Estimating Pawpularity of Pet Adoption

Germayne Ng, Amanda Solis
Georgia Institute of Technology
{gng3, asolis31}@gatech.edu

Abstract

In this work, we provide an overview of our experience in the PetFinder.my Kaggle competition, in which we are tasked with building an interpretable machine learning system that can predict the popularity score of rescue pet images. The popularity score is used by Malaysia's leading animal welfare platform in two ways: to provide accurate recommendations to users who are searching for a pet and to guide users who upload the image on ways to generate the most appealing pet image.

In order to evaluate our approaches, we establish several simple baselines for popularity score prediction. For our main models, we train several deep CNNs using transfer learning and vision transformers. We also explore data augmentation approaches, pseudo-labelling to increase our data size and complexity and visualization techniques to identify features that result in higher or lower scores.

1. Introduction

PetFinder.my is Malaysia's leading animal welfare platform and is dedicated to getting homeless and stray animals in Malaysia adopted. They currently use an experimental feature, the "Cuteness Meter", to rank and recommend pet photos- this helps to both unite families with pets that they find adorable and to help animals find loving homes faster. The current feature uses criteria such as picture composition, photography quality, pet pose and background to judge pet attractiveness, and although artificial intelligence-based, is sometimes inaccurate. In order to build a better ranking system and to improve animal welfare, PetFinder.my hosted a Kaggle competition where they challenged the Machine Learning community to develop a model to analyze raw pet images and predict their popularity score.

In this paper, we provide an overview of our experience as entrants in the competition and detail the deep learning approaches we use to build, optimize, and evaluate our pet popularity prediction models. To aid with model evaluation and interpretability, we use methods to visualize features and image parts that the model uses to generate its popularity score. Methods to interpret a prediction can help suggest improvements to uploaded images to give rescue animals a higher chance of finding happy homes.

2. Method

In this section, we will describe the data used as well as how we decide to formulate this as a deep learning optimization problem. We will discuss the various model architecture considered, augmentations as well as additional technical details of how we improved the model's learning ability and performance.

2.1. Problem Setup

The goal from this competition is to predict the popularity of a given pet's shelter photo as well as meta data. The popularity score is derived from the pet's profile's traffic data across various web and mobile platform and range from 0 to 100. This means we can approach this as a regression as well as classification problem. From a regression standpoint, we will aim to optimize the mean square error as a loss criterion. Alternatively, as a classification problem, we first scale the data via min-max scaling, to make the target label $y \in [1, 100]$. Then we can simply treat this as a classification problem on soft-labels. Hence we will optimize the binary cross entropy loss.

Framing the problem as a pseudo classification task provides the advantage to utilize many techniques or augmentation that might seem to be more suitable specifically for classification loss functions, such as mixup, cutmix or even pseudo-labelling (which we will see later on). Hence, we decided to optimize the model using binary cross entropy based on the scaled target labels.

2.1.1 Data

The competition organizer provided a large number of various pet images, along with meta data as well as the pawpuparity score. Majority of the images contains pet dogs as well as cats based on different pet's profile page from PetFinder. Each image contains a unique ID as well as list of meta data which includes binary indicators for presence

of subject focus, eyes, face...etc It is worth stating that the organizers have explicitly mention that the meta data are manually labelled for key visual quality and composition parameters. However, the meta data are not used in anyway to derived the pawpularity scores. We visualized 30 samples from the images in fig 1. In total, there are 9912 unique pet profile images in which we will later split between training as well as validation set in order to optimize the model's parameters.

2.2. Modelling Architectures

In this section, we will describe some of the architectures utilized in our experiments. We will describe how we set up the different baseline models using machine learning regressors as well as image based neural networks. For the list of pretrained models, we utilized the collection of weights from timm library. [27]

2.2.1 Baseline Models

In order to gauge the success of our deep learning experiments, we establish several traditional machine learning baselines. The first, and simplest, is a naive baseline based on the global popularity score mean of the training data (mean=38.04). The second uses a traditional computer vision approach that has been used extensively in facial recognition [8] to generate histogram of oriented gradients (HOG) features. These features are then fed into a linear regression model. Our final baseline is the most sophisticated and the most comparable to our later experiments, as it leverages a pre-trained CNN model, ResNet-34 (discussed further in section 2.2.2), to generate image embeddings. These are then fed into a linear regression model.

