Cat Breed Classification Using Convolutional Neural Networks

Willy Fitra Hendria
willyfitrahendria@gmail.com
February 2020

I. Definition

A. Project Overview

Classifying cat breeds can be a difficult task for a nonexpert since cats have a variety of breeds, which come in a variety of colors, patterns, facial structure, fur length, etc. By building a machine learning model to predict the breed of a cat based on a single image, hopefully it can be helpful to identify the cat breeds more easily.

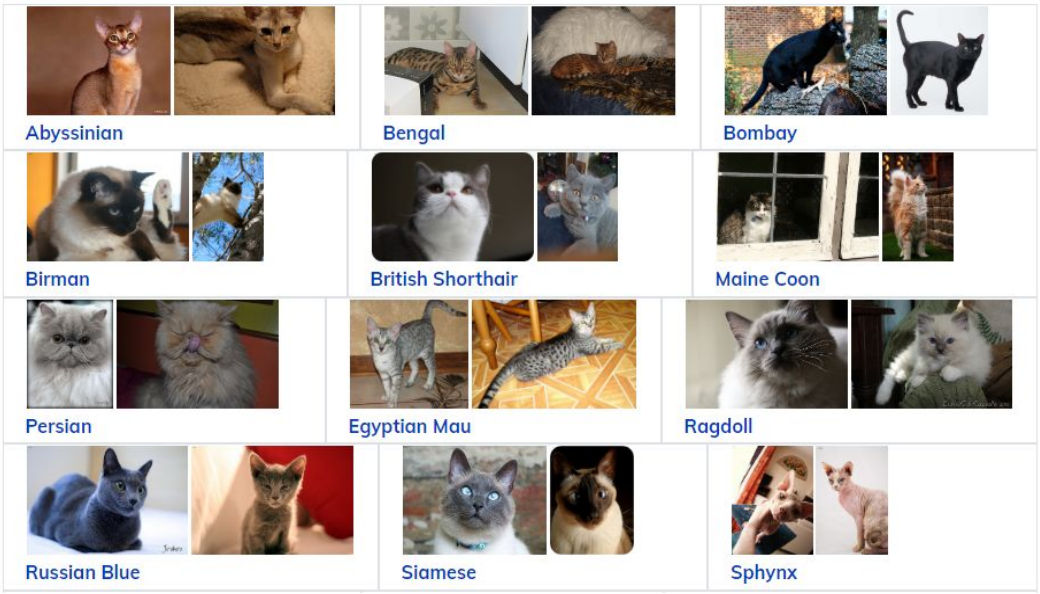

Fig. 1. Sample of labeled cat breeds in the *Oxford-IIIT Pet Dataset* [1]

In this project, I built a machine learning model capable of classifying images of cat breeds on the *Oxford-IIIT Pet Dataset* [1]. The convolutional neural networks (CNNs) model built in this project has a better accuracy compare than a previous model created by Parkhi et al. [2]. Transfer learning is performed to transfer the knowledge of a previously trained CNN in order to get a better accuracy.

B. Problem Statement

Considering there are lots of cat breeds, identifying the breeds can be difficult for a nonexpert. And considering the features among different breeds can be similar, building an image classifier for cat breed classification can be a challenging task. It can be difficult to do the feature

engineering manually since the shape, color, pattern, or facial structure among different breeds can be similar. Incorrectly selecting the features can significantly affect the accuracy of the machine learning model.

This project solves cat breed classification's problem by using CNNs to classify cat breeds from labeled images of cats. CNNs are able to automatically extract the image features, and also able to transfer the knowledge of the previously trained CNNs.

Following are the tasks involved in this project:

- Download the Oxford-IIIT Pet Dataset, and filter out the dog images
- Load and split the dataset into training and testing folders
- Explore the distribution of classes in both training and testing data
- Transform the images to have the same scale and normalization with the pretrained models for transfer learning
- Train and tune hyperparameters of a simple-CNN from scratch using grid search with cross validation, and evaluate the the model on the testing data.
- Train and tune hyperparameters of pretrained ResNet-50 using grid search with cross validation, and evaluate the model on the testing data
- Add image augmentations for training the pretrained model, and evaluate the model
- Predict some sample of images using the best model found

The final model is expected to have a better accuracy compare than the previous study by Parkhi et al. [2] which only has 66.12% as the best accuracy. The model should be able to identify a cat breed given an image of a cat.

C. Metrics

Accuracy is one of good choices to evaluate a fairly balanced dataset. It is necessary to evaluate the accuracy since we want to compare the result with the previous study. In addition, top-3 accuracy is also used since we want to know how good a model is in providing some suggestions. In the top-3 accuracy, a prediction is true when the correct class is within the top three predicted classes.

Accuracy = sum of true predictions / number of predictions.

F1-score or F-score combines precision and recall [3] in one metric. It is a better metric to evaluate imbalanced class distribution. Since our dataset is not perfectly balanced (as we will explore later), this metric is also used to evaluate the models.

F-score = 2×precision×recall / (precision + recall)

We will also check the confusion matrix to identify if the model has some difficulties to classify some of the classes. In the confusion matrix, the number of correct and incorrect predictions for each class are summarized in the form of count matrix.

II. Analysis

A. Data Exploration

The Oxford-IIIT Pet Dataset contains 37 classes of dogs and cats with roughly 200 color images for each class, which consists of 12 cat breeds and 25 dog breeds. Here, only the cat images will be used for the training and the evaluation.

Based on the provided metadata files, splitting the cat images produces 1188 images for training and 1883 images for evaluation. It's not a common splitting ratio i.e 70/30 or 80/20, but this project will just follow the provided splits since we want to compare the evaluation result on the same data of the previous study. After the splitting, 29 cat images are not being splitted to either training or testing dataset. The unused images are ignored as it is expected.

Fig. 2. Sample images of a specific cat breed in the *Oxford-IIIT Pet Dataset*, which has variations in scale, pose, lighting, and background

The dataset has variations in scale, pose, lighting, and background, which are well-known challenges of computer vision problem. Since the scale of the images are not equal, resizing and cropping the images to have the same scaling is necessary.

