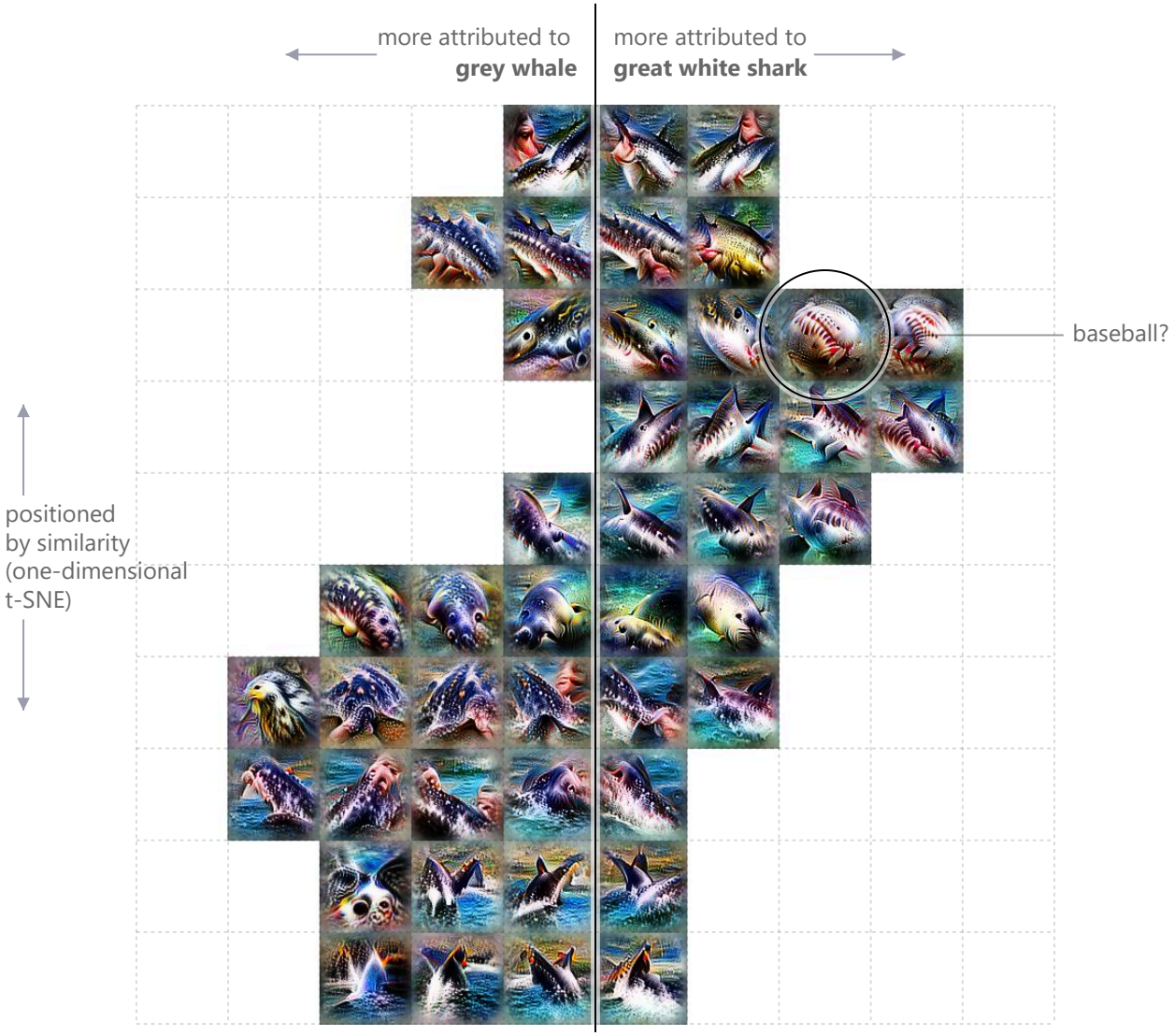
By adding a picture of one of the concepts seen in the visualization above we can change the classification. With an added picture of a steam train the confidence of “scuba diver” classification rises and “snorkel” drops significantly.

