Dog Breed Classification Project

Yuyao Wang
yuyaow@bu.edu

May 2020

Data Overview with Example Images

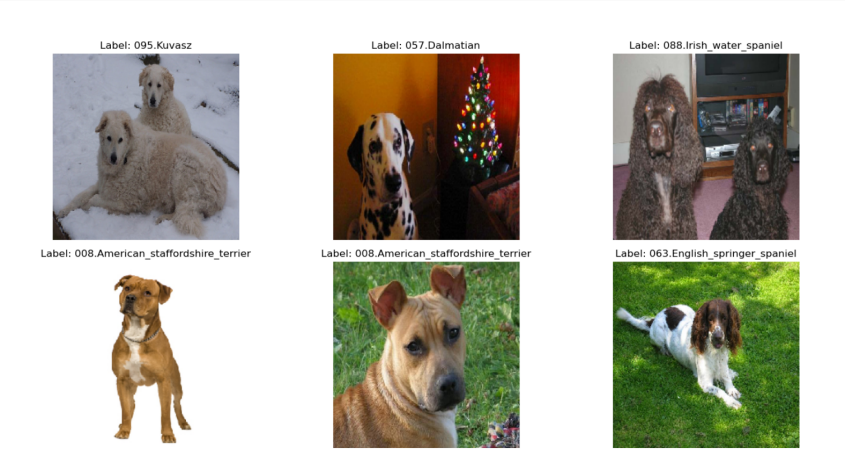


Figure 1: Example Images from the Dog Breed Dataset with Labels

- **Total Dog Categories:** 133 breeds
- **Total Dog Images:** 8351 images
- **Training Set:** 6680 images
- **Validation Set:** 835 images
- **Test Set:** 836 images

Dog Breed Classification

Objective: Develop a model to classify images of dogs into one of 133 distinct dog breeds.

Dataset:

- **Training Set:** 6680 images across 133 breeds.
- **Validation Set:** Used during training for model evaluation.
- **Test Set:** Used to assess final model performance.

Each breed has a designated folder within the train, validation, and test directories, loaded as labeled data.

Approach: Custom CNN

Custom CNN Model:

- Initial approach: A CNN built from scratch for classification.
- Achieved moderate accuracy: Around 17% on the test set.
- Network components: Convolutional layers, max-pooling, batch normalization, fully connected layers, dropout, and L2 regularization.
- Training: RMSprop optimizer and categorical cross-entropy loss.
- Result: Struggled with overfitting, indicating insufficient complexity for 133 breeds.

Approach: Transfer Learning

Transfer Learning with Pre-Trained Models:

- Leveraged pre-trained models such as ResNet50 and VGG16.
- Used pre-trained convolutional layers for feature extraction.
- Transitioned to VGG16 with bottleneck features for better classification.
- Added custom fully connected layers on top of VGG16 bottleneck features.

Evaluation and Performance

Evaluation Metrics:

- Accuracy: Key metric for model performance.
- Validation Loss: Assessed to tune model and prevent overfitting.
- Test Set: Final performance assessment.

Custom CNN: Architecture Overview

- **Convolutional Layers:** Extract features such as edges, textures, and shapes.
- **Max Pooling Layers:** Reduce feature map dimensions and computational cost.
- **Batch Normalization:** Stabilize learning and improve training speed.
- **ReLU Activation:** Add non-linearity for learning complex patterns.
- **Global Average Pooling:** Reduce parameters, minimize overfitting.
- **Dropout:** Randomly drop neurons to prevent overfitting.
- **Fully Connected Layers:** Perform classification with softmax for final output.

Custom CNN: Regularization Techniques

- **L2 Regularization:** Adds a penalty on large weights, preventing overfitting.
- **Dropout:** Randomly deactivates neurons during training for better generalization.

Custom CNN: Training Strategy

- **Optimizer:** RMSprop with learning rate $1e-4$ for complex architectures.
- **Loss Function:** Categorical crossentropy for multi-class classification.
- **Callback:** ModelCheckpoint to save best-performing model based on validation accuracy.
- **Metric:** Accuracy used to track performance during training.