HOG is a feature descriptor used in computer vision for object detection applications. It divides the image into even cells and then counts events of gradient orientation in each cell, thereby compressing and encoding the original image and retaining the structure of the object (edges and edge orientation). In our implementation, we generate HOG features with 9 orientations per histogram and use a window size of 32 by 32 pixels, selected as a reasonable balance between resolution and noise. In Figure 2, HOG features are displayed on the right, next to the original image which was sampled from the competition training data. As can be seen, HOG reduces a lot of background noise in the image and highlights object edges. Due to its success in facial feature discrimination, we thought it would be an improvement over just predicting a constant like we are doing with the mean baseline.

ResNet-34 is a residual deep learning network trained on ImageNet and released by Microsoft. It is one model, among many, that we can choose from to establish a baseline transfer learning model. This flavor of transfer learning uses the pre-trained model as a fixed feature extractor, meaning we take ResNet-34 pretrained on ImageNet and remove the last fully-connected layer. Then, we pass the new pet images through the model to generate a vector for every image containing the activations of the last hidden layer which can be passed in as features to a linear regression model. Many studies show that features obtained in this way are a very effective method for visual recognition tasks [2].

2.2.2 Transfer Learning Image Models

Many prominent ImageNet based models have shown remarkable impact when it comes to utilizing pretrained weights for image based downstream tasks over the past few years. In [25], it is shown that ImageNet based models have positive impact on downstream tasks such as classification. These models are trained on the ImageNet dataset, with 1000 classes. Similarly, we will be using the well-known image based architectures: Resnet, Densenet as well as Efficientnet, with some slight modifications. We will discuss in more details in the experiment section.

ResNet as known as residual networks [10] was introduced back in 2015 and was considered state of the art at that time. It was inspired partially by VGG's architecture and introduced the concept of residual connections via connection shortcuts, which allows the deep architecture to optimize easier as well as gain performance in terms of accuracy. These skip connections better optimization and reduces model degradation for deep networks. In particular, we selected the resnet-34 variant which contains 34 weighted layers as described in the original paper.

EfficientNet released in 2020 [16] focuses on scaling convolution neural networks in areas such as network's width, depth, image resolution. An effective compound scaling method was introduced to scale any convolution neural networks and retain model efficiency. The resulting networks trained have shown comparable state of the art performances against some of the above mentioned ImageNet based convolution models ranging from 1.5 times up till 21 times smaller in parameter size. We selected the baseline EfficientNet-B0 for our experiments.

Apart from convolutional based architectures, transformer based architectures are starting to gain momentum in terms of achieving good results. Transformers are regarded mostly as state of the art for naturally language procoessing (NLP) tasks. More recently, transformer based vision models such as Vision Transformers (ViT) as well as have been Swin Transformer (Swin) have shown promising and competitive results. We aim to incorporate these architectures as well as transfer learning to benchmark along side the convolutional models described above.

ViT also known as Vision Transformers are described

in [12] where transformers are applied directly to image patches along with sequence of linear embeddings. The application is analagous to NLP where image patches are treated like word tokens. The authors found that vision transformers worked well with largest images despite lacking inductive bias as compared to convolution neural networks. Furthermore, from (The Vit process the images by first paritioning the image into 16 by 16 patches, transformed linearly to patch vectors and along with positional embeddings are processed in a transformer encoder, similar to the BERT model.

In 2021, *Swin transformer* further enhances the application of transformer based architecture to computer vision domain. In [29], the authors proposed a hierarchical feature map representation that merges patch windows as it goes deeper in the layer. The hierarchical representations starts with small sized patches and gradually merged with neighbouring patches as the layer goes deeper. It also introduces a shifted window concept which in the subsequent layer, connects between neighbouring non-overlapping windows in previous layers, This enables self-attention in a given layer that previous was limited to a window to have cross window connection in the next layer.

3. Experiments

In this section, we will outline the various experiments as well as discuss the empirical results. The code can be found in [17].

3.1. Experiment Setup

To ensure fair comparisons, transfer learning was done by utilizing the 5 different models as the backbone architecture. We then do the same modifications to all 5 pretrained models: removing the last fully connected layers as well as the head and adding 2 linear layers with activation PRelu in between. PRelu, also known as Parametric Rectified Linear Unit that was introduced in [11] shows improvement in model fitting as well as generalized the Rectified activation units (Relu) by introducing learnable parameters with no additional computation costs. Note that for the convolution neural network models, we retain the global average layers while for the transformer based models, we simple add the final output to our linear layers. The architecture can be seen in figure 3. For transformer based architecture, there is no global average pooling layer and the output is simply passed directly to the full connected layers.

