Visualizing and Understanding Convolutional Networks

Team 40 - Wonkru

1. Aim

Convolutional neural networks are state-of-the art for image related deep learning techniques. However, how and why they work is often taken as a black box.

In this project, we seek to demystify how CNNs perform well by visualizing intermediate feature layers, and what patterns in the training set result in their activation. The analysis of these visualizations helps us to observe the contribution of different layers to the classification and reveal which input features are learnt by which parts of the model.

2. Convolutions and Deconvolutions

2.1 Convolutions (downsampling)

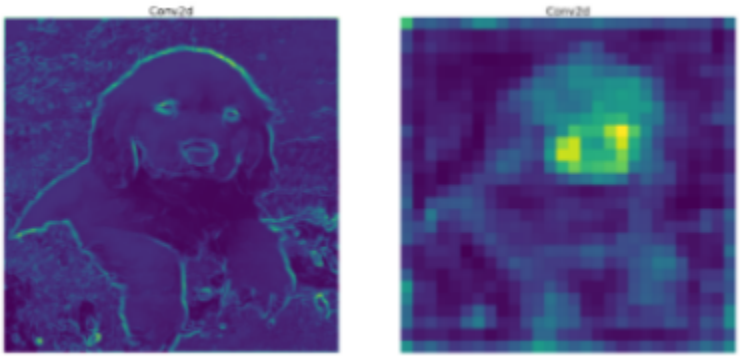
The **convolution operation** involves “convolving” or sliding a filter matrix over the input matrix to generate a new output. Each filter helps to learn “features" of the image. The convolution operation reduces the dimensions of the image (downsamples) as we go deeper into the network and creates an abstract representation of the input image.

2.2 Deconvolutions (upsampling)

The **deconvolution operation** can be thought of as the “backward process” of the convolution operation. The convolution operation isn’t always invertible, so we cannot simply take the inverse of the operation in deconvolution. Instead, it uses techniques such as unpooling and transpose filtering to approximate the process.
The deconvolution operation increases, or up samples, the dimensions of the input.

3. Visualizing using Convolutions

The convolutional neural network creates an abstract representation of the input image. Hence, as we go deeper into the network, we get more pixelated representations of the image. Visualizing these abstract feature maps directly is not beneficial, since it is difficult to analyze abstract representations with no knowledge of what their inputs mean or the features they represent.



For example, above, the left image is a feature map in layer 2. Since it has gone through less convolutions, it is not yet abstract. The image on the right is an output of layer 10, clearly more abstract and pixelated. Hence, we use deconvolutions to visualize the input features which activate a particular feature map.

4. The Need for Deconvolutions

- The deconvolution operation maps the low-resolution feature activities in intermediate layers back to the dimensions of the input pixel space. This is the main technique used for the visualization in the paper.
- By deconvolving the output of the convolution at each layer, we can visualize what the network sees at each layer in the original dimensions of the input image.
- The visualizations reveal what features in the input image activate different feature maps in the intermediate layers.
- They help us analyze what different layers in the network learn, and the evolution of these learned features as we go deeper into the model.

5. Visualizing using Deconvolutions

A deconvnet can be thought of as a convnet model with the same components (filtering and pooling) but in reverse. Each layer in the convnet has a deconvnet layer associated with it. For a particular layer, the deconvnet layer approximately reconstructs the features given as input to the corresponding convnet layer (obtained from the previous convnet layer). By passing the feature map we wish to visualize through the deconvnet, all the way to the input layer, in a backwards fashion, we get a visual representation of what the feature map is learning.

The architecture proposed in the paper has 2 parts:

- The “convnet” which converts the input image into abstract feature maps.
- The “deconvnet” which converts an output back to the dimensions of the original input.

The goal now is to visualize and understand what stimulus activates different feature maps in the model. We wish to analyze what features in the input are learned by which parts of which layers in the convolutional neural network. We use a deconvnet for this purpose as follows:

- Choose an activation or feature map to visualize.
- Set all other activations in the layer to zero.
- Pass the feature maps as input to the attached deconvnet layer.
- The deconvolution involves unpooling, rectification and filtering. This reconstructs the activity in the input which resulted in the chosen activation.
- This process is done in each deconvnet layer until we reach the input space.

6. Techniques used in Deconvolution

1. **Unpooling:** During max-pooling in the convnet, we store the indices of the maximum values in each window (referred to as “switch variables” in the paper). While max-unpooling, we fill these indices with the reconstructions from the previous layer and fill the remaining positions with zeros. This preserves the structure of the original input before pooling. Hence, unpooling gives us an approximate inverse of the pooling operation.

2. **Rectification:** This makes sure our feature reconstructions in each layer are always positive. It is the equivalent of applying the activation function in the convnet.

3. **Transpose Filtering:** To counter the filtering step in the convnet, the deconvnet applies the transposed version of the same filters to the rectified feature maps obtained as input from the previous step.

7. Occlusion Sensitivity

7.1 Concept

Occlusion sensitivity is the concept to understand which parts of the image given as input are important for performing the deep learning task. It is used as follows:

- Here, the task is image classification where this concept helps to classify the images better. Occlusion sensitivity gives us a high-level understanding of what image features a network uses to make a particular classification, and provides insight into the reasons why a network can misclassify an image.
- Different portions of the image are occluded systematically with a grey square and the output of the classifier is observed. The mask moves across the image, and the change in probability score for a given class is measured as a function of mask position.
- The size of the grey square is changed to observe the different heatmaps. This further helps to make a classification decision.

7.2 Heatmaps

Heatmaps are a graphical representation of the data/output denoted by color. It makes the visualization easier. It is used for the occlusion sensitivity understanding because:

- Heatmaps tell us which parts of the image are important for the classification task. So, when the important portion of the image is occluded then the probability score for the predicted class falls sharply.
- Correct mask size and stride parameters are chosen after experimentation. For the higher resolution occlusion results, smaller mask size and stride are chosen.
- Smaller mask size and stride leads to more use of memory and takes longer time to compute.
- A smaller mask size indicates finer details but can lead to more noisy results.

