AI Lab: Neural Networks

Enias Cailliau, Miguel De Strooper

1 December 2017

Question 1

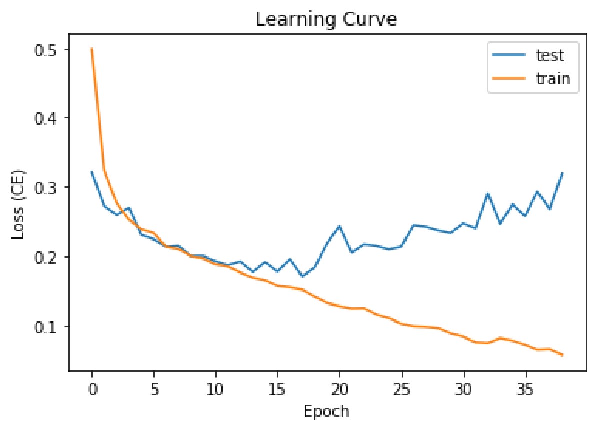
The file `prep.py` is responsible for loading the negative and positive data samples. The negative samples contain images of people looking neutral/not smiling while positive samples contain pictures of people laughing. After these images are loaded, its pixel values are loading as grey values (as the photos only include a single channel as opposed to traditional photos containing three channels for red, green and blue tints). After these steps, the resolution of the pictures are reduced using the `block_reduce` function (part of the `skimage.measure` Library). Since the `block_size` parameter is set to 1, no reduction is performed, and the original dimensions of the images remain untouched (64x64). A last and essential step completes a simple form of normalisation in which the pixel values of the images are normalised using a division (255). Thanks to this operation the pixel values are now between 0 and 1 which improves training performance.

Parameter	**Value**
shape X	(13165, 64, 64)
shape y	(13165,)
# training samples	13165
sample shape	(64, 64)
# output labels	2

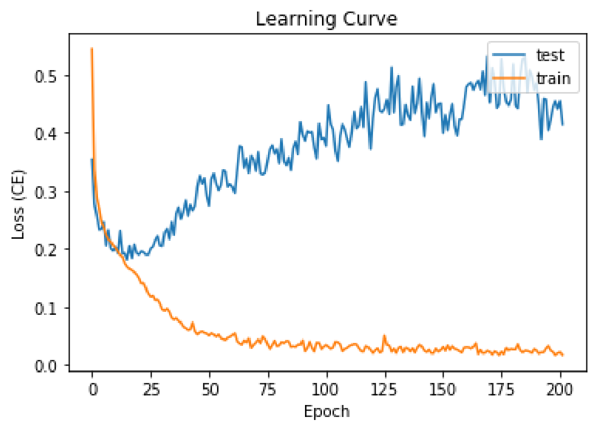
Question 2

During learning the `categorical_crossentropy` has been used as the loss function. Figure 1 displays two learning curves that relate the number of epochs with the train and test error values. Figure 1a uses an early stopping criterion that looks at the loss of the test set. If this stops decreasing for more than 25 epochs, it halts the learning process. Because of this strategy, we can see that the learning process has been terminated after 37 epochs. In contrast, Figure 1b displays a learning process in which early stopping only considers the loss of the training set. This criterion enabled the learning process to continue despite overfitting to the training set.

Both curves show that splitting the dataset into a training and testing set is required to avoid the network to overfit the training set which would result in inadequate performance on new examples to be classified. In the first part of the learning curves (before the train and test curves cross each other) we can say that the network is trying to generalise well without overfitting to the training set. However, after this intersection, we can see that the network starts to specialise since it works too hard to make sure that it does not make any errors for the training samples. This specialisation is visible through the train loss curve decreasing towards zero while the test loss curve keeps increasing. If the network achieves a training loss of zero, this means that it does not make any errors for classifying samples of the training set.



