Lund University
Faculty of Engineering (LTH)

EDAN95

Applied Machine Learning

Project 3

Author:
Marcel Attar, 941127-2173

The report was handed in on: December 2, 2019

1 Collecting a Dataset

I downloaded the data from Kaggle and used the provided code from GitHub [3] to prepare the data and split it into training, validation, and test sets, using a 60/20/20 ratio as specified in the lab notes [2].

2 Building a Simple Convolutional Neural Network

I created a convolutional neural network with 4 convolutional layers, with relu activation functions and maxpooling after each layer. I ended with 2 dens layers, the first one with a relu activation function and the last with a softmax, since it's a multiclass classification problem. To avoid overfitting I added a dropout layer.

Results

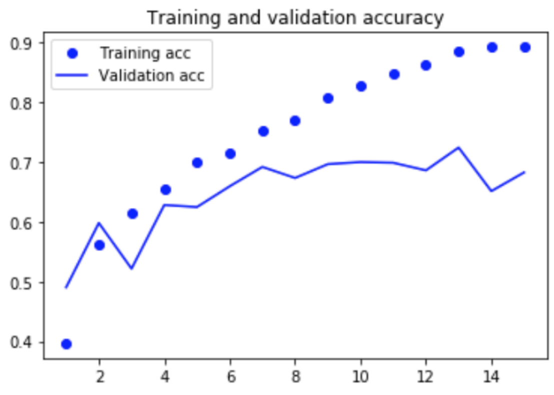


Figure 1: The training and validation accuracy for the first model, using a dropout layer towards the end of the network.

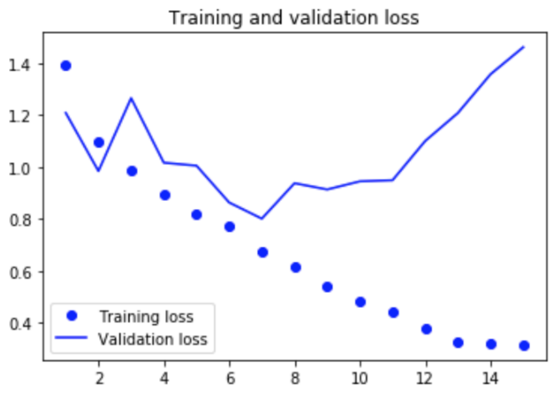
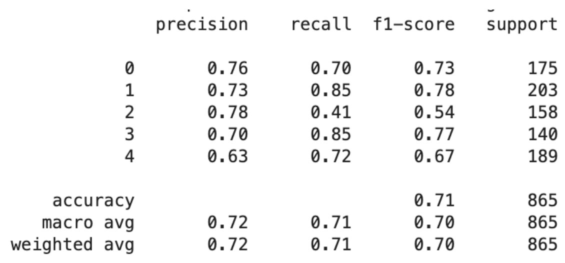


Figure 2: The training and validation loss for the first model, using a dropout layer towards the end of the network.

There is still some overfitting since the training and validation accuracy have some separation between them.

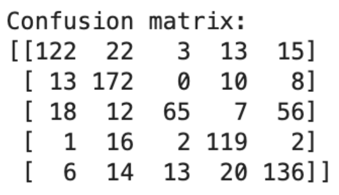


```
              precision    recall  f1-score   support

           0       0.76      0.70      0.73       175
           1       0.73      0.85      0.78       203
           2       0.78      0.41      0.54       158
           3       0.70      0.85      0.77       140
           4       0.63      0.72      0.67       189

    accuracy                           0.71       865
   macro avg       0.72      0.71      0.70       865
weighted avg       0.72      0.71      0.70       865
```

Figure 3: The precision, recall and f1-score of the model.



```
Confusion matrix:
[[122  22   3  13  15]
 [ 13 172   0  10   8]
 [ 18  12  65   7  56]
 [  1  16   2 119   2]
 [  6  14  13  20 136]]
```

Figure 4: The confusion matrix of the model.

I see here that I got an accuracy of 0.71, fairly good for a simple model.