B. Exploratory Visualization

As can be seen in Fig 3 and Fig 4, even though not perfectly balanced, the class distribution of both training and testing data are fairly balanced. Thus, we don't really need to work on the approach of imbalanced data, e.g. sampling.

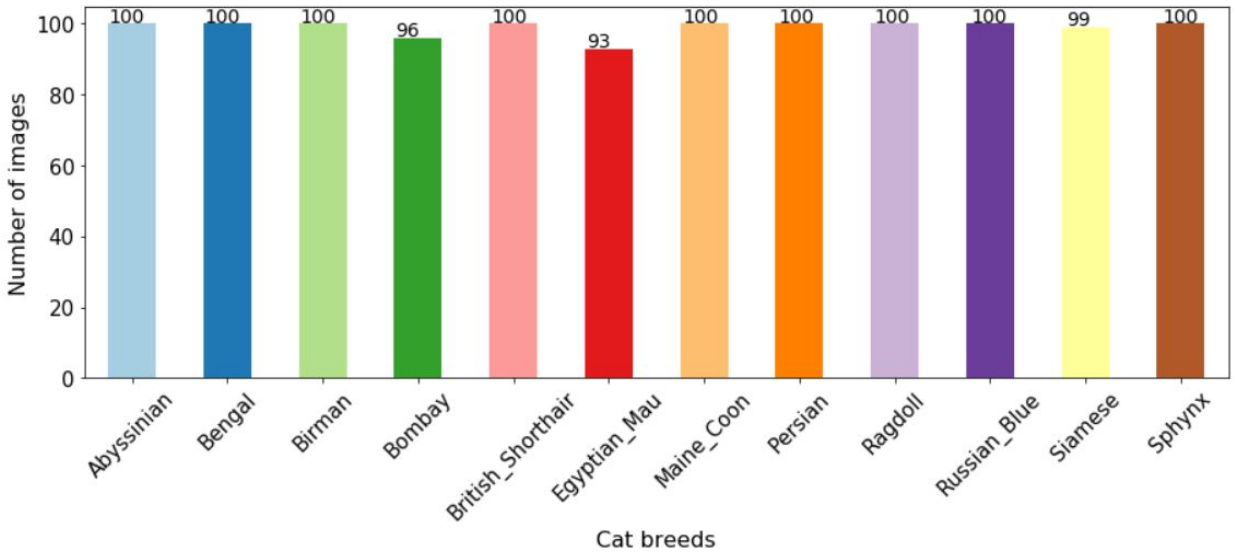

Fig. 3. Class distribution of training data

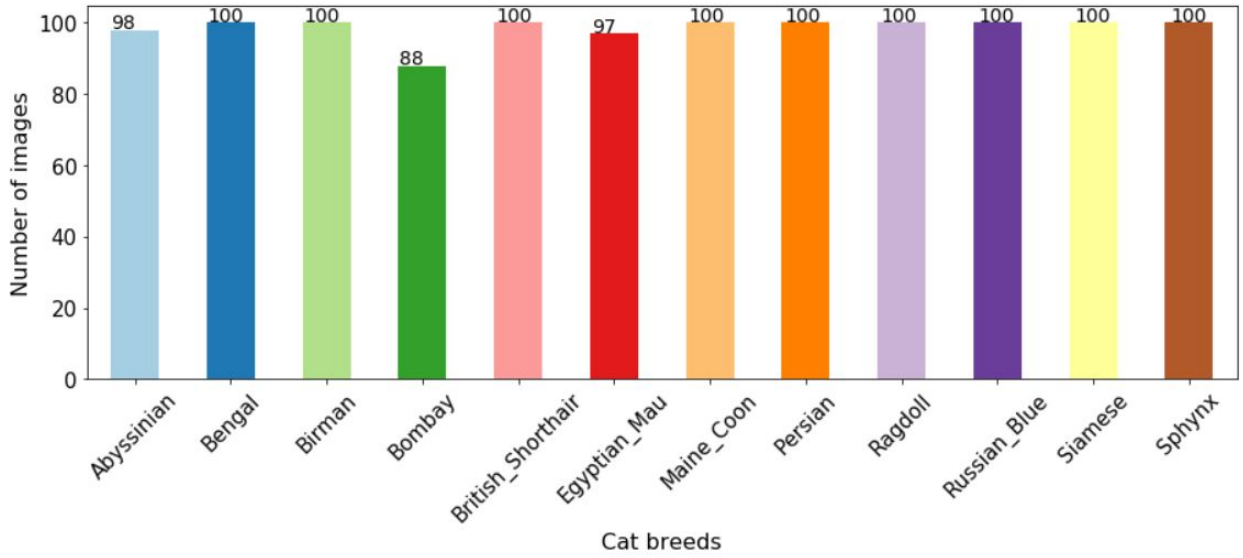

Fig. 4. Class distribution of testing data

C. Algorithms and Techniques

A convolutional neural network (CNN) is a class of deep learning, that has been proven to be very successful in image classification tasks. In CNNs, the image features are learned automatically in the process of training. When multiple classes share many similar features as in this cat breed classification problem, it can be difficult to engineer the features manually. Here, CNNs have potential to flexibly extract the features. Besides, by using CNNs, transfer learning can also be performed to transfer the knowledge of previously trained CNNs. Hussain et al. [4] found transfer learning could improve the accuracy of an image classification task.

Hyperparameters are the variables set before training, which determine the network structure and also how the network is trained. Following are a few common hyperparameters usually be tuned in a deep neural network:

- Learning rate
- Momentum
- Weight decay
- Number of hidden layers
- Number of hidden units for different layers
- Activation function
- Batch size
- Number of epochs

In this project, *learning rate* and *batch size* are tuned by using Grid Search method. Grid Search generates parameters by using the exact combination of parameters in the grid. Grid Search method is chosen because we want to investigate the exact combination of some specified hyperparameters with respect to the validation accuracy.

To evaluate the hyperparameter tuning, Cross Validation method is used for each combination of parameters. Cross Validation is good to reduce risk of overfitting, and also good to provide an unbiased estimation. Here, Stratified Cross Validation is used to ensure each class distribution is (approximately) equal in all the folds. Other than learning rate and batch size, the 'best' number of epochs is also decided based on the result of Cross Validation.

