Homework 3 – Introduction to Deep Learning (CEE690.06)

Aaron Williams

Problem 1:

a) Derive an expression for the receptive field, in pixels, for one convolutional layer given its filter size and stride.

To complete the given expression, we need four variables:

$r_s = receptive\ field$

$k_j = size\ of\ filter$

$s_l = stride\ of\ layer$

Receptive Field represents the spatial connectivity of two pixels. Given a filter of 3x3, the receptive field is 3x3 as that is the spatial association drawn on to the next layer. For a single layer, the equation is as follows.

$$r_s = k_j$$

For multiple layers, ignoring the stride term we get the following as there will be a single edge overlap when we build our receptive field:

$$r_s = k_{1j} + k_{2j} - 1^n$$

Including the stride term for a 2-layer, we get the following:

$$r_s = k_{1j} + s_{1l} * \left(k_{2j} - 1^n\right)$$

As you can see, the stride of the prior layer is applied to the filter size of the next as the next filter will span the shortened dimensions of the initial image. To make the equation fully duplicatable, we track a term with all previous stride values:

$$s_{all} = s_{all} * s_{prev}$$

$$r_{next} = r_{prev} + s_{all} * (k_{next} - 1^n)$$

b) Extend the previous expression to the case where the convolution is followed by pooling operator given its size and stride.

A pooling operation has the same receptive field equation as the one for a convolutional layer. For that reason, we calculate the receptive field using the same equation as for the 2-layer convolution shown in part a).

$$r_s = k_{conv} + s_{conv} * \left(k_{pool} - 1^n\right)$$

c) Calculate the receptive field of VGG16 right before the first fully connected layer, for a 224x244x3 input image.

The determination of Receptive Field resolves as follows:

Description of Layers	Filter Size	Stride	Total Stride	Receptive Field After
Input Image	1x1	1x1	1x1	1x1
2 x Convolution Layers	3x3	1x1	1x1	5x5
1 x Max Pooling Layer	2x2	2x2	1x1	6x6
2 x Convolution Layers	3x3	1x1	2x2	14x14
1 x Max Pooling Layer	2x2	2x2	2x2	16x16
3 x Convolution Layers	3x3	1x1	4x4	40x40
1 x Max Pooling Layer	2x2	2x2	4x4	44x44
3 x Convolution Layers	3x3	1x1	8x8	92x92
1 x Max Pooling Layer	2x2	2x2	8x8	100x100
3 x Convolution Layers	3x3	1x1	16x16	196x196
1 x Max Pooling Layer	2x2	2x2	16x16	212x212
Fully Connected Layer				

So that the receptive field for VGG16 is 212x212 before it hits the fully connected layer. This spans nearly the entire image.

Problem 4:

a) What is the validation accuracy of the CNN with and without pooling?

The Validation Accuracy of the CNN without pooling was approximately 0.980 for batch size 100 on 50,000 iterations using the Adam Optimizer over a few tests. With pooling the validation accuracy improved to an average of 0.984 with the same settings. In terms of comparison, the loss characteristics show that with pooling converges faster than without. Pooling is also faster in terms of training time making it generally superior to the

b) Did you observe any performance improvements after adding dropout?

I did not observe any performance improvements after adding dropout. In fact, my performance declined back to pre-pooling levels. Additionally, I was forced to use a dropout level of >=0.75 and limit my iterations despite the non-convergent loss as my accuracy kept dropping out for some reason after a certain iteration count. Perhaps that was because I had the dropout plugged into my next fully connected layer? I changed my code to match the in-class code where I didn't see how the layers were connected but I didn't notice any significant different in my results in loss or accuracy curves wit

c) How does the CNN model compare, in terms of performance, to the multi-class logistic regression and multi-class MLP from HW2?

I don't know have the exact speed or performance estimation as I performed the algorithms on two separate devices, but the Validation Accuracy was significantly improved adding the convolutional layers and pooling. It's worth noting that comparing the two models, with and without pooling, I saw improvement to speed when I added pooling.

d) How does the number of trainable parameters in the CNN models compare to that of the multi-class logistic regression and multi-class MLP from HW2?

The number of trainable parameters increase by layer, specifically with regards to convolutional layers where size, depth, and stride are all trainable. Dropout and pooling also have training parameters. Generally, these layers are added with respect to what worked in the past so have fixed settings. To get a truly optimal classifier, however, I imagine that there's a lot of tuning and tweaking with regards to these algorithms.

Problem 5:

a) How many hours did this assignment take you? (There is No correct answer here, this is just an information gathering exercise)

1st Part – 2 hours

2nd Part – 3 hours

3rd Part – 1.5 hours

4th Part – 1 hours

In total, this assignment only took about 7.5 hours.

b) Verify that you adhered to the Duke Community Standard in this assignment (https://studentaffairs.duke.edu/conduct/about-us/duke-community-standard), i.e., write "I adhered to the Duke Community Standard in the completion of this assignment"

I adhered to the Duke Community Standard in the completion of this assignment.