Anna Dollbo 1843672
Flavius Marc 0862037

INFOMVC 2022
Group 25

Report Assignment 4: Convolutional Neural Networks

1. Model descriptions

1.1. Baseline Model

The baseline (hereafter BL) model was inspired by the original LeNet-5 architecture (proposed by Yann LeCun et al. in 1989), with two convolutional layers followed by pooling layers and two fully connected layers before the output layer. Since the requirement is to get hands-on experience with training and validating Convolutional Neural Network (CNN) models, we decided to choose LeNet-5, which is a kind of feed-forward neural network whose artificial neurons can respond to a part of the surrounding cells in the coverage range and perform well in large-scale image processing.

1. Input. MNIST images are 28x28 which is smaller than what LeNet-5 expects. Therefore, we performed a preprocessing step by padding the images with zeros to bring the MNIST images' size to 32x32.
2. In the first convolutional layer, the convolution operation is made with the filter size of 5x5 and with 6 such filters, stride = 1, using rectified linear unit (ReLU) activation function (*The function returns 0 for any negative input, but for a positive value x, it returns that value back*). As a result, we get a feature map of 28x28x6, where the number of channels is equal to the number of filters applied. L2 Regularizer added.
3. Next, subsampling is made with an average pooling layer that has a filter size of 2x2 and stride = 1. Since the pooling layer doesn't affect the number of channels, the resulting feature map is 14x14x6.
4. After this comes the second convolution layer with 16 filters of 5x5 and stride 1. ReLU activation is used. L2 Regularizer added.
5. For the second pooling layer a filter size of 2x2 with stride 2 is applied which results in a reduced feature map to 5x5x16. Afterwards, we flatten the output shape of the final pooling layer to a single vector.
6. The final pooling layer has 120 filters of 5x5 with stride 1. ReLU activation function used.
7. The next is a fully connected layer with 84 neurons that result in the output of 84 values. ReLU activation function used.
8. The last layer is the output layer with 10 neurons and Softmax function. The Softmax gives the probability that a data point belongs to a particular class. The class with the highest probability is then predicted as the true class.
9. Output 10 values.

The loss function for the model was categorical cross-entropy, since it is used in multi-class classification tasks. The model was additionally installed with L2 regularisation, which works on the cost function by minimising the squared magnitude of the weights. L2 regularisation penalises representations that are too complex and is preferable to L1 regularisation in computer vision tasks. Since overfitting is a common problem for neural networks, a choice was made to include regularisation in the baseline, rather than test it as a separate model. `Adam` was used as an optimiser since it tends to be the best of the adaptive optimisers and converges faster than many other optimisers, which is an advantage for a task that has only 15 epochs.

1.2. MaxPooling Model

Since the BL model made use of average-pooling, which smoothes the image, we wanted to see the effect of using max pooling instead. `MaxPooling2D()` gives a more sharp image focused on the maximum value, and hence it retains the most prominent features of the feature map. The usefulness of using max pooling instead of average pooling depends on the data, and we were interested to see which of the two pooling-techniques benefitted the dataset.

1.3. Filter model

The filter model is the name given to a model that has more filters than the BL model. The first convolutional layer has 32 filters and the second convolutional layer has 64 (as compared to 6 and 16 in the BL model). The number of filters was chosen after validation, and the main goal was to see if more relevant information extracted from noisy raw pixel data could increase model performance. More filters have the ability to encode more specific features of the image and this would make the CNN able to extract more complex features from the image data.

1.4. Convolution model

Convolution model is the name of a model that has one extra convolutional layer and pooling layer than the BL model. More convolutional layers are expected to extract more detailed features of the image, which might lead to better accuracy for some image classification tasks. We wanted to see if this dataset would benefit from this.

1.5. Learning rate model

The learning rate model is the name of the model that differs in how it treats learning rate as compared to the BL model. Here we opted to complete one of the choice tasks, by having the learning rate decrease by half every five epochs. To manage this, `LearningRateScheduler()` was called after every epoch, which checked if the epoch fulfilled a certain criteria, and then adjusted the learning rate accordingly. In the BL model, the learning rate is managed by the Adam optimiser, which has an individual learning rate for each parameter in the network. This means that the Adam optimiser is not compatible with the `LearningRateScheduler()` function, and the optimiser for this model was therefore chosen to be Stochastic Gradient Descent (SGD), which is one of the most common optimisation algorithms in machine learning. We tested the model both with initial learning rate 0.1 and 0.01, as they are both standard starting points for machine learning models, in order to evaluate which one is better for this model that only runs for 15 epochs.

