In the name of Allah, the Beneficent, the Merciful.

Assignment No. 10

Digital Image Processing

Submitted To: Dr. Muhammad Reza Mohammadi

Submitted by: Muhammad Hasnain Khan

Student I'd: 401722316

Q. No. 1: Questions related to the given paper.

a) What is the advantage of using Wavelet in the CNN network and what problem can it solve in the network?

Advantages:

One benefit of using wavelets in CNNs is that they can improve the accuracy of image classification. Additionally, they can make the network more resistant to noise, meaning that small changes in images will have less of an impact on the network's output. This can be useful for applications where the input images may be noisy or of low quality. Additionally, wavelets can be applied to various types of wavelets, such as Haar, Daubechies, and Cohen, making them a flexible tool for improving CNNs.

Problems Wavelets Solve:

Wavelets are used in the given paper to improve the performance of convolutional neural networks (CNNs) on image classification tasks. Specifically, they are used to suppress the effects of noise on the final prediction, making the network more resistant to small changes in the input images. This can be useful for applications where the input images may be noisy or have other imperfections. Additionally, wavelets can be applied to various types of wavelets, making them a flexible tool for improving CNNs.

b) In which part of the network and in what way is Wavelet used? And what were the challenges of implementing it in the network?

The DWT/IDWT wavelets are used in the network. They are used to improve the performance of classification and segmentation tasks. The layer has been tested on the ImageNet dataset and has shown improved accuracy and noise-robustness.

The main challenges in designing DWT and IDWT layers are how data is passed forward and backward through the layers.

c) Briefly explain WaveCNets.

WaveCNets are a type of artificial neural network that uses wavelet transformations in its layers. They are designed to improve the performance of traditional convolutional neural networks by using wavelets to extract more features from the input data, allowing the network to learn more complex patterns and make more accurate predictions. Wavelets are mathematical functions that can be used to analyze and represent signals and data in a more efficient and effective way than other techniques, such as Fourier transforms.

Q. No. 2 (a):

If we convolve a 7x7 image with a 7x7 kernel using a stride of 1, the output will be a 1x1 image with 3 channels. We can also calculate by the following formula:

$$output\ size = \frac{size\ of\ image - kernal\ size}{size\ of\ stride} + 1$$

$$output\ size = \frac{7-7}{1} + 1 = 1$$

Q. No. 2 (b):

If we convolve a 7x7 image with a 3x3 kernel using a stride of 1, the output will be a 5x5 image with 3 channels. We can also calculate by the following formula:

$$output\ size = \frac{size\ of\ image - kernal\ size}{size\ of\ stride} + 1$$

$$output\ size = \frac{7-3}{1} + 1 = 5$$

Q. No. 2 (c):

In general, the 3x3 kernel will result in a larger number of parameters and therefore may be more effective at extracting features from the image. This is because the 3x3 kernel can be "slided" over the image multiple times, allowing it to capture more information about the image compared to a 7x7 kernel, which can only be applied once.

Additionally, using a 3x3 kernel may also allow for the extraction of more non-linear features from the image compared to a 7x7 kernel, which may be more effective for certain tasks. However, these are general statements and the effectiveness of each kernel will ultimately depend on the specific network and its goals.

Q. No. 3:

Image A

12	52	10	54	56
104	235	74	96	72
12	12	52	222	220
201	32	36	35	35
82	35	35	35	35

Image B

12	52	10	54
104	235	74	96
12	12	52	222
201	32	36	35

a) Convolve the following filter with the image A with stride = 1

0	0	2
0	4	1
0	1	10

12	52	10	54	56
104	235	74	96	72
12	12	52	222	220
201	32	36	35	35
82	35	35	35	35

*

0	0	2
0	4	1
0	1	10

$$output\ size = \frac{size\ of\ image - kernal\ size}{size\ of\ stride} + 1$$

$$output\ size = \frac{5-3}{1} + 1 = 3$$

So, the resultant image would be as follows:

1566	2772	2990
640	1008	1637
653	1008	1000

b) Convolve the following filter with the image A with stride = 2

0	0	2
0	4	1
0	1	10

12	52	10	54	56
104	235	74	96	72
12	12	52	222	220
201	32	36	35	35
82	35	35	35	35

*

0	0	2
0	4	1
0	1	10

$$output\ size = \frac{size\ of\ image - kernal\ size}{size\ of\ stride} + 1$$

$$output\ size = \frac{5-3}{2} + 1 = 2$$

So, the resultant image would be as follows:

1566	2990
653	1000

c) Apply maximum pooling on image B with stride = 2

When applying maximum pooling with stride 2 on an image of size 4x4, the resultant image will be of size 2x2. This is because maximum pooling with stride 2 will take the maximum value from each 2x2 block of the input image and create a new 2x2 output image using these maximum values.

So, for the input image B the resultant image after applying maximum pooling with stride = 2, would be as follows:

235	96
201	222

d) Apply average pooling on image B with stride = 2

If we apply average pooling with stride 2 on an image of size 4x4, the resultant image will also be of size 2x2. This is because average pooling with stride 2 will take the average value from each 2x2 block of the input image and create a new 2x2 output image using these average values.

