Deep Learning for the PetFinder Adoption Prediction Problem

Christopher Hittner and Weronika Zamlynny

May 2019

Abstract

The PetFinder.my Adoption Prediction Kaggle competition, poses the challenge of predicting how quickly an animal will be adopted given a large variety of data associated with an animal's profile page on PetFinder's website. The data provided includes numerical data (both categorical and continuous), image data, text data, as well as additional preprocessed json metadata. This allows us to use a variety of methods, such as feed-forward classification methods, natural language processing, computer vision, and dimensionality reduction.

1 Introduction

Adoption Prediction as it is given by the Kaggle competition is a Multi-class Classification problem. AdoptionSpeed, the variable being predicted, can be one of five values: 0 - animal was adopted the same day as it was listed, 1 - animal was adopted between 1 and 7 days, 2 - animal was adopted between 8 and 30 days, 3 - animal was adopted between 31 and 90 days, 4 - no adoption after 100 days. PetFinder notes that this dataset did not include any animal adopted between 90-100 days. The number of days to adoption is not provided.

It is good to note that although this data is provided as categorical, this problem could also be looked at as a regression problem. Each number has a relative meaning compared to the numbers around it. The categories do not have a linear relationship between them, but a mapping could be used to convert this to a regression problem.

1.1 Train and Test Data

Kaggle provided a separate train dataset with 14993 data points and a test dataset with 3948 data points. Only the training dataset includes y labels. As a result, to optimize models, a validation set was defined on a random 20% of the train data. The test dataset was only used to generate the Kaggle submission.

1.2 Data Columns

The provided data can be split into numerical, categorical, text, and image data.

The numerical columns are: age (in months), quantity (a profile may include more than one animal), fee (the cost of adoption), video amount, and photo amount.

The categorical columns are: type (cat or dog), breed 1 and breed 2 (307 breed labels provided in addition to no breed) - two breed labels are provided if the animal has multiple breeds, gender (male, female, or mixed if there are multiple animals), color 1-2-3 (7 color labels are provided, each animal may have up to 3 colors listed, maturity size (how large the animal will be at maturity on a scale 0-4, could potentially be converted to numerical), fur length (scale of 0-4, could potentially be converted to numerical), vaccinated (whether or not the animal is up to date with its vaccinations, or unknown), dewormed (yes/no/unknown), sterilized (yes/no/unknown), health condition (scale 0-3), and state (which state in Malaysia the post is from, 15 labels provided). Each categorical column was one hot encoded before use.

The text columns are: name, rescuer id, and description. Additional preprocessed metadata from Google's Natural Language API was provided for the description field. This included an overall sentiment which was

extracted and used as an additional numerical parameter. In addition, the number of sentences in the description was also extracted as an additional numerical parameter.

Each animal also had images associated with it's ID. Some pets had 0 images, but others had many available images. The quality and contents of the images varied greatly from pet to pet. Each image also had additional metadata from Google's Vision API.

1.2.1 Breeds and Dimensionality Reduction

The breed of an animal was an important data column, but due to it being a categorical data point with 309 labels, it made it difficult to include into any model. Its large size would cause the number of parameters to explode and also it would out weight all the other parameters in the model. With both breed columns one hot encoded, there would be 614 columns added to the dataset.

As a result, breed became a perfect candidate for dimensionality reduction. Below are graphs of the distribution of breeds for dogs and for cats - the labels are distinct with 241 dog breeds and 66 cat breeds. The graphs are cut off at 50 for dogs, and 20 for cats to make the graphs easier to view and the tails have few values. It is easy to see that the distributions are heavily skewed, with a few breeds being significantly more common than the rest. Since breeds are not equal between dogs and cats, it is important to be careful when reducing dimensionality or all cat breeds would be removed.

