

---

# DD2424 Course Project

---

Eren Özogul Muhammet

Marcello Krahforst

Youngbin Pyo

## Abstract

↳ good

This project explores various transfer learning strategies for adapting pre-trained convolutional neural networks (CNNs) to new datasets, with a focus on the Oxford-IIT Pet Dataset. Using ResNet-34 as our base model, we systematically compare fine-tuning a fixed number of layers simultaneously and implementing a gradual unfreezing schedule through extensive experiments, achieving over 99% accuracy on binary classification (cat vs. dog) and 91% accuracy on the more challenging 37-class breed classification task. We demonstrate that selectively unfreezing six residual blocks yields optimal results. Additionally, we examine the impact of various regularization techniques, including data augmentation, batch normalization freezing, learning rate scheduling, and cutmix regularization finding that a step-decay scheduler with frozen batch-norm layers produces the most robust performance. Further experiments with imbalanced datasets reveal that weighted cross-entropy loss offers more consistent improvements compared to oversampling techniques. By training a model pre-trained on the Oxford-IIT Pet Dataset on the Oxford Flowers 102 Dataset we induced Catastrophic Forgetting. Our findings provide practical insights for efficiently adapting pre-trained CNNs to specialized image classification tasks with limited data, highlighting the importance of strategic layer selection and appropriate regularization choices.

## 1 Introduction

It is very common to use base models that were trained on similar data as the target dataset and only training the last few layers in order to specialize the network to a specific dataset. There are many ways to adapt a pre-trained CNN to a new dataset. In this project we explore various strategies and try to maximize performance by implementing various regularization and training techniques with the aim to better inform the reader on the best methods to use in this situation. Furthermore, we explore the advent of catastrophic forgetting which is important to be aware of when using different datasets with a single model.

## 2 Related Work

Data regularization is a very important topic in machine learning and one such technique, called "CutMix", was introduced in the paper by Sangdoo Yun et al (3). The authors claim to gain an improvement in top-1 accuracy of +2.28% when using their method with ResNet50 on the ImageNet Classification and also beating other simpler regularization methods such as the Cutout regularization. We decided to explore the advantage we could get from this method in our project.

### 3 Data

This project uses the Oxford-IIT Pet Dataset which consists of images of various cat and dog breeds with 37 categories in total. The dataset consists of roughly 200 images per class, 7349 in total. We split the data into 3669 test images, 368 validation and 3312 training images. In order to use the dataset with our basemodel (ResNet34) we need to rescale the images to a size of 256 and crop into the center of the image to get a size of 224x224. Furthermore, we map the breed classes to indices from 0 to 36.

space

### 4 Method

#### 4.1 Binary Classification Problem

Our first task is to classify between dog and cat images, thus we have a binary classification problem. Our method is to replace the final layer of the ResNet34 model with a layer of 2 nodes. Before replacing the final layer, we first froze all layers of the network. After replacing the final layer, we unfroze the fully connected last layer. We paired this method with the cross-entropy loss and Adam optimizer (with LR = 1e-3 and L2 regularizer = 1e-4). After training with 24 epochs we get a final accuracy over 99%. Transfer learning method helped us to have an access to a model which was trained with a huge amount of data helped us achieve this high accuracy without needing big training data and long training time. An alternative approach was to unfreeze more layers, but we did not have to do that because our method already gave us high accuracy. Training more layers would have increased the run time which was not needed. Another small change would have been using a bigger model, which was also not needed in our case due to achieving high accuracy.

A = C

unnecessary no

#### 4.2 Multi-Class Classification Problem

To tackle the more challenging 37-class breed classification problem, we again built on transfer learning but went significantly beyond head-only training. To be able to make finetuning process easier, we combined most of the functions under NeuralNetwork class. We used 2 strategies to achieve high accuracy for multi-class classification which are fine-tuning 1 layers and gradual unfreezing.

#### 4.3 Finding best parameters for regularization

I don't know about code!  
↳ keep it general

What is it?

In order to find the best parameters for our random resizing and random rotation regularization, we compare the training results with a few selected parameter values to a baseline model without regularization after training for 15 epochs. We decided to train for 15 epochs to increase the chances of overfitting the data, thus showing the effect of regularization.

**Strategy 1: Fine-tune 1 layers simultaneously** In Strategy 1, we simultaneously unfroze the top L residual blocks (for L=1 to 9), pass the number of unfrozen layers as an argument to NeuralNetwork instance for each L, and trained for 10 epochs, recording the final test accuracy for each configuration. By doing this search, we see what is the optimum number of layers for us to train our model to get maximum accuracy. We saw that changing number of layers did not change the final accuracies too much but we managed to get a slightly better accuracy by unfreezing more layer.

After determining the optimal number of layers to fine-tune, we ran a comprehensive grid search over layer-wise learning rates and schedulers, data augmentation (random flips, rotations, crops, and scaling), L2 weight decay, and batch-norm behavior (either fine-tuning or freezing both parameters and running statistics). Our data-augmentation pipeline (random resized crops, horizontal flips, and  $\pm 15^\circ$  rotations) was inspired by the regularization techniques covered in the course, increasing robustness to pose and scale without collecting more images. These experiments produced only marginal accuracy gains over our baseline. We also evaluated ResNet-50 in place of ResNet-34 -but despite the heavier compute cost, it failed to deliver meaningful improvements. Ultimately, we stuck with ResNet-34 and achieved our best result—a modest uplift—by freezing batch-norm layers and applying a step-decay scheduler.

good point

**Strategy 2: Gradual freezing** In Strategy 2, we replaced our fixed-L approach with a “gradual unfreezing” schedule controlled by an unfreezing rate  $uf\_rate$ . For each  $uf\_rate \in$

what is it? > explain

not!

\* when there is no further improvement

*just describe the method!*

*what is it?*

{0.3, 0.5, 0.8, 1, 2, 3}, we progressively unfroze more of ResNet-34's top residual blocks as training progressed, trained for 10 epochs, and recorded the final test accuracy. We got similar results as in strategy 1 and we continued with the best unfreezing rate.

