Did you know that stray animals pose significant health and environmental risks? They’re capable of attracting predators into suburban residential areas and can potentially carry diseases.

As such, many organisations have thus taken it upon themselves to adopt strays and supply them to loving families in need of companionship.

However, one of the biggest hurdles faced by these animal welfare groups is the typical lack of funding and trained staff to manage such large animal populations, so the more stray animals that find homes faster, the better.

Petfinder is one such organisation, and their current basic algorithm ranks the "Pawpularity" of pet photos for prospective adopters based on various metrics such as traffic data across web and mobile devices, but their algorithm is quite rudimentary. In this project, we will attempt to analyze the raw images and metadata of thousands of pet profiles to predict the Pawpularity of pets in an attempt to alleviate the burden on animal welfare groups and help stray dogs and cats find caring families, creating a win-win scenario.

Let’s give a brief overview of the data, which comes in two formats. Firstly, there is a set of 9912 images, and secondly, each of those images has corresponding metadata stored in "train.csv". Each raw image is a JPEG file in RGB format with a unique name called its 'Id', and can come in a variety of sizes and aspect ratios. Each image 'Id' has a corresponding row in the csv file which also holds values of certain metadata features of that image, such as 'Face' (whether or not the animal face is clear) and 'Action' (whether or not the animal is in the middle of an action like playing or jumping), as well as the image's popularity score (our desired response variable which ranges from 1 to 100.) Excluding the pawpularity score, there are 12 such features which are listed here, and each of them takes a binary value.

A constraint of the project is that submissions must be made through notebooks and the GPU runtime must not exceed 9 hours. Therefore, it is important not to overcomplicate our models and some tests that I have conducted will not be shown in the notebook and only in this presentation. The metric used to compute the model's goodness-of-fit is the Root Mean Squared Error, defined like so, where y\_i denotes the actual pawpularity score of pet profile i and y\_i\_hat denotes the predicted score of pet profile i. Just for reference, the top ranking leaderboard score is 17.65.

I decided to take a three-step approach in investigating this project. First, I conducted some exploratory data analysis of the metadata, and then experimented with some simple models to predict score from metadata alone. Finally, if the metadata proved insufficient, I would build models to predict score from raw images.

The reason we attempt simple models first is the ease of interpretability. In the future, if additional metadata features are introduced, the model can be easily retrained to incorporate the new predictor. Furthermore, the regression coefficients of linear models correspond to the pawpularity score increase associated with having that feature, so by examining the regression equation, we can easily see which features contribute most to the Pawpularity score.

As there are no missing values in the metadata, no data imputation is required. We first plot the histogram of the Pawpularity scores observed in the training dataset and observe that the distribution is right-skewed and the peak appears to be in the 20-40 score range. This suggests that our model may perform poorly if the test distribution has more datapoints in the relatively underpopulated 60-90 range.

Here is the correlation heatmap for the 12 features. Notice that no single feature appears to have good correlation with the response variable Pawpularity, and the most notable correlations are between Human/Occlusion, Info/Collage, Face/Eyes, Group/Near, and Blur/Eyes.

When conducting regression, we must check to see that multicollinearity assumptions are not violated; if they are, the regression coefficients lose their interpretability. Therefore, we calculate the Variance Inflation Factor of each predictor; which can be intuitively thought of as the proportion of variability of a predictor that can be explained by the other predictors.) In general, a VIF over 10 indicates multicollinearity and we should remove that predictor.

I observed that indeed both 'Eyes' and 'Face' predictors have VIF over 10, so I chose to remove the feature 'Eyes' since it has strong negative correlation with another feature 'Blur'. I also used principal component analysis (PCA) to determine the optimal number of features in the predictive model and found that 9-10 features can preserve over 90% of the variance, so indeed one or two features can be dropped.

We also run some exploratory plots on each of the predictors to see how they are distributed. We separated the possible pawpularity scores into 10 quantiles, and since all the predictors are indicator variables, we plotted the frequency of each predictor for each score quantile. Overall, there does not appear to be any noticeable trends of any predictor with the score value; i.e. each score band has roughly the same proportion of that predictor. Some predictors have low frequencies at every score quantile, such as 'Subject Focus', where the proportion of pets that have it ranges between 0.02 and 0.035.

