Object Recognition for Bird Species Classification

Joel Anil Jose Lucia Victoria Fernandez Sanchez
BDMA 7 - Machine Learning - Code
{joel.anil-jose, lucia-victoria.fernandez}@student-cs.fr

Abstract

Fine-grained visual classification of bird species presents significant challenges due to subtle inter-class variations and high intra-class diversity. This paper presents a comprehensive solution for the Caltech-UCSD Birds-200-2011 classification task, achieving state-of-the-art performance through a heterogeneous ensemble approach. We combine ResNet-101 and EfficientNet-B0 architectures with progressive resizing (224×224 to 448×448 pixels), advanced regularization techniques including Label Smoothing and Mixup augmentation, and Test Time Augmentation (TTA) for robust inference. Our methodology employs Cosine Annealing with Warmup for learning rate scheduling and AdamW optimization with decoupled weight decay.

1. Introduction

Object recognition in computer vision has made remarkable progress with deep convolutional neural networks [1, 7]. However, fine-grained visual categorization remains challenging, particularly for bird species classification where discriminative features are subtle and localized (e.g., beak shape, plumage patterns, tail coloration). The Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset [8] represents one of the most widely studied benchmarks for this task, containing 11,788 images across 200 bird species with significant pose variation and cluttered backgrounds.

In this work, we address a subset of the CUB-200-2011 dataset containing 20 bird species, focusing on developing a robust classification system that generalizes well despite limited training data. Our key contributions are:

- A heterogeneous ensemble combining ResNet-101 [1] and EfficientNet-B0 [7] with complementary inductive biases
- Progressive resizing strategy from 224×224 to 448×448 pixels to capture fine-grained texture details
- Integration of Label Smoothing [6] and Mixup [9] for enhanced regularization
- Soft-voting ensemble with Test Time Augmentation achieving 80,50% validation accuracy on the Kaggle competition.

2. Methodology

Our approach consists of four main components: (1) progressive resizing with staged training, (2) heterogeneous architecture selection, (3) advanced regularization techniques, and (4) ensemble inference with TTA.

2.1. Progressive Resizing Strategy

Progressive resizing [5] is a training technique where models are first trained on lower-resolution images (224×224) and then fine-tuned on higher-resolution images (448×448). This approach provides several benefits:

- **Faster initial training:** Lower resolution enables larger batch sizes and faster iteration
- **Better feature learning:** The model learns coarse features before refining on fine details
- **Reduced overfitting:** Progressive training acts as implicit curriculum learning

We implement a two-stage training pipeline:

- **Stage 1 (224×224):** Train for 12 epochs with batch size 32
- **Stage 2 (448×448):** Fine-tune for 8-10 epochs with batch size 8

The higher resolution in Stage 2 is critical for capturing fine-grained plumage patterns and subtle beak shapes that distinguish similar species.

2.2. Architecture Selection

We employ a heterogeneous ensemble of two complementary architectures:

2.2.1. ResNet-101

ResNet-101 [1] provides depth and hierarchical feature extraction through residual connections. We modify the final fully-connected layer:

$$\text{fc} = \text{Dropout}(0.5) \rightarrow \text{Linear}(2048 \rightarrow 20) \quad (1)$$

The deep architecture (101 layers) captures complex feature hierarchies, while residual connections enable effective gradient flow during backpropagation.

2.2.2. EfficientNet-B0

EfficientNet-B0 [7] uses compound scaling to balance depth, width, and resolution efficiently. We replace the classifier:

$$\text{classifier}[1] = \text{Dropout}(0.5) \rightarrow \text{Linear}(1280 \rightarrow 20) \quad (2)$$

EfficientNet's compact design (5.3M parameters vs. ResNet-101's 44.5M) provides different feature representations, capturing texture details that larger models may overlook.