*> you don't know at this point!*

In Strategy 2, we fixed our gradual-unfreezing rate to optimum rate we got and then ran a full grid search over data augmentation, batch-norm behavior, L2 weight decay, and three scheduler options—no scheduler, a StepLR ( $\gamma = 0.1$  every 5 epochs), and the OneCycleLR policy. We added one more scheduler than strategy 1 to see if it changes the accuracy. We kept the head at  $lr = 1 \times 10^{-3}$  and the backbone at  $lr = 1 \times 10^{-4}$  (with optional weight decay =  $10^{-2}$ ), and when batch-norm was frozen we disabled both its gradient updates and running-statistic recalculations. Across these 24 configurations, we found that OneCycleLR generally converged fastest but often over-adapted—dropping test accuracy into the high-70s on several settings—whereas StepLR delivered more consistent, modest gains. The best result came without augmentation, with batch-norm frozen, with weight decay, and using the StepLR schedule. This confirms that while OneCycleLR can accelerate training, the simpler StepLR plus selective regularization strikes the optimal bias-variance balance for our multi-class task.

*schedulers? What is it?*

*what is 'head', 'backbone'?*

#### 4.4 Fine-tuning with imbalanced classes

Once the results from both strategies are out, we can test the dataset using an imbalanced class and fine-tuning. For each cat breed, only 20% of the images were used for training. Each breed had 100 images, with some exceptions like Bombay, English cocker spaniel, and Newfoundland. The total number of samples used for 20% training data added up to 734. We believed that this particular approach had the best representation of observing the impact of an imbalanced class.

To compensate for the imbalanced training set, two additional strategies were implemented: weighted cross-entropy and over-sampling of the minority class. This analysis compares all three strategies together to have a clear view of the functionality of all three approaches. *also 'baseline'?*

In the first approach, we modified the standard cross-entropy loss function by introducing class weights inversely proportional to the number of training samples per class. This penalizes the model more heavily for misclassifying underrepresented classes, thereby encouraging it to learn more balanced representations. The weight for each class  $i$  was computed as:

$$w_i = \frac{\frac{1}{n_i}}{\sum_{j=1}^C \frac{1}{n_j}} \cdot C$$

where  $n_i$  is the number of samples in class  $i$ , and  $C$  is the total number of classes. These normalized weights were passed to the CrossEntropyLoss function in PyTorch.

In the second approach, we used a weighted random sampling strategy to oversample the minority classes during mini-batch formation. Specifically, we calculated the weights per sample based on the frequencies of the inverse class and used WeightedRandomSampler to draw replacement training samples. This ensured that the model was more frequently seeing underrepresented classes during training, without modifying the original dataset distribution.

#### 4.5 Catastrophic Forgetting

*shouldn't you train all 102?*

Furthermore, we try to induce catastrophic forgetting by training our best multiclass dog vs. cat model on the filtered data from the Flower102 dataset containing only the first 37 classes such that the prediction dimension between the two datasets matches. First, we train our model for 15 epochs on the Pet dataset. This model is then trained on the Flower102 dataset until we achieve a reasonable accuracy. Finally, this model is tested on the Pet dataset again.

We try two strategies for retraining the network with the six last layers unfrozen simultaneously on the new dataset. The first strategy uses a sort of step learning rate where we start training the network with a high learning rate of 0.1 for the final fully connected layer and 0.01 for the other layers for four epochs. Afterwards, we reduce the learning rates by a factor of 0.01. This is a fairly aggressive strategy. The second strategy on the other hand trains with a consistent learning rate of 0.01 and 0.001 for the fully connected and other layers respectively for 20 epochs.

*good part*

*x this is not how it works. The last layer is retrained anyway, right? How did you select those 37 classes?*

## 4.6 CutMix Regularization

The final experiment consists of implementing the CutMix Regularization method, which randomly selects an area in the images in a batch and replaces them with parts from other images in the same batch. The size of the replaced patch is controlled by the  $\lambda$  parameter which represents the ratio between cutout and original image ( $\lambda = 0.5$  leads to a similar size between cutout and original image whereas larger or smaller values lead to a larger cutout or larger original image). The  $\lambda$  parameter can not be controlled directly but is instead chosen randomly by the beta distribution which is set as  $\beta(\alpha, \alpha) \lambda$ . The  $\alpha$  parameter can be set as a parameter. Furthermore, the loss calculation has to be updated to include the two ratios of the cutout and original images. Thus the loss becomes:

$$\mathcal{L} = \lambda \cdot \mathcal{L}(\text{label}_A) + (1 - \lambda) \cdot \mathcal{L}(\text{label}_B)$$

The aim of this method is to prevent overfitting the data and improving the generalization capabilities of the model as introduced in the paper about CutMix mentioned in the related works.

*cite again*

## 5 Experiments

### 5.1 Binary Classification Problem

To validate our ResNet-34 transfer-learning pipeline on the binary dog vs. cat task, we conducted the following experiments:

*< so long to check for convergence?*

- **Training and Validation Curves.** We trained for 24 epochs and logged per-epoch train/val accuracies (Table I). Validation accuracy exceeded 99 % by epoch 3 and remained above 99 % thereafter, demonstrating rapid convergence.
- **Final accuracy on test data** As in Table I after training we got accuracy of 99.18% . This result show us that our model successfully classifies our dog vs cat task with a high accuracy.

Table 1: Training and Validation Accuracy per Epoch (ResNet-34 Binary Classification)

| Epoch             | Train Accuracy (%) | Validation Accuracy (%) |
|-------------------|--------------------|-------------------------|
| 1                 | 98.85              | 99.18                   |
| 2                 | 98.97              | 99.46                   |
| 3                 | 99.18              | 99.73                   |
| 4                 | 98.76              | 99.46                   |
| 5                 | 99.40              | 99.18                   |
| 6                 | 98.91              | 99.73                   |
| 7                 | 99.58              | 99.73                   |
| 8                 | 99.43              | 99.46                   |
| 9                 | 98.82              | 99.18                   |
| 10                | 98.91              | 99.73                   |
| 11                | 99.18              | 99.46                   |
| 12                | 99.52              | 99.46                   |
| 13                | 99.12              | 99.46                   |
| 14                | 99.00              | 99.46                   |
| 15                | 99.43              | 99.18                   |
| 16                | 98.85              | 99.46                   |
| 17                | 99.40              | 99.18                   |
| 18                | 99.06              | 99.46                   |
| 19                | 98.94              | 99.46                   |
| 20                | 99.52              | 99.46                   |
| 21                | 99.52              | 99.18                   |
| 22                | 99.79              | 99.46                   |
| 23                | 99.25              | 99.46                   |
| 24                | 99.64              | 99.73                   |
| Test Accuracy (%) |                    | <b>99.18</b>            |