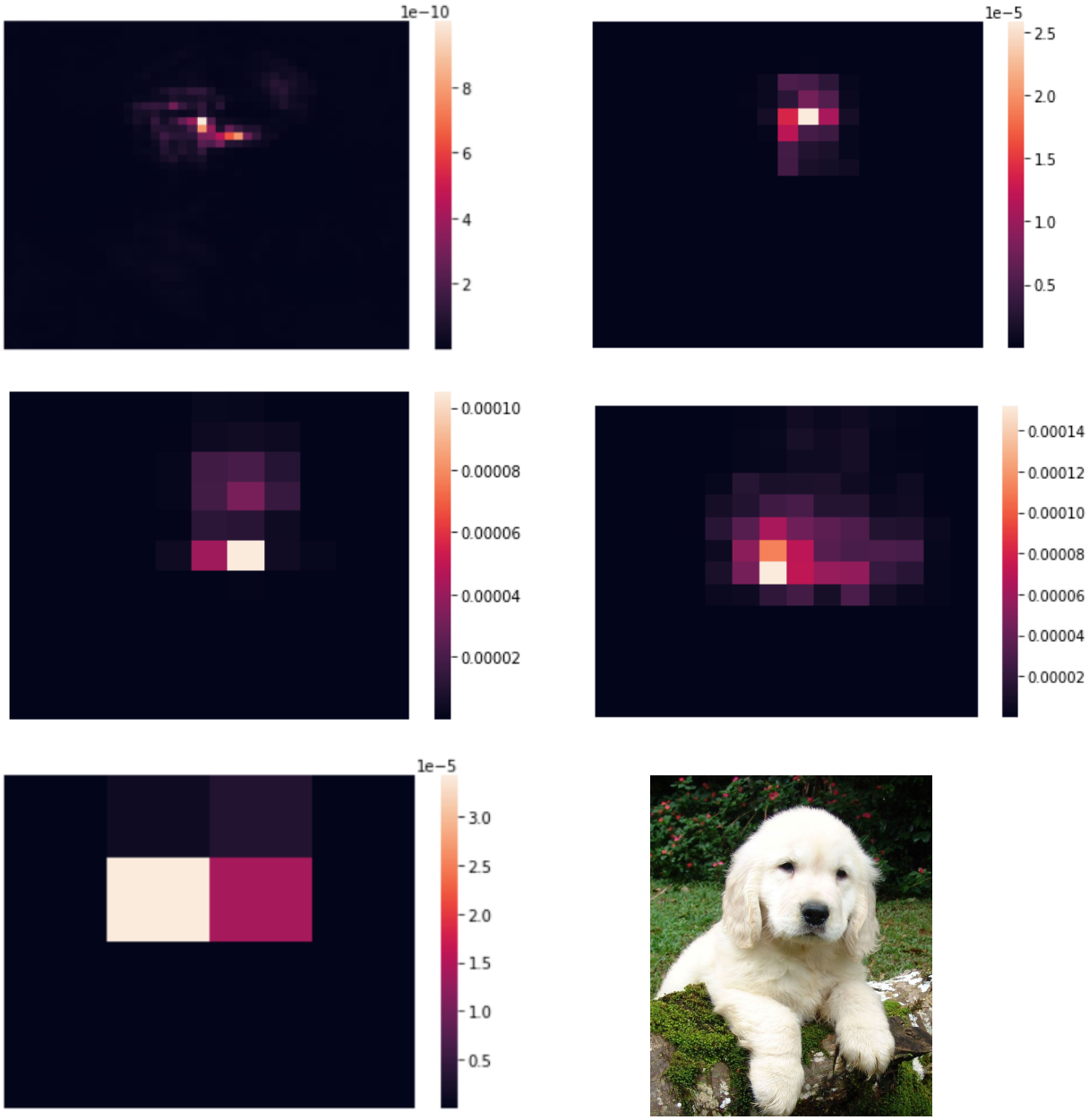
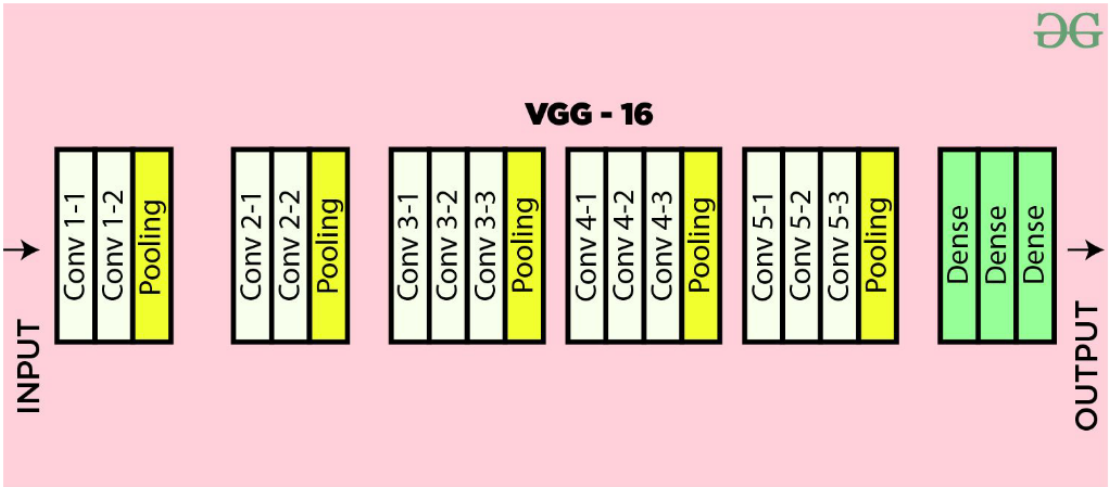


Figure: The heat maps obtained after using (occluding size, occluding stride) are (10, 5); (50, 10); (75, 15); (100, 10); (100, 50); original image respectively (from top left towards right).

Observations and analysis: The above heatmaps show for different values of occlusion size and stride, which parts of the image have a higher contribution to the classification and which parts have a negative contribution. Clearly, the network is able to understand that the dog is present at the center of the image, and surrounding regions have little to no impact on the classification. Our network is robustly able to identify and classify the dog.

7. Implementation Details

7.1 Model architecture



- We have analyzed the intermediate feature maps of the pretrained VGG16 model.
- The model architecture has been implemented from scratch using pytorch.
- This made it easy to implement the architecture of the corresponding deconvnet due to their similarity.

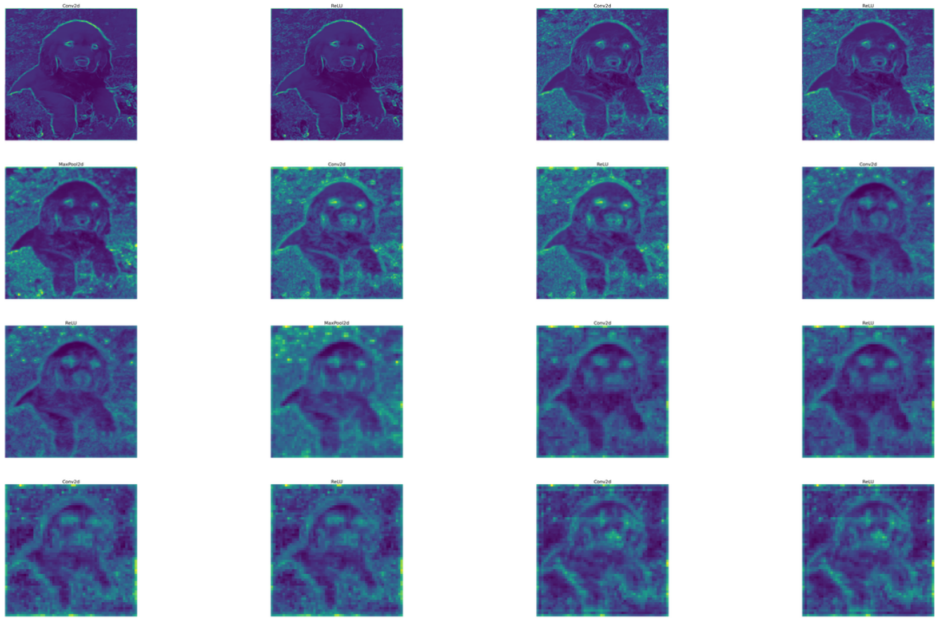
7.2 Code structure

- One notebook for visualizing raw outputs of the VGG16 network without using deconvolutions, including feature maps of intermediate layers and their corresponding weights (submitted during mid eval).
- One notebook implementing VGG16 convnet and deconvnet architecture, and visualizing the deconvolution outputs of chosen feature maps (main portion of this project).
- One notebook implementing occlusion sensitivity analysis on the ImageNet dataset and plotting heatmaps for analysis of the same.