We use a variety of simple models on the metadata, namely logistic-regression, which treats the problem as a 100-class classification problem, linear regression, ElasticNet regression which has L1 and L2 penalties, decision tree regressor, and random forest regressor with 50 base estimators. For evaluation, the mean 5-fold cross validation RMSE is used as an estimate of the test error.

Here are our 5-fold cross validation RMSE results. None of these are particularly leaderboard-worthy,

so we also did some additional work with the Decision Tree regressor by using grid search on the pruning parameter alpha

by sampling 50 equally spaced values in this interval POINT. We ended up finding alpha=0.1 to be the parameter value with the best 5-fold cross-validation MSE. The pruned decision tree is much easier to interpret and yielded a slight improvement in 5-fold RMSE.

As for our random forest regressor, we evaluated the importance of each of the features in the model and found the most significant features were Occlusion, Face and Near. Using backwards elimination, we dropped features in the random forest model starting with the least important ones and keeping the most important features, then found the lowest mean 10-fold CV RMSE for each subset of features and considered the smallest subset of features within 1SD of that value.

Our 5-fold cross-validation mean RMSE received a slight improvement to 20.63 with these features.POINT.

In summary, we see that logistic regression performed the poorest (without even taking into account the class imbalance, since many scores are in the 20-40 range), followed by random forest and decision tree with similar performance. ElasticNet regression and simple linear regression had the lowest 5-fold mean cross-validation RMSE, though we observe that the slight decrease in RMSE in ElasticNet regression is likely negligible. Further experiments with the random forest also showed that increasing the number of estimators for random forest did not improve the 5-fold mean RMSE by a significant amount. I also tried adaptive boosting methods such as AdaBoost and XGBoost, but the RMSE score was quite similar at 20.812, and adding further estimators would likely increase overfitting, as I demonstrated in my Titanic experiment.

So is this really the end of the line for simple models? Maybe not. The problem with our dataset is that although the pawpularity scores are discrete-valued from 1 to 100, it is more similar to a regression problem than a multiclass classification. However, the previous models we used didn’t take into account that the output range of values needs to be constrained.

This brings us to the idea of ordinal regression, which is defined as learning a classifier for ‘ordered’ classes such that the average loss over all input-output pairs is minimised. It differs from the standard multiclass classification in that the 0-1 loss is not sensitive to the distance among target values. For instance, in cross-entropy, if a picture is of a dog, it doesn't matter if we misclassify it as a spider or as a cat: either is equally wrong. But in ordinal regression, as there is an 'order' between labels, the loss function becomes lower as the between-class 'distance' decreases. If we were to apply ordinal regression, we might consider that a dog is closer to a cat than a spider, and hence the cat would be a 'less incorrect' prediction. This is what we want in our model since it is evaluated via RMSE, and the magnitude of misclassification does matter.

Hence, we attempt to fit two variants of ordinal regression to our metadata; linear ordinal ridge regression, and logistic ordinal regression. Here are the 5-fold mean RMSE scores, and we can see that the ordinal regression method shows significant improvement only for the logistic regression approach, but still performs poorly in comparison with the other models. This suggests there may be a limit to the predictive strength of the metadata towards the pawpularity score and we should turn to the raw image data. Furthermore, observing the range of predictions made by each of these models reveal extremely similar and worrying results - they tend to predict all images to have pawpularity scores around the most common 31-40 score band.

In order to extract features from raw images, I used convolutional neural networks. Instead of building an architecture from scratch, I opted to use the preexisting ResNet architecture that has been proven to perform well. Manually designed CNNs are more prone to bugs, have no theoretical evidence to support claims that they perform better than the well-known architectures, generally take more time to converge and train, and performance may be poorer, whereas ResNet models pretrained on ImageNet classification are readily available. The ResNet architecture was selected over AlexNet and VGG in particular due to its skip connections, which prevent the vanishing/exploding gradient problem that causes early layers to update extremely slowly. It does so by causing the gradient to pass through multiple layers (the Conv blocks) without any change in magnitude.

