Cross-Species Transfer Learning: From Dog breed to Cat breed Classification using Deep Learning Techniques

Orestis Kotrotsios, Prodromos Kampouridis
Dept. of Digital Systems, University of Piraeus
Institute of Informatics and Telecommunications, National Center for Scientific Research “DEMOKRITOS”
Athens, Greece

25 June 2023

ABSTRACT

In this paper, we investigate the potential of Artificial Intelligence to enhance dog breed and cat breed classification using Transfer Leaning and Deep Learning techniques. This project is the critical component of the Deep Learning course curriculum and is part of the Master’s program at National Center for Scientific Research “Demokritos” and the University of Piraeus. Our idea is to use AI to leverage knowledge gained from dog breed classification to improve the accuracy of cat breed classification. To achieve this, we try several ways of transfer learning, including varying the number of frozen layers and the learning rate. We also perform cross dataset evaluation by training on dogs and testing on cats, and vice versa. Finally, we analyze the learning curve to determine how much data is necessary to transfer to the target dataset. This simulation shows how important is the dataset size in a transfer learning training.

KEYWORDS

deep learning; transfer learning; cross dataset evaluation; dog breed classification; cat breed classification; learning curve

1 INTRODUCTION

The ability to accurately identify different breeds of dogs and cats has numerous applications, including in animal shelters, veterinary clinics and breeding programs. Due to the plenty physical characteristics and subtle variations among breeds it can be a challenge to classify them correctly. Additionally, accurate breed classification can also help with understanding health issues and behavior traits related to specific breeds. The advances in technology such as deep learning can also play a significant role to address this issue.

2 DATASETS

In this study, we used two different datasets for our transfer learning task. The first one is the Stanford Dogs dataset which contains images of 120 breeds of dogs from around the world. Specifically, this dataset has been built using images and annotations from ImageNet. The second dataset we used is The Oxford-IIIT Pet Dataset with 29 categories, 17 breeds of dogs and 12 breeds of cats with roughly 200 images for each class.

Figure 1: Sample images from the Cat and Dog training data

3 DATA PREPROCESSING

To prepare the data for our analysis, we first organized the images into separate directories for cats and dogs. We then further organized the images by breed, creating a separate directory for each breed within the cats and dogs directories. After organizing the data, we created zip archives and reuploaded the dataset to Kaggle for our convenience.
Initially, we found that the results were not satisfactory and we successfully addressed this issue by merging the common classes from the Stanford Dogs and Oxford-IIT Pet datasets to increase the number of samples for dogs.

3.1 DATA AUGMENTATION

We applied data augmentation techniques to increase the size and diversity of our training set. We used the ImageDataGenerator class to apply horizontal and vertical flipping to our dog images at first. This helped us artificially increase the number of training samples and we achieved to boost our model's performance score by 2%.

3.2 GENERATIVE ADVERSARIAL NETWORKS

In addition to traditional data augmentation techniques, we also used GANs to generate synthetic data for our training set. We used a pretrained GAN model from TensorFlow Hub.

While we observed an increase in validation accuracy, the testing results were not satisfactory and upon further investigation, we discovered that these synthetic images were very similar to each other, resulting in a biased training set.

4 CNN MODEL

In this section, we introduce a simple Convolutional Neural Network (CNN) model that we have created and trained for dog breed classification. What's more, we present the training history, evaluate the performance on the test dataset and save the weights for future use in transfer learning, specifically for the cat breeds.

It should be noted that various structures were tested until we chose the appropriate model for the task. For instance, we continuously added regularization until we achieved satisfactory results.

4.1 MODEL STRUCTURE

The first layer in the model is an input layer that takes in dogs images of shape (224, 224, 3). The images are then passed through a BatchNormalization layer to normalize the pixel values.

The next layer is a 2D Convolutional layer with 32 filters, each of size 3x3. The activation function used is ReLU which introduces non linearity into the model. This is followed by another BatchNormalization layer and a SpatialDropout2D layer with a rate of 0.1 to reduce overfitting. Next, a MaxPooling2D layer with a pool size of 2x2 is added, to reduce the spatial dimensions. Besides, more blocks of Convolutional /BatchNormalization/SpatialDropout/Pooling layers with increasing numbers of filters, are added in order to extract more complex features from the input images.

After, the final convolution layer, the output is passed through a GlobalMaxPooling2D layer to reduce again the spatial dimensions of the input and prepare the data for input to the fully connected (Dense) layers. Lastly, two Dense layers are added. The first one has 512 units and uses a ReLU activation function, and the second one has the number of units equal to the number of classes and uses a softmax activation function.

