CS 7150: Deep Learning — Summer-Full 2020 — Paul Hand

Week 2 — Preparation Questions For Class
Due: Monday April 11, 2020 at 12:00 PM Eastern time via Gradescope

Name: Saurabh Vaidya
Collaborators: Sumeet Gajjar

You may consult any and all resources in answering the questions. Your goal is to have answers that are ready to be shared with the class (or on a hypothetical job interview) as written.

Directions: Read the articles 'Deep Learning' (Three Giants Paper) and 'Visualizing and Understanding Convolutional Networks'. (ZFNet)

Questions on Three Giants:

Question 1. *Briefly explain the selectivity-invariance dilemma.*

Response:
The selectivity-invariance dilemma basically is that we want our feature extractor to be such that it selects features that are important from the point of view of discrimination but not that are affected by irrelevant aspects. In context of image the irrelevant aspects could be pose, brightness etc. Thus a good feature extractor should be selective about the discriminatory features and invariant to irrelevant features.

Question 2. *Provide a sentence (or two) from the Three Giants Paper that you find interesting and would like to discuss in class.*

Response:
"When trained to predict the next word in a news story, for example, the learned word vectors for Tuesday and Wednesday are very similar, as are the word vectors for Sweden and Norway".

I found this sentence interesting. The fact that we can have networks actually understand semantic and logical relationships between words by implicitly learning them sounds encouraging.

Questions on ZFNet:

Question 3. *Summarize the contributions of the paper. Be as succinct yet comprehensive as possible.*

Response:
This paper presented novel visualization techniques to get insights into functions of intermediate layers of convnets. Activation of each layer's feature maps can be projected back into the image pixel space. It also provides ways to improve architecture of our convnet and also shows the ability of training convnets on ImageNet dataset to generalize well on other datasets when the softmax classifier is retrained.

Question 4. *Precisely explain what is being plotted in Figure 2. Make sure to explain why the organization/presentation of layer 1 is different than layers 2 and higher. You do not need to summarize the visualization algorithm in your explanation.*

Response:
A convnet was trained to solve image classification task. The trained convnet was used to visualize activations of its inner layers by attaching a deconvnet. A deconvnet net was attached to each layer of that trained convnet. Fig 2 shows 9 activations for a particular feature map at each layer and projects those activations in the image pixel space.

A concrete example for different representation of higher layers could be seen in layer 3 row 1 col 1 with the mesh. As compared to layer 1 which learns simple structures, this feature map learned a complex concept such as a mesh. Now different mesh's will have different units activated in that feature map. But all activations represent the same concept(mesh). Thus our feature map activations for 9 inputs might look little different from each other. Whereas in layer 1 row 1 col 1, a feature map only learned the concept of a line. The 9 inputs had the same activation in the feature map. Thus, a single visualization was enough for us to know what that feature map was learning.

Hence for simple concepts such as lines, there is no variation in the activations, but in higher layers, a given feature map has different sections activated for different inputs.

Question 5. *In Figure 2, why even bother showing the visualizations instead of the patches? Provide two examples where the visualization is needed (and the image patches are insufficient) to understand the qualitative interpretation of a particular feature map.*

Response:
The visualizations solely focus on the discriminatory features in an image patch. Patches by themselves might not show any useful or discriminatory information in that image.

For example in layer 5 row 1, col 2 , the image patches shows nothing significant in common but the visualization corresponding to that shows that the feature map was actually learning about the background of the image.

Another example is seen in layer 5 row 3 col 2. Based on the patches, its not clear what is being learned as there are humans, dogs and cars. But looking at the visualizations we can see that the feature map is trying to learn something like a facial structure and often misinterpreting wheel with spokes as a facial structure. Another example is layer 3 row 2 and col 1. There is nothing common in those image patches but after looking at the visualization, its clear that the feature map activates on presence of red color objects.

Question 6. *What do the authors mean by "invariances" of a feature map? Why does the method presented in this paper better show the "invariances" of the feature maps in comparison to gradient- or Hessian-based analysis of the provided neuron?*

Response:
Invariance of feature map means that even if the images are modified through translations, scaling or rotation, the feature map will still give the same prediction(output) as in unmodified image. The activations are not exactly the same, with small differences in values of feature maps,

but the overall output for the feature map remains the same. If a feature map is not invariant, then any modification mentioned above would also change the result (output/activation) of that feature map.

Authors approach is different than Hessian or gradient based approaches in that it provides a non-parametric view of invariances. Unlike gradient and Hessian approaches we do not need to tune the parameters for a unit(neuron) and change the input values to maximize the unit's activation (to find the optimum of that unit). The author's approach projects the activations of the unit in image space for images from the train/validation set that maximize the feature activation. The gradient approach would not give any information about the unit's invariance and the Hessian approach would inaccurately approximate the complex invariances of higher layer due to quadratic approximation.