Custom CNN: Layer-by-Layer Overview

- **Convolutional Layers:** Progressive increase in filters (16 to 128) to capture detailed features.
- **Max Pooling:** 3×3 pooling reduces feature map size.
- **Batch Normalization:** Speeds up training and stabilizes the network.
- **ReLU Activation:** Enables non-linear feature learning.
- **Global Average Pooling:** Averaging feature maps for better generalization.
- **Dropout:** Applied before fully connected layers to prevent overfitting.
- **Fully Connected Layers:** Final classification via softmax for dog breed classification.

Model Checkpointing and Saving

- **ModelCheckpoint:** Saves the best model based on validation accuracy.
- **Weights Saved:** The best weights are stored in `weights.best.from_scratch.h5`.
- **Model Retrieval:** Best model can be retrieved for testing or deployment.

Custom CNN: Model Architecture: Layer Overview

Layer (Type)	Output Shape	Param #
conv2d	(None, 224, 224, 16)	448
max_pooling2d	(None, 74, 74, 16)	0
conv2d_1	(None, 74, 74, 32)	4,640
max_pooling2d_1	(None, 24, 24, 32)	0
conv2d_2	(None, 24, 24, 64)	18,496
max_pooling2d_2	(None, 8, 8, 64)	0
conv2d_3	(None, 6, 6, 128)	73,856

Table 1: Convolutional and Pooling Layers

Custom CNN: Batch Normalization, Activation, and Dropout Layers

Layer (Type)	Output Shape	Param #
batch_normalization (BatchNorm)	(None, 6, 6, 128)	512
activation (Activation)	(None, 6, 6, 128)	0
max_pooling2d_3 (MaxPooling2D)	(None, 2, 2, 128)	0
dropout (Dropout)	(None, 2, 2, 128)	0

Table 2: BatchNorm, Activation, MaxPooling, and Dropout Layers

Custom CNN: Global Average Pooling and Dense Layers

Layer (Type)	Output Shape	Param #
global_average_pooling2d	(None, 128)	0
dense	(None, 512)	66,048
dropout_1	(None, 512)	0
dense_1	(None, 133)	68,877

Table 3: Global Pooling and Dense Layers

Custom CNN: Parameter Summary

- **Total Parameters:** 232,229 (907.14 KB)
- **Trainable Parameters:** 231,973 (906.14 KB)
- **Non-Trainable Parameters:** 256 (1.00 KB)

Custom CNN: Training Progress Overview

Epoch 1/15

- Training Loss: 5.13, Training Accuracy: 1.34%
- Validation Loss: 4.90, Validation Accuracy: 2.99%

Epoch 8/15

- Training Loss: 3.61, Training Accuracy: 13.48%
- Validation Loss: 4.22, Validation Accuracy: 7.43%

Epoch 15/15

- Training Loss: 3.04, Training Accuracy: 21.66%
- Validation Loss: 3.50, Validation Accuracy: 17.25%

Custom CNN: Key Training Results

- **Accuracy Improvement**: - Training accuracy increased from **1.34%** to **21.66%**. - Validation accuracy improved from **2.99%** to **17.25%**.
- **Loss Improvement**: - Training loss decreased from **5.13** to **3.04**. - Validation loss improved from **4.90** to **3.50**.
- **Fluctuations in Validation Loss**: - There were fluctuations in validation loss, suggesting potential overfitting.

Custom CNN: Training Loss and Accuracy Analysis

- **Significant Improvement at Epoch 9**: - At Epoch 9, the validation loss improved from **4.22** to **3.77**, leading to an increase in validation accuracy.
- **Slow Accuracy Growth**: - The model exhibited slow and steady accuracy growth, indicating that optimization or regularization adjustments may be necessary.
- **Proposed Adjustments**: - Introduce stronger regularization such as dropout or weight penalties. - Consider fine-tuning the learning rate and adjusting the training schedule for better performance.