D. Benchmark

For the benchmark, we will use the accuracy result (Table 1) from a paper entitled *Cats and Dogs* [2], which was evaluated using the *Oxford-IIIT Pet Dataset*. The evaluated models were built using some engineered features based on the combination of shape, appearance, and segmentation. As can be seen in Table 1, the best accuracy achieved was 66.12%, and this will be used as the benchmark.

Table 1. Comparison of cat breed classification accuracy (yellow box) among different models defined in a paper entitled *Cats and Dogs* [2]

.	Shape	Appearance		Classification Accuracy (%)				
		layout type	using ground truth	family	breed (S. 4.2)		both (S. 4.3)	
				(S. 4.1)	cat	dog	hierarchical	flat
1	✓	–	–	94.21	NA	NA	NA	NA
2	–	Image	–	82.56	52.01	40.59	NA	39.64
3	–	Image+Head	–	85.06	60.37	52.10	NA	51.23
4	–	Image+Head+Body	–	87.78	64.27	54.31	NA	54.05
5	–	Image+Head+Body	✓	88.68	66.12	57.29	NA	56.60
6	✓	Image	–	94.88	50.27	42.94	42.29	43.30
7	✓	Image+Head	–	95.07	59.11	54.56	52.78	54.03
8	✓	Image+Head+Body	–	94.89	63.48	55.68	55.26	56.68
9	✓	Image+Head+Body	✓	95.37	66.07	59.18	57.77	59.21

III. Methodology

A. Data Preprocessing

Before the images loaded into memory, the dog images and non-image files are filtered out from ‘images’ folder. The cat images are splitted into ‘train’ and ‘test’ folder based on the provided metadata. Each image is placed into a subfolder corresponding to its own class.

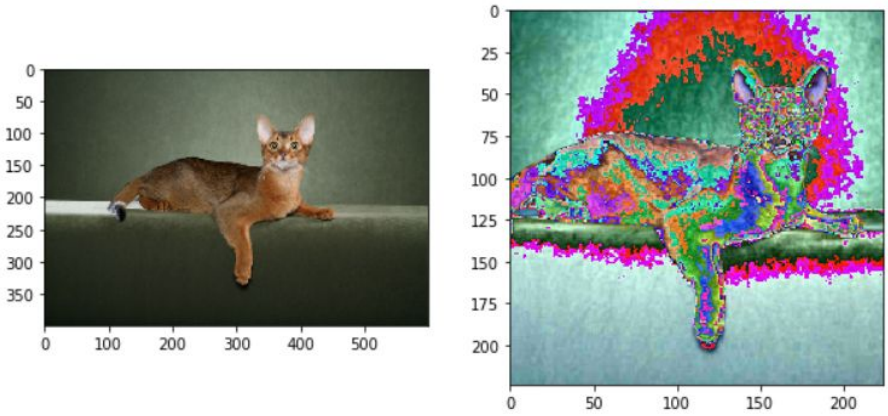

Fig. 5. Before (left) and after (right) resizing, cropping, and normalization applied to an image

After the images loaded, the images are resized and cropped to 224x224 pixels, converted to tensor, and normalized with mean and standard deviation. The transformations are necessary in order to use a pretrained model for transfer learning.

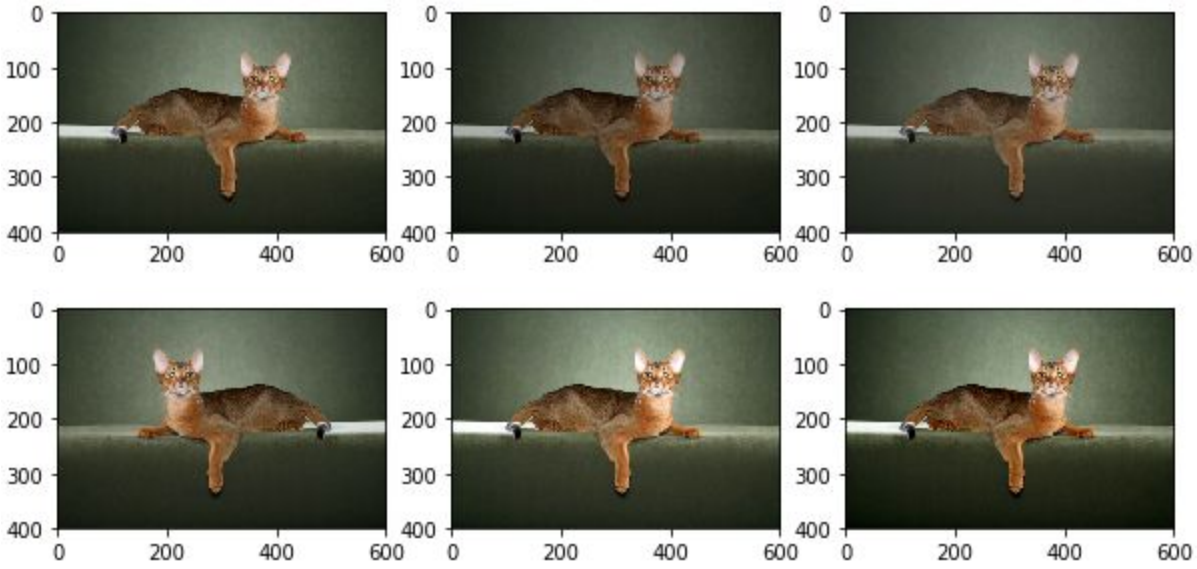

Fig. 6. Original (top-left), darker (top-mid), lower contrast (top-right), horizontal flipped (bottom-left), brighter (bottom-mid), and higher contrast (bottom-right) images

In order to improve the performance, some image augmentations are applied, such as random horizontal flip, random brightness, and random contrast transformation. As can be seen in Fig.

6, the transformations still produce valid images. We are still able to tell the transformed images similar to the original image.