(a) Learning curve 1: early stopping based on the loss of the test set



(b) Learning curve 2: early stopping based on the loss of the training set

Figure 1: Learning curves depicting the influence of early stopping

Question 3

The experimental approach used to answer this question can be found in Table 1. From this table, we can see that lowering the test split results in an increase of the training accuracy and a decrease of the test accuracy. This observation can be explained through the following reasoning: The training process learns through 10 epochs from a training set. When the test split is smaller, this training set contains a larger amount of training data, which means that the model can better fit the training data during training. However, since there is a smaller test set it needs to be correct in many cases to achieve a high accuracy. Therefore, the smaller test set results in a lower accuracy. Figure 2 visualises the discussed conclusion.

Test Split	**Train Accuracy**	**Test Accuracy**
0.5	99.87%	97.75%
0.2	99.94%	96.92%
0.1	99.70%	95.90%
0.01	99.79%	95.45%

Table 1: Different data split ratios and its influence on train and test accuracy

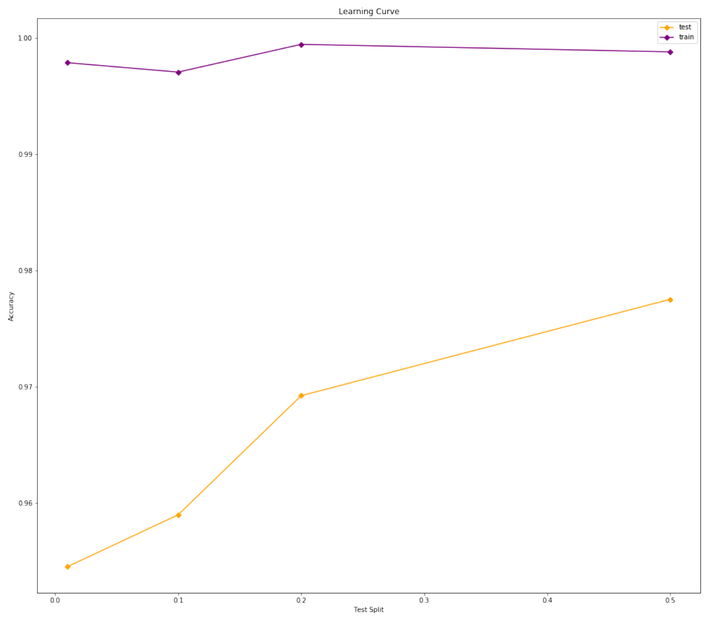


Figure 2: The influence of the ratio between test and training set on accuracy

Question 4

The problem with using accuracy as the only performance measure lies in its definition; it only looks at the proportion in which the classifier is correct (see Equation 1). Accuracy is so straightforward it can result in an overly simplistic conclusion. An example in which accuracy results in a wrong conclusion is provided in Table 2. This table concerns a test set with five samples for which the reference label is known. The accuracy of the simple classifier is 80%. However, we can see that this classifier always outputs False as a label and therefore has no predictive power. Adding other performance metrics such as precision (Equation 2) and recall (Equation 3) improves our insights into the performance. In our case, the precision and recall will both be 0%. These values allowed us to see that there is something wrong with the current implementation.

A perfect classifier would have a precision and recall that are both 100%. Using precision-recall allows us to evaluate the performance of a model better as precision-recall would be 0 for our simple example.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

Sample	**True label**	**Predicted Label**
1	N	N
2	N	N
3	N	N
4	P	N
5	N	N

Table 2: Simple classification example

	Predicted	
	Positive	**Negative**
Positive	0 (TP)	1 (FN)
Negative	0 (FP)	4 (TN)

Table 3: Confusion matrix

Question 5

Figure 3 displays the network architecture that is used to address the current question. We applied grid search on the third convolutional layer (indicated with a red dashed outer line), considering kernel quantities between 32 and 256. The results of this analysis are depicted in Figure 4 and detailed in Table 4.

Note that the training accuracy remains modest here (around 93%) This is because the network mentioned above is complex. Therefore, requires more than 10 epochs to train to perform optimally. We had to use 10 epochs as this was part of the assignment. However, for the next questions, we use more epochs and early stopping strategies to find the optimal result. This will be described in the next sections.

