Artificial Intelligence Practicum 3
Neural networks

Garben Tanghe
Olivier Van den Nest
Group 16

November 30, 2019

Question 1

There are 20685 training samples, 10317 validation images, and 9950 test samples. This is approximately a $50\,\%$ - $25\,\%$ - $25\,\%$ split.

The images are $(100, 100, 3)$-shaped tensors. The first 2 dimensions represent the number of pixels of an image, here thus 100 by 100 pixels. The third dimension stands for the number of color channels, here thus 3 for red, green, and blue (RGB). Each value in this tensor is a 16-bit floating point number.

For new images hold:

- They must have an aspect ratio of 1:1. (Square)
- They must be 100 by 100 pixels.
- They preferably have a white background.
- The fruit or vegetable must be centered in the images.
- They must be cropped so that the fruit accounts for the largest part of the image. (Zoomed-in)
- They must have RGB-values in the same order as in the original dataset.
- If the images have RGB-values from 0 to 255, they must be normalized, so they become floating point numbers between 0 and 1 (with 0 black and 1 white, thus the intensity of that color).

Question 2

The classes, in alphabetical order, are fruits and vegetables in ['apple', 'banana', 'cherry', 'grape', 'onion', 'peach', 'pear', 'pepper', 'plum', 'potato', 'tomato'].

The labels from y_train, y_valid, and y_test are numbers from 0 to 10 (11 class labels). 0 corresponds to 'apple', 1 to 'banana', etc. They are just the index of the label in the list

above.

y_train has [4330, 959, 2296, 2296, 900, 1148, 2426, 1184, 1212, 1206, 2728] samples from each corresponding class.
y_valid has [2164, 478, 1148, 1148, 450, 574, 1212, 592, 602, 603, 1346] occurrences from those class labels.
y_test has [2060, 477, 1148, 1121, 429, 574, 1212, 592, 550, 595, 1192] samples from the class list.
A histogram of this data is given in Figure 1. There is quite some class imbalance, as there are for example far more apples than bananas. This is not a nice property of the dataset. However, the different subsets have approximately the same distribution, which is a desired property.

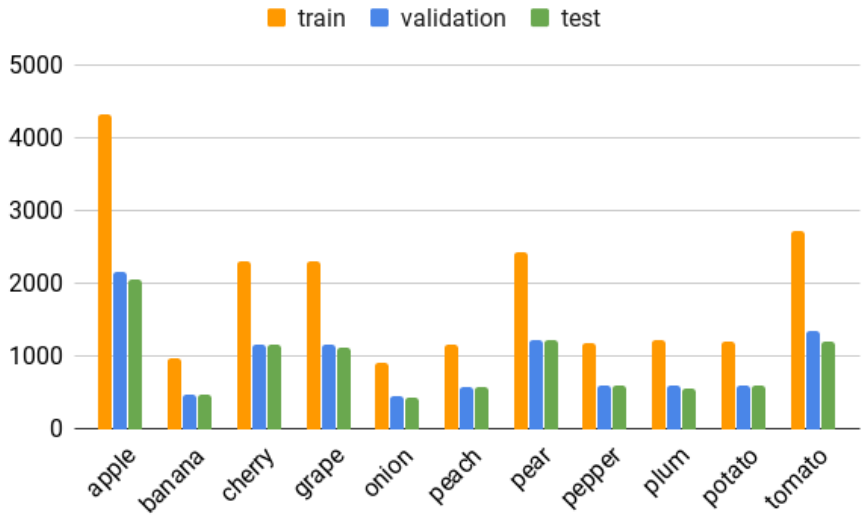


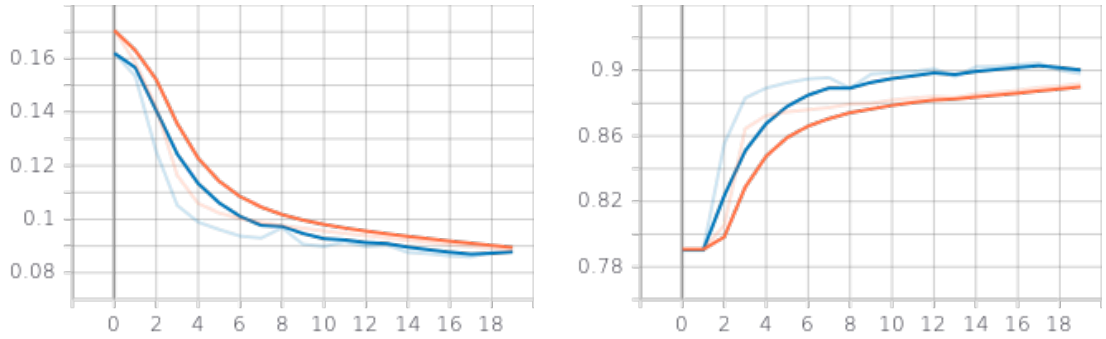
Figure 1: Histogram of all training, validation and test samples.