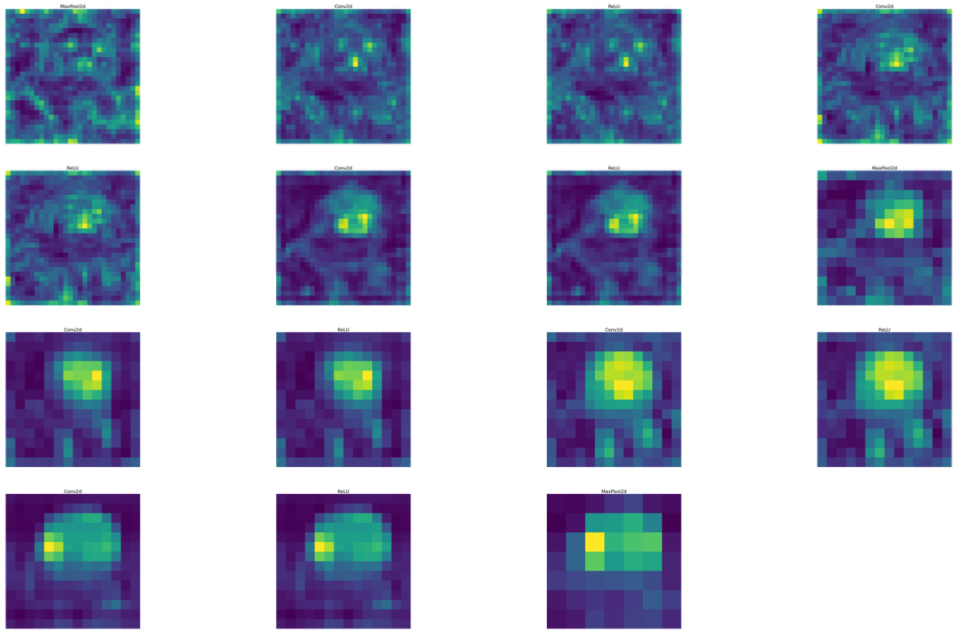
6. Results and Outputs -

We visualize the outputs of the intermediate layers. Here, we assume the convolutional layer, activation function (ReLu) and max pooling to be separate layers which we can observe the output of. This gives us a total of 30 layers in the chosen architecture. The visualizations are done for a single input taken from **ImageNet** shown below.

6.1 Visualizing output of intermediate layers



Left to right: Outputs of layers 0 - 15

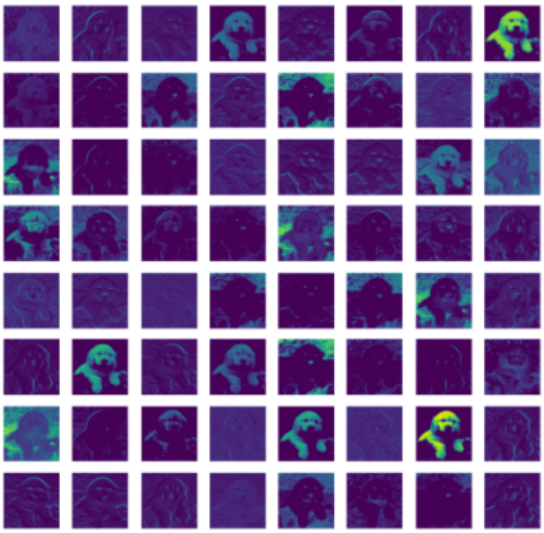


Left to right: Outputs of layers 16 - 30

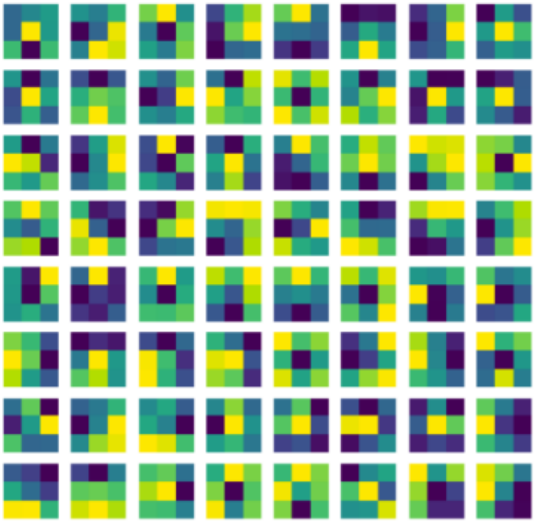
Observations and analysis: As discussed earlier, we observe that as we go deeper into the network, the feature maps become more pixelated and abstract due to the nature of the convolution operation.

6.2 Visualizing all feature maps and weights in a particular layer

6.2.1 First layer



Filter outputs



Filter weights

Observations and analysis: As discussed earlier, we can directly observe and analyze feature maps in the first layer without using deconvolutions. This is because we know the input the layer receives (the original image) and because the outputs of the first layer are not too abstract. We can observe the following:

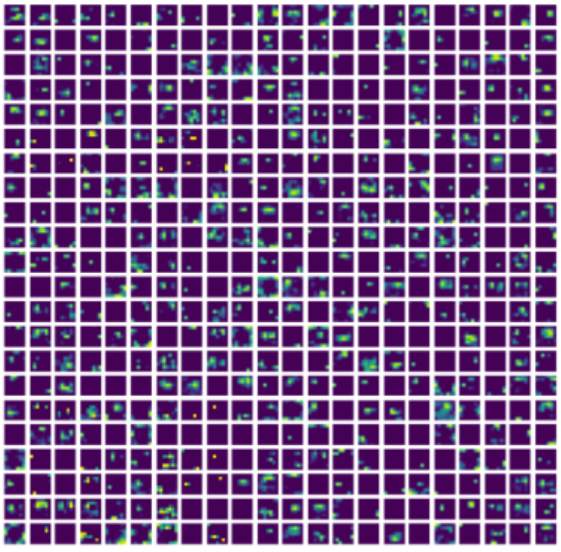
1. The 2nd filter seems to be performing edge detection and learning some edges of the dog.
2. The 7th filter seems to recognise which pixels belong to the dog, possibly by recognizing the color of the dog's fur, or foreground analysis.
3. The 17th filter, similarly, seems to recognize the background, that is, parts of the image which are not part of the dog.