B. Implementation

1) Convolutional Neural Networks

The convolutional neural networks (CNNs) in this project are implemented using PyTorch library. The initial model is a simple vanilla CNN with one convolutional, one max pooling layer, and one fully connected layer. The common ReLU activation function is used for the hidden layers, and LogSoftMax is used for the output layer. The loss function is using NLLLoss. Thus, a combination of LogSoftMax and NLLLoss is equal to CrossEntropyLoss implementation. Besides, 50% dropout regularization is applied before the fully connected layer. The dropout is used for reducing overfitting and improving the generalization.

In this project, ResNet-50, is used in the transfer learning. A ResNet is a convolutional neural network (CNN) architecture which can support very deep convolutional layers with a strong performance, by utilizing skip connections, or shortcuts to jump over some layers.

For the transfer learning, all of the model parameters are freezed, and the final layer of the models are changed to have the same number of output as the classes of the problem of this project, which is 12 classes of cat breeds.

2) Hyperparameter Tuning

The implementation of hyperparameter tuning is utilizing ParameterGrid of sklearn library. The ParameterGrid is used to iterate over parameter value combinations so that we can use it for implementing Grid Search optimization technique. For the grid, we try different combinations of batch size with value 32 and 64, and learning rate with value 0.001 and 0.0001. The best model is the model which has the lowest cross validation loss among all combinations of the hyperparameters.

3) Cross Validation

StratifiedKFold of sklearn library is used to implement the Cross Validation. It provides the iterator of cross validation by providing both train and valid indices to split the data. Since it is a stratified version, the distribution of each class across all folds is preserved. StratifiedKFold requires a list of target class as a parameter of its function. But ImageFolder dataset of PyTorch provides a list of image and class tuples. Thus, it is necessary to generate the list of the target class from the ImageFolder dataset, in order to use the StratifiedKFold of sklearn. In this project, the number of folds used is 5, and the metrics computed at the cross validation are average loss and average accuracy across all folds.

C. Refinement

In my initial attempt, a simple CNN trained from scratch achieved 27.0499% accuracy, 26.8971% average f-score, and 56.9738% top-3 accuracy on the test data. The best parameters

found from the hyperparameter tuning via grid search and cross validation for this model are 64 for the batch size and 0.0001 for the learning rate, with 28.3670% validation accuracy and 99.7895% training accuracy in 10 epochs.

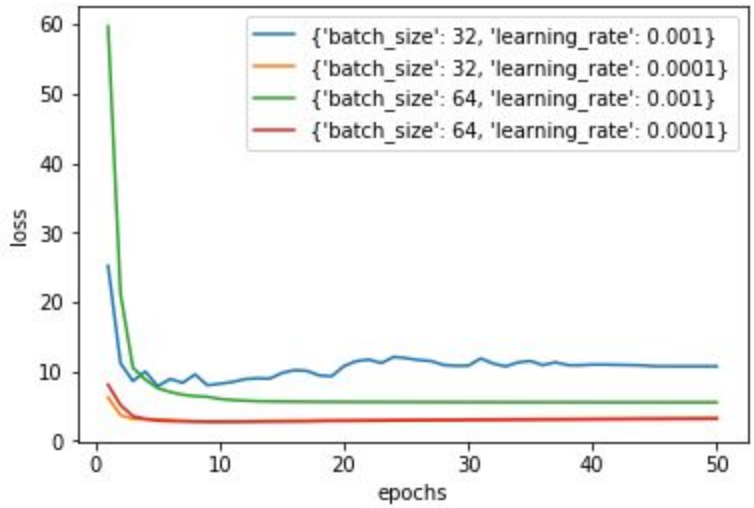

Fig. 7. SimpleCNN - Validation loss per epoch of all evaluated hyperparameters

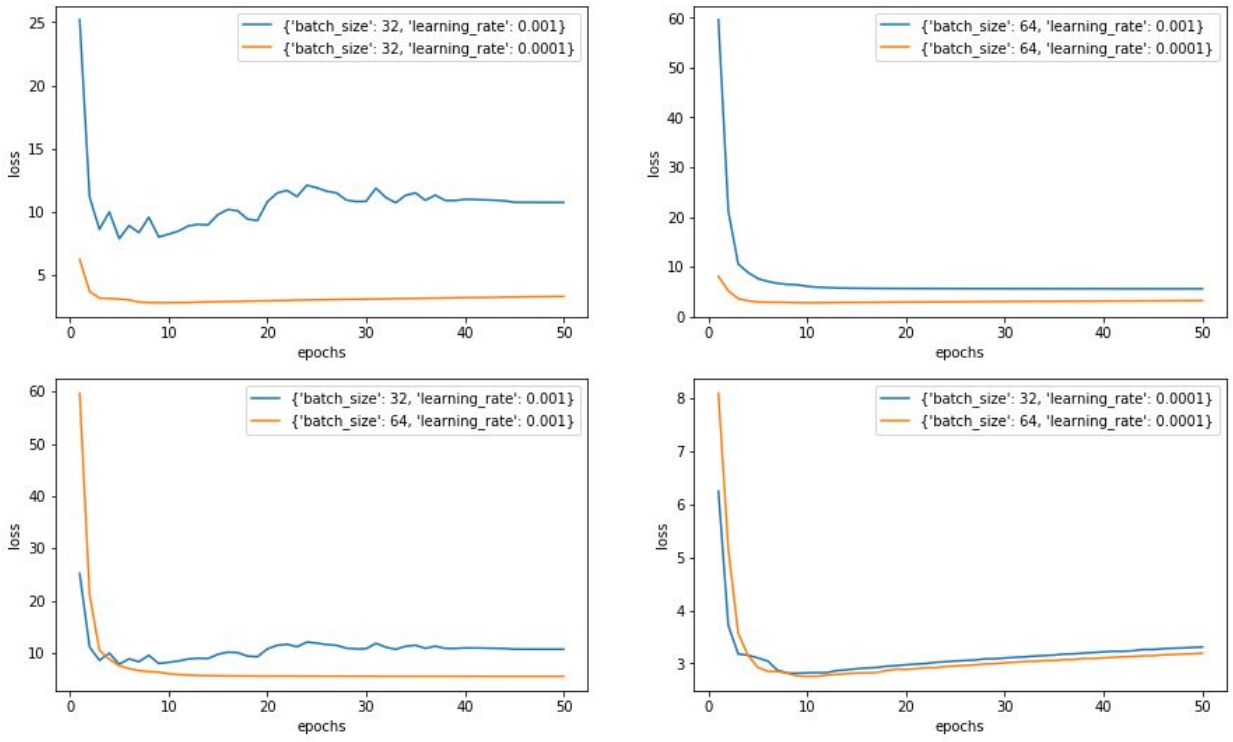

Fig. 8. Simple CNN - Validation loss per epoch of two hyperparameter combinations

As can be seen in Fig. 8 at the bottom-right plot, the validation loss of the best parameters found is starting to increase at 10 epochs; it is a sign of overfitting. It is easy to overfit the data

since we don’t have enough data to train the neural network from scratch. So transfer learning can be a solution for this problem.