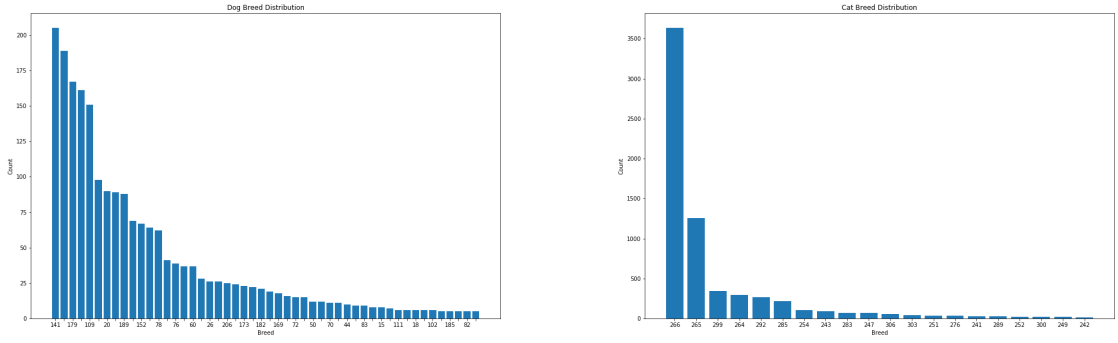

Figure 1: Breed Distributions of Dogs and Cats

The simplest method to reduce the dimensionality could be to simply take the top 10 breeds for dogs and for cats. Another option is doing Principal Component Analysis (PCA) as covered in class. PCA was run to reduce the dimensionality of both dog and cat breeds to 64, (limited by the number of cat breeds). Smaller increments such as 32, 16, and 10 were tested, but 64 was found to have the best results.

1.3 Quadratic Weighted Kappa

Kaggle uses the quadratic weighted kappa to score submissions for the competition. This scoring metric penalizes random guesses, by applying a larger penalty the further a guess is from the correct value. Since the y values for this problem are in a range of [0, 4], a guess of 0 on a 3 will be more penalized than a guess of 2 on a 3.

$$\kappa = \frac{p_{correct} - p_{guess}}{1 - p_{guess}}$$

$$\kappa_w = 1 - \frac{\sum_{i,j} w_{ij} p_{ij}}{\sum_{i,j} w_{ij} e_{ij}}$$

2 Related Works

2.1 Classification Using Neural Networks

In class, we learned about the use of neural networks for the purpose of performing classification. A notable instance was the use of the Cats Versus Dogs dataset to train a convolutional neural network to distinguish between cats and dogs. In general, neural networks have been used to achieve extremely effective classification and regression of various subject material. Here, we implement several types of classifiers to process several forms of input data to determine the adoption time.

3 Methods

3.1 Baselines

Baseline metrics were run on the numerical data only, including the one-hot-encoded categorical data. Table 1 shows the accuracies and kappa scores of various baseline methods. Throughout the project, it has been a challenge to improve on the baselines, especially when certain models tended to always guess category 4 with the highest probability.

Method	Accuracy	Kappa Score
Random Guess	0.19439	-0.00309
Guess 0	0.02754	0.0
Guess 1	0.20454	0.0
Guess 2	0.26936	0.0
Guess 3	0.21702	0.0
Guess 4	0.28151	0.0
Linear Regression		
Training	0.26304	0.19863
Validation	0.26713	0.19384
Decision Tree		
Training	0.98618	0.98601
Validation	0.34307	0.20123

Table 1: Baseline Accuracies

Both linear regression and the decision tree were trained and run using available models in sklearn. The decision tree gives a surprisingly good accuracy and a better quadratic weighted kappa compared to many deep learning models. For this problem, it may be worth to look more into methods using decision trees to attain greater accuracies and scores. In Figure 2, are the first three layers of the constructed decision tree.

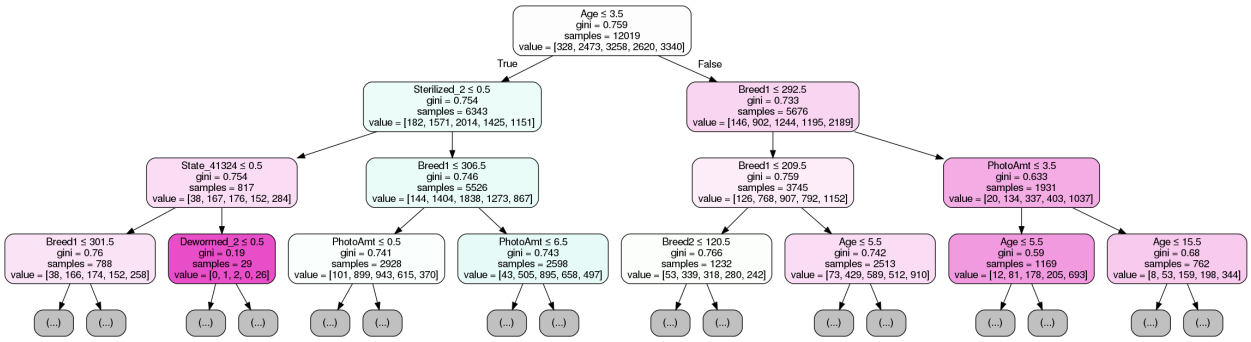

Figure 2: Decision Tree

3.2 Multilayer Perceptron

The multilayer perceptron used in our project is a three-layer dense feedforward network where the multiclass labels are initially embedded into multidimensional vectors. The embedding inputs are used to provide a representation of the inputs while removing any notions of ordering. For instance, the breed label is converted from one of the 307 labels to a 18 dimensional vector. After converting these labels to vectors, they are concatenated with each other, as well as with the numerical attributes and passed through the three feedforward labels.