3 Using Image Augmentation

In this section we started to use image augmentation, i.e. altering the training images with different parameters such as shear, zoom, rotation, etc.. I did this by using the ImageDataGenerator module in python. The aim of this is to reduce the overfitting, reducing the separation between training and validation accuracy.

Results

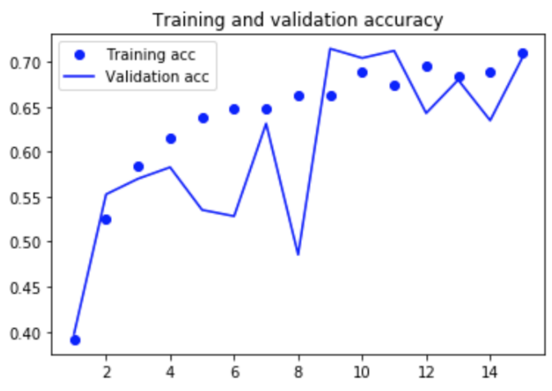


Figure 5: The training and validation accuracy for the second model.

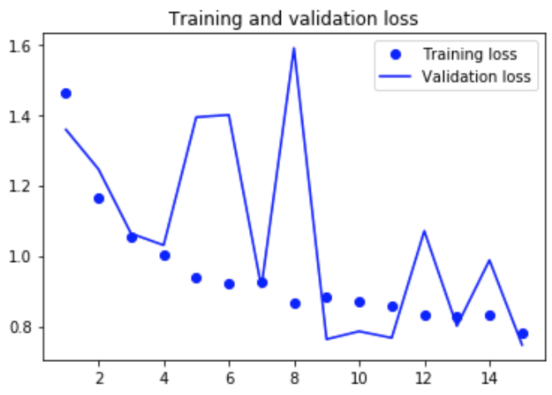
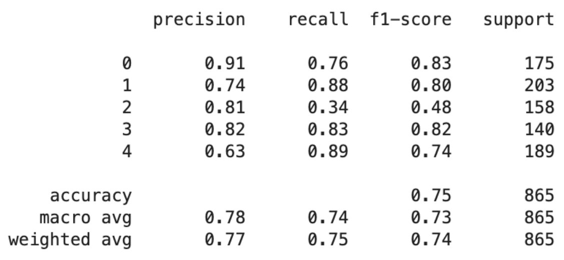


Figure 6: The training and validation loss for the second model.

As we see in figure 13 the training accuracy has decreased but the validation accuracy has increase, which of course is preferable, i.e. the overfitting has decreased. The variance of the validation loss function has increased since....

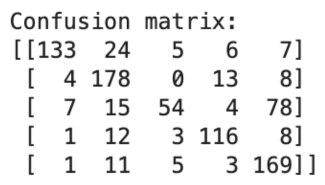


```
              precision    recall  f1-score   support

           0       0.91      0.76      0.83       175
           1       0.74      0.88      0.80       203
           2       0.81      0.34      0.48       158
           3       0.82      0.83      0.82       140
           4       0.63      0.89      0.74       189

    accuracy                           0.75       865
   macro avg       0.78      0.74      0.73       865
weighted avg       0.77      0.75      0.74       865
```

Figure 7: The precision, recall and f1-score of the model.



```
Confusion matrix:
[[133  24   5   6   7]
 [  4 178   0  13   8]
 [  7  15  54   4  78]
 [  1  12   3 116   8]
 [  1  11   5   3 169]]
```

Figure 8: The confusion matrix of the model.

We got a higher accuracy score, 0.75, using the augmented figures.

4 Using a Pretrained Convolutional Base

As a final attempt to increase the validation accuracy we downloaded a pretrained convolutional model from Keras (InceptionV3), that had been trained on the ImageNet dataset.

4.1 Fast Feature Extraction Without Data Augmentation

In this first approach we sent in our data to the pretrained network and call the predict method and save the output features in arrays. These arrays was later used to train our own network which consisted of two dense layers with a dropout layer in-between.

