Supplementing Deep Classification Models with Homogeneous Parallel Transfer Learning

Calvin I. Cronin

Northwestern University School of Professional Studies

MSDS 458: Artificial Intelligence and Deep Learning

Address to which correspondence should be addressed:

calvin.i.cronin@gmail.com

Abstract

As the power of Artificial Intelligence (AI) increases, so does model complexity and hardware requirements, raising the forefront's barrier to entry. The increasingly prevalent application of Transfer Learning adopts the knowledge of a preexisting model trained for a separate task. The novel approach explored here, designated as Parallel Transfer Learning (PTL), examines how adding related training data impacts a deep learning model that classifies images of only cats and dogs. By maximizing the utility of available data and related knowledge, Transfer Learning is a promising tool for democratizing AI enablement. The different augmentations of training data demonstrated that added data similar to both existing classes could facilitate bias in the model, while added data similar to the target classes separately showed potential for marginal improvements to classification performance.

1 Introduction

Transfer learning is not a new concept in the field of machine learning, it even predates widespread use of deep learning. As the collective contributions and knowledge of the deep learning field grows, so does the collection of available pre-trained models, especially those trained with hardware/data not commercially available or easily accessible. The advent of Chat-GPT and other massive generative models was a landmark event in this regard, and many offer APIs that encourage transfer learning with their models. So in recent years the widespread adoption of transfer learning has gained traction. It empowers users to make better use of preexisting resources and circumnavigate limitations of data, hardware, knowledge, and time spent cultivating a model from scratch.

Despite the unprecedented potential of transfer learning, it still has limitations and requires significant contextual and technical knowledge to implement properly. This begs the question of whether there is a more accessible way to at least partially leverage the mechanics and benefits of transfer learning. So in an effort to further understand the limitations and benefits of transfer learning, this paper introduces a variation titled Parallel Transfer Learning. Transfer Learning is typically a linear and sequential process where a large scale or related source model is trained, and its architecture and learned features are augmented to make predictions for a smaller target dataset. The pretrained model is the dominant presence in the model. PTL is not sequential and doesn't require pretraining. Instead it combines the target task data with a smaller set of related supplemental data, trains on the combined dataset all at once, and then makes predictions about just the target data.

Another way of looking at PTL is training a multi-class classifier, then making predictions on data that represents a smaller subset of the initial classes. The key being that the initial extraneous classes are deliberately selected in relation to the smaller target subset. This paper focuses on implementing this with Convolutional Neural Network (CNN) aimed at distinguishing between images of cats and dogs. The same CNN model was trained to classify cats, dogs, and other related animals (foxes, wolves, jaguars) and attempted to classify the same test set containing images of only cats and dogs.

The scope of PTL is slightly different from that of traditional transfer learning. Contemporary transfer learning focuses on using powerful existing resources to make a comprehensive model. PTL is an exploration in using additional data as a lightweight and easy to implement model booster. So the scope of this paper is far from exhaustive and more focused on laying the groundwork for pressing questions in the space.

2 Literature Review And Definitions

Transfer Learning as a whole is a well researched and documented field. Along with it's most prevalent procedures and concepts, Transfer Learning is most prominently defined by (Weiss, Khoshgoftaar, and Wang 2016, 1-40) as the following:

> *A domain D is defined by two parts, a feature space* χ *and a marginal probability distribution* $P(X)$*, where* $X = \{x_1, \dots x_n\} \in \chi$*. Given a source domain* D_S *with a corresponding source task* T_S *and a target domain* D_T *with a corresponding task* T_T*, transfer learning is the process of improving the target predictive function* $f_T(\cdot)$ *by using the related information from* D_S *and* T_S*, where* $D_S \neq D_T$ *or* $T_S \neq T_T$*.*

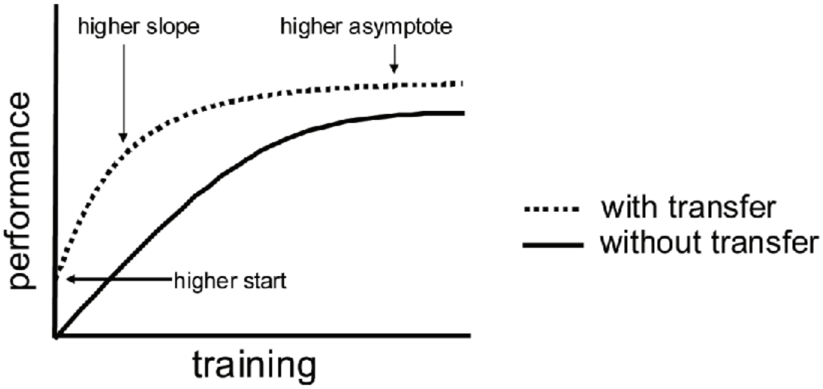
This publication also makes the distinction between homogeneous and heterogeneous transfer learning. In homogeneous transfer learning the source and target domains share the same feature space $(\chi_S = \chi_T)$. In heterogeneous transfer learning the domains have different feature spaces $(\chi_S \neq \chi_T)$. The publication also introduces negative transfer, a potential detrimental outcome where the model performs worse than if trained from scratch with just the target data. The models demonstrated in this paper are homogeneous as image size and input methods were the same for target classes and supplemental classes.