4.2 HYPERPARAMETER TUNING

To achieve the best results, we performed hyperparameter tuning. Particularly, we used a grid search approach to find the best set of the parameters for the model. We also used the Adamax optimization algorithm during training. For each combination in the grid, we trained the model and evaluated its performance on the validation set. The best parameters are presented in the table below.

	Batch size	**epochs**	**LR**	**L1**	**L2**
Best Value	32	100	0.001	0.0001	0.0001

Table 1: Best results of hyperparameter Grid Search for the CNN model

The highest validation accuracy achieved through the grid search was 83.3%.

4.3 RESULTS

All the training process was monitored using a custom callback that we additionally created. This adjusted the learning rate and stopped training early if the validation loss was not improving for 3 epochs in the row.

4.3.1 PERFORMANCE

The performance evaluation of our CNN model is presented in the table below.

Layer	**Trainable Parameters**	**Validation Accuracy**	**Validation Loss**	**Test Accuracy**	**Test Loss**
1	1.841.368	83.6%	1.397	83%	1.468

Table 2: CNN Model's performance

Despite the small size of the dataset, the model achieves impressive validation and test accuracy, with relatively low losses.

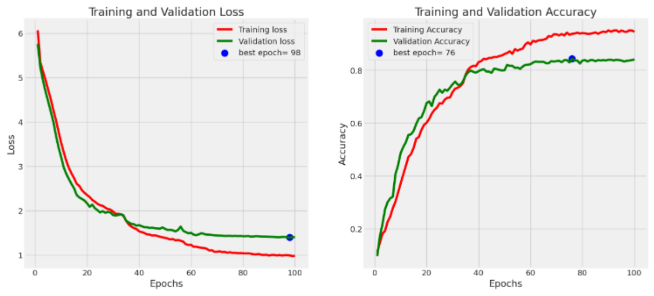


Figure 2: Accuracy and loss for Validation and Training of the CNN model

4.3.2 CLASSIFICATION REPORT

Furthermore, we created a classification report that displays metrics such as precision, recall, f1 score, and support for each dog breed and the outcomes were pretty good too.

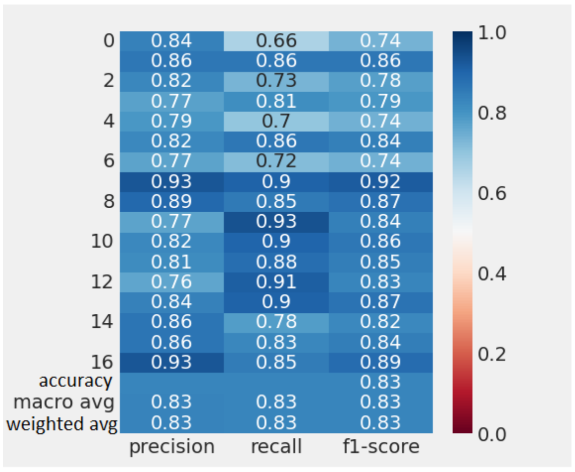


Figure 3: Classification report of the CNN model

4.3.2 CONFUSION MATRIX

To gain further understanding of the model's performance we also generated a confusion matrix. The results were satisfactory as well and as expected due to the high validation/test accuracy, and our model was not biased towards a specific class.

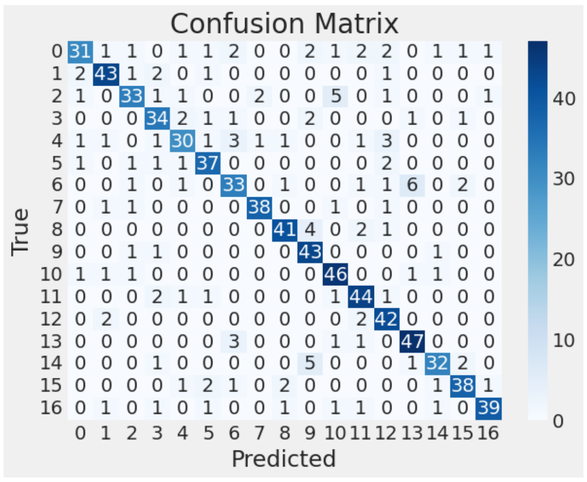


Figure 4: Confusion matrix of the CNN model

5. TRANSFER LEARNING

Transfer learning is a technique where a model, trained on one task, is repurposed on a second related task. In the transfer learning the trained weights of the first model are used to initialize the new model that has the same structure as the previous with the only difference the size of the output. After the initialization the output layer and some more layers can be trained so that the weights will adjust to the new task. This gives us the ability to train a new model with better performance and without the need to big datasets.