Results

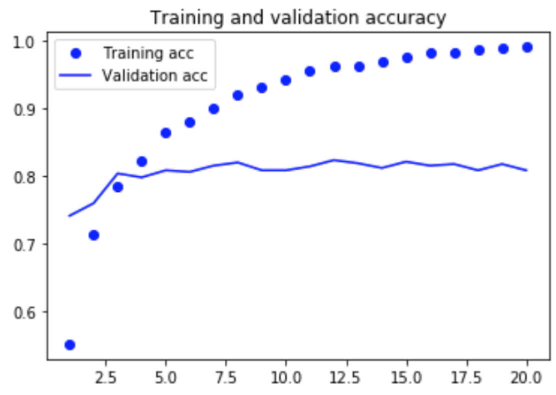


Figure 9: The training and validation accuracy for the second model.

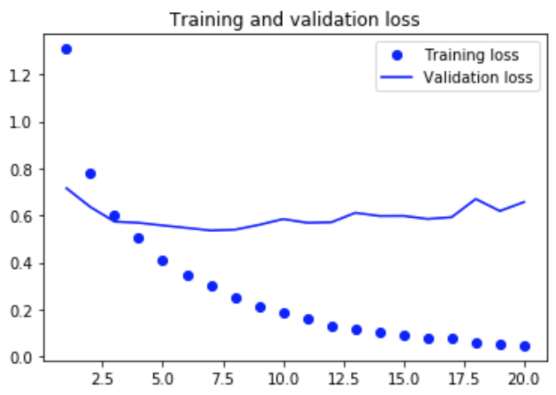
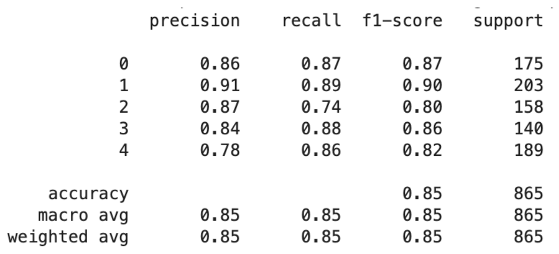


Figure 10: The training and validation loss
for the second model.

We see a clear imporvement in both accuracy and loss when using a pretrained network. After just one epoch we reach a validation accuracy of around 0.75 since the network is pretrained.

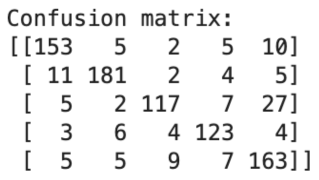


```
              precision    recall  f1-score   support

           0       0.86      0.87      0.87       175
           1       0.91      0.89      0.90       203
           2       0.87      0.74      0.80       158
           3       0.84      0.88      0.86       140
           4       0.78      0.86      0.82       189

    accuracy                           0.85       865
   macro avg       0.85      0.85      0.85       865
weighted avg       0.85      0.85      0.85       865
```

Figure 11: The precision, recall and f1-score of the model.



```
Confusion matrix:
[[153   5   2   5  10]
 [ 11 181   2   4   5]
 [  5   2 117   7  27]
 [  3   6   4 123   4]
 [  5   5   9   7 163]]
```

Figure 12: The confusion matrix of the model.

The accuracy of 0.85 can be compared to 0.75 from the previous model.

4.2 Feature Extraction With Data Augmentation

For our final approach we won't extract the features but instead use the entire conv. base and add two dense layers on top of it. However, we freeze all the parameters in the conv. base, i.e. we don't update the parameters. This allows us to use image augmentation.

Results

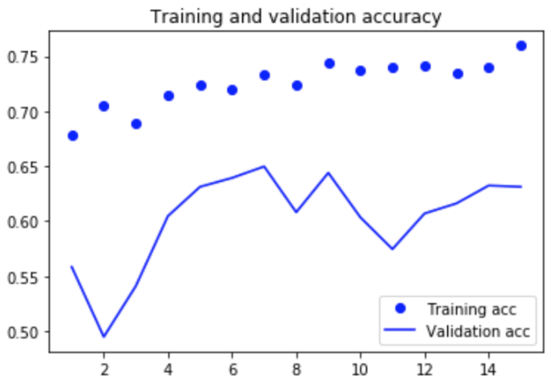


Figure 13: The training and validation accuracy for the second model.