We illustrate the learning curve of the transfer learning models on two of the fold for the training loss as well as the validation RMSE. We selected fold 2 and fold 5. For the former, the reason is that most models seems to perform the worst for this fold, which may illustrate more challenging out of fold validation data. For the latter, most models performed the best. The figures can be found in figure 4. More discussion of the results will be in the evaluation section.

3.1.1 Image Augmentations

It is established that for deep neural networks to succeed, we need a large number of training data [3] [9]. One way to prevent deep learning models from overfitting as shown in earlier works such as AlexNet is to perform random cropping and flipping as well as changing RGB intensity are applied to inflate the training examples [1]. In [23], the authors described that data augmentation approaches the overfitting problem directly from the root: the training dataset. Augmentation works with the assumption that more information can be extracted by the large amount of new augmented data generated from the original training dataset. We define our baseline of augmentation to include the following: horizon and vertical flips, affine transformation, color jitter and lastly: random resized crop (RRC). We visualized some of augmentation on a mini batch of training data in figure 5.

Another popular learning principle as a form of augmentation we can adopt to make our deep neural network more robust is to introduce *mixup*, which was proposed in 2018 [13]. Mixup is simply a form of data augmentation where we constructs virtual training examples as well as target labels by performing a convex combination of images as well as their corresponding target labels within a given sampled batch of training data. The procedure is as follows: we first sample training examples along with their respective target labels and linearly interpolate them as follow:

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda)x_j \\ \tilde{y} &= \lambda y_i + (1-\lambda)y_j \end{aligned} \tag{1}$$

where $\lambda \sim \text{Beta}(\alpha, \alpha)$. This allows us to sample the mixup vicinal distribution and produce virtual feature along with virtual target vectors. Note that (x_i, y_i) and (x_j, y_j) are 2 separate images sampled randomly from the training data. In practice, we perform mixup within a batch of sampled training data.

As discussed in the original paper, introducing mixup yields several benefits: it encourages our neural network to behave linearly between training examples, and reduce oscillations when predicting out of sample. In other words, it improves the model's ability to generalize and discourages memorization. The authors showed that implementing mixup lead to overall better stability in terms of model prediction and improves state of the art models to generalize on ImageNet and CIFAR datasets. We also visualize how a mixup will alter the images in figure 6. Notice how the images are blended with a random selected pair of image. The resulting target labels are blended linearly in a similar manner as well. We will test 2 different values of α based on the mixup paper.

A related improvement technique known as *cutmix* was introduced as an improvement over mixup [22]. The authors described one of the drawback of mixup is that virtual samples generated are locally ambiguous and unnatural and hence, may confuse the model. Cutmix improve over this by selecting a patch region from a different image and replacing the region. Target labels are then blended based on the proportion of area of patches selected from other images. Similarly, cutmix utilizes the beta distribution in determining the region of patches to crop. One of the main differences in our experiment is that mixup is done within mini-batches of images while cutmix is done across images within the dataset. Our implementaion is based on this repository [14]. Instead of working on one-hot-encoded labels, we modified the target labels to blend on our scaled pawpularity. We used the default cutmix parameters defined in the paper. In particular, we follow the configurations used by the authors for CIFAR with distribution $\beta\ (1.0, 1.0)$, probablility set to 0.5 and allowing a mix of 2 images patches to be blended with the primary source image.

3.1.2 Overparameterization

Another area we wanted to explore is to answer the question if larger model with more parameters help in better generalizing. [7] showed and proved theoritically that overparameterization generalize better than smaller neural networks. Furthermore, advancement in NLP architectures involved larger parameters over time with better results [28]. In our case, due to initial computational constraints, we were only able to train on smaller variants of prominent models, especially on recent transformer based architectures. By utilizing pytorch's automatic mixed precision [19], we are able to train our model on a larger variant without getting out of memory error. For both models, we utilized the same augmentations described above as well as using mixup for both benchmarks. We only replace the model backbone for comparison and retain the same full connected layer.