*plot instead  
of table  
→ easier to  
interpret*

*table just  
wastes space*

## 5.2 Finding best parameters for regularization

The baseline model has a final train-test accuracy gap of 8.15% whilst a random scaling factor in the interval  $[0.8, 1]$  resulted in the lowest gap with 6.11%. The best random rotation parameter was 15 degrees which resulted in a train-test gap of 7.3% whilst retaining the best accuracy. However, the accuracy with regularization is always lower than the baseline accuracy. We decided to use 0.8 as a parameter for the random scaling and 15 degrees for random rotation. The results can be seen in Table 6

## 5.3 Multi-Class Classification Problem

**Strategy 1: Fine-tune 1 layers simultaneously** To assess the impact of varying how many layers we fine-tune simultaneously (Strategy 1), we conducted the following experiments:

- **Test Accuracy vs # Trainable Layers.** We trained for 10 epochs for each  $L \in \{1, \dots, 9\}$ , freezing all but the last  $L$  residual blocks and the new head. Final test accuracies are reported in Table 2. Performance peaked at  $L = 6$  (90.24%), illustrating that unfreezing six blocks strikes the best balance between specialization and overfitting.
- **Extended Hyperparameter Grid.** Focusing on the  $L = 6$  configuration, we ran a grid search over data augmentation, batch-norm behavior, L2 weight decay, and a StepLR scheduler (Table 3). The highest test accuracy (91.11%) was achieved with augmentation off, batch-norm frozen, no L2 decay, and StepLR enabled, confirming that selective regularization and scheduling can yield modest gains.
- **Key Takeaways.**
  - Unfreezing more than six blocks ( $L > 6$ ) offered no further benefit and increased training cost.
  - Data augmentation alone had limited impact on test accuracy without batch-norm and scheduler adjustments. → possible explanation?
  - Freezing batch-norm layers consistently improved stability when fine-tuning fewer blocks.

Table 2: Final Test Accuracies (%) for Strategy 1 with Varying Numbers of Trainable Layers

| # Trainable Layers | Test Accuracy (%) |
|--------------------|-------------------|
| 1                  | 89.45             |
| 2                  | 89.32             |
| 3                  | 89.70             |
| 4                  | 89.42             |
| 5                  | 89.70             |
| 6                  | 90.24             |
| 7                  | 89.26             |
| 8                  | 89.37             |
| 9                  | 89.72             |

(optional):  
plot bar chart  
instead of table

**Strategy 2: Gradual unfreezing** To evaluate the “gradual unfreezing” approach (Strategy 2), we performed two sets of experiments:

- **Test Accuracy vs. Unfreezing Rate.** We trained for 10 epochs with  $uf\_rate \in \{0.3, 0.5, 0.8, 1, 2, 3\}$  and recorded final test accuracies (Table 4). Performance peaked at  $uf\_rate = 0.3$  and again at  $uf\_rate = 2.0$  (both 90.27 %), illustrating that a moderate unfreezing pace best balances stability and specialization. What is unfreezing pace?
- **Extended Hyperparameter Grid.** Fixing  $uf\_rate = 2.0$ , we ran a 24-configuration grid search over data augmentation, batch-norm behavior, L2 weight decay, and three schedulers (none, StepLR, OneCycleLR) (Table 5). OneCycleLR generally yielded lower accuracy, but we included it in Strategy 2 tests even though it was not evaluated in Strategy 1, just to see if we get better results. The best configuration (90.24%) used no augmentation, froze batch-norm, applied L2 decay, and employed StepLR.

Hypersparameters only True/False to the methods  
or also e.g.  $\eta$ , optimizer, etc?

Table 3: Strategy 1 Extended Grid Results

| Aug   | Freeze BN | L2    | Scheduler | Test Accuracy (%) |
|-------|-----------|-------|-----------|-------------------|
| False | False     | False | False     | 89.62             |
| False | False     | False | True      | 90.87             |
| False | False     | True  | False     | 90.38             |
| False | False     | True  | True      | 90.57             |
| False | True      | False | False     | 90.30             |
| False | True      | False | True      | 91.11             |
| False | True      | True  | False     | 90.24             |
| False | True      | True  | True      | 90.54             |
| True  | False     | False | False     | 88.14             |
| True  | False     | False | True      | 89.70             |
| True  | False     | True  | False     | 87.33             |
| True  | False     | True  | True      | 89.70             |
| True  | True      | False | False     | 88.28             |
| True  | True      | False | True      | 89.56             |
| True  | True      | True  | False     | 86.43             |
| True  | True      | True  | True      | 89.45             |

• Key Takeaways.  $\rightarrow$  top!

- A moderate unfreezing rate ( $uf\_rate = 0.3$  or  $2.0$ ) outperforms both very slow and very aggressive schedules.
- OneCycleLR had lower test accuracy, but was tested here to try ??
- StepLR coupled with batch-norm freezing and L2 regularization yields the most robust performance for multi-class breed recognition.

• Comparison with Strategy 1. just reprob?

- Strategy 1's extended grid (with  $L = 6$ ) achieved a peak test accuracy of 91.11% (Table 3), surpassing Strategy 2's best of 90.24%.
- Given the slight performance edge, Strategy 1 is preferred for maximum accuracy, whereas Strategy 2 remains a viable alternative when a gradual unfreezing schedule better fits resource or implementation constraints.

#### 5.4 Why is the performance not improving further? nice research!

After plotting the confusion matrix for our best model (strategy 1 with 6 unfrozen layers) in figure 1 it becomes clear that there are very few classes that are holding our model's performance back but two classes stand out: the Staffordshire Bull Terrier (class 34) and the American Pit Bull Terrier (class 13). After looking at the two dog breeds in figures 3 and 4 one can see that these two breeds indeed look very similar, which explains the classification errors. We tried training only on those two classes for one epoch but unfortunately this did not work and we were unable to fix the issue.

#### 5.5 Fine-tuning with imbalanced classes

what is the difference between those numbers? After experimenting with the imbalanced dataset with standard entropy loss on 10 epochs and a batch size of 32, our final test accuracy resulted in 8.6%. Table 7 shows the change in training and validation accuracy over each epoch. The correlation between the sample count and the accuracy was quite low at 0.0978. Average accuracy across all classes was 8.80% very poor results.

The results from weighted cross-entropy loss and over-sampling had different results. For weighted cross-entropy, as shown in Table 8, most breeds had slight improvements in the accuracies, while oversampling returned varying results. Table 9 shows that the accuracies for some breeds such as Staffordshire bull terrier, wheaten terrier, and Yorkshire terrier drop quite significantly while others either didn't have much improvements or small boosts in its accuracy. good part, but do you have an explanation for it? would be nice!