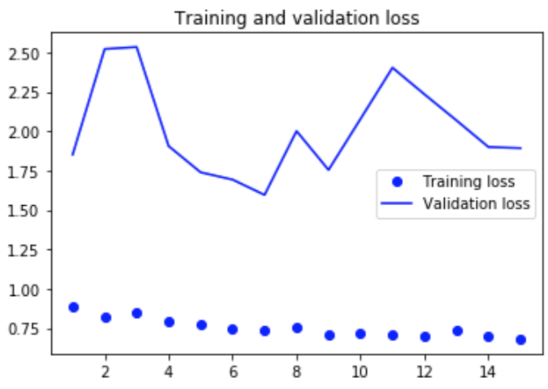


Figure 14: The training and validation loss
for the second model.

It's clear that this approach performed worse for this dataset compared to the previous model.

5 Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization

We were asked to read and comment on the article *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization* [4] by Selvaraju et al as well as running Chollet's notebook on *Github* [1]

The article describes a method of trying to visualize *what* features a convolutional network look at in images. It describes the problem that many complex neural network faces, when the model generate a misclassification it's often by a failure that's spectacularly disgraceful and leaves the humans dumbfounded. This "black box" problem is what the article is trying to address. The notebook also tackles the same problem but instead with three different methods (the last of which is the one described in the article).
Below we see two examples of a classification of 'cats' and 'dogs' and what regions the network finds important (the figures are from the provided article).

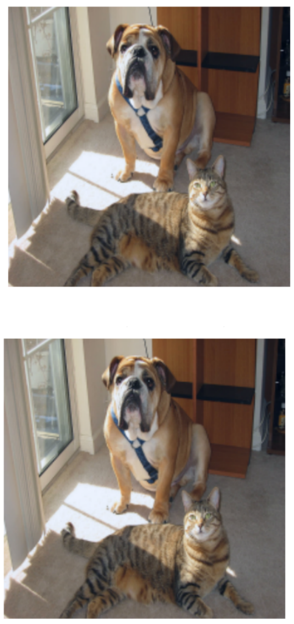


Figure 15: Original input image. [4]

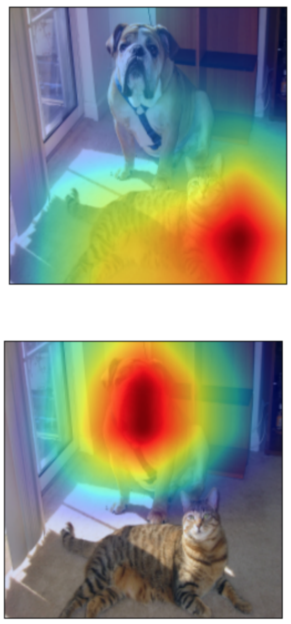


Figure 16: Image with the superimposed heatmap. [4]

The network finds the chest of the cat particularly "interesting" to be able to classify it as a cat. For the dog it's the head region that is of interest. The method uses gradient information that flows into to the last convolutional layer of the CNN and is thus able to understand the importance of each neuron. They compute the gradient for class c, y^c with respect to the feature map A^k. These are then global-average-pooled as

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \quad \frac{\partial y^c}{\partial A_{ij}^k} \tag{1}$$

These weights α_k^c captures the importance of feature maps k for class c.

References

[1] Francois Chollet. Deep learning with python notebooks. URL: `https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/5.4-visualizing-what-convnets-learn.ipynb`.

[2] Pierre Nugues. Lab session 3. URL: `http://cs.lth.se/edan95/lab-programming-assignments/lab-session-3/`.

[3] Pierre Nugues. pnugues: Github. URL: `https://github.com/pnugues/edan95`.

[4] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *eprint arXiv:1610.02391*, October 2016. `doi:https://arxiv.org/abs/1610.02391v3`.