Feature Visualization for Damped Ly-alpha Absorption Lines Convolutional Neural Network

Project by
Samuel Gabbard

MS Project Chair  MS Project Reader  MS Project Supervisor
Prof. Alex Pang  Prof. Angus Forbes  Prof. Xavier Prochaska
Highlights

Feature visualization for Damped Ly-alpha Absorption Lines Convolutional Neural Network
Samuel Gabbard

- Demonstrate how to apply existing feature visualization techniques to one-dimensional models
- Explore and analyze features of a convolutional neural network trained on spectra sightline data
- Create visualization tool to analyze spectra data passed into the convolutional neural network model
Feature visualization for Damped Ly-alpha Absorption Lines Convolutional Neural Network

Samuel Gabbard\textsuperscript{a,e,1} (Researcher)

\textsuperscript{a}University of California Santa Cruz, 1156 High St. Santa Cruz, CA, United States

\textbf{ARTICLE INFO}

\textbf{Keywords:}
interpretability
feature visualization
optimization objectives
DLA’s

\textbf{ABSTRACT}

Neural networks (NN) are well known for their ability to find patterns in a diverse set of data, e.g. text, video, images and more. Specifically, convolutional neural networks (CNN) have performed very well in problems dealing with image classification and image recognition. Despite CNN’s high levels of accuracy, it is very difficult for humans to determine what the model is learning, and how it interprets different features of an image. Since these networks are differentiable by nature, we can optimize an input image to activate a particular neuron or channel, which allows us to visualize the different features a model has learned. There has been a lot of success performing this feature visualization on CNNs trained on traditional 3-D images, however it has not been applied to CNNs with one-dimensional input. In this project we explore just that, by applying feature visualization to a 1-Dimensional CNN trained to detect damped Ly-alpha (DLA) Absorption Lines in astronomical spectra. At a minimum, these efforts provide insight on how to improve the algorithm. More ambitious, we may test whether the machine has learned how to predict the underlying profile predicted from quantum mechanics.

\textbf{1. Introduction}

Humans and CNN’s differ in the way they process data, which makes interpreting these algorithms a challenging task. When a CNN is performing image classification, it is often times finding high level features that are uninterpretable. This means although these models may perform with high accuracy, it is difficult for humans to assess or even visualize what has been learned. The issue of interpreting complex models has become an important topic in deep learning.

A breakthrough technique introduced by Erhan et al. [3], outlines a process of understanding the hidden units of a CNN, by maximizing the activation of a specific hidden unit. In doing this, we visualize what a unit inside a model has been trained to learn. Over the years, several papers have taken this idea and developed it to learn more about deep networks[6, 4, 13, 11, 7]. One motivation behind this field of research, is that if we are able to better understand how deep networks learn, we can better develop these models. Another is to provide a level of interpretability which is vital to legal, ethical, and political concerns on the application of deep learning.

One specific recent work that has gained traction in the deep learning community is the paper ‘Feature Visualization’. by Olah et al. [8]. This paper is a part of a larger thread of research being pursued by members of Google Research and the Google Brain Team, that creates clean and interpretable images for high-level features. This is done by optimizing an input image to maximize a given “objective”. An objective can be anything from a neuron, channel, or even entire layer. In addition to Feature Visualization, the authors have created other techniques to further understand complex models [5, 9, 1]. In ‘Differentiable Image Parameterization’, Mordvintsev et al. [5] introduce different ways to parameterize the input image to improve optimization. In ‘Building Blocks of Interpretability’, Olah et al. [9] combine feature visualization with real data images to understand how the model is interpreting a given data set example. Finally, in ‘Activation atlas’, Carter et al. [1] demonstrate a method to show millions of activations from a classification network in an interpretable way using dimensionality reduction techniques.

At the heart of the efforts of the Google Research and Google Brain Teams, lies feature visualization. Impressively, the creators have made available the infrastructure used to create their visualizations. Furthermore, they demonstrate the power behind these tools by applying feature visualization to the award-winning Inception convolutional neural network [12]. Inception is a complex CNN trained on millions of images from the imagenet dataset which classifies images into 1000 distinct classes [2]. Applying feature visualization to the Inception model created powerful visualizations that gave insight into high level features the model has learned. Of all the models studied under this framework, all of them, including Inception, take in traditional three-dimensional image input (height x width x color channel). This means the visualizations of features are also three-dimensional which makes them easy to interpret.