Of the convolutional models, this method proved to be the most effective. We trained the model using the Adam optimizer with a learning rate of 2e-4 for 96 iterations. During training, the model was able to achieve an accuracy around 35 percent on the validation set with a kappa score of 0.11353 on the set. The kappa score on Kaggle was 0.12349.

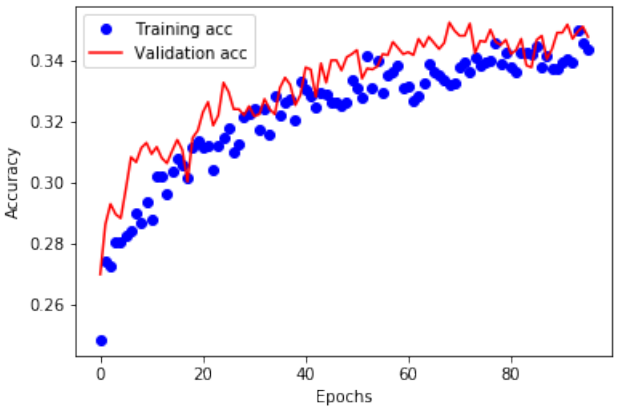

Figure 3: Fully Connected Classifier Accuracy

3.3 Single Image CNN

We also experimented with using the image set on its own. To do this, we utilized an architecture similar to ResNet to convert 64 by 64 images to one of the five categories. We trained this model using the Adam optimizer with a learning rate of 2e-4 for 96 iterations. Using this method, our model accuracy was 28.73 percent with a kappa score of 0.0397. We aimed to use this as part of a combined architecture alongside the attribute-only model. However, we encountered memory issues while using Google Colab and did not end up pursuing it.

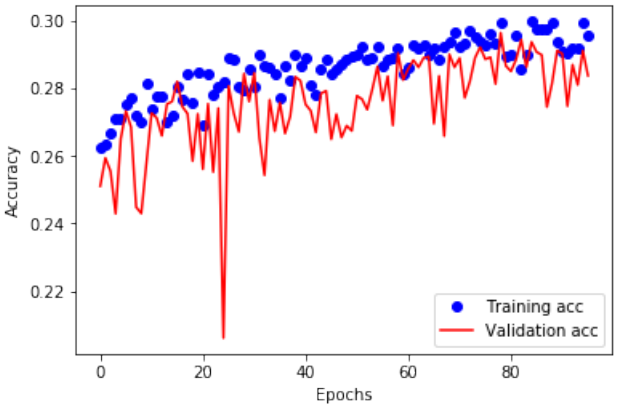

Figure 4: Fully Connected Classifier Accuracy

3.4 Numerical Model

This layer takes in only numerical data, and one-hot encoded categorical data. This is the same data which was used for the baselines. This simple numerical model uses a few Dense layers to run its classification. It achieved an accuracy of 35.24% and a kappa score of 0.24183. This was a simple model, which could then be combined with the below text and breed models.

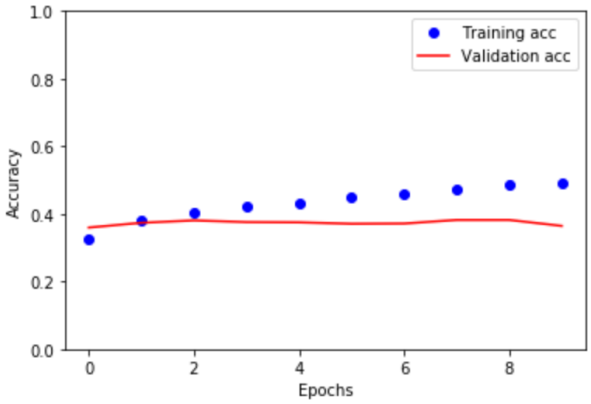

Figure 5: Numerical Model Accuracy

3.5 Text LSTM

The description field provides raw text data for natural language processing. One issue with the description field is that descriptions can be two languages: either English of Malay. Currently, nothing is done about the two languages, although information on which language is used is provided in the metadata from Google's Natural Language API. It is possible for this metadata to be helpful, although there are few Malay entries compared to English. There can also be missing descriptions, these are ignored and filled with 0's.