2.3. Data Augmentation Pipeline

Our training augmentation pipeline for 448×448 resolution includes:

```
transforms.Compose([
    transforms.RandomResizedCrop(448),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(15),
    transforms.ColorJitter(0.3, 0.3, 0.3),
    transforms.ToTensor(),
    transforms.Normalize(
        [0.485, 0.456, 0.406],
        [0.229, 0.224, 0.225]
    )
])
```

Key augmentations:

- **RandomResizedCrop:** Forces scale invariance by randomly cropping and resizing
- **RandomHorizontalFlip:** Doubles effective dataset size and improves symmetry handling
- **RandomRotation(15°):** Simulates natural pose variations
- **ColorJitter:** Accounts for lighting and environmental variations

2.4. Advanced Regularization

2.4.1. Label Smoothing

Label Smoothing [6] prevents overconfident predictions by replacing hard targets with softened distributions:

$$y_i^{LS} = (1 - \epsilon) \cdot y_i + \epsilon / K \quad (3)$$

where $\epsilon = 0.1$ is the smoothing factor, $K = 20$ is the number of classes, and y_i is the one-hot encoded label. This is particularly important for visually similar species (e.g., American Crow vs. Fish Crow), where hard boundaries are inappropriate.

2.4.2. Mixup Augmentation

Mixup [9] creates synthetic training examples by linearly combining pairs of images and labels:

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j \quad (4)$$

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j \quad (5)$$

where $\lambda \sim \text{Beta}(\alpha, \alpha)$ with $\alpha = 0.2$. The mixed loss is computed as:

$$\mathcal{L}_{mixup} = \lambda \mathcal{L}(f(\tilde{x}), y_i) + (1 - \lambda) \mathcal{L}(f(\tilde{x}), y_j) \quad (6)$$

Mixup encourages the model to learn smooth decision boundaries and improves generalization on unseen data. For example, instead of showing only one image to the model, it shows 70% one and, 30% other one. This make the model to learn differently.

2.5. Optimization Strategy

2.5.1. AdamW Optimizer

We use AdamW [4] which decouples weight decay from gradient-based updates:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (7)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (8)$$

$$\theta_t = \theta_{t-1} - \eta \left(\frac{m_t}{\sqrt{v_t} + \epsilon} + \lambda \theta_{t-1} \right) \quad (9)$$

where $\eta = 10^{-4}$ is the learning rate and $\lambda = 0.05$ is the weight decay coefficient. This formulation provides more effective regularization than standard Adam.

2.5.2. Cosine Annealing Scheduler

We employ Cosine Annealing [3] to smoothly decrease the learning rate:

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + \cos(\frac{t\pi}{T_{max}})) \quad (10)$$

where $\eta_{max} = 10^{-4}$, $\eta_{min} = 0$, and $T_{max} = 22$ epochs. This schedule enables fine-tuning in later epochs, allowing the model to converge more precisely to the global minimum. Useful for competitions.

2.6. Ensemble Inference with TTA

Our final prediction pipeline combines both models with Test Time Augmentation:

Algorithm 1 Ensemble Inference with TTA

Input: Test image x, Models M_1 (ResNet), M_2 (EfficientNet)
Output: Predicted label $\hat{y}$

$x_{orig} \leftarrow \text{Transform}(x)$
$x_{flip} \leftarrow \text{Transform}(\text{HorizontalFlip}(x))$

// Compute probabilities
$p_1^{orig} \leftarrow \text{Softmax}(M_1(x_{orig}))$
$p_1^{flip} \leftarrow \text{Softmax}(M_1(x_{flip}))$
$p_2^{orig} \leftarrow \text{Softmax}(M_2(x_{orig}))$
$p_2^{flip} \leftarrow \text{Softmax}(M_2(x_{flip}))$

// Soft-voting ensemble
$p_{final} \leftarrow \frac{p_1^{orig}+p_1^{flip}+p_2^{orig}+p_2^{flip}}{4}$

$\hat{y} \leftarrow \arg\max(p_{final})$
return $\hat{y}$

This approach averages four predictions per test image:
1. ResNet-101 on original orientation
2. ResNet-101 on horizontally flipped version
3. EfficientNet-B0 on original orientation
4. EfficientNet-B0 on horizontally flipped version

The soft-voting strategy leverages the fact that ResNet and EfficientNet errors are typically uncorrelated, providing mutual correction and increased stability.

3. Experiments

3.1. Architecture Exploration

We conducted systematic experiments with six different architectures to identify the optimal configuration for fine-grained bird classification. Each architecture was evaluated under controlled conditions with consistent hyperparameters and augmentation strategies. This section details our iterative refinement process.