Current strategies for transfer learning are depicted in (Li and Hoiem 2017). The two primary approaches are fine-tuning and feature extraction. In fine-tuning, the source model is trained to completion and then without reinitializing any weights, the whole model is further trained to a slighter degree with the target data. In feature extraction, additional layers specific to the target task are built on top of the bottom of the pretrained source model. Then these new layers are initialized and trained while the weights of the base source model remain unchanged. PTL is distinctly different from these two, but the article also mentions Multitask Learning (MTL)which is explored further (Caruana 1997, 41-75).

MTL is the most mechanically similar strategy to PTL, but there are some key conceptual differences. Both MTL and PTL train on related tasks. But MTL aims to improve the performance of multiple tasks simultaneously, whereas PTL is only incorporating related data in order to boost the performance of a singular goal. Additionally, the components of the neural network that pertain to the respective tasks of MTL are overlapped and entwined, but not exactly the same. PTL is still more simple and lightweight as all data shares the same exact network.

The three common measures by which transfer learning may improve a learning are identified in (Torrey and Shavlik 2010, 60-94):

2.1 Three ways in which transfer might improve learning: a higher performance at the very beginning of learning, a steeper slope in the learning curve, or a higher asymptotic performance.

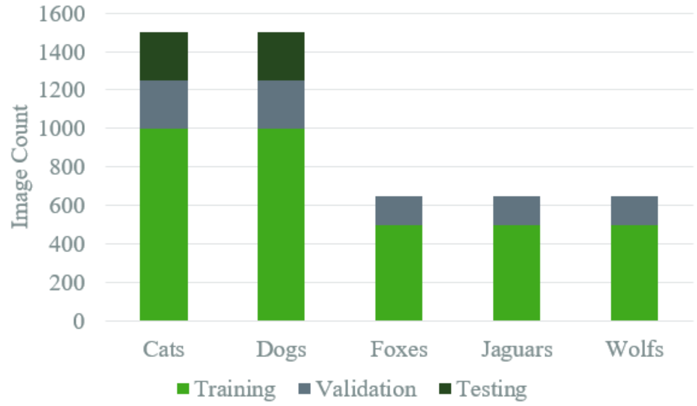


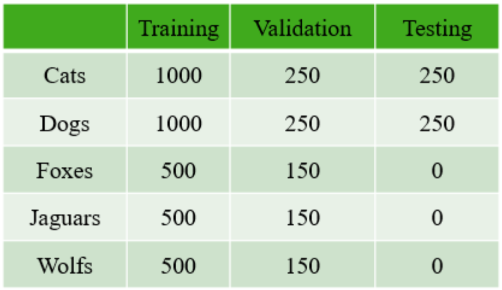
While the means of evaluation used in section 5 are not exactly the same as outlined in fig. 2.1, they are based around the same principle. Lastly, the TensorFlow tutorial on Transfer Learning (Tensorflow 2023) provided both the data ultimately used for the target task. It also gives translations between Transfer Learning concepts and their code representations that were utilized to implement the experiments found in this paper.

3 Data Processing

The dataset used in this paper had 4950 unique images in total. While they were not necessarily all used in the same model iteration, Fig 3.1 gives the breakdown of all the images by their class and function.