This model requires text processing; the description is first tokenized then encoded with a dictionary created from both the train and test descriptions. This is then aligned to the average length of a description of 30 words/tokens. This is then passed into the model, which starts with an embedding layer then three stacked LSTM layers. This text only model has a validation accuracy of 35.24% and a kappa score of 0.15724. Run on the only dogs, the accuracy was 37.43% and kappa score was 0.16306. Run on only cats, the accuracy was 30.11% and kappa score was 0.07824. Interestingly, the descriptions of dogs were much more telling than the descriptions of cats. Unfortunately, due to the distributions of dogs to cats the values average out when combined, similar to the accuracy when run together.

3.6 Breeds Model

The breeds model focused specifically on the breeds with reduced dimensionality as explained above. Post PCA breeds were passed through a few dense layers to do classification. The breeds model achieved 30.70% accuracy and a kappa score of 0.18215; this was initially run on both dog and cat breeds combined. Dogs-only achieved an accuracy of 38.74% and a kappa score of 0.05523. Cats-only achieved an accuracy of 33.16% and a kappa score of 0.01397. With the breeds it is very clear that running the dogs and cats separately is beneficial, including running PCA separately. Future models involving breeds were run separately on cats and dogs.

3.7 Combined Models

All the following combined models used the previously explained models as frozen pretrained layers, with their results concatenated. This is then run through another couple dense layers to preform classification. Combining the numerical model with the text model achieved 34.72% accuracy with a kappa score of 0.21645.

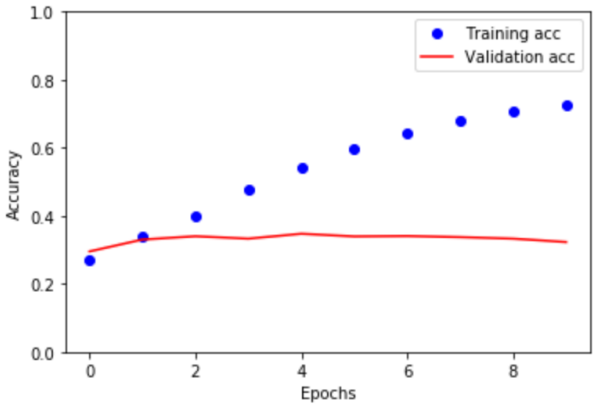

Figure 6: Text Model Accuracy

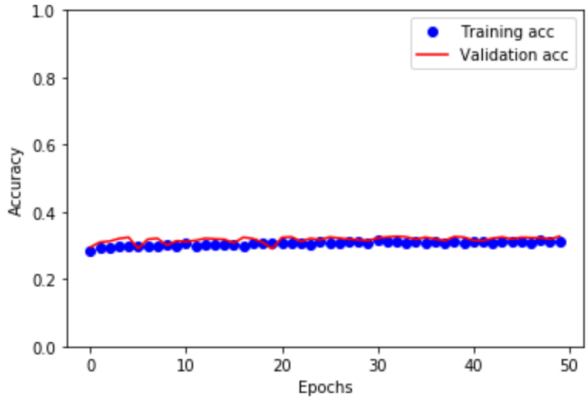

Figure 7: Breeds Model Accuracy

When running the breeds only model, it was learnt that training cats and dogs separate was beneficial to the accuracy, therefore for the combined models with breeds two separate models were trained for dogs and cats. The predictions of the models were then pasted back together to return the final result. Combining the numerical with text and breeds achieved an accuracy of 35.43% and a kappa score of 0.17875. (35.71% accuracy and 0.20272 kappa on dogs, and 35.05% accuracy and 0.14657 kappa on cats.) This combined model proved to be one of the best performing deep learning models in both accuracy and kappa score. On Kaggle, it achieved a kappa score of 0.13922, which unfortunately did not beat the decision tree baseline.

4 Results

Table 2 shows the validation accuracies and scores of all the models explained in the previous section.

Table 3 below shows the resulting Kaggle scores of the models which performed best on the validation data. These scores are calculated on a test set to which we do not have access to the labels of. The best score was received on the baseline decision tree model from sklearn. Deep learning models appeared to have difficulty getting a high score, with the highest achieved quadratic weighted kappa score of 0.13922.