But what if we want to apply feature visualization to a CNN with one-dimensional input? And, what if these one-dimensional data are the stochastic flux spectra of distant quasars, i.e. highly unnatural images? What types of visualizations will appear, and how will we be able to interpret them? In this paper, we address these questions by applying feature visualization to a CNN that was trained on one-dimensional astronomical spectra [10]. To the best of
Feature visualization for DLA CNN

Figure 1: An input image starting from random noise and being optimized over 2048 steps to maximize the activation of a single neuron (layer mixed4a unit 11) from the Inception model.

Figure 2: Channel objective feature visualizations of different layers from the Inception model. Model seems to learn more and more complex features as layers progress.

2. Feature Visualization

The infrastructure mentioned previously that we used in our work is called Lucid. Lucid provides tools for creating visualizations of custom models and is built on Tensorflow. To demonstrate the power behind Lucid and feature visualization, in this section I will highlight some interesting features when tested on the Inception model [12].

2.1. Creating the optimizations

Feature visualization is the process of optimizing an input image to maximize an activation objective. We can think of an objective as whatever unit of the model we are trying to understand. For example, it could be a single neuron in the network, a single channel, or even an entire layer. Once we have defined our objective, we start with a random noisy input image and use gradient descent to optimize the input to maximize our objective. The result is a visual representation of a high level feature that the given objective has been trained to ‘observe’. Figure 1 shows an example of an input image being optimized for a neuron objective. We see the noisy image being tweaked and becoming more clear as the pixel values are changing to optimize the objective.

Although we may create visualizations for every neuron in a model, these optimizations are unlikely to completely describe the model. Neural networks build up there understanding of images not by activating just one neuron at each layer, but rather a combination of neurons. To to better understand our model, we can either create objectives that maximize the activation of multiple neurons, or go broader and look at channel objectives, or even layer objectives. In our work, we found it most helpful to create optimizations of channels, instead of individual neurons.

2.2. Understanding what the model has learned

Once we have created optimizations for our model, we build an understanding of its learning. If we examine how the visualizations change over the hidden layers of the model, we can identify features the model learns. Figure 2 provides an example. The visualizations from left to right are handpicked channel visualizations from the layers mixed3a, mixed4a, mixed4c, and mixed4e of the Inception model. You will notice as we get deeper into the model, we begin to see more complex features being discovered, including some that even represent entire parts of objects.

2.3. Semantic Dictionaries

To derive deeper insight into the network, however, it is even more useful to examine how these visualizations correspond to input images. For example, we may examine how the Inception model interprets an image of a cat and a dog (Figure 3). This is achieved by feeding the image through the network and calculating the activation vectors at each layer. In this example, Inception has reshaped the image of the cat and dog into a shape of 14 x 14 x 528. The 14x14 is the height and width dimensions of the image and the 528 corresponds to the number of channels. Each activation vector represents a part of this new image, and has the activation values of each of these 512 channels. We then just focus on the channels in each section with the highest activation levels to gain insight into how the model is interpreting the input.

The first thing we notice from this figure is that the top image has much higher activation values than the bottom image. This makes sense since the image is that of a dog and a cat, and the bottom example is focusing in on grass and therefore is not an important part of the image. We also see in the top image the highest activated channel seems to resemble the face of a dog and the area of interest is just a patch of fur on the dogs face. This tells us that the channels activated are being influenced by the surrounding parts of the image. It is clear to see that this technique for analyzing input is useful, however it can be expanded further which we detail in section 7.

In the remainder of the paper we will discuss how we applied the tools introduced in this section to the one-dimensional model trained to detect damped Ly-alpha systems in astronomical data [10].

3. CNN for Damped Ly-α detection

In this section we provide a brief overview of the model we will be studying, which was defined by Parks et al. [10]. The model follows a pretty standard CNN architecture; an overview can be found in Figure 4.
Figure 3: Two examples of the top channel visualizations from the fourth layer of the Inception model. The top example is focusing on a section on the dog’s face, and the bottom example is focuses on a section in the grass.

Figure 4: Model architecture for Damped Ly-alpha Convolutional Neural Network. Model follows a standard CNN architecture with three convolutional layers and two fully connected layers.

The model is comprised of three convolutional layers that are each followed by a max pooling layer. The third and final pooling layer feeds into a fully connected layer with 350 neurons. Following the first fully connected layer is the final layer of the model which contains 3 separate fully connected layers, one for detecting DLA classification, one for localization, and one for column density measurements. In our work we are only interested in the first output, DLA classification.