Question 3

Around 20 epochs, the training loss starts to converge. This can be seen in Figure 2a. Since the validation loss is going down along with the training loss, the model is generalizing instead of specializing. It learns general features to distinct between apples and others. The model does not specialize/overfit to the training data, which is wanted.

Question 4

The output in y_test_pred is an array with one element for each test sample. Each element represents the predicted probability (as a decimal number between 0 and 1) of the input being an apple. For example, the first output is 0.9948294. The neural network is thus over $99\,\%$ sure that the input input image was from an apple. When the first test input is plotted, there can be verified that this is indeed an apple.

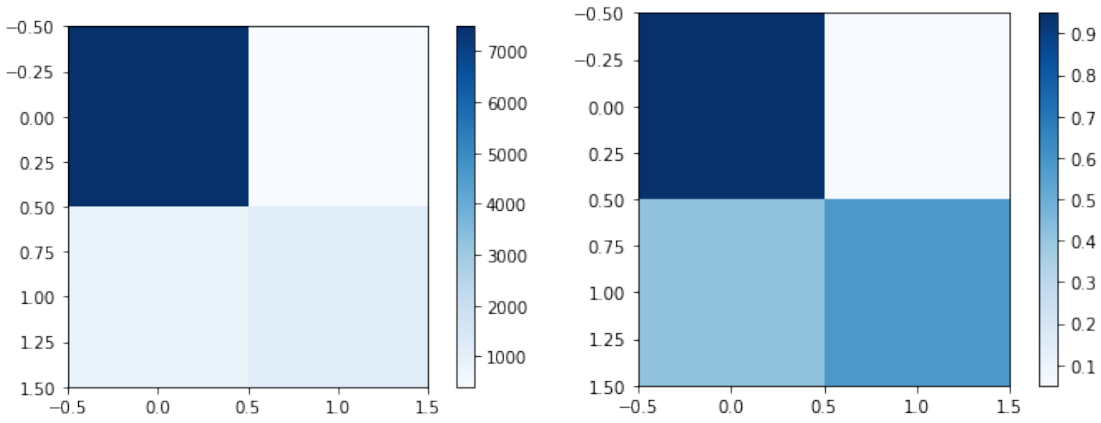


(a) Loss (b) Accuracy

Figure 2: Train and validation curves for apple only for 20 epochs.

When the model predicts values greater than or equal to 0.5, 'apple' is predicted as class label and if values are lower than 0.5, 'other' is assigned. This binary classification is thus done by means of a threshold.

The confusion matrix and its normalized version are shown in Figure 3. From the near-diagonal matrix in Figure 3a, there can be seen that almost every prediction is correct. This plot also shows that there are many more 'no apples' (True Negatives in the top left cell) than 'apples' (True Positives on the bottom right). From Figure 3b, there is derived that there are slightly more images predicted as 'no apple', while they are in fact an apple (False Negatives in the bottom left cell), than the other way around (False Positives, top right).



(a) Absolute (b) Normalized

Figure 3: Confusion matrices for apples only after 20 epochs.

The accuracy metric can be largely influenced by the number of true negatives, which was also found out above. The F1-score considers both the precision and recall and tries to find a balance between them. Its definition is given in Equation 1. This is a way to counter class imbalance, that punishes wrong predictions of the lesser represented classes harder. Since the model has more False Negatives (FN) than False Positives (FP), recall is lower than precision, which can be improved by choosing the F1 score as a metric. Binary crossentropy (BCE) is a specific case of crossentropy with 2 classes. Its definition is given in Equation 2. Where y is the true class label (0 for the first class

'apple' or 1 for the second class 'no apple') and p is the probability for the second class. This metric does not punish overcorrectly classified samples, unlike mean-squared error (MSE). It is also less influenced by outliers. Focal loss is a metric invented by Facebook AI Research (FAIR). It can work for both binary and multiclass classification while taking into account class imbalance. Its formula is added in Equation 3 for completeness. p_t is equal to p if $y = 1$, $1 - p$ otherwise.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{1}$$

$$BCE = y * log(p) + (1 - y) * log(1 - p) \tag{2}$$

$$FL(p_t) = -(1 - p)^{\gamma} log(p_t) \tag{3}$$

The ROC curve can be found in Figure 4. The True Positive Rate (TPR) has a steep slope early on, when the threshold for apple classification is still high. This is nice, but can still be improved a little further. Ideally, the blue line would rise fast to the top left corner, meaning no more non-apples are classified to apples when the threshold for apple classification is lowered.

