Accessible Dog Breed Recognition for Lost Found Services and Pet Shop Management

Vakati, Sai Vyshali, vakati.s@northeastern.edu
Sirivella, Siva Abhishek, sirivella.s@northeastern.edu
Nalamalapu, Yaswanth Reddy, nalamalapu.y@northeastern.edu

Abstract

This project addresses the need for efficient and accurate dog breed classification in resource-constrained environments, such as mobile devices or lightweight platforms. Using the lightweight MobileNet architecture, which minimizes trainable weights, the solution is practical and accessible. Images were preprocessed with bounding box cropping, data augmentation, and standardization. Custom layers, including a convolutional layer with batch normalization and max-pooling, were added to adapt the architec- ture. Fine-tuning was performed by un- freezing 10, 30, and 50 layers, replacing ReLU with ELU to prevent dead neurons, and optimizing alpha values. Training was stabilized by reducing the learning rate, increasing the batch size from 32 to 64, and using dropout and early stopping to prevent overfitting. An inverted residual block was tested as a replacement for the added convolutional layer, but while it re- duced parameters, it resulted in lower ac- curacy. Kernel size experiments for the depth wise convolution layer also failed to improve performance. The final model, with a 17.4% increase in parameters for a 2.8% accuracy gain, uses the added con- volutional layer, achieving 80% accuracy while maintaining significantly fewer parameters than heavy architectures like In- ception and VGGNet. This efficient, accu- rate solution is accessible for diverse usersand applications.

1 Introduction

1.1 Overview of the Project

The ability to identify dog breeds quickly and accurately has significant applications in both rescue and commercial settings. This project addresses the challenge of automating dog breed recognition in resource-limited environments using deep learning. The solution is built with the lightweight MobileNetV2 architecture, making it suitable for deployment on mobile and low power devices. The system aims to classify dog breeds with high accuracy from images, facilitating lost and found dog recovery, efficient pet shop cataloging, and shelter management. By using transfer learning and fine-tuning strategies, the model achieves consistent performance while minimizing computational resource usage.

1.2 Motivation

Dog breed identification is a critical task in multiple domains. In lost and found scenarios, quick breed recognition can expedite the recovery process and reunite pets with their owners. Shelters and rescue organizations benefit from automated systems that refine breed identification, reducing manual effort and improving operational efficiency. Similarly, pet shops require an organized catalog of dog breeds for inventory management and record-keeping. However, existing systems based on advanced architectures like VGGNet and Inception are resource-intensive, requiring substantial computational power, which limits their usability in resource-constrained environments such as mobile devices or small-scale operations. This project addresses these challenges by using the lightweight MobileNetV2 architecture, which significantly reduces computational overhead while maintaining high accuracy. By using MobileNetV2, the solution ensures accessibility, scalability, and ease of deployment on mobile devices, bridging the gap between efficiency and practical usability in both rescue and commercial domains.

1.3 Dataset

This project uses the Stanford Dogs Dataset for dog breed classification task. The dataset consists of 20,580 images across 120 different breeds. It includes 12,000 images for training and 8,580 images for testing, with each image labeled according to its breed. Additionally, the dataset provides bounding box coordinates for each image, enabling precise cropping around the dogs to improve model focus and accuracy.

2 Background

It is essential to have systems that are not only highly accurate but also lightweight, accessible, and easily deployable across a variety of platforms.

Current state-of-the-art solutions for dog breed classification rely heavily on hybrid models that integrate multiple convolutional neural network (CNN) architectures, such as ResNet and VGGNet. These models leverage advanced data augmentation techniques like image rotations, flips, and brightness adjustments to expand training datasets and prevent overfitting. Additionally, Principal Component Analysis (PCA) has proven effective in refining feature selection. By combining CNN-based feature extraction with classifiers like Support Vector Machines (SVM), these models have achieved exceptional accuracy rates of up to 95.24% for 120 dog breeds. VGG19 achieves a training accuracy of 85% and a validation accuracy of 55%, while InceptionV3 achieves a training accuracy of 52.49% and a validation accuracy of 34.84%[3]. Although these models are highly accurate, their computational complexity and resource demands limit their practicality in real-world applications.