Loading ...

1.	**steam locomotive**	**89.6%**
2.	snowplow	6.2%
3.	jeep	0.8%
4.	tractor	0.5%
5.	scuba diver	0.4%
6.	passenger car	0.3%

As the picture of the train gets bigger, the classification for “steam locomotive” overwhelms, but “scuba diver” remains.

The failure mode here seems to be that the model is using its detectors for the class “steam locomotive” to identify air tanks to help classify “scuba diver”. We’ll call these “multi-use” features — detectors that react to very different concepts that are nonetheless visually similar. Let’s look at the differences between a “grey whale” and a “great white shark” to see another example of this issue.



In this example we see another detector that seems to be playing two roles: detecting red stitching on a baseball and a sharks’s white teeth and pink inner mouth. This detector also shows up in the activation atlas at layer mixed5b filtered to “great white shark” and its attribution points towards all sorts of balls, the top one being “baseball”.

Let’s add a picture of a baseball to a picture of a “grey whale” from ImageNet and see how it effects the classification.

Loading ...

1.	**grey whale**	**91.0%**
2.	killer whale	7.5%
3.	great white shark	0.7%
4.	gar	0.4%
5.	sea lion	0.1%
6.	tiger shark	0.1%

Loading ...

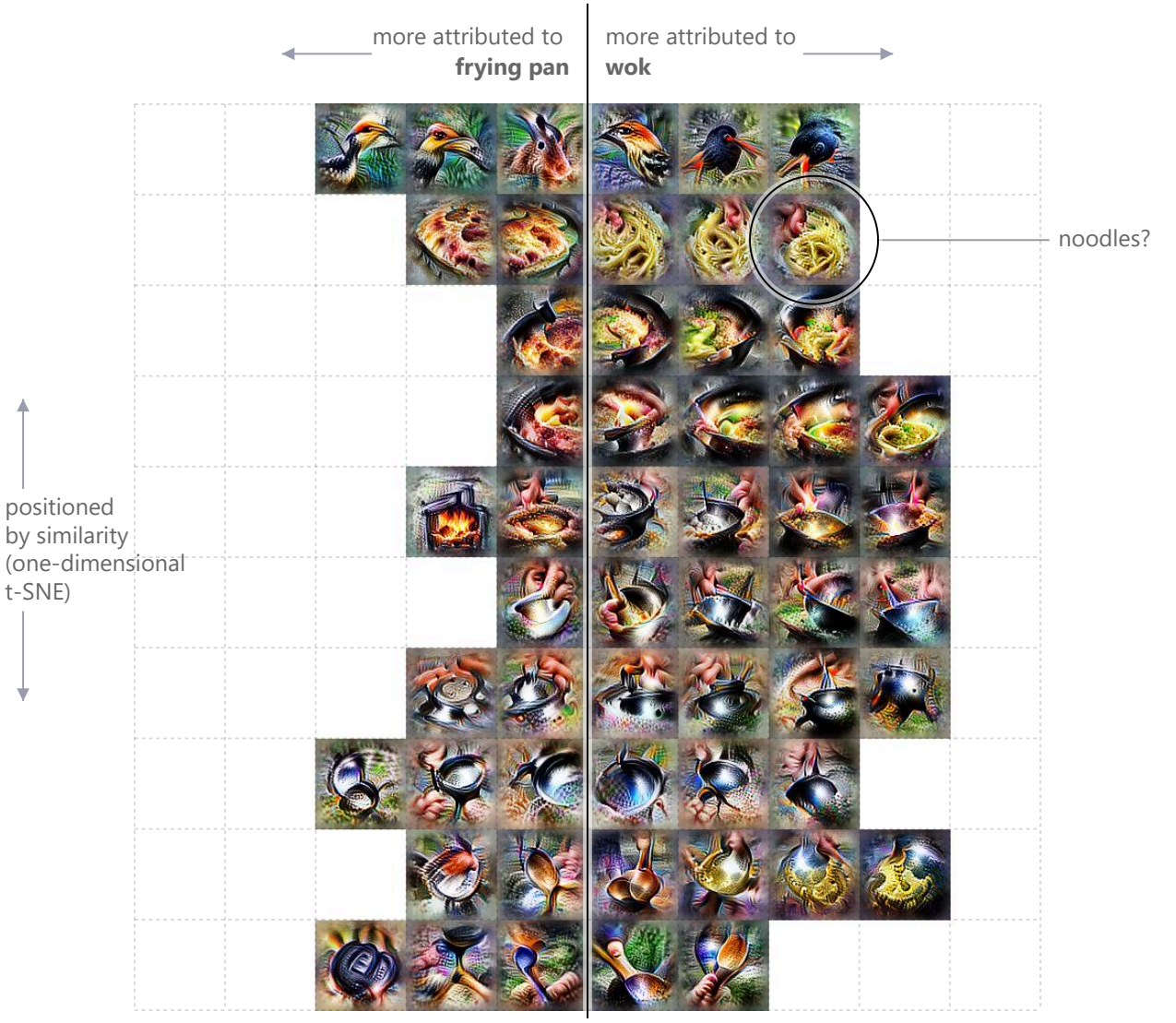
1.	**great white shark**	**66.7%**
2.	baseball	7.4%
3.	grey whale	4.1%
4.	sombrero	3.2%
5.	sea lion	3.1%
6.	killer whale	2.7%