1.6. Dropout model

Even though the BL model already has L2 regularisation, we were interested in seeing the effect that dropout would have on the model performance, which is why we made a sixth model just to get a better understanding of this regularisation technique. Dropout changes the input to the next layer by setting some of the input units to 0, which means that different neural networks are trained at each step. This helps prevent overfitting since no network during training will be the same. A dropout layer was instantiated after each of the two pooling layers of the BL model, as well as before the final output layer. The dropout layers after the convolutional layers were set to drop 20% of the input units per step of the training, and the last dropout layer was set to drop 10% of the units. These numbers were chosen in accordance with the literature (Park & Kwak, 2016).

2. Model performances

2.1. Baseline Model

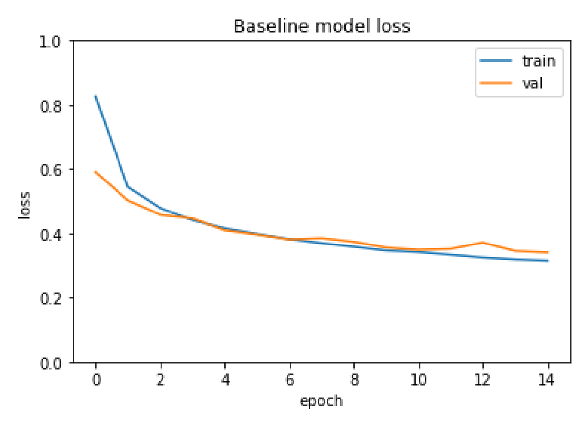
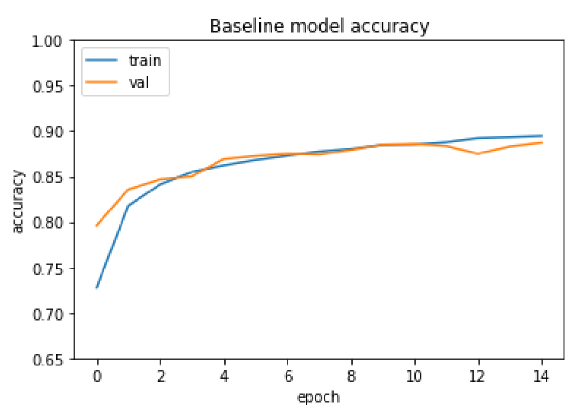
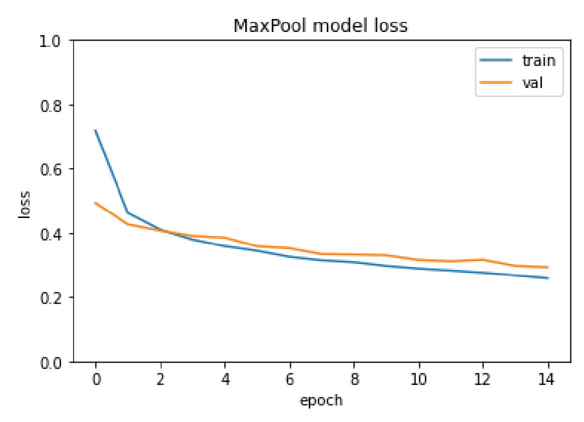
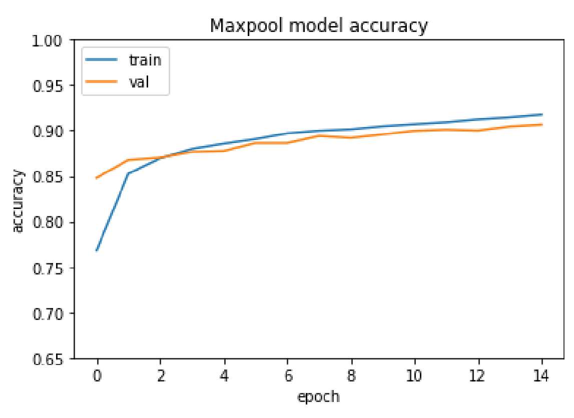
	
Training Loss: 0.3039	**Training Accuracy: 0.8974**
Validation Loss: 0.3212	**Validation Accuracy: 0.8936**

Table 1. Baseline Model loss/accuracy

The BL model performs quite well with a training and validation accuracy of 89%. However the loss is still quite high, meaning that there are probably many more tweaks to the model that can be done to get a good performance. As observed in the plot, the validation loss gets higher around epoch 12, while the training loss continues to lower. This might indicate that the model has begun to overfit.