Despite their high accuracy, these hybrid models pose challenges for deployment in resource-limited environments. Their significant computational requirements make them impractical for small-scale platforms, such as mobile devices or low-budget facilities. Furthermore, the complexity of hyperparameter tuning in hybrid models, where CNNs, PCA, and SVM each require specific settings, adds to the difficulty of optimizing these systems. Training and maintaining such resource-intensive systems also involve considerable computational and financial overhead.

This project aims to address these challenges by leveraging MobileNetV2, a lightweight CNN architecture optimized for mobile and embedded devices. By providing a scalable, resource-efficient solution, the project ensures accessibility while maintaining accuracy, enabling deployment in both rescue operations and commercial settings.

3 Approach

3.1 Data Preparation

1. **Loading and Resizing Images:** The dataset was loaded directly from the TensorFlow. Images were resized to (224, 224) to match the input dimensions required by MobileNetV2.

2. **Cropping Using Bounding Boxes:** The Stanford Dogs Dataset includes bounding box annotations, which were used to crop images around the dogs allowing the model to focus on the relevant features of the dogs.

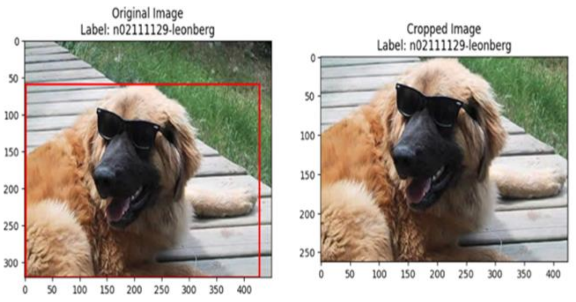


Figure 1: Sample image from dataset

3. **Data Augmentation:** Various augmentation techniques, such as random rotations and contrast adjustments, were applied to help the model generalize well to variations in the images.

4. **Batch Size:** A batch size of 32 was used during training to balance memory usage and computational efficiency.

3.2 Model Architecture and Training

1. Model training began with a baseline approach using MobileNetV2. A dense layer with 120 units and a softmax activation function for classification into 120 classes was utilized. The Adam optimizer with the default learning rate was utilized, and early

stopping with a patience of 5 epochs was implemented to save computational resources

2. Next, 10 layers from the bottom of the model were unfrozen and made trainable, but this configuration demonstrated a similar level of performance as the initial baseline. To investigate further, activation distributions were analyzed to check for the impact of dead neurons. The histograms revealed a significant number of dead neurons. To address this, ReLU activations were replaced with ELU, applied with the default alpha value of 1, which effectively mitigated the issue.

3. **Experimenting with Alpha:** After addressing the dead neuron issue by replacing ReLU with ELU, experimented with alpha values (0.5, 1.5, and 2). The values of 1.5 and 2 yielded the highest accuracy, with 2 showing slightly better performance. However, alpha 1.5 was chosen due to its smoother updates during training, leading to more stable convergence, despite the marginal difference in accuracy.

4. To stabilize the training process and improve convergence, the learning rate in the Adam optimizer was decreased from the default value to 0.0001. The batch size was increased to 64 to reduce oscillations. Additionally, a convolutional layer with a kernel size of 5 and padding='same' was added, along with batch normalization, to improve learning stability and facilitate smoother convergence.

5. The number of unfrozen layers was again set to 10, and the configuration was tested with a learning rate of 0.0001. A dropout layer with a rate of 0.2 was added to stabilize training and prevent overfitting. Next, the number of unfrozen layers was increased from 10 to 30, requiring a reduction in the learning rate to 0.00001 to accommodate the increased complexity and ensure more stable convergence. The dropout layer with a rate of 0.2 was retained for consistency across configurations. Finally, the number of unfrozen layers was extended to 50.

