Pet Breeds Multiclass Classification

Muni Eshwar Evakattu

November 2024

1 Overview

This project aims to develop a deep learning model using PyTorch to classify images of different pet breeds, including both dogs and cats. Accurately classifying pet breeds can have applications in fields like pet adoption, veterinary care, and animal research.

2 Dataset

The dataset used in this project contains images of 23 of the most common pet breeds, including 15 dog breeds and 8 cat breeds. Each class has 170 images (except for the 'mumbai cat' class), with the images stored in separate folders. All images are in .jpg or .jpeg format.

- **Total Images:** 3,910 (23 classes, 170 images per class, except 'mumbai cat')
- **Classes:** 15 dog breeds, 8 cat breeds
- **Image Format:** .jpg, .jpeg
- **Preprocessing:** Images resized to 224x224, normalized to ImageNet standards

3 Sample Images from Dataset

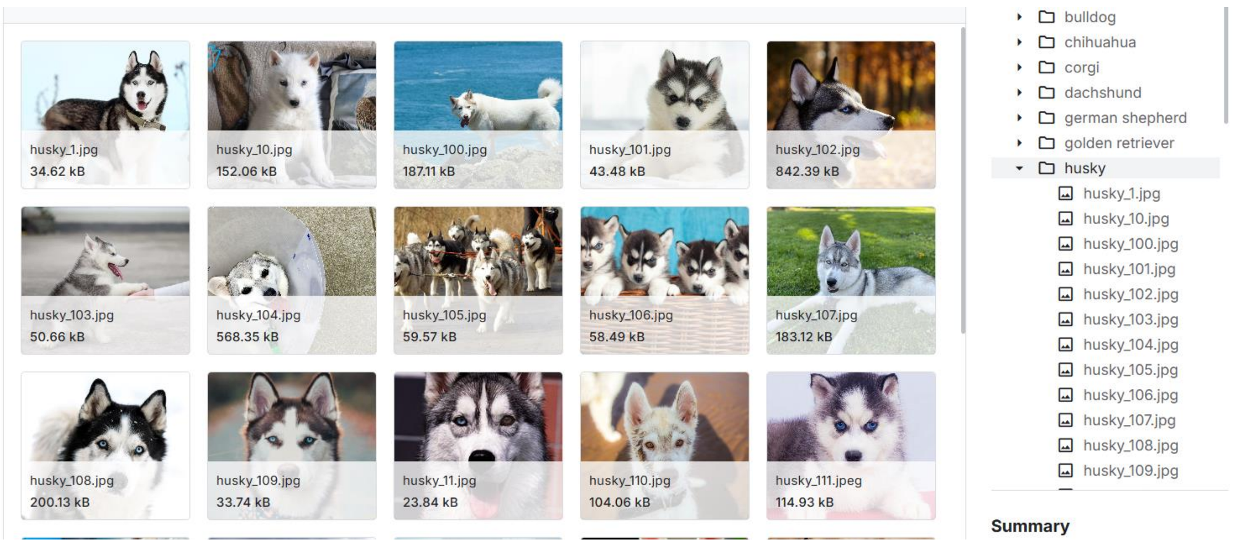


Figure 1: Sample pet images from different breed classes

4 Objectives

The main objectives of this project are:

1. Develop and evaluate multiple versions of a ResNet-based deep learning model for the pet breed multiclass classification task.
2. Analyze the performance of the models, including training accuracy, validation accuracy, test accuracy, and loss.
3. Compare the different model versions and identify the best-performing model for practical deployment.
4. Provide recommendations for further improving the model's generalization and classification performance.

5 Model Versions and Adjustments

5.1 Model Version 1

- **Base:** ResNet10 with modified pooling and classification layers.
- **Optimizer:** Adam, learning rate of 0.001.
- **Scheduler:** StepLR with gamma of 0.1, step every 7 epochs.
- **Results:**

 Base Model: ResNet18 with modified classification head, pretrained weights, and frozen early layers to prevent overfitting.
 Modifications:

1. **Layers Frozen:** All but the last few layers, keeping the final layers trainable.
2. **Classification Head:** Added Batch Normalization and Dropout layers for regularization.
3. **Fully Connected Layers:**

First layer: BatchNorm1d(num_features), Dropout(0.4), Linear(num_features, 256), ReLU.
Second layer: BatchNorm1d(256), Dropout(0.4), Linear(256, 24).