Loading ...

1.	**baseball**	**100.0%**
2.	rugby ball	0.0%
3.	golf ball	0.0%
4.	ballplayer	0.0%
5.	drum	0.0%
6.	sombrero	0.0%

The results follow the pattern in previous examples pretty closely. Adding a small-sized baseball does change the top classification to “great white shark”, and as it gets bigger it overpowers the classification, so the top slot goes to “baseball”.

Let’s look at one more example: “frying pan” and “wok”.



One difference stands out here — the type of related foods present. On the right we can clearly see something resembling noodles (which have a strong attribution toward the class “carbonara”). Let’s take a picture from ImageNet labeled as “frying pan” and add an inset of some noodles.

Loading ...

1.	**frying pan**	**76.5%**
2.	wok	15.8%
3.	stove	5.4%
4.	spatula	1.0%
5.	Dutch oven	0.5%
6.	mixing bowl	0.2%

Image from ImageNet labeled as “frying pan” with the classification confidences from Inceptionv1 for the top six classes.

Loading ...

1.	**wok**	**63.2%**
2.	frying pan	35.1%
3.	spatula	0.6%
4.	hot pot	0.5%
5.	mixing bowl	0.1%
6.	stove	0.1%

By adding a picture of some noodles, also from ImageNet, we can see the top classification change from “frying pan” to “wok.”

Loading ...

1.	**carbonara**	**96.4%**
2.	plate	2.3%
3.	wok	0.6%
4.	meat loaf	0.1%
5.	dishwasher	0.1%
6.	rotisserie	0.1%

As we make the picture of noodles larger, its influence overpowers the other classifications, but “wok” remains ranked above “frying pan”.

Here the patch was not as effective at lowering the initial classification, which makes sense since the noodle-like icons were plotted more toward the center of the visualization thus having less of a difference in attribution. We suspect that the training set simply contained more images of woks with noodles than frying pans with noodles.

Testing dozens of patches on thousands of images

So far we’ve only shown single examples of these patches. Below we show the result of ten sample patches (each set includes the one example we explored above), run on 1,000 images from the ImageNet training set for the class in question. While they aren’t effective in all cases, they do flip the image classification to the target class in about 2 in 5 images. The success rate reaches about 1 in 2 images if we also allow to position the patch in the best of the four corners of the image (top left, top right,

bottom left, bottom right) at the most effective size. To ensure our attack isn't just blocking out evidence for the original class, we also compare each attack to a random noise image patch.