3.1.3 Pseudo labelling

Prior to this Kaggle competition, PetFinder.my participated in another competition with a similar setup and goal: use pet images and profile metadata to develop ML models to predict how quickly a pet will be adopted and make recommendations to increase the pet's adoptability. Unlike the current competition, pet images were not limited to the pet's single profile image, but included all images associated with a pet. Having access to both train test images from this competition allows us to leverage data from a similar domain in a semi-supervised setting (a training paradigm where we have access to both labeled and unlabeled data). One popular way to leverage unlabeled data is via self-supervised learning or self-training. We use a self-training method inspired by [5] which consists of three steps. First, a teacher model is trained on the Pawpularity dataset. Then the teacher model generates pseudo labels on Adoption dataset. Finally, a student is trained to optimize the loss on human labels and pseudo labels jointly.

We train a student tiny swin model with mixup and cutmix augmentations and experiment with various pseudo-label probability threshold cutoffs to determine if fewer, but higher quality pseudo-labeled predictions impacts performance. To generate the pseudo-labels, we use a large swin model trained on the labeled Pawpularity competition training data and mixup augmentations as our teacher. Due to its size (200 million parameters) we do not report online performance for this model on the Pawpularity test data, however, it does achieve the lowest 5-fold CV mean RMSE in our experiments with the original data (see results in 3) so we use it for pseudo-labeling the 72,728 additional Adoption competition images.

3.1.4 Optimizer & Model averaging

One of a challenge is to determine the correct epoch for saving deep neural networks. Optimal epoch can be determined via early stopping based on the validation data set's evaluation metric during training at the end of each epoch. Furthermore, a good approach to reduce variance is to consider an ensemble of models near the optimal epoch. However, it is a challenge in determining this optimal epoch and hence, one needs to save all the weights for every epoch. This is also not scalable.

In [20], it is shown that averaging of model weights results in a model that is better in generalization. In particular, it introduced stochastic weight average (SWA) by averaging the model's parameters at different point in time during training with a customized scheduler that provides specific cyclical or constant learning rate.

Aggregation of model parameters can also be done using exponential moving average [26] [6], where a teacher model is updated by taking exponential weighted average of the student model's weight. This procedure shown improvement in prediction quality. Furthermore, in [4], it is mentioned that exponential moving average of weights helped in reducing variance in models saved.

Motivated by these successes, we hope to make use of model parameter average to avoid the dilemma of selecting the best epoch but rather aim for a combination of model parameters. This can be parameterized by α which determines the weightage of the recent parameters. Higher α will allow the latest epoch's parameter have a large proportion over early epochs. Hence, we define our exponential moving average of parameters as such:

$$\theta_t = \alpha * \theta_t + (1 - \alpha) * \theta_{t-1} \quad (2)$$

We trained a swin transformer model along with the best mixup augmentation. We set α to be 0.99.

4. Evaluation of results

As discussed in the previous section, we have 9912 unique pet images along with corresponding meta data as well as pawpularity scores as target labels. As such, we will be splitting the data into training set as well as validation set. In terms of evaluating the model offline, we have Images in the validation set will be used to optimize the model parameters. For online evaluation, the competition organizers have a collection of 6800 test images that are only available for inference by submitting our model weights as well as inference code online. For the scope of this paper, we will focus only on the 25% of the test images for our online evaluation. In terms of validation scheme, we went with averaging across 5 fold stratified cross validation. we found that the offline evaluation based on the average 5 fold error is correlated to the online error and since evaluation online submission is limited, we will restrict online evaluation only when necessary.

Traditional ML Baselines Our simplest traditional baseline, the global mean, performed surprisingly well considering the mean RMSE was not far off from more sophisticated models. This actually implies some data quality issues, as we would hope that a much more sophisticated model could improve performance more significantly [18]. Also surprising was that the HOG regression model showed no improvement over the mean baseline as HOG features are typically very useful in facial recognition. However, since so much information about the image is lost, the global representation that HOG features describe do not provide sufficient signal to improve popularity prediction.

Furthermore, ResNet34 regression model outperforms our simpler baselines and even outperforms some of our more complex models, including the experiment where we actually fine-tune the ResNet-34 weights with the competition training data.

Baseline Models and transfer learning. We illustrated the results in table 1. In general, the swin transformer model was able to achieve the best result. Surprisingly, the Vit model performed the worst. However, as shown in [29], this clearly highlights the strength of swin model's hierarchical feature maps (as layers gets deeper) with shifted window approach over Vit's feature maps of fixed low resolution (as layers gets deeper). Moreover, Vit was shown to require large-scale training datasets such as JFT-300M to perform well. [12]. For the ImageNet convolution models, we can see ResNet as well as DenseNet performing pretty well. This shows the inductive bias in convolution networks work well in general computer vision problems.