\* what does that mean? what is the correlation? put the number (0.0978) into perspective,

otherwise it's hard to say if it is good or bad.

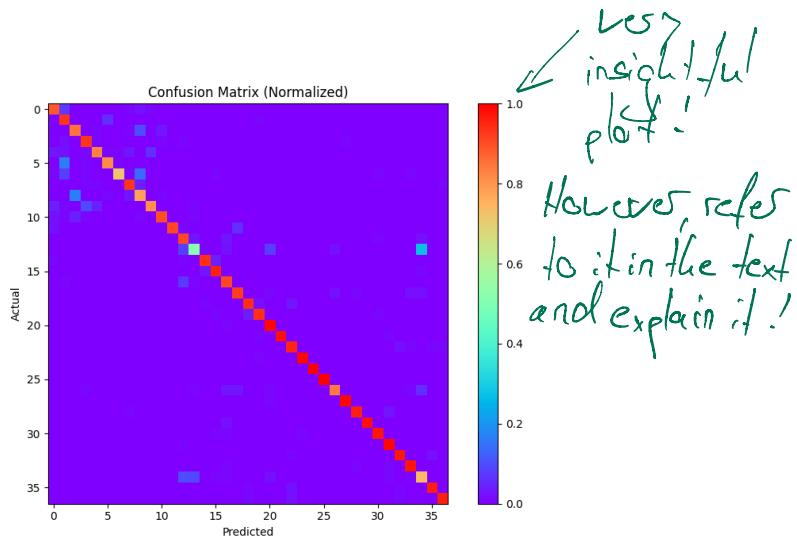
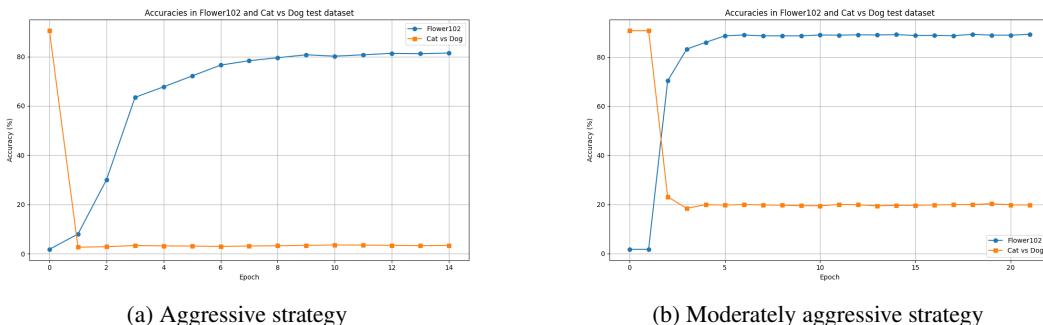


Figure 1: Confidence Matrix on Cat vs Dog Dataset for our best model

## 5.6 Catastrophic Forgetting

We perform a test on both the Flower102 and Pet Datasets after each epoch in order to track the development of the accuracies. It is interesting to see that with both strategies the accuracy on the Cat vs Dog Dataset plummets already almost to its final convergence value. However, the accuracy on the Pet Dataset after training with the aggressive strategy is lower with 31.14%; The less aggressive strategy has a final accuracy of 19.73%. The plots of the accuracies over the epochs for both strategies can be seen in figures 2 (you can find larger plots in the appendix). When looking at the confusion matrices for the less aggressive strategy in figures 10 and 11, it is clear that the predictions for class 19 are quite accurate on both datasets, which might mean that they share many similarities. However, they do not seem to be very similar as can be seen in figures 6 and 5. This error is likely caused by the low number of samples from class 19 in the Flower102 dataset (only 10 images in the train set). However, the confusion matrices shown in figures 8, 10, 9 and 11 clearly show that the model forgot about most of the dogs and cats in the dataset.



## 5.7 CutMix Regularization

After ensuring that the CutMix Regularization is working as intended by taking a look at a sample output that can be seen in figure 7, we run a linear search in order to find the most promising alpha parameter (parameter for the beta distribution) with the largest accuracy gains for our model. The results can be found in table 10. The best alpha is 0.5, but unfortunately it did not bring any final test accuracy gains. The accuracy for  $\alpha = 0.5$  during the search was 89.04% but after training with the same parameters for 20 epochs the final test accuracy was only 88.79%, thus not leading to any increase in accuracy. However, it managed to keep the train accuracy always under 80%, while the

why is it less then?  
explain what  $\alpha$  and  $\beta$  is!

model with the same parameters without regularization already reached a training accuracy of 100% after very few epochs. → so it was fitted?

## 6 Conclusion

Referring back to ResNet-34, the model itself performs very well even with a very small number of epochs, easily reaching 99% validation accuracy. To study how the model performs for a multi-class classification problem, experiments were performed for fine-tuning  $l$  layers, simultaneously and gradually unfreezing layers.

What were you talking about before? best (?) CNNs perform worse in other classes  
The first approach was to find the test accuracies by increasing the number of trainable layers. However, the accuracy began to drop after 6 layers, indicating that 6 layers is the maximum number of layers required for tuning. The second approach, on the other hand, had small improvements compared to the first approach where we used different unfreezing rates for the test accuracy. The accuracy and the increasing rate seemed to have a periodic relationship, which may be further researched in the future. \* sb) no clue what this is

With regards to the imbalanced training, weighted cross-entropy achieved modest improvements for several breeds and yielded a more balanced overall performance, with an average per-class accuracy increase. Notably, breeds like Abyssinian showed a 14.29% improvement, while others experienced small or negligible gains. In contrast, oversampling led to large gains in a few classes, such as Abyssinian (+28.57%), but at the cost of significant drops for others (e.g., Yorkshire terrier: -90%), highlighting the risk of overfitting and instability. → very nice part!

Furthermore, we were able to induce catastrophic forgetting by training a pre-trained model on another dataset and show that the learning rate has a crucial impact on how much the model forgets. Besides that, our implementation of CutMix regularization did not help improve our final test accuracy but was very effective in preventing the model from being able to overfit the training data.

