Report of Iterative Development

Machine Learning Project

Brian Jakobs, Vera Eising, Joram Brokkelkamp, Damla Baspinar

Minor AI

Teachers: Tim Doolan, Wouter Vrielink
Supervisor: Tim Doolan

Date: the 18th of January, 2022

Contents

1. Milestone I: First Model

1.1 Introduction

Sheltered cats and dogs display specific characteristics which can be considered determinants of adoption[1]. Since animals that are not adopted can be subject to suffering in different forms[2], it seems logical for shelters to attempt to focus their resources on determinants with an increasing effect on the likelihood that a given animal will be adopted[1]. Petfinder.my, a Malaysian platform regarding animal welfare, operates using a similar concept[3]. The organization uses artificial intelligence to determine a score for a Cuteness Meter, which corresponds to the attractiveness of pictures of cats and dogs. A higher score correlates to a more attractive picture. This score can be used to improve photo-quality in order to optimize the chances of adoption. The current algorithm does not consistently function adequately as it is still under development[4]. The model described in this report seeks to provide an improved version of this algorithm. The aim of this report is to elucidate the process of formation of said model.

The first version of the model in question is a convolutional neural network which takes images of animals and designated attractiveness scores as inputs and performs a number of computations to estimate the score of attractiveness for another given image.The rationale behind this approach is that deep learning is known to be successful in tasks that require image recognition[5]. The goal regarding the performance of this model, is to achieve a model that shows a decrease in its loss over a number of epochs. This shows the designed model has the capacity to be trained.

1.2 Data Analysis and Preprocessing

The data provided for every sample is composed of the images of a set of pets and additional tabular data regarding every given image. The tabular data states whether the following composition parameters were present or absent in a given image: "*Focus (Pet stands out against uncluttered background, not too close / far), Eyes (Both eyes are facing front or near-front, with at least 1 eye / pupil decently clear), Face (Decently clear face, facing front or near-front), Near (Single pet taking up significant portion of photo (roughly over 50% of photo width or height), Action (Pet in the middle of an action (e.g., jumping)), Accessory (Accompanying physical or digital accessory / prop (i.e. toy, digital sticker), excluding collar and leash), Group (More than 1 pet in the photo), Collage (Digitally-retouched photo (i.e. with digital photo frame, combination of multiple photos), Human (Human in the photo), Occlusion (Specific undesirable objects blocking part of the pet (i.e. human, cage or fence). Note that not all blocking objects are considered occlusion), Info (Custom-added text or labels (i.e. pet name, description)), Blur (Noticeably out of focus or noisy, especially for the pet's eyes and face. For Blur entries, "Eyes" column is always set to 0)*"[6]. Additionally, the tabular data also includes the "*Pawpularity*", which reflects each pet profile's page view statistics[6]. It was noted that this dataset only contains 9912 samples, which is a relatively small amount for training.

The tabular data was analyzed using the *Normal Equation*. It was established that the relation between the features and the outcome "*Pawpularity*" was not linear. Furthermore, it was established that the tabular data does not contain any missing values.

The data used for this version of the model comprises two parts: the "*Pawpularity*" within the tabular data and the images of pets.

Regarding the images, every image was resized to a shape of 128x128x3.All provided testing data had this specific shape, whereas there the shapes of the samples in the training data differed. Therefore, it was decided to resize all training data to the same shape as testing data. Additionally, these relatively large images allow training with a large amount of information.
Furthermore, the data was split into training and testing data, corresponding with 67% and 33% of all data respectively.

No further preprocessing or augmentation was applied to the images.

1.3 Model Pipeline and Training

The current predictive model is a convolutional neural network. The input of a given sample is the image of a pet, whereas the output is the "*Pawpularity*" score.
The architecture of the convolutional neural network consists of an input layer, three hidden layers and an output layer.
With the exception of the last layer, a ReLu activation function was used in every layer. The output layer provides one output. No activation function was applied in this layer. Given that the task at hand is a regression problem, no non-linear activation function should be necessary in the last layer.
Further details regarding the model pipeline are presented in Figure 1.