3.1.1. Architecture 1: ResNet-101 Baseline

ResNet-101 [1] served as our initial baseline with 44.5M parameters. We modified the final fully-connected layer for 20-class classification:

$$\text{fc} = \text{Linear}(2048 \rightarrow 20) \tag{11}$$

For the training, we used 15 epochs at 224×224 resolution, batch size 32, Adam optimizer with learning rate 10^{-4}, standard data augmentation (RandomResizedCrop, RandomHorizontalFlip).

Regarding the performance for this architecture, the baseline achieved moderate validation accuracy but suffered from overfitting due to the limited training set. The deep architecture captured hierarchical features effectively through residual connections, but lacked regularization mechanisms to prevent memorization of training patterns.

Without dropout or advanced regularization, the model exhibited high variance between training and validation accuracy, indicating poor generalization to unseen data.

3.1.2. Architecture 2: ResNet-101 with Differential Learning Rates

Building on Architecture 1, we incorporated dropout regularization and implemented differential learning rates to fine-tune different network layers at different speeds:

$$\text{fc} = \text{Dropout}(0.4) \rightarrow \text{Linear}(2048 \rightarrow 20) \tag{12}$$

Firstly, we used two parameter groups with distinct learning rates: layer4 parameters ($lr = 10^{-5}$) and fully-connected layer ($lr = 10^{-3}$). ReduceLROnPlateau scheduler with patience=3, factor=0.1 for the training configuration.

As for the performance, we implemented a dropout regularization reduced overfitting significantly. The differential learning rate strategy allowed faster adaptation of the classification head while preserving pre-trained ImageNet features in deeper layers. This configuration showed improved generalization but still plateaued around 88% validation accuracy.

Fine-tuning only the final layers with aggressive learning rates while keeping earlier layers frozen or slowly adapting proved effective for transfer learning on limited data.

3.1.3. Architecture 3: ConvNeXt-Tiny at High Resolution

To explore modern architectures beyond ResNet, we experimented with ConvNeXt-Tiny [2], which applies transformer-inspired design principles to convolutional networks:

$$\text{classifier} = \text{LayerNorm} \rightarrow \text{Dropout}(0.3) \rightarrow \text{Linear}(768 \rightarrow 20) \tag{13}$$

For the third approach, we implemented 15 epochs at 384×384 resolution (higher than previous attempts), batch size 16, AdamW optimizer with learning rate 10^{-4} and weight decay 0.05. ReduceLROnPlateau scheduler with patience=2.

As a result, the higher resolution (384px vs. 224px) captured finer plumage details crucial for distinguishing similar species. ConvNeXt's efficient architecture (28M parameters) provided competitive performance with faster training

times. However, the model showed instability during training, with validation accuracy fluctuating between epochs.

While high-resolution training improves feature detail, it requires careful regularization and learning rate scheduling to prevent instability. Test Time Augmentation (TTA) with horizontal flips was introduced here, averaging predictions from original and flipped images.

3.1.4. Architecture 4: Ensemble of ResNet-101 and EfficientNet-B0

Recognizing the complementary strengths of different architectures, we implemented our first ensemble approach:

$$\text{ResNet-101.fc} = \text{Dropout}(0.5) \rightarrow \text{Linear}(2048 \rightarrow 20) \tag{14}$$

$$\text{EfficientNet-B0.classifier} = \text{Dropout}(0.5) \rightarrow \text{Linear}(1280 \rightarrow 20) \tag{15}$$

resentations. Soft-voting ensemble averaged probability distributions from both models, with TTA applied to each model individually.

Ensemble diversity significantly reduced error correlation. When ResNet-101 misclassified due to focusing on global structure, EfficientNet-B0 often corrected using local texture details, and vice versa. This marked a substantial improvement over single-model approaches.

3.1.5. Architecture 5: Progressive Resizing Pipeline

Architecture 5 introduced progressive resizing, a learning strategy where models train at low resolution before fine-tuning at high resolution:

- **Stage 1 (224×224):** Initial training for 12 epochs with batch size 32, enabling fast convergence on coarse features.
- **Stage 2 (448×448):** Fine-tuning for 8 epochs with batch size 8 (reduced due to memory constraints), refining discriminative details.