The input to this model is the raw flux values from sightline data. The model uses a sliding window approach and only looks at 400 length segments at a time. An illustration of an example spectra and the sliding window approach can be seen in figure 5.

Before diving into what features our model has learned, it’s important to show what a DLA actually looks like in a spectrum and highlight the features that a human would use to characterize it. Figure 6 gives a zoomed in look of a 400 length spectra sightline with a DLA present in the middle. There are two main features a human would look for when first classifying a DLA. The first strong feature is the set of
nearly zero flux values, that’s apparent from Channels XX-XX. This is where the DLA has entirely absorbed the light from the background source yielding a zero Flux. The second strong features are the wings of the curve, highlighted by the red dashed line. It is key to understand the features a human would use to identify a DLA, because it can help us interpret some of the visualizations we will make.

When creating visualisations, we are mostly interested in the first three layers and not the fully connected layers. In each of the three layers we want to examine the convolutional layer, the convolutional layer immediately after the ReLu activation function has been applied, and the max pooling layer. For the rest of the paper we will refer to these layers as CONV1, CONV1RELU, POOL1, CONV2, CONV2RELU, POOL2, CONV3, CONV3RELU, and POOL3.

4. Feature Visualization of the Damped Ly-α CNN

To apply feature visualization to our DLA model, there were a few steps we needed to follow as outlined by the authors in ‘Feature Visualization’. Thankfully the framework was designed to allow users to import custom models and easily create visualizations. The first step in this process is to create a frozen graph of our model. The authors who designed the DLA CNN model [10] provided us with the code to reconstruct the model graph, the hyper-parameters of the model, and the checkpoint file that held all the weights. From these three pieces we were able to freeze the graph and create a Tensorflow ‘protobuf’ file that held the graph definition with all weights. To fully import our model into the Lucid framework we pass in the ‘protobuf’ file and specify the input shape and the names of the output nodes. With this information Lucid now knows how to create the optimizations, all we need to do is specify the desired objective.

4.1. Transformation Robustness Problems

One of the most significant contributions from Olah et al. [8], is their work in creating images that are clean and free of noise. They do this by applying what is called ‘transformation robustness’. The idea behind transformation robustness is during the optimization process to make minor tweaks to the input image to stabilize the visualization and reduce noise. This is achieved by jittering, rotating, or scaling the input image after each optimization step. An example of the power behind transformation robustness can be seen in figure 7.

Although transformation robustness is a powerful tool to create feature visualizations, it actually caused many problems when adopted to our DLA CNN model. This is because when applying these techniques, we are slightly changing the size and shape of our input. The Inception model is not affected by this because it is able to account for varying input sizes. In our model, when trying to create any kind of optimization we would receive an input to reshape error from Tensorflow. To get around this we either apply no transformations, or we have to change the input image size to account for the transforms. We experimented with both, and describe each in greater detail later.
4.2. Initial 3D visualizations

Our first attempt in creating visualizations was to optimize an input for a traditional three-dimensional image. Although the model input is a one-dimensional array of flux values, we can create any size optimization as long as it is a multiple of 400. We started with optimizations of the size 116 x 116 x 3, and apply some level of transformation robustness by jittering with size four pixels. In order to prevent the error discussed in the previous section, we pad our image by four pixels as well. This makes the actual input image we are optimizing 120 x 120 x 3, which is a multiple of 400. We optimized each objective for 512 steps, we chose this because optimizing for more steps does not change the image significantly. Also, it is very time consuming to create these objectives, visualizing a single objective with 512 optimization steps takes only 40 seconds on a CPU. We created visualizations for every neuron and channel in the convolutional layers which took a significant amount of time. Figure 8 shows some of the more interesting visualizations from our first attempts. Some of the visualizations were either blank or very blurry. It seems the model had a difficult time optimizing for some of the objectives, and no pattern was ever discovered.

Although these visualizations are interesting and create patterned images, they provide limited insight into this one-dimensional model. The whole point of feature visualization is to see what the model is learning and understand it. These visualizations lack any obvious interpretability. This demonstrates a challenge of creating feature visualizations for inherently one-dimensional models.