2.2. MaxPooling Model

	
Training Loss: 0.2787	**Training Accuracy: 0.9108**

Validation Loss: 0.3103	**Validation Accuracy: 0.8987**

Table 2. MaxPooling Model loss/accuracy

The complexity stays the same as the BL model (number of parameters have not changed). We also get better results in loss/accuracy due to the fact that some image features, which in this case are the most prominent in intensity. In the BL model, average pooling is used which gives a more smooth image, but it seems like useful information is lost this way and this makes the MaxPooling perform better for this dataset. However, since the training loss is smaller than the validation loss, the model has potentially started to overfit slightly at the end, and training might have wanted to be stopped a bit earlier.

2.3. Filter model

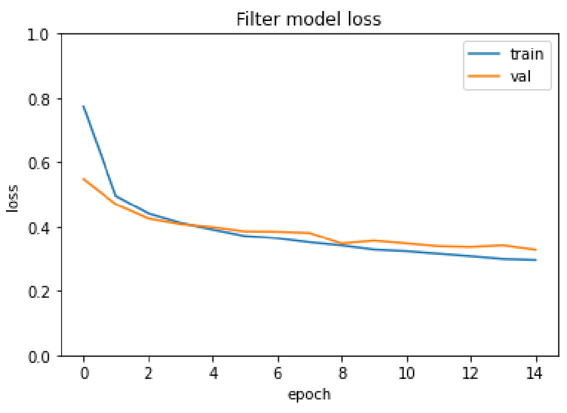
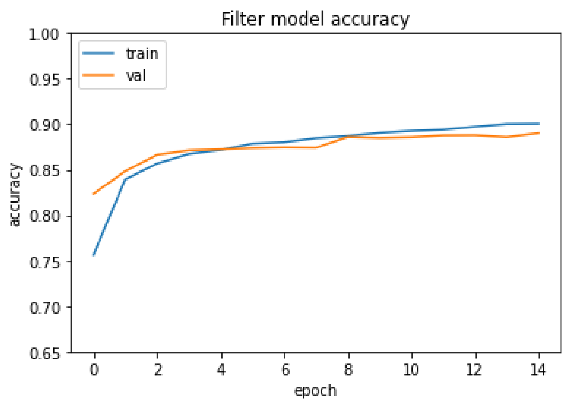
	
Training Loss: 0.2936	**Training Accuracy: 0.9028**
Validation Loss: 0.3250	**Validation Accuracy: 0.8953**

Table 3. Filter Model loss/accuracy

Compared to the BL model, the complexity increases significantly (number of parameters rose by 300%), which also increases the duration of training. We get better results in accuracy because it seems like more complex patterns can be extracted from the data in the 2 convolutional layers. However, the rise in accuracy as compared to the baseline model is rather small, suggesting that more filters did not necessarily benefit the dataset. This is probably due to the simplicity of the dataset where all objects are in the same position of the image and taken from the same angle. If the objects were taken from other angles it might be possible that more filters would have been necessary. Again, since the training loss is smaller than the validation loss, the model has potentially started to overfit slightly at the end, and training might have wanted to be stopped a bit earlier.

2.4. Convolution model

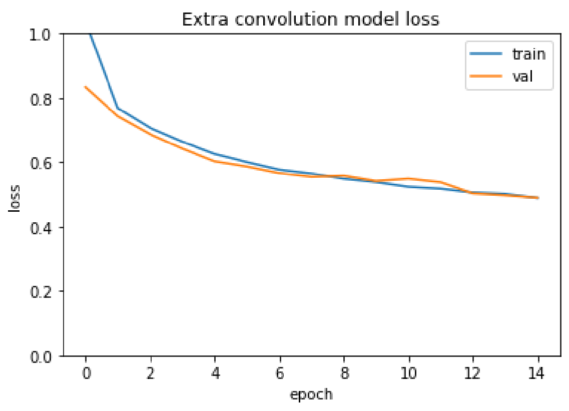
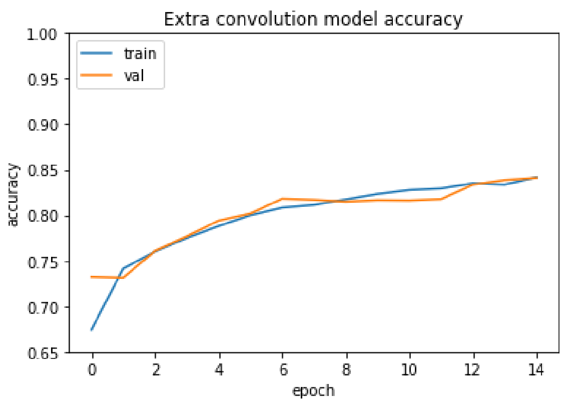
	
