Ensemble Learning: A Comparative Study of Performance on Small Datasets

Mahan Mohammadi **Rosie Wang** **Youyang Li**

Abstract

This project aims to improve the accuracy and robustness of predictive models for cat breed images classification by exploring the effectiveness of ensemble-based classification techniques. The approach involves taking the CLIP, our custom-created CNN, pre-trained ResNet34, pre-trained DenseNet and pre-trained ResNeXt32 models and running each of them on large cat breeds images dataset and comparing their individual performances with each other. Then, the three best-performing models among those models are chosen for ensembling on small cat breeds images dataset using combined bagging and stacking techniques of ensembling. By comparing the performance of training on a smaller dataset with ensembling and training each of those models individually on larger datasets, our results demonstrated that ensemble learning has a favorable impact on the classification outcome even when using smaller dataset.

1 Introduction

1.1 Background and Context

Ensemble-based classification techniques have gained popularity in machine learning as a way to improve the accuracy and robustness of predictive models. Ensemble methods involves combining the predictions of multiple models, each of which has been trained on the same dataset, in order to produce a more accurate and reliable prediction than any single contributing model. Previous studies have shown that ensemble learning has been successfully applied in a wide range of domains including image classification, which will be the focus of this project [1].

1.2 Motivation

Ensemble learning combines the strengths of multiple models to reduce variance, improve accuracy, and increase stability in predictions. It is especially useful for handling complex or diverse datasets, and can address issues such as class imbalance and noisy data. Ensemble learning is also valuable for training with small datasets, which may produce biased or incomplete models due to a lack of representative samples or high variability. However, larger datasets tend to produce more accurate and robust models at the cost of longer training times and increased computational expense. In this project, we used ensemble learning investigate its effectiveness on small datasets by combining the results of multiple models. This provides a simple framework that can be easily extended to other datasets, and a starting point for those who wish to further explore ensemble learning without the need to assemble everything from the ground up. The results were then compared to those achieved by training a single model on a larger dataset.

1.3 Research Gap and Objective

This project aims to classify cat breeds by combining pre-trained DenseNet, pre-trained ResNet34, and pre-trained ResNeXt32 models. The proposed framework demonstrates the effectiveness of ensemble learning on small datasets and can be extended to a wider range of datasets. There is a wide variety of datasets for cats available, making it a desirable choice for the purpose of evaluating the effect of the

ensemble method. These pre-trained models are commonly used for image classification purposes. DenseNet is based on the idea of "dense connections". ResNet on the other hand, introduced residual connections. ResNeXt is an extension of ResNet that introduces the concept of group convolutions, which can improve the representational power of the network without significantly increasing the number of parameters. In addition, CLIP and a custom CNN model were also trained on the large dataset, but were ruled out for ensemble learning due to their poor performance.

2 Related Works

There have been several studies pertaining to transfer learning and ensemble models that motivated our project. One study proposed using transfer learning through an ensemble model with two existing architectures (ResNet50 and VGG16) to classify sheep breeds in a farm in Oceania [2]. Another study of the Diversified Ensemble Neural Network (DENN), which is an ensemble of multiple neural networks, each with different architectures, showed that ensemble networks can significantly improve the accuracy and stability of many machine learning tasks (e.g., text sentiment analysis, image classification) on small datasets [3]. This is also supported in another study that showed that ensemble models can achieve similar results as the state of the art models with these lowered costs [4]. Finally, a study proposed the use of hyper-deep ensembles and hyper-batch ensembles to improve the state-of-the-art models used for image classification with lower computational costs, memory costs, and budget costs than conventional ensembles. These studies provide valuable insights into ensemble learning and transfer learning and offer several suggestions for future research.It compares it results with five state-of-the-art transfer learning models (ResNet50, VGG16, VGG19, InceptionV3, and Xception) and finds its ensemble model to be significantly more efficient and accurate than the existing models alone. Finally, a study that directly motivated our project was one that used transfer learning to classify dog breeds and distinguish dogs and wolves. This study evaluated many models (i.e., ResNet101, ResNet50, VGG16, VGG19, DenseNet201, etc), concluding that ResNet101 performed the best. The conclusion of this paper suggested our project as a future study: "In future ensemble models can be applied which can boost the accuracy of wolf and dog breed dataset. Combining two best models can improve the efficiency of the model created." [5]