Model: Mile stone I

Layer (type)	Output Shape	Params
Convolution (2D)	(128, 128, 3)	1792
Max pooling (2D)	(64, 64, 64)	0
Flatten	(262144)	0
Dense 1	(512)	134218240
Dropout (0.1)	(512)	0
Dense 2	(256)	131328
Dense 3	(128)	32869
Dense 4	(1)	129

Total params: 134,384,385
Trainable params: 134,384,385
Non-trainable params: 0

Figure 1: Details regarding the model pipeline

Concerning the training of the model, the Mean Squared Error was used as the loss function for optimization. Additionally, the Root Mean Squared Error was used as a metric to judge the performance of the model, since this is customary in regression problems such as the one at hand. The optimizer Adam was applied. The model was trained using 20 epochs.

1.4 Evaluation and Conclusions

Regarding the model loss, there is a clear decrease for the training data (Figure 2). Although this decrease seems to stagnate after the first epoch in Figure 2, it is actually consistent over the course of 20 epochs. The loss for the validation loss also consistently decreases through 20 epochs.
A similar trend was observed in the root mean squared error; the error decreases over the course of 20 epochs in both the training and the validation data, though this is not clearly visible in the graph of the learning curve (Figure 2).
The decreases in both the model loss as well as the root mean squared error are a manifestation of the trainability of the model, thus the goal of forming a model with the capacity to be trained was obtained.

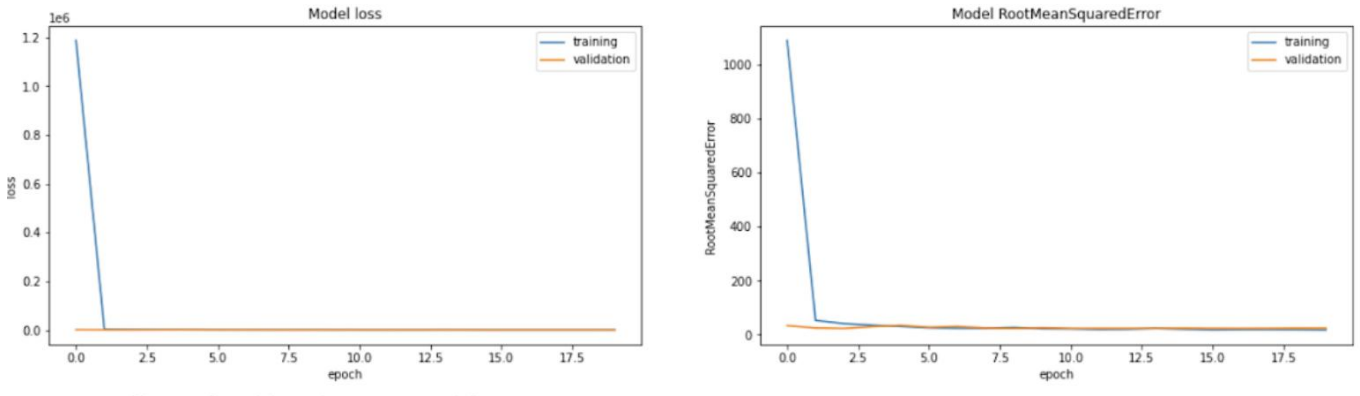

Figure 2: This figure shows the graphs of convergence of the model over 20 epochs.

Figure 2 shows that the model loss and root mean squared error were relatively high after the first epoch. Because of this the relatively small nuances between the training and the validation values are not visible in the learning curves. In order to clarify this image, the y-axis for the model loss was limited to a value of 10000 and the y-axis for the root mean squared error was limited to 100 for the model loss and root mean squared error respectively. The outcome is illustrated in Figure 3.