6. For the 50-layer configuration, additional experimentation with dropout rates was conducted. While the dropout rate of 0.2 was retained during earlier trials for the 10, 30, and 50 layer configurations, further testing was performed with higher dropout rates of 0.3 and 0.5. Among these, the dropout rate of 0.5 proved to be the most effective, resolving overfitting issues observed in prior experiments and resulting in the highest accuracy of 80%.

7. To improve model efficiency while maintaining similar accuracy, the standard convolutional layer was replaced with an inverted residual block (with depthwise separable convolutions and expansion). Both configurations provided almost the same accuracy, but the inverted residual block significantly reduced the number of parameters.
   - Standard Convolution (5x5, input=1280, output=512): The Conv2D layer adds approximately 17.4% more parameters than the Inverted Residual Block, achieving 80% accuracy.
   - Inverted Residual Block (Expansion=6, input=1280, output=512): Significantly reduced the number of parameters achieving 77.20% accuracy.

8. **Impact of Different Kernels on Model Performance:** The effect of kernel size on model accuracy was explored by testing different kernel sizes in the depth wise convolution layers of the inverted residual block. The following results were observed:
   - 2x2 Kernel: Achieved 73.00% accuracy.
   - 3x3 Kernel: Achieved 73.26% accuracy.
   - 4x4 Kernel: Achieved 73.52% accuracy.
   - 5x5 Kernel: Achieved 77.20% accuracy.
   - 6x6 Kernel: Achieved 73.80% accuracy.

   The 5x5 kernel provided the best accuracy at 77.20%, while larger kernels (6x6) led to a slight decrease in performance. The optimal choice for this model was found to be the 5x5 kernel, balancing accuracy and computational efficiency.

Therefore, the configuration with the standard convolutional layer was chosen as the final model, as it provided the best balance of accuracy, with a considerable increase in the number of parameters.

4 Results

The results demonstrate the impact of various optimizations and experiments conducted during the project. Experimentation with the alpha parameter of the ELU activation function revealed that while an alpha value of 2 achieved higher accuracy, 1.5 was chosen for its greater stability. Testing different dropout rates showed that a dropout rate of 0.5 provided the best results, achieving a final accuracy of 80%. The model was trained for 50 epochs, with training and validation loss and accuracy graphs showing steady improvements and minimal overfitting. Kernel size experimentation for depthwise convolution layers highlighted trade-offs between accuracy and parameter efficiency. Additionally, a classification report was generated, providing detailed insights into the precision, recall, F1-score, and support for each class, further validating the model's reliability and performance. Finally, predictions on the test set demonstrated the model's effective performance on unseen data.

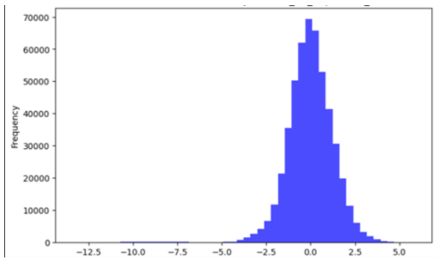


Figure 2: Histogram

Alpha	Accuracy
0.5	69.83%
1	68.16%
1.5	70.30%
2	70.52%

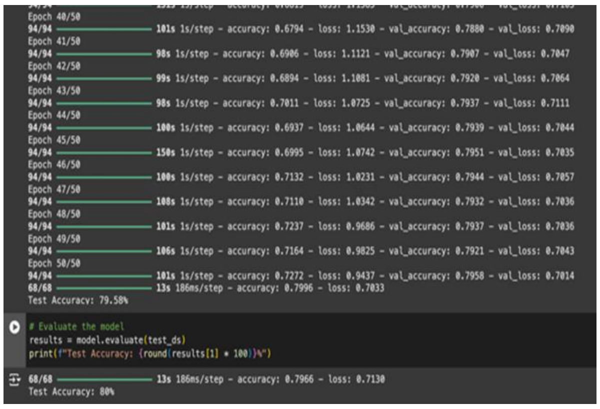
Table 1: Experimentation with Alpha

Dropout Rate	Accuracy
0.2	77.60%
0.3	77.39%
0.5	80%

Table 2: Experimentation with Dropout

Kernel Size	Accuracy
2 x 2	73.00%
3 x 3	73.26%
4 x 4	73.52%
5 x 5	77.20%
6 x 6	73.80%

