Dog Breed Classification using Deep Learning and Transfer Learning

Author: Jesper Nielsen
Mercantec
Machine Learning Elective Course
28/02/2025

Code Repository

GitHub Implementation: https://github.com/Jesper-N/dog-breed-classifier

Dataset: https://www.kaggle.com/datasets/khushikhushikhushi/dog-breed-image-dataset

Abstract

This white paper presents the development of a highly accurate dog breed classification system using convolutional neural networks and transfer learning techniques. The model can identify 10 popular dog breeds with approximately 99% validation accuracy, demonstrating the effectiveness of leveraging pre-trained models for specific image classification tasks. Using MobileNetV2 as the base architecture allows for efficient performance on standard computing hardware without requiring specialized GPU acceleration. This paper details the methodology employed, including data preparation, model architecture, training approach, and evaluation metrics. The results show exceptional accuracy across all breeds, with minimal confusion between similar-looking breeds. This implementation provides a practical example of how modern deep learning techniques can be applied to create effective image classification systems with limited computational resources.

1. Introduction

The task of automatically identifying dog breeds from images presents an interesting computer vision challenge with numerous practical applications. From veterinary medicine and animal shelters to smartphone applications for pet owners, the ability to accurately classify dog breeds from photographs can enhance service quality and user experience across multiple domains. This project addresses the classification of 10 distinct dog breeds using deep learning techniques optimized for standard computing environments.

Dog breed classification serves as an excellent case study in fine-grained image classification. Unlike general object recognition, breed identification requires the model to distinguish between categories with subtle differences in physical characteristics, making it particularly challenging. Some breeds share similar coat colors, body structures, or facial features, requiring models to learn nuanced distinctions.

Recent advancements in deep learning, particularly in convolutional neural networks (CNNs) and transfer learning, have made it possible to achieve high accuracy in image classification tasks without requiring extensive computational resources or massive labeled datasets. This project leverages these technologies to build an efficient and accurate dog breed classifier.

The implementation focuses on balancing accuracy with efficiency, ensuring the model can run on standard hardware without requiring specialized GPU acceleration. This approach makes the solution more accessible and deployable in various real-world scenarios where computational resources may be limited.

2. Problem Statement

2.1 Challenges in Dog Breed Classification

Automatic dog breed classification presents several unique challenges:

1. Fine-grained visual categorization: Unlike general object recognition, breed identification requires distinguishing between visually similar categories.
2. Intra-class variation: Dogs of the same breed can vary significantly in appearance due to differences in pose, lighting, background, and individual variations.

3. Limited training data: Obtaining large, balanced datasets for all dog breeds can be difficult, especially for rare breeds.
4. Computational efficiency: High-performing models often require significant computational resources, limiting their practical deployment.
5. Generalization: Models trained on specific datasets may not generalize well to real-world images taken under different conditions.

2.2 Current Approaches and Limitations

Traditional computer vision approaches to dog breed classification relied on hand-crafted features like color histograms, texture descriptors, and shape analysis. While these methods provided some level of accuracy, they required significant domain expertise and extensive feature engineering.

With the rise of deep learning, CNN-based approaches have become the standard for image classification tasks. Large models like ResNet, Inception, and EfficientNet have achieved impressive results on benchmark datasets. However, these models often require:

- Extensive computational resources for training and inference
- Large amounts of labeled training data
- Specialized hardware (GPUs) for acceptable performance

For practical applications, especially those running on standard hardware or mobile devices, these requirements present significant limitations. Additionally, the "black box" nature of deep neural networks can make it difficult to understand and address misclassifications.

3. Methodology

3.1 Dataset

The project utilizes a dataset comprising 10 dog breeds, with approximately 100 images per breed. The dataset includes the following breeds:

- Beagle
- Boxer
- Bulldog
- Dachshund
- German Shepherd
- Golden Retriever
- Labrador Retriever
- Poodle
- Rottweiler
- Yorkshire Terrier

This selection covers a diverse range of dog breeds with distinct physical characteristics, sizes, and coat types, allowing for a robust evaluation of the classification model's capabilities. The dataset is structured with separate directories for each breed, facilitating easy loading and processing.