Overall, this project provides actionable insights into how to fine-tune CNNs for new tasks with limited or imbalanced data. Key recommendations include selectively unfreezing layers, regularizing with frozen batch norm and schedulers, and carefully choosing imbalance handling techniques based on trade-offs between fairness and stability. Additionally one should always be careful when training a model on different datasets and make sure that the features are very similar in both in order to prevent potential forgetting. Future work could explore methods to overcome the miss-classification of the dog breeds, ensemble methods, contrastive learning, or rehearsal-based strategies to mitigate forgetting in sequential learning settings.

## References

[1] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar, Cats and Dogs, in *Proceedings of the British Machine Vision Conference (BMVC)*, 2012. ↗ explain!

[2] Oxford-IIIT Pet Dataset, Omkar M. Parkhi *et al.*, <https://www.robots.ox.ac.uk/~vgg/data/pets/> 2012.

[3] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo.

CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. ↗ did you come up with it?

arXiv preprint arXiv:1905.04899, 2019.

<https://arxiv.org/abs/1905.04899>

(1) Papers: Catastrophic forgetting? ↗ what is the origin?  
A Appendix / project code ↗ How do similar features (Pet <=> Flowers) look like?

The code for this project can be found on [this](#) GitHub repo.

## B Appendix / supplemental material

I have the feeling that plate/tables are more or less randomly (order) put into the appendix and not according to relevance/importance.



Figure 3: American Pit Bull Terrier (class 13)

What is  
the bone b4  
here?



Figure 4: Staffordshire Bull Terrier (class 34)



Figure 5: Giant White Arum Lily (class 19)



Figure 6: English Setter (class 19)



Figure 7: Example images after cutmix regularization

super interesting!