3 Methods

3.1 Images Preprocessing

For both the small cat breeds images dataset and large cat breeds images dataset, preproceesing is done on the images used for training, validation and test respectively. First of all, for all these images to be consistent and conformant in terms of shape, the following non-augmenting transformations are done to the images:
(1). Necessary operations are done to do `CenterCrop(size=224)`, which crops the center of the image to a size of 224×224 pixels.
(2).Image is converted to a PyTorch tensor. Pixels values of the tensor are normalized like Imagenet standards by having `transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])`
For input images used for training, in addition to the aforementioned non-augmenting transformations, data augmenting transformations are performed on those images as well as follows:
1. `RandomResizedCrop(size=256, scale=(0.8, 1.0))`: This randomly crops and resizes the image to a size of 256 pixels. The crop is taken from a random location within the original image, and its size is determined by scaling the original size between 0.8 and 1.0.
2. `RandomRotation(degrees=15)`: This randomly rotates the image by up to 15 degrees in either direction.
3. `RandomHorizontalFlip()`: This randomly flips the image horizontally with a probability of 0.5.

These transformations help to augment the training data by introducing random variations to the images. This can help to improve the robustness of the models used in this project and reduce overfitting.

3.2 Models Used

3.2.1 CLIP model

The first model we explored is the CLIP model from OpenAI. To adapt it to our problem, a list of tokens is created where each token is a description of the image that corresponds to one of the breeds of cat. Since 8 labels (cat breeds) are considered, there are 8 tokens. In order for the CLIP model to run, these tokens are tokenized and encoded and each image is encoded for each of the 8 labels.

3.2.2 Custom CNN model

The representation of the custom-created CNN model is shown in the **Appendix** A section. A preprocessed image of size $224 \times 224 \times 3$ is inputted into the model, and by going through several blocks of the model, the output label will be obtained.

3.2.3 Pre-trained models

The pre-trained models used are ResNet34 (`resnet34(pretrained=True)`), DenseNet (`densenet161(pretrained=True)`) and ResNeXt32 (`resnext50_32x4d(pretrained=True)`). A linear layer is added to these models to match the number of output labels.

3.2.4 Ensembling method

For ensembling, a version of stacking and bagging are combined.[10] Here is the algorithm box:

Algorithm 1 Ensembling Method

Get $n = 3$ different bootstraps samples (X_1, X_2 and X_3) from the training set of the small cat breeds images dataset
for $k = 1, \ldots, n$ **do**
    $out1$ = Applied `ResNeXt32Pre_Trained()` to X_k
    $out2$ = Applied `PretrainedDenseNet()` to X_k
    $out3$ = Applied `ResNet34Pre_Trained()` to X_k
    out = Concatenated $out1$, $out2$ and $out3$
    Since there are three models, apply `linear`$(3 \cdot 8, 8)$ to out to get the predicted output labels
    v_k = validation accuracy of ensembling method on X_k
    t_k = test accuracy of ensembling method on X_K
end for
return $\frac{v_1+v_2+v_3}{n}$, $\frac{t_1+t_2+t_3}{n}$ ▷ where $n = 3$

Figure 1: Ensembling method

3.2.5 Model Evaluations and Hyperparameters

The cross-entropy loss is used to evaluate the loss for the training phase and validation phase for each of the models. Also, a batch size of 64 is used for all the models. The number of epochs is 10 and Adam optimizer (`torch.optim.Adam`) is used. A maximum learning rate of 0.01 is utilized, and the gradient clip is 0.1. Finally, weight decay is 1e-4. All of these hyperparameters are from the following project done for dog breeds image classification [6].

4 Experiments and Results

4.1 Datasets

The **original** dataset is created using two datasets obtained from Kaggle which are the "Cat Breeds Dataset"[7] and "Cats and Dogs Breeds Classification Oxford Dataset"[8]. To create the **original** dataset, a mixture of images from both Kaggle datasets are taken for the following 8 cat breeds (that correspond to the labels): Abyssinian, Bengal, Birman, Bombay, Egyptian Mau, Persian, Ragdoll and Russian Blue. Now, 40% of the **original** dataset corresponds to the small cat breeds images dataset and the rest corresponds to the large cat breeds images dataset. Each of these two datsets are separated into training/validation/test set. Necessary preprocessing is done for the images in each set. Finally, the data loader is written to do multi-process loading of data into the GPU memory.

