Bird Species Recognition Using Fine-Tuned Pretrained CNNs

Xiang Zheng
School of Data Science, Fudan University
`21307110169@m.fudan.edu.cn`

Abstract

Bird species recognition is crucial for biodiversity monitoring and conservation efforts. In this study, we explore the effectiveness of fine-tuning pretrained convolutional neural networks (CNNs) for this task. Using the CUB-200-2011 dataset, we fine-tune a ResNet18 model and conduct experiments to determine optimal hyperparameters and training strategies. Our results demonstrate the efficacy of transfer learning in achieving good accuracy rates, highlighting the potential of CNNs in bird species identification.

Note: You can download source code from the repo and follow the readme to download data and `.pth` files.

1 Introduction

Recognizing bird species from images is a challenging task because birds can look very different depending on their appearance, pose, and the environment. Accurate bird identification is important for biodiversity monitoring, ecological research, and conservation. Convolutional neural networks (CNNs) have recently been very successful in image classification tasks, including recognizing different bird species.

In this project, we use a CNN by fine-tuning a pretrained model on the CUB-200-2011 dataset, which contains images of 200 bird species. ResNet18, the pretrained model on the ImageNet dataset, are used here for its good performance and efficiency in image recognition. ResNet18 is simple and effective, making it a good choice for fine-tuning. We modify ResNet18 to have an output layer that matches the 200 classes in the CUB-200-2011 dataset while keeping the pretrained weights for the earlier layers. For the evaluation of the model, we use both top-1 accuracy and top-5 accuracy, inherited from the metric on ImageNet.

The report is organized as follows: Section 2 describes the CUB-200-2011 dataset. Section 3 explains the architecture of the modified ResNet18 model. Section 4 discusses the training process, experimental setup, and results. Section 5 concludes the study, showing how effective transfer learning is for recognizing bird species and further improvements of the model.

2 Dataset

The CUB-200-2011 dataset is a key benchmark for fine-grained visual categorization, particularly in recognizing bird species. It contains 11,788 images of 200 different bird species, with about 60 images per species on average. This dataset is challenging due to the high variability within each species and the subtle differences between species.

2.1 Sample Images

Figure 1 shows some examples from the dataset, highlighting the variety in appearance, pose, and environment.

Figure 1: Sample images from the CUB-200-2011 dataset.

2.2 Train/Test Split

The dataset is carefully divided into training and testing subsets to ensure a balanced distribution of images across all classes. Table 1 shows this division.

Subset	Number of Images	Percentage
Training Set	5,994	50.8%
Testing Set	5,794	49.2%

Table 1: Train/test split of the CUB-200-2011 dataset.

The CUB-200-2011 dataset is essential for testing the effectiveness of convolutional neural networks in fine-grained image classification because of its wide range of classes and detailed variations within each class.

3 Model Architecture

We use ResNet18 as the base architecture for bird species recognition due to its proven effectiveness and efficiency in image classification tasks. ResNet18 has 18 layers, including convolutional layers, pooling operations, and residual blocks, which provide a strong framework for our task.

Input images are resized to 224×224 pixels and normalized using `transforms.Normalize`. This normalization aligns the data with the ImageNet dataset, using the mean values [0.485, 0.456, 0.406] and standard deviations [0.229, 0.224, 0.225]. We also apply data augmentation by randomly rotating the training images between -15 and 15 degrees to improve the model's generalization.

For the output layer, the modified ResNet18 keeps its original structure up to the fully connected layer, which is adjusted to predict the 200 classes of the CUB-200-2011 dataset. Kaiming initialization is also applied here aimed at finding a better starting point.

Hence, the architecture of the modified ResNet18 can be summarized as follows:

- **Input Layer:** Accepts input images of size $(3, 224, 224)$, representing RGB images.
- **Convolutional Layers:** Multiple convolutional layers with batch normalization and Rectified Linear Unit (ReLU) activation functions.
- **Residual Blocks:** Residual blocks with convolutional layers and identity shortcuts to facilitate gradient flow and support deeper networks.
- **Global Average Pooling:** A pooling layer that averages each feature map across spatial dimensions, reducing the dimensions to 1×1.
- **Fully Connected Layer (Output Layer):** A final fully connected layer that outputs 200 classes, corresponding to the bird species in the CUB-200-2011 dataset.

4 Training & Experiment Results

The training process aims to optimize model performance and prevent overfitting. We start with a grid search to find the best combination of fine-tuning learning rate (`fine_tuning_lr`) and output layer learning rate (`output_lr`) for a small epoch. Using these optimal hyperparameters, we train the model for an extended number of epochs to balance complexity and generalization.