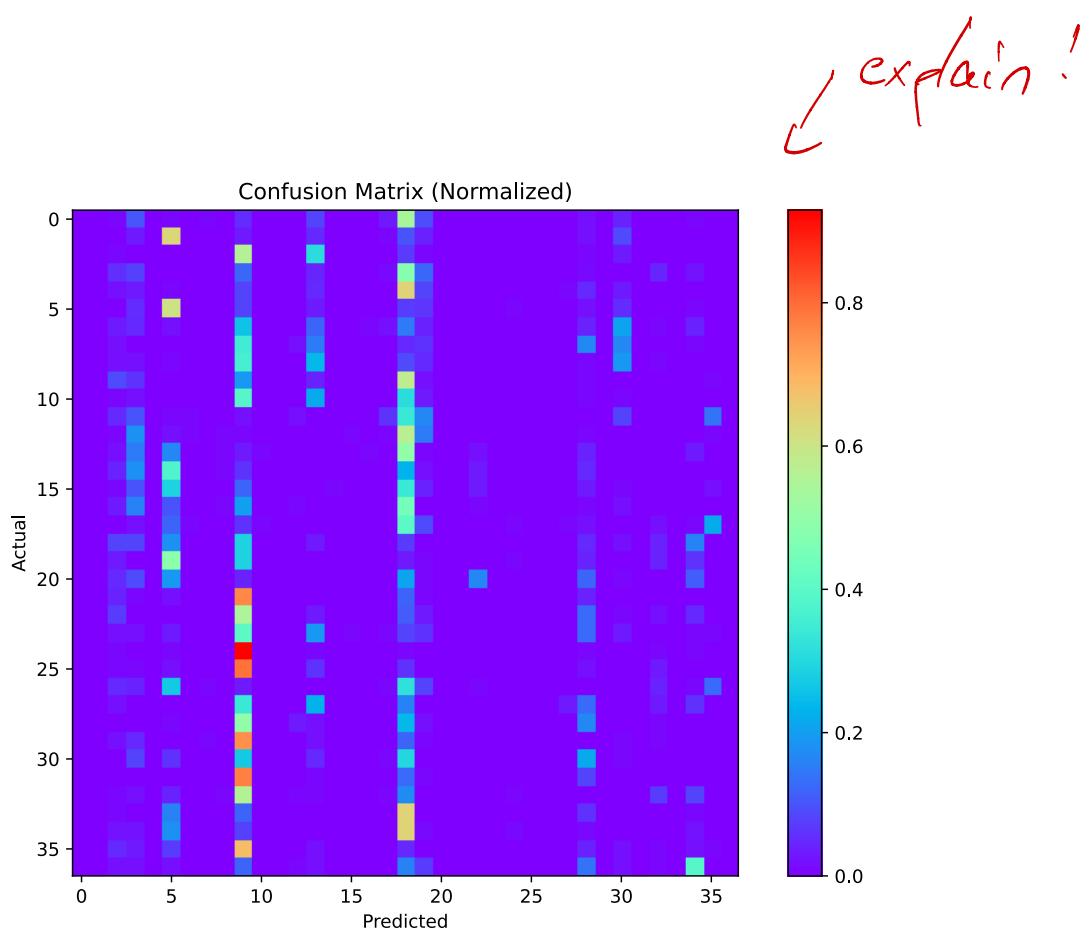


Figure 8: Confidence Matrix on Cat vs Dog Dataset after aggressive training

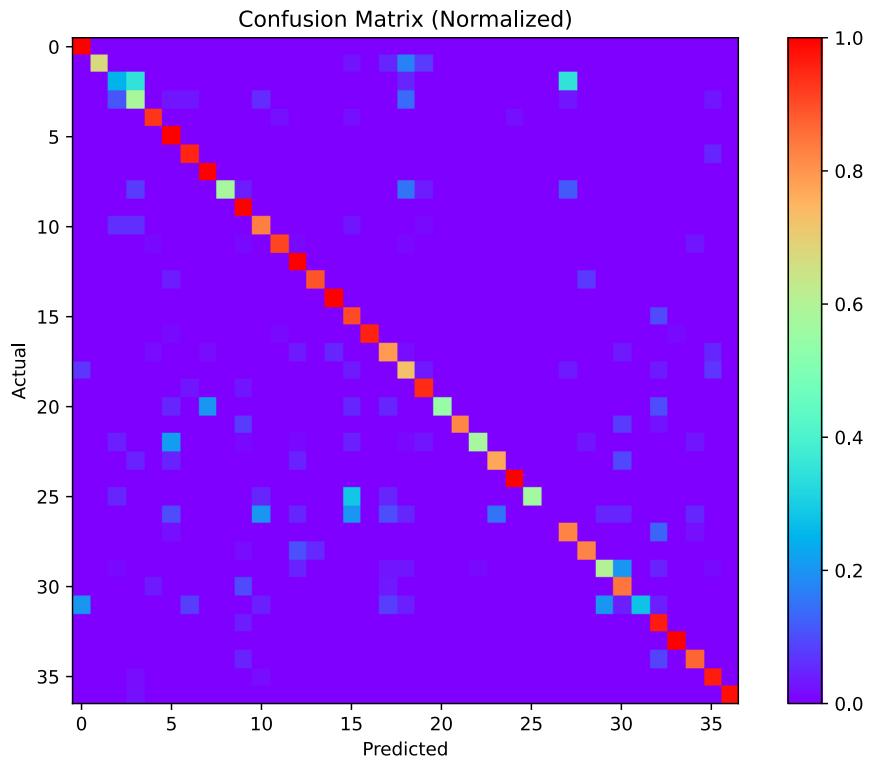


Figure 9: Confidence Matrix on Flower102 Dataset after aggressive training

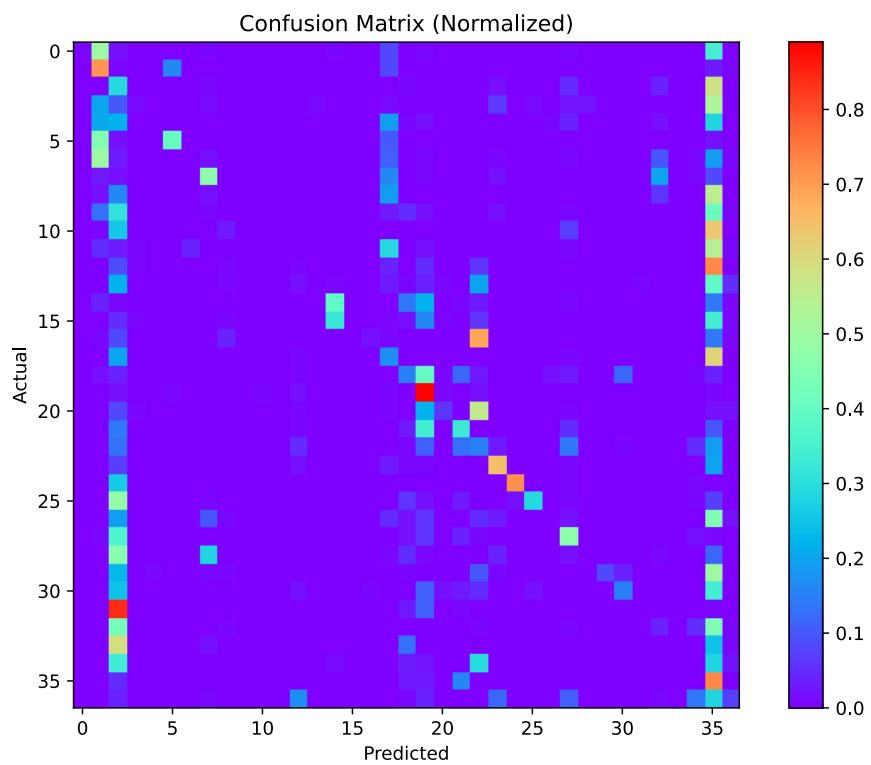


Figure 10: Confidence Matrix on Cat vs Dog Dataset after moderately aggressive training

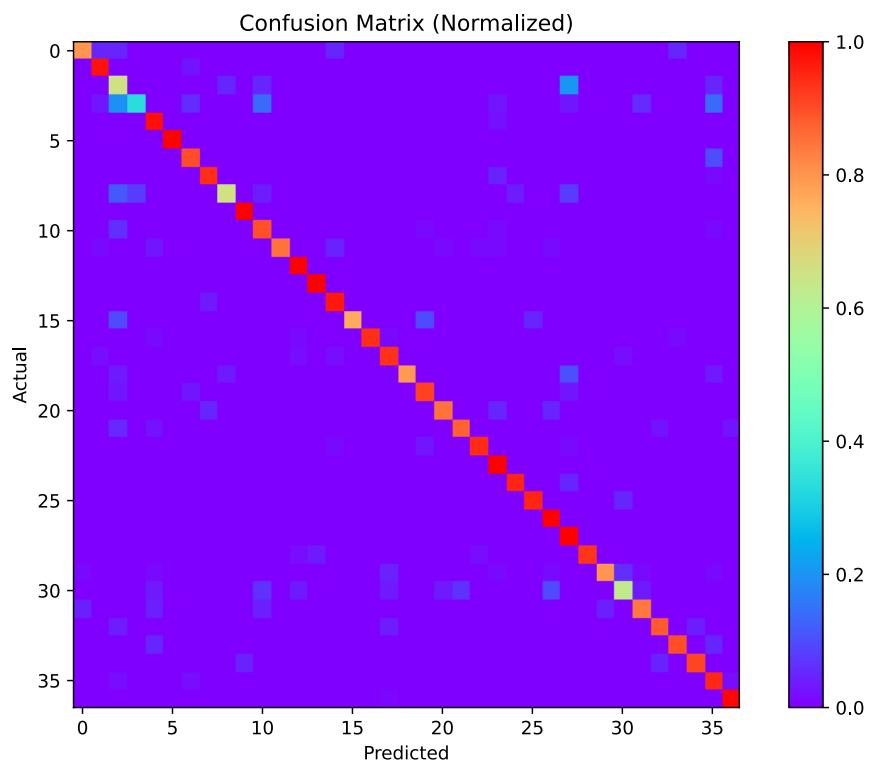


Figure 11: Confidence Matrix on Flower102 Dataset after moderately aggressive training

Table 4: Final Test Accuracies (%) for Strategy 2 with Varying Unfreezing Rates

| Unfreezing Rate | Test Accuracy (%) |
|-----------------|-------------------|
| 0.3             | 90.27             |
| 0.5             | 89.07             |
| 0.8             | 89.37             |
| 1.0             | 89.67             |
| 2.0             | 90.27             |
| 3.0             | 89.94             |