DYNAMIC CORNER POSITION

Choosing the strongest size and corner position for each example.

	Success rate Previously correctly classified images that the patch changed to the target class.		**Probability change** Mean probability assigned to the target class before and after patch.	
snorkel → scuba diver	85.0%	7,031 / 8,270	+56.8%	3.7% → 60.5%
grey whale → great white shark	40.5%	3,539 / 8,740	+29.8%	1.8% → 31.6%
frying pan → wok	44.9%	2,799 / 6,240	+24.1%	10.3% → 34.4%

CONSISTENT CORNER POSITION

Choosing a consistent size and corner position across all examples.

	Success rate		**Probability change**	
snorkel → scuba diver	65.4%	5,409 / 8,270	+43.0%	3.7% → 46.7%
grey whale → great white shark	18.8%	1,639 / 8,740	+16.3%	1.8% → 18.1%
frying pan → wok	25.6%	1,595 / 6,240	+12.7%	10.3% → 23.0%

DETAILS FOR EACH PATCH TESTED

snorkel → scuba diver

Strongest size and corner positior

	Success rate		**Probability change**	
 Patch 01	88.1%	729 / 827	+60.3%	3.7% → 64.0%
 Patch 02	83.4%	690 / 827	+53.0%	3.7% → 56.7%
 Patch 03	92.4%	764 / 827	+66.9%	3.7% → 70.6%
 Patch 04	93.0%	769 / 827	+66.7%	3.7% → 70.4%
 Patch 05	88.0%	728 / 827	+58.8%	3.7% → 62.5%
 Patch 06	85.5%	707 / 827	+52.0%	3.7% → 55.7%
 Patch 07	86.2%	713 / 827	+57.4%	3.7% → 61.2%
 Patch 08	56.1%	464 / 827	+30.7%	3.7% → 34.5%

	Success rate		Probability change	
 Patch 09	92.0%	761 / 827	+64.8%	3.7% → 68.5%
 Patch 10	85.4%	706 / 827	+57.6%	3.7% → 61.3%
 Random noise	13.9%	115 / 827	+9.5%	3.7% → 13.2%

Our "attacks" can be seen as part of a larger trend (eg. [17, 18, 19, 20]) of researchers exploring input attacks on models other than the traditional epsilon ball adversarial examples [21]. In many ways, our attacks are most similar to adversarial patches [17], which also add a small patch to the input image. From this perspective, adversarial patches are far more effective, working much more reliably. Instead, we see our attacks as interesting because they are synthesized by humans from their understanding of the model, and seem to be attacking the model at a higher level of abstraction.

We also want to emphasize that not all class comparisons reveal these type of patches and not all icons in the visualization have the same (or any) effectiveness and we've only tested them on one model. If we wanted to find these patches more systematically, a different approach would most likely be more effective. However, the class activation atlas technique was what revealed the existence of these patches before we knew to look for them. If you'd like to explore your own comparisons and search for your own patches, we've provided a notebook to get you started: TRY IN A NOTEBOOK

Conclusion and Future Work

Activation atlases give us a new way to peer into convolutional vision networks. They give us a global, hierarchical, and human-interpretable overview of concepts within the hidden layers. Not only does this allow us to better see the inner workings of these complicated systems, but it's possible that it could enable new interfaces for working with images.

Surfacing Inner Properties of Models

The vast majority of neural network research focuses on quantitative evaluations of network behavior. How accurate is the model? What's the precision-recall curve?

While these questions can describe how the network behaves in specific situations, it doesn't give us a great understanding of *why* it behaves the way it does. To truly understand why a network behaves the way it does, we would need to fully understand the rich inner world of the network — it's hidden layers. For example, understanding better how InceptionV1 builds up a classifier for a fireboat from component parts in mixed4d can help us build confidence in our models and can surface places where it isn't doing what we want.