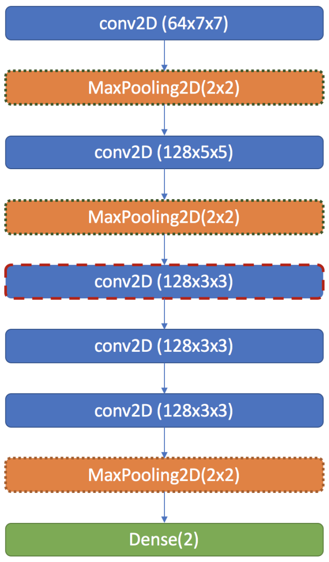
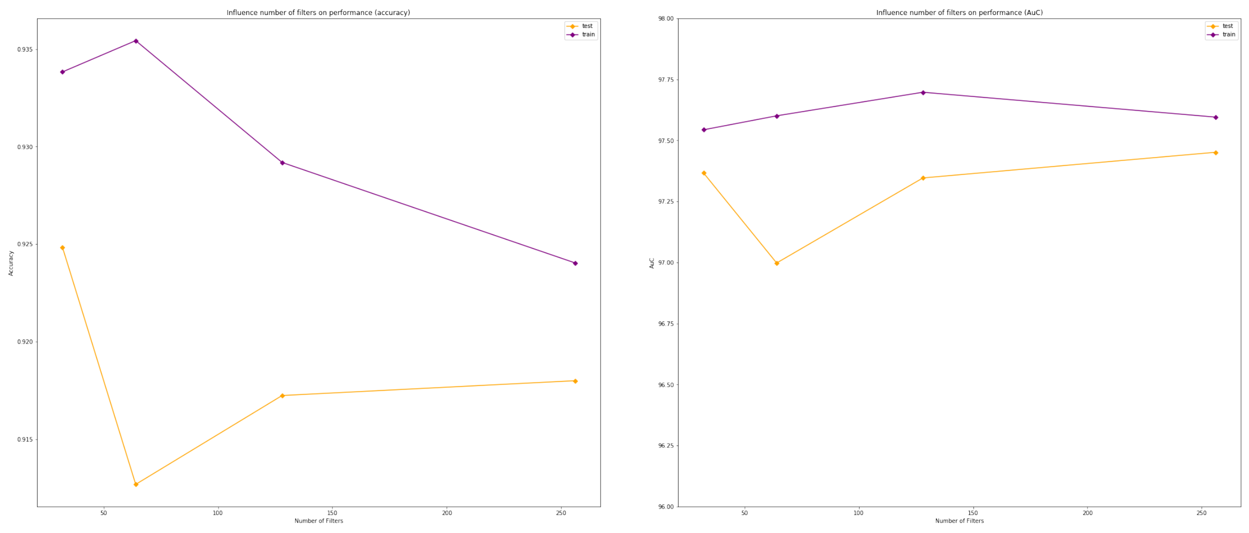


Figure 3: Network structure



(a) Accuracy performance

(b) AuC performance

Figure 4: Analysis: influence number of kernels on performance

# Kernels	Train accuracy(%)	Test accuracy(%)	Test AuC(%)	Test AuC(%)
32	93.38	92.49	97.54	97.37
64	93.54	91.27	97.60	97.00
128	92.92	91.72	97.70	97.34
256	92.40	91.80	97.60	97.45

Table 4: Analysis: influence number of kernels on performance

Question 6

Our best performing network is visualised in Figure 3. It uses several large kernels since research showed that this performed well for images of 64x64. We came to this model using a custom `metrics.py` class that prints out different metrics (accuracy, precision and AuC) at the end of each epoch. Furthermore, we wrote our personal early stopping implementation that halts the training process when the validation AuC does not improve for more than 10 epochs. We also played with different forms of data augmentation. We built a custom augmented dataset and also experimented augmentation generators from Keras. We had the best results by leaving data augmentation out of the equation which means that we are overfitting to the dataset. But as the goal was to achieve over 97% AuC this is as expected.