The initial model was improved after using the pretrained model of ResNet-50 for transfer learning. It achieved 86.2214% accuracy, 86.1862% average f-score, and 97.2105% top-3 accuracy on test data. The best parameters found from the hyperparameter tuning via grid search and cross validation for this model are 32 for the batch size and 0.001 for the learning rate, with 90.2357% validation accuracy and 98.6322% training accuracy in 25 epochs. It achieved much better performance (Fig. 9, Fig. 10) than the initial model.

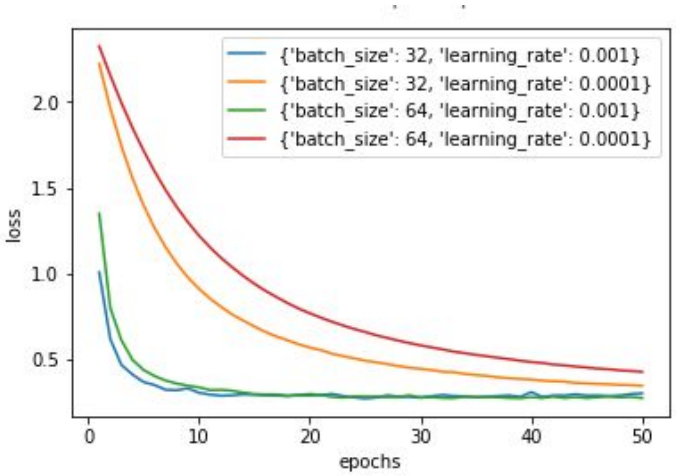

Fig. 9. ResNet-50 - Validation loss per epoch of all evaluated hyperparameters

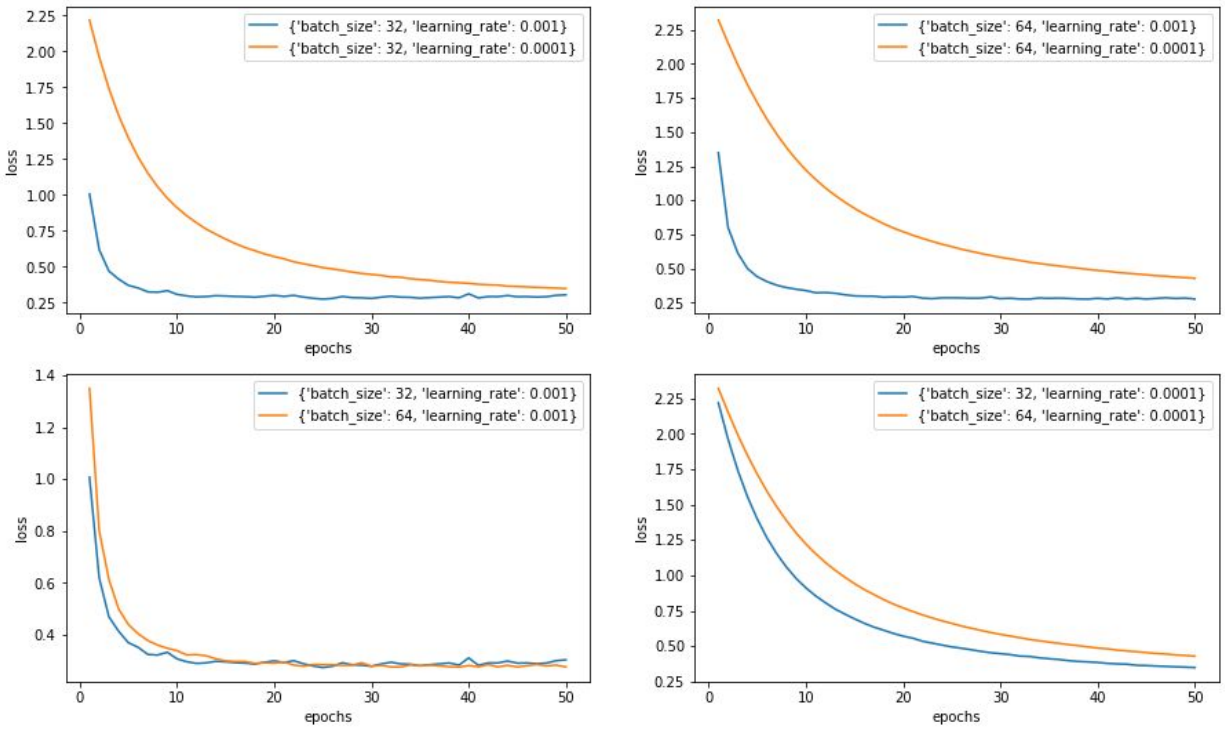

Fig. 10. ResNet-50 - Validation loss per epoch of two hyperparameter combinations

The pretrained model was slightly improved after adding image augmentations. It achieved 87.6585% accuracy, 87.6553% average f-score, and 97.2105% top-3 accuracy on test data. The best parameters found from the hyperparameter tuning via grid search and cross validation for this model are 64 for the batch size and 0.001 for the learning rate, with 90.2357% validation accuracy and 99.6633% training accuracy in 38 epochs. Therefore, this model is selected as the final model used for making predictions.