3.1 Collective Dataset By Classification and Model Function



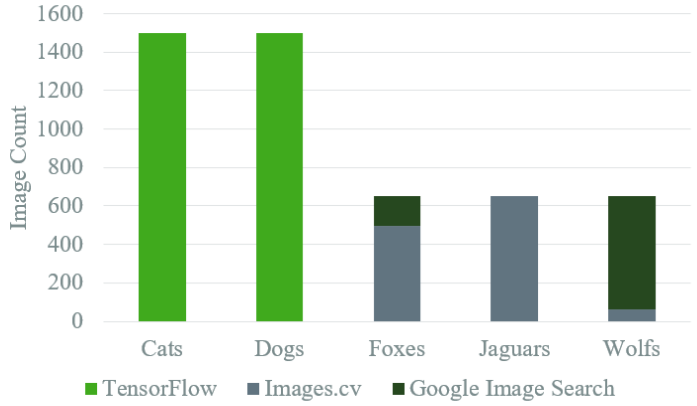


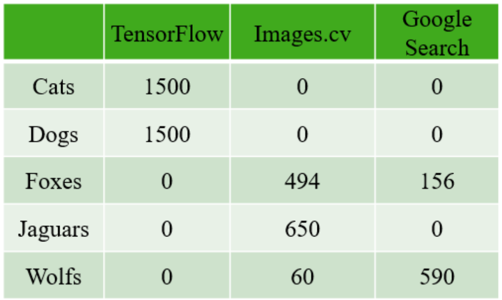
	Training	Validation	Testing
Cats	1000	250	250
Dogs	1000	250	250
Foxes	500	150	0
Jaguars	500	150	0
Wolfs	500	150	0

3.1 Data Sourcing

The data was collected from a variety of open source datasets and some rudimentary web scraping. All of the target task data (cats and dogs) was sourced from the TensorFlow Tutorial dataset. Most of the fox, jaguar and wolf images were sourced from the computer vision focused website images.cv(). The rest of the dataset was filled in by Google image searching the respective labels and using the Chrome extension “Download All Images” to do as the name suggests.

3.1 Collective Dataset By Classification and Data Source





	TensorFlow	Images.cv	Google Search
Cats	1500	0	0
Dogs	1500	0	0
Foxes	0	494	156
Jaguars	0	650	0
Wolfs	0	60	590

3.2 Data Preparation

A small amount of the data preparation was handled with a hands-on approach, and the rest was handled automatically with TensorFlow’s ingestions tools. For hands-on, the image files that originated from Google search were manually scanned to validate whether they were accurate for the assigned label (fox/wolf). Then the image files were placed in a specific nested directory structure that aligns with the expectations of TensorFlow’s function calls.

TensorFlow’s ImageDataGenerated was used to pull in the images and perform randomized augmentations to help the data generalize better after being trained. The images were in color with their pixel values ranging from 1-256. This was normalized so that all pixel values were between 0-1. The images were rotated up to 20 degrees from their initial position, some images were flipped/reversed. The function performed width shift, height shift, zoom, and shear transformations all with the range parameter 0.2. The images were all resized to 150x150 pixels, and grouped into batches of 32.

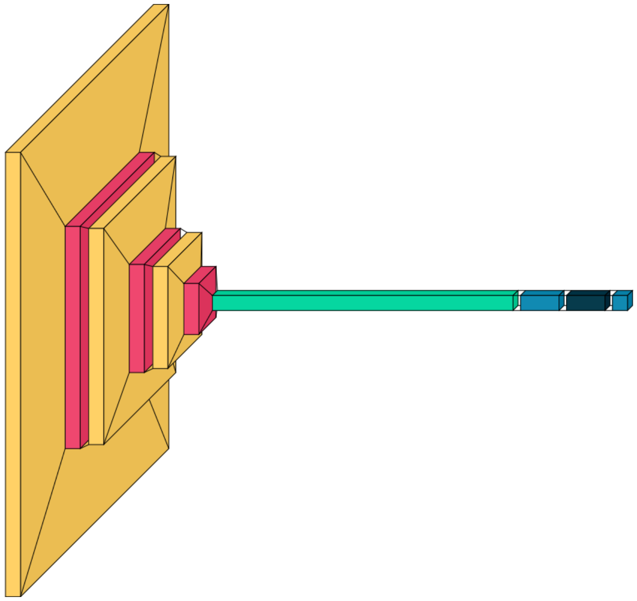
4 Methods

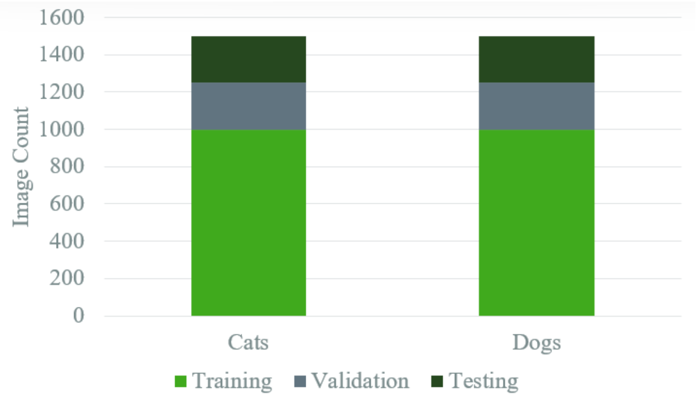
The general procedure structured path of comparison. In all, 3 models/trials were run. A baseline followed by two combinations of different supplemental classes added to the training data. These additional classes were also added to the validation data to make sure the training evolved reasonably well, and that any surprising outcomes with the test data were not due to poor/over training. The models were all compiled with the Adam optimizer and were trained for 20 epochs. But the supplemental data was never added to the final test data, as all three models made predictions on a dataset of only cats and dogs with their predictions also limited to either cats or dogs. Finally the efficacy of these predictions were evaluated and compared.