Table 3: Experimentation with Kernel size



```
Epoch 40/50
94/94 ━━━━━━━━━━━━━━━━━━━━ 101s 1s/step - accuracy: 0.6794 - loss: 1.1530 - val_accuracy: 0.7880 - val_loss: 0.7090
Epoch 41/50
94/94 ━━━━━━━━━━━━━━━━━━━━ 98s 1s/step - accuracy: 0.6906 - loss: 1.1121 - val_accuracy: 0.7907 - val_loss: 0.7047
Epoch 42/50
94/94 ━━━━━━━━━━━━━━━━━━━━ 99s 1s/step - accuracy: 0.6894 - loss: 1.1081 - val_accuracy: 0.7920 - val_loss: 0.7064
Epoch 43/50
94/94 ━━━━━━━━━━━━━━━━━━━━ 98s 1s/step - accuracy: 0.7011 - loss: 1.0725 - val_accuracy: 0.7937 - val_loss: 0.7111
Epoch 44/50
94/94 ━━━━━━━━━━━━━━━━━━━━ 100s 1s/step - accuracy: 0.6937 - loss: 1.0644 - val_accuracy: 0.7939 - val_loss: 0.7044
Epoch 45/50
94/94 ━━━━━━━━━━━━━━━━━━━━ 150s 1s/step - accuracy: 0.6995 - loss: 1.0742 - val_accuracy: 0.7951 - val_loss: 0.7035
Epoch 46/50
94/94 ━━━━━━━━━━━━━━━━━━━━ 100s 1s/step - accuracy: 0.7132 - loss: 1.0231 - val_accuracy: 0.7944 - val_loss: 0.7057
Epoch 47/50
94/94 ━━━━━━━━━━━━━━━━━━━━ 108s 1s/step - accuracy: 0.7110 - loss: 1.0342 - val_accuracy: 0.7932 - val_loss: 0.7036
Epoch 48/50
94/94 ━━━━━━━━━━━━━━━━━━━━ 101s 1s/step - accuracy: 0.7237 - loss: 0.9686 - val_accuracy: 0.7937 - val_loss: 0.7036
Epoch 49/50
94/94 ━━━━━━━━━━━━━━━━━━━━ 106s 1s/step - accuracy: 0.7164 - loss: 0.9825 - val_accuracy: 0.7921 - val_loss: 0.7043
Epoch 50/50
94/94 ━━━━━━━━━━━━━━━━━━━━ 101s 1s/step - accuracy: 0.7272 - loss: 0.9437 - val_accuracy: 0.7958 - val_loss: 0.7014
68/68 ━━━━━━━━━━━━━━━━━━━━ 13s 186ms/step - accuracy: 0.7996 - loss: 0.7033
Test Accuracy: 79.58%
```

```
# Evaluate the model
results = model.evaluate(test_ds)
print(f"Test Accuracy: {round(results[1] * 100)}%")
```

```
68/68 ━━━━━━━━━━━━━━━━━━━━ 13s 186ms/step - accuracy: 0.7966 - loss: 0.7130
Test Accuracy: 80%
```

