Deep Learning 1 - Homework 2

Pedro M.P. Curvo
MSc Artificial Intelligence
University of Amsterdam
`pedro.pombeiro.curvo@student.uva.nl`

Part 1

Question 1.1

a)

The expression for $f_{1,1}$ is given by:

$$f_{1,1} = g_{1,1}h_{0,0} + g_{1,2}h_{0,-} + g_{2,1}h_{-,0} + g_{2,2}h_{-,-} \tag{1}$$

As we can see, we didn't use the full receptive field of the filter, since it goes beyond the image. To address this problem, we can pad the image using several techniques, such as zero-padding, reflection padding, warp padding, etc.

b)

b. i)

Type	**Dataset**	Round Accuracy (%)										**Mean**	**Std. Dev.**
		0	1	2	3	4	5	6	7	8	9		
Valid	Validation	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0000	0.0000
	Test	0.3	0.1	0.0	0.1	0.0	0.0	0.1	0.3	0.0	0.0	0.0900	0.1136
Replicate	Validation	98.6	98.1	98.8	98.5	97.1	98.8	98.2	98.0	98.5	98.5	98.3100	0.4784
	Test	92.9	92.8	88.8	94.7	95.3	91.7	96.3	94.7	95.8	95.0	93.8000	2.1629
Reflect	Validation	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0000	0.0000
	Test	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0000	0.0000
Circular	Validation	99.9	99.9	99.7	99.9	98.9	99.7	99.7	99.5	99.8	99.4	99.6400	0.2939
	Test	81.5	82.7	77.4	86.9	83.2	82.6	86.5	79.8	81.1	84.4	82.6100	2.7504
sconv	Validation	98.8	99.0	98.8	99.1	99.0	99.3	98.7	98.8	98.9	98.5	98.8900	0.2119
	Test	6.5	6.9	6.5	6.1	6.2	6.5	6.2	6.1	6.4	6.3	6.3700	0.2326
fconv	Validation	88.4	87.6	88.6	89.9	88.1	89.9	85.8	87.1	88.2	87.1	88.0700	1.1984
	Test	88.4	88.6	88.6	89.9	89.1	89.9	88.9	88.4	88.9	87.8	88.8500	0.6249

Table 1: Accuracy results for Net1

b. ii)

Looking at the samples from the dataset, the pattern for class label 0 is to have a green on the right and a red on the left. While the pattern for class label 1 is to have a red on the right and a green on the left.

The samples in the train set and the ones in the test set differ in the position of the green and red boxes, but keeping the same pattern for each class label. For example, in the README.md, we can observe that the samples for class label 0 are concentrated in the top ot the image and the samples for class label 1 are concentrated in the bottom and in the test set, the samples for class label 0 are concentrated in the bottom and the samples for class label 1 are concentrated in the top. This means that the network will have to learn the pattern for each class label, regardless of the position of the boxes, i.e., the network will have to learn the pattern of the boxes and not the position of the boxes.

c)

c i)

The variables affecting the `conv_type` are the `padding_size` and the `pad_type`. For *valid*, *sconv* and *fconv* we have the following differences:

- *valid*: No padding is added to the image, since the size of the padding is 0
- *sconv*: The padding added is a zero padding, with size 1
- *fconv*: The padding added is a zero padding, with size 2

c ii)

If we look into the network architecture we can see that we first use a kernel size of 3 with a stride of 1 and then a stride of 2 maintaining the kernel size of 3. With this setup, the convolution operation will reduce the spatial dimensions of the feature maps. By using the *valid*, we are adding no padding to the image, which means that the output size of the convolution operation will be the smallest of the tree. This lead to the reduction of the spatial dimensions of the feature maps, leading to less spatial information being passed to the next layer. *sconv* retains slighty more spatial information, since we are adding a padding of 1 to the image, which will lead to a slight increase in the output size of the convolution operation. This will lead to a slight increase in the spatial information passed to the next layer. *fconv* retains the most spatial information, since we are adding a padding of 2 to the image, which will lead to a significant increase in the output size of the convolution operation. This will lead to a significant increase in the spatial information passed to the next layer.