Training Loss: 0.4916	**Training Accuracy: 0.8368**
Validation Loss: 0.4919	**Validation Accuracy: 0.8403**

Table 4. Convolution Model loss/accuracy

Compared to the BL model, complexity is decreased (number of parameters reduced by ~60%). One more convolutional layer and average pooling layer is added. This time, the output shape of the final pooling layer is reduced (Flatten) and the duration of training decreases. Compared to the BL model, we get worse results in loss/accuracy. We think this is because of the complexity of the features that the model extracts. Basically, we assume that the model starts extracting features which can generalise to any class, therefore, not capturing the important distinctions between the different classes. Therefore, looking for more details in features is not benefitting this dataset.

2.5. Learning rate model

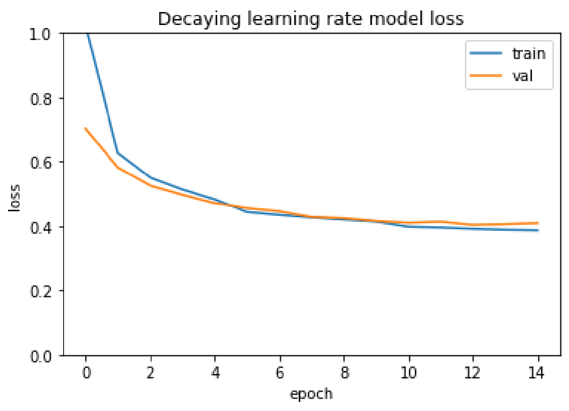
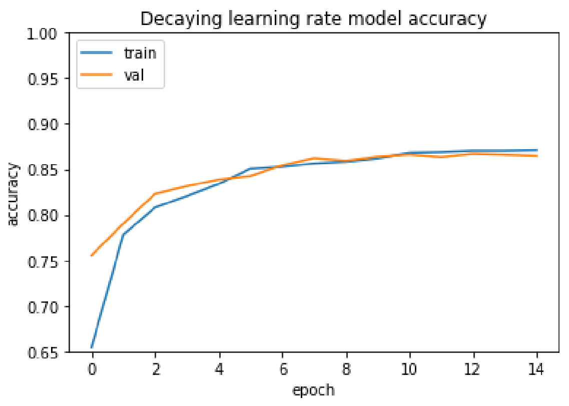
	
Training Loss LR 0.1: 0.3943	**Training Accuracy LR 0.1: 0.8660**
Validation Loss LR 0.1: 0.4029	**Validation Accuracy LR 0.1: 0.8648**

Table 5. Learning Rate[0.1] Model loss/accuracy

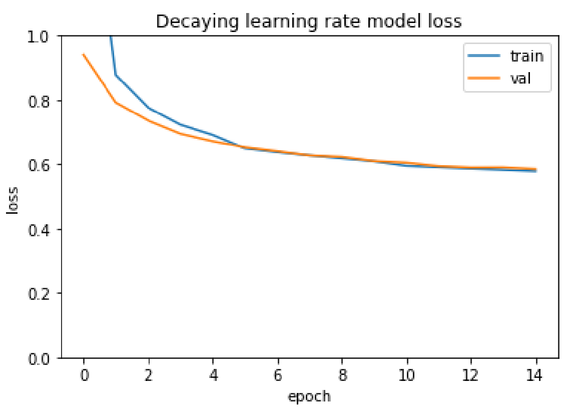
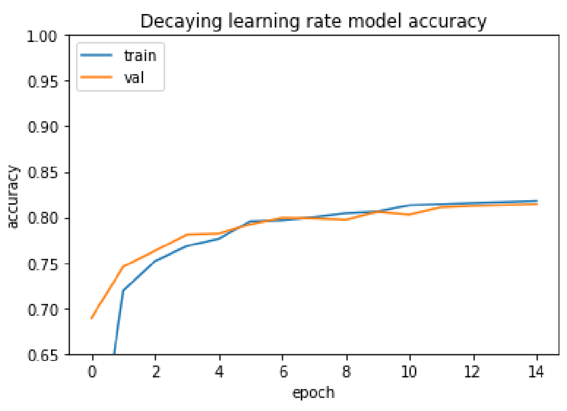
	