 Optimizer: Specified in config (default: Adam with a learning rate of 0.001).
 Scheduler: StepLR with gamma=0.1, stepping down every 7 epochs (or as configured).
 Results:

Train Loss: 1.1520, Train Acc: 83.30%
Validation Loss: 1.1287, Validation Acc: 83.71%
Test Loss: 1.0855, Test Acc: 86.04%

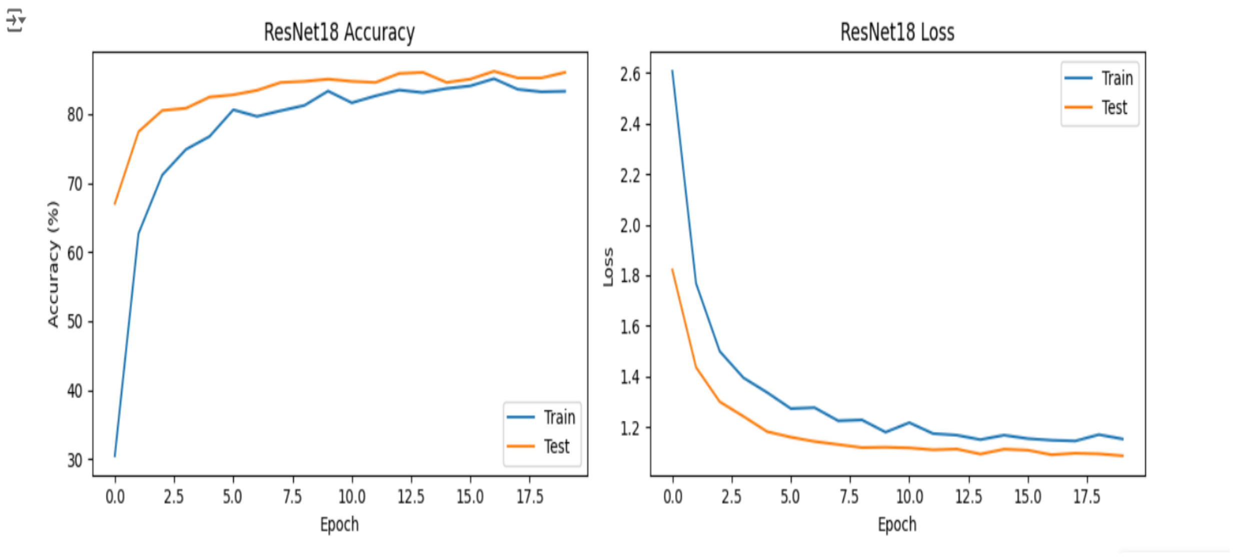


Figure 2: Model version 1 Training and Testing Loss and Accuracy curves across epocs

Observations:

The ResNet18 model shows strong learning in the early epochs, with both training and test accuracy quickly increasing and stabilizing around 80-85%. The test accuracy consistently matches or slightly exceeds training accuracy, indicating good generalization without overfitting. The gradual reduction and convergence of loss for both sets further confirm that the model is well-optimized for this task.

5.2 Model Version 2

- **Base Model:** ResNet50 with modified classification head, pretrained weights.

- **Modifications:**

  Classification Head: Added a dropout layer for regularization.

  Fully Connected Layer: Dropout(0.5), Linear(num_features, 24).