Here is a diagram of the ResNet-50 architecture with our slight modification, which added batch normalization, dropout, linear layers with an output size of 1 and a sigmoid since we want to predict the score. Dropout and batch normalization replaced the final fully connected layer to lower generalisation error rates on the test set. During training, a dropout unit zeroes the output of a given neuron randomly with probability p, and as neurons become unreliable, the network is able to learn various independent representations of the network that may have been overlooked without dropout. Batch normalization is also used to convert the batched outputs between layers into a form with zero mean and unit variance, leading to increased network speed and stability. Finally, a sigmoid unit was included at the end to map the real-valued output into a value in the interval [0,1] as the pawpularity scores [1,100] will also be scaled to this range. Alternative functions that map to [0,1] could be used here, such as hyperbolic tan or softsign function. I actually did try these alternative functions with all other parameters kept the same and found that the difference in RMSE was negligible.

Let’s move on to data preprocessing. Since we are testing the 'cuteness' of an animal, any significant cropping, random erasure, blur, or colour jitter is likely to have an adverse and unpredictable effect on the score. As such, our data augmentation is limited to only horizontal flips, which we conduct on-the-fly with probability 0.5 to save RAM. All images are also resized to the expected input size of 224x224x3 since they come in varying input dimensions, and the pixels in each input channel of each image are normalized with their mean and standard deviation. Image normalization is important as without it, the ranges of distributions of feature values would vary during training and certain weights could end up at different scales which impedes training progress. However, there are quite a few open-source models on Kaggle that seem to use image augmentation to improve RMSE, so this assumption about augmentation may be false.

The dataset was randomly split into roughly 80% training set and 20% validation set. The model was then trained on the training set for 20 epochs (with early stopping if validation set RMSE failed to improve for 4 epochs) Colab's free GPU with batch size of 32 and the SGD optimizer with momentum of 0.9 and L2 regularization of 0.001. The loss function used was the mean-squared-error (MSE) loss. The standard dropout value of p=0.5 was used in the final layer. Various learning rates were tested (since too large learning rates may cause the model to skip over certain parameter values, while low learning rates mean training time takes too long) and the best validation RMSE scores are given below. Note that the training and validation sets have roughly the same proportion of pets (stratification) in each score band. Early stopping with a patience of 4 was used to make it so that if the validation RMSE failed to decrease for 4 epochs, the model would stop training. The motivation for our configuration comes from Srivastava’s paper, which has empirically shown that dropout combined with L2 regularization along with high momentum yields the best performance. However, as training is computationally-intensive, grid search of hyperparameters such as L2 regularization coefficient and momentum were not used, nor was k-fold cross-validation.

We can see that the best learning rate appears to be 0.001. The model converges relatively quickly to a validation RMSE of 17.9965 for ResNet-50 in 14 epochs, which took about 40 minutes in real time. It can be shown that increased depth using the ResNet-50 model over ResNet-18 does improve the performance slightly, but it is unclear if even deeper ResNets will perform better as we also have to consider how deeper networks will take even longer to train as there are more parameters to update. As we can see, the model performs slightly better than the simple regression and tree-based models based solely on metadata.

Let’s visualise how our model predicts pawpularity scores. tSNE, or t-distributed Stochastic Neighbour Embedding is used to graphically visualise higher-dimensional data in 2-dimensions, and here I have two plots that show the differences in certain images. The coloured frames surrounding each image indicate the actual pawpularity score of that animal, where warmer colours like red, orange, and yellow indicate lower scores, while cooler colours like blue, teal and green indicate higher ones, with purple being the 91-100 score band. We can see the model separates our data into 5 main clusters, putting phenotypically similar images together. For instance, these images on the right are images of animals with darker fur like black cats and labradors. We notice that the purple dots (high pawpularity scores) correspond to the tail end of this cluster alongside many pale-coloured dogs and close up shots. Overall, we see that the model seems to separate animals into different classes well, but does poorly when it comes to separating images into their appropriate score buckets. Clearly, further finetuning is needed.

As our validation set was stratified to be similarly distributed to the training set and hence has a large proportion of pets in the 30-40 score range, to further improve model generalizability, we constructed a distribution-aware (weighted) RMSE loss function in accordance with the label-distribution smoothing approach recommended by Yang’s paper. From our training set, we would compute the average proportion of each score band in the training set, and use its inverse to weight the RMSE loss. This reweighting procedure is similar to the LDAM loss used in classification of imbalanced datasets, as our new loss function would cause the model to heavily penalize RMSE of rarely seen score bands (60-80 range) and ensures that the model does not largely learn to predict the most common score values in the 30-40 range.