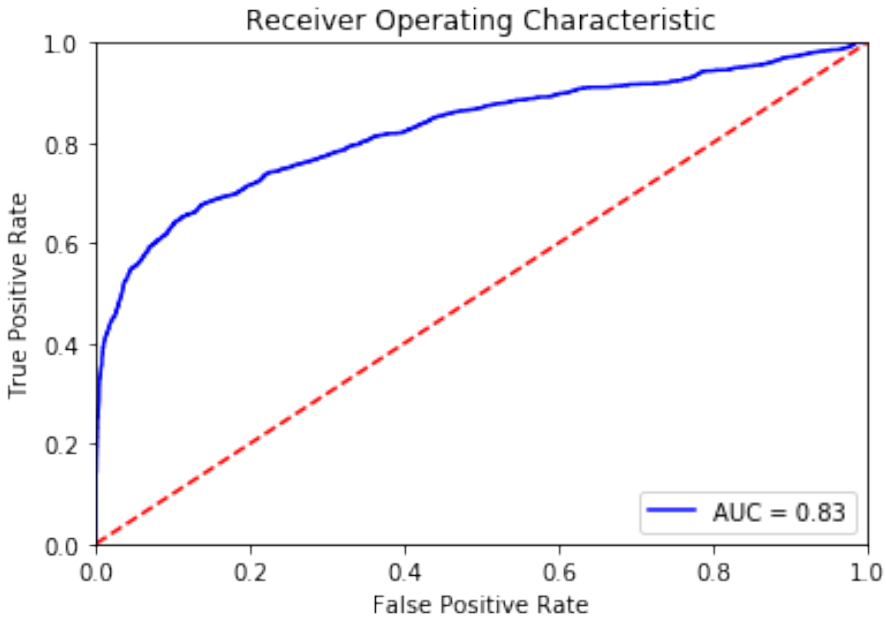


Figure 4: ROC curve for apples only after 20 epochs.

Question 5

The learning rate defines how fast weights are being updated during the backtracking algorithm of the Stochastic Gradient Descent (SGD) optimizer.

The moment the loss starts to rise, the learning rate becomes to high. In general, a high learning rate moves quickly towards the optimum, but when it is to big, it will jump over the optimum. A low learning rate will crawl slowly towards the optimum, but when it is to low, it might take too long to reach the optimum. A good balance or a decreasing

learning rate over time is advised, so that the optimum is found in a reasonable amount of time. Based on the results in Figure 5, a learning rate between 0.01 and 1 should be a good choice.

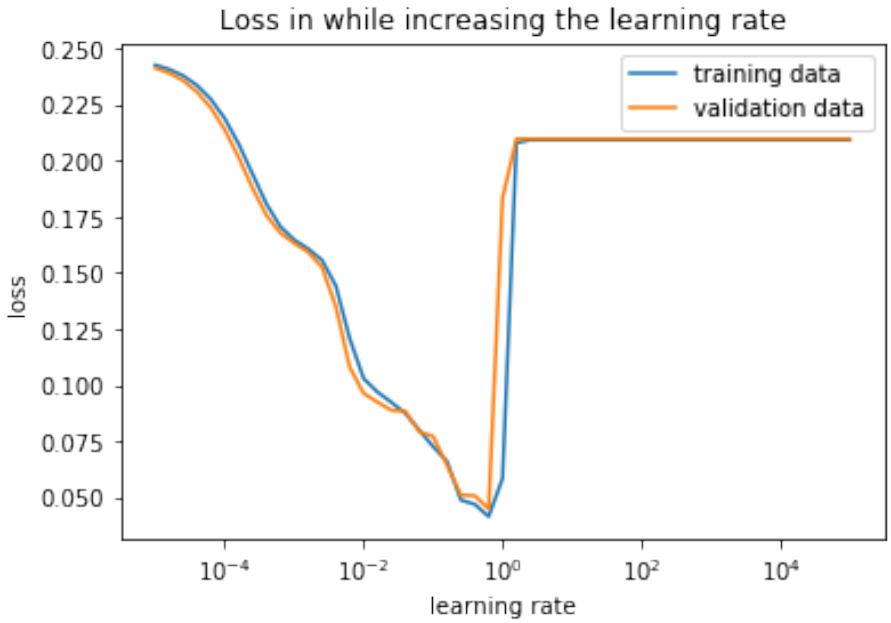


Figure 5: Varying learning rate for apples only.