Training Loss LR 0.01: 0.5781	**Training Accuracy LR 0.01: 0.8165**
Validation Loss LR 0.01: 0.5837	**Validation Accuracy LR 0.01: 0.8153**

Table 6. Learning Rate[0.01] Model loss/accuracy

Compared to the BL model, the complexity stays the same (number of parameters have not changed). Only the optimizer is changed to SGD. We can observe more erratic behaviour during training, which is assumed to come from the shifts in learning rate that happens every five epochs. We evaluated two models, one with an initial learning rate of 0.1, and one with an initial learning rate of 0.01. The model with a learning rate of 0.1 gives better results out of the two, which might indicate that having a higher learning rate is more beneficial when the training only lasts 15 epochs. Still, it does not give as good results as the BL model, which might be because of the choice of optimiser, where the BL model's use of Adam - with its flexible learning rate adaptation - is advantageous.

2.6. Dropout model

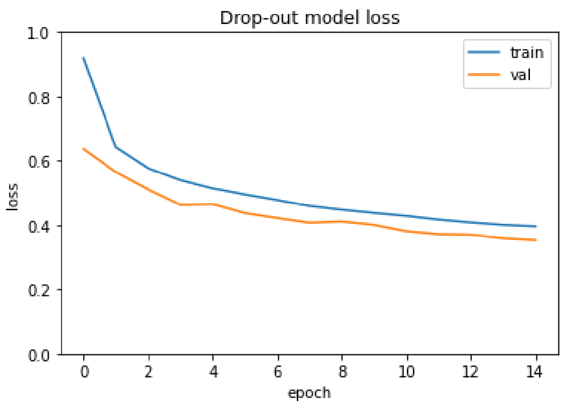
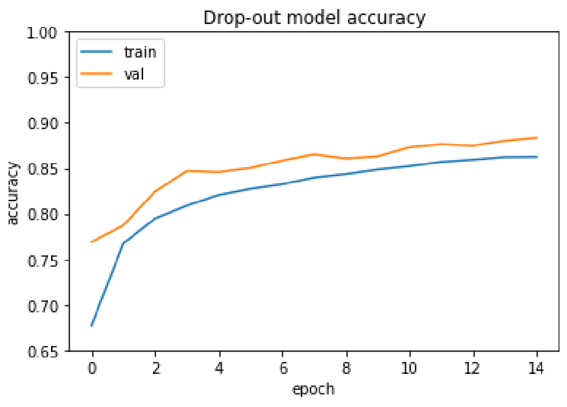
	
Training Loss: 0.4493	**Training Accuracy: 0.8365**
Validation Loss: 0.4025	**Validation Accuracy: 0.8543**

Table 7. Dropout Model loss/accuracy

Compared to the BL model, the complexity stays the same (number of parameters have not changed). Since a regularisation method has already been applied in the baseline model, applying another measure for overfitting leads to underfitting, as shown in the above table. Here, removing activation of certain neurons during training leads to a loss of information that is needed for the task at hand.

3. Table sorted based on Accuracy (Training/Validation)

Rank	**Model**	**Training Loss**	**Validation Loss**	**Training Accuracy**	**Validation Accuracy**
1	MaxPooling	0.2787	0.3103	0.9108	0.8987
2	Filter	0.2936	0.3250	0.9028	0.8953
3	Baseline	0.3039	0.3212	0.8974	0.8936
4	LR 0.1	0.3943	0.4029	0.8660	0.8648
5	Convolution	0.4916	0.4919	0.8368	0.8403
6	Dropout	0.4493	0.4025	0.8365	0.8543
7	LR 0.01	0.5781	0.5837	0.8165	0.8153

Table 8. Accuracy Table

4. Test set performance and evaluation

Model	**Validation Loss**	**Test Loss**	**Validation Accuracy**	**Test Accuracy**
MaxPooling	0.3103	0.3209	0.8987	0.8951
Filter	0.3250	0.3433	0.8953	0.8830

Table 9. Test Performance Table

The two models differ in architecture in respect to their pooling techniques and number of filters applied. In terms of performance, the MaxPooling model performs better on the test set than the Filter model, suggesting that the training of the MaxPooling model has been successful at extracting features that generalises well to new data. For the Filter model, the test accuracy is quite a bit lower than the validation accuracy, meaning that the same could not be said for this model. From this we can deduce that looking at sharp contrasts in the pixel values is better for classification than extracting more features.

