CS5242 Neural Networks and Deep Learning

Semester 1, AY2022-23
Homework 5

Agrawal, Shubhankar
A0248330L

September 2022

1 Question 1

1.1 Transformations Character Learning Task

- In a character recognition task, we do not want to augment the data with transformations that would change the correct class.
- In the case of letters and numbers of English language, transformations of rotation and flipping would not be allowed as they can change the class.
- For example, rotation could change a $6 \rightarrow 9$ and flipping could change $b \rightarrow d$.
- Additionally random cropping is a transformation that would be detrimental to the process as well.
- For example, the number 8 could be randomly cropped vertically to a 3 or horizontally to a 0 which is not ideal.

1.2 Visual Recognition Invariance

- In the case of image recognition there are certain invariances that cannot be generated by augmentation of an image to train the model.
- Given that any object is a 3 dimensional body, the train data might only consider the picture of the object from one side.
- As a result the views other sides would not be captured and augmentations would only be generated using this 2d figure of the image.
- This would not be appropriate as the model would be incapable of predicting images in the test data if they simply contained rotated views of the same object.
- In some cases Generative Adversarial Networks (GANs) are used for data augmentation.
- GANs could potentially change an image from one style to another, perhaps into one that has not been learned for that target class.
- For example, an image of a monument by an artist (to learn the features of the monument) could be completely transformed to another by another artist (in a different style with unrecognizable features) which would not be appropriate in this case.

2 Question 2

We are given an input of shape $h \times w \times c_1$ which we apply on several different kernels with no padding. We also have the assumption that $c_1 \geq hw \geq 3$. We have the general formula for calculating the size of output of convolutional layers using the input and kernel size with the below formulae, which we modify after setting $padding = 0$ and $dilation = 1$

$$L_{out} = \left\lfloor \frac{L_{in} - kernel\_size}{stride} \right\rfloor + 1$$

Conv 1D: For the 1D convolution over an input of shape $h \times w$, the convolution applies along the width. Since the height changes from $h \to 1$, we know that height of the kernel is also h. Additionally, using the formula above, we can get the kernel width = 2. The third dimension of the kernel would be the same as the number of input channels (c_1), and the number of kernels would equal the number of output channels (c_2).

Conv 2D: Using the same formula as above, we know that the 2D dimensions of our output size would be $\lfloor (h-3)/3 \rfloor + 1$ and $\lfloor (w-3)/3 \rfloor + 1$. The third dimension of the output size would be the same as the number of kernels chosen which is c_2

Conv 3D: For a 3D convolution on a 2D image, we apply the formula along all three dimensions to get the final output size as $(\lfloor (h-3) \rfloor + 1) \times (\lfloor (w-3) \rfloor + 1) \times (\lfloor (c_1 - 2) \rfloor + 1)$
The values are populated in Table 1 below.

Convolution Type	Kernel Size	Number of Kernels	Stride	Output Size
1D	$h \times 2 \times c_1$	c_2	1	$1 \times (w-1) \times c_2$
2D	$3 \times 3 \times c_1$	c_2	3	$(\lfloor (h-3)/3 \rfloor + 1) \times (\lfloor (w-3)/3 \rfloor + 1) \times c_2$
3D	$3 \times 3 \times 2$	1	1	$(\lfloor (h-3) \rfloor + 1) \times (\lfloor (w-3) \rfloor + 1) \times (\lfloor (c_1 - 2) \rfloor + 1)$

Table 1: Convolutional Layer Transformations

The final values are:

$$(a) = h \times 2 \times c_1$$
$$(b) = c_2$$
$$(c) = (\lfloor (h-3)/3 \rfloor + 1) \times (\lfloor (w-3)/3 \rfloor + 1) \times c_2$$
$$(d) = (\lfloor (h-3) \rfloor + 1) \times (\lfloor (w-3) \rfloor + 1) \times (\lfloor (c_1 - 2) \rfloor + 1)$$

3 Question 3

The new convolution block and relevant part of the AlexNet architecture are showen in Figures 1 and 2 respectively. Each of the CONV1 and CONV2 blocks are replaces by the new convolutional blocks, which encompasses the original convolutional layer along with two additional 1x1 convolutions.

3.1 Increase in Number of Parameters

The increase in the number of parameters for each of the additional convolutional layer would be equal to the size of the kernel times the number of kernels.

$$\begin{aligned} Convolutional\,Layer\,Parameter\,Size &= Kernel\,Size \times \#In\,Channels \times \#Out\,Channels \\ &= k * k * C_{in} * C_{out} \end{aligned}$$

For each of the convolutional layers, the number of output channels of the new block should be same as that from the original layer. The C_{out} for the first layer needs to match the C_{in} for the first layer since C_{out} represents the number of kernels for the layer which needs to remain constant across the two layers.