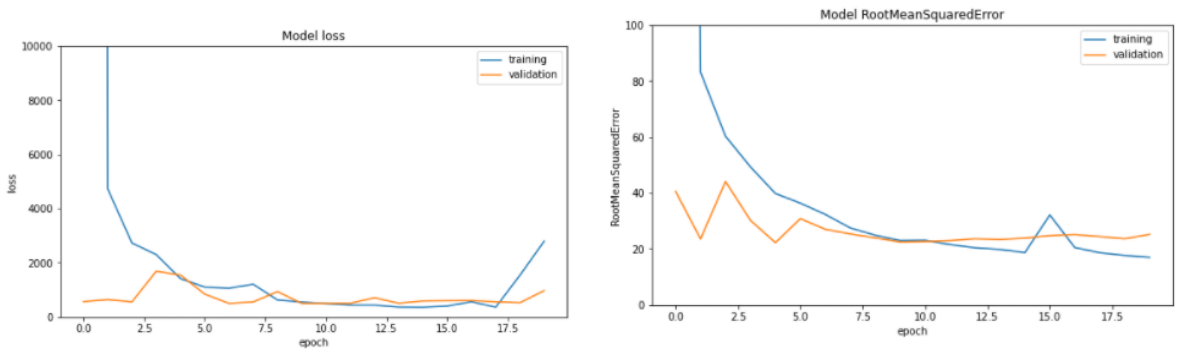

Figure 3: This figure shows the graphs of convergence of the model over 20 epochs with limited y-axes.

In Figure 3, the graph of the model loss shows that after approximately 17 epochs, the model loss and root mean square error both start to increase for the training data. This is a manifestation of divergence rather than convergence in gradient descent. This could be caused by a learning rate that is too high.

Given the small amount of available data,the application of k-fold cross validation and data augmentation could improve the training process of the model in future versions. Additionally, the use of preprocessing and augmentation methods in the images could derive better results. In addition, the increasing difference between the training loss and the validation loss during the last epochs seems to imply overfitting. Lastly, the problem of divergence should be solved.

2. Milestone II: Early stopping

2.1 Introduction

In the previous version of the predictive model, the learning curves appeared to be quite similar for the training and validation data up until epoch 16. After this epoch, the model loss steeply moves upwards (see Figure 3). This problem will be addressed in this version of the model by applying the method of early stopping. Early stopping is traditionally used to prevent overfitting[7]. However, it seems that it could also be a way to improve the current learning curves.

2.2 Data Analysis and Preprocessing

No adaptations regarding data analysis or preprocessing were made in the current version of the model. See section 1.2 for a description of the current situation.

2.3 Model Pipeline and Training

In this version of the model, only 16 epochs were used to train the model. No further improvements were made with regard to the model pipeline.

2.4 Evaluation and Conclusions

In this version of the model,the model loss no longer shows a lift towards the last epochs. Both the model loss as well as the model root mean squared error now seem to be on a similar trend downwards. 16 epochs seems to be the optimal number of epochs for the model in its current state.

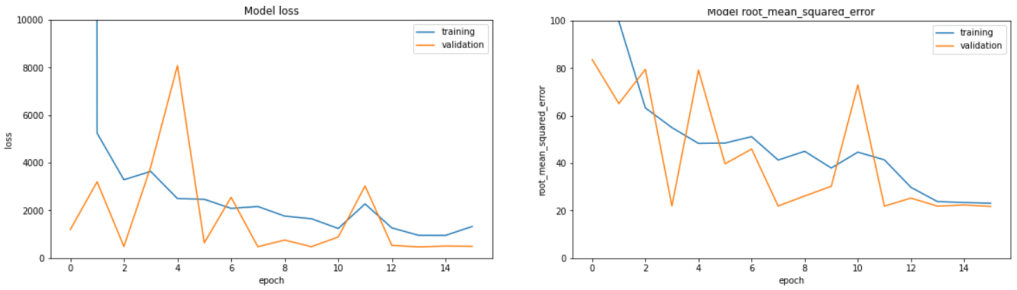

Figure 4: This figure shows the graphs of convergence of the model over 16 epochs with limited y-axes.

The learning curves both seem to show a stagnation at the last epochs rather than a decrease in the model loss or the model root mean squared error. It seems as if the model lacks complexity to fit the data better.

3. Milestone II: A Deeper Convolutional Neural Network

3.1 Introduction

The previous version of the convolutional neural network consisted of a single Conv2D layer. Adding more of such layers should correspond to adding complexity to the predictive model. This could enable the predictive model to learn more high level features. Which illustrates the rationale for making a deeper neural network for the current version of the model. This could enhance the performance of the model, which should manifest itself in a decreased loss and mean squared error in the learning curves at the last epochs, compared to the previous version of the model.

3.2 Data Analysis and Preprocessing

In this version of the model, no modifications were made concerning data analysis or preprocessing. See section 1.2 for the most recent updates regarding this matter.

3.3 Model Pipeline and Training

The convolutional model was expanded with three additional Conv2D layers. Figure 5 provides additional details regarding the model pipeline.