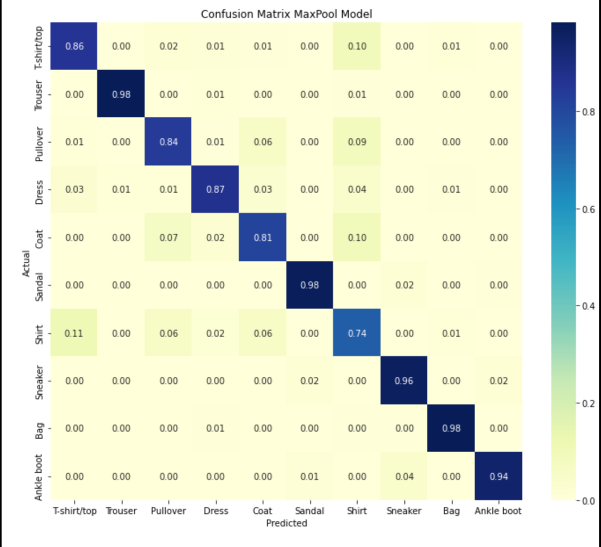
5. Choice Tasks

5.1. Learning Rate

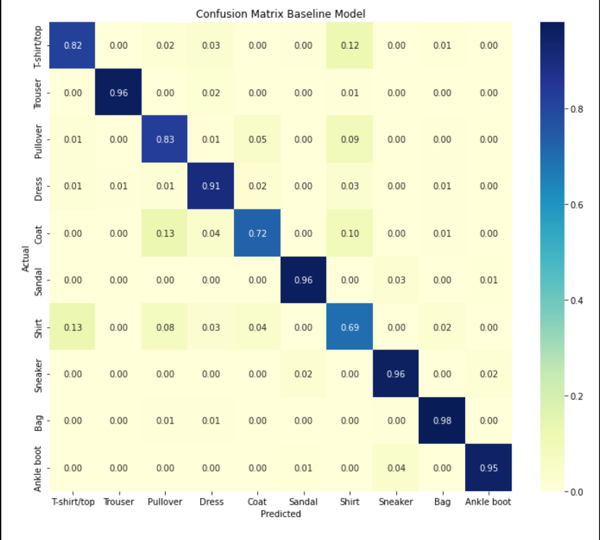
The implementation of the Learning Rate choice task has been already discussed in section 1.5 together with a performance review (section 2.5) with respect to the BL model.

5.2. Confusion Matrix

In Table 10 is the confusion matrix for the MaxPool model (left) and the BL model (right) using the test set.



Actual \ Predicted	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
T-shirt/top	0.86	0.00	0.02	0.01	0.01	0.00	0.10	0.00	0.01	0.00
Trouser	0.00	0.98	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00
Pullover	0.01	0.00	0.84	0.01	0.06	0.00	0.09	0.00	0.00	0.00
Dress	0.03	0.01	0.01	0.87	0.03	0.00	0.04	0.00	0.01	0.00
Coat	0.00	0.00	0.07	0.02	0.81	0.00	0.10	0.00	0.00	0.00
Sandal	0.00	0.00	0.00	0.00	0.00	0.98	0.00	0.02	0.00	0.00
Shirt	0.11	0.00	0.06	0.02	0.06	0.00	0.74	0.00	0.01	0.00
Sneaker	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.96	0.00	0.02
Bag	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.98	0.00
Ankle boot	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.04	0.00	0.94