Question 6

For the following models, MSE is used as loss and the learning rate is fixed on 0.5. The last convolutional layer will have a varying number of output feature maps, while the rest of the model remains unchanged. Each of the networks is then trained for 10 epochs. The performance from each model is measured as the accuracy on the test set.

For $k = 3$ output feature maps, the accuracy is 0.9169.
For $k = 4$ output feature maps, the accuracy is 0.9130.
For $k = 5$ output feature maps, the accuracy is 0.9221.
For $k = 6$ output feature maps, the accuracy is 0.9624.
For $k = 7$ output feature maps, the accuracy is 0.9661.
For $k = 8$ output feature maps, the accuracy is 0.9556.
For $k = 9$ output feature maps, the accuracy is 0.9201.

The total number of parameters in a convolutional layer is $(n*m*l+1)*k$. Due to weight sharing, this number does not have to be multiplied with the size of the input feature maps. The variables n and m come from the filter size of the convolutional kernel. l is the number of input feature maps, while k is for the number of feature maps in the output. The $+1$ comes from the bias. For the convolutional layers used in the network, the number of parameters is thus $(3 * 3 * 3 + 1) * 4 = 112$, $(3 * 3 * 4 + 1) * 4 = 148$, and $(4 * 4 * 4 + 1) * k = 65 * k$, in that order. For the optimal value of $k = 7$, this results in $65*7 = 455$. The number of parameters in a fully-connected dense layer is $(l+1)*k$. The

meaning of these variables remains unchanged. For the first dense layer, the number of parameters is $(112+1)*4 = 452$. The last dense layer has $(4+1)*1 = 5$ parameters. Average pooling and flattening layers have no parameters.

The effect of the amount of feature maps can be described as follows: every node in a convolutional layer represents one feature map/filter. The more feature maps, the better this layer (and model) can describe the training data. However, if too much feature maps are used, the layer will be specialized only on this training data and it will be very sensitive to noise from other data. This also implies bad generalization. On the other hand, if too few feature maps are used, the model will generalize too much, but will underfit since a lot of valuable information will not be extracted.

Question 7

The number of outputs in the last dense layer is 11, because there are 11 classes.

Sigmoid only works for 2 classes in binary classification. For multiple classes in multi-class classification another activation function is needed in the last layer. Softmax is a way to normalize the logits in the output, to values between 0 and 1. It is defined by the formula in Equation 4. $\sigma_i^{'}$ is the probability corresponding to the score/logit σ_i and C is the number of classes in the classification problem.

$$\sigma_i^{'} = \frac{e^{\sigma_i}}{\sum_{j=0}^{C} e^{\sigma_j}} \tag{4}$$

The categorical crossentropy (CCE) is used as loss for multi-class classification. The loss itself is specific for one category (hence the name) and is defined using the formula in Equation 5. It is simply the logarithm of the output of the softmax layer. The behavior with respect to outliers and overcorrectly classified samples is the same as for BCE, meaning that they will be punished less hard.

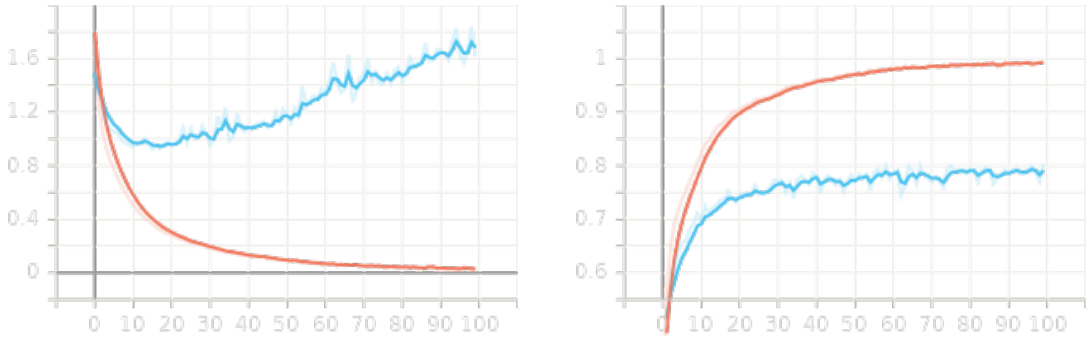
$$CCE = -log\left(\frac{e^{\sigma_i}}{\sum_{j=0}^{C} e^{\sigma_j}}\right) = -log(\sigma_i^{'}) \tag{5}$$

Question 8

The network with 7 output feature maps in the last convolutional layer is now used for multiclass classification. The loss has been changed to CCE, while the optimizer is now an Adam optimizer with the default parameters. There is trained for 100 epochs and the accuracy is again used as performance metric.