4.1 Model 1 - Baseline

The baseline model was a simple CNN as depicted below, along with the data usage breakdown.

4.1.1 Baseline Model Simple Architecture





4.1.2 Baseline Data Utility

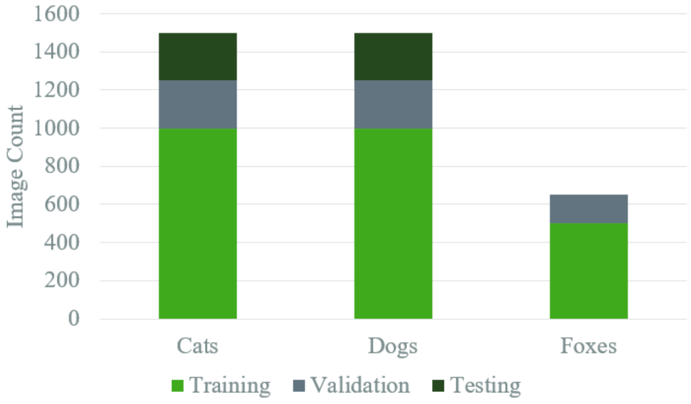
In the model visualization in Fig 4.1.1 The yellow represents convolutional layers with ReLU activation functions. The red represents MaxPooling layers. The aqua represents a flattening layer. The first blue is the first hidden layer which also has a ReLU. The dark blue layer is a dropout layer for regularization. And the last blue layer is the output layer with a sigmoid activation function for this trial. Further details can be found in the appendix.

This model structure is almost exactly the same as in the subsequent trials. This is because the focus of this paper is to gauge a change in performance, not optimize performance. So the objective is to find a simple but functional model, and hold the model itself as constant as possible as the dataset varies over subsequent iterations of supplemental data. This baseline model was run with only images of cats and dogs for training, validation, and testing.

4.2 Model 2: Cat-Dog-Fox

The concept being studied by Model 2 is the effect of supplemental data that is roughly equally similar to the two target classes. Empirically, foxes seemed the best choice for something equally similar to cats and dogs, so a smaller dataset of it was added to the training and validation data.

4.2.1 Cat-Dog-Fox Data Utility

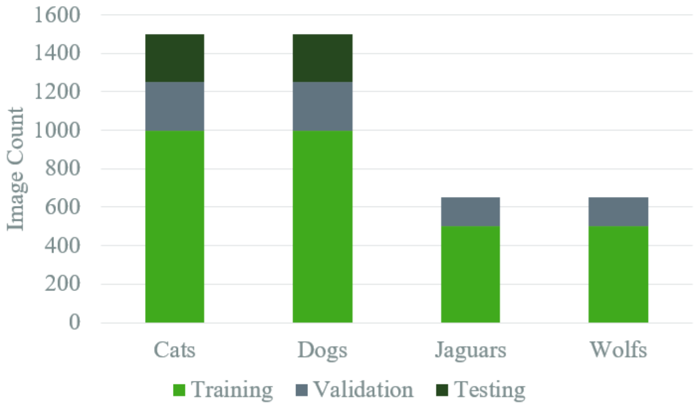


The model for section 4.1 was then used again here, albeit with some very small alterations necessary for a multiclass classification model. The output layer has an additional node and now uses a softmax activation function. And while the model was trained as multiclass, it did not make predictions as one. Again, the test set consisted of only cat and dog images. But for this iteration, the predictions made by the model during final testing were first run through a custom function that forces a prediction of either cat or dog. This function looked at the one-hot encoding probability values that the CNN outputs, and it chooses dog or cat based on the higher value, regardless of what the value for fox was.

4.3 Model 3: Cat-Dog-Jag-Wolf

The third and final trial was very similar to the second, but with a slightly different supplemental data setup. The concept being explored in this iteration was the effect of additional classes for each target class, but each additional class is related only to their respective target class and not the others. The implementation that made the most sense conceptually was the addition of a class for wolves (more related to dogs than cats), and a class for jaguars (more related to cats than dogs). So these were implemented in the training set and validation set as seen below.