- **Optimizer:** Specified in config (default: Adam with a learning rate of 0.001).
- **Scheduler**: StepLR with gamma=0.1, stepping down every 7 epochs (or as configured).
- **Results:**

Train Loss: 0.6856, Train Acc: 99.33%
Validation Loss: 0.9729, Validation Acc: 87.92%
Test Loss: 0.9621, Test Acc: 89.12%

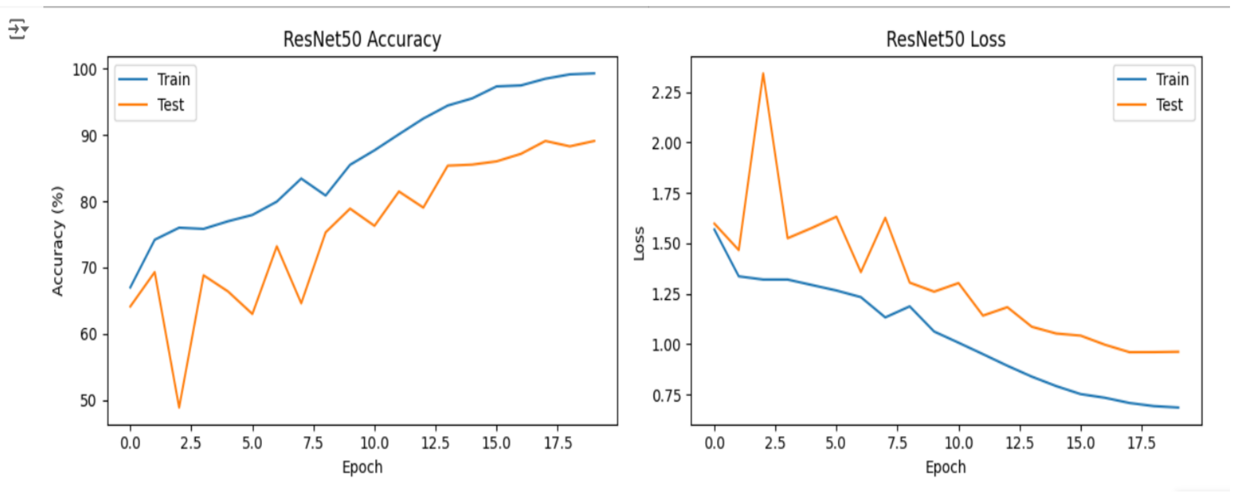


Figure 3: Model version 1.1 Training and Testing Loss and Accuracy curves across epochs.

Observations:

The training accuracy of the ResNet50 model steadily improves over epochs, reaching close to 100%, while the test accuracy increases more slowly and fluctuates, stabilizing around 80%. This trend suggests the model might be overfitting, as training accuracy is significantly higher than test accuracy. The training loss decreases smoothly, indicating effective learning during training. However, the test loss fluctuates and remains relatively high compared to the training loss, further indicating potential overfitting.

5.3 Model Version 3

- **Base Model:** ResNet18 with a custom classification head, trained from scratch (no pretrained weights).

- **Modifications:**

Classification Head: Added two dropout layers with a dropout probability of 0.3 for regularization.

Added batch normalization layers to stabilize learning.

Fully Connected Layers:

Linear Layer 1: Linear(num_features, 128) followed by ReLU activation.

Linear Layer 2: Linear(128, 24) for 24-class classification.

Fully Connected Layer: Dropout(0.5), Linear(num_features, 24).