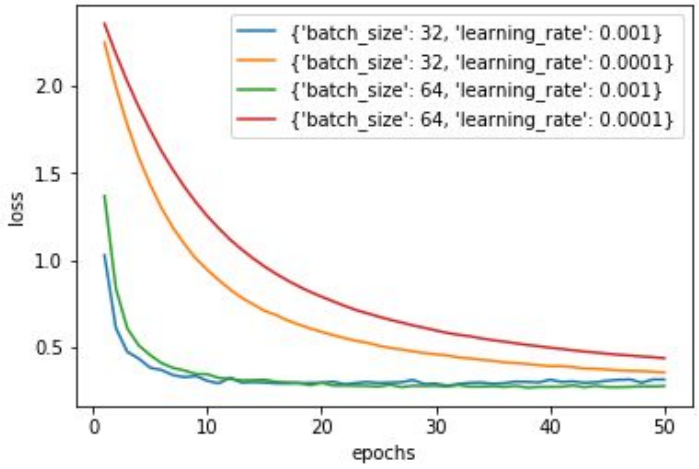

Fig. 11. ResNet-50 with image augmentation - Validation loss per epoch of all evaluated hyperparameters

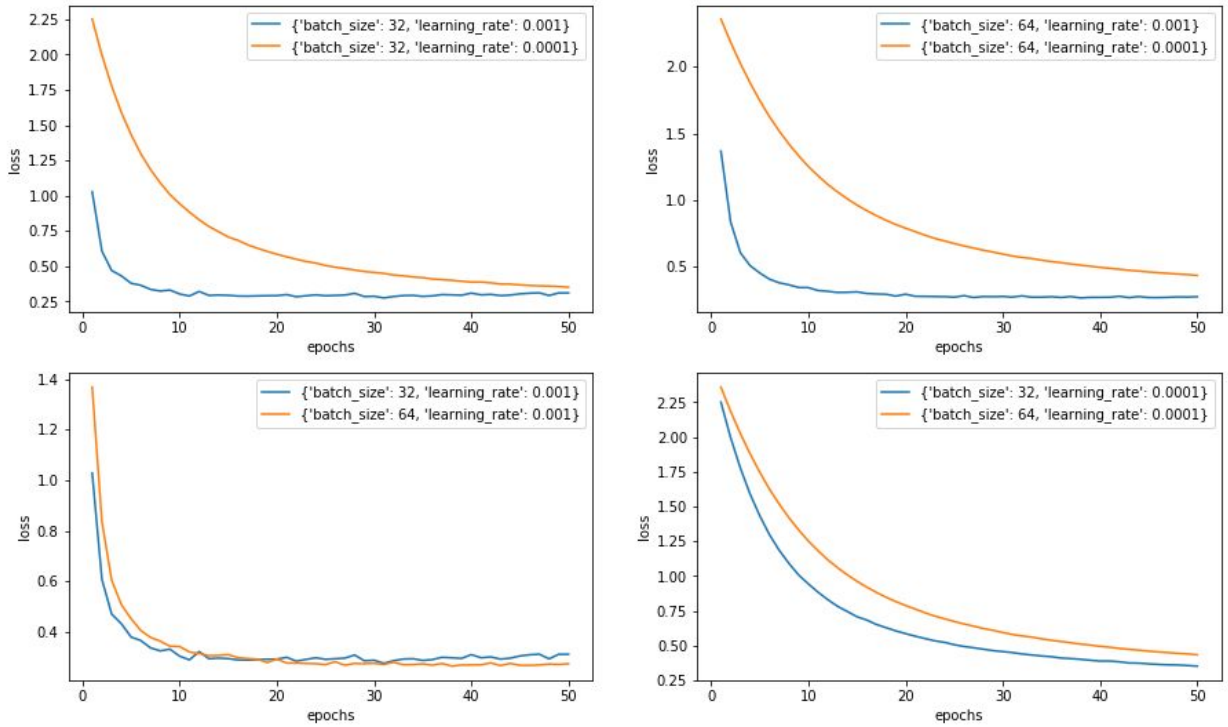

Fig. 12. ResNet-50 with image augmentation - Validation loss per epoch of two hyperparameter combinations

IV. Results

A. Model Evaluation and Validation

For every defined architecture, both grid search and cross validation are used to tune the hyperparameters and to evaluate the models. The best parameters found are used to retrain the model on a full train data. Then the trained model is evaluated on the testing data. The model which has the best accuracy on the testing data is chosen as the final model.

Table 2. Evaluation results of different models

	Model	Accuracy (%)	Average F-score (%)	Top-3 Accuracy (%)
1	Benchmark Model	66.12	-	-
2	Simple CNN (Scratch)	27.0499	26.8971	56.9738
3	ResNet-50 (Pretrained)	86.2214	86.1862	97.2105
4	**ResNet-50 (Pretrained) + Augmentation**	**87.6585**	**87.6553**	**97.2105**

As can be seen in Table 2, the final model (ResNet-50 with augmentation) has a better accuracy than our benchmark model. Simple CNN trained from scratch couldn't achieve a better accuracy because the data is not enough to train a neural network from scratch. To overcome the problem, transfer learning technique by using pretrained ResNet-50 is used. The result is significantly better than both the simple CNN and the benchmark model.