Overall, swin transformer's shifted window based self-attention has shown effectiveness on our dataset. In particular, the key design of swin's shifting of window paritions enable self-attention computed in the previous layer of a given window to cross boundary to a different window. This provides contextual information learnt from a given window to a different collective window in the next layer. It is interesting to note that the hierarchical representations in the swin architecture produces similar feature map resolution, going from big to small as with VGG and ResNet.

Augmentations, mixup and cutmix. We illustrate the results in 2, by comparing the impact of various augmentations on top of our results shown in baseline. For mixup, We tested 2 values of α as inspired by the paper for distribution β (α, α): 0.2 and 0.4. We found that setting α to 0.4 resulted in a better validation error. Overall we can see better model performance mixup augmentation against no mixup for our problem. The mixup α parameter defines the strength of mixup interpolation, as described in the original paper. Higher α results in virtual examples that are further from the training examples and helps to prevent memorization. As for cutmix, we see slight improvement as compared to mixup. We are not able to replicate the differences in performance between cutmix and mixup as described in the paper and believe that further empirical experiments on the parameters will likely yield even better performance.

Overparameterization. Our empirical results indeed showed that larger variants of these pretrained models gave improvement across all 5 folds of validation error. We can see this in table 3. However, note that the large Swin model contains 200 million parameters as compared to the tiny swin model with 28 million parameters.

Exponential moving average of weights. Our results can be found in table 4. We are able to get improvement by changing the way we aggregate our model's parameters.

Pseudo-labels. As illustrated in table 5, using self-training to pseudo-label the Adoption competition data significantly increased model performance. Using all additional 72k image pseudo-labels as training data (totaling 82k train images) gives the best online performance with a mean RMSE of 18.04 on the Pawpularity test data. While the additional images increase performance, they also make training much more expensive and time-consuming, taking approximately 3.5 hours per each fold versus 1.5 hours when only 20k training images are used. We were excited, but not surprised by these results, since self-training has been proven to be extremely beneficial in high data/strong augmentation regimes [5].

5. Analysis

In this section, we will compare and contrast the different image models utilized in our experiment by visualizing the

feature maps and to reason about the possible differences in performances, especially in the context of our problem. We will explore impact of input pixels and feature maps using saliency maps and grad-cam. Finally, we will explore the differences of representations using centered kernel alignment (CKA).

5.1. Saliency maps

Saliency maps are useful for understanding which pixels in an image are relevant to the network during classification. They are calculated as the gradient of the class score of interest with respect to the input. So for each input pixel, we can determine how much the predicted class probability would go up (for positive gradient) or down (for negative gradient) if we changed it. In figure 7, saliency maps for several Pawpularity training images are shown for two different models: our ResNet34 baseline transfer learning model and our best pseudo-labeled swin tiny transformer model.

The swin transformer model's saliency maps show that the model is more accurately able to hone in on the subject of interest, as the largest gradients are concentrated on the main subject's figure and seldomly outside of it. Alternatively, while the traditional CNN model is influenced by the main subject's figure, it is also influenced by random pixels in the background that seemingly have nothing to do with the subject.

5.2. Gradient-Weighted Class Activation Maps

Gradient-weighted Class Activation Maps (GradCAM) visualizations can be useful for highlighting regions of an image that positively or negatively affect the class of interest. Rather than backpropogating the gradient all the way through the image, in GradCAM, we backpropagate the gradient through to the last convolutional layer to produce a coarse localization map. In order to produce this localization map, we have to weight each pixel of each feature map in the layer with the gradient before we average over the feature maps. This helps GradCAM to decide how important each feature map is to the class of interest and to produce the final heatmap highlighting important regions of the image. These heatmaps are displayed in figure 8 for our ResNet34 baseline transfer learning model and our best pseudo-labeled swin tiny transformer model.

Compared to the saliency maps, the GradCAM class activation maps are much harder to interpret since both models exhibit inconsistent behavior- in some cases the background positively influences the score and in others the subject's face or figure positively influences the score. While they do share this quality and are both good at segmenting the subject from the background, one difference is that the Swin tiny model appears to have a wider focus, i.e. more pixels in the image are activated. This is inline with our expectations, as recent approaches have shown that trasnformers are able to go beyond convolutions in terms of being able to capture long-range/global visual dependencies [21].