The loss and accuracy in function of the epochs can be seen in Figure 6. The validation loss goes down until around epoch 16 and then rises again, but the validation accuracy is the highest at epoch 100. The last checkpoint will now be used for the remaining part of the question. The loss on the testset is 2.28, while the accuracy is $77.9\,\%$.

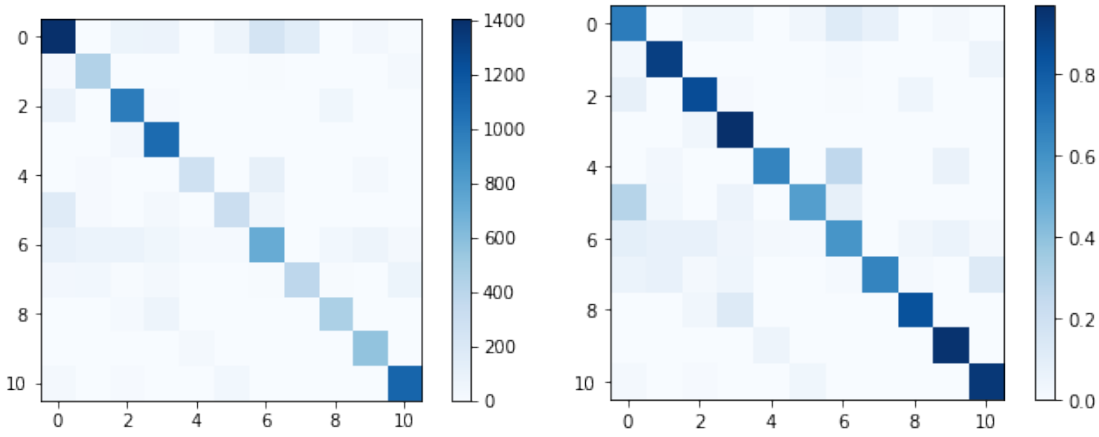
The confusion matrix and its normalized version can be seen in Figure 7. Both are almost diagonal matrices, which means that the classifier performs well. The hardest classes are the darkest cells that are not on the main diagonal of the normalized matrix.



(a) Loss (b) Accuracy

Figure 6: Train and validation curves for all classes for 100 epochs.

The peaches (5) are the hardest to predict, as they are often predicted as apples (0). The unions (4) are often confused with pears (6).



(a) Absolute (b) Normalized

Figure 7: Confusion matrices for all fruits after 100 epochs.

Question 9

Example images of all types for the binary classifier can be found in Figure 8, while an example of a correct and incorrect prediction is provided in Figure 9.
Figure 8a is correctly classified as not an apple (True Negative) with 91.5 % certainty.
Figure 8b is incorrectly classified as an apple (False Positive) with 100.0 % certainty.
Figure 8c is incorrectly classified as an not apple (False Negative) with 99.5 % certainty.
Figure 8d is correctly classified as an apple (True Positive) with 100.0 % certainty.
Figure 9a is correctly classified as an apple with 64.9 % certainty.
Figure 9b is incorrectly classified as a tomato with 99.2 % certainty.

Using larger models like InceptionResNetv2, which are pretrained on very large datasets like ImageNet, will certainly outperform the simple models used in this lab session. By using transfer learning, the bottom layers no longer have to learn many small features as they are already learned on other datasets and thus the top layers can focus solely

(a) True Negative

(b) False Positive

(c) False Negative

(d) True Positive

Figure 8: Example images of all types for the binary classifier.

(a) Correct (b) Incorrect

Figure 9: Example images of correct and incorrect predictions of the multi-class classifier.

on the classification task. The images in the dataset are all well preprocessed, but in the real world this will not be the case. Currently, the model does not know what to do when the parts of the image around the fruit is not white. Many images of fruit and vegetables in their context should be used, so that the model can detect the class labels independent of the context. Data augmentation could help to improve performance of all networks even further. Using the top-k predictions could also be useful for some applications, as top-k accuracy will always be higher than the accuracy of the class with highest probability. For example in the Colruyt case, the top 3 predictions are being displayed. Afterwards, the customer has to indicate the correct one.