4.2 Results

After training each model on the large dataset, it is observed that CLIP and the custom CNN did not perform well for the task at hand. However, RestNet34, DenseNet, and RestNeXt32 all showed promising results and were chosen as candidates for further analysis using ensemble learning methods. Among the three, DenseNet achieved the highest test accuracy of 0.7916.

Table 1: Model Performance

Model					
Name	Training accuracy	Training loss	Validation accuracy	Validation loss	Test accuracy
CLIP	N/A	N/A	N/A	N/A	0.125
Custom CNN	0.3451	1.6083	0.3307	1.6073	0.3199
Pre-trained ResNet34	0.8948	0.3653	0.8351	0.4704	0.7810
Pre-trained DenseNet	0.9383	0.2376	0.8077	0.5340	0.7916
Pre-trained ResNeXt32	0.8961	0.3393	0.8594	0.4428	0.7508

For the ensembling, a version of the stacking technique and bagging technique were combined. Table 2 in the appendix shows the ensembling results on each of the three bootstrap training samples from the training set of the small cat breeds image dataset. Based on this table, the average validation and test accuracies for the three different bootstrap training samples are 0.8864 and 0.7936.

5 Discussion

The result matches the general perception that pretrained models tend to perform better than training a model from scratch. We also found that the CLIP model did not perform well for the cat breed classification task. CLIP model is a complex architecture that uses a combination of text and image inputs. This complexity may make it difficult to fine-tune the model on a relatively small dataset for our specific task, as it may require a large amount of data to adjust the model's parameters effectively.

Pretrained models are trained on large and diverse datasets, often with millions of images, using complex architectures and advanced optimization techniques. This allows the models to learn a wide range of features and patterns from the data. When fine-tuned on the specific cat breed classification task, the pretrained models already have a good understanding of the underlying features and patterns in the data, which can help them perform better than models trained from scratch (like the custom CNN). Additionally, fine-tuning a pre-trained model requires fewer iterations of training, which explains that they achieved a good result with just 10 epochs.

In order to evaluate the effectiveness of ensemble learning, we ensembled (using the combined bagging and stacking technique) those three pretrained models on a smaller dataset that represented $2/3$ of the original dataset. Bagging involves training multiple instances of the same model on different subsets of the training data and averaging their predictions at inference time. Stacking involves training a meta-model on the predictions of the individual models. Based on the obtained results, the ensembling done in this project on the small dataset only increased the validation and test accuracy by a marginal amount compared with the individual model trained on larger datasets. While we expected the results to be more significant, as the models performed well individually, and stacking and training with different bootstrap training samples should lead to a better result, this discovery is similar to previous work done on evaluating the performance of ensemble algorithms, where the authors concluded that the use of bagging, boosting and stacking did not achieve a notable increase in accuracy [9]. The marginal improvement on performance when using ensemble learning could be explained by the fact that individual models are already highly accurate and the cat breed dataset is relatively simple.

6 Conclusion

In this project, the performance of several models was compared and evaluated on both a small and large dataset for cat breed classification. Five different models were explored (i.e., CLIP, a custom CNN model, pre-trained ResNet34, pre-trained DenseNet, and pre-trained RestNeXt32) on the large dataset, and it was observed that the pre-trained models showed the best results. Thus, these models were used in an ensemble process (on the small dataset) to generate an ensemble model combining stacking and bagging techniques. According to our literature review, very few of these specific models or techniques have been evaluated in the same project altogether, especially for the cat breed image classification task. By comparing the performance of the ensemble model trained on the small dataset and that of individual models trained on the large dataset, the results suggest that indeed, ensembling the three pre-trained models using the technique described above can improve the model performance. This further suggests the importance of choosing the right models for the correct dataset sizes and tasks and the effectiveness of ensemble learning in improving model performance. Further research could explore ensemble methods using different models and techniques, the impact of different pre-training datasets, and the impact of tuning different hyperparameters on model performance.