Table 5: Strategy 2 Grid Search Results

| Aug   | Freeze BN | L2    | Scheduler | Test Accuracy (%) |
|-------|-----------|-------|-----------|-------------------|
| False | False     | False | none      | 89.34%            |
| False | False     | False | steplr    | 89.70%            |
| False | False     | False | onecycle  | 78.06%            |
| False | False     | True  | none      | 89.72%            |
| False | False     | True  | steplr    | 90.05%            |
| False | False     | True  | onecycle  | 78.63%            |
| False | True      | False | none      | 89.34%            |
| False | True      | False | steplr    | 89.64%            |
| False | True      | False | onecycle  | 77.24%            |
| False | True      | True  | none      | 89.13%            |
| False | True      | True  | steplr    | 90.24%            |
| False | True      | True  | onecycle  | 78.90%            |
| True  | False     | False | none      | 89.04%            |
| True  | False     | False | steplr    | 89.07%            |
| True  | False     | False | onecycle  | 74.63%            |
| True  | False     | True  | none      | 88.58%            |
| True  | False     | True  | steplr    | 88.93%            |
| True  | False     | True  | onecycle  | 73.18%            |
| True  | True      | False | none      | 88.39%            |
| True  | True      | False | steplr    | 89.37%            |
| True  | True      | False | onecycle  | 75.28%            |
| True  | True      | True  | none      | 87.90%            |
| True  | True      | True  | steplr    | 89.13%            |
| True  | True      | True  | onecycle  | 76.70%            |

mark best!

| Parameter       | Value | Train Acc (%) | Test Acc (%) | Gap (%) |
|-----------------|-------|---------------|--------------|---------|
| Baseline        | —     | 98.07         | 89.92        | 8.15    |
| Random Scaling  | 0.8   | 95.56         | 89.45        | 6.11    |
| Random Scaling  | 0.6   | 96.50         | 89.18        | 7.32    |
| Random Scaling  | 0.5   | 97.43         | 88.83        | 8.60    |
| Random Rotation | 10°   | 96.04         | 87.30        | 8.74    |
| Random Rotation | 15°   | 95.98         | 88.68        | 7.30    |
| Random Rotation | 25°   | 94.17         | 86.74        | 7.43    |

Table 6: Regularization parameter sweep: train accuracy, test accuracy, and the gap.

Table 7: Training and Validation Accuracy per Epoch (Baseline Model)

| Epoch             | Train Accuracy (%) | Validation Accuracy (%) |
|-------------------|--------------------|-------------------------|
| 1                 | 8.18               | 31.29                   |
| 2                 | 48.21              | 62.59                   |
| 3                 | 74.11              | 75.51                   |
| 4                 | 83.99              | 76.87                   |
| 5                 | 89.95              | 80.95                   |
| 6                 | 92.84              | 84.35                   |
| 7                 | 94.89              | 82.99                   |
| 8                 | 94.55              | 82.99                   |
| 9                 | 96.93              | 85.71                   |
| 10                | 97.79              | 86.39                   |
| Test Accuracy (%) |                    | 8.67                    |

↳ What? Why?  
 ↳ Explain!

Table 8: Per-class Accuracy Changes (Weighted CE vs Baseline)

| Breed                       | Baseline (%) | Weighted CE (%) | Change (%) |
|-----------------------------|--------------|-----------------|------------|
| Abyssinian                  | 55.10        | 69.39           | +14.29     |
| Bengal                      | 0.00         | 0.00            | +0.00      |
| Birman                      | 0.00         | 0.00            | +0.00      |
| Bombay                      | 0.00         | 0.00            | +0.00      |
| British_Shorthair           | 0.00         | 0.00            | +0.00      |
| Egyptian_Mau                | 32.00        | 10.00           | -22.00     |
| Maine_Coon                  | 1.00         | 1.00            | +0.00      |
| Persian                     | 0.00         | 0.00            | +0.00      |
| Ragdoll                     | 0.00         | 0.00            | +0.00      |
| Russian_Blue                | 17.00        | 15.00           | -2.00      |
| Siamese                     | 1.00         | 0.00            | -1.00      |
| Sphynx                      | 0.00         | 0.00            | +0.00      |
| american_bulldog            | 0.00         | 0.00            | +0.00      |
| american_pit_bull_terrrier  | 0.00         | 0.00            | +0.00      |
| basset_hound                | 0.00         | 0.00            | +0.00      |
| beagle                      | 0.00         | 0.00            | +0.00      |
| boxer                       | 0.00         | 0.00            | +0.00      |
| chihuahua                   | 0.00         | 0.00            | +0.00      |
| english_cocker_spaniel      | 0.00         | 0.00            | +0.00      |
| english_setter              | 0.00         | 0.00            | +0.00      |
| german_shorthaired          | 0.00         | 0.00            | +0.00      |
| great_pyrenees              | 0.00         | 0.00            | +0.00      |
| havanese                    | 2.00         | 1.00            | -1.00      |
| japanese_chin               | 0.00         | 0.00            | +0.00      |
| keeshond                    | 0.00         | 0.00            | +0.00      |
| leonberger                  | 0.00         | 0.00            | +0.00      |
| miniature_pinscher          | 0.00         | 0.00            | +0.00      |
| newfoundland                | 0.00         | 0.00            | +0.00      |
| pomeranian                  | 0.00         | 0.00            | +0.00      |
| pug                         | 0.00         | 0.00            | +0.00      |
| saint_bernard               | 0.00         | 0.00            | +0.00      |
| samoyed                     | 1.00         | 2.00            | +1.00      |
| scottish_terrrier           | 0.00         | 0.00            | +0.00      |
| shiba_inu                   | 0.00         | 0.00            | +0.00      |
| staffordshire_bull_terrrier | 62.92        | 62.92           | +0.00      |
| wheaten_terrrier            | 74.00        | 72.00           | -2.00      |
| yorkshire_terrrier          | 90.00        | 91.00           | +1.00      |

↑ Why so many zeros? what does it mean?  
 13 takeaways / bullet / explanation?

Table 9: Per-class Accuracy Changes (Oversampling vs Baseline)