Engaging with this inner world also invites us to do deep learning research in a new way. Normally, each neural network experiment gives only a few bits of feedback — whether the loss went up or down — to inform the next round of experiments. We design architectures by almost blind trial and error, guided by vague intuitions that we build up over years. In the future, we hope that researchers will get rich feedback on what each layer in their model is doing in a way that will make our current approach seem like stumbling in the dark.

Activation atlases, as they presently stand, are inadequate to really help researchers iterate on models, in part because they aren't comparable. If you look at atlases for two slightly different models, it's hard to take away anything. In future work, we will explore how similar visualizations can compare models, showing similarities and differences beyond error rates.

New interfaces

Machine learning models are usually deployed as black boxes that automate a specific task, executing it on their own. But there's a growing sense that there might be an alternate way for us to relate to them: that instead of increasingly automating a task, they could be used more directly by a person. One vision of this augmentation that we find particularly compelling is the idea that the internal representations neural networks learn can be repurposed as tools [22, 23, 24]. Already, we've seen exciting demonstrations of this in images [25, 26, 27] and music [28, 29] .

We think of activation atlases as revealing a machine-learned alphabet for images — a collection of simple, atomic concepts that are combined and recombined to form much more complex visual ideas. In the same way that we use word processors to turn letters into words, and words into sentences, we can imagine a tool that would allow us to create images from a machine-learned language system for images. Similar to GAN painting [30], imagine using something like an activation atlas as a palette — one could dip a brush into a "tree" activation and paint with it. A palette of concepts rather than colors.

While classification models are not generally thought of as being used to generate images, techniques like [5] has shown that this is entirely possible. In this particular instance, we imagine constructing a grid of activations by selecting them from an atlas (or some derivation), then optimizing an output image that would correspond to the user's constructed activation matrix.

Such a tool would not necessarily be limited to targeting realistic images either. Techniques like style transfer [31] have shown us that we can use these vision networks to create nuanced visual expression outside the explicit distribution of visual data it was trained on. We speculate that activation atlases could be helpful in manipulating artistic styles without having to find an existing reference image, or they could help in guiding and modifying automated style transfer techniques.

We could also use these atlases to query large image datasets. In the same way that we probe large corpuses of text with words, we could, too, use activation atlases to find types of images in large image datasets. Using words to search for something like a “tree” is quite powerful, but as you get more specific, human language is often ill-suited to describe specific visual characteristics. In contrast, the hidden layers of neural networks are a language optimized for the sole purpose of representing visual concepts. Instead of using the proverbial thousand words to uniquely specify the image one is seeking, we can imagine someone using the language of the activation atlas.

And lastly, we can also liken activation atlases to histograms. In the same way that traditional histograms give us good summaries of large datasets, activation atlases can be used to summarize large numbers of images.

In the examples in this article we used the same dataset for training the model as we did for collecting the activations. But, if we use a different dataset to collect the activations, we could use the atlas as a way of inspecting an unknown dataset. An activation atlas could show us a histogram of *learned concepts* that exist within the images. Such a tool could show us the semantics of the data and not just visual similarities, like showing histograms of common pixel values.

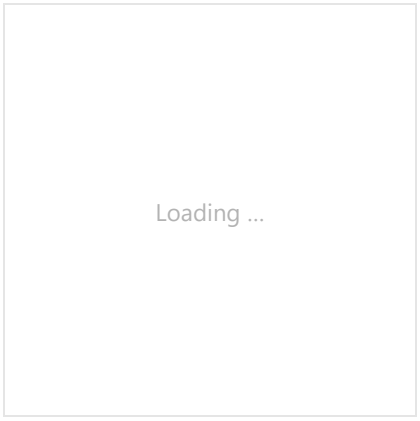
While we are excited about the potential of activation atlases, we are even more excited at the possibility of developing similar techniques for other types of models. Imagine having an array of machine learned, but human interpretable, languages for images, audio and text.

