CS 577 - F22 - Assignment 4 Report

Due date: 11/08/2022

Anas Puthawala - A20416308
Professor Gady Agam

① I - 4x4 RGB → 4x4x3

$R = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$ G: $\begin{bmatrix} 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \end{bmatrix}$

filter = $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

B: $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix}$

$\begin{bmatrix} {}^1 2_1 & {}^1 2_1 & {}^1 2_1 & {}^1 2_1 \\ {}^1 2_2 & {}^1 2_2 & {}^1 2_2 & {}^1 2_2 \\ {}^1 2_3 & {}^1 2_3 & {}^1 2_3 & {}^1 2_3 \\ {}^1 2_4 & {}^1 2_4 & {}^1 2_4 & {}^1 2_4 \end{bmatrix}$

shape after: $\frac{4-3}{1} + 1 = 1+1 = 2$ ~~$\frac{W-F}{S}+1$~~ = 2×2

r: $\begin{bmatrix} 9 & 9 \\ 9 & 9 \end{bmatrix}$ G: $\begin{bmatrix} 18 & 18 \\ 18 & 18 \end{bmatrix}$

B: $\begin{bmatrix} 18 & 18 \\ 27 & 27 \end{bmatrix}$

~~Both~~ final conv: $\begin{bmatrix} 45 & 45 \\ 54 & 54 \end{bmatrix}$ #

② Zero padding

$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & {}^1 2_1 & {}^1 2_1 & {}^1 2_1 & {}^1 2_1 & 0 \\ 0 & {}^1 2_2 & {}^1 2_2 & {}^1 2_2 & {}^1 2_2 & 0 \\ 0 & {}^1 2_3 & {}^1 2_3 & {}^1 2_3 & {}^1 2_3 & 0 \\ 0 & {}^1 2_4 & {}^1 2_4 & {}^1 2_4 & {}^1 2_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

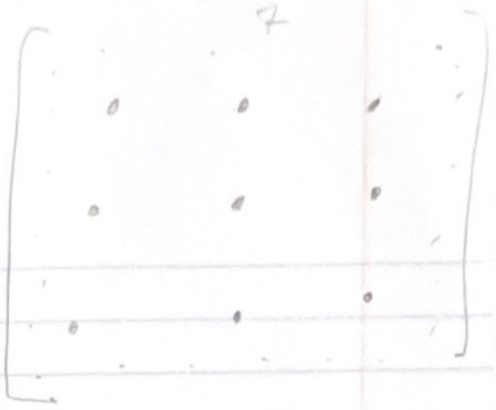
r: $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{*} \begin{bmatrix} 4 & 6 & 6 & 4 \\ 6 & 9 & 9 & 6 \\ 6 & 9 & 9 & 6 \\ 4 & 6 & 6 & 4 \end{bmatrix}$ # ①

G: $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 & 2 & 0 \\ 0 & 2 & 2 & 2 & 2 & 0 \\ 0 & 2 & 2 & 2 & 2 & 0 \\ 0 & 2 & 2 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 8 & 12 & 12 & 8 \\ 12 & 18 & 18 & 12 \\ 12 & 18 & 18 & 12 \\ 8 & 12 & 12 & 8 \end{bmatrix}$ # ②

B: $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 2 & 2 & 2 & 2 & 0 \\ 0 & 3 & 3 & 3 & 3 & 0 \\ 0 & 4 & 4 & 4 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{*} \begin{bmatrix} 6 & 9 & 9 & 6 \\ 12 & 18 & 18 & 12 \\ 18 & 27 & 27 & 18 \\ 14 & 21 & 21 & 14 \end{bmatrix}$ # ③

① + ② + ③ = $\begin{bmatrix} 18 & 27 & 27 & 18 \\ 30 & 45 & 45 & 30 \\ 36 & 54 & 54 & 36 \\ 26 & 39 & 39 & 26 \end{bmatrix}$ # Answer #

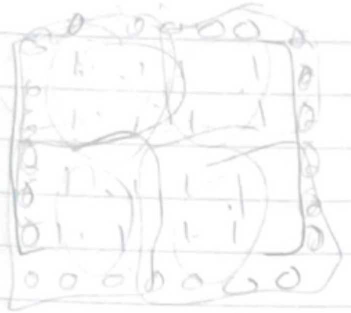
(3) dilation rate = 2. new filter:



3×3 $l=2$

$$(I *_l k)(t) = \sum_{\tau} I(t - l\tau)\, k(\tau)$$

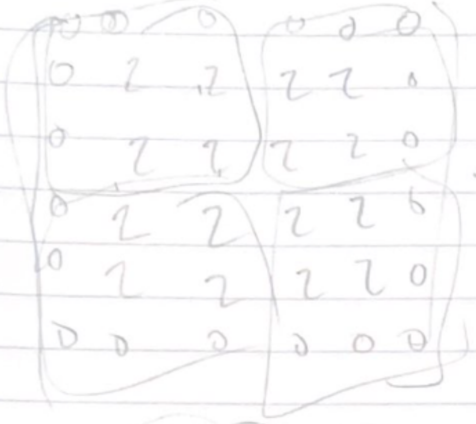
r =



$$\xrightarrow{*} \begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix}$$

+

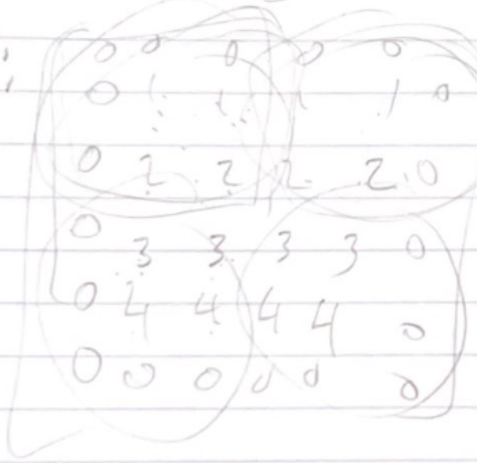
G:



$$\xrightarrow{*} \begin{bmatrix} 8 & 8 \\ 8 & 8 \end{bmatrix}$$

+

B:



$$\xrightarrow{*} \begin{bmatrix} 6 & 6 \\ 14 & 14 \end{bmatrix}$$ #

$$\begin{bmatrix} 18 & 18 \\ 26 & 26 \end{bmatrix}$$ #

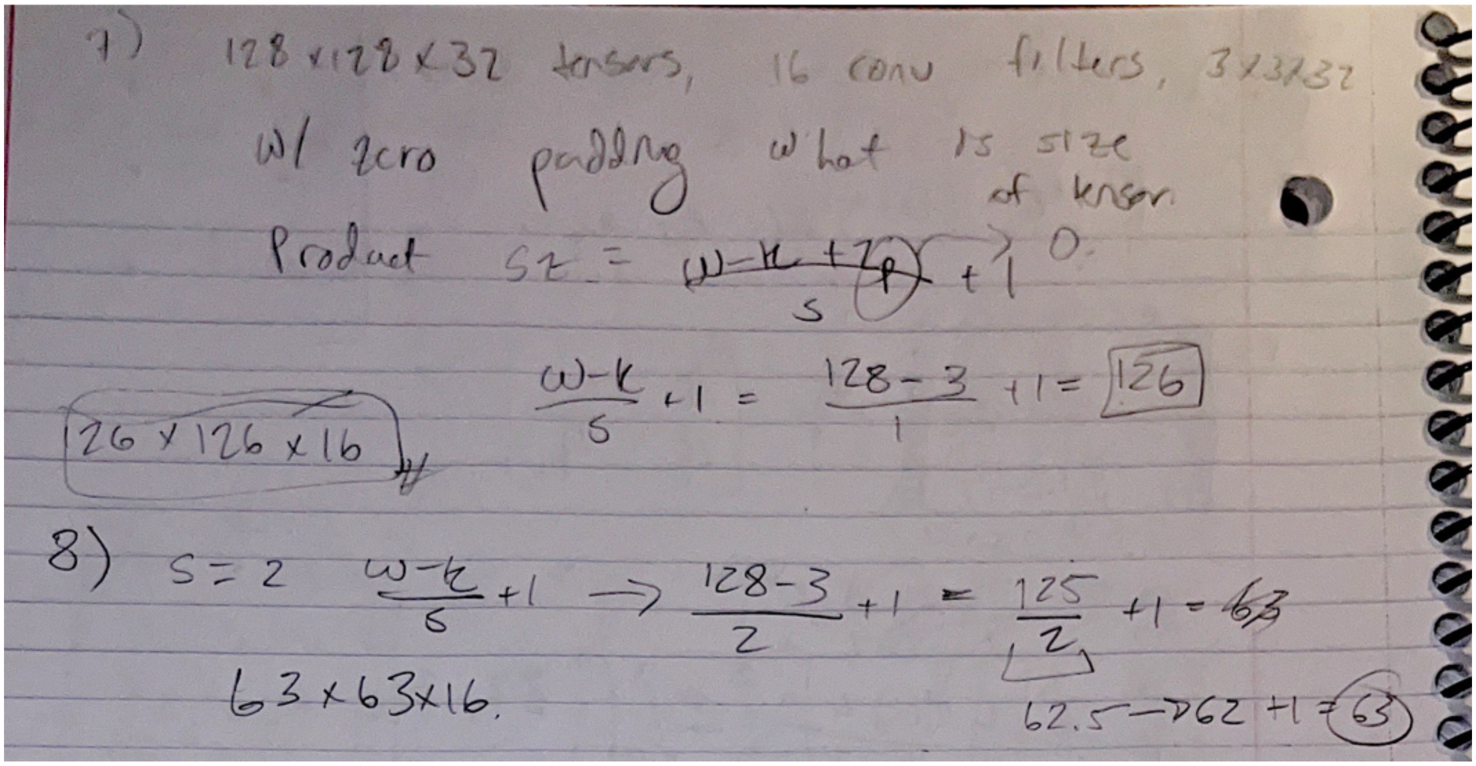
4) The convolution is effectively a weighted sum. When comparing to the template matching interpretation of a a discriminant, it's a weighted sum of the weights and the input features. Similarly in the case for convolution ,it's a weighted sum of the filter and the input feature (the

image). When the filter resembles the image we can expect a higher response. The model is basically trying to find similarity / high matches in the image.

5) When you pool between layers or use a conv. stride like 2, then the image dimensions are adjusted accordingly between each layer and we will end up with a pyramid where the input of one pixel corresponds to many pixels later on in the pyramid. It's basically one pixel contains condensed information from other pixels due to the property of a convolution which is a weighted sum. And it's that condensed information that contains information about the multiple pixels at lower / higher levels in the pyramid. In this way, with a fixed window size and using a pyramid you can analyze feature information at various sampled dimensions.

6) You can increase the number of filters because that'll be tacked on in the channels (depth) dimension. The purpose of this is basically we want to compress the information down into as much as we can learn. There are two benefits to this:

1. It's easier to flatten and feed into a dense network (and we want the most important and condensed features to be fed into the network) and
2. 2. We end up learning more abstract universal information from the feature space that may be important in our task at hand (wether that is classification, object detection, etc.)



7) 128×128×32 tensors, 16 conv filters, 3×3×32 w/ zero padding what is size of tensor product

$$Sz = \frac{w-k+2\boxed{p}}{s} + 1 \quad p \to 0.$$

$$\frac{w-k}{s}+1 = \frac{128-3}{1}+1 = \boxed{126}$$

$$\boxed{126 \times 126 \times 16}$$

8) $s = 2$

$$\frac{w-k}{s}+1 \rightarrow \frac{128-3}{2}+1 = \frac{125}{2}+1 = \cancel{63}$$

$63 \times 63 \times 16.$

$$62.5 \rightarrow 62 + 1 = 63$$

9) A 1x1 convolution is simply a weighted sum of the inputs in a specific index. If you have a singular 1x1 convolution applied to for example an image of 28 x 28 x 64 you'll reduce the dimension to 28 x 28 x 1 as it'll perform the weighted sum over the 64 depth. You can stack the 1x1 convolutions and have for example, 15 or 20 so you end up with an image of 28 x 28 x 15 or 20. So the number of filters is responsible for reducing the number of channels.