2nd filter

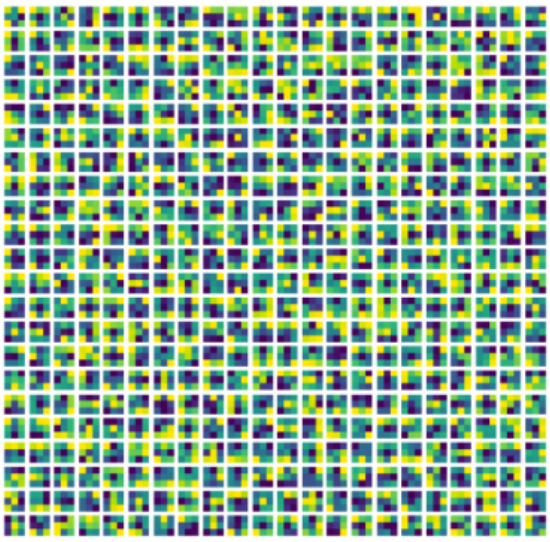
7th filter

17th filter

6.2.2 Last layer



Filter outputs

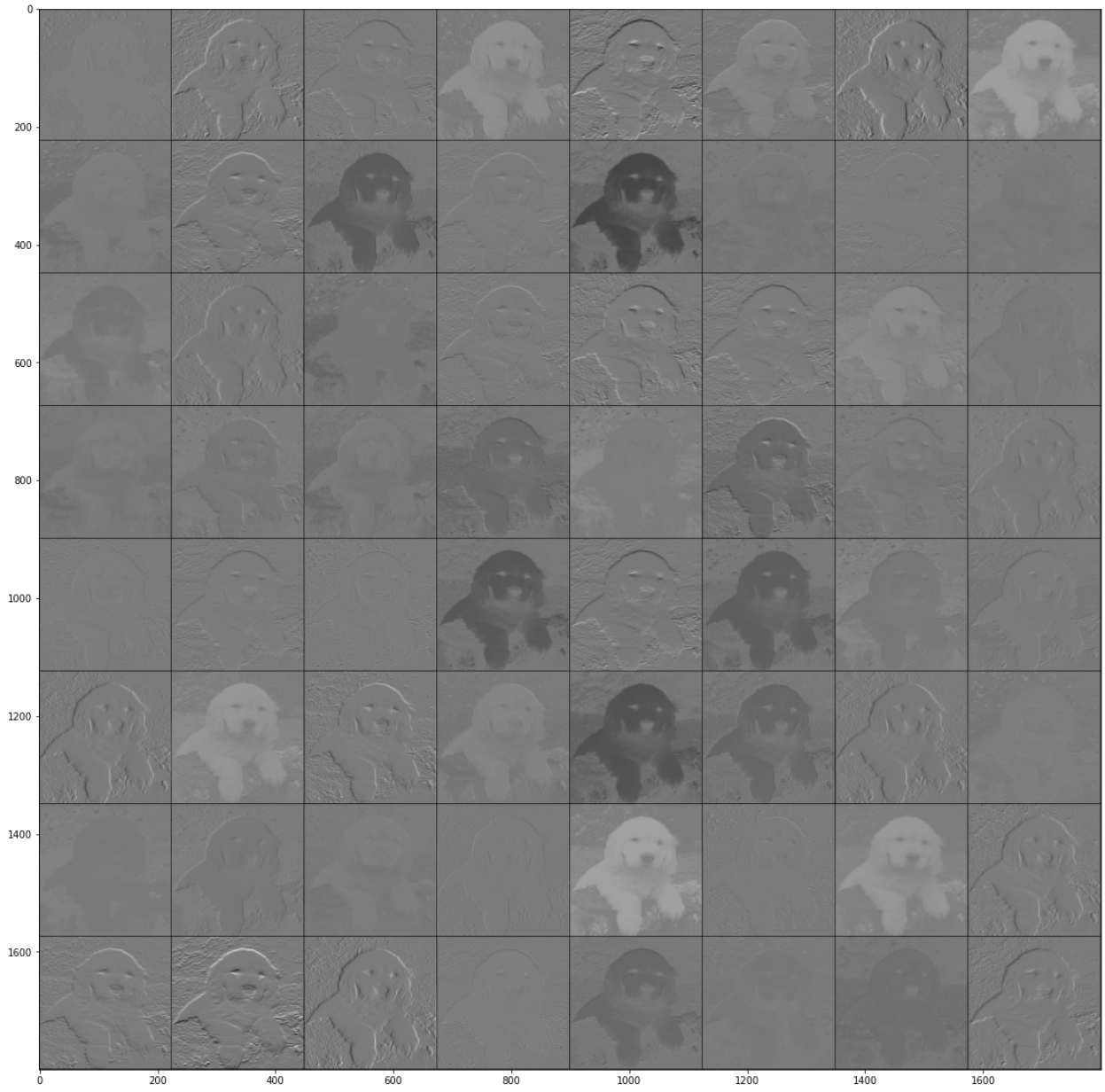


Filter weights

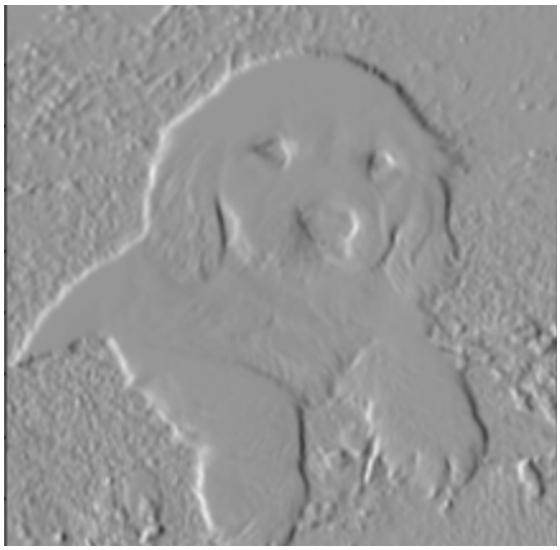
Observations and analysis: In the last layer, due to multiple convolution operations, the feature maps are too abstract to analyze directly. To understand what they are really doing, we use deconvolutions.

6.3 Visualizing feature maps using deconvolutions

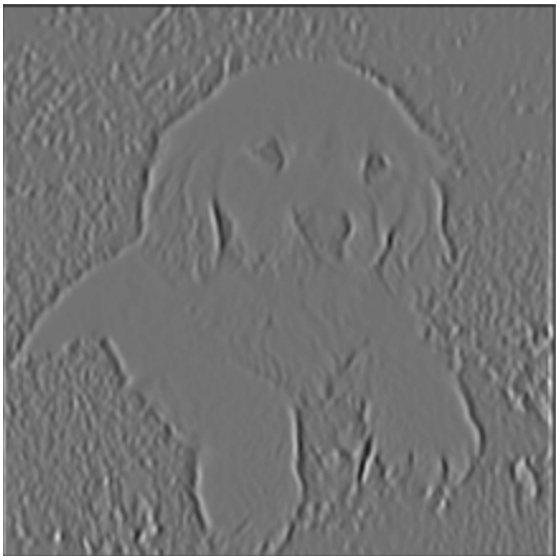
We choose a layer followed by a particular feature map we wish to analyze. Some outputs with their analysis are highlighted below.



