Report “Pawpularity”

by Cass Maes, Demi Soetens, Bente van Katwijk and Ilse Feenstra.

Chapter 1: Basic model

Introduction:

In western countries, owning pets is tremendously popular and pet owners would do anything to protect them. However, this is not so common everywhere in the world. In most countries, animals are living outside on the streets, without an owner taking care of them. They often end up in shelters, where they might be euthanized because of acquired trauma or diseases.

The Malaysian website PetFinder.my is a website where you can find over 180 000 of these stray animals to adopt. What is remarkable about this website is that it uses a basic Cuteness Meter to rank their pet photos. Their goal is to predict the ‘Pawpularity’ of the pet photos, utilizing several features and the performance of thousands of pet profiles. This score could help shelters to create better pet profiles, by providing them with better pictures of their pets. Consequently, this could lead to a higher probability of the pet being adopted by a loving owner and eventually even preventing shelters from being overcrowded.

The Project

For our project, we will analyze thousands of raw images and tabular data from PetFinder.my’s pet profiles to create a model that is able to accurately predict the ‘Pawpularity’ of pet photos. The model will take 12 features into consideration, such as whether or not the face of the pet is displayed in the photo, if it is an action shot, if a blur is added to the photo and if there is a human being in the photo. Using these features, the images and the given ‘Pawpularity’ scores, the model will be able to predict this score for our test images.

First version model

The first version of our model will be a simple convolutional neural network that is able to predict the ‘Pawpularity’. First we will preprocess and combine the data, which is explained in the section below. Subsequently, we will train our model with this data. Our goal is to make a prediction that is slightly better than a random prediction.

Data analysis and preprocessing

Firstly, we checked if there were any samples that were missing tabular data, which was not the case. The distribution of the data was slightly left-skewed with the peak at 30. Noticeably was the outlier at a score of 100 (figure 1).

Secondly, we looked at the image data and saw that not all images had the same size and orientation. In order to correct this, we reshaped each of the images to images of the size 64 x 64 pixels. We also sorted the dataframe based on the order of the images so they are indexed to the right tabular data. We used this combined data in our Convolutional Neural Network model.

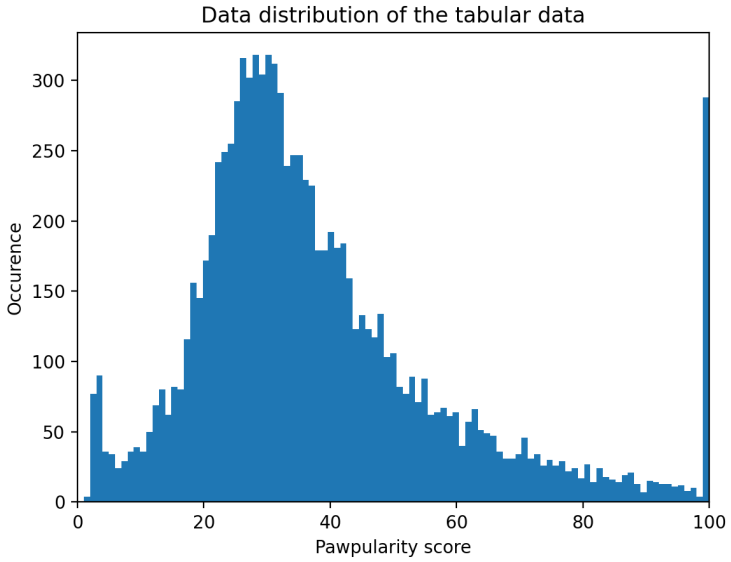

Figure 1: Histogram of the data distribution of the tabular data. On the x-axis is the Pawpularity score presented and on the y-axis the occurrence of each score. The graph is slightly left-skewed with an outlier at the score of 100.

Model Pipeline and Training

At first we started with 2 separate models, one tabular data model and another for the actual images. The input of the tabular data was a csv file with the rows representing the samples, and 12 columns that represent the features a photo could possess. If a feature was present in a photo, this feature got score 1 and if the feature was absent, a score of 0 was given. This data was used to build a Neural net with 1 dense layer of 20 nodes with a relu activation function. The input of the image model was the reshaped images (64 x 64) and has the model type Convolutional neural network with a regression output and 1 hidden layer of 20 nodes. We added 2 convolutional layers, where the first one has 64 filters and the second one 128. The kernel size for both is 3,3 with a relu activation function. After each of these layers we added a max pooling layer with a pool size of 2,2 and a stride of 2. After a flattening layer we added a dense layer where the units are equal to the amount of hidden nodes. This layer has a relu activation as well.