3.2 Data Preparation and Augmentation

Data preparation is a critical component of the machine learning pipeline. The implementation includes several key steps:

3.2.1 Train-Validation Split

The dataset is split into training (80%) and validation (20%) sets using a stratified approach to ensure balanced representation of each breed:

```
def prepare_dataset():
    # Data directories
    train_dir = os.path.join(DATA_DIR, "train")
    val_dir = os.path.join(DATA_DIR, "validation")

    # Create directories if they don't exist
    for directory in [train_dir, val_dir]:
        if not os.path.exists(directory):
            os.makedirs(directory)

            # Create breed subdirectories and symlinks
            for breed in os.listdir(DATA_DIR):
                breed_path = os.path.join(DATA_DIR, breed)
                if os.path.isdir(breed_path) and breed not in ["train",
"validation"]:
                    breed_images = os.listdir(breed_path)

                    # Create breed directory in train and validation
                    train_breed_dir = os.path.join(train_dir, breed)
                    val_breed_dir = os.path.join(val_dir, breed)

                    os.makedirs(train_breed_dir, exist_ok=True)
                    os.makedirs(val_breed_dir, exist_ok=True)

                    # Split 80/20
                    split_idx = int(len(breed_images) * 0.8)

                    # Create links
                    for i, img in enumerate(breed_images):
                        src = os.path.abspath(os.path.join(breed_path,
img))

                        if i < split_idx:
                            dst = os.path.join(train_breed_dir, img)
                        else:
                            dst = os.path.join(val_breed_dir, img)

                        if not os.path.exists(dst):
                            os.symlink(src, dst)
```

The implementation uses symbolic links rather than copying files to conserve disk space while maintaining the organized directory structure required for TensorFlow's dataset loading utilities.

3.2.2 Data Augmentation

To increase the effective size of the training dataset and improve model generalization, data augmentation techniques are applied to the training images:

```
# Create data augmentation
data_augmentation = keras.Sequential([
    keras.layers.RandomFlip("horizontal"),
    keras.layers.RandomRotation(0.2),
    keras.layers.RandomZoom(0.2),
    keras.layers.RandomTranslation(0.1, 0.1),
])
```

These augmentations create variations of the training images by:

- Horizontally flipping images (simulating dogs facing different directions)
- Rotating images by up to 20% (accounting for different camera angles)
- Zooming in or out by up to 20% (simulating different distances from the camera)
- Translating images by up to 10% horizontally and vertically (accounting for different positioning)

3.2.3 Image Preprocessing

Before feeding the images to the model, they are preprocessed according to the requirements of the MobileNetV2 architecture:

```
# Performance optimization
train_ds = train_ds.map(
    lambda x, y: (data_augmentation(x), y),
    num_parallel_calls=tf.data.AUTOTUNE
).map(
    lambda x, y: (preprocess_input(x), y),
    num_parallel_calls=tf.data.AUTOTUNE
).cache().prefetch(tf.data.AUTOTUNE)
```

The preprocessing includes:

- Resizing images to 224×224 pixels
- Normalizing pixel values to the range expected by MobileNetV2
- Applying data augmentation (for training set only)
- Optimizing data loading with caching and prefetching for better performance

3.3 Model Architecture

3.3.1 Base Model Selection

MobileNetV2 was selected as the base model for this project because it balances performance with computational efficiency:

```
def create_model(num_classes):
    base_model = keras.applications.MobileNetV2(
        input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3),
        include_top=False,
        weights="imagenet"
    )

    # Freeze base model
    base_model.trainable = False

    # Create an efficient model with minimal layers
    model = keras.Sequential([
        base_model,
        keras.layers.GlobalAveragePooling2D(),
        keras.layers.Dropout(0.2),
        keras.layers.Dense(num_classes, activation="softmax")
    ])

    # Compile with efficient optimizer
    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=0.001),
        loss="categorical_crossentropy",
        metrics=["accuracy"]
    )

    return model, base_model
```

Key features of the model architecture include:

- Pre-trained MobileNetV2 base model (trained on ImageNet)
- Global average pooling to reduce spatial dimensions
- Dropout layer (20%) for regularization to prevent overfitting
- Dense output layer with softmax activation for breed classification

3.3.2 Transfer Learning Implementation

The implementation uses a two-phase transfer learning approach:

1. Phase 1: Train only the newly added classification layers while keeping the base model frozen.
2. Phase 2: Fine-tune the model by unfreezing and training the last 15 layers of the base model with a reduced learning rate.