Model: Mile stone II

Layer (type)	Output Shape	Params
Convolution 1 (2D)	(128, 128, 3)	1792
Max pooling 1 (2D)	(64, 64, 64)	0
Convolution 2 (2D)	(64, 64, 128)	73856
Max pooling 2 (2D)	(32, 32, 128)	0
Convolution 3 (2D)	(32, 32, 256)	295168
Max pooling 3 (2D)	(16, 16, 256)	0
Convolution 4 (2D)	(16, 16, 512)	1180160
Max pooling 4 (2D)	(8, 8, 512)	0
Flatten	(32768)	0
Dense 1	(512)	16777728
Dropout (0.1)	(512)	0
Dense 2	(256)	131328
Dense 3	(128)	32869
Dense 4	(1)	129

Total params: 18,493,057
Trainable params: 18,493,057
Non-trainable params: 0

Figure 5: Details regarding the model pipeline

3.4 Evaluation and Conclusions

In this version of the model, the training loss around the last epoch is approximately 200. In the earlier version this value used to be approximately 2000. There is a clear decrease in model loss between the respective models, thus the model seems to be trained better in its more complex form. The trade-off is that the model now appears to be overfitting, given that the training loss still seems to follow a decreasing trend, whereas the validation loss has stagnated, starting at approximately 6 epochs.
The situation is quite similar concerning the model mean squared error; the error for this version is lower compared to earlier versions, but the increasing difference between the values for the training data and the validation data imply overfitting (see Figure 6).

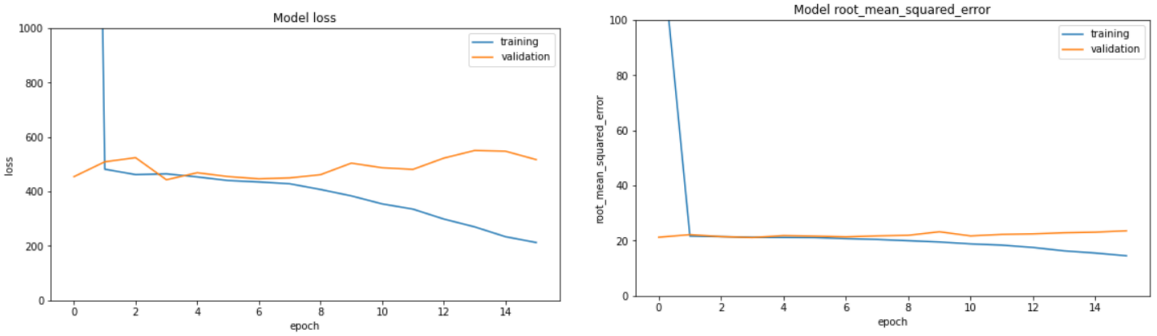

Figure 6: This figure shows the graphs of convergence of the model over 16 epochs with limited y-axes.

4. Milestone II: Normalization

4.1 Introduction

The ReLu activation function was chosen for the hidden layers of the predictive model to avoid problems such as vanishing gradient during backpropagation in training[8]. As the ReLu function is only non-linear around the coordinate (0,0) and non-linearity is a necessity for a model to learn complex decision boundaries[9], the input of the model should be normalized to have a mean of zero and standard deviation of one. In this version of the model, normalization will be applied to the input samples. More efficient training should expectedly manifest itself in earlier conversion to a local optimum during gradient descent, by showing that learning curves stagnate earlier compared to earlier versions of the model.

4.2 Data Analysis and Preprocessing

Referring back to section 1.2, the only update with regard to the data is the application of normalization to the image features. The pixels were normalized to have a mean value of zero and a standard deviation of one. In this way, the pixels of every sample (i.e. image) were normalized towards zero.

4.3 Model Pipeline and Training

In this version, the model pipeline remains unchanged to the pipeline described in section 3.3.

4.4 Evaluation and Conclusions

Normalization was applied under the assumption that it would make training more efficient. In the prior version of the model, the learning curves stagnated after approximately 8 epochs. In this version, this event occurs after approximately 1 epoch (Figure 7). It seems that the training process of the model has indeed become more efficient. Simultaneously, overfitting has become more visible.