Outputs of layer 0 (indexed from left to right in row-major fashion)



Activation Index = 6



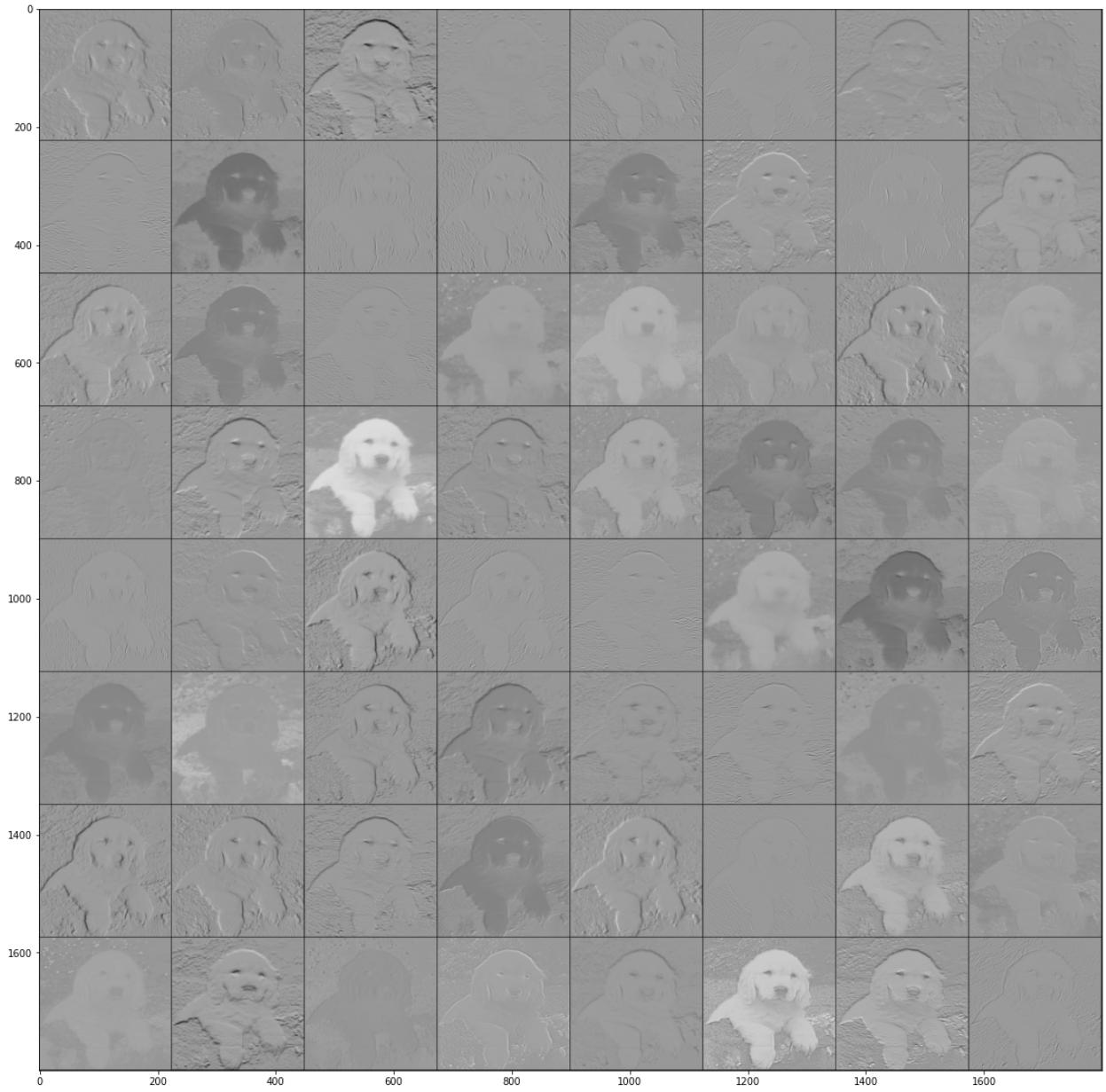
Deconvolution Output

Observations and analysis: The above image represents the 6th feature map in layer 0. By observing the output after deconvolution, the feature map seems to be learning the edges of the dog. Further, since this feature map is in the first layer, we can make a similar observation by looking at the feature map itself.

Activation Index = 10

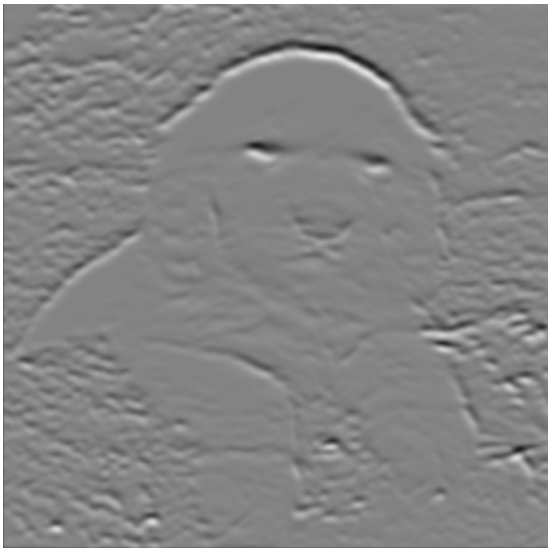
Deconvolution Output

Observations and analysis: The above image represents the 10th feature map in layer 0. By observing the output after deconvolution, the feature map seems to be learning the the extent of the dog's body, possibly by color. It is able to separate the foreground from the background in the image.



Outputs of layer 2 (indexed from left to right in row-major fashion)