Then, we merged these models using the concatenate function of the tensorflow.keras library. The model has the tabular neural network and the convolutional neural networks as inputs and one hidden layer with 20 hidden nodes and a relu activation function. The output of the model used a linear activation function. We used 20 epochs to train the model with a split of 90% training data and 10% validation data. An overview of the model with all the layers is shown in figure 2.

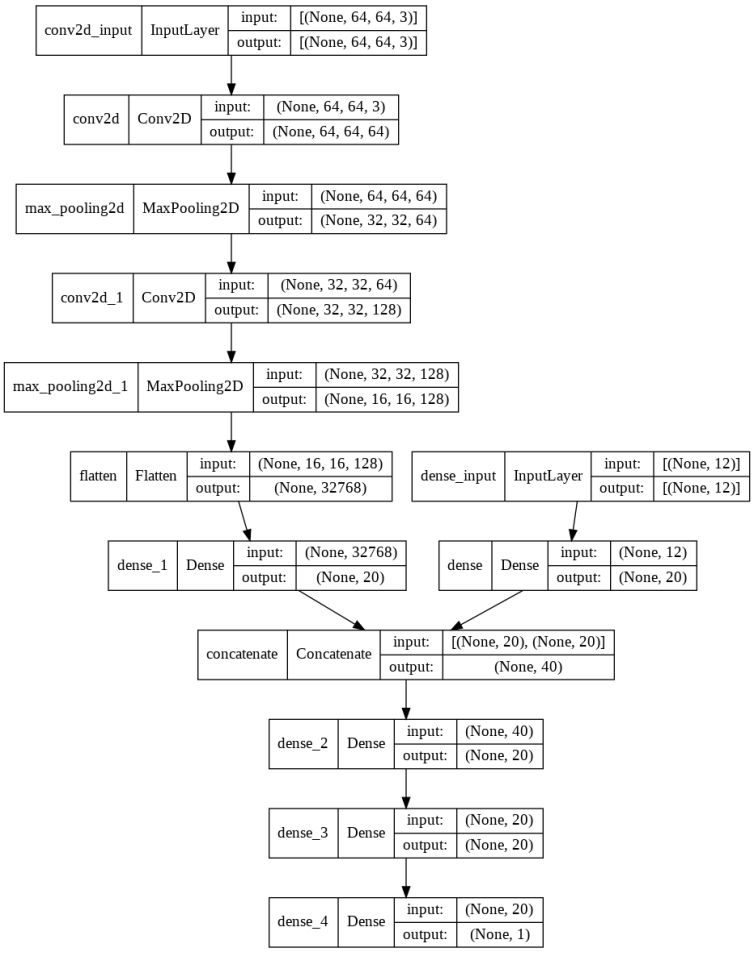

Figure 2: Overview of the layers in the convolutional network for images (left), neural network for tabular data (right) and concatenated network (below).

Evaluation and conclusions

In this first version of our model we just scaled the images and used a linear activation function for the output layer. Figure 3 shows the training loss in blue and the validation loss in orange, while the model is being trained. The final training loss is 381.4533 and the final validation loss is 429.6325.

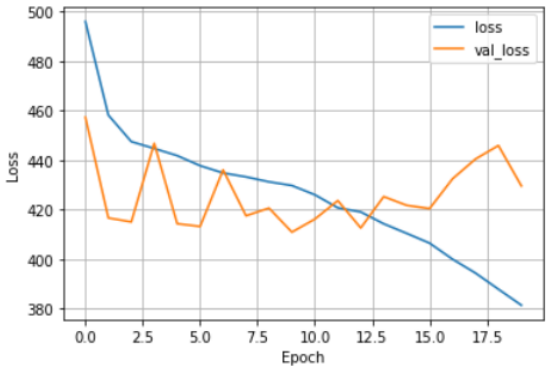

Figure 3: Graph showing the training loss (blue) and validation loss (orange) of the concatenated model on each epoch during training the model.

The model does seem to overfit on the training data, since the training loss is decreasing, while the validation loss is increasing. This means that the model is fitting the training data too specifically, so it does not generalize well to new examples.

In later versions, we can try to add preprocessing methods to prevent the model from overfitting. There are multiple methods that can prevent overfitting, such as input normalization, batch normalization and adding dropout layers. Moreover, we now used a linear activation function for the final output. This means that the output could have any number. The possible output, however, ranges from 0 to 100. In later versions we could set a minimal and maximal output threshold of 0 and 100, respectively. Furthermore, we now only show the loss of the model. This is however hard to interpret without other values to compare the loss to. The root squared mean error could be used to calculate the accuracy of the model and this should be done in following versions.