7 References

[1] W. H. Beluch, T. Genewein, A. Nurnberger, and J. M. Kohler, “The power of ensembles for active learning in image classification,” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[2] K. Chaturvedi, “Wolf and dog breed image classification using Deep Learning Techniques,” Wolf and Dog Breed Image Classification Using Deep Learning Techniques - NORMA@NCI Library, 01-Jan-1970.

[3] S. Zhang, M. Liu, and J. Yan NeurIPS Proceedings. “The diversified ensemble neural network - NIPS,” NeurIPS Proceedings.

[4] F. Wenzel, J. Snoek, D. Tran, R. Jenatton. “Hyperparameter Ensembles for Robustness and Uncertainty Quantification” NeurIPS Proceedings.

[5] K. Chaturvedi. (2020). "Wolf and Dog Breed Image Classification Using Deep Learning Techniques".

[6] Ankit Vashisht, Jovian, "ankitvashisht12 /dog-breed-classifier-final", 2020.
https://jovian.ml/ankitvashisht12/dog-breed-classifier-final/v/7#C27

[7] MA7555, "Cat Breeds Dataset," Kaggle, 2020.
https://www.kaggle.com/datasets/ma7555/cat-breeds-dataset

[8] DR. AVICENNA, "Cats and Dogs Breeds Classification Oxford Dataset," Kaggle, 2019.
https://www.kaggle.com/datasets/zippyz/cats-and-dogs-breeds-classification-oxford-dataset

[9] I. Syarif, E. Zaluska, A. Prugel-Bennett, and G. Wills, “Application of bagging, boosting and stacking to intrusion detection,” Machine Learning and Data Mining in Pattern Recognition, pp. 593–602, 2012.

[10] Yash Khandelwal, "Ensemble Stacking for Machine Learning and Deep Learning," Analytics Vidhya, 2021 https://www.analyticsvidhya.com/blog/2021/08/ensemble-stacking-for-machine-learning-and-deep-learning/

8 Contributions

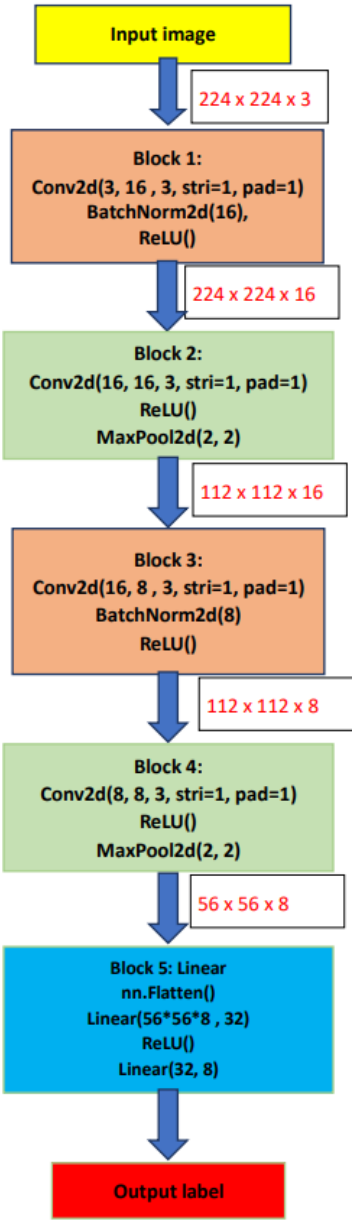
Mahan Mohammadi: Code, Methods, Datasets, proof-reading, Appendix
Rosie Wang: Abstract, Related Works, Conclusion
Youyang Li: Abstract, Introduction, Discussion, Results