Question 7. *The authors saw that their classifier's performance was not invariant to image rotation except for the category 'entertainment center.' Explain what is going on with this category. What could you do to improve the rotational invariance of the classifier?*

Response:
With 'Entertainment Center' the classifier was invariant to rotation due to it being rotational symmetric. Thus the object was identical in many rotations to its original form. Looking at the graph c4, the image is predicted positive at angles 0,90,180,270,360, having order of 4 rotational symmetry. Thus at these rotation angles, the rotated image looks similar as original image to the network. Hence, we see same prediction from the network at those rotation angles.

To improve rotational invariance of the classifier we can train the classifier by augmenting each data by rotating it while keeping the same class label.

Question 8. *How useful/general do you find the process the authors used for improving the architecture of a network by visualizing the activations of its inner layers? What are the challenges of this as a general process? What alternative procedures can you envision for improving neural network architectures?*

Response:
The approach used by authors seems certainly useful in improving the architecture of the network by visualization of the inner layers. However, some experience is needed to interpret the results of visualizations and understand where the shortcomings of the existing architecture are. The challenges I can see with this would be that while it may make sense for CNNs It is not intuitive for other networks like RNNs and may also not be intuitive for tasks other than images.

Secondly, the deconvet has to modified for every different CNN architecture for each of its layers. For example for a layer 5 visualization we would have to construct a deconvet with the number of layers as in the Convnet with same number of relu activation and pooling layers. Also, there is additional memory footprint needed in storing the switch settings for pooling.

Another approach that can be used to improve our network architecture would be to use visualization in figuring out which feature maps do not activate much for most of the images for a trained model. Then we can remove that many number of features from a layer and retrain the network. If the network performance does not degrade then we can remove that many number of features permanently. Similarly, if all of the feature maps for a given layer activate for almost all dataset images, then we can add more feature maps in the layers and retrain it. If the network performance does not degrade, we can make that change permanent.

One way to reduce the number of layers could be experimenting with last Convolution layers. For example, if our architecture has 7 conv layers before the fully connected layers, we can remove the 7^{th} layer completely and train our model with 6 layers. If the performance does not change then the removed layer was probably redundant. Similarly we can keep experimenting with some higher conv layers and also with fully connected layers to reach to a pruned architecture while maintaining same performance levels.

Question 9. *Identify which parameters in Figure 3 are user specified, and which are derived from those user-specified parameters. Verify that the derived parameters are correct.*

Response:
The user specified parameters are : Input image size and For all layers 1-5 - filter size, stride length, number of channels and filters, pooling size, pool stride, padding size, number of units in layer 6,7 and softmax units

The derived parameters are: output size of feature map after convolution and pooling.

Output size after applying a filter f with stride s and padding p on a image of size n is given by formula $\frac{n+2p-f}{s}+1$

We can verify these derived parameters. Lets take the convolution output size after layer 1 convolution. Given a filter of size 7x7 and stride of 2 and padding 1 on both sides, the output size should be $\frac{224-7+2}{2}+1$ which is 110.5 which we round off by flooring to 110. And as per the figure that is the correct value.

We can also verify the output size of feature map after pooling. In layer 1, convolution output size is 110, given pool size of 3 and stride of 2, padding of 1 on both sides, output is $\frac{110-3+2}{2}+1 = 55.5 \approx 55$

For layer 2, filter size 5 and stride 2, convolution output size is $\frac{55-5}{2}+1 = 26$. Pooling size 3 and stride 2, output after pooling is $\frac{26-3+2}{2}+1 = 13.5 \approx 13$

Similarly in layer 3,4,5 there is a padding of 1 on both sides and no pooling thus output size is $\frac{13+2*1-3}{1}+1 = 13$. Thus all derived parameter values are verified.

Finally for layer 5 with pooling size 3 and stride 2 and no padding, output size is $\frac{13-3}{2}+1 = 6$

Question 10. *The authors make the claim that their result brings into question the utility of benchmarks with small (i.e. $< 10^4$) training sets. Why do they say this? Do you agree? Are small datasets of no value anymore?*

Response:
They say that because their pre trained models are better feature extractors, being trained on datasets in size of a million. When they train the same feature extractor from scratch on Caltech datasets, their performances degrades significantly too. Caltech and ImageNet are similar datasets with caltech dataset being smaller in size. Thus, when model trained on ImageNet outperforms the same model trained on Caltech, authors question whether the benchmarks that have been achieved on datasets of similar size as Caltech are worth using.

In the context of having the best image classifier- one that solves selectivity-invariance problem, generalizes beyond training data, learns components and hierarchies implicitly, I agree with authors. In order to have such a powerful classifier, we might need a lot of data as opposed to conventional machine learning algorithms. Thus, benchmarks from smaller datasets cannot advocate a model's performance. For use cases where data would inherently be less in size, a *'small dataset trained'* model's performance cannot be a useful indicator of its power and utility.

However I believe that small datasets might be useful. If we have pretrained models, we would not require a huge dataset for a new image classification task. In such cases a small dataset would be enough for retraining the final layers of our network and still build a powerful classifier. This would require less computational resources and would be quicker to train.