Technical Notes and Limitations

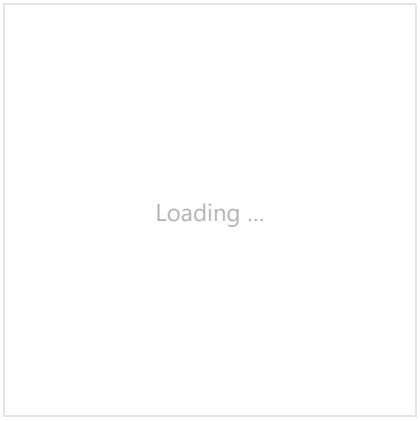
In this section we note some of the limits and pitfalls that we’ve noticed while developing and working with activation atlases.

Sampling based

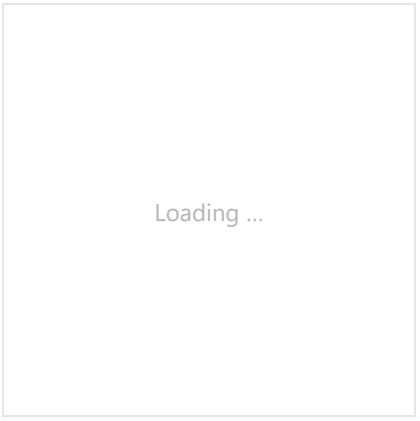
Activation Atlases are a sample-based method which can only show a manifold of sampled activations. First, the dataset from which those activations are sampled from needs to come from the same distribution as the one that we are interested in. In this article we sample from the training set because we are interested in what the model has learned to recognize. Second, we need to provide enough samples to span the full manifold we want to observe. In this article we generally used one million activations but we found that 100,000 activations was often sufficient.



10,000 activations



100,000 activations



1,000,000 activations

Doesn't Surface Compositionality

Neural network activations have an underlying compositional, combinatorial structure. We can mix together a couple hundred basis neurons to get any activation vector. Unfortunately, this has the problem of being an exponentially large space and hard to find the interesting activations. Activation Atlases solve this problem by sampling the interesting activation vectors, but completely lose the original compositional structure.

By not surfacing the compositional structure, activation atlases don't really support us thinking about novel combinations of directions. They also necessarily can't show a lot of connections between different parts of the atlas, and have to collapse local structure down to two dimensions.

Surfacing compositionality is deeply connected to the high-dimensional nature of the original space. As a result, it's naturally hard to do in 2D. That said, it may be possible to partially surface compositionality, like the neuron addition diagram in Feature Visualization [2] did. Another early prototype is a neuron activation grid. We vaguely suspect that there may be a way to thread the needle with clever use of matrix factorization, but we do not yet know how.

No alignment between layers

There's no way to link two views together. For instance, when looking at two layers side-by-side, similar features appear in random locations.

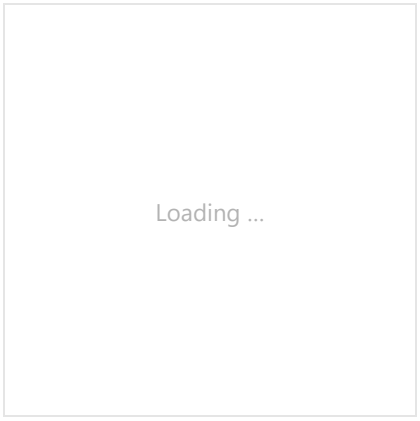
Computationally expensive

It's computationally expensive. While it depends on many factors, activation atlases generally take somewhere between several minutes and several hours, end to end.

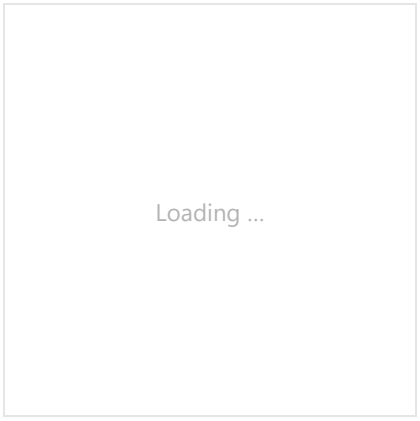
Dependency on hyperparameters

Because it is based on dimensionality reduction, the final output can be very sensitive to the hyperparameters chosen for the reduction step. UMAP and t-SNE are both reasonable choices and each could also produce different results.