Activation Index = 2



Deconvolution Output

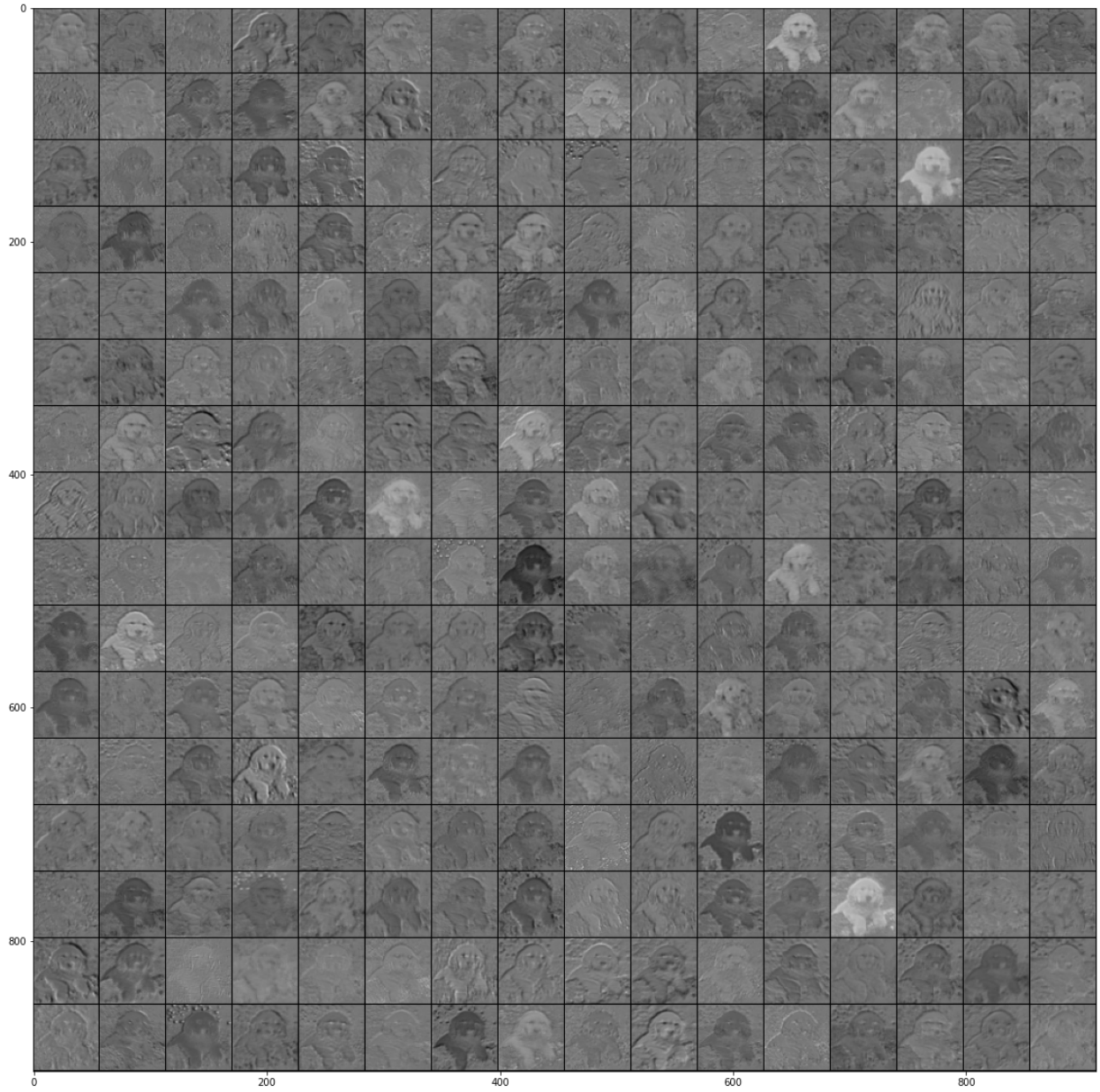
Observations and analysis: The above image represents the 2nd feature map in layer 2. By observing the deconvolution output, the feature map seems to be learning certain specific edges identifying the dog, specifically the curved upper portion of its head, as well as the region around its eyes. Other parts of the dog like its body and paws do not seem to activate this feature layer.

Activation Index = 22

Deconvolution Output

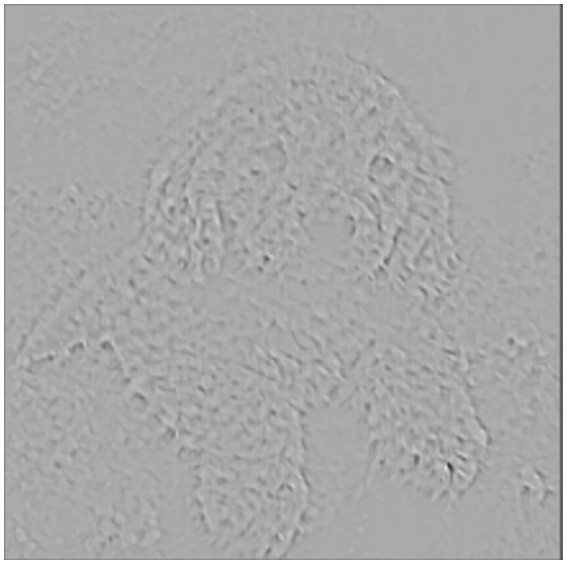
Observations and analysis: The above image represents the 22nd feature map in layer 2. Similar to above, the feature map seems to be learning certain specific edges

identifying the dog, this time activated by edges around its paws and toes, as well as its nose. These features were not well identified by the previous feature map.



Outputs of layer 10 (indexed from left to right in row-major fashion)

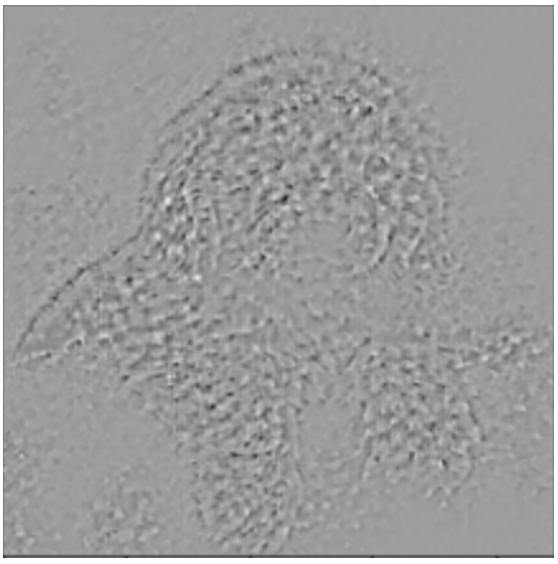
Activation Index = 35



Deconvolution Output

Observations and analysis: The above image represents the 35th feature map in layer 10. We observe that as we move deeper into the network, the feature maps begin to learn more abstract features which are more difficult to analyze. This activation seems to be learning something about the extent of the dog and its shape as a single entity.

