Fine-Grained Bird Classification on CUB-200-2011

Project Report

1. Task Overview

The objective of this project is to perform fine-grained image classification on the Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset. The task involves training a Convolutional Neural Network (CNN) to accurately distinguish between 200 different species of birds, a challenging problem due to the subtle inter-class variations and high intra-class variations (e.g., differences in pose, lighting, and background). The model's parameter count was constrained to a maximum of 10 million.

2. Submission Links

- **Link to GitHub Repository:**

  https://github.com/OGCoderOggy/SOC_25-Intro_to_Deep_Learning.git

- **Link to Final Model Checkpoint (fine_tuned_best_model.h5):**

  https://drive.google.com/file/d/15HlUS6Te9x_wQD2qTlxIq9a_9oiDQWtp/view?usp=sharing

3. Model Architecture

To achieve high accuracy while staying within the parameter budget, a **Transfer Learning** approach was adopted using a pre-trained **EfficientNetV2B0** model.

- **Base Model:** EfficientNetV2B0, pre-trained on the ImageNet dataset. The top classification layer of the original model was removed. EfficientNet is a highly parameter-efficient architecture, making it ideal for this task.
- **Custom Classification Head:** A new head was added on top of the frozen base model layers:
  1. GlobalAveragePooling2D: To flatten the feature maps from the base model.
  2. BatchNormalization: To stabilize training and improve generalization.
  3. Dropout (0.5): A strong regularization technique to prevent overfitting.
  4. Dense (200, activation='softmax'): The final output layer, with one neuron for each of the 200 bird species.
- **Total Parameters:** The resulting model has approximately **7.1 Million** parameters, which is well below the 10M limit.

4. Training Details

The model was trained using a two-phase fine-tuning strategy to maximize performance.

- **Dataset:** CUB-200-2011 with the official train/test split.
    - Training Images: 5,994
    - Testing Images: 5,794
    - Number of Classes: 200
- **Preprocessing & Augmentation:**
    - All images were resized to (224, 224).
    - Labels were one-hot encoded to be compatible with the chosen loss function.
    - On-the-fly data augmentation was applied during training to create a more robust model:
        - RandomHorizontalFlip
        - RandomRotation (10%)
        - RandomZoom (20%)
        - RandomContrast (20%)
- **Training Strategy:**
    - **Phase 1: Feature Extraction (10 Epochs)**
        - The EfficientNetV2B0 base model was **frozen**.
        - Only the custom classification head was trained.
        - **Optimizer:** Adam with a learning rate of 1e-3.
        - This phase allows the new layers to adapt to the features extracted by the powerful base model without corrupting the pre-trained weights.
    - **Phase 2: Fine-Tuning (15 Epochs)**
        - The entire model, including the EfficientNetV2B0 base, was **unfrozen**.
        - Training continued for an additional 15 epochs.
        - **Optimizer:** Adam with a very low learning rate of 1e-5.
        - This allows the entire network to make small, nuanced adjustments to better fit the specific features of the bird dataset.
- **Hyperparameters:**
    - **Batch Size:** 16

    ◦ **Loss Function:** CategoricalCrossentropy with **Label Smoothing (0.1)**. Label smoothing is a regularization technique that prevents the model from becoming overconfident and improves generalization.

5. Results and Performance

- **Final Evaluation Metrics:**
    ◦ **Final Test Loss:** 2.0271
    ◦ **Final Test Accuracy:** 0.7123 (71.23%)
- **Training & Validation Curves:** The following graphs illustrate the model's accuracy and loss over the entire training process, including both the feature extraction and fine-tuning phases. The dotted vertical line indicates the switch from Phase 1 to Phase 2.

6. Conclusion

The use of transfer learning with a pre-trained EfficientNetV2B0 model proved highly effective for the fine-grained bird classification task. The two-phase training strategy, starting with a frozen base and then fine-tuning the entire network with a low learning rate, allowed the model to achieve high accuracy while respecting the parameter constraints. Regularization techniques like data augmentation, dropout, and label smoothing were crucial in preventing overfitting and improving the model's ability to generalize to the test set. The final model achieved a test accuracy of 71.23%.