5.1 CNN

In our case, the pretrained dog breed classification is used as a starting point for the cat breed classification model so that we can evaluate the difference in the performance between a model that is trained in a dataset from scratch and a model that was trained with transfer learning.

The dog breed classification model is a Convolutional Neural Network following the same structure that was explained above in the Section 4.1.

5.1.1 METHODOLOGY

Firstly, we create a new dog breed classification model and we load the pretrained weights we saved earlier. Furthermore, we remove the last layer of the model and replace it with a new dense layer with the number of units equal to the number of cat breeds. This new layer is added to produce outputs for the cat breed classification task and then we compile the model with the same optimizer, loss function, and evaluation metric. Before training the cat breed classification models, all layers except the last one are frozen. Consequently, freezing these layers allows our model to retain the knowledge it has gained from being trained on dog breeds while adapting to the new task of classifying cat breeds. Moreover, the cat breed classification model is then trained on the cat breed dataset and only the weights of the last layer are updated. After training, we tried to unfreeze different number of convolutional layers and retrained it again to compare the results. Finally, this allowed the model to fine tune its weights on the cat breed dataset for each combination of trainable and untrainable layers.

5.1.2 RESULTS

After training, only the output layer of the CNN model for the cats, the results are not very satisfactory as it can be seen in the table 2, so we tried different combinations of hyperparameter for the transfer learning and the results are analyzed below.

Layer	Trainable Parameters	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
1	6.156	51%	2.226	57.6%	2.126

Table 3: Model performance with only Output layer trained

It is important to note, that our training has been halted at epoch 89 after 4 adjustments of learning rate with no improvement and the time elapsed was almost 13 minutes.

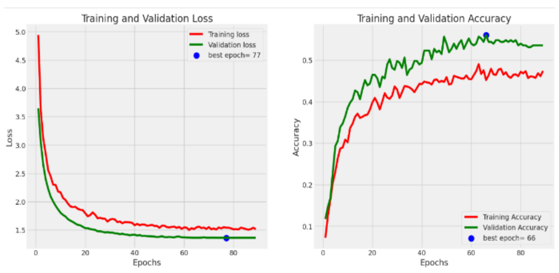


Figure 5: Validation accuracy and loss for the training of the output layer

For the different methods that we tried for the transfer learning we decided to freeze or unfreeze one Convolutional layer and one BatchNormalization layer together as a block. After this we tried to train the model with 1, 2 or 3 blocks trainable respectively with various learning rate values. In the Table 4 below, we present the best results having learning rate equal to 0.001, for 3 different variations of training. We did not set more blocks as trainable cause then it would be almost as if we were training a model without pretrained weights, since the model only has 5 convolutional layers.

Block	Trainable Parameters	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
1	1.449.996	77.8%	1.826	75.1%	1.815
2	1.745.676	78.6%	1.511	75.8%	1.400
3	1.819.788	81.1%	1.364	80.5%	1.331

Table 4: Comparison of model performance for 3 different blocks

As it can be seen from the Table 4, the best transfer learning method was the one with 3 blocks set as trainable, with the accuracy of the model to reach up to 81% and the loss to be 1.364 for the Validation and 80.5% and 1.331 for the Test. It is worth mentioning that each training took almost the same time, lasting approximately 5 minutes.

The total time of training was 18 minutes.

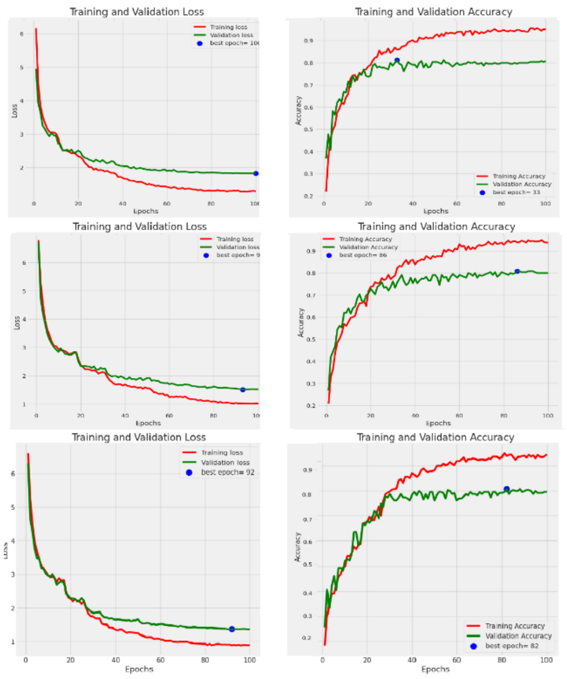


Figure 6: Validation accuracy and loss for 1, 2, and 3 trainable convolutional layers, respectively

Lastly, in Table 5 we are introducing the results of the CNN model of the previous section when it was trained in the cat breeds dataset from the beginning.