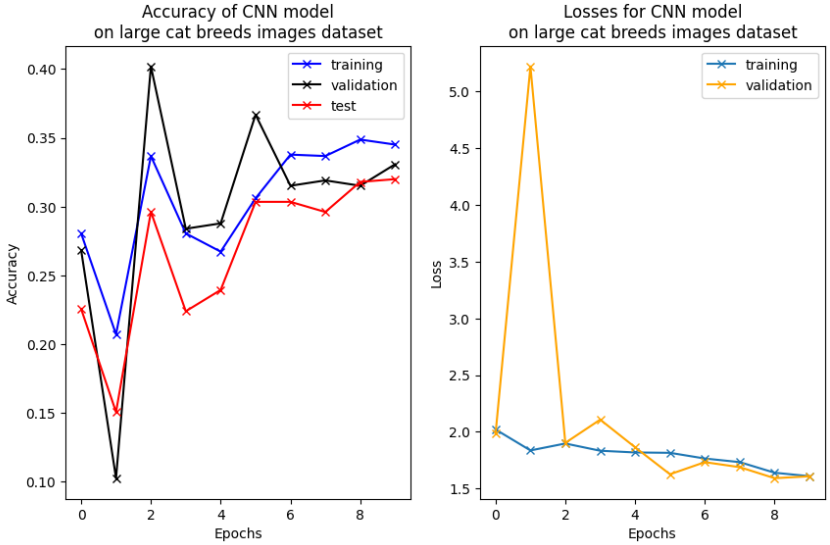
A Appendix

1- CLIP model results:

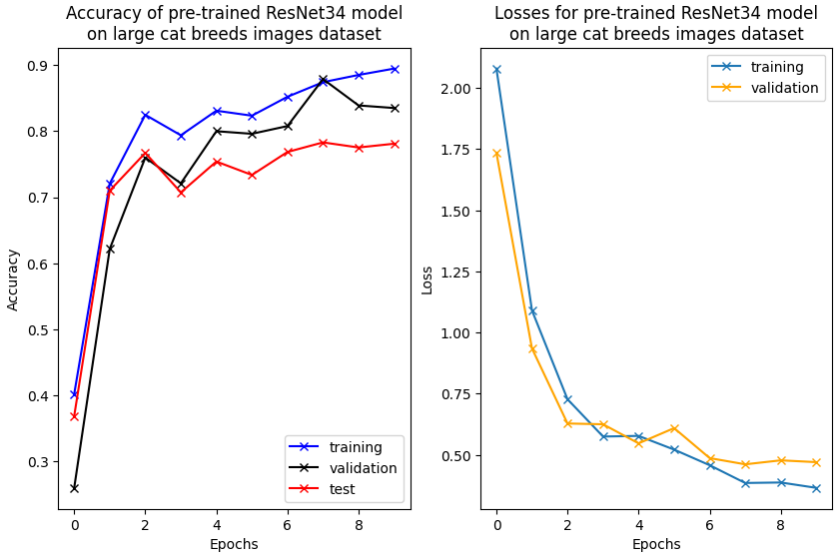
```
Accuracy on class Abyssinian is :0.0737913486005089
Accuracy on class Bengal is :0.09923664122137404
Accuracy on class Birman is :0.10178117048346055
Accuracy on class Bombay is :0.25190839694656486
Accuracy on class EgyptianMau is :0.10687022900763359
Accuracy on class Persian is :0.11450381679389313
Accuracy on class Ragdoll is :0.13740458015267176
Accuracy on class RussianBlue is :0.11450381679389313
Total Accuracy: 0.125
```

2- Custom CNN model: (next page)