Figure 3: Training over 50 Epochs with an Accuracy of 80%

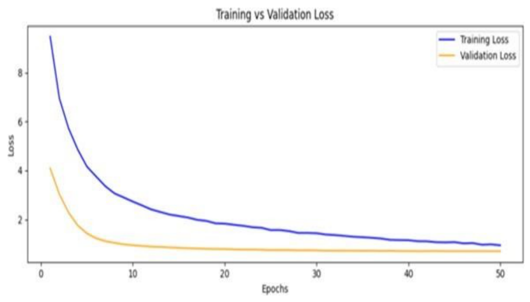


Figure 4: Training and Validation Loss

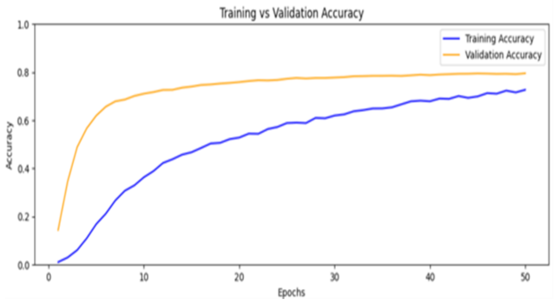
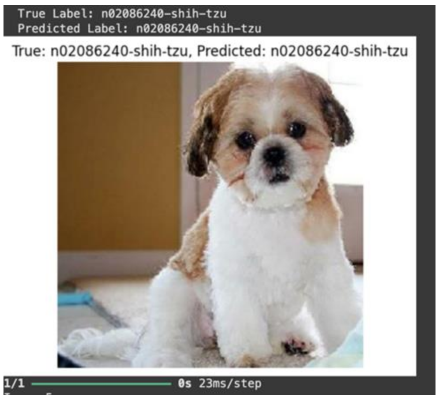
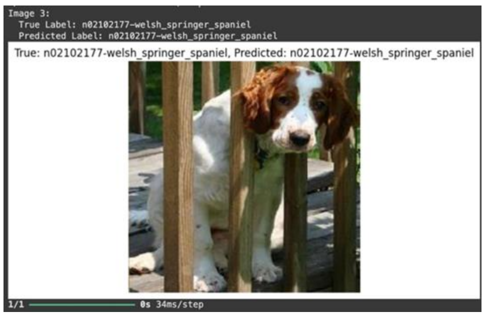


Figure 5: Training and validation Accuracy



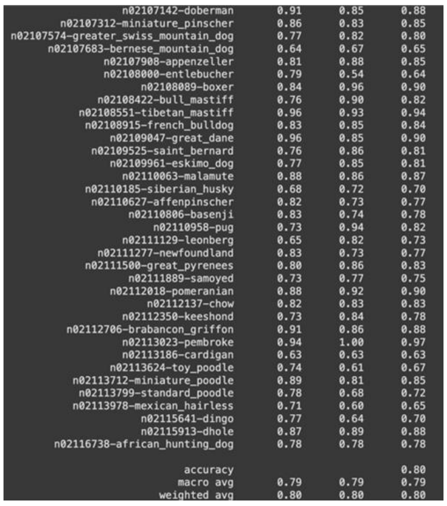
```
True Label: n02086240-shih-tzu
Predicted Label: n02086240-shih-tzu
True: n02086240-shih-tzu, Predicted: n02086240-shih-tzu
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
```

Figure 6: Prediction 1



```
Image 3:
  True Label: n02102177-welsh_springer_spaniel
  Predicted Label: n02102177-welsh_springer_spaniel
True: n02102177-welsh_springer_spaniel, Predicted: n02102177-welsh_springer_spaniel
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
```