5 Challenges

The highest kappa score on the Kaggle leaderboards is 0.46613. This is a clear indication that the problem is non-trivial as it is difficult to achieve a high quadratic weighted kappa score. Even using accuracy as a

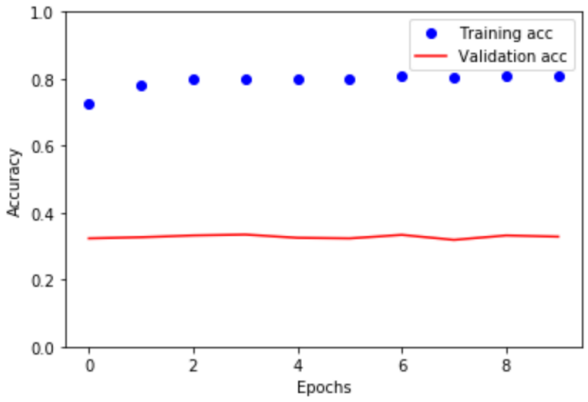

Figure 8: Numerical + Text + Breeds Model Accuracy

Method	Accuracy	Kappa Score
Multilayer Perceptron	35%	0.11353
Single Image CNN	28.73%	0.0397
Numerical	35.24%	0.24183
Dogs Only Numerical	36.45%	0.23433
Cats Only Numerical	33.54%	0.22069
Text LSTM	35.24%	0.15724
Dogs Only Text LSTM	37.43%	0.16306
Cats Only Text LSTM	30.11%	0.07824
Numerical + Text LSTM	34.72%	0.21645
Breeds	30.70%	0.18215
Dogs Only Breeds	38.74%	0.05523
Cats Only Breeds	33.16%	0.01397
Numerical + Text + Breeds	35.43%	0.17875
Dogs Only Breeds	35.71%	0.20272
Cats Only Breeds	35.05%	0.14657

Table 2: Validation Accuracies and Scores

metric, it was difficult to attain an accuracy above 35%, compared to random guessing at 20% accuracy.

One challenge that we came across was the ease with which underfitting could occur. In several cases, it was seen that the classifiers would train to output the probability distribution of the dataset. For instance, the models would always choose category 4 because more items are in that category than others. Another issue we dealt with was the lack of sanitation on the image dataset. We found that while there were a lot of images provided, some were designed in complex ways that might be non-trivial for a network to learn without overfitting. For example, one data sample used a graphic to make a collage of images of the pet, where the pet took up only a small portion of the overall image. Furthermore, the size and dimensions of each image could vary greatly, ranging from around 64 by 64 to hundreds of pixels on each dimension.

Another issue was the difference in the distributions of cats versus dogs. For example, the set of cat and dog breeds is completely disjoint, meaning that certain combinations of encodings are not possible. This is despite the use of the same breed field to encode the animal breed. By accounting for these differences, we can aim to treat cats and dogs differently, making labeling of cats a different problem from labeling dogs. Hence, an ideal attempt at the problem could aim to train two separate models: one for cats and one for dogs. In this situation, the dog-only model achieved much higher accuracy than the cat-only model.

Method	Kappa Score
Decision Tree	0.17013
Attribute-Only NN Classifier	0.12349
Single-Image CNN Classifier	0.03970
Numerical + Text + Breeds Classifier	0.13922

Table 3: Kaggle Scores

6 Future Work

In the future, we wish to explore more advanced methods to extend our models, particularly the use of ensemble learning. Since learning from attributes and NLP data has shown the most promising results, we may aim to use these as the basis for such a model. Furthermore, we wish to further explore the possibility of using CNN to enhance predictions despite said methods being ineffective for this problem. One potential method for doing this would be to pretrain the CNN on the Cats Versus Dogs dataset. The motivation would be to design a base network that can distinguish between the two in order to learn a meaningful encoding.

It is possible that neural networks are simply not the best tool for this problem, as the decision tree had the best results. It would be worthwhile to look into methods of improving the basic decision tree, for example with boosted decision trees or a random forest approach.

7 Conclusion

In our project, we explored the use of deep learning for solving the PetFinder.com Adoption Prediction problem. In doing so, we found that it did not yield outstandingly effective results. While it does hold up with the baseline models, the classification accuracy was marginally but noticeably better than random guessing. We do believe that there is potential in treating the problem as a regression problem, but we were unable to achieve effective results. We also found that the use of images in the problem proved to be less effective, working only marginally better than the 'Guess 4' model. This was expected, as the image dataset includes a large variety of image dimensions and contents varying in noise.

8 Honor Pledge

We pledge our honor that we have abided by the Stevens Honor System.

Christopher Hittner

Weronika Zamlynny