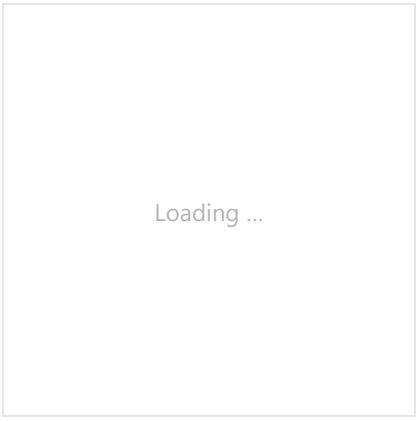
Because the choice of a grid size is somewhat arbitrary, much like choosing a histogram bin size [32], your choice of grid size will determine the effectiveness of the final output. Choosing the wrong bin size can sometimes *hide* important features in the averages. In some of the later examples we bind this value to the zoom level so the detail of the visualizations dynamically change as you zoom in. A more principled clustering method might be better.



32 x 32 When the grid is too small, the feature visualizations are too small to see.



8 x 8 When just right, the activation atlas shows all the concepts with feature visualizations that are big enough to see.



2 x 2 When the grid is too big, concepts are lost in the averages. One sees less diversity within related concepts.

Dimensionality reduction vs clustering

One might think that a true clustering algorithm, like k-means, might produce a more robust decomposition of the activation manifold. After all, centroids produced by such an algorithm are by definition close to clusters of activations the network produces, a property that discretizing the dimensionality-reduced activations doesn't guarantee. We experimented with different clustering techniques such as k-means, spherical k-means, and DBSCAN. However, the images produced by visualizing the resulting centroids were subjectively worse and less interpretable than the technique described in this article. Also, dimensionality reduction followed by 2D binning allows for multiple levels of detail while preserving spatial consistency, which is necessary for making atlasses zoomable. Thus, we preferred that method over clustering in this article — but finding tradeoffs between these techniques remains an open question.

Author Contributions

Shan Carter wrote the majority of the article and performed most of the experiments. Zan Armstrong helped with the interactive diagrams and the writing. Ludwig Schubert provided technical help throughout and performed the numerical analysis of the manual patches. Ian Johnson provided inspiration for the original idea and advice throughout. Chris Olah provided essential technical contributions and substantial writing contributions throughout.

Acknowledgments

Thanks to Kevin Quealy and Sam Greydanus for substantial editing help. Thanks to Colin Raffel, Arvind Satyanarayan, Alexander Mordvintsev, and Nick Cammarata for additional feedback during development.

We're also very grateful to Phillip Isola for stepping in as acting Distill editor for this article, and to our reviewers who took time to give us feedback, significantly improving our paper.

The photo used to illustrate a sub-manifold in the introduction was taken by Alexandru-Bogdan Ghita.

Discussion and Review

Review 1 - Anonymous

Review 2 - Anonymous

Review 3 - David Bau

Footnotes

1. More specifically, We show a linear approximation of the effect of these average activation vectors of a grid cell on the logits. [↩]

2. With the exception of the first hidden layer. [↩]

3. We avoid the edges due to boundary effects. [↩]

4. In the case of mixed5b, determining this contribution is fairly straightforward because the relationship between activations at mixed5b and the logit values is linear. When there are multiple layers between our present one and the output — and as a result, the relationship is non-linear — it's a little less clear what to do. In this article, we take the simple approach of forming a linear approximation of these future layers and use it to approximate the effect of our activations. [↩]

5. For class activations we generally have better results from using t-SNE for the dimensionality reduction step rather than UMAP. We suspect it is because the data is much more sparse. [↩]

References

1. **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.

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

3. **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.

4. **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

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

6. **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.