When fine-tuning the model on the CUB-200-2011 dataset, we update the parameters throughout the network. We use a smaller learning rate for the earlier layers and a larger learning rate for the fully connected layer. This strategy helps the model retain useful features learned from the ImageNet dataset while adapting to the specific task of bird species recognition.

After identifying the best hyperparameters and training epoch, we validate the model's performance. To assess the benefits of using a pretrained model, we also train a modified ResNet18 with random weight initialization using the same hyperparameters. This comparison highlights the improvements achieved through transfer learning.

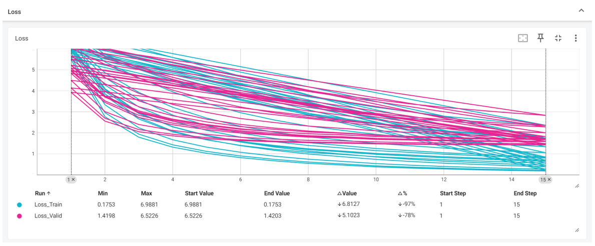
By the way, unless otherwise specified, the parameters used in this section are set to their default values, as detailed in Table 2. Additional parameters with default values are added here to improve the model's performance.

Variable	Value
`num_epochs`	30
`fine_tuning_lr`	0.0001
`output_lr`	0.001
`momentum`	0.9
`weight_decay`	0.0001
`gamma`	0.1
`step_size`	30

Table 2: Default configuration of the model.

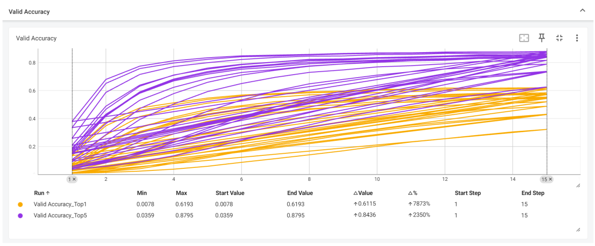
4.1 Grid-Search Optimal Learning Rates

In the initial phase, we conducted a grid search to find the best combination of fine-tuning and output layer learning rates. Each combination was used to train the model for 15 epochs. Figure 2, 3 visualize the results and table 3 summarizes them.



	Run ↑	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
●	Loss_Train	0.1753	6.9881	6.9881	0.1753	↓6.8127	↓-97%	1	15
●	Loss_Valid	1.4198	6.5226	6.5226	1.4203	↓5.1023	↓-78%	1	15

Figure 2: Loss during the grid search



	Run ↑	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
●	Valid Accuracy_Top1	0.0078	0.6193	0.0078	0.6193	↑0.6115	↑7873%	1	15
●	Valid Accuracy_Top5	0.0359	0.8795	0.0359	0.8795	↑0.8436	↑2350%	1	15

Figure 3: Accuracy during the grid search

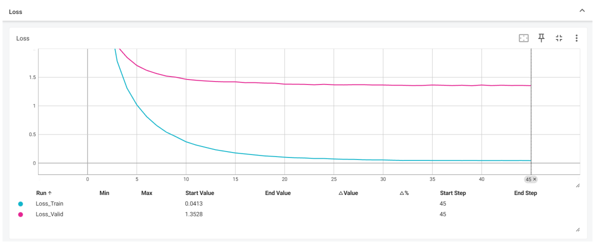
Fine-tuning LR	Output LR	Top-1 Accuracy
0.00005	0.00100	0.322
0.00005	0.00200	0.430
0.00005	0.00500	0.519
0.00005	0.00800	0.543
0.00005	0.01000	0.547
0.00010	0.00100	0.428
0.00010	0.00200	0.487
0.00010	0.00500	0.553
0.00010	0.00800	0.570
0.00010	0.01000	0.575
0.00050	0.00100	0.571
0.00050	0.00200	0.583
0.00050	0.00500	0.608
0.00050	0.00800	0.618
0.00050	0.01000	0.619

Table 3: Validation accuracy for different combinations of fine-tuning and output layer learning rates.

From the figures, we can find that as learning rate grows, the loss becomes smaller and accuracy becomes larger (here the tooltips of each plot are not shown). According to the table, the best combination, a fine-tuning learning rate of 0.0005 and an output layer learning rate of 0.01, achieved the highest top-1 validation accuracy of **61.9261%**. This combination are used for further training to determine the optimal number of epochs and to compare with a model using randomly initialized weights.