4.2.1 Cat-Dog-Jag-Wolf Data Utility



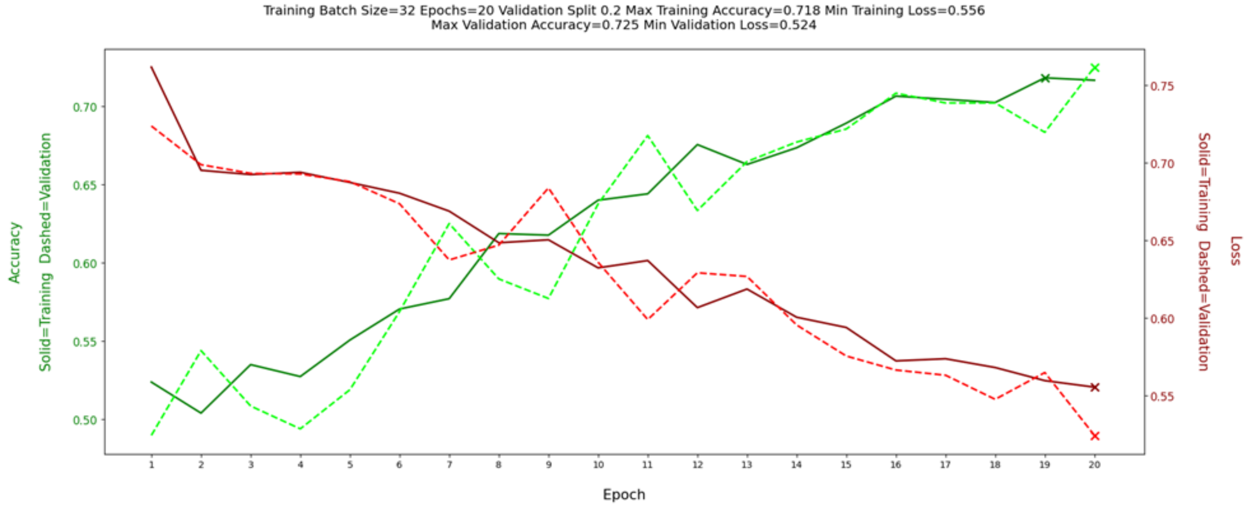
The same multiclass adjustments as in Model 2 were made, and predictions on the same dog-cat test dataset were made.

5 Results

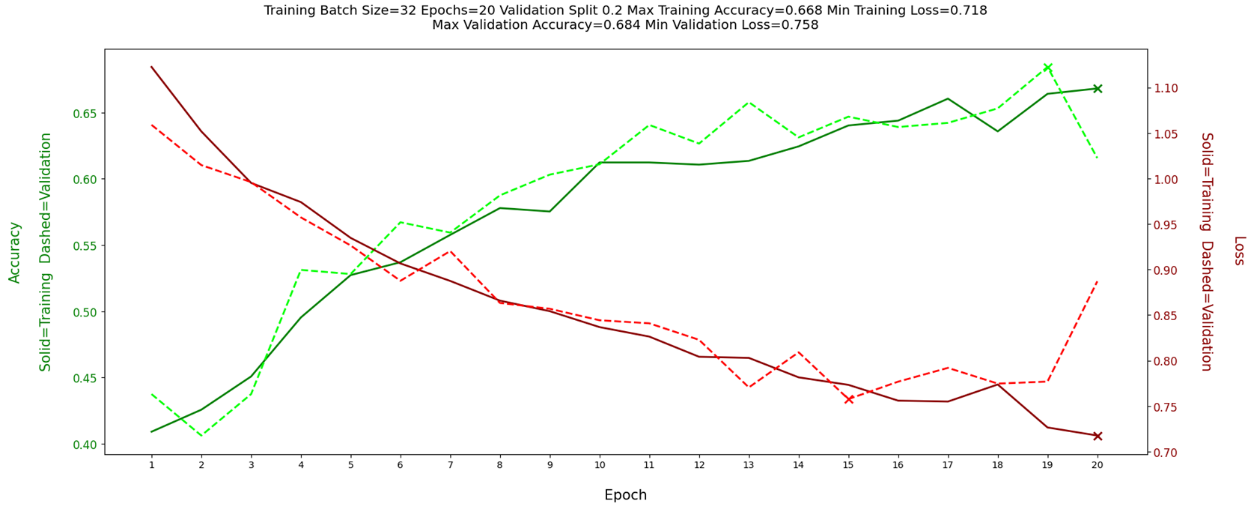
5.1 Training Evaluation and Metrics

Below in Figures 5.1.1 - 5.1.3 the training history over epochs for each of the three models are visualized.