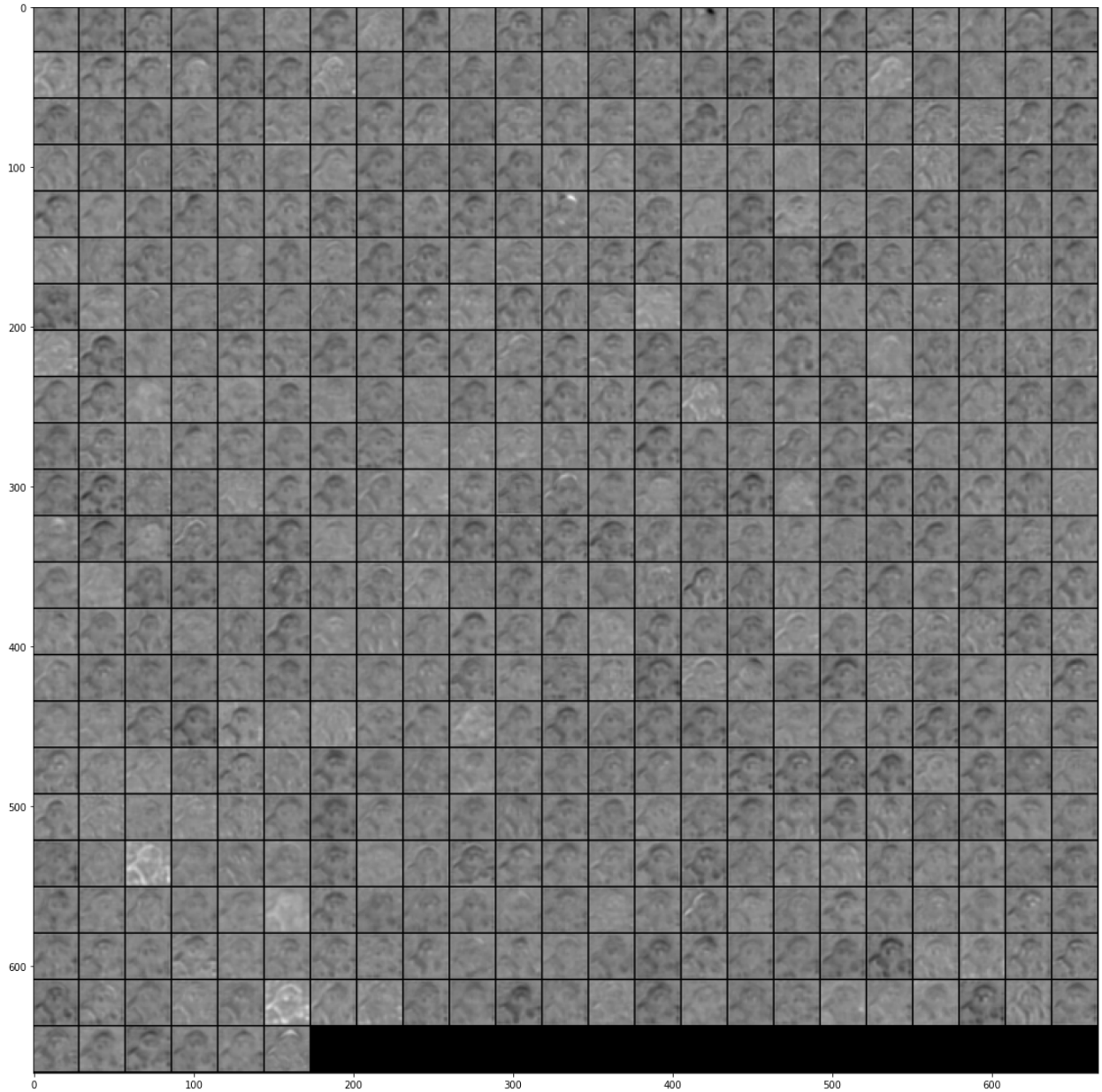
Activation Index = 45



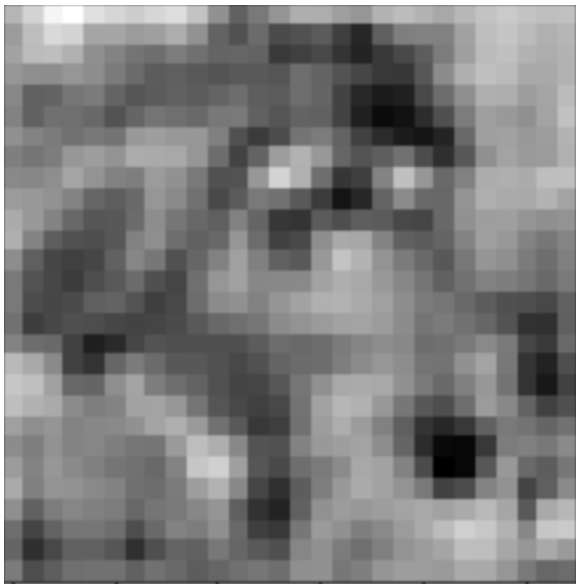
Deconvolution Output

Observations and analysis: The above image represents the 45th feature map in layer 10. Again, it seems to be learning more abstract features about the dog's extent or

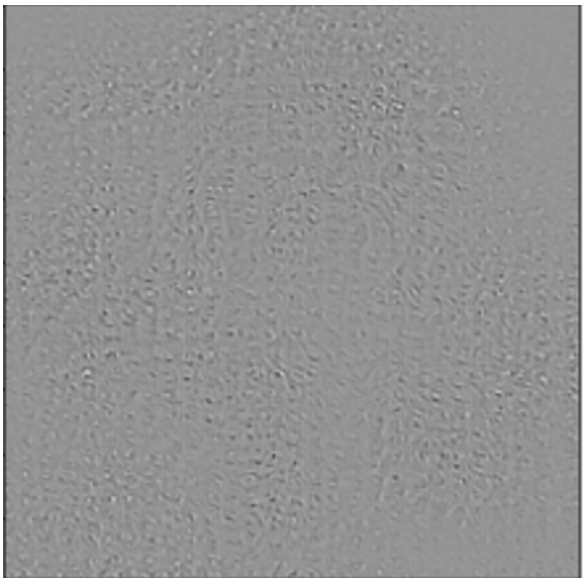
fur, with more concentration on its body and less on its outline as compared to the above image.



Outputs of layer 20 (indexed from left to right in row-major fashion)