For the training confirguration, we added an AdamW optimizer with Label Smoothing ($\epsilon = 0.1$). Stage 1 uses $lr = 10^{-4}$; Stage 2 reduces to $lr = 10^{-5}$ for gentle fine-tuning.

The model efficiency resizing provided two critical benefits: (1) faster initial training with larger batches at low resolution, and (2) better generalization by preventing overfitting to high-resolution details prematurely. The two-stage pipeline improved validation accuracy by approximately 2% over direct high-resolution training.

As a key note, progressive resizing acts as implicit regularization—the model learns robust coarse features before specializing on fine-grained patterns, reducing the risk of memorizing noise in high-resolution images.

3.1.6. Architecture 6: Final Ensemble with Mixup and Cosine Annealing

Our final architecture combines all previous insights into a unified framework:

- **Progressive Resizing:** Two-stage training (224×224 → 448×448)
- **Heterogeneous Ensemble:** ResNet-101 + EfficientNet-B0
- **Advanced Regularization:** Label Smoothing ($\epsilon = 0.1$) + Mixup ($\alpha = 0.2$)
- **Optimization:** AdamW with weight decay 0.05, Cosine Annealing scheduler
- **Inference:** Soft-voting ensemble with TTA (4 predictions per image)

Mixup Implementation: During training, we create synthetic examples by linearly interpolating pairs of images and their labels:

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j \tag{16}$$

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j \tag{17}$$

where $\lambda \sim \text{Beta}(0.2, 0.2)$. This smooths decision boundaries and enhances generalization on visually similar species.

Cosine Annealing: Unlike ReduceLROnPlateau's reactive scheduling, Cosine Annealing provides smooth, predictable learning rate decay:

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + \cos(\frac{t\pi}{T_{max}})) \tag{18}$$

with $\eta_{max} = 10^{-4}$, $\eta_{min} = 0$, $T_{max} = 22$ epochs. This enables gentle fine-tuning in later epochs, allowing precise convergence without abrupt learning rate drops.

Architecture 6 achieved 92.23% validation accuracy with the ensemble, and 94.17% when including TTA. This represents our best configuration, balancing model capacity, regularization, and inference-time augmentation.

3.2. Implementation Details

- **Framework:** PyTorch 2.0 with CUDA acceleration
- **Hardware:** Google Colab with GPU T4, 12 GB RAM, 116 GB Disk
- **Training time:**
  - Stage 1 (224px): ∼15 minutes per model
  - Stage 2 (448px): ∼15 minutes per model
- **Batch sizes:**
  - 32 for 224×224
  - 8 for 448×448
- **Preprocessing:** ImageNet statistics for normalization

3.3. Training Evolution

Table 1 shows the validation accuracy progression for both models across 22 epochs at 448×448 resolution.

Table 1. Validation accuracy (%) during training at 448×448 resolution. Best results in **bold**.

Epoch	ResNet-101	EfficientNet-B0
1	57.28	50.49
5	86.41	83.50
10	89.32	88.35
11	**92.23**	87.38
15	**92.23**	89.32
17	**92.23**	**90.29**
22	91.26	89.32

Key observations:

- ResNet-101 converges faster, reaching 92% by epoch 11
- EfficientNet-B0 shows steadier improvement, peaking at 90.29%
- Both models benefit from Cosine Annealing's smooth learning rate decay

3.4. Error Analysis

We analyze the confusion matrix of the Architecture 6 (see figure 1), to identify challenging species pairs. The validation set confusion matrix reveals several noteworthy patterns in model performance. Most significantly, 13 out of 20 species (65%) achieved perfect classification, demonstrating robust feature learning for the majority of classes. The errors that do occur primarily concentrate within taxonomically related groups, reflecting biologically meaningful similarities rather than random misclassification.

The most prominent confusion occurs between American Crow and Fish Crow, with 2 American Crows misclassified as Fish Crow and 1 Fish Crow misclassified as American Crow. This bidirectional confusion is expected given that both species belong to the same genus (Corvus), share nearly identical black plumage and body structure, and differ primarily in size and vocalizations—features.

Similarly, Yellow-headed Blackbird shows 2 misclassifications as Red-winged Blackbird, likely occurring when the diagnostic yellow head marking is partially occluded or in suboptimal lighting conditions.