Figure 7: Prediction 2



```
                n02107142-doberman       0.91      0.85      0.88
       n02107312-miniature_pinscher      0.86      0.83      0.85
n02107574-greater_swiss_mountain_dog     0.77      0.82      0.80
     n02107683-bernese_mountain_dog      0.64      0.67      0.65
             n02107908-appenzeller       0.81      0.88      0.85
             n02108000-entlebucher       0.79      0.54      0.64
                   n02108089-boxer       0.84      0.96      0.90
            n02108422-bull_mastiff       0.76      0.90      0.82
         n02108551-tibetan_mastiff       0.96      0.93      0.94
          n02108915-french_bulldog       0.83      0.85      0.84
              n02109047-great_dane       0.96      0.85      0.90
          n02109525-saint_bernard        0.76      0.86      0.81
              n02109961-eskimo_dog       0.77      0.85      0.81
                n02110063-malamute       0.88      0.86      0.87
         n02110185-siberian_husky        0.68      0.72      0.70
          n02110627-affenpinscher        0.82      0.73      0.77
                 n02110806-basenji       0.83      0.74      0.78
                     n02110958-pug       0.73      0.94      0.82
                n02111129-leonberg       0.65      0.82      0.73
            n02111277-newfoundland       0.83      0.73      0.77
          n02111500-great_pyrenees       0.80      0.86      0.83
                 n02111889-samoyed       0.73      0.77      0.75
              n02112018-pomeranian       0.88      0.92      0.90
                    n02112137-chow       0.82      0.83      0.83
                n02112350-keeshond       0.73      0.84      0.78
       n02112706-brabancon_griffon       0.91      0.86      0.88
                n02113023-pembroke       0.94      1.00      0.97
                n02113186-cardigan       0.63      0.63      0.63
              n02113624-toy_poodle       0.74      0.61      0.67
        n02113712-miniature_poodle       0.89      0.81      0.85
         n02113799-standard_poodle       0.78      0.68      0.72
        n02113978-mexican_hairless       0.71      0.60      0.65
                   n02115641-dingo       0.77      0.64      0.70
                   n02115913-dhole       0.87      0.89      0.88
     n02116738-african_hunting_dog       0.78      0.78      0.78

                           accuracy                          0.80
                          macro avg      0.79      0.79      0.79
                       weighted avg      0.80      0.80      0.80
```

Figure 8: Classification Report

5 Discussion of Results

5.1 Histogram of Activations [Figure 2]

The histogram represents the activation distribution for one specific layer, but similar patterns were observed across multiple layers, with a majority of activation values being negative.

Impact of ReLU: Since ReLU sets all negative activations to zero, applying it to these layers would result in a significant loss of valuable features across multiple layers, potentially hindering the model's performance.

To address this issue, ELU (Exponential Linear Unit) was chosen as the activation function. Unlike ReLU, ELU retains negative values preserving meaningful features.

This observation across several layers guided the decision to replace ReLU with ELU, which improved feature representation, reduced the risk of dead neurons, and enhanced overall model performance and stability.

5.2 Experimentation with Alpha

The alpha parameter in the ELU activation function controls the smoothness of the curve, impacting how well the model learns. Experimentation was done with different alpha values (0.5, 1, 1.5, and 2) and recorded their corresponding accuracies. Although an alpha value of 2 achieved the highest accuracy (70.52%), 1.5 was chosen because it provided smoother and more stable training.

5.3 Experimentation with Dropout

To address overfitting observed during training, Experimentation was done with different dropout rates (0.2, 0.3, and 0.5) to determine the optimal rate for achieving a balance between regularization and model performance. A dropout rate of 0.5 achieved the highest accuracy of 80%, effectively mitigating overfitting while preserving the model's ability to generalize. Although dropout rates of 0.2 and 0.3 resulted in slightly lower accuracies, they provided insights into the impact of regularization strength on the model's performance. The decision to use a dropout rate of 0.5 in the final model was guided by these results, as it provided the best trade-off between preventing overfitting and maintaining high accuracy.

5.4 Analysis of Training vs Validation Loss and Accuracy

Training and Validation Loss: The training loss steadily decreases throughout the 50 epochs, showing effective learning. The validation loss follows a similar downward trend, stabilizing toward the end with a small gap between the two, indicating good generalization without significant overfitting.

Training and Validation Accuracy: Both training and validation accuracy improve steadily over the epochs. The validation accuracy stabilizes around 80%, closely tracking the training accuracy, which indicates consistent performance across both datasets.

Smooth Convergence: The graphs show smooth and stable convergence, highlighting the effectiveness of techniques such as ELU activation, optimized dropout rates, and fine-tuned learning rates for stabilizing training and preventing oscillations.