After training our model with this new loss function and early stopping, our best validation RMSE scores are:

We denote Model A to be ResNet-50 with MSE loss and Model B to be the ResNet-50 with weighted RMSE loss. Notice that Model B is a significant improvement over Model A, achieving a validation RMSE of 13.37!

However, our Kaggle test RMSE is unfortunately not that good despite the otherwise leaderboard-topping validation RMSE results, indicating that the distribution of scores in the test set is likely wildly different compared to the given data. Therefore, we decided to tinker with other CNN structures and see if they would provide better results, and chose the EfficientNet structure as it can be easily scaled up and its performance on the CIFAR-100 dataset is comparable to that of ResNet-50.

EfficientNet conducts grid search to find the relationship between different scaling dimensions of the baseline network under some resource constraint and determines a compound scaling coefficient for each dimension. This method has been shown to improve model accuracy and efficiency compared to conventional scaling methods like ResNet. A neural architecture search is also used to determine a "good" framework, and EfficientNet largely relies on the mobile inverted bottleneck convolution (MBConv) block, which divides the original convolution into a depthwise and then a pointwise convolution to reduce computational cost with minimal accuracy loss. MBConv also uses a linear bottleneck in the final layer of each block to prevent loss of information from the ReLU unit. The EfficientNet structure is given below, and we make some slight modifications to the final layer by adding BatchNormalization, Dropout, and two Dense layers:

This model, which we will call Model C, was trained for 20 epochs with Adam optimizer (since Adam generally converges faster than SGD), initial learning rate of 0.001, the MSE loss, dropout parameter of p=0.2, and early stopping. Model C achieved a validation RMSE of 14.8244, which is slightly worse than the validation set. Unlike our previous ResNet models, we could see that both the loss function and the model's validation RMSE is continuing to decrease at a steady pace even after the 20th epoch, which suggests that the model could continue to be trained until the validation RMSE starts to stagnate.

Nevertheless, due to time constraints, Model C was submitted to Kaggle and achieved an excellent result of 18.044, which is a significant improvement on the prior models and not too far from the top score of 17.65.

There are still several factors we can consider in future analyses. The metadata is given in both the training and test datasets, they were manually labeled and although predicting the score of the raw image is useful, it may be more helpful if we are able to extrapolate which features are the most associated with Pawpularity, since that can inform PetFinders how to take pictures that are more likely to be popular. Therefore, a CNN could be built to extract a 12 by 1 feature vector from each image corresponding to the metadata.

Additionally, the pawpularity score itself is not solely influenced by photo composition, but rather the specific animal in that picture. Certain users may have a preference for certain breeds of animals, so if the information is available, incorporating information such as cat/dog breed, age, gender and disposition instead of a single "Info" variable may provide further opportunities for enhancing performance. For that reason, a standardised pawpularity score for all animal breeds may be a misleading metric.

A more rigorous search of the hyperparameter space could be used to further finetune model performance, as well as selecting from different optimizers (i.e. Adagrad, RMSprop), different values of L2 regularization and the dropout parameter. Moreover, ensembling CNNs by training each of them on subsets of the entire given data and averaging their predictions could be used to improve generalization, although this is extremely time and resource-intensive.

Finally, it may be that the premise of using CNN for feature extraction is flawed. Recently, the vision transformer model type, or ViT, has been used to achieve state of the art results for image classification, treating each image as a sequence of image patches and embedding each of them into a position, which is fed into a transformer encoder. The self-attention mechanism present in ViT helps it detect hierarchies and alignments between different image patches, which may be useful for predicting a nebulous property like pawpularity (that humans probably can’t even predict from the images.)

In conclusion, the overall structure of EfficientNet may be more suitable to this problem compared to ResNet due to its better performance on the test set, lower training cost with the mobile bottleneck convolution blocks, and grid search of neural architecture. Linear and tree-based models based on the metadata alone are likely too simplified to provide any meaningful insight, and only make predictions near the median score range. However, the metadata inputs could potentially be concatenated with the CNN output at some stage and combining both forms of input data may yield even better results.