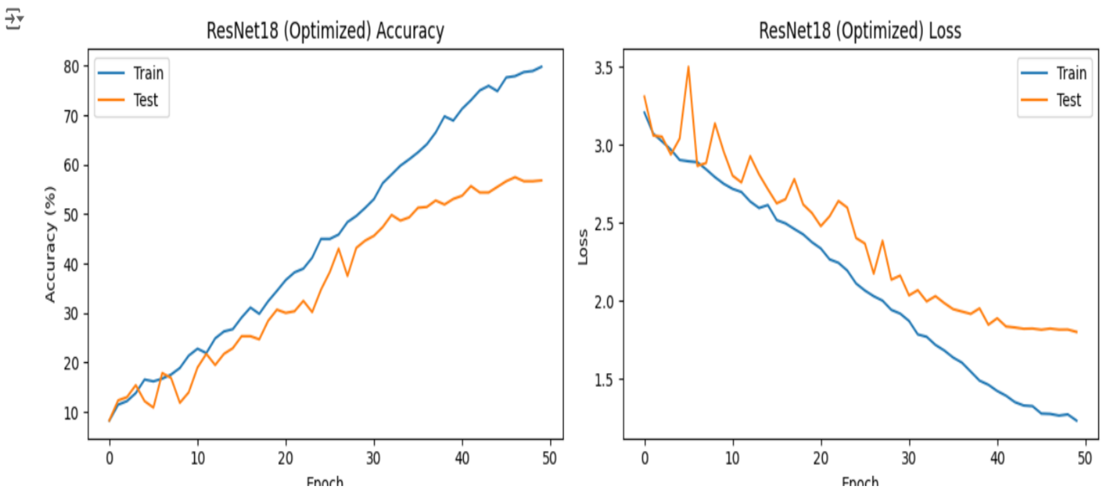
- **Optimizer:** Configured as Adam with a learning rate of 0.001, weight decay of 0.0001, and momentum of 0.9.
- **Scheduler**: ReduceLROnPlateau scheduler with a patience of 3 epochs and a factor of 0.1 for dynamic learning rate adjustment.

Training Configuration:

- **Epochs:** 50
- **Batch Size:** 32
- **Device:** GPU-enabled for faster training.
- **Results:**

Train Loss: 1.2346, Train Acc: 79.81%
Validation Loss: 1.7402, Validation Acc: 59.37%

```
Test Loss: 1.8022, Test Acc: 56.82%
```



Observations:

The training accuracy of the optimized ResNet18 model increases steadily across the 50 epochs, reaching around 80%. Test accuracy also improves, but at a slower rate, stabilizing around 40-45%. This suggests the model is learning effectively on the training set, but there may be a moderate level of overfitting due to the gap between train and test accuracy. The training loss decreases consistently, indicating that the model's optimization is successful. However, the test loss, although decreasing overall, shows fluctuations, which may indicate the model's difficulty in generalizing to unseen data. The regularization in the classification head helps improve performance, but additional regularization or tuning may be required for further test accuracy improvement.

5.4 Model Version 4

- **Base Model:** EfficientNet-B0 with pretrained weights, modified for enhanced performance and regularization.

- **Modifications:**

Layer Freezing:

Early layers in blocks.0, blocks.1, and blocks.2 are frozen to accelerate training and reduce overfitting.

Classification Head:

- Enhanced for better regularization and performance:
- Batch normalization layer added before the fully connected layers.
- Dropout layers with a 0.3 dropout probability for regularization.
- Fully Connected Layers:
    - Linear Layer 1: Linear(in_features, 512) followed by ReLU activation.
    - Linear Layer 2: Linear(512, 24) for 24-class classification.

.

Optimizer and Scheduler:

- **Optimizer:** Adam (learning rate: 0.001, weight decay: 0.0001, momentum: 0.9).
- **Scheduler:** ReduceLROnPlateau with a patience of 3 epochs and a factor of 0.1, dynamically