The equation to calculate the number of parameters is provided in Equation 4. The complexity of the network resides in the number of filters used for every layer which is also clear from the calculations of the parameters in Table 6. The number of filters is significant for all the layers. Hence, we needed around 35 epochs to achieve a network that performed optimally.

$$\#parameters = (filter\_height * filter\_width * input\_channels + 1) * number\_of\_filters \tag{4}$$

The metrics for our best model are shown in Table 5. It greatly passes the requirement of the ROC AuC which needs to be higher than 97 %. We also want to note that this network quickly converges to its best state. The values of the table are calculated after the third epoch. We did find that a good initialisation is critical to finding this solution.

Train accuracy(%)	Test accuracy(%)	Train AuC(%)	Test AuC(%)
99.48	98.86	99.95	99.90

Table 5: The metrics for our best performing model

Layer type	**Output shape**	**# parameters**
Conv2D	None, 58, 58, 64	(7*7*1+1)*64=3200
Activation	None, 58, 58, 64	0
MaxPooling2	None, 29, 29, 64	0
Dropout	None, 29, 29, 64	0
Conv2D	None, 25, 25, 128	(5*5*64+1)*128=204928
Activation	None, 25, 25, 128	0
MaxPooling2	None, 12, 12, 128	0
Dropout	None, 12, 12, 128	0
Conv2D	None, 10, 10, 128	(3*3*128+1)*128=147584
Activation	None, 10, 10, 128	0
Conv2D	None, 8, 8, 128	(3*3*128+1)*128=147584
Activation	None, 8, 8, 128	0
Conv2D	None, 6, 6, 128	(3*3*128+1)*128=147584
Activation	None, 6, 6, 128	0
MaxPooling2	None, 3, 3, 128	0
Dropout	None, 3, 3, 128	0
Flatten	None, 1152	0
Dense	None, 2	(1152+1)*2=12306
Activation	None, 2	0
Total		653186

Table 6: Network structure

Question 7

By occluding different regions of the image important conclusions can be derived. The different probability calculations are depicted in Table 7. As we expect is Figure 5 classified by the neural network as a smile with high certainty. This image is the baseline of our first experiment. The lower confidence that the CNN has for Figure 6 infers that we confused the network by occluding the eyes and cheeks. This confusion would be the same when the person would wear glasses indicating that more samples should have been available in our training set.

The high probability for Figure 7 indicates that the mouth is one crucial feature of the network. The CNN does not need the white teeth for this feature but a contrast on the position where the mouth should be and the mouth corners. This hypothesis will further be diagnosed in the next experiment.

For the last experiment, Figure 8 proves that teeth are not essential to recognise a smile. However, the contrasting difference between the position of the mouth and the mouth corners are. The smiling man has no teeth but is positively categorised as smiling. Figure 9 and Figure 10 makes our conclusion complete; the generated mouth results in a significant rise of the probability to contain a smiling person, where it was first unlikely to appear.

File	**Smile probability (%)**	**Neutral probability (%)**
Figure 5	91.35	8.65
Figure 6	77.42	22.58
Figure 7	98.76	1.24
Figure 8	99.37	0.63
Figure 9	0.59	99.41
Figure 10	99.97	0.032