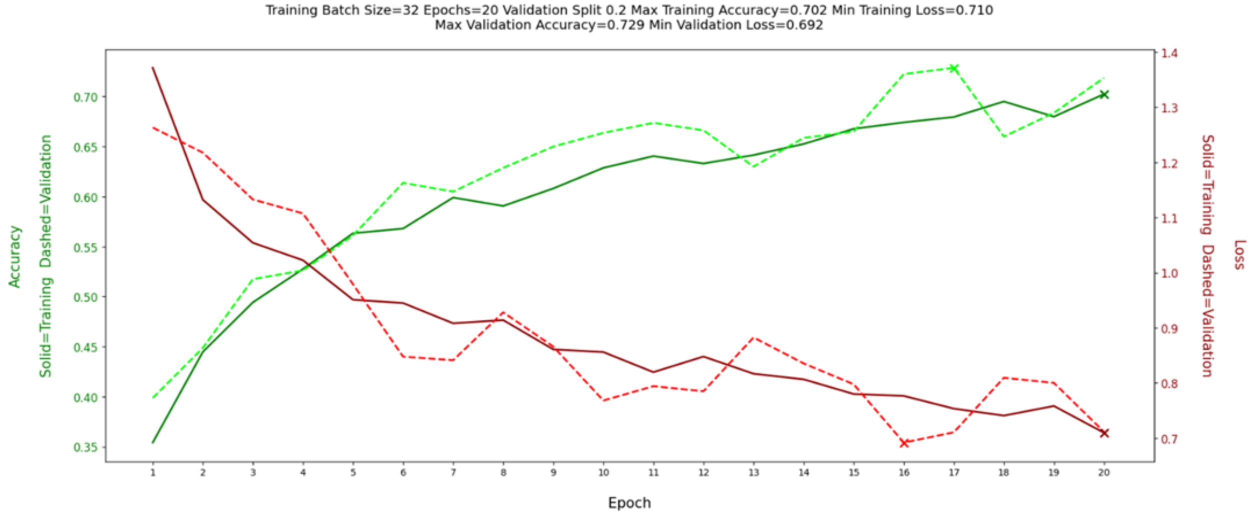
5.1.1 Baseline Model Training and Validation Metrics



5.1.2 Cat-Dog-Fox Model Training and Validation Metrics



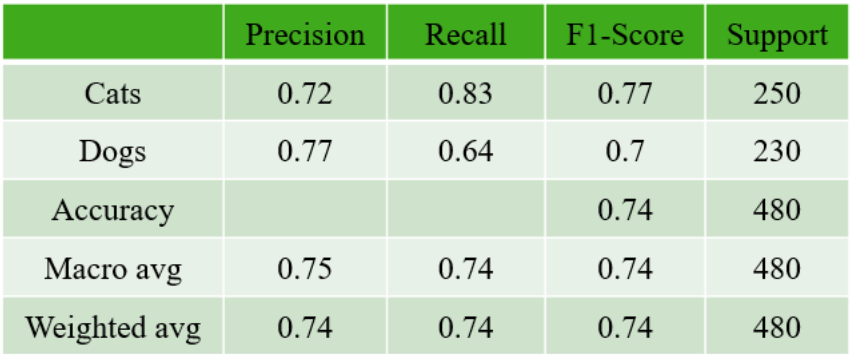
5.1.3 Cat-Dog-Jag-Wolf Model Training and Validation Metrics



5.2 Classification Efficacy

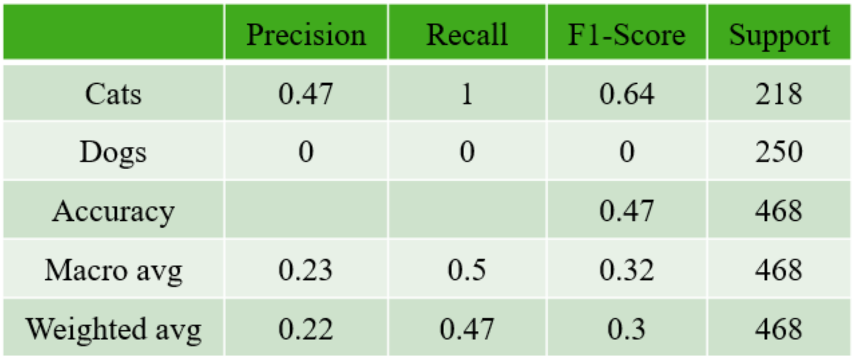
Figures 5.2.1 - 5.2.3 present the classification reports for each model after making predictions on the same shared Cat-Dog test dataset.