	Trainable Parameters	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
CNN	1.843.933	56.9%	2.227	62.2%	2.107

Table 5: Model performance when trained without transfer learning

By comparing the Tables 4 and 5, we can notice that the results after the transfer learning are much better than the ones from the model that was trained without transfer learning. The model that was trained with transfer learning has 25% better accuracy from the base model and almost half the loss.

5.2 EFFICIENTNET

We continued by experimenting with the EfficientNetB3 structure and its pretrained weights from the ImageNet dataset. This dataset is significantly larger than our dog breeds dataset, allowing us to compare the results of a larger model with ours.

5.2.1 METHODOLOGY

The process of the transfer learning for this model is similar with the CNN model. We firstly created a model with the structure of EfficientNet and used the weights of ImageNet from the TensorFlow library. We then replaced the output layer of the EfficientNet with an output layer that match the size of the cat's breeds and trained the new output layer. Then we proceeded to train last 10 layers of the model to increase its performance a bit more.

5.2.2 RESULTS

The highest validation accuracy achieved was 89.5% while the loss is 0.665. Besides, the training time elapsed was now almost 2 minutes and only 4 epochs to reach these scores.

	Trainable Parameters	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
Efficient Net	1.720.332	89.5%	0.665	91.2%	0.312

Table 6: EfficientNet performance with transfer learning using ImageNet weights

As shown in the figure below, both training and validation accuracy were already starting at very high values while the losses begin with values under 0.7 value. This can be attributed to the use of pretrained weights from ImageNet during transfer learning.

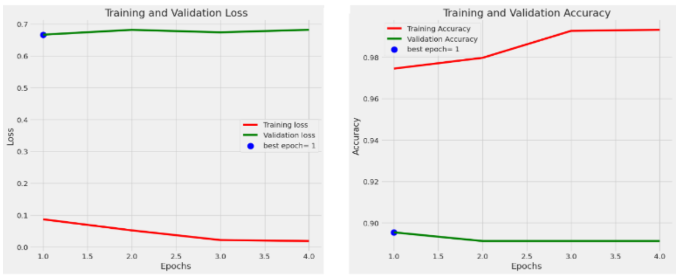


Figure 7: Comparison between Training Accuracy/loss and Validation accuracy/loss of the EfficientNet

5.3 SIDE-BY-SIDE COMPARISON

EfficientNet outperforms the CNN model in both validation and test performance, with a 14% lead in test accuracy. Furthermore, EfficientNet's training time is significantly shorter at just 2 minutes, compared to CNN's 15 minutes. This is due to the use of pretrained weights from ImageNet during transfer learning. In addition, it makes sense that EfficientNet performs better since ImageNet already includes 3 classes from our cat breeds dataset. Lastly, it is important to highlight that EfficientNet has almost 11 million total parameters, while the CNN has almost 2 million.

Model	Total Parameters	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
CNN	1.841.368	81.1%	0.707	77.5%	0.742
Efficient Net	10.801.979	89.5%	0.665	91.2%	0.312

Table 7: Transfer learning performance for 2 different models

5.4 LEARNING CURVE

In this section, we studied the impact of the dataset's size on the transfer learning results. We used the optimal parameters that gave us the best results in the transfer learning of the CNN model, specifically with 3 trainable convolutional layers.

In addition, we created 9 new datasets from the existing one, each one ranging from 20% to 100% of the original dataset in increments of 10%. We then followed the transfer learning process for each of these 9 new datasets. The Figure below illustrates the significance of dataset size in transfer learning. Initially, when the dataset is 20% of the original one (about 400 images) the validation accuracy is low and improves as the dataset increases. Similarly, validation loss decreases as the size increases.

Interestingly, we notice that transfer learning does not require a large dataset. With just about 200 images for each cat breed, we achieved 80% validation accuracy, compared to 60% without transfer learning. Furthermore, even with half the size of the original dataset, performance results remain satisfactory, as the rate of improvement decreases significantly after the half size.