5.3. Representation Similarity

Inspired by the work described in [15], we aim to explore qualitative differences in terms of representations learnt from the different models. We can perform linear centered kernel alignment (CKA) based on the gram matrices of the last representation output prior to the fully connected layers. Based on the weights of models trained on the first fold, we evaluated the CKA between various architectures on a mini-batch of out-of-fold samples. This is to ensure there is no data leakage. Similar to the authors, we compute the CKA between 2 gram matrices based on the representation output from 2 different neural networks. The gram matrices captures a particular layer's representation correlations. Furthermore, based on [24] the benefits of CKA is that it is invariant to orthogonal, isotropic scaling of representations and is able to capture intuitive notions of similarity, despite different initialization of parameters.

We visualize the pairwise differences via linear CKA between the various architectures using their final representation output in figure 9. The convolution models with similar inductive bias showed similar CKA scores. One surprising finding is the Swin transformer having some degree of similarity to the convolution models. This is may due to hierarchical based architecture converging to smaller feature representation emulates to some degree similar inductive bias present in convolution based architectures. Lastly, ViT results in very different representation as compared to the other models. This emphasizes previous findings in [15] where higher layers found in ViT have different representations as compared to the well known convolution architectures. Further exploration will be required to dive deeper to explain qualitative differences.

6. Conclusion

In this paper, we explore various machine learning based solutions to accurately predict pawpularity score for PetFinder: with simple traditional vision methods to various established vision architectures - ranging from known convolution architectures with inductive biases to recent transformer inspired architectures. We also explore the impact of various augmentation techniques, aim to alleviate the dillema of model selection in an epoch by exploring impact of model aggregation. Furthermore, we also found improvement in utilizing pseudo-labelling techniques via self-supervision. Last but not least, we end the paper by analyzing the model using visualization techniques such as saliancy and grad-cam, as well as compare and contrast representation across architectures using centered kernel alignment.

References

[1] Ilya Sutskever Alex Krizhevsky and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. 2012. 3

[2] Josephine Sullivan Ali Sharif Razavian, Hossein Azizpour and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. 2014. 2

[3] Peter Norvig Alon Halevy and Fernando Pereira. The unreasonable effectiveness of data. 2009. 3

[4] David Wong Oliver Lange Todd Hester Luis Perez Marc Nunkesser Seongjae Lee Xueying Guo Brett Wiltshire Peter W. Battaglia Vishal Gupta Ang Li Zhongwen Xu Alvaro Sanchez-Gonzalez Yujia Li Austin Derrow-Pinion, Jennifer She and Petar Veličković. Eta prediction with graph neural networks in google maps. 2021. 4

[5] Tsung-Yi Lin Yin Cui Hanxiao Liu Ekin D. Cubuk Quoc V. Le Barret Zoph, Golnaz Ghiasi. Rethinking pre-training and self-training. 2020. 4, 5

[6] Pavel Izmailov Ben Athiwaratkun, Marc Finzi and Andrew Gordon Wilson. There are many consistent explanations of unlabeled data: Why you should average. 2019. 4

[7] Alon Brutzkus and Amir Globerson. Why do larger models generalize better? a theoretical perspective via the xor problem. 2018. 4

[8] Lourdes Ramirez Cerna. Face detection: Histogram of oriented gradients and bag of feature method. 2013. 2

[9] Saurabh Singh Chen Sun, Abhinav Shrivastava and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. 2017. 3

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. 2015. 3

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. An image is worth 16x16 words: Transformers for image recognition at scale. 2021. 3, 5

[13] Yann N. Dauphin David Lopez-Paz Hongyi Zhang, Moustapha Cisse. mixup: Beyond empirical risk minimization. 2018. 3

[14] Ildoo Kim. Cut mix. https://github.com/ildoonet/cutmix, 2019. 4

[15] Simon Kornblith Chiyuan Zhang Maithra Raghu, Thomas Unterthiner and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? 2021. 6

[16] Quoc V. Le Mingxing Tan. Efficientnet: Rethinking model scaling for convolutional neural networks. 2020. 2

[17] Germayne Ng and Amanda Solis. Pawnet. https://github.com/germayneng/pawnet, 2021. 3