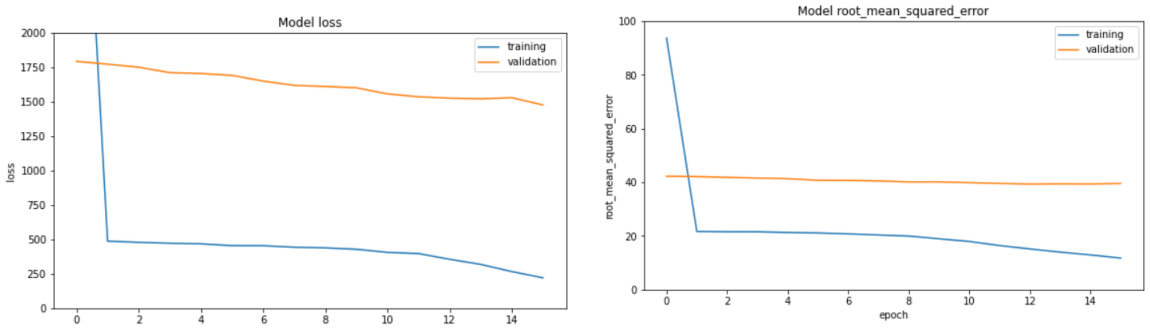

Figure 7: This figure illustrates the model loss and model root mean squared error after the input of the model was normalized.

5. Milestone II: Dropout

5.1 Introduction

A problem regarding the earlier versions of the problem is overfitting. In order to address this problem, in this version, dropout will be applied to the model. Dropout is a regularization method that prevents a neural network relies too heavily on specific nodes by randomly selecting nodes and setting them to zero during training. Because of this, the neural network becomes less likely to rely on specific nodes or weights in training[10].

5.2 Data Analysis and Preprocessing

In this version of the model, no adjustments were applied in data analysis or preprocessing. See section 2.2 for the most recent modifications regarding this matter.

5.3 Model Pipeline and Training

Several dropout layers were added to the model.
Before every dense layer, a dropout layer was added with a dropout rate of 0.4, which means approximately 40% of the inputs in these layers will be randomly set to zero.
Additionally, a dropout layer with a dropout added before the last Conv2D layer, with a dropout rate of 0.1. This dropout layer has a lower dropout rate compared to the dropout layers before the dense layers in order to preserve the information regarding high-level image features earlier in the convolutional neural network.
Figure 8 presents a summary of the pipeline concerning the current model.

Model: Mile stone II, dropout

Layer (type)	Output Shape	Params
Convolution 1 (2D)	(128, 128, 3)	1792
Max pooling 1 (2D)	(64, 64, 64)	0
Convolution 2 (2D)	(64, 64, 128)	73856
Max pooling 2 (2D)	(32, 32, 128)	0
Convolution 3 (2D)	(32, 32, 256)	295168
Max pooling 3 (2D)	(16, 16, 256)	0
Dropout 1 (0.1)	(16, 16, 256)	0
Convolution 4 (2D)	(16, 16, 512)	1180160
Max pooling 4 (2D)	(8, 8, 512)	0
Flatten	(32768)	0
Dropout 2 (0.4)	(512)	0
Dense 1	(512)	16777728
Dropout 3 (0.4)	(512)	0
Dense 2	(256)	131328
Dropout 4 (0.4)	(512)	0
Dense 3	(128)	32869
Dropout 5 (0.4)	(512)	0
Dense 4	(1)	129

Total params: 18,493,057
Trainable params: 18,493,057
Non-trainable params: 0

Figure 8: Details regarding the model pipeline

5.4 Evaluation and Conclusions

Figure 9 shows that there is a much smaller difference between the training and validation in both model loss as well as model root mean squared error, compared to prior versions of this convolutional neural network. This implies the model is no longer overfitting.