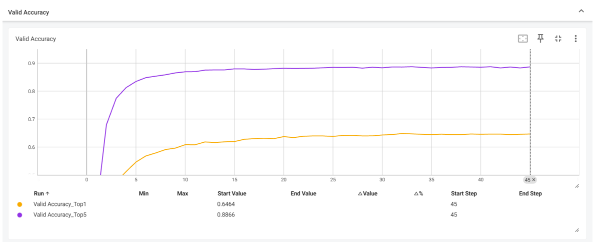
4.2 Finding Optimal Number of Epochs

Next, we varied the number of epochs from 5 to 45, in increments of 5, to determine the optimal training duration while avoiding overfitting. Table 4 presents the results, and Figures 4 and 5 display the corresponding loss and accuracy during training.



	Run ↑	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
●	Loss_Train			0.0413				45	
●	Loss_Valid			1.3528				45	

Figure 4: Loss during training



	Run ↑	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
●	Valid Accuracy_Top1			0.6464				45	
●	Valid Accuracy_Top5			0.8866				45	

Figure 5: Accuracy during training

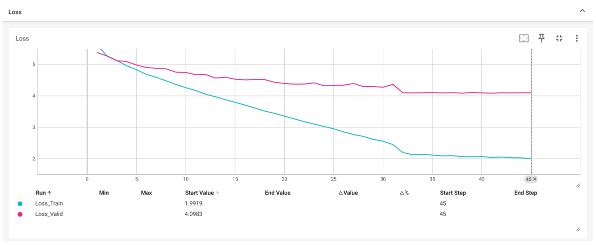
Epochs	Top-1 Accuracy	Top-5 Accuracy
5	0.546	0.834
10	0.608	0.869
15	0.619	0.879
20	0.637	0.882
25	0.639	0.885
30	0.643	0.886
35	0.648	0.887
40	0.648	0.887
45	0.648	0.887

Table 4: Validation accuracy for different numbers of epochs.

We observed that the loss and accuracy plateaus after approximately 25 epochs. Subsequently, the training loss decreases very closely to 0, indicating overfitting of the model. By analyzing the accuracy table and training logs, we found that the model achieved its best validation accuracy at epoch 32. Hence, we applied the early stopping technique and saved the model's best parameters at epoch 32. The top-5 accuracy is **88.6089%**, and the top-1 accuracy is **64.7912%**.

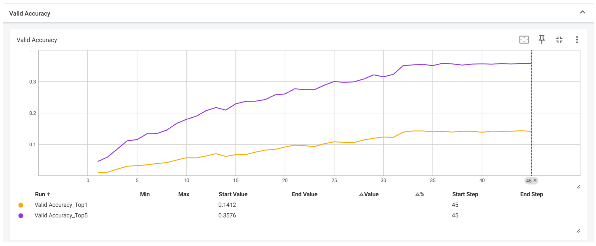
4.3 Training with Randomly Initialized Weights

After training for a larger number of epochs with pre-trained weights, we now investigate the results when using randomly initialized weights. The loss and accuracy trends are depicted in Figures 6 and 7, respectively.



	Run ↑	Min	Max	Start Value ↑	End Value	△Value	△%	Start Step	End Step
●	Loss_Train			1.9919				45	
●	Loss_Valid			4.0983				45	

Figure 6: Loss curve with randomly initialized weights



	Run ↑	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
●	Valid Accuracy_Top1			0.1412				45	
●	Valid Accuracy_Top5			0.3576				45	

Figure 7: Accuracy curve with randomly initialized weights

It is notable that there is a steep gap in both loss and accuracy around epoch 30. This anomaly may be attributed to the application of an inadequate learning rate decay technique. Nevertheless, despite this observation, the key finding remains that the model initialized with random weights achieves a top-1 accuracy of only around 15%. This stark contrast to the performance of our pre-trained model underscores the efficacy of the pre-training technique employed in this study.

5 Conclusion

Our study demonstrates the effectiveness of fine-tuning pretrained CNNs for bird species recognition tasks. By leveraging transfer learning techniques and optimizing hyperparameters, we achieved top-1 accuracy of **64.7912%** and top-5 accuracy of **88.6089%** on the CUB-200-2011 dataset, which is much higher than the accuracy of **14.3424%** using randomly initialized weights. Our findings also underscore the potential of CNNs in biodiversity monitoring and conservation efforts.

In conclusion, further research is warranted to explore advanced training strategies, such as ensemble learning and data augmentation, to enhance model performance. Additionally, investigating the generalizability of the proposed approach to other species recognition tasks could broaden its applicability in ecological research and conservation biology.