With that being said, we can extrapolate that larger feature maps will keep more information that allows to distinguish between the classes in the dataset, which leads to the better performance of the model in the accuracy metric as observed. Hence, the order `acc_valid` < `acc_sconv` < `acc_fconv` is expected, corresponding to the amount of spatial information retained by the feature maps.

c iii)

The `reflect` padding mirrors the image along its edges, while `fconv` uses zero padding. Reflect padding can reduce abrupt transitions at the edges by reflecting nearby pixel values, but it can introduce patterns at the boundaries that do not exist in the original image. These patterns can alter the feature extraction near the edges and might reduce the ability for the model to generalize.

In this dataset, where each image has a black background with one green box and one red box, zero padding performs better as it adds zero-value borders that match the original black background, allowing the network to focus on the actual box patterns. Reflect padding, applied at all layers, can create artificial features at the edges, such as duplicating and inverting the boxes, which may harm the model's ability to learn the actual patterns, since it can mix the patterns of the boxes.

c iv)

Zero padding creates a sudden change in pixel values at the image boundaries, resulting in high derivative values across the receptive fields near the edges. This can cause the CNN to confuse these abrupt changes with actual edges, particularly the edges of the green and red boxes in the dataset that we are trying to predict. The network may then interpret the zero padding as an actual feature, leading to worse performance in edge detection, hence worse performance in identifying the boxes.

Replicate padding, on the other hand, extends the edge pixels into the padded area, ensuring a smoother transition, since the pixel will have the same value as the nearest edge pixel. This helps preserve the local context near the edges by maintaining some statistical properties of the image, such as the derivatives. This is beneficial when detecting the edges of the boxes, since replicate padding prevents the network from confusing the padded area with the actual edges. As a result, the replicate padding improves the model's performance in detecting the true edges of the boxes, leading to a better accuracy.

d)

d i)

Type	**Dataset**	Round Accuracy (%)										**Mean**	**Std. Dev.**
		0	1	2	3	4	5	6	7	8	9		
Valid	Validation	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0000	0.0000
	Test	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0000	0.0000
Replicate	Validation	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0000	0.0000
	Test	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0000	0.0000
Reflect	Validation	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0000	0.0000
	Test	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0000	0.0000
Circular	Validation	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0000	0.0000
	Test	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0000	0.0000
sconv	Validation	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0000	0.0000
	Test	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0000	0.0000
fconv	Validation	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0000	0.0000
	Test	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0000	0.0000

Table 2: Accuracy results for Net2

d ii)

As we can observe by the results, the model is able to achieve 100% accuracy on the validation set for all padding types, but it fails to generalize to the test set, achieving 0% accuracy, performing worse in every padding type in `Net2` with lower accuracies than `Net1`.

d iii)

In the `Net2`, we added an additional linear layer and we remove the AdaptiveMaxPool2d layer. The additional linear layer increases exponentially the number of parameters in the model, which can lead to overfitting. If we look into the `utils.py` file, we can see the the test dataset is shifted by 16 pixels relative to the validation and training dataset. This can mean that the model `Net2` is overfitting to the training and validation dataset by memorizing the overall positions of the boxes and, since they are shifted in the test dataset, the model is unable to generalize to the test dataset, leading to the 0% accuracy. This makes sense if we also consider that the model is achieving 100% accuracy in the validation set, while getting 0% accuracy in the test set in all padding types.