Custom CNN: Training and Validation Accuracy

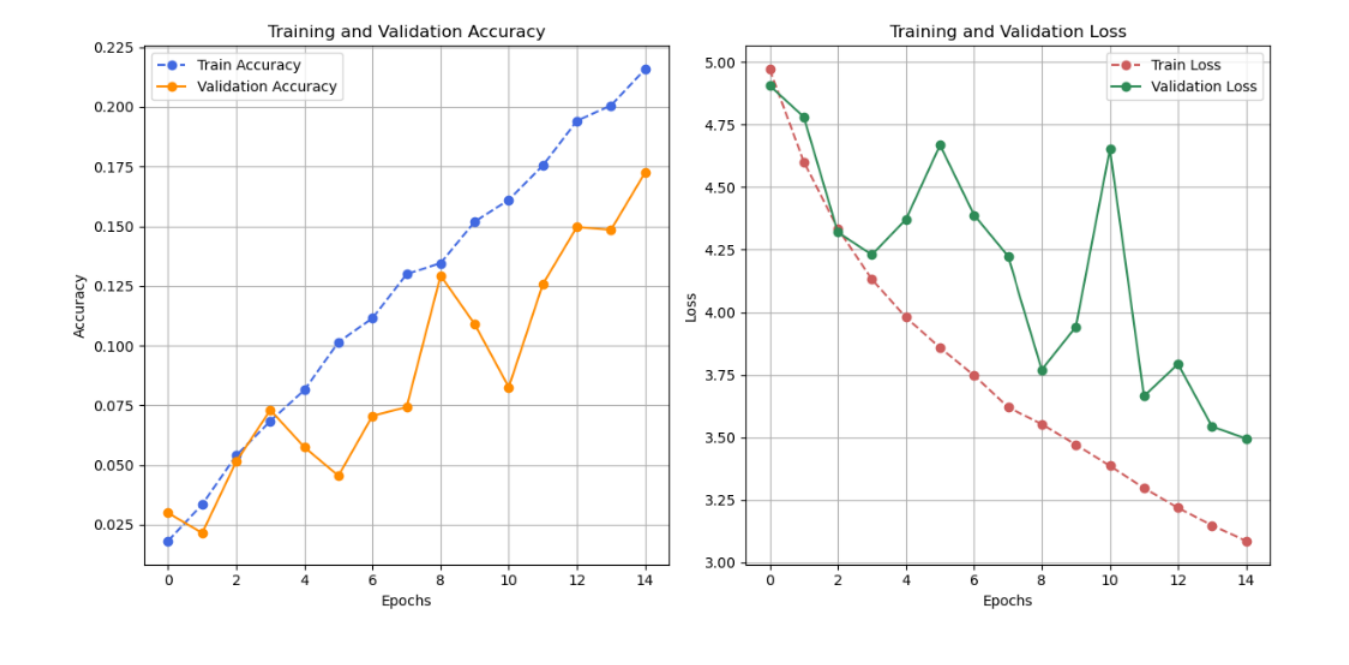


Figure 2: Training and Validation Accuracy and Loss Curves

Summary of Training and Validation Results

- **Training Accuracy**: Steadily improved from 2% to 22% over 15 epochs, indicating consistent learning progress.
- **Validation Accuracy**: Fluctuated throughout the epochs, ending at 18%. This suggests some generalization but with less stability.
- **Training Loss**: Gradual decrease from 5.0 to 3.0, reflecting successful minimization of errors on the training set.
- **Validation Loss**: Showed fluctuations but a downward trend overall, although it remained higher than the training loss, indicating potential overfitting.

Test Accuracy: 16.87%

Model Performance:

- The model achieved a test accuracy of 16.87%.
- This indicates that the model correctly classified approximately 16.9% of the test images.
- While the model demonstrates some learning, this performance is far from an acceptable level, indicating a need for further tuning or model adjustments.

Exploring Transfer Learning with Pre-trained Models

Overview:

- Utilize pre-trained models like ResNet50 or VGG16 for dog breed classification.
- Pre-trained on ImageNet, these models provide learned features, enhancing accuracy and generalization.
- Reduces training time significantly.