10) The purpose of the convolution layers is to extract features or patterns from images and the earlier layers simply extract higher level features / patterns like edges. But in the deeper convolution layers we begin extracting lower level features like shape of someones eye or nose or if there even is a nose or not etc.

11) I. | Max pool stride of 2
2x2 filter

R: $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \xrightarrow{*} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}_{\#}$

G: $\begin{bmatrix} 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \end{bmatrix} \xrightarrow{*} \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}_{\#}$

B: $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix} \xrightarrow{*} \begin{bmatrix} 2 & 2 \\ 4 & 4 \end{bmatrix}_{\#}$

$\rightarrow \begin{bmatrix} 5 & 5 \\ 7 & 7 \end{bmatrix}_{\#}$

12) Pooling layers are used to down-sample the dimensions of the feature set. Thereby reducing the parameters to learn from and also helping to lower computation cost. It also helps to reduce the dimension and get it as small as possible before going ahead and flattening to put it through a FCN.

13) Performing data augmentation helps to increase the generalizability of the model by increasing the data the model has to train with and effectively learn parameters for. It's most useful in cases when the model is overfitting and/or there is not a sufficient amount of data to begin with.

14) Purpose of transfer learning is basically using models people have already spent thousands of dollars and time to develop and train in an attempt to improve performance for our unique special use-case. It's most useful when the target data is somewhat similar to the data that the people trained the initial model on and when we want to get close to achieving state of the art performance without spending as much money in training and time to develop a model.

15) So when we use transfer learning, we are getting an entire model that someone has trained and the model has weights already defined for a lot of the internal layers / parameters. We want to freeze those weights when we fine-tune so we don't destroy the OPTIMAL setting that someone has already acquired and that is unique to that specific model. Because when we start training the gradient back-prop will flow back and it will destroy the weights and parameters UNLESS we freeze layers.

16) You want to start off by
1: adding your customized layer at the end of the model (so this may be like binary classifier, multi-class classifier, etc.).
2: next ensure that the weights are frozen initially in the model to start with and train the classifier head that you just
3: After the classifier head is trained, you want to unfreeze the weights of the network and re-train the entire model

17) The inception block itself has multiple operations conducted on the input feature which then get concatenated together. The idea is that having the different operations can better aid in detecting features at different locations due to the different size variations occurring from the different operations.

18) Residual blocks allow memory to flow more efficiently in deeper networks. It does this by creating an identity mapping to activations earlier in the network as a form of feedback as the network progresses and gets deeper. This serves to circumvent some of the negative outcomes experienced by deep neural networks (such as vanishing gradient or exploding gradient, for example).

19) You want to create another model from the output of a specific layer FROM the existing original model. You need to use the `Model` class instead of `Sequential` b/c it allows for multiple outputs. You need to then get activations for the sample you want to visualize activations for and you can do that by simply calling .predict on the newly created model. Then you can just visualize the activations of the various layers by plotting.
The purpose of doing this is to confirm that our model actually is learning useful features, for example there may be a tumor in the figure and we want it to classify wether there is a tumor or not and ideally we want it to localize on the tumor before classifying if there is a feature or not. We don't want it to localize on random parts and then just classify if there is a feature.
Also another purpose of doing this is to see what the model is actually learning and visualize what gets activated / search for in the deeper layers / shallower layers of the network.

20) You can use gradient ascent to find out what should be the input that will maximize the response of the filter. It's a similar process except it's just the inverse of the gradient descent algorithm, once you have the response of the filter you can visualize it and see what is causing the maximal response. Doing this can ultimately help us learn what the filters are learning and if they're learning important features.

21) Follow the steps to visualize class activations heatmaps:
1. Feed image to network
2. Compute gradients of selected output node w.r.t. each channel of the target layer where activation is to be computed
3. Compute the average gradients of each channel
4. Add the activations of each channel weighted by their average gradient magnitude
5. Superimpose the activations on the input image

When pooling gradients if you use something like max-pool, the higher weight gets given to the channel w/ the higher gradient and the higher gradient ultimately means that the sol'n was more sensitive to that respective channel.
The purpose of this is to visualize the original input image and see what parts of the image contributed to resulting to the solution