4.3. One-dimensional visualizations

To create visualizations that offer greater insight, we optimize a one-dimensional image with the same size as our model input. Essentially we are optimizing a 400 length array, which can be inspected like the input astronomical spectra. We do not apply any transforms, and we only need to optimize for 128 steps (the smaller input image takes much less time to create the visualizations). We begin by creating visualizations for individual neurons (figure 9). These visualizations don’t tell us the full story, so we focus on channel visualizations for the remainder of our work. In the next section we will go more in detail on analyzing the channel visualizations.

5. Interpreting Visualizations

At this stage we have created one-dimensional channel visualizations for all the convolutional layers in our model. Now we wish to understand what the model has learned. Initially, we had few expectations for visualizations that would result beyond the intuition built from the analysis of Inception. In analogy with their work, we thought that the first convolutional layer of the model may prefer basic features of a line. Then, deeper into the second and third convolutional layers, the optimizations might look more like astronomical spectra. And, perhaps, some neurons may show DLA-like curved features. The optimizations do not follow this exactly, but we do observe key distinctions in the optimizations throughout the layers. And, several of the features resemble characteristics of DLAs.

5.1. First convolutional layers

In the first layer of our model, CONV1, all the optimizations follow simple patterns. The 'pixel' values of the optimized image are mostly single-valued either 1 or 0, although note this is not binary. Some of the visualizations are also more noisy and resemble a picket-fence type pattern. Figure 10, shows four handpicked channel optimizations from this layer. These four demonstrate what we see in all optimizations from CONV1.

As expected, the first optimizations have little complexity and do not resemble real astronomical spectra. Because this is the first layer, the model has learned only low-level features. This changes in the next layer (CONV1RELU) where very interesting optimizations appear (Figure 11), including features that represent curves akin to DLA profiles. This is very exciting since we know that a key feature for a human to understand a DLA is the curved shape of the DLA endpoints. We also observe an interesting ‘bump’ pattern in figures 11b and 11c. These patterns are actually quite weird, since no such features like these exist in real spectrum data.

In the first pool layer (POOL1), we see features similar to that of CONV1RELU. Many of the optimizations have the curved shape we observed earlier but with a little more noise or variation (Figure 12). We also see some optimizations that follow similar patterns to CONV1, however the spikes are more spread out. This spacing of these spikes compared to the ones we observe in figures 10b or 10c, more resemble the actual data.
Figure 9: One dimensional neuron visualizations from the DLA CNN model. By default the neuron objective optimizes for the center neuron in each channel. These visualizations were handpicked to show some of the patterns we observed.

Feature visualization for DLA CNN

(a) CONV1 unit 23  (b) CONV1 unit 51  (c) CONV1RELU unit 12  (d) CONV2 unit 61  (e) CONV2RELU unit 20  (f) CONV3 unit 4

Figure 10: Channel visualizations for CONV1.

(a) CONV1, channel 0  (b) CONV1, channel 15  (c) CONV1, channel 19  (d) CONV1, channel 80

Figure 11: Channel visualizations for CONV1RELU.

(a) CONV1RELU, channel 23  (b) CONV1RELU, channel 26  (c) CONV1RELU, channel 53  (d) CONV1RELU, channel 88

5.2. Second convolutional layer - conv2, conv2relu, pool2

In the second convolutional layers, we do not see the dramatic changes observed for the first few convolutional layers. Previously we saw the optimizations go from straight lines with peaks, to curves, to curves with heightened variation or noise. The optimizations in CONV2, CONV2RELU, and POOL2, however all look very similar to each other. In these layers we observe some similar patterns from the previous layers but also a few new ones. Several of the optimizations contain a mix of high and low values that are spaced out and create almost a ‘hills’ and ‘valleys’ kind of image (figures 13b, 13c, 13d). A new feature that comes up in these layers is what appears to be replicas of spectra, an example can be seen in figure 13a. As a matter of fact, the majority of optimizations in these layers have this look.

5.3. Third convolutional layer

In the third convolutional layer, we no longer see any optimizations that look like spectra. In CONV3 the visualizations are mostly very noisy where the images are alternating from high and low values. In CONV3RELU and POOL3, we also see noisy optimizations, however we observe again what seems to be ‘plateaus’ or ‘hills’, where we have consistent parts of high values with no dips. Figure 14 shows handpicked examples of visualizations from these layers. It is important to remember that DLA classification is not the only task of this model. It is also responsible for localizing the DLA and determining the column density measurements. It could be that this layer is more concerned about these two other tasks, and not so much about classifying a
DLA, in which case explains why our visualizations are not helpful.