Question 1.2

a)

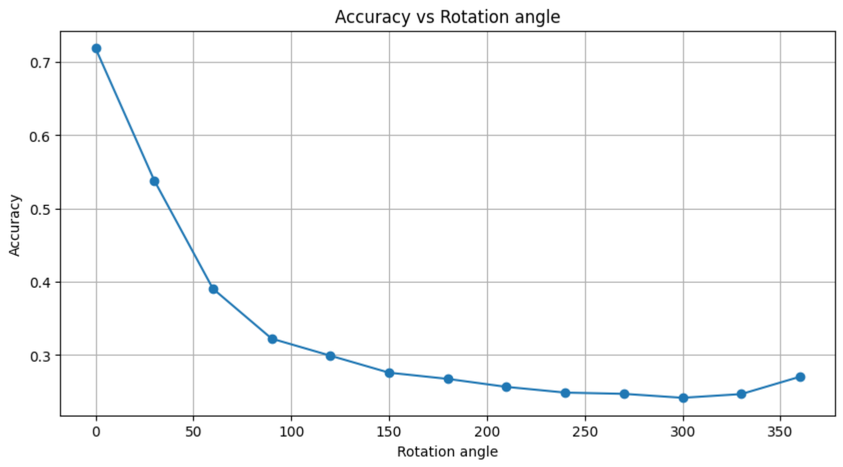

Figure 1: Inference respect to angle of rotation

CNNs are not naturally rotationally invariant because the filters they learn during training are sensitive to the orientation of features in the images. When an image is rotated, the learned filters may no longer align with the features in the new orientation. This misalignment reduces the model's ability to recognize patterns, leading to a drop in accuracy. The convolutional layers detect local patterns such as edges and textures in a fixed direction, and rotating the image changes the spatial relationships between these features, making it harder for the network to match them correctly.

If the model is not trained on rotated images, it cannot learn to recognize objects or patterns from different angles. The performance of the model is highly dependent on the orientation of the test images, and it is likely to perform worse when presented with images that were not seen during training. Even though the model might perform better at certain rotation angles (e.g., 90° or 180°), accuracy typically drops as the angle deviates from those orientations. This demonstrates that CNNs require rotation-invariant training data or additional techniques like data augmentation to improve their ability to recognize rotated images.

b)

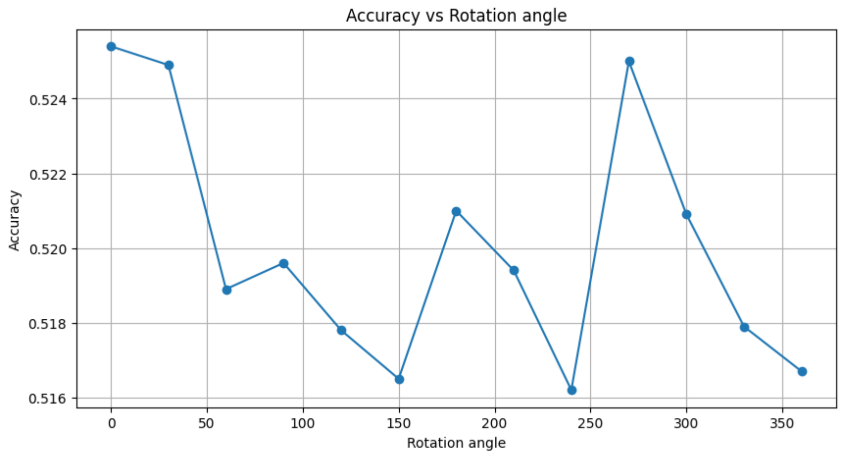

Figure 2: Inference respect to angle of rotation with data augmentation

When the model is trained on an augmented dataset with random rotations, it is exposed to a wide variety of orientations for the objects in the images. This allows the model to learn features that are not dependent on a specific orientation. By incorporating rotated versions of the images during training, the model learns to recognize patterns and objects regardless of their angle. This helps the network develop rotational invariance, as the filters it learns become capable of detecting the same features from different perspectives.

As a result, the model becomes more robust to rotation, and its accuracy stabilizes across different angles during inference. With this training approach, the network does not rely on a fixed alignment of the features, and can instead generalize across a range of orientations. This improved performance across various rotation angles indicates that the network has learned to identify the objects or patterns in the images without being influenced by their orientation, thereby overcoming the limitations of rotational sensitivity found in networks trained only on images with fixed orientations.

Part 2