5.5 Calculation of Parameters for the added Layers

Expansion phase
(1*1*1280)*(6*1280)
= 1280*7680
= 9,830,400
Depth wise Separable Convolution
5*5*7680
= 192,000
Projection Phase
(1*1*7680)*512
= 3,932,160
Total: 13,954,560
Conv2D
(5*5*1280)*512
= 16,384,000
Percentage Difference in parameters
((16,384,000-13,954,560)/13,954,560)*100
= 17.4%

5.6 Experimentation with Kernel Size

Optimal Kernel Size: A kernel size of 5*5 achieved the highest accuracy of 77.20%, significantly outperforming smaller and larger kernel sizes.

Comparison with Convolutional Layer: Despite optimizing the depthwise convolution kernel size, the highest accuracy (77.20%) achieved with the inverted residual block was still lower than the accuracy achieved by the standard convolutional layer (80%).

6 Conclusion

This project successfully developed an efficient and accurate dog breed classification model for resource-constrained environments. By using the lightweight MobileNetV2 architecture, the model achieved a final accuracy of 80% while maintaining a low parameter count, making it highly efficient and scalable. Key optimizations, such as replacing ReLU with ELU to address dead neurons, fine-tuning layers, and carefully adjusting learning rates, contributed to stable training and improved performance.

Dropout rates were systematically tested to mitigate overfitting, with a dropout rate of 0.5 providing the best balance between generalization and accuracy. Preprocessing techniques, including bounding box cropping and data augmentation, enhanced the model's ability to focus on relevant features, ensuring good performance across diverse input variations.

Despite the slight increase in parameters from adding a convolutional layer, the resulting accuracy gains justified the trade-off, making the final configuration an effective solution for lightweight dog breed classification. This project demonstrates the feasibility of creating an accessible and accurate classification system that balances efficiency and performance, making it suitable for diverse real-world applications such as lost-and-found services, shelter management, and pet shop cataloging

7 Future Works

While the current model achieves a strong balance between accuracy and efficiency, there are several areas for further exploration and experimentation:

Data Augmentation: Experimenting with additional data augmentation techniques could enhance the model's ability to generalize to diverse real-world scenarios.

Kernel Size Optimization in Convolutional Layer: Since kernel size experimentation for depthwise convolution layers yielded valuable insights, future work could extend this analysis to the standard convolutional layer to determine the impact of kernel size on feature extraction and overall model performance.

Regularization Techniques: Beyond dropout, experimenting with L1 or L2 regularization for the unfrozen layers could help in reducing overfitting and ensuring smoother weight updates.

Separate Learning Rate Tuning: Since the learning rate was adjusted based on model behavior during experiments, conducting a more systematic learning rate search (e.g., using a grid search) could identify the optimal learning rate, potentially improving training efficiency and stability.

References

[1] Khan, Safdar, Doohan, Nitika, Gupta, Manish, Jaffari, Sakina, Chourasia, Ankita, Joshi, Kriti, Panchal, Bhupendra. (2023). Hybrid Deep Learning Approach for Enhanced Animal Breed Classification and Prediction. *Traitement du Signal*, **40**, 2087-2099. `https://doi.org/10.18280/ts.400526`.

[2] Cui, Y., Tang, B., Wu, G., Li, L., Zhang, X., Du, Z., Zhao, W. Classification of Dog Breeds Using Convolutional Neural Network Models and Support Vector Machine. *Bioengineering (Basel)*, **11**(11):1157. Published November 17, 2024. `https://doi.org/10.3390/bioengineering11111157`. PMID: 39593817; PMCID: PMC11591900.

[3] Valarmathi, B., Gupta, Srinivasa, Gopalakrishnan, Prakash, Reddy, R., Saravanan, S., Palanisamy, Shanmugasundaram. (2023). Hybrid Deep Learning Algorithms for Dog Breed Identification—A Comparative Analysis. *IEEE Access*, **11**, 77228-77239. `https://doi.org/10.1109/ACCESS.2023.3297440`.