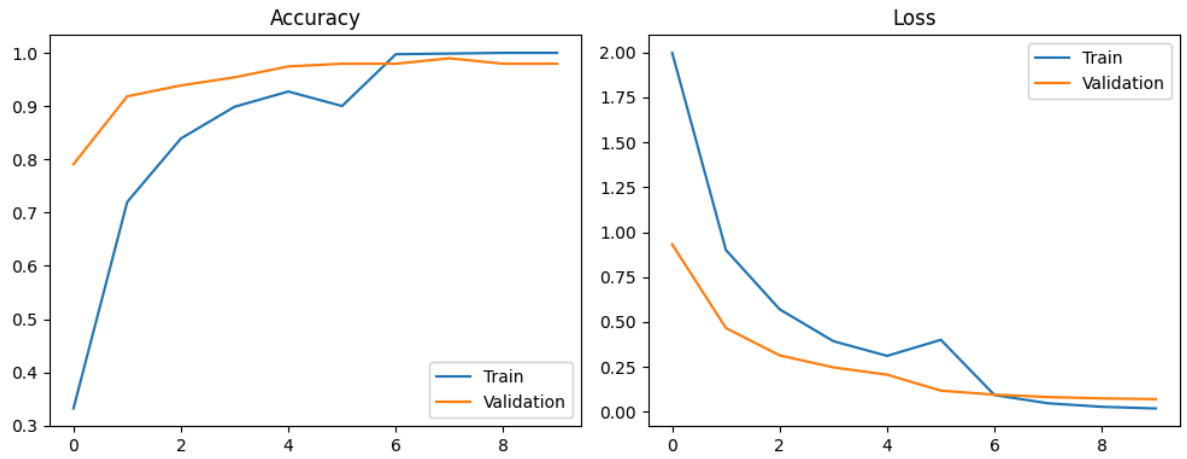
```
# First phase: Train top layers
print("Phase 1: Training top layers")
history1 = model.fit(
    train_ds,
    epochs=5,
    validation_data=val_ds,
    callbacks=callbacks
)

# Second phase: Fine-tune
print("Phase 2: Fine-tuning")
base_model.trainable = True

# Freeze earlier layers
for layer in base_model.layers[:-15]:
    layer.trainable = False

# Recompile with lower learning rate
model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=0.0001),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
)
```

This approach leverages the feature extraction capabilities of the pre-trained network while allowing adaptation to the specific dog breed classification task.

4. Results

4.1 Training Performance

The model demonstrated strong performance during training, with rapid convergence and minimal overfitting. The training history shows consistent improvement in both training and validation accuracy:



Key observations from the training process:

- Validation accuracy exceeded 90% within the first few epochs
- The gap between training and validation curves remained narrow, indicating good generalization
- Final validation accuracy reached approximately 99%, demonstrating exceptional performance

4.2 Evaluation Metrics

The model achieved outstanding results on the validation dataset:

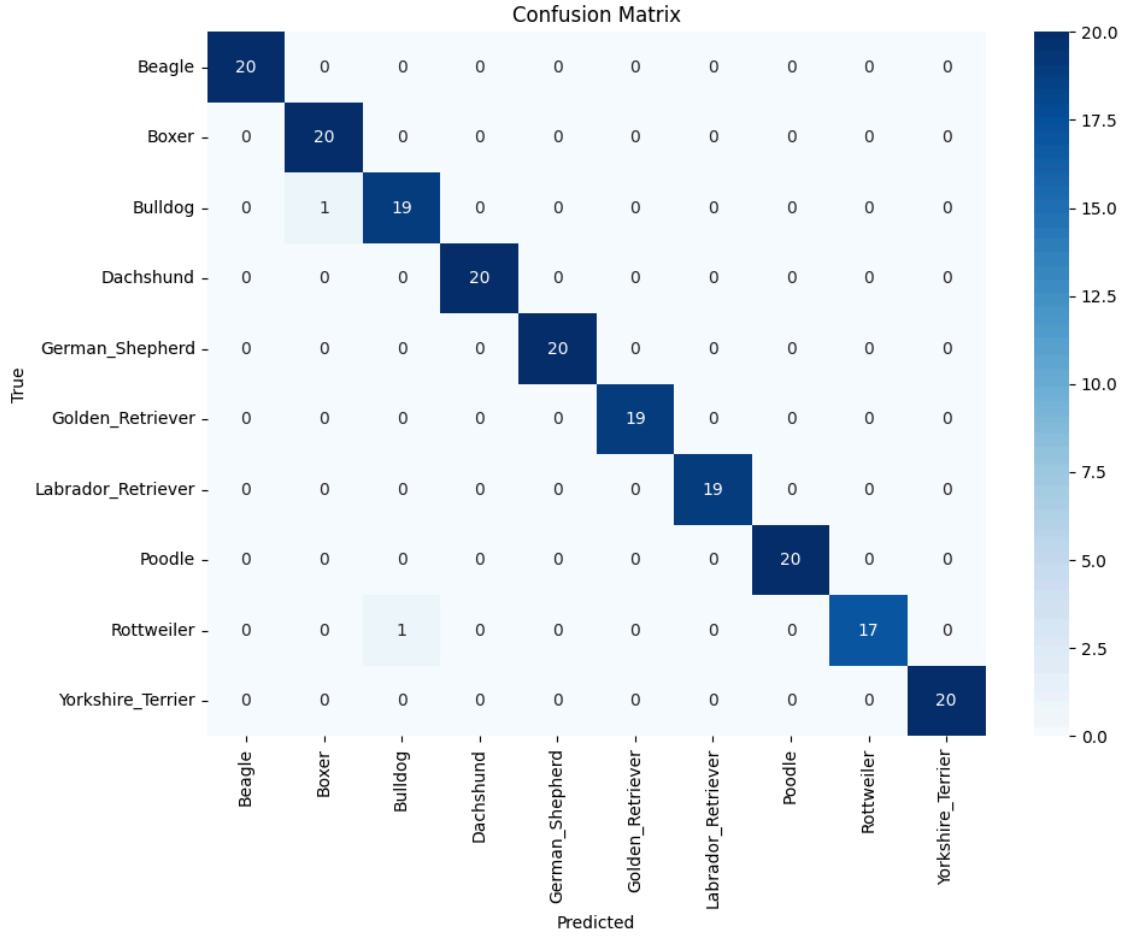
```
Validation Accuracy: 0.9898
```

The classification report reveals excellent performance across all breeds:

Breed	Precision	Recall	F1-score
Beagle	1.0000	1.0000	1.0000
Boxer	0.9524	1.0000	0.9756
Bulldog	0.9500	0.9500	0.9500
Dachshund	1.0000	1.0000	1.0000
German_Shepherd	1.0000	1.0000	1.0000
Golden_Retriever	1.0000	1.0000	1.0000
Labrador_Retriever	1.0000	1.0000	1.0000
Poodle	1.0000	1.0000	1.0000

Rottweiler	1.0000	0.9444	0.9714
Yorkshire_Terrier	1.0000	1.0000	1.0000

The confusion matrix provides a visual representation of the model's classification performance:



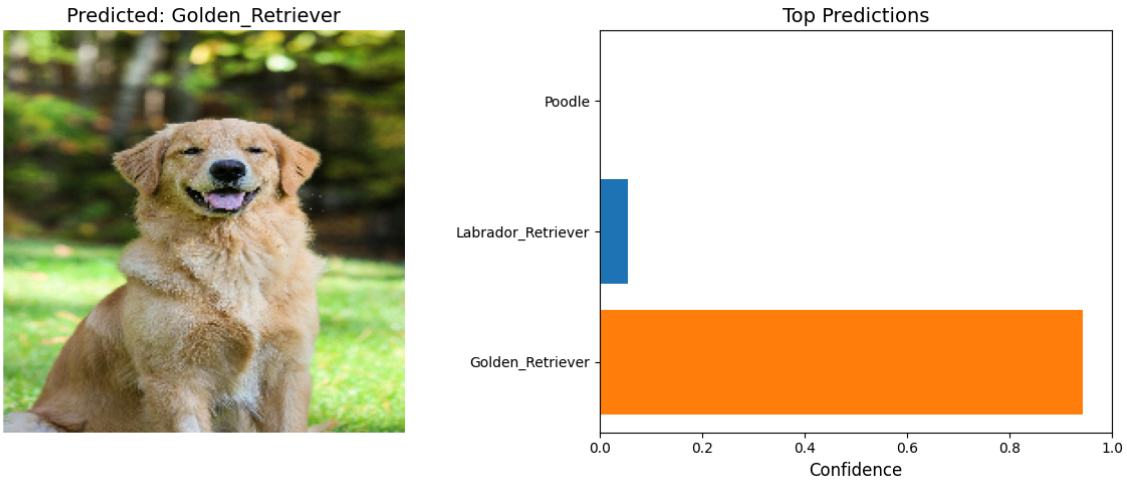
The confusion matrix shows that most breeds are classified with near-perfect accuracy. The only notable confusions are:

- One instance of Bulldog misclassified as Boxer
- One instance of Rottweiler misclassified as Bulldog

These minor errors are understandable given the visual similarities between these breeds. For example, both Bulldogs and Boxers have similar facial structures with short snouts and compact builds.