So, for the input image B the resultant image after applying average pooling with stride = 2, would be as follows:

100.75	58.5
64.25	86.2

e) Apply global average pooling on image A and B

Global Average Pooling for Image A:

To apply global average pooling on an image of size 5x5, we first need to flatten the image into a 1D array of size 25. This can be done by concatenating each row of the input image into a single array. So, the 1D array of the Image A would be like this:

[12, 52, 10, 54, 56, 104, 235, 74, 96, 72, 12, 12, 52, 222, 220, 201, 32, 36, 35, 35, 82, 35, 35, 35, 35]

Next, we can take the average of all the values in the 1D array to get the global average value. In this case, the global average value would be:

$$\frac{12+52+10+54+56+104+235+74+96+72+12+12+52+222+220+201+32+36+35+35+82+35+35+35+35}{25}$$

$$= \frac{1844}{25} = \mathbf{73.76}$$

Global Average Pooling for Image B:

To apply global average pooling on an image of size 4x4, we first need to flatten the image into a 1D array of size 16. This can be done by concatenating each row of the input image into a single array. So, the 1D array of the Image B would be like this:

[12, 52, 10, 54, 104, 235, 74, 96, 12, 12, 52, 222, 201, 32, 36, 35]

Next, we can take the average of all the values in the 1D array to get the global average value. In this case, the global average value would be:

$$\frac{12+52+10+54+104+235+74+96+12+12+52+222+201+32+36+35}{16} = \frac{1239}{16}$$

$$= \mathbf{77.43}$$

Q. No. 4:

a) Calculate the input and output dimensions of each layer.

The **first layer** of the Convolutional Neural Network (CNN) is the input layer, which will have dimensions equal to the shape specified in the problem statement:

128 x 128 x 3

The **next layer** is a convolutional layer with 32 filters and a kernel size of (5, 5). The number of dimensions for this layer will depend on the number of filters and the size of the input. Since the input has dimensions 128x128x3 and the kernel size is (5, 5), the output of this layer will have dimensions as follows:

128 – 5 + 1 = 124 x 124 x 32

The **second layer** of the CNN is a convolutional layer with 64 filters and a kernel size of (5, 5). The input to this layer will have dimensions *124x124x32*, and the stride is (2, 2).

The output of this layer will have dimensions as follows:

124 – 5 + 1 = 120 x 120 x 64

However, since the stride is (2, 2), the output dimensions will be reduced by half in each dimension, resulting in an output of dimensions like this:

120 / 2 = 60 x 60 x 64

Thus, the dimensions of the second convolutional layer are *60x60x64.*

The **third layer** of the CNN is an average pooling layer with a pool size of (10, 10). The input to this layer will have dimensions *60x60x64*.

The output of this layer will have dimensions *60/10=6x6x64*. Thus, the dimensions of the third layer are *6x6x64*.

The **fourth layer** of the CNN is a flatten layer. This layer is used to flatten the input tensor into a single vector. The input to this layer will have dimensions *6x6x64.*

After **flattening**, the output will have dimensions *6x6x64=2304*.

Thus, the dimensions of the fourth layer are *2304*.

The **fifth layer** of the CNN is a dense (fully-connected) layer with 32 units. The input to this layer will have dimensions *2304*.

The output of this layer will also have dimensions *32*.

Thus, the dimensions of the fifth layer are 32.

Dense (fully-connected) layers are used in neural networks to connect all of the neurons in one layer to all of the neurons in the next layer.

In this case, the **dense layer** has 32 units, which means that each input neuron is connected to each of the *32* output neurons. This allows the network to learn more complex patterns and make more accurate predictions.

The **last layer** of the CNN is a dense (fully-connected) layer with 10 units. The input to this layer will have dimensions *32*.

The output of this layer will also have dimensions *10*.

Thus, the dimensions of the last layer are 10.

b) Calculate the number of trainable parameters of the network

The number of trainable parameters in the network can be calculated by summing the number of parameters in each layer.

In the first **Conv2D** layer, there are 32 filters with a kernel size of 5x5x3 (5x5 for the spatial dimensions and 3 for the input color channels). So, we can calculate the trainable parameters for this layer with the following formula:

$$parameters = num\ of\ filter * (filter\ size * filter\ size * input\ depth\ of\ previous\ layer + bias)$$

$$parameters_{layer1} = 32 * (5 * 5 * 3 + 1) = 2432$$

In the second **Conv2D** layer, there are 64 filters with a kernel size of 5x5x32 (5x5 for the spatial dimensions and 32 for the number of input filters). So, we can calculate the trainable parameters for this layer with the following formula:

$$parameters = num\ of\ filter * (filter\ size * filter\ size * input\ depth\ of\ previous\ layer + bias)$$

$$parameters_{layer2} = 64 * (5 * 5 * 32 + 1) = 51264$$

We can ignore the **AveragePooling2D** layer because it does not have any trainable parameters.

After the pooling layer, we have a **Flatten** layer which simply flattens the 3D output of the pooling layer into a 1D array. This layer does not have any trainable parameters also.

In the first **Dense** layer, there are 32 units and 2304 input features which we calculated in the part a of the question. So, we can calculate the trainable parameters for this layer with the following formula:

$$parameters = output\ units * (input\ features + bias)$$

$$parameters_{layer5} = 32 * (2304 + 1) = 73760$$

In the second **Dense** layer, there are 10 units and 32 input features, So, we can calculate the trainable parameters for this layer with the following formula:

$$parameters = output\ units * (input\ features + bias)$$

$$parameters_{layer6} = 10 * (32 + 1) = 330$$

For total number of trainable parameters in the network, we can add the trainable parameters of all the layers. So, the total number of trainable parameters in the network will be as follows:

$$parameters_{Total} = 2432 + 51264 + 73760 + 330 = \mathbf{127786}$$