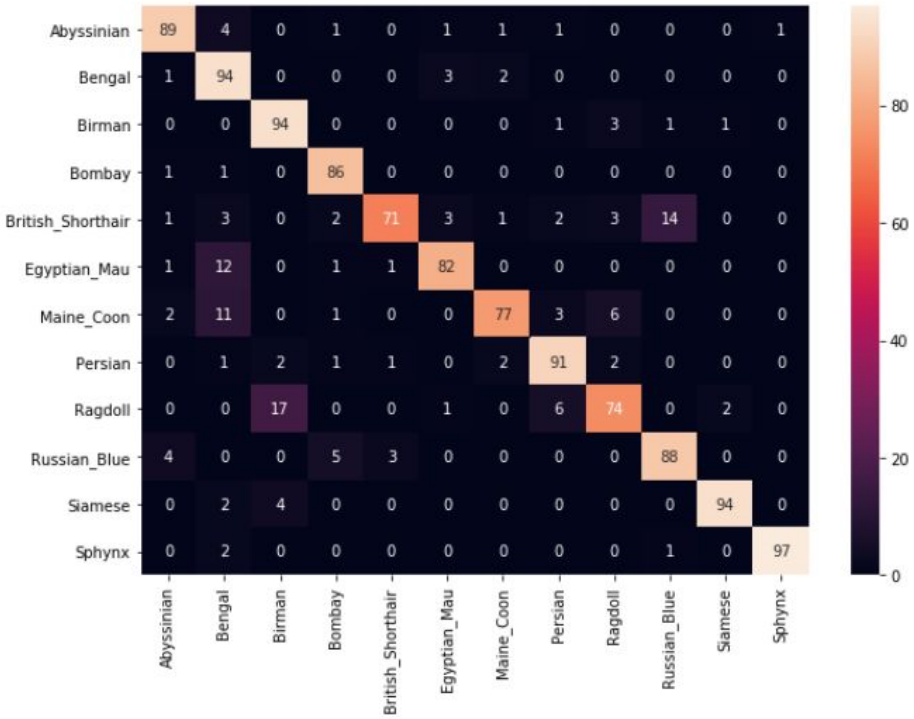

Fig. 13. ResNet-50 with image augmentation - Confusion matrix

By looking at the confusion matrix (Fig. 13), the final model has some difficulty in predicting some classes. The top 3 misclassified classes (Fig 14), ranked by number of misclassifications, are Ragdoll predicted as Birman, British Shorthair predicted as Russian Blue, and Egyptian Mau predicted as Bengal.

	actual	prediction	count
90	Ragdoll	Birman	17
52	British_Shorthair	Russian_Blue	14
56	Egyptian_Mau	Bengal	12

Fig. 14. ResNet-50 with image augmentation - The top 3 misclassified classes (ranked by number of misclassifications)

Despite having some misclassifications, the performance of the final model at each class is good enough (better than random guess). As can be seen in Fig 15, Ragdoll has the lowest f-score of 78.7234%, and Sphynx has the highest f-score of 97.9798%.

	fscore
Abyssinian	0.903553
Bengal	0.817391
Birman	0.866359
Bombay	0.929730
British_Shorthair	0.806818
Egyptian_Mau	0.877005
Maine_Coon	0.841530
Persian	0.892157
Ragdoll	0.787234
Russian_Blue	0.862745
Siamese	0.954315
Sphynx	0.979798

Fig. 15. ResNet-50 with image augmentation - F-score per class

To check the robustness of the final model, we transformed an image of a specific class which was correctly predicted, and predicted the class of the transformed image. The transformations applied are rotation, flipping, noise addition, brightness changes, and contrast changes.

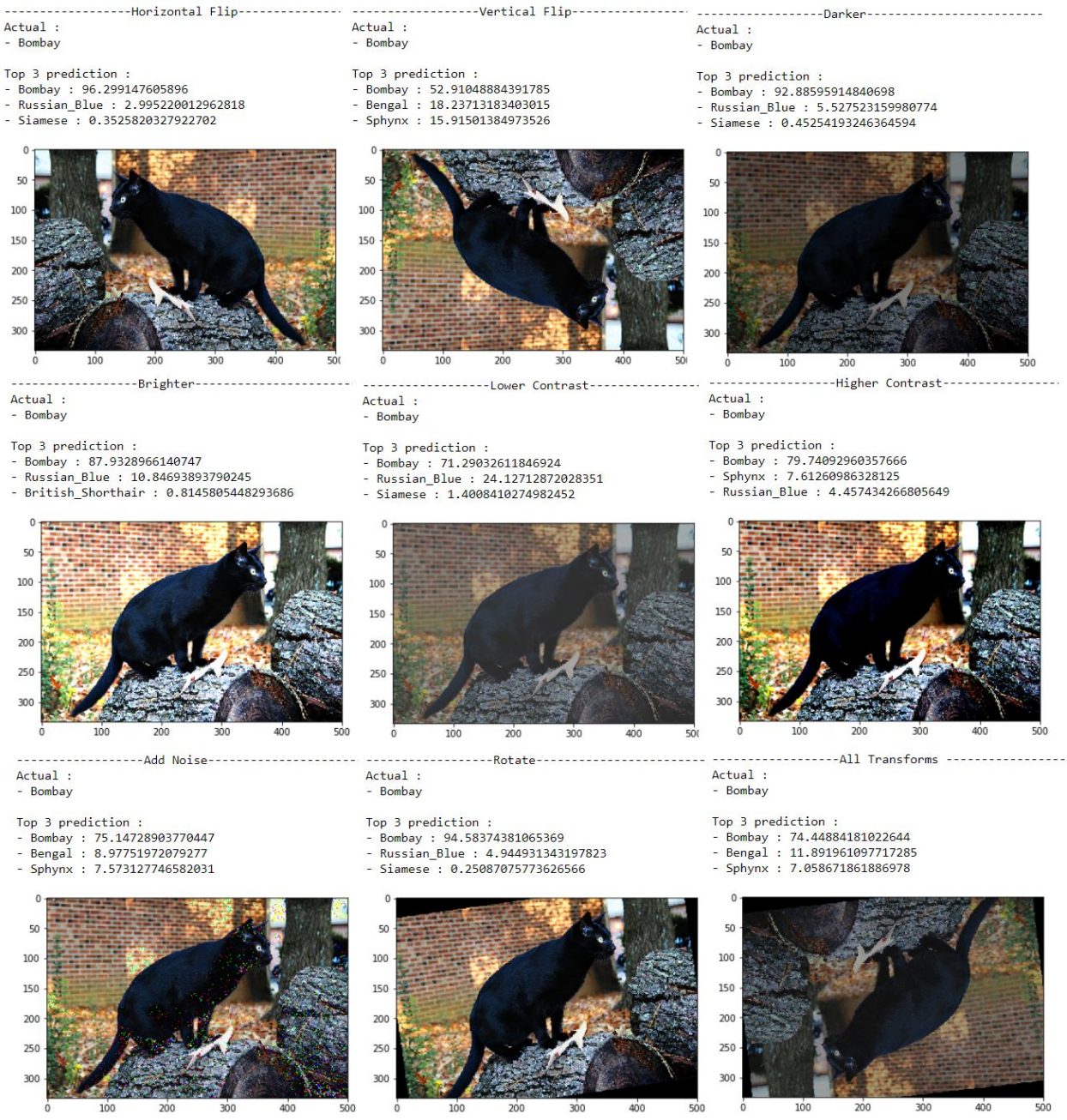

Fig. 16. Prediction results on transformed images of Bombay breed

Even after applying the transformations, the class of a sample image is still correctly predicted. All transformed images of Bombay are correctly predicted as Bombay (Fig 15.).