5.2.1 Baseline Model Classification Report



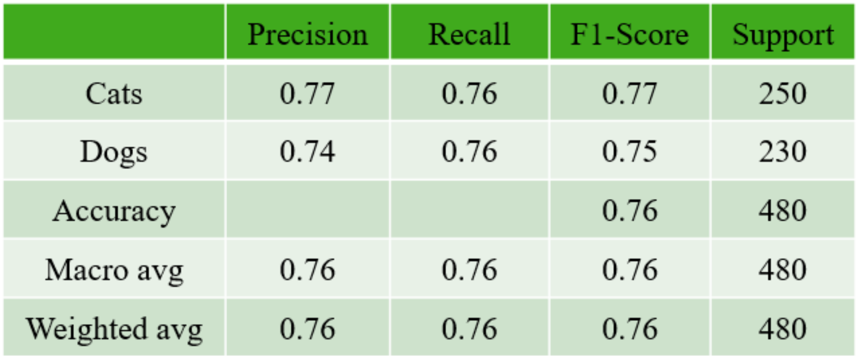
	Precision	Recall	F1-Score	Support
Cats	0.72	0.83	0.77	250
Dogs	0.77	0.64	0.7	230
Accuracy			0.74	480
Macro avg	0.75	0.74	0.74	480
Weighted avg	0.74	0.74	0.74	480

5.2.2 Cat-Dog-Fox Model Classification Report



	Precision	Recall	F1-Score	Support
Cats	0.47	1	0.64	218
Dogs	0	0	0	250
Accuracy			0.47	468
Macro avg	0.23	0.5	0.32	468
Weighted avg	0.22	0.47	0.3	468

5.2.3 Cat-Dog-Jag-Wolf Model Classification Report



	Precision	Recall	F1-Score	Support
Cats	0.77	0.76	0.77	250
Dogs	0.74	0.76	0.75	230
Accuracy			0.76	480
Macro avg	0.76	0.76	0.76	480
Weighted avg	0.76	0.76	0.76	480

6 Analysis and Interpretation

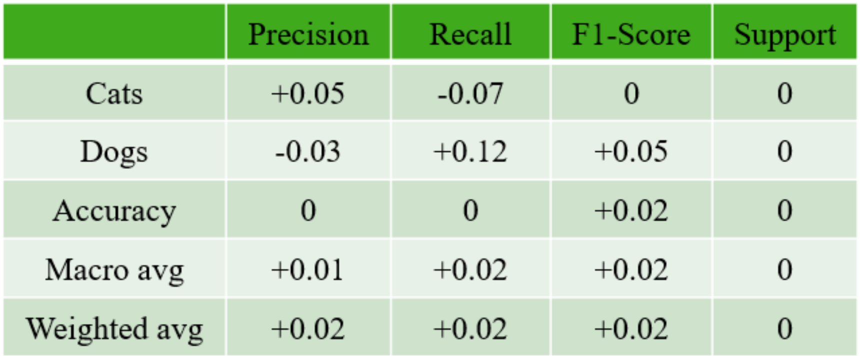
When examining how the different models' accuracy and loss evolved over time/epochs, they all demonstrated comparable results. They all seemed to train reasonably well with the validation loss/accuracy following their training counterparts well. There were some slight differences in the loss/accuracy cures and how steep they were at a given epoch. But generally I would say all of them are fairly conservative/flat, and if model optimization was the objective, these trials could benefit from an increased learning rate. Additionally given the trajectory and clear grouping of the curves, it is evident that the models have learned, but are actually somewhat undertrained. Again, because maximized performance is not the focus, the models being partially trained with the same number of epochs is actually perfect. It means comparisons between the models can be made at a benchmarked level of training, without wondering if measured differences could erroneously be attributed to overtraining or oversight.

When looking at the classification reports, the Baseline model performed well, but not great. It had an overall accuracy of 74%, with fairly balanced performance between the precision and recall of both target classes. Again, this accuracy is actually perfect for the purposes of benchmarking, as the model was clearly successful but definitely has room for improvement.

The Dog-Cat-Fox model did not perform well at all, almost to an unexpected degree. It was not successful at all with an overall accuracy of 47%. It had perfect recall with all the cats images in the test set, but at the trade off of 100% incorrect precision and recall with dogs. The model shifted drastically to only predict cats 100% of the time, regardless of input. This was initially assumed to be due to some extraneous error, but none was found. The one-hot formatted outputs were showing very similar values for cats and dogs most of the time, but always slightly higher in favor of the cat label. This is clearly an example of negative transfer learning as outlined in section 2, where model performance is worse than without the added data. The exact mechanics are subject to further investigation, but it seems like the addition of the fox data either introduced bias in the model. One possible explanation is that the smaller fox dataset made the whole training set unbalanced. These results do not indicate this model is impossible to train properly, but it is worth noting how drastically the added data affected the model.

The Dog-Cat-Jag-Wolf model showed more promising results with an accuracy of 76%. The performance metrics for this model were very similar to the baseline model, but marginally better. Not nearly enough to assert claims about the supplemental data being a superior approach, but the results are not trivial either. Fig 6.1 shows the difference from the baseline performance. A positive number means that the Dog-Cat-Jag-Wolf model had a higher metric in that area.