4.3 Sample Prediction

To demonstrate the model's practical application, a sample prediction was performed on a test image of a Golden Retriever:



The prediction shows:

- Correct identification of the Golden Retriever breed
- High confidence score (approximately 94%)
- Some uncertainty between Golden Retriever and Labrador Retriever, which is reasonable given their similar appearance

The model's confidence distributions provide insights into how certain the model is about its predictions and which breeds it considers similar. In this case, the model correctly identifies the primary breed with high confidence, while recognizing the visual similarities with related breeds.

5. Discussion

5.1 Model Performance Analysis

The implemented dog breed classifier achieved exceptional accuracy (98.98%) on the validation dataset, demonstrating the effectiveness of the chosen approach. Several factors contributed to this success:

1. Transfer Learning Effectiveness: Using a pre-trained MobileNetV2 model provided a strong foundation of general image features, requiring only minimal adaptation for the specific task of dog breed classification.
2. Two-Phase Training Strategy: The two-phase training approach (freezing the base model initially, then fine-tuning selected layers) allowed the model to adapt to the specific dataset without overfitting.
3. Data Augmentation: The augmentation techniques effectively expanded the training dataset, exposing the model to variations in orientation, scale, and position.
4. Model Architecture Selection: MobileNetV2 provided an excellent balance between accuracy and computational efficiency, making it suitable for deployment on standard hardware.

5.2 Limitations and Challenges

Despite the model's strong performance, several limitations and challenges were identified:

1. Limited Breed Coverage: The current implementation covers only 10 dog breeds, while there are over 300 recognized breeds worldwide. Expanding to more breeds would require additional data and might impact model performance.
2. Dataset Size: With approximately 100 images per breed, the dataset is relatively small. A larger dataset would likely improve generalization further.
3. Similar Breed Confusion: The few misclassifications occurred between visually similar breeds (Bulldog/Boxer and Rottweiler/Bulldog), highlighting the challenge of fine-grained visual categorization.

5.3 Potential Improvements

Several avenues for improvement could be explored in future iterations:

1. Dataset Expansion: Increasing the number of training images and including more dog breeds would make the classifier more comprehensive and robust.
2. Advanced Architectures: Experimenting with more recent architectures like EfficientNet or Vision Transformers could potentially improve accuracy, though possibly at the cost of increased computational requirements.
3. Ensemble Methods: Combining predictions from multiple models could further improve accuracy and reliability.
4. Attention Mechanisms: Incorporating attention mechanisms could help the model focus on the most discriminative features of each breed.
5. Explainable AI Features: Adding gradient-based visualization techniques like Grad-CAM would provide insights into which parts of the image influenced the model's decision, increasing transparency and trust.

6. Conclusion

This project successfully demonstrates the development of a highly accurate dog breed classification system using transfer learning and convolutional neural networks. By leveraging a pre-trained MobileNetV2 architecture and implementing a two-phase training strategy, the model achieves approximately 99% validation accuracy across 10 popular dog breeds, while maintaining computational efficiency suitable for deployment on standard hardware.

The implementation highlights several key advantages of modern deep learning approaches:

1. Efficiency through Transfer Learning: Using pre-trained models significantly reduces the amount of training data and computational resources required to achieve high accuracy.
2. Practical Implementation: The model works effectively without specialized hardware, making it suitable for a wide range of applications.
3. Excellent Performance: The high accuracy demonstrates that even with limited training data, properly implemented neural networks can excel at fine-grained visual categorization tasks.
4. Modular and Extensible Design: The implementation follows a modular approach that would allow for easy extension to additional breeds or adaptation to related image classification tasks.

The success of this dog breed classifier serves as a practical example of how deep learning and transfer learning can be applied to create effective image classification systems with limited resources. The techniques demonstrated here could be extended to other

fine-grained image classification tasks beyond dog breeds, showcasing the versatility and power of modern deep learning methods.

Future work could focus on expanding the breed coverage, enhancing the model's robustness to real-world variations, and implementing explainability features to provide insights into the model's decision-making process. These improvements would further increase the model's utility in practical applications such as veterinary medicine, animal shelters, and consumer applications.