B. Justification

As mentioned before, the final result of this project has a better accuracy than the benchmark model (Table 2). So this model is better than the benchmark model for helping a nonexpert to identify a breed of a cat from an image. Even though the model has some difficulties in

predicting some breeds (Fig. 14), it has no significant problem on predicting most of the breeds. Even the class which has the lowest f-score is still making a good prediction (Fig. 15). The model is also having a high top-3 accuracy (Table 2), so it can be trusted enough to provide the top 3 recommendations of a correct breed. Besides, the model is robust enough on some variation of transformed images.

Currently the International Cat Association (TICA) recognizes a total of 71 cat breeds [5]. But this model only trained on 12 breeds dataset.. Thus, this model only useful for a limited range of breeds. A dataset with a complete cat breeds is needed for training a machine learning model in order to significantly help a nonexpert identifying most of the cat breeds.

V. Conclusion

A. Free-Form Visualization

By looking at some of the misclassified predictions (Fig. 17) from the top 3 misclassified classes (Fig. 14), it is understandable that the model was confused because the breeds look similar. Despite having the problem, the actual labels are still within the top 3 predictions.

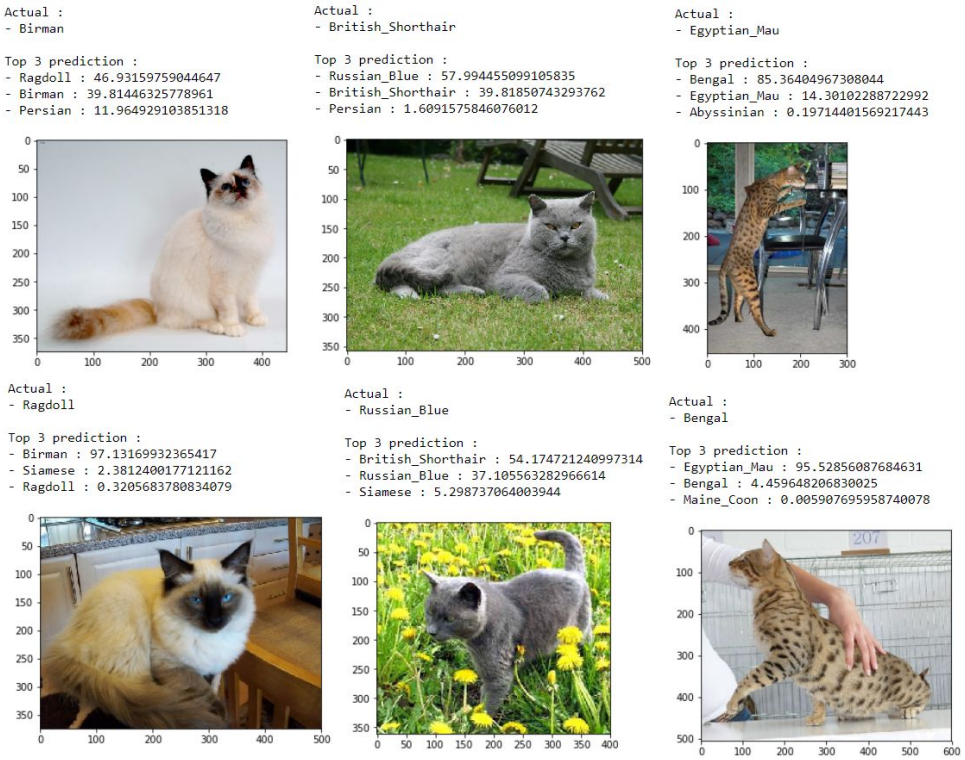

Fig. 17. Prediction results of the top misclassified classes. Birman predicted as Ragdoll (top-left), Ragdoll predicted as Birman (bottom-left), British Shorthair predicted as Russian Blue (top-mid), Russian Blue predicted as British Shorthair (bottom-mid), Egyptian Mau predicted as Benga (top-right)l, and Bengal predicted as Egyptian Mau (bottom-right)

B. Reflection

In summary, the entire process of this project is following below steps:

1. Research the domain background and problem, including the relevant research and dataset
2. Investigate some solutions for a similar type of problem
3. Preprocess the selected dataset
4. Build and tune the models
5. Evaluate the results
6. Create the documentation

I found that the 4th step is the most interesting and the most difficult at the same time. Building the models requires a lot of research, including hyperparameter tuning, cross validation, transfer learning with the pretrained models, training on GPU, and the implementation in PyTorch.

CNN with transfer learning showed a good result for this cat breed classification problem. As a state of the art in image classification, CNN worked as expected.

C. Improvement

The model built in this project only works for a limited range of breeds. It's not able to identify other than the 12 breeds in the dataset. Using a dataset which contains the all of 71 recognized breeds will make the cat breed classifier more useful.

Investigating more hyperparameter combinations and architectures may improve the accuracy. Besides, adding more training data and carefully checking incorrectly labeled images may also improve the accuracy.

As a future work, multi-label classification can be considered for this cat breed classification. There could be more than one breed in an image that needs to be identified by a machine learning model.

References

1. The Oxford-IIIT Pet Dataset, Department of Engineering Science, University of Oxford, 2012. [Online]. Available: https://www.robots.ox.ac.uk/~vgg/data/pets/
2. Parkhi, O.M., Vedaldi, A., Zisserman, A., and Jawahar, C.V.,"Cats and Dogs," in CVPR, 2012
3. Scikit-learn Developers. "Precision-Recall." scikit-learn. Available: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html (accessed Jan. 30, 2020)
4. Hussain, M., Bird, J.J., and Faria, D.R, "A Study on CNN Transfer Learning for Image Classification," in UK Workshop on Computational Intelligence, 2018
5. The International Cat Association. "Home Page." TICA.org. Available: https://tica.org (accessed Jan. 18, 2020)