Table 7: CNN evalutation

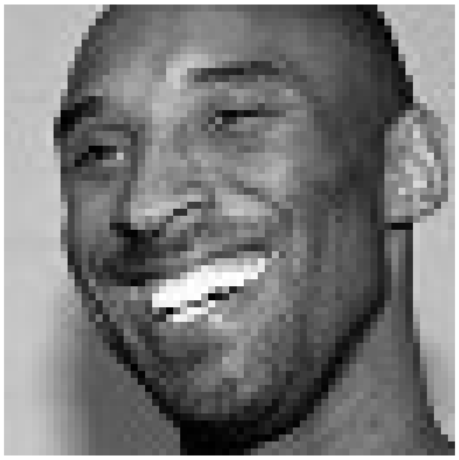


Figure 5: Smile teeth

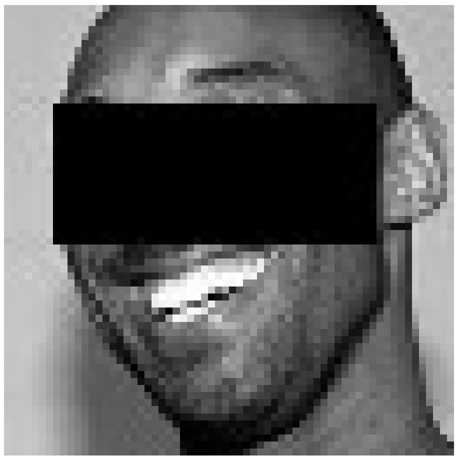


Figure 6: Smile teeth: occluded eyes and cheecks

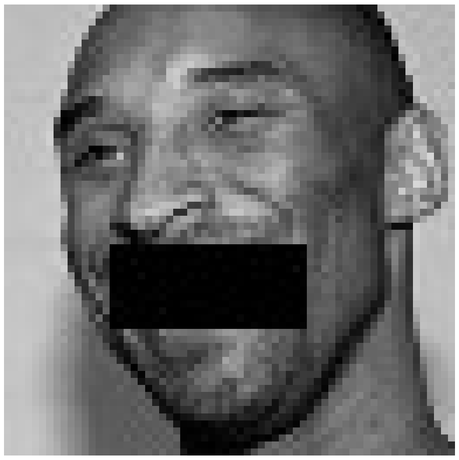


Figure 7: Smile teeth: occluded mouth

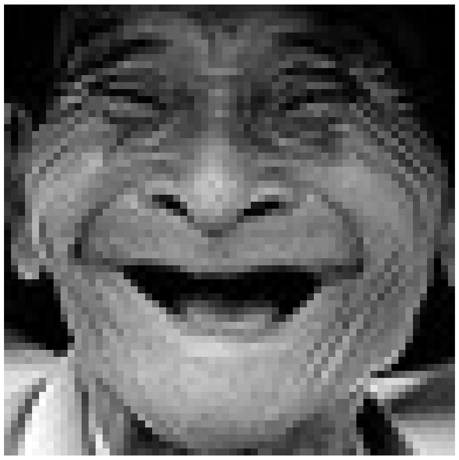


Figure 8: Smile open no teeth

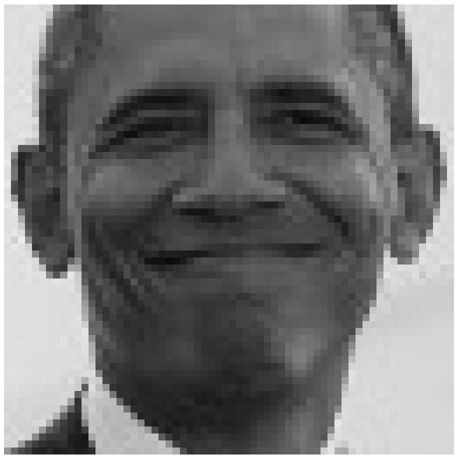


Figure 9: Smile closed teeth

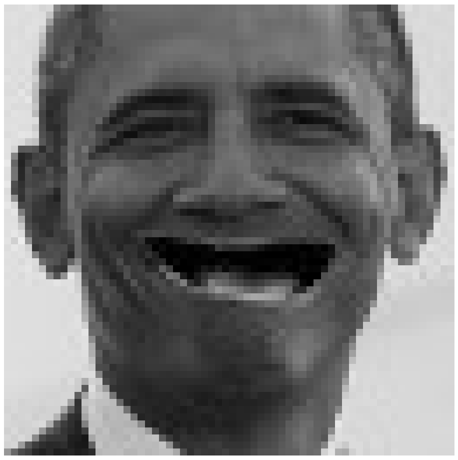


Figure 10: Smile closed: generated mouth with no teeth