7. **Understanding deep image representations by inverting them** [PDF]
   Mahendran, A. and Vedaldi, A., 2015. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5188--5196. DOI: 10.1109/cvpr.2015.7299155

8. **The Building Blocks of Interpretability**
   Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K. and Mordvintsev, A., 2018. Distill. DOI: 10.23915/distill.00010

9. **t-SNE visualization of CNN codes** [link]
   Karpathy, A., 2014.

10. **Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks** [PDF]
    Nguyen, A., Yosinski, J. and Clune, J., 2016. arXiv preprint arXiv:1602.03616.

11. **Imagenet: A large-scale hierarchical image database** [PDF]
    Deng, J., Dong, W., Socher, R., Li, L., Li, K. and Fei-Fei, L., 2009. Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 248--255. DOI: 10.1109/cvprw.2009.5206848

12. **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

13. **Imagenet large scale visual recognition challenge**
Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. and others,, 2015. International Journal of Computer Vision, Vol 115(3), pp. 211--252. Springer.

14. **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.

15. **UMAP: Uniform Manifold Approximation and Projection**
McInnes, L., Healy, J., Saul, N. and Grossberger, L., 2018. The Journal of Open Source Software, Vol 3(29), pp. 861.

16. **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.

17. **Adversarial Patch** [PDF]
Brown, T.B., Mané, D., Roy, A., Abadi, M. and Gilmer, J., 2017. CoRR, Vol abs/1712.09665.

18. **Adversarial examples in the physical world**
Kurakin, A., Goodfellow, I. and Bengio, S., 2016. arXiv preprint arXiv:1607.02533.

19. **A rotation and a translation suffice: Fooling cnns with simple transformations**
Engstrom, L., Tran, B., Tsipras, D., Schmidt, L. and Madry, A., 2017. arXiv preprint arXiv:1712.02779.

20. **Unrestricted adversarial examples**
Brown, T.B., Carlini, N., Zhang, C., Olsson, C., Christiano, P. and Goodfellow, I., 2018. arXiv preprint arXiv:1809.08352.

21. **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.

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

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

24. **Machine Learning for Visualization** [link]
Johnson, I., 2018.

25. **Image-to-image translation with conditional adversarial networks**
Isola, P., Zhu, J., Zhou, T. and Efros, A.A., 2017. arXiv preprint.

26. **TopoSketch: Drawing in Latent Space**
Loh, I. and White, T., 2017. NIPS Workshop on Machine Learning for Creativity and Design.

27. **Generative visual manipulation on the natural image manifold**
Zhu, J., Krahenbuhl, P., Shechtman, E. and Efros, A.A., 2016. European Conference on Computer Vision, pp. 597--613.

28. **ML as Collaborator: Composing Melodic Palettes with Latent Loops** [link]
McCurry, C., 2018.

29. **A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music**
Roberts, A., Engel, J., Raffel, C., Hawthorne, C. and Eck, D., 2018. arXiv preprint arXiv:1803.05428.

30. **GAN Dissection: Visualizing and Understanding Generative Adversarial Networks** [link]
Bau, D., Zhu, J., Strobelt, H., Bolei, Z., Tenenbaum, J.B., Freeman, W.T. and Torralba, A., 2018. arXiv preprint arXiv:1811.10597.

31. **A neural algorithm of artistic style** [PDF]
Gatys, L.A., Ecker, A.S. and Bethge, M., 2015. arXiv preprint arXiv:1508.06576.

32. **Exploring Histograms** [link]
Lunzer, A. and McNamara, A., 2017.

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

```
Carter, et al., "Activation Atlas", Distill, 2019.
```

BibTeX citation

```
@article{carter2019activation,
  author = {Carter, Shan and Armstrong, Zan and Schubert, Ludwig and Johnson, Ian and Olah, Chris},
  title = {Activation Atlas},
  journal = {Distill},
  year = {2019},
  note = {https://distill.pub/2019/activation-atlas},
  doi = {10.23915/distill.00015}
}
```