6. Testing visualizations with the data

In the previous section we examined the pattern of the optimizations from the different layers of the model. We observe the model learning interesting features across the network, e.g. the curve shapes from CONV1, CONV2, and POOL1, to the spectra like optimizations in CONV2, CONV2, and POOL2. These visualizations are interesting, but do not provide a complete story of what our model is learning. To go deeper, we now pass into our model actual spectra and look at the features that are being most positively activated. This sheds some light on the specific features that the model keys on when interpreting real examples. In our analysis we look at 8 different spectra sightlines and explore the features of the model activated when a DLA is or is not present.

To understand our analysis, it is helpful to know how the input data is being reshaped in our DLA CNN model. As an example lets consider the first layer CONV1. The input to the model is 400 x 1, and when the first convolution step is applied (CONV1) the data is reshaped to 134 x 1 x 100. We think of this new shape as the original array (length 400) being resized to a length of 134. The 100 corresponds to the number of channels in CONV1, which we have already created visualizations for, as shown in the previous section.

There are two main ways we go about analyzing the input data at any layer in our model. The first approach is to create a semantic dictionary similar to the one we discussed in section 2.3. Sticking with our example of CONV1, this would allows us to visualize the top four channels being activated at any of the 134 positions of our data. In some cases this proves to be very helpful. However, it is a tedious task to look through many noisy spectra.

Our second approach is to look at the top activated channels over the entire data, and not just one position at a time. This is helpful when trying to determine which features are being activated when a DLA is or is not present. To do this we keep track of how much each channel is being activated at every position, and then examine the ones with the highest overall activation amounts.

In analyzing the spectra, we use both of these methods. In addition to inspecting the features activated, we may analyze the activation values themselves in what we call an ‘activation plot’. For example, when a DLA is present in the data, is the model most highly activated on the edges of the DLA, or within its profile? We keep track of the max activation values in each of the positions in the spectra, i.e. for CONV1 we will have 134 outputs with each amount representing the highest channel activation at a point.

6.1. Examining the first convolutional layer

In CONV1 it is very difficult to to find a pattern between the model input and the features. As a matter of fact almost every model input we analyzed showed channels with the same activation amounts. This is to be expected since it is the first layer, and the features learned are likely to be present in all spectrum. One channel in particular stood out as having a high positive activation with any model input, and a few stood out with having a very high negative acti-
Figure 16: Activation plots for CONV1(b), CONV1RELU(c), and POOL1(d).

Figure 17: CONV2RELU channel visualizations for features responding to DLA’s

viation amount. Figure 15a shows the channel visualization that consistently has a high positive activation and figure 15b shows one of the channel visualizations with the highest negative activation amounts. One interesting thing to note is that these features are pretty much exact opposites of each other.

The CONV1RELU layer is also difficult to find a pattern between features and the input. Also, just like in CONV1 where channel 80 was consistently the highest activated channel, this happens in CONV1RELU. No matter what input, this channel is always highly activated and the visualization looks the exact same. This same pattern occurs in POOL1 as well.

One interesting thing to note about this first layer is that the channel activation amounts seem to be ‘smoothing’ the models input. This is demonstrated by the activation plots in figure 16. Intuitively this makes a lot of sense. The input is just an array of numbers, so where we have high values in the input, the corresponding channel activations are high as well. This can also be one possible explanation for the consistent positive channel in CONV1 and CONV1RELU. In these layers, channel 80 is consistently the highest positively activated channel, and the corresponding optimization is essentially an input with all high values. In CONV1, the highest negative channels are those with optimizations with all low values.

6.2. Examining the second convolutional layer

In the second convolutional layers we begin to see some more concrete patterns matching features to the appearance of DLAs. The first thing to note is that in CONV2, CONV2RELU, and POOL2, many of the features resembled that of spectra. However, when analyzing these features with model input, none of these spectra features were important to the model. As a matter of fact with any model input we analyzed, these specific features never had any significant positive or negative activation amounts. This was surprising because we thought maybe the model had begun to learn and understand spectra, when in reality this was misleading. It seems although these feature resemble spectra, the model is very rarely using them as significant features.

The model begins to get interesting in CONV2RELU and POOL2. There are five channels that are consistently highly activated when the model input contains a DLA, and much less activated when the model does not detect a DLA. These channel visualizations can be seen in figures 17 and 18.