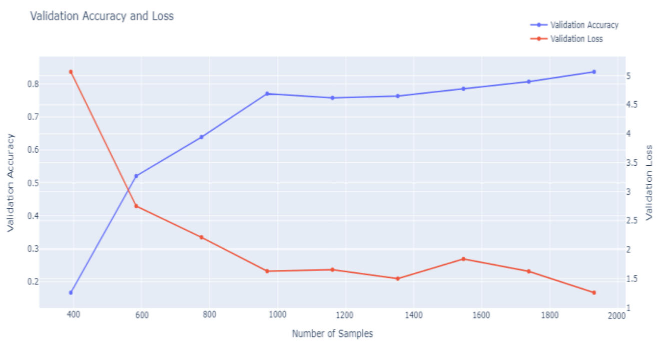


Figure 8: Learning Curve with 3 convolutional layers trainable

6. CROSS-DATASET EVALUATION

To reverse the transfer learning process, we trained a new deep learning model on the cat breed classification task using the cat dataset. As a starting point we used the weights from the pretrained cat's model. We tried to freeze and unfreeze a different number of blocks and the best one was the same as it was before, from dog to cat breed transfer learning. Similarly to the previous sections, we evaluated the performance of the transferred model on the validation and test sets.

Model	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
Cats to Dogs	83.1%	1.436	80.9%	1.557

Table 8: Transfer learning performance from cats to dogs

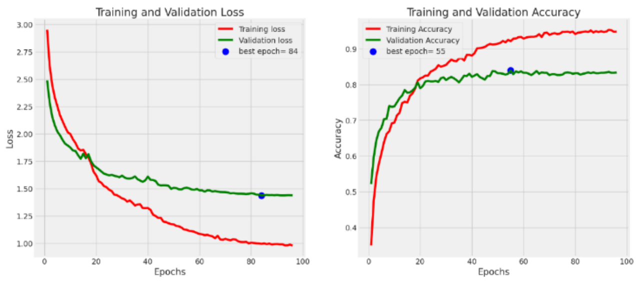
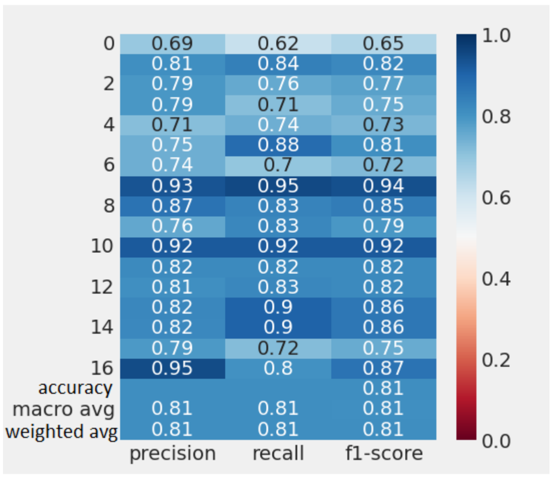


Figure 8: Training and Validation accuracy/loss of the new transferred model



	precision	recall	f1-score
0	0.69	0.62	0.65
1	0.81	0.84	0.82
2	0.79	0.76	0.77
3	0.79	0.71	0.75
4	0.71	0.74	0.73
5	0.75	0.88	0.81
6	0.74	0.7	0.72
7	0.93	0.95	0.94
8	0.87	0.83	0.85
9	0.76	0.83	0.79
10	0.92	0.92	0.92
11	0.82	0.82	0.82
12	0.81	0.83	0.82
13	0.82	0.9	0.86
14	0.82	0.9	0.86
15	0.79	0.72	0.75
16	0.95	0.8	0.87
accuracy			0.81
macro avg	0.81	0.81	0.81
weighted avg	0.81	0.81	0.81

Figure 9: Classification report of the new transferred model

Moreover, we printed a classification report which includes again important metrics for our task, for each dog class now. The results remain very good, almost equal to the first training on the dogs dataset.

To gain further understanding of the model’s performance, we generated a confusion matrix using the predictions of the transferred model on the test dataset and there was no bias towards a specific class.