VGG vs Traditional CNNs: Key Differences

Key Differences:

- **Depth of Layers:** Traditional CNNs usually have 2-5 convolutional layers, whereas VGG significantly increases depth with 16-19 layers.
- **Convolution Kernel Size:** VGG uses fixed 3x3 convolution kernels, while earlier CNNs often used larger kernels like 5x5 or 7x7.
- **Consistent Architecture:** VGG adopts a regular design with uniform kernel sizes and strides, making it more optimized compared to the varied architectures of earlier CNNs.
- **Use of Fully Connected Layers:** Similar to traditional CNNs, VGG uses fully connected layers for classification, but with richer features extracted from deeper layers.

Methodology

Steps:

1. **Pre-trained Model:** Use VGG16 without top layers.
2. **Freeze Layers:** Fix convolutional layers as feature extractors.
3. **Add Custom Layers:** Tailor layers for dog breed classification (133 classes).
4. **Fine-tuning:** Train custom layers first, then fine-tune the entire network.

Generating Bottleneck Features and DogVGG16Data.npz

Key Concept:

- Pre-process images from the training, validation, and test sets, and pass them through a pre-trained VGG16 model (without top layers).
- The model extracts high-level features, known as bottleneck features, which represent useful patterns like edges and textures.
- These features are stored in a ‘.npz‘ file, allowing efficient storage and compression of multiple arrays.
- This process significantly reduces training time while maintaining strong performance, simplifying the training of custom classifiers.

Model Summary: "sequential_1"

Model Layers:

Layer (type)	Output Shape	Param #
GlobalAveragePooling2D	(None, 512)	0
Dense	(None, 133)	68,229

Total Parameters: 68,229 (266.52 KB)

Trainable Parameters: 68,229 (266.52 KB)

Non-trainable Parameters: 0 (0.00 B)