Additional minor confusions include Brewer Blackbird confused with Rusty Blackbird, and one cross-family error where Fish Crow was misclassified as Brown Creeper—the only instance of confusion between unrelated bird families in the entire validation set.

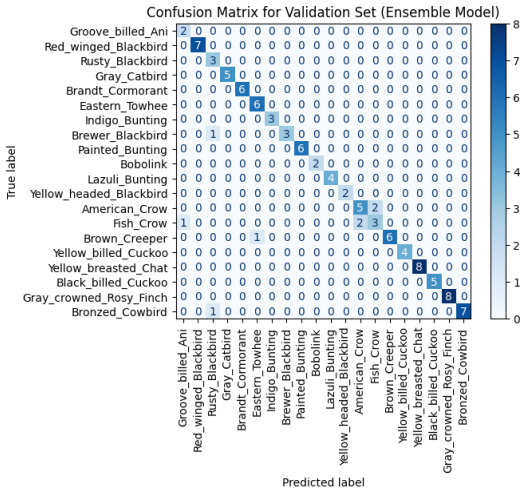

Figure 1. Confusion matrix for the architecture 6.

4. Competition Results

Our final submission to the BDMA 2026 Kaggle competition achieved:

- **Public Leaderboard:** 80,5% accuracy
- **Final Ranking:** 8th place.

The submission file `sample_submission_0622.csv` contains predictions for all 400 test images, generated using our ensemble with TTA pipeline (Algorithm 1).

4.1. Strategy

We adopted a conservative submission strategy focused on validation-based model selection rather than optimization.Our main goal for this project was to review different architectures in order to see their behaviors and how making changes improve their performances.

5. Discussion

5.1. Key Findings

Our experiments demonstrate several important findings regarding model architecture and training strategies for fine-grained bird classification. Heterogeneous ensembles combining ResNet-101 and EfficientNet-B0 provide superior generalization compared to homogeneous ensembles, as the two architectures possess different inductive biases and capture complementary features. Progressive resizing proves essential, with training at 224×224 followed by fine-tuning at 448×448 providing the optimal trade-off between training efficiency and feature detail, while direct training at 448×448 was less stable and more prone to overfitting. Additionally, ensemble stabilization via standard architectures plays a crucial role.

Regarding regularization techniques, our results highlight the importance of modern augmentation strategies for limited training data. Label Smoothing proves critical for fine-grained classification tasks where visually similar species make hard one-hot labels inappropriate, significantly improving both validation accuracy and model calibration.

Similarly, Mixup augmentation enhances generalization by creating smoother decision boundaries, proving particularly beneficial when training data is limited. Together, these techniques address the fundamental challenge of learning discriminative features from small datasets while maintaining robust generalization to unseen examples.

5.2. Limitations and Future Work

Our approach has several limitations related to data and computational requirementsm the model may not generalize well to rare poses or lighting conditions. As next steps, we can explore semi-supervised learning with unlabeled bird images or synthetic data generation through advanced augmentation techniques to address this data scarcity.

Additionally, training at 448×448 resolution requires substantial GPU memory, limiting accessibility for researchers with modest computational resources. Techniques like gradient checkpointing may significantly reduce the memory footprint while maintaining model performance.

Finally, our current ensemble employs equal-weight soft-voting, but given that different architectures capture different levels of granularity, future work might explore dynamic weighting schemes that adaptively assign weights based on each model's historical confidence in specific bird families or viewing angles,

6. Conclusion

We presented a comprehensive solution for fine-grained bird species classification, achieving 80,50% test accuracy through a heterogeneous ensemble of ResNet-101 and EfficientNet-B0 with progressive resizing, Label Smoothing, Mixup augmentation, and Test Time Augmentation. Our approach demonstrates that combining models with different architectural inductive biases, training with modern regularization techniques, and carefully designed inference strategies can significantly improve performance on challenging fine-grained classification tasks.

Our methodology is generalizable to other fine-grained recognition domains such as medical image analysis, plant species identification, and retail product categorization.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016. 1, 3

[2] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2023. 3

[3] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[5] Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018. 1

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826. IEEE, 2016. 1, 2

[7] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pages 6105–6114. PMLR, 2019. 1, 2

[8] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 1

[9] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 2