[18] Adriano Passos and other kaggle participants. kaggle discussion. 5

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. pages 8024–8035, 2019. 4

[20] Timur Garipov Dmitry Vetrov Pavel Izmailov, Dmitrii Podoprikhin and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. 2019. 4

[21] Ashish Vaswani Irwan Bello Anselm Levskaya Jonathon Shlens Prajit Ramachandran, Niki Parmar. Stand-alone self-attention in vision models. 2019. 6

[22] Seong Joon Oh Sanghyuk Chun Junsuk Choe Sangdoo Yun, Dongyoon Han and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. 2019. 4

[23] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. 2019. 3

[24] Honglak Lee Geoffrey Hinton Simon Kornblith, Mohammad Norouzi. Similarity of neural network representations revisited. 2021. 6

[25] Linda Studer, Michele Alberti, Vinaychandran Pondenkandath, Pinar Goktepe, Thomas Kolonko, Andreas Fischer, Marcus Liwicki, and Rolf Ingold. A comprehensive study of imagenet pre-training for historical document image analysis. 2019. 2

[26] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. 2018. 4

[27] Ross Wightman. Pytorch image models. *GitHub repository*, 2019. 2

[28] Barret Zoph William Fedus and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. 2021. 4

[29] Yue Cao Han Hu Yixuan Wei Zheng Zhang Stephen Lin Ze Liu†, Yutong Lin† and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. 2021. 3, 5

A. Figures

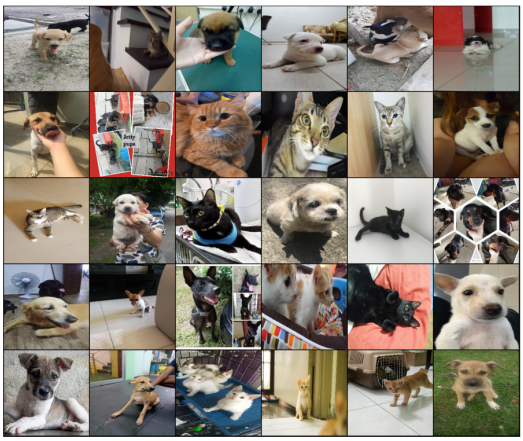

Figure 1. Examples of training data provided. Each pet image has a corresponding meta data as well as pawpularity score

Figure 2. Examples of resized image on the left (128x64) and HOG features on the right.

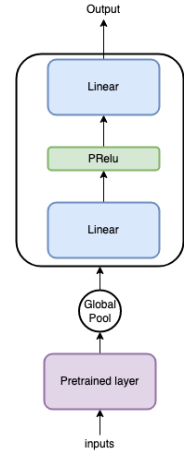

Figure 3. Transfer learning baseline image architecture

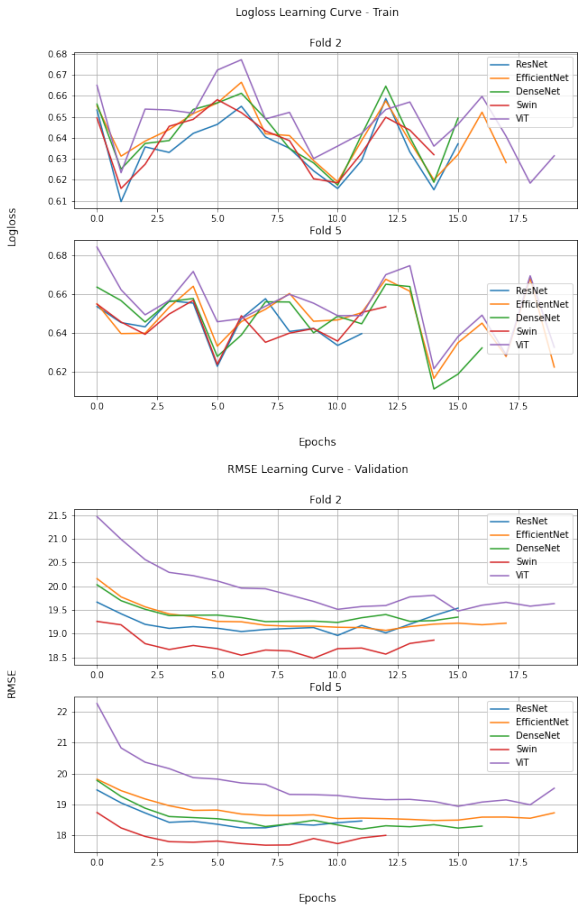

Figure 4. Learning Curves on selected folds

Figure 5. Base augmentation applied to training images set

Figure 6. Visualization of first 7 images amongst the batch of image data. We compare original, mixup and cutmix based on our implementation