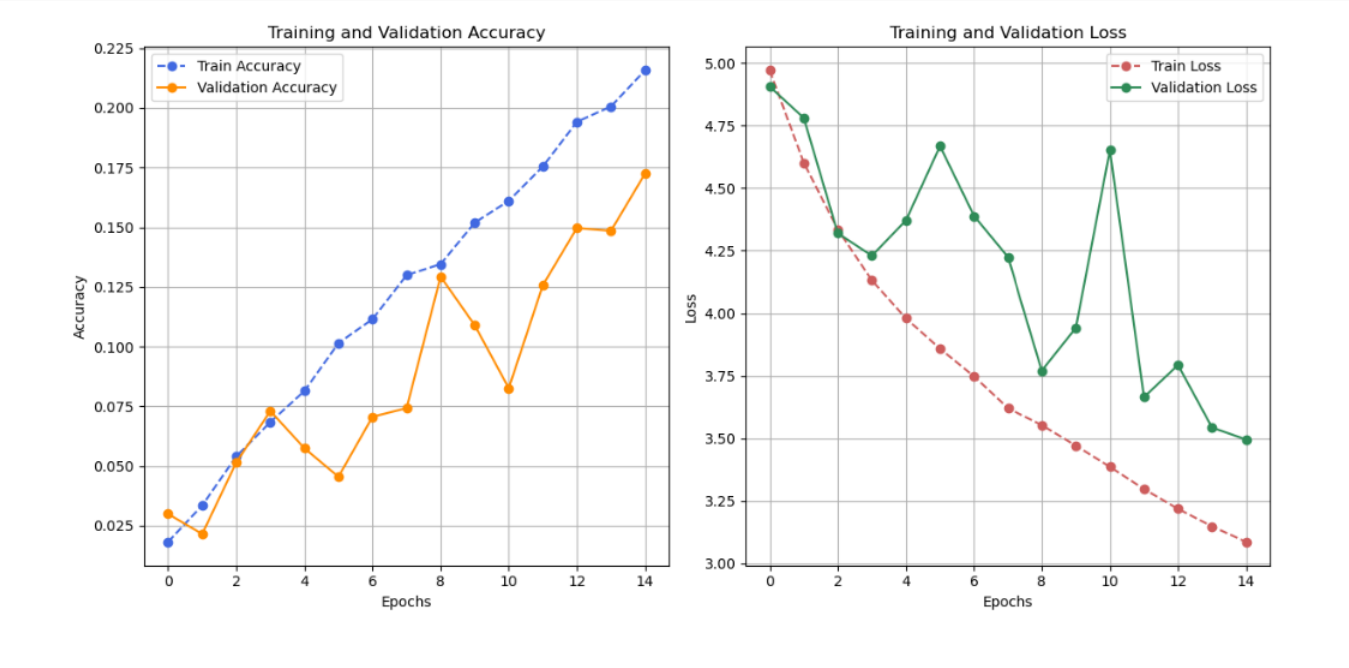
VGG16 Training Results - Summary

Key Observations:

- **Training Accuracy:** Increased from 10.86% to 99.51%, indicating strong performance on the training set.
- **Validation Accuracy:** Improved from 43.11% to 74.73%, showing steady generalization but room for improvement.
- **Overfitting:** Training accuracy is much higher than validation accuracy, suggesting possible overfitting.
- **Validation Loss:** Despite accuracy improvements, validation loss fluctuates, indicating challenges in stabilizing the model across different classes.

Training and Validation Trends

- **Epoch 1:** Training accuracy: 10.86%, Validation accuracy: 43.11%
- **Epoch 10:** Training accuracy: 96.78%, Validation accuracy: 72.33%
- **Epoch 20:** Training accuracy: 99.51%, Validation accuracy: 74.73%



VGG Results Analysis

Key Observations:

- **Training Accuracy:** Rapid increase, reaching nearly 100%.
- **Validation Accuracy:** Plateaued around 70%, indicating potential overfitting.
- **Training Loss:** Decreases sharply and stabilizes close to 0, showing strong fitting on training data.
- **Validation Loss:** Stabilizes at a higher value, indicating weaker generalization to unseen data.

Test Accuracy: 75.48%

Model Performance:

- The model achieved a test accuracy of 75.48%.
- This indicates that the model correctly identified the breed of dogs in the test set approximately 75% of the time.
- The result shows that the model has fairly good generalization ability on unseen data after training.