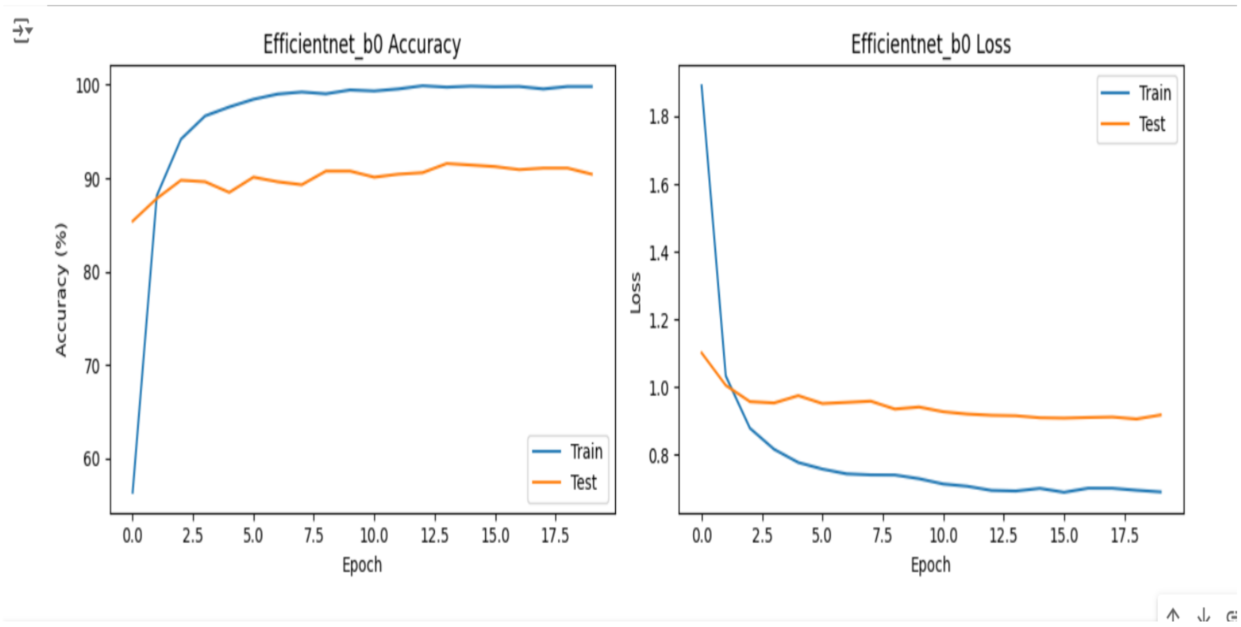
adjusting learning rate on performance plateaus.

Training Configuration:

- **Epochs:** 50
- **Batch Size:** 32
- **Device:** GPU-enabled for efficient training.

- **Results:**

Train Loss: 0.6905, Train Acc: 99.81%
Validation Loss: 0.9364, Validation Acc: 90.54%
Test Loss: 0.9178, Test Acc: 90.42%



Observations:

The EfficientNet-B0 model shows high training accuracy, approaching nearly 100%, while test accuracy stabilizes around 90%, indicating some overfitting. The training loss decreases steadily, but the test loss remains higher, confirming this trend. Additional regularization or data augmentation could help improve generalization and reduce overfitting.

5.5 Model Version 5

- **Base Model:**

  Architecture: EfficientNet-B0

  Pretrained: No (trained from scratch)

- **Modifications:**

  Classification Head:

  - Enhanced for better performance and regularization.
  - **Batch Normalization**: Added before fully connected layers.
  - **Dropout**: Dropout layers with a probability of 0.3 for regularization.

  Fully Connected Layers:

  Linear Layer 1: Linear(in_features, 128) followed by ReLU activation.

  Linear Layer 2: Linear(128, 24) for 24-class classification.