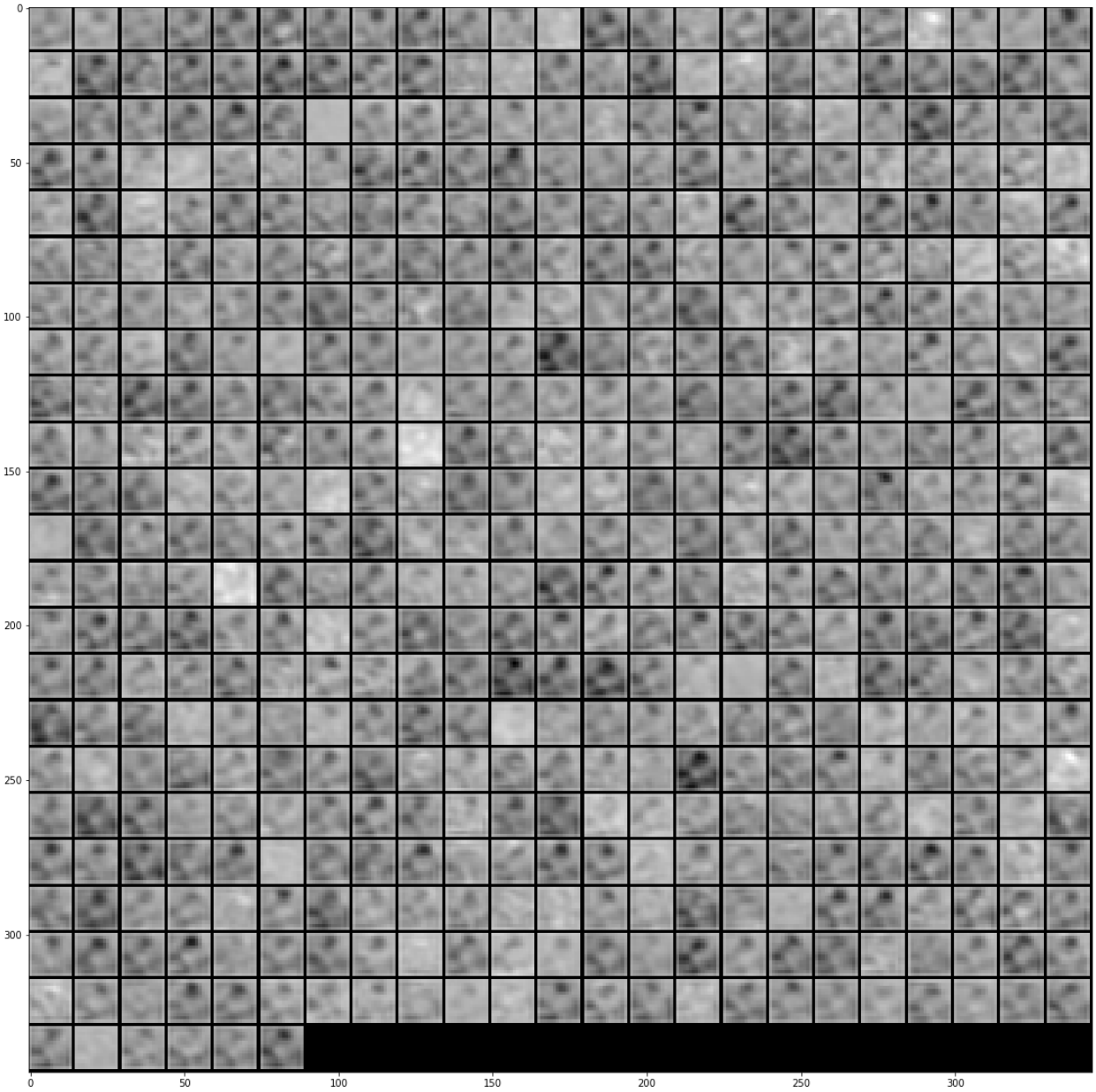


Activation Index = 371

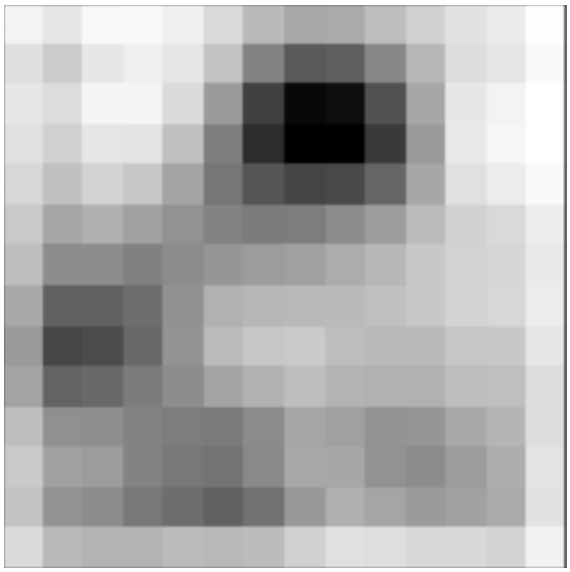


Deconvolution Output

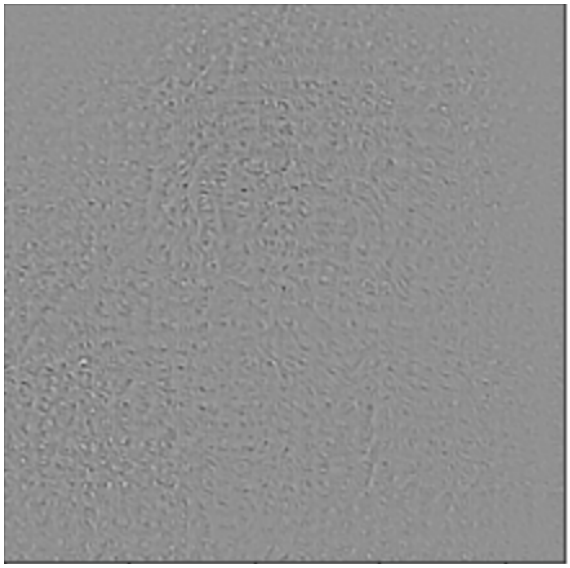
Observations and analysis: The above image represents the 371st feature map in layer 20. The feature map is more abstract and pixelated and begins to look more like noise than a dog in its raw form, due to repeated convolutions. Its deconvolution output too seems too abstract to concretely analyze.



Outputs of layer 28 (indexed from left to right in row-major fashion)



Activation Index = 455



Deconvolution Output

Observations and analysis: The above image represents the 455th feature map in layer 28. The feature map is now too abstract to make sense of, and does not have any resemblance to the input image of a dog. The deconvolution output, once again, seems to be learning abstract features which are difficult to quantify. However, note that on close observation, the deconvolution output does indeed have the same shape and structure of the dog, as in the previous output. We can make out its eyes and nose along with a few edges of its body.

8. Conclusion

Through this paper, we understood and implemented a novel method to observe what different feature maps in a convolutional neural network learn. We demonstrated the downsampling nature of simple convolutions and why they are not sufficient to analyze what intermediate layers of convolutional neural networks really learn. We explores the deconvolution operation as an upsampling technique, used as a backward process of convolutions to transform abstract feature spaces into a representation of the original input which we can analyze. By deconvolving different feature maps in different layers, we gained an intuition of what activated that particular feature. Visualization ultimately revealed that the features are not random, uninterpretable patterns; instead, they show many intuitively desirable properties such as compositionality, edges and class discrimination as we ascend the layers of the deconvnet, which can be used to truly understand what the network is doing rather than taking it as a black box. We also demonstrated through a series of occlusion experiments that the model, while trained for classification, is susceptible to local structure in the image and is not just using broad scene context, indicating its generalizability as well. The project gave us a deep understanding and appreciation of the power of convolutional neural networks.

9. Work Distribution

Note: Everybody contributed in every part, the following distribution only indicates the segments in which the person devoted the most time.

Team member	Task in hand
Harshwardhan	- Constructed VGG16 convnet model using pytorch. - Implemented visualization of intermediate feature maps and weights of the VGG16 model. - Analyzing results of convolutional visualization and explanation of the same in the report. - Analysis and documentation of visualization concepts in the report and slides.
Prayushi	- Constructed VGG16 convnet model using pytorch. - Implementing occlusion sensitivity for ImageNet data. - Tweaking the occluding size and occluding stride to observe the results using heatmap and analyzing results. - Analysis and documentation of visualization concepts in the report and slides.

Ashwin	- Constructed reverse VGG16 deconvnet model using pytorch. - Implemented visualization of intermediate feature maps using the deconvnet. - Analyzing results of deconvnet visualization and explanation of the same in the report. - Analysis and documentation of visualization concepts in the report and slides.
Ayush	- Constructed reverse VGG16 deconvnet model using pytorch - Implemented visualization of intermediate feature maps using the deconvnet. - Analyzing results of deconvnet visualization and explanation of the same in the report. - Analysis and documentation of visualization concepts in the report and slides.