| Breed                       | Baseline (%) | Oversampling (%) | Change (%) |
|-----------------------------|--------------|------------------|------------|
| Abyssinian                  | 55.10        | 83.67            | +28.57     |
| Bengal                      | 0.00         | 0.00             | +0.00      |
| Birman                      | 0.00         | 0.00             | +0.00      |
| Bombay                      | 0.00         | 0.00             | +0.00      |
| British_Shorthair           | 0.00         | 0.00             | +0.00      |
| Egyptian_Mau                | 32.00        | 1.00             | -31.00     |
| Maine_Coon                  | 1.00         | 0.00             | -1.00      |
| Persian                     | 0.00         | 0.00             | +0.00      |
| Ragdoll                     | 0.00         | 0.00             | +0.00      |
| Russian_Blue                | 17.00        | 0.00             | -17.00     |
| Siamese                     | 1.00         | 0.00             | -1.00      |
| Sphynx                      | 0.00         | 0.00             | +0.00      |
| american_bulldog            | 0.00         | 1.00             | +1.00      |
| american_pit_bull_terrrier  | 0.00         | 11.00            | +11.00     |
| basset_hound                | 0.00         | 5.00             | +5.00      |
| beagle                      | 0.00         | 9.00             | +9.00      |
| boxer                       | 0.00         | 1.00             | +1.00      |
| chihuahua                   | 0.00         | 5.00             | +5.00      |
| english_cocker_spaniel      | 0.00         | 0.00             | +0.00      |
| english_setter              | 0.00         | 0.00             | +0.00      |
| german_shorthaired          | 0.00         | 0.00             | +0.00      |
| great_pyrenees              | 0.00         | 0.00             | +0.00      |
| havanese                    | 2.00         | 0.00             | -2.00      |
| japanese_chin               | 0.00         | 0.00             | +0.00      |
| keeshond                    | 0.00         | 0.00             | +0.00      |
| leonberger                  | 0.00         | 0.00             | +0.00      |
| miniature_pinscher          | 0.00         | 0.00             | +0.00      |
| newfoundland                | 0.00         | 0.00             | +0.00      |
| pomeranian                  | 0.00         | 0.00             | +0.00      |
| pug                         | 0.00         | 0.00             | +0.00      |
| saint_bernard               | 0.00         | 0.00             | +0.00      |
| samoyed                     | 1.00         | 0.00             | -1.00      |
| scottish_terrrier           | 0.00         | 0.00             | +0.00      |
| shiba_inu                   | 0.00         | 0.00             | +0.00      |
| staffordshire_bull_terrrier | 62.92        | 0.00             | -62.92     |
| wheaten_terrrier            | 74.00        | 0.00             | -74.00     |
| yorkshire_terrrier          | 90.00        | 0.00             | -90.00     |

same  
as table 8

Table 10: Test Accuracy for Different  $\alpha$  Values (CutMix)

| $\alpha$ | Test Accuracy (%) |
|----------|-------------------|
| 0.1      | 88.53             |
| 0.2      | 88.69             |
| 0.3      | 88.80             |
| 0.4      | 88.61             |
| 0.5      | 89.04             |
| 0.6      | 88.63             |
| 0.7      | 88.96             |
| 0.8      | 88.74             |
| 0.9      | 88.66             |

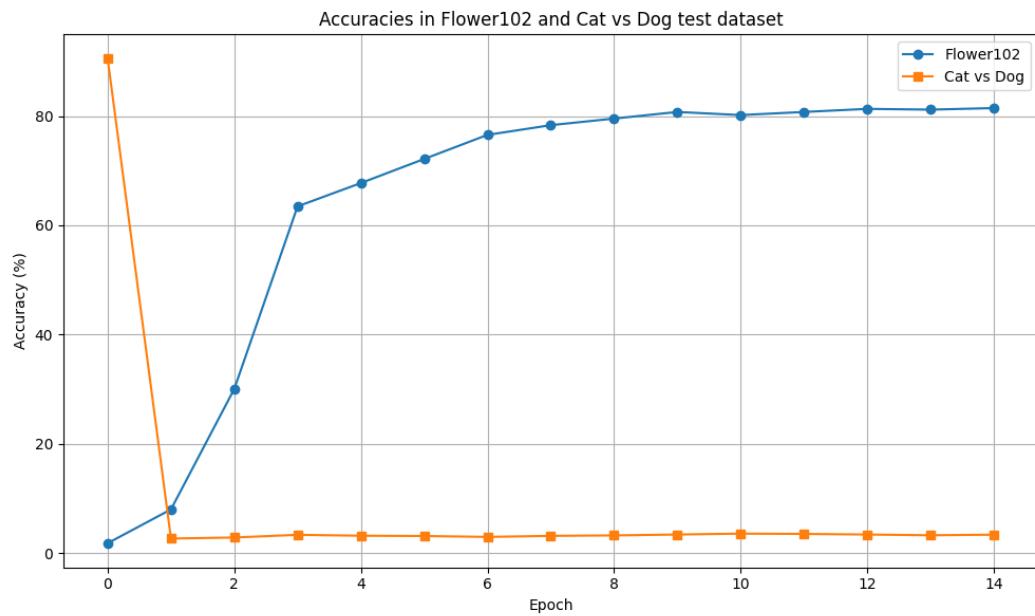


Figure 12: Aggressive strategy

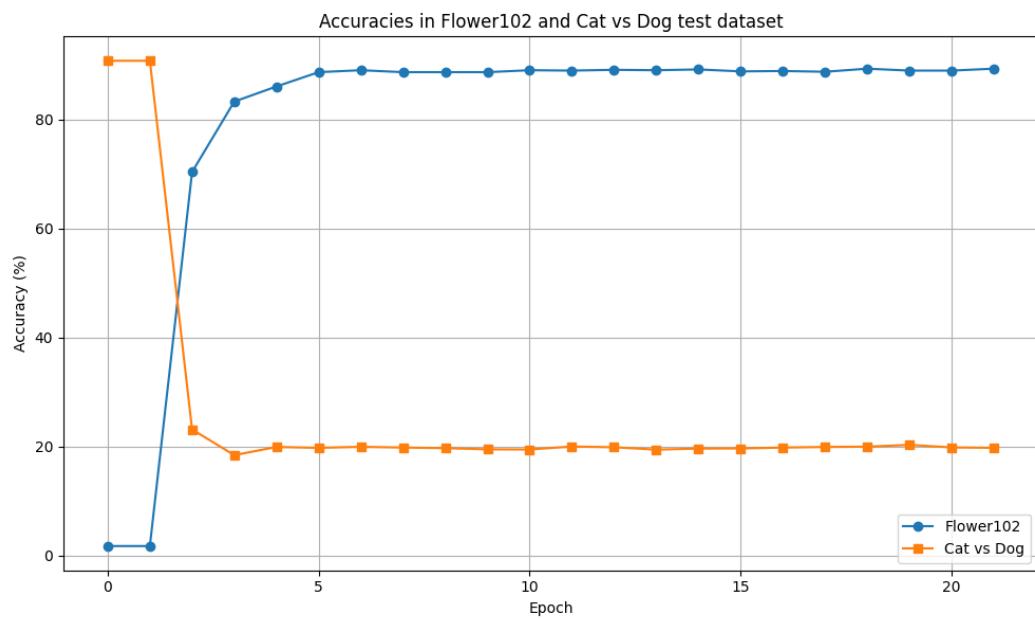


Figure 13: Moderately aggressive strategy