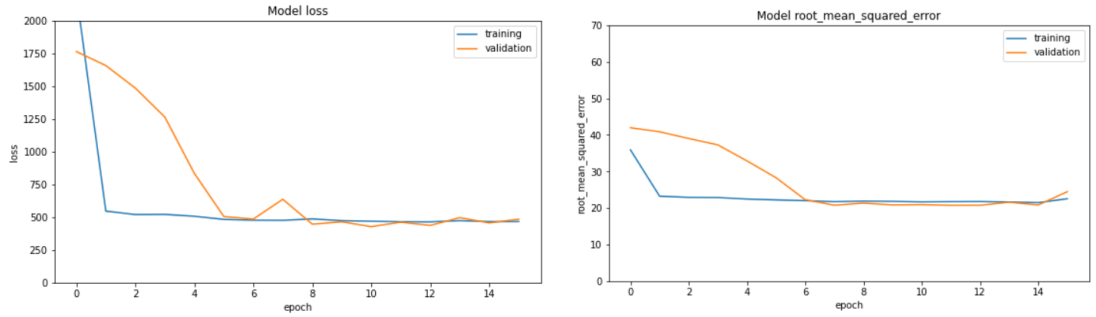

Figure 9: This figure illustrates the model loss and model root mean squared error after the application of dropout.

6. Milestone II: Batch Normalization

6.1 Introduction

In the previous versions of the model the input nodes of the model were normalized to have a mean and standard deviation of zero and one respectively. Considering the depth of the convolutional neural network. it seems appropriate to additionally normalize the values of all hidden layers. This will be done by applying batch normalization[10]. Batch normalization reduces the problem of input values changing in between layers and makes the training process of a neural network more efficient[11].

6.2 Data Analysis and Preprocessing

No further improvements regarding data analysis or preprocessing were made in this version of the model.

6.3 Model Pipeline and Training

In the current version of the model, a batch normalization layer was added after every maxpool layer. For details regarding the model pipeline. see Figure 10.

Model: Mile stone II, batch normalization

Layer (type)	Output Shape	Params
Convolution 1 (2D)	(128, 128, 3)	1792
Max pooling 1 (2D)	(64, 64, 64)	0
Convolution 2 (2D)	(64, 64, 128)	73856
Max pooling 2 (2D)	(32, 32, 128)	0
batch normalization 1	(32, 32, 128)	512
Convolution 3 (2D)	(32, 32, 256)	295168
Max pooling 3 (2D)	(16, 16, 256)	0
batch normalization 2	(16, 16, 256)	1024
Convolution 4 (2D)	(16, 16, 512)	1180160
Max pooling 4 (2D)	(8, 8, 512)	0
batch normalization 3	(8, 8, 512)	2048
Flatten	(32768)	0
Dropout 1 (0.4)	(32768)	0
Dense 1	(512)	16777728
Dropout 2 (0.4)	(512)	0
Dense 2	(256)	131328
Dropout 3 (0.4)	(256)	0
Dense 3	(128)	32869
Dropout 4 (0.4)	(128)	0
Dense 4	(1)	129

Total params: 18,496,641
Trainable params: 18,494,849
Non-trainable params: 1,792

Figure 10: Details regarding the model pipeline

6.4 Evaluation and Conclusions

Figure 11 shows that the graphs for the model loss and model root mean squared error are now very similar to one another. Especially noteworthy, is that the course of the validation cost is less smooth compared to prior versions. Additionally, the model seems to overfitting again, as both the model loss as well as the model root mean squared error continue to decrease for the training data, whereas they stagnate for the validation data, starting after approximately 10 epochs.

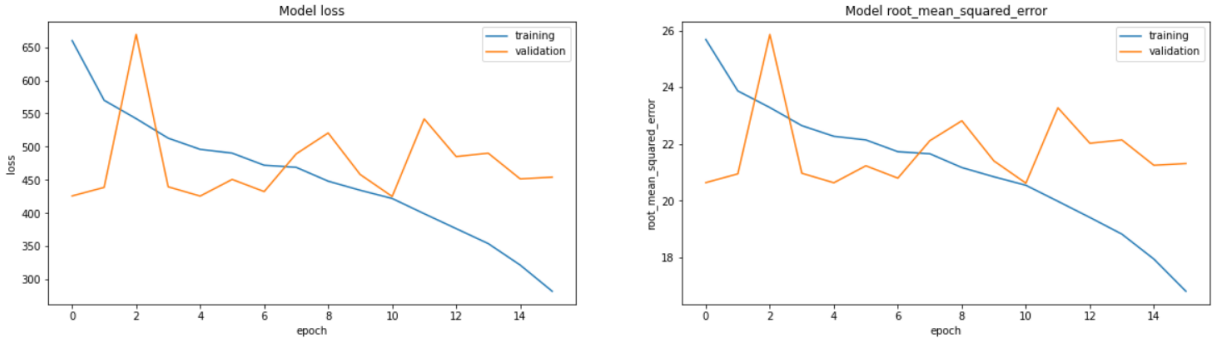

Figure 11: This figure illustrates the model loss and model root mean squared error after the application of batch normalization.