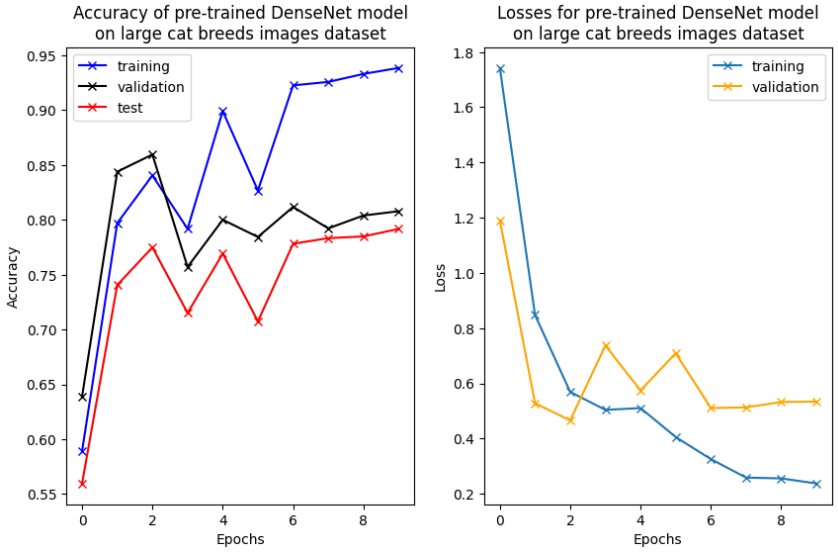




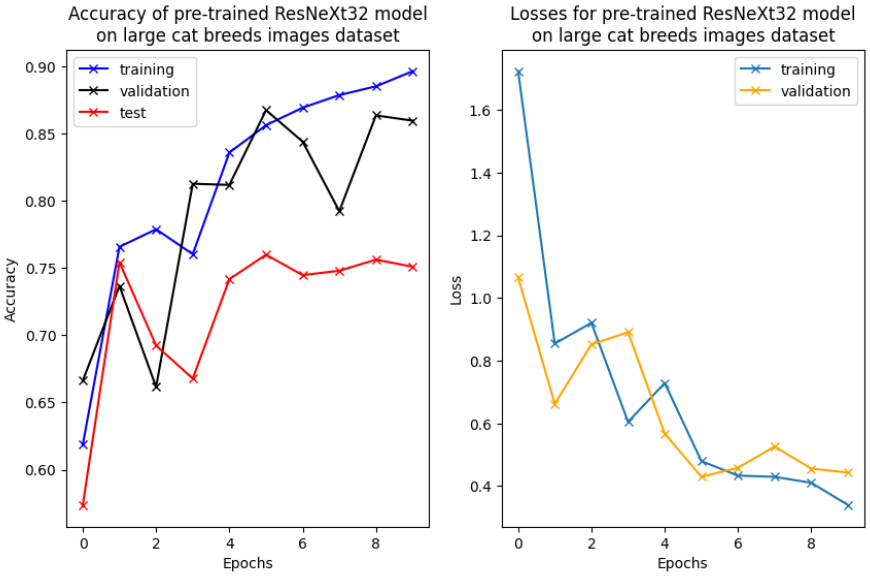
3- Pre-trained ResNet34 model:



4- Pre-trained DenseNet model:

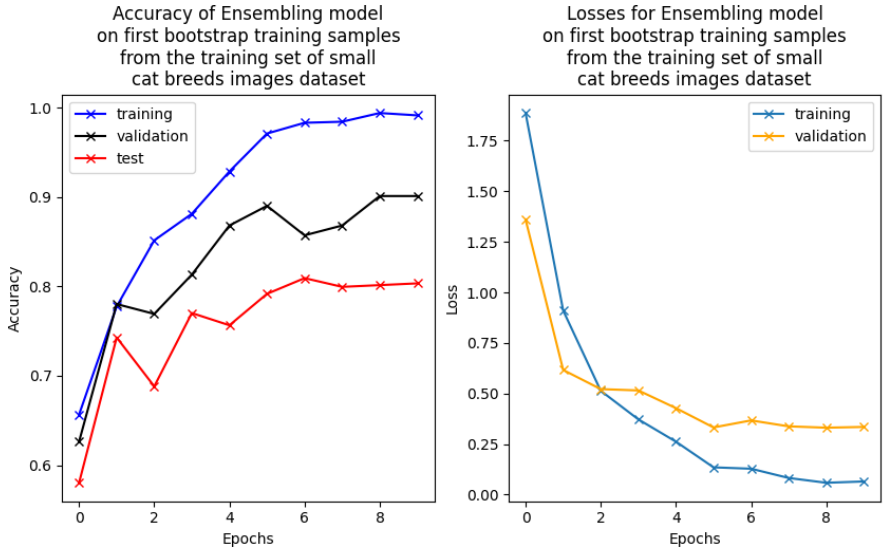


5- Pre-trained ResNeXt32 model:



6-Ensembling method and results:

```python
class ModelEnsemble(ImageClassificationBase):
    def __init__(self):
        super().__init__()

        self.model1 = ResNeXt32Pre_Trained()
        self.model2 = PretrainedDenseNet()
        self.model3 = ResNet34Pre_Trained()

        self.lin = nn.Linear(3*8, 8)  # input size is 24 because we have 3 models with output size of 8

    def forward(self, x):
        out1 = self.model1(x)
        out2 = self.model2(x)
        out3 = self.model3(x)

        # concatenate the outputs of the three models
        out = torch.cat((out1, out2, out3), dim=1)

        # pass the concatenated output through the logistic regression layer
        out = self.lin(out)

        return out
```