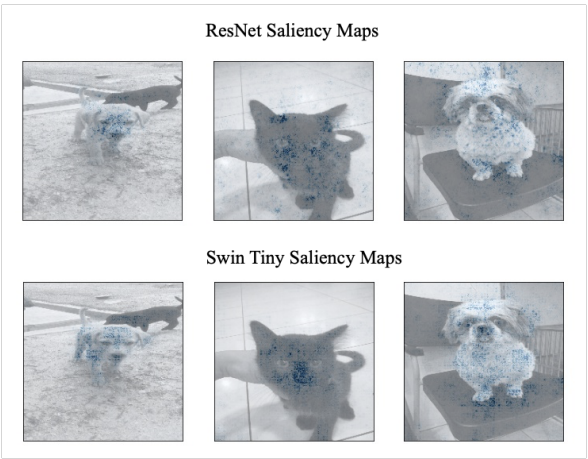

Figure 7. Saliency maps applied to training images set.

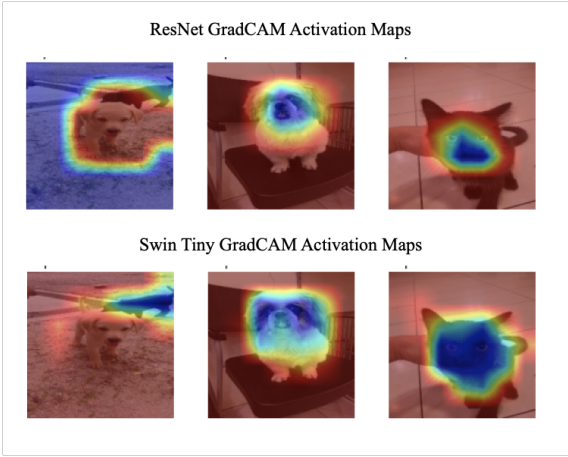

Figure 8. GradCAM applied to training images set.

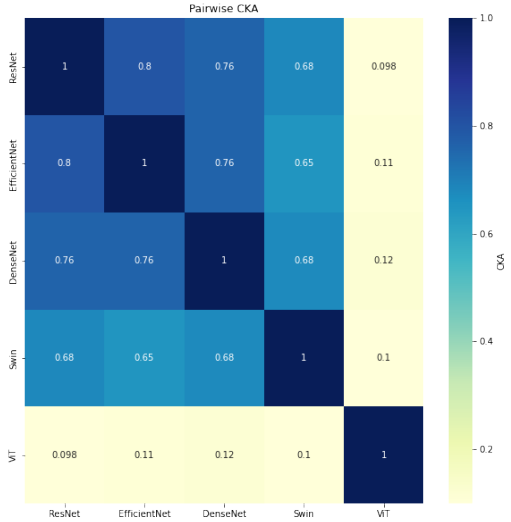

Figure 9. Differences in representation in various models trained on pawpularity using CKA

Models	Mean Validation RMSE
Global Mean	20.51
HOG regression	20.59
Resnet34 regression	18.96
Resnet34	19.01
Efficientnet-B0	19.04
ViT Tiny Patch16-224	19.493
Swin Tiny	18.40

Table 1. Overview of baseline linear models and transfer learning models

Augmentation	Mean Validation RMSE
Base A (Augmentation)	18.40
Base A + RRC	18.37
Base A + RRC + MU 0.2	18.35
Base A + RRC + MU 0.4	18.26
Base A + RRC + CM	18.24

Table 2. Experimentation on alpha parameter in mixup

B. Tables

Folds	Tiny Swin (RMSE)	Large Swin (RMSE)
1	18.40	18.11
2	18.55	18.31
3	18.29	17.87
4	18.10	17.79
5	17.85	17.89

Table 3. Comparing impact of larger parameter model

Model Update	Best Fold / Mean Validation RMSE
No EMA	17.85 / 18.24
EMA	17.74 / 18.13

Table 4. Impact of exponential moving average of weights

Probability Threshold	Count Training Images	Online RMSE
0.0	82,680	18.05
0.3	68,753	18.09
0.5	21,516	18.25

Table 5. Pseduo-labeling experiment results using different probability threshold cutoffs.