We are also able to combine these visualizations with where they are occurring in the model input using our semantic dictionary approach. We notice that these features are highly activated when at positions ‘inside’ the DLA core. Figure 19 shows an example of our input, a section inside the core of our DLA, and the four highest activated features. Note that all four of these features have been identified by our second approach as only having high positive activations when DLAs are present.

We can wrap up the analysis of CONV2, CONV2RELU, and POOL2 by creating activation plots for each. Recall in the first layers the activation plots showed ‘smoothing’ of the models input. In these layers we see the model has been trained to respond highly to areas inside the DLA curve. When a DLA is present we see a big spike in activation amounts around the center of the input. When there is no DLA the activation amounts are much smaller, and do not have a consistent peak. Figure 20 shows the activation plots of the model responding to an input with a DLA present.

6.3. Examining the third convolutional layer

The third convolutional layer follows a similar pattern to what we see in the second layer. Just like CONV1 and CONV2, it is hard to find a pattern in CONV3. The optimizations are too noisy, which makes it difficult to be able to understand the differences. In CONV3RELU and POOL3 we are able to identify four channels that are consistent with being active
in the presence of a DLA. These four are the channels 28, 43, 58, and 76, which can be seen in figures 21 and 22.

The activation plots of these layers don’t help us find patterns in where these features are most important. We see a mix of high activation levels at the begging, middle, end, or in some cases throughout.

7. Going deeper with channel visualizations

So far we have created visualizations for each channel in our model, and shown which ones have the highest activation’s. However, we can still expand and go further with our analysis. Instead of just looking at the four highest activated channels, we can create a new objective that optimizes over all of them. The result of this is now one visualization that is a combination of all channels. We can piece these new visualizations together and create what's called an activation grid. Taking a look at figure 23, we see this process applied to the image of a dog and a cat in the Inception model. In this figure we have two activation grids from the fourth and fifth layer of Inception. These grids give us an even more detailed look at how Inception understands the input at different layers. In this particular example we see many of the features resembling a dog's snout and cats whiskers, which tells us these are key features for classifying a dog and a cat. We also notice in the fifth layer there are basically no features that would resemble grass. The model is keying in on the important features as we get deeper and deeper.

We take this process and apply it to our model in the same way. For example lets say we are looking at the layer CONV1, which has taken the input data of size 400 x 1, and reshaped it to 134 x 100 x 1. Previously we had 100 channel visualizations for each of these 134 positions, however now we create new objectives that optimize over all the 100 channels. The result is 134 new visualizations, one for each position of the new shape. Since our model and data is one-dimensional, our grids will not be squares but rather long rows. Essentially we are just lining up each of these new visualizations side by side. However when we do this, the grids are very noisy. As we look closer we see the majority of the noise comes on the edges of the visualizations. When lining up the images next to each other these edges make the image harder to interpret. In order to avoid this, we cut off the edges of the visualization to create a more smooth image. We compare these new smooth images to the originals, and the overall shape is still preserved, just with not as much noise.

Figure 18: POOL2 channel visualizations for features responding to DLA’s

Figure 19: Conv2relu highest activated features in the section of the green rectangle.

Figure 20: Activation plots for CONV2(b), CONV2RELU(c), and POOL2(d).

Samuel Gabbard et. al.: Preprint submitted to Elsevier
Figure 21: \texttt{CONV3RELU} channel visualizations for features responding to DLA's.

Figure 22: \texttt{POOL3} channel visualizations for features responding to DLA's.

Figure 24 shows two examples of our activation grids in practice. The left shows how the model is interpreting a 400 length sightline with a DLA present, and the right shows an example with no DLA. We notice that \texttt{CONV1} and \texttt{CONV1RELU} are following a similar pattern we observed with the activation graphs, where the model seems to be smoothing the input. In cases where we have dips in flux values, we see dips in the visualization from \texttt{CONV1} and \texttt{CONV1RELU}. This gives us an idea that in the first layer, the model has been trained to look for ‘dips’ in flux values, which we know to be a useful feature for identifying a DLA. The remaining visualizations from the different layers are a little more difficult to interpret. It’s important to remember these layer visualizations are a group of channel visualizations lined up next to each other. We observed in previous sections the channels tend to get more noisy as we move deeper into the model, so lining up these images next to each other is also going to create a noisy image. We notice in the final \texttt{POOL3} layer, in the case of a DLA being present, the model visualization is much more smooth then when there is not a DLA present. We created activation grids for several different examples, both with and without the occurrence of a DLA. All of our examples followed the patterns we detailed in this section.