7. Milestone II: Additional Dropout

7.1 Introduction

Given that the previous version of the model seemed to be overfitting again. it makes sense to apply a regularization method at this stage of the model. For this reason, dropout will be applied again in this chapter.

7.2 Data Analysis and Preprocessing

No additional improvements regarding data analysis or processing were made in this version of the model.

7.3 Model Pipeline and Training

In this version, a dropout layer was added before every Conv2D layer. In accordance with earlier insights (section 5.3), these dropout layers have a relatively low dropout rate of 0.1 to preserve the more robust data earlier in the model.

Model: Mile stone II, batch normalization

Layer (type)	Output Shape	Params
Convolution 1 (2D)	(128, 128, 3)	1792
Max pooling 1 (2D)	(64, 64, 64)	0
Dropout 1 (0.2)	(512)	0
Convolution 2 (2D)	(64, 64, 128)	73856
Max pooling 2 (2D)	(32, 32, 128)	0
batch normalization 1	(32, 32, 128)	512
dropout 2 (0.2)	(32, 32, 128)	0
Convolution 3 (2D)	(32, 32, 256)	295168
Max pooling 3 (2D)	(16, 16, 256)	0
batch normalization 2	(16, 16, 256)	1024
dropout 3 (0.2)	(16, 16, 256)	0
Convolution 4 (2D)	(16, 16, 512)	1180160
Max pooling 4 (2D)	(8, 8, 512)	0
batch normalization 3	(8, 8, 512)	2048
dropout 4 (0.2)	(8, 8, 512)	0
Flatten	(32768)	0
Dropout 5 (0.4)	(32768)	0
Dense 1	(512)	16777728
Dropout 6 (0.4)	(512)	0
Dense 2	(256)	131328
Dropout 7 (0.4)	(256)	0
Dense 3	(128)	32869
Dropout 8 (0.4)	(128)	0
Dense 4	(1)	129

Total params: 18,496,641
Trainable params: 18,494,849
Non-trainable params: 1,792

Figure 12: Details regarding the model pipeline

7.4 Evaluation and Conclusions

In accordance with the learning curves in Figure 13, it is uncertain whether additional dropout solves the problem of overfitting in this model. The validation loss and root mean squared error both seem to follow a decreasing path until they reach a peak at the last epoch. With

this number of epochs, the learning curves cannot show whether the decreasing trend will continue or halt after 16 epochs.

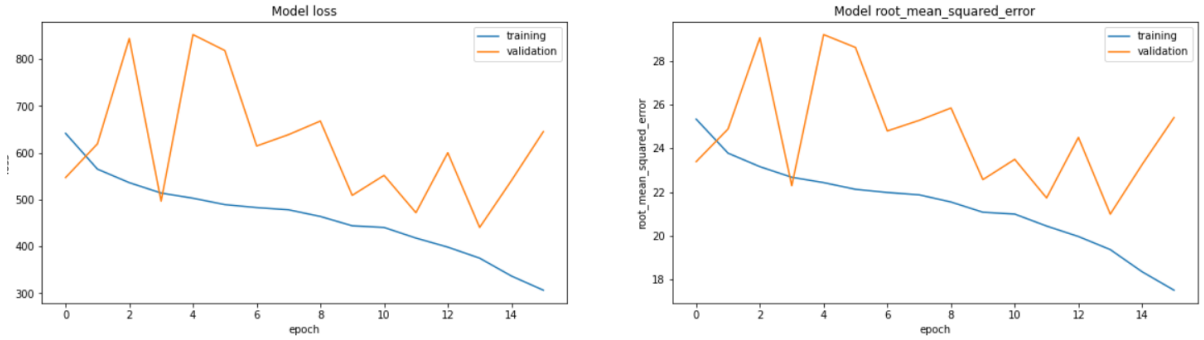

Figure 13: This figure illustrates the model loss and model root mean squared error after applying additional dropout

8. Milestone II: Adding Epochs

8.1 Introduction

In the previous version of the model, it was unclear whether applying additional dropout in the model was successful in preventing overfitting. In order to make this more visible, a version of the model with more epochs will be tested in this chapter. The trend in the validation cost should show whether the model is still overfitting.