Part of AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: **96** 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: **256** 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer

Figure 1: AlexNet Architecture

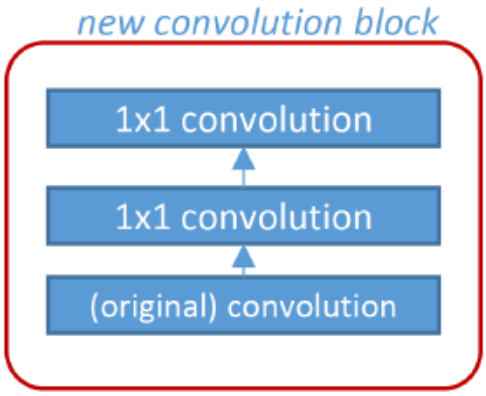

Figure 2: New Convolutional Block

CONV1 For CONV1, we calculate the parameters for the additional two 1x1 convolutional layers as follows:
Additional Layer 1:

$$k = 1$$
$$C_{in} = 96$$
$$C_{out} = 96$$
$$Total\ Parameters := 1 \times 1 \times 96 \times 96 = 9216$$

Additional Layer 2:

$$k = 1$$
$$C_{in} = 96$$
$$C_{out} = 96$$
$$Total\ Parameters := 1 \times 1 \times 96 \times 96 = 9216$$

Increase in parameters for CONV1: $9216 * 2 = 18432$

CONV2 For CONV2, we calculate the parameters for the additional two 1x1 convolutional layers as follows:
Additional Layer 1:

$$k = 1$$
$$C_{in} = 256$$
$$C_{out} = 256$$
$$Total\ Parameters := 1 \times 1 \times 256 \times 256 = 65536$$

Additional Layer 2:

$$k = 1$$
$$C_{in} = 256$$
$$C_{out} = 256$$
$$Total\ Parameters := 1 \times 1 \times 256 \times 256 = 65536$$

Increase in parameters for CONV1: $65536 * 2 = 131072$
Total Increase in parameters: $18432 + 131072 = 149504$

3.2 Increase in Memory

Making calculations using single precision, we know 1 parameter costs 4 bytes or 4B.
Increase in memory of parameters $= 149504 * 4 = 598016B$

Similarly, we would have the same memory for the gradients = $598016B$

Thus, the total memory for these parameters would be = $2 * 598016B = 1196032B \sim 1200KB \sim 1.2MB$

Additionally, we would also see a memory increase for the output values of these new layers that are added. This would be calculated by the size of each of their outputs.

Conv1 Additional Layer 1 Output Size: $55 \times 55 \times 96 = 290400$
Conv1 Additional Layer 2 Output Size: $55 \times 55 \times 96 = 290400$
Conv2 Additional Layer 1 Output Size: $27 \times 27 \times 256 = 186624$
Conv1 Additional Layer 2 Output Size: $27 \times 27 \times 256 = 186624$

Total memory from saving outputs would be = $(2 * 290400 + 2 * 186624) * 4 = 954048 * 4 = 3816192B \sim 2816KB \sim 3.8MB$

Thus, the total increase in memory would be = $1196032B + 3816192B = 5012224B \sim 5012KB \sim 5MB$

4 Question 4

The impact of the following changes on a Neural Net on its bias and variance are summarised in Table 2 below.

Id		Bias	Variance
1	Adding weight decay	Increases	Decreases
2	Increasing the number of hidden units per layer	Decreases	Increases
3	Using dropout during training	Increases	Decreases
4	Adding more training data (drawn from the same distribution as before)	No change	Decreases

Table 2: Convolutional Layer Transformations

1. Adding weight decay leads to weights becoming smaller over iterations. As a result, less of the target value is explained by the input features leading to a **lower variance**. As a consequence, the value of the bias terms would increase leading to a **higher bias**

2. Increasing the number of hidden layers leads to more weights being trained to learn values from the inputs. This leads to more of the target variables being explained by the inputs leading to a **higher variance**. As a result, there is a **lower bias** that is entailed by the model.

3. Using dropout would lead to turning some of the weights to 0 which would could potentially reduce the explanability of the model leading to a **lower variance**. This missing explanability would have to be factored in by the bias accounting for a **higher bias**

4. Adding more training data would make our model less prone to overfitting thereby leading to a **lower variance**. Additionally, since more data only leads to a better fit, this would not increase the bias leading to **no change in bias**.