8. Spectra Visualization Tool

Another part of our work was to create a simple web application to help with the analysis of the DLA CNN. We found it difficult to analyze the spectra data one sightline at a time. Each sightline contains well over 3000 flux values, and the models sliding window approach can only look at 400 at a time. In order for us to analyze the features and channel visualizations, we needed to create a tool to help us. The tool allows us to upload a python numpy array file that contains flux data, and from there we can pan to the section of interest and visualize the different channels at any layer. Once a layer is selected we can see the average activation plots of each of the channels, as well as the top four at a given point of the data, figure 19 shows an example of this functionality. The tool also shows the activation plots we discussed earlier, an example of these plots are in figure 16. We recommend you checkout this tool to get a better sense of our work and the interesting visualizations of the model, the tool can be found at https://stormy-shore-28190.herokuapp.com/.

9. Discussion and Conclusion

In this project we showed how we can use existing techniques for feature visualization and apply them to a one dimensional model. Initially we ran into problems when trying to get our model to work with this framework, this again was mainly because it was designed to work with models that take in traditional 3D images (height x width x color channel). After some adjustments, we were successful in creating one-dimensional visualizations that we could learn from. These visualizations, although many still difficult to interpret, did give us a much better understanding of what the model learned. In the early layers the model seems to be smoothing the input and looking for dips in flux values, which is a feature of DLA’s. Based on the channel visualizations form the first layer we also see the model has learned features that seem to represent the curved shape of the ‘wings’ of a DLA. This is especially important because these ‘wings’ are a key feature for distinguishing a true DLA from what appears to be a DLA but actually isn’t. The visualizations in the remaining layers become a little more difficult to interpret, but we are still able to notice some patterns. In the second layer we notice the key features for identifying a DLA have high values in the center of the visualizations. We also see this in the activation plots of this layer, which tells us the model in the second layer has now shifted focus from the wings of the DLA to the actual dips in flux values, another characteristic of DLA’s. The third layer is a little more difficult to interpret, however we do see in cases when a DLA
Feature visualization for DLA CNN

(a) Input image to inception model  (b) Activation grid - MIXED4D  (c) Activation grid - MIXED5A

Figure 23: Activation grids for two different layers in the Inception model, with an input image of a dog and a cat.

Figure 24: Activation grids for two example spectrum. Left spectrum contains a DLA, the right does not. The visualizations from top to bottom: Spectra input, input resized, CONV1, CONV1RELU, POOL1, CONV2, CONV2RELU, POOL2, CONV3, CONV3RELU, and POOL3

is present the feature visualizations have consistently high values in the middle of the visualization.

The most interesting thing about this project is that there is no way to measure our success numerically, meaning we cannot say that the model is some percentage more interpretable after creating these visualizations. This project is more about showing how we can use feature visualization to understand complex one-dimensional convolutional networks. An important part of this analysis is to point out that not one technique we used is good enough to paint the full picture. The channel visualizations show us the features the model has learned, but aren’t specific to the input data. The activation plots give us a sense of what parts of the input are most important (highest activation) at different points of the data, but don’t tell us anything about the features. The visualizations of the image at different layers are specific to the input data, but in some layers are too noisy to understand. However, when combining all these different techniques we
are able to better understand what the model has learned and how it is making its classifications.

One other challenge we faced during this project was trying to remove our bias of what we think the model should be learning. When analyzing the spectra data, we often tried looking for patterns that weren’t really there. It was as if we already knew what the model was learning and we were just trying to find examples to support our ideas. It was important for us to take a step back and just analyze what we were seeing, without trying to make it more than what it really was.

10. Future work

There are several ways we can expand on this work. The first way is to take our new understanding of this model and tweak the architecture to perform better. This is one of the main reasons as to why interpretability of neural networks is important. We now understand how the model interprets data, and we can leverage this information to improve the accuracy of the model.

Another way to expand on what we’ve done is apply feature visualization to other one-dimensional models and see what we can find. As mentioned earlier, applying these techniques to one-dimensional data has not been done. We have shown that not only is it possible, but the visualizations are useful enough to learn from. We could apply this work to a time-series model, since time series data is all one-dimensional, and test whether or not we can learn something about time-series features.

References