Predicted Breed: Chihuahua

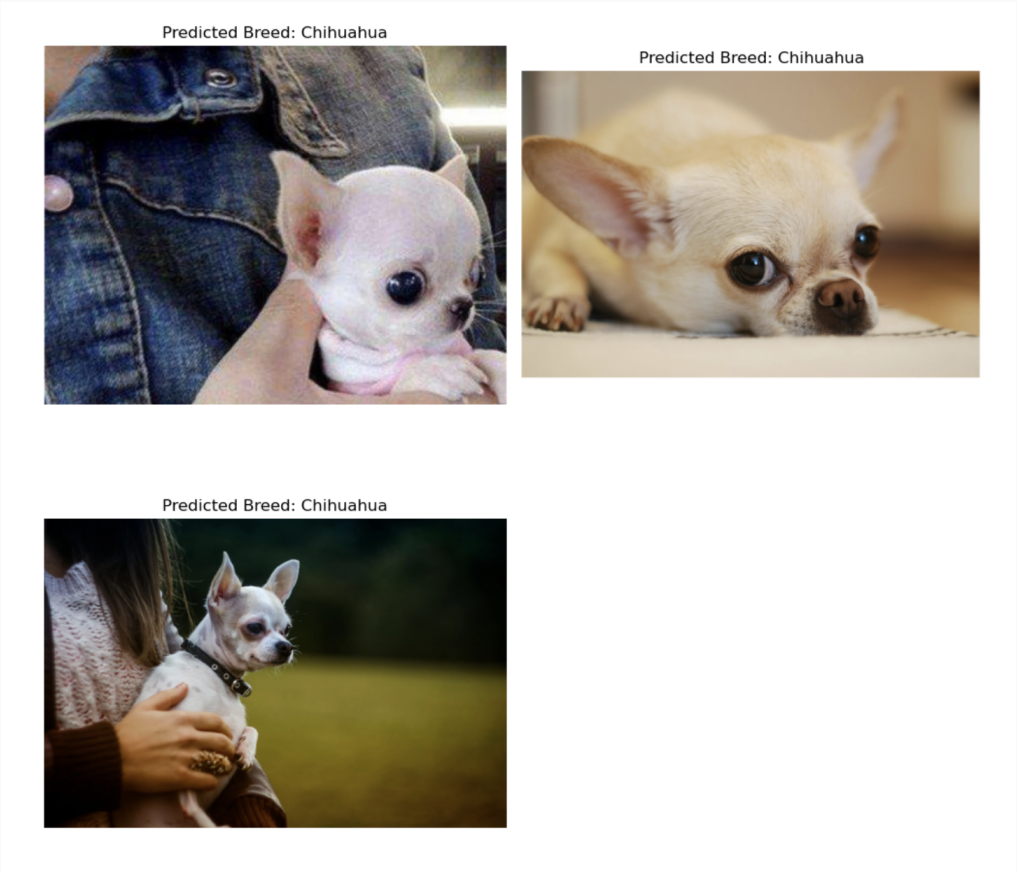


Figure 4: Predicted Breed: Chihuahua

Predicted Breeds: Great Pyrenees & Bernese Mountain Dog

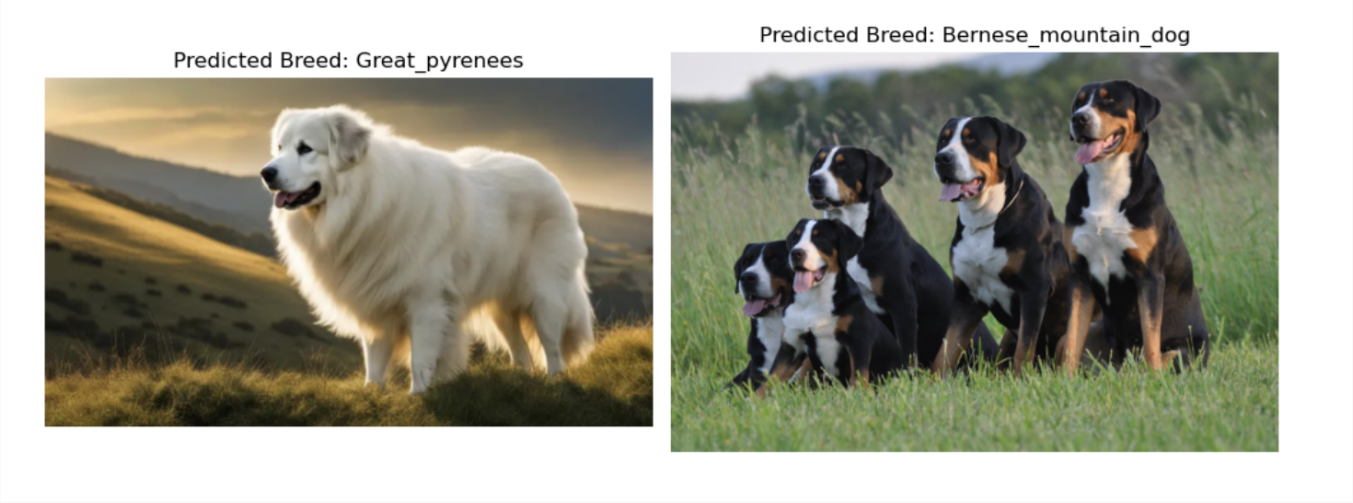


Figure 5: Predicted Breeds: Great Pyrenees and Bernese Mountain Dog

Predicted Dog Breeds Analysis

Summary of Results:

- The model successfully predicted diverse dog breeds, demonstrating robustness across various breed types, though further refinement is needed for improved accuracy.