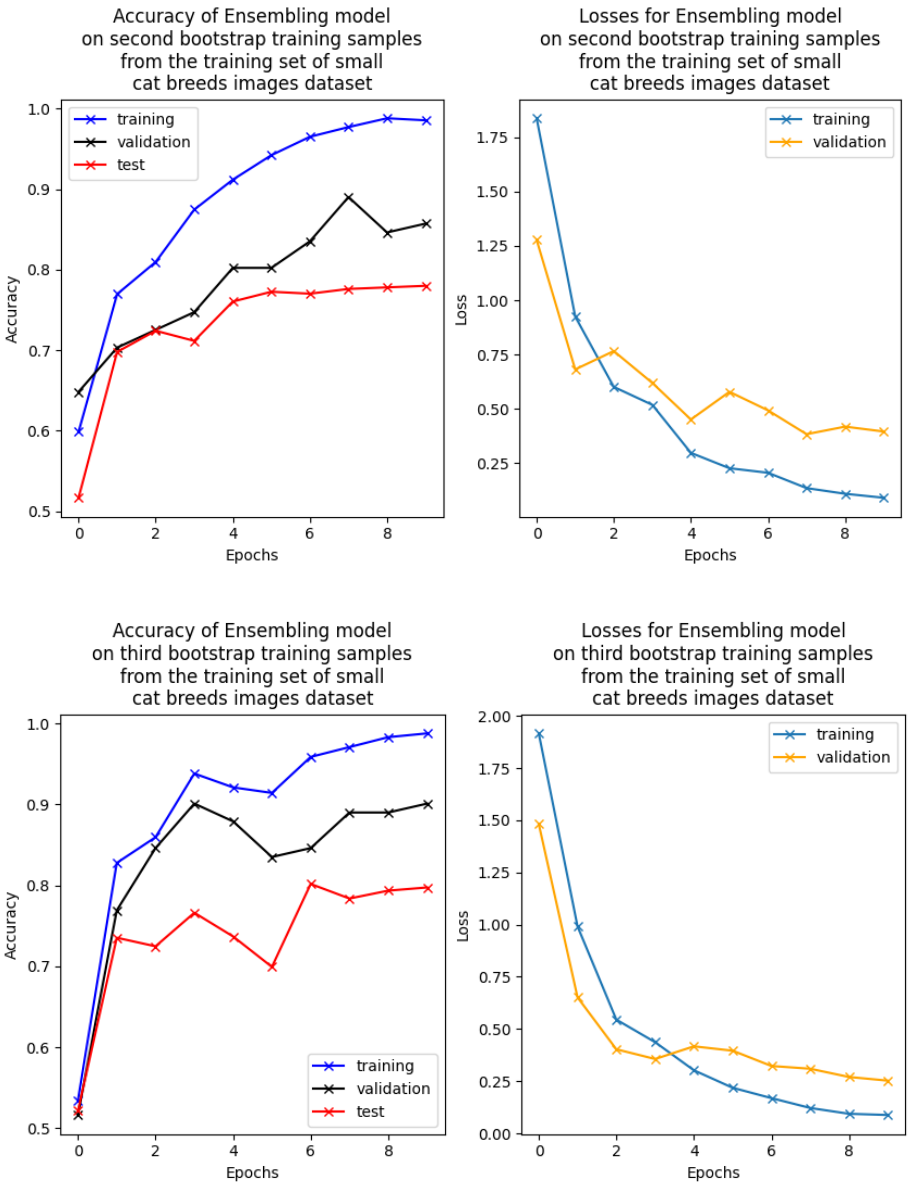


Table 2: Ensemble learning Performance

Bootstrap	Training accuracy	Training loss	Validation accuracy	Validation loss	Test Accuracy
1	0.9913	0.0650	0.9011	0.3346	0.8034
2	0.9853	0.0908	0.8571	0.3957	0.7799
3	0.9879	0.0879	0.9011	0.2524	0.7975
Average	N/A	N/A	0.8864	N/A	0.7936