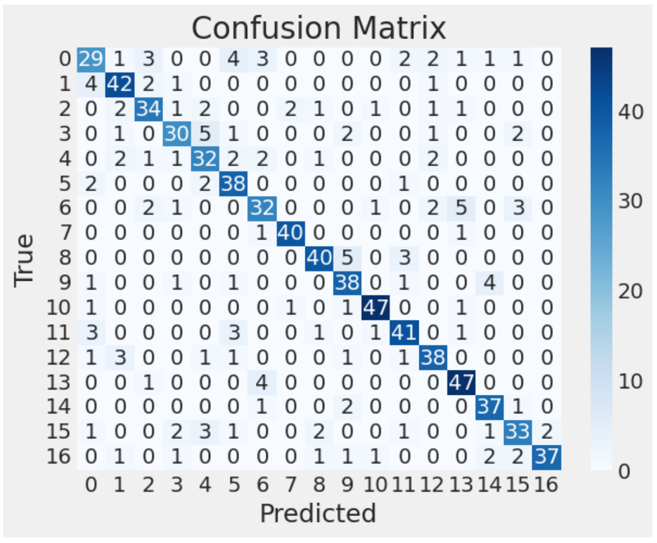


Figure 10: Confusion Matrix of the new transferred model

Consequently, by evaluating the results of both transfer learning processes (dogs → cats and cats → dogs) we can conclude that our model generalizes very well in new data since training just the output layer, we achieved to get accuracy over 50% that become even better after training few more layers, almost as good as the original.

7. DEMO

The Demo is an interactive application with a graphical user interface (GUI) that enables users to classify images of cats and dogs according to their breeds, respectively. The application is developed using the Tkinter library and uses our pre trained models to make predictions.

When the user drops an image into the designated area of the GUI, the image is loaded, resized, and displayed on a canvas. The user can then click the *Make Prediction* button to classify the image firstly. The application uses a binary classification model to determine whether the image is of a cat or a dog. Depending on the result, it then uses either our cat breed classification model or a dog breed classification model to predict the breed of the animal in the image. The predicted breed is then displayed on the GUI, with the green color indicating that the prediction is correct while the red color not.

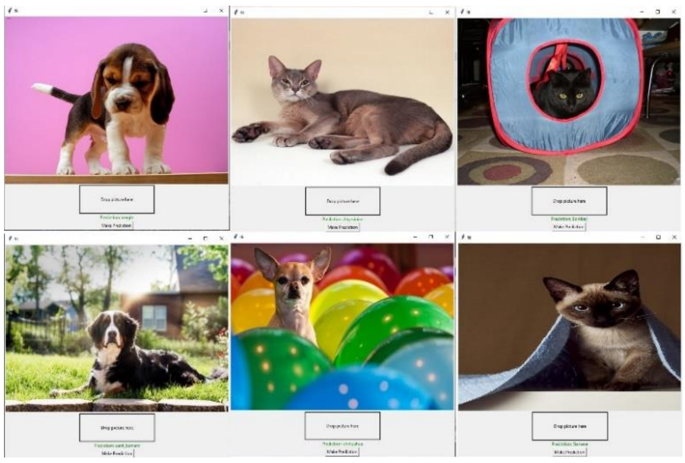


Figure 11: A collection of random images from the internet along with their corresponding predicted labels

8. CONCLUSION

To conclude, our study demonstrates the potential of transfer learning and deep learning techniques to enhance the accuracy of cat breed classification by leveraging knowledge gained from dog breed classification, and vice versa. Our experiments with varying the number of frozen layers and the learning rate showed promising results in improving the performance of the model. What’s more our cross dataset evaluation provided valuable insights into the importance of dataset size in transfer learning. In future work, we plan to explore the use of other deep learning model structures, such as residual networks, to compare the performance of very different structures. To sum up, our findings can contribute to expanding the knowledge in the research of different animal species transfer learning and its potential application across very diverse domains.

9. ACKNOWLEDGMENTS

This work was supported by the Principal Researcher of Multimodal Machine Learning, Giannakopoulos Theodoros, at the Institute of Informatics and Telecommunications of National Center for Scientific Research "Demokritos". For more details, please visit the website of the Cross-Institutional MSc programme in Artificial Intelligence, https://msc-ai.iit.demokritos.gr/

REFERENCES

[1] Goodellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning (Adaptive Computation and Machine Learning series). The MIT Press.

[2] Chollet, F. (2017). Deep Learning with Python. Manning Publications.
[3] Yang, Q., Zhang, Y., Dai, W., & Pan, S. J. (2020). Transfer Learning. Cambridge University Press.
[4] Wang, J., & Chen, y. (2023). Introduction to Transfer Learning: Algorithms and Practice. Springer
[5] Sarkar, D., Bali, R., & Ghosh, T. (2018). Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras. Packt Publishing Ltd.