Optimizer and Scheduler

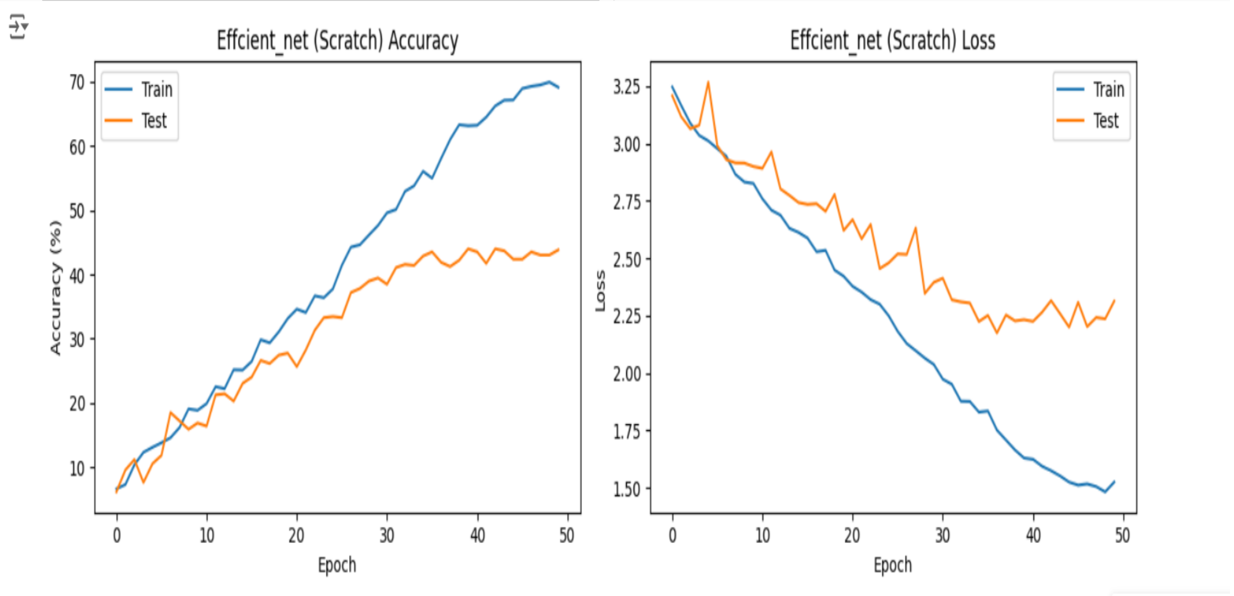
- **Optimizer: Adam**
    - Learning Rate: 0.001
    - Weight Decay: 0.0001
    - Momentum: 0.9
- **Scheduler: ReduceLROnPlateau**
    - Patience: 3 epochs
    - Factor: 0.1
    - Dynamically adjusts learning rate when performance plateaus.

Training Configuration

- Epochs: 50
- Batch Size: 32
- Device: GPU-enabled for efficient training

- **Results:**

Train Loss: 1.5245, Train Acc: 69.12%
Validation Loss: 2.4143, Validation Acc: 47.81%
Test Loss: 2.3134, Test Acc: 43.83%



Observations:
The graphs show the training and testing performance of an EfficientNet model trained from scratch.

1. **Accuracy**: The training accuracy steadily improves, reaching around 65% by epoch 50, while the test accuracy plateaus at approximately 40%. This indicates that the model is learning but may be overfitting to the training data.
2. **Loss**: Both training and test loss decrease over time, with the training loss reducing more significantly than the test loss. The test loss plateaus after around epoch 20, suggesting that further improvements in training do not generalize well to unseen data.

This model is trained from scratch without any influence of pre-defined weights, generally models in these cases take more time and many epochs to converge. Due to computational complexities, I could only run 50 epoch, I feel this model will do better with more epochs.

6. Conclusion

In this project, we developed and evaluated multiple deep learning models for multiclass classification of pet breeds (dogs and cats) using images. We explored five model versions, including ResNet18, ResNet50, and EfficientNet-B0, with different configurations and modifications.

- **Pretrained models** (ResNet18 and EfficientNet-B0) performed the best, achieving test accuracies around 86-90%, with EfficientNet-B0 showing the highest accuracy at 90%.
- **Training from scratch** (ResNet18 and EfficientNet-B0) struggled to generalize, with lower test accuracies (~56% and 43%, respectively), highlighting the challenge of training without pretrained weights.
- **Regularization techniques** like dropout and batch normalization helped stabilize training, but overfitting was still an issue, particularly with deeper models.
- **Transfer learning** proved to be more effective for this task, as pretrained models performed better and required fewer epochs to achieve strong results.

Key learnings include the importance of pretrained models, the effectiveness of regularization, and the need for more training epochs for models trained from scratch. Future work could focus on fine-tuning models, using data augmentation, and training for longer periods to improve performance.

Links:
https://colab.research.google.com/drive/1QkUY3ZaUtGbro8IFRY1XWyjtVhSYXiJc?usp=sharing
https://colab.research.google.com/drive/1m0r11LfkkSTv4F8r1f04pCcRAst6xqF1?usp=sharing