8.2 Data Analysis and Preprocessing

No additional improvements regarding data analysis or processing were made in this version of the model.

8.3 Model Pipeline and Training

In this version of the model, the training process was completed over the course of 50 epochs.

8.4 Evaluation and Conclusions

After adding epochs, the trend in the validation data clearly shows that the model is still overfitting after applying additional dropout. Figure 11 in section 6.4 shows a steep decrease in training loss, whereas the validation loss seems quite stable around an approximate value of 500. The current model with additional epochs shows that additional dropout caused an increase in validation loss. Thus, it has become evident that additional dropout does not have a value addition to the development of this model. Therefore, additional dropout will not be applied in future versions of the model.

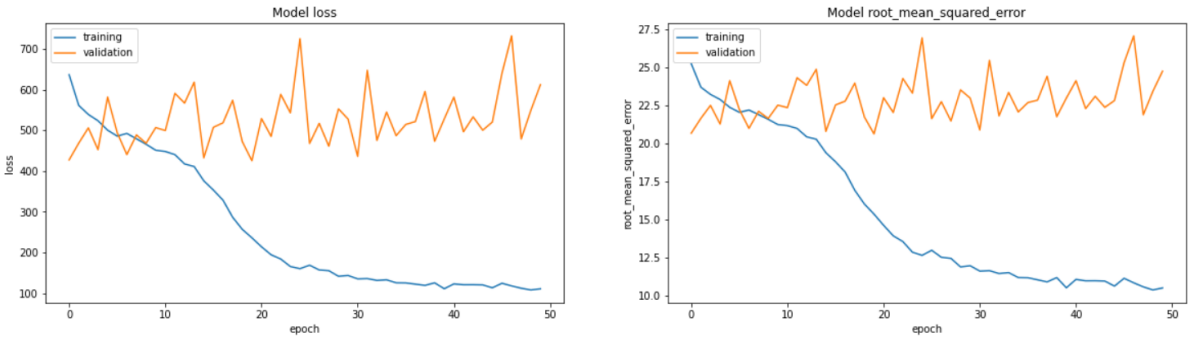

Figure 14: This figure illustrates the model loss and model root mean squared error after training the model with 50 epochs.

References

1. Lepper M, Kass PH, Hart LA. Prediction of adoption versus euthanasia among dogs and cats in a California animal shelter. J Appl Anim Welf Sci. 2002;5(1):29-42.
2. PETA. The Deadly Consequences of 'No-Kill' Policies. [Internet]. Available from: https://www.peta.org/features/deadly-consequences-no-kill-policies/. [Accessed at 11-01-2021].
3. Petfinder.my. About Petfinder.my. [Internet]. Available from: https://www.petfinder.my/about.htm. [Accessed at 11-01-2021].
4. Petfinder.my. Cuteness Meter — How Attractive Are Your Photos?. [Internet]. Available from: https://www.petfinder.my/campaigns/cutenessmeter.htm. [Accessed at 11-01-2021].
5. Molnar C. Interpretable Machine Learning. Lulu.com; 2020.
6. Kaggle. PetFinder.my - Pawpularity Contest: Data. [Internet]. Available from: https://www.kaggle.com/c/petfinder-pawpularity-score/data. [Accessed at 12-01-2021].
7. Machine Learning Mastery. Use Early Stopping to Halt the Training of Neural Networks At the Right Time. [Internet]. Available from: https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/. Accessed at 17-01-2021].
8. towards data science. The Vanishing Gradient Problem. [Internet]. Available from: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484. [Accessed at 16-01-2021].
9. DeepLearningAI. Why non-linear activation functions. [Video]. 2017. Available from: https://youtu.be/NkOv_k7r6no. [Accessed at 16-01-2021].
10. DeepLearningAI. Dropout regularization. [Video]. 2017. Available from: https://youtu.be/D8PJAL-MZv8. [Accessed at 16-01-2021].
11. DeepLearningAI. Normalizing activations in a network. [Video]. 2017. Available from: https://youtu.be/tNIpEZLv_eg. [Accessed at 16-01-2021].