Actual \ Predicted	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
T-shirt/top	0.82	0.00	0.02	0.03	0.00	0.00	0.12	0.00	0.01	0.00
Trouser	0.00	0.96	0.00	0.02	0.00	0.00	0.01	0.00	0.00	0.00
Pullover	0.01	0.00	0.83	0.01	0.05	0.00	0.09	0.00	0.00	0.00
Dress	0.01	0.01	0.01	0.91	0.02	0.00	0.03	0.00	0.01	0.00
Coat	0.00	0.00	0.13	0.04	0.72	0.00	0.10	0.00	0.01	0.00
Sandal	0.00	0.00	0.00	0.00	0.00	0.96	0.00	0.03	0.00	0.01
Shirt	0.13	0.00	0.08	0.03	0.04	0.00	0.69	0.00	0.02	0.00
Sneaker	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.96	0.00	0.02
Bag	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.98	0.00
Ankle boot	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.04	0.00	0.95

Table 10. Confusion Matrices

From the confusion matrix we can see that the MaxPool model is more successful than the BL model at distinguishing between different classes. Both models struggle most with classifying shirts, which is often mistaken for coats or t-shirts/tops. The MaxPool model has gotten better at classifying shirts, coats, pullovers and t-shirts/tops, which is not surprising since these are items that are very similar in shape, and therefore often gets misclassified. However, compared to the BL model this has led to a decrease in accuracy for other classes, which is a tradeoff that can happen when we reduce the loss for certain classes, leading to an increase in the misclassification rate for others. Overall, the confusion matrix shows us what labels are being classified correctly or not, which can give us additional information on what could change in the model architecture.

5.3. Output visualisation

To help understand the workings of the model and what features are represented at each layer, outputs from both convolutional layers and their respective pooling layers were obtained and visualised for the MaxPool model and the BL model.

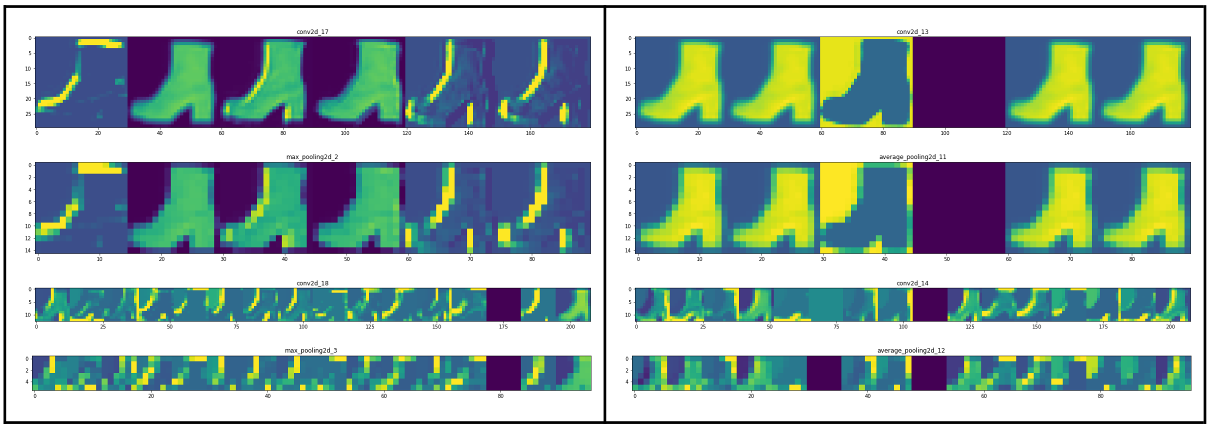


Table 11. Output Visualisation. Left: MaxPool model, Right: BL model

As can be seen in the table above, using `MaxPooling2D()` instead of `AveragePooling2D()` gives a very different representation in the feature maps. We can see that max pooling specifically pays attention to and exaggerates the brighter pixels, which manages to highlight the contours of the shoe in the image. Using average pooling instead leads to a more smooth and even representation of the object. In the second convolutional layer we see a closer representation of certain aspects of the data, where the feature maps are zoomed in on certain parts of the original image. This leads to the feature map capturing the object at different positions, which explains how CNNs can classify objects invariant of position in the image. This is not as important for this dataset though, as the objects are all positioned in the middle of the image and at the same angle.

Link to model weights:
https://drive.google.com/drive/folders/1OWh4QP_EafhurMinZKvGzmgqdd6OUQA2?usp=sharing

Bibliography

Park, S., & Kwak, N. (2016). Analysis on the Dropout Effect in Convolutional Neural Networks. *ACCV.*

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, & L. D. Jackel. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541-551, Winter 1989.