6.1 Cat-Dog-Jag-Wolf vs Baseline Classification Report



	Precision	Recall	F1-Score	Support
Cats	+0.05	-0.07	0	0
Dogs	-0.03	+0.12	+0.05	0
Accuracy	0	0	+0.02	0
Macro avg	+0.01	+0.02	+0.02	0
Weighted avg	+0.02	+0.02	+0.02	0

Whether or not this represents an overall improvement depends on the hypothetical use case of the model. But more than just raw predictive power increase, the supplemental wolf and jaguar data seemed to balance out the model a bit more. This model had significantly less tradeoff between precision/recall and between the target classes. This is further supported by findings in (Caruana 1997, 41-75) that MTL and joint training can have the effect of regularizers.

7 Conclusion and Future Work

Ultimately the Cat-Dog-Jag-Wolf model produced the best results of the experimental trials. While it was not a conclusive and total victory for PTL, it did demonstrate marginal improvements to both total accuracy and how robust different types of classification predictions would be. And even the complete failure of the Cat-Dog-Fox model provides value. It issues a warning of how massively detrimental bias can be introduced by closely related or unbalanced data.

PLT wrestles with some large questions that are imperative to both human and machine cognitive reasoning. Does knowledge about what is and is not a wolf help shape our understanding of what a dog is? Along with these large scale questions are the immediate needs the PLT seeks to address. As AI scales, it needs ways to remain agile and accessible. PLT in its current form may not be the perfect, dynamic, and lightweight model booster. But to at least some degree, these models gave evidence of parallel/shared feature representation working to a model's advantage with a very simple setup.

There are so many variations on the experimental trials that could illuminate novel relationships and implications for training with data from different domains. Repeating the trials with larger supplemental datasets so that the collective training set is balanced could remedy some of the observed bias issues. And training with less related image data could reveal limitations on shared feature learning. Not to mention further training and optimizing the models show here to see if their maximum performance varies. Any explorations in this space are an investment in the future enablement of AI practitioners.

References

Caruana, Rich. 1997. “Multitask Learning.” *Machine Learning* 28: 41–75. https://doi.org/10.1023/A:1007379606734

Li, Zhizhong, and Derek Hoiem. 2017. “Learning Without Forgetting.” *Lecture Notes in Computer Science* 9908, https://doi.org/10.1007/978-3-319-46493-0_37

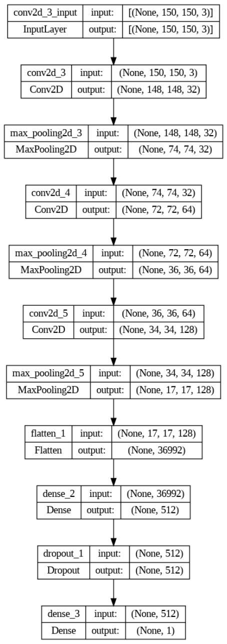
TensorFlow. “Transfer learning and fine-tuning.” Last modified October 27, 2023. https://www.tensorflow.org/tutorials/images/transfer_learning.

Torrey, Lisa, and Jude Shavlik. 2010. "Learning Algorithms for RBF Functions and Subspace Based Functions.” In Handbook of research on machine learning applications and trends: algorithms, methods, and techniques, edited by Emilio Soria Olivas, et al., 60-94. Hershey, PA: IGI global.

Weiss, Karl, Taghi M. Khoshgoftaar, and DingDing Wang. 2016. "A survey of transfer learning." *Journal of Big data* 3, no. 1 (May): 1-40. https://doi.org/10.1186/s40537-016-0043-6

Appendix

1.Entire CNN Structure:


conv2d_3_input | input: | [(None, 150, 150, 3)]
InputLayer | output: | [(None, 150, 150, 3)]
conv2d_3 | input: | (None, 150, 150, 3)
Conv2D | output: | (None, 148, 148, 32)
max_pooling2d_3 | input: | (None, 148, 148, 32)
MaxPooling2D | output: | (None, 74, 74, 32)
conv2d_4 | input: | (None, 74, 74, 32)
Conv2D | output: | (None, 72, 72, 64)
max_pooling2d_4 | input: | (None, 72, 72, 64)
MaxPooling2D | output: | (None, 36, 36, 64)
conv2d_5 | input: | (None, 36, 36, 64)
Conv2D | output: | (None, 34, 34, 128)
max_pooling2d_5 | input: | (None, 34, 34, 128)
MaxPooling2D | output: | (None, 17, 17, 128)
flatten_1 | input: | (None, 17, 17, 128)
Flatten | output: | (None, 36992)
dense_2 | input: | (None, 36992)
Dense | output: | (None, 512)
dropout_1 | input: | (None, 512)
Dropout | output: | (None, 512)
dense_3 | input: | (None, 512)
Dense | output: | (None, 1)