# Bird Species Classification using CNNs

By Maisey Toczek

*Abstract*— **This final project explores the ability of various CNN (convolutional neutral network) architectures to accurately identify different species of birds.**

## I. INTRODUCTION

You look out your kitchen window to see a small bird with a rosy head and streaky brown feathers. What kind of bird could it be? With over 1000 species inhabiting the United States alone, learning to accurately identify birds can be a time-consuming skill to develop. To make identification even more complicated, many species will appear entirely different depending on whether they are male or female, juvenile or adult. While there is a lot of joy in being able to quickly name the birds you see, sometimes you need a little help with figuring it out, and bird guidebooks can often be incomplete or difficult to carry. What if there was a tool to help you with this? Snap a picture of the bird, upload it to the application, and get identification quickly. This makes getting into birding accessible and hassle-free.

For this project, 3 convolutional neural networks (CNNs) architectures were trained on a data set of approximately 12,000 images containing 200 species (classes) of birds [12]. ResNet, one of the three architectures, had been used and refined on this dataset by other third-parties, and it was able to achieve 80% accuracy [6]. AlexNet and SqueezeNet were chosen as the remaining two architectures.

In this work, various fine tuning techniques were used to improve the overall accuracy of the AlexNet and SqueezeNet models with the goal of matching the accuracy capabilities shown by previous works. Techniques such as data normalization, batch normalization, data augmentation, and variable rate scheduling were used to improve the overall accuracy of the AlexNet and SqueezeNet models. However, even with these techniques, only 47% and 38% accuracies, respectively, were achieved. ResNet remains the best architecture for creating a model to identify bird species. There still remains areas for improvement, but this project is a great starting point for solving this problem.

## II. LITERATURE REVIEW

Identify secondary classifications of animals can be tricky (eg. Primary: Bird, Secondary: Blue jay). Traditional identification methods such as background knowledge, bird guides, and online resources are either take time to develop a skill or do not make it easy to narrow down on the bird you saw. Beyond bird watching, correctly identifying animal species is important to conservation, ecology, and agriculture [11]. This demand for species identification fits well with the ability of CNNs to accurately classify images.

Convolutional Neural Networks (CNNs) are a class of deep learning models designed specifically for processing image data. They are built around convolutional layers, which extract spatial features from images by applying filters that detect patterns such as edges, textures, and shapes. These features are progressively refined through pooling layers, which reduce dimensionality while retaining important information, making the model more efficient. Activation functions like ReLU (Rectified Linear Unit) introduce non-linearity, enabling CNNs to learn complex representations. Fully connected layers at the end of the network aggregate extracted features to make predictions [13].

CNNs have been extensively applied to automate the identification of different animals within an environment [11], addressing a range of use cases across various habitats and imaging conditions, including in underwater photography of marine animals [1]. These models must contend with the challenge of distinguishing between visually similar species, as well as dealing with environmental variations like lighting, pose, and background clutter. There has been great success

in identifying different types of animals within environments (eg. Bear vs. Deer) [11], but traditional CNNs struggle when discerning between animals within the same family. Some more recent research shows success with using Multi-Part Convolutional Neural Networks (MP-CNNs) to succeed in fine-grained classification. This is where you are classifying on a secondary level (again Bird vs Blue Jay) [3].

The performance of CNNs in animal species identification varies depending on the architecture used, with shallow and deep models offering different strengths. Early architectures like AlexNet laid the foundation for CNN-based species identification, providing strong feature extraction capabilities in relatively simple networks. However, deeper architectures such as ResNet have significantly outperformed earlier models by introducing innovations like residual connections, which enable the training of very deep networks without performance degradation. These more recent architectures are able to better capture the smaller details required for distinguishing between species.

## III. METHODS

As previously mentioned, three CNN architectures, more specifically Deep Neural Networks (DNNs) were chosen: AlexNet, SqueezeNet, and ResNet.

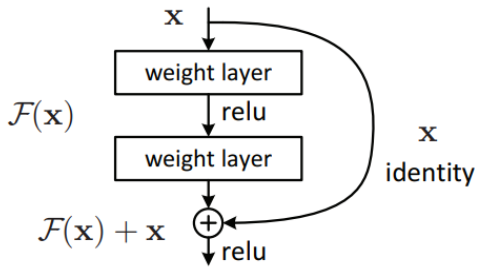
### A. Architectures

AlexNet, introduced in 2012, consists of 8 layers: 5 convolutional layers and 3 fully connected layers. ReLU is used as the activation function due to its non-linearity. AlexNet's architecture was designed to process high-dimensional image data and reduce overfitting in large-scale image recognition tasks [9]. However, its large parameter count makes it more computationally demanding compared to more modern architectures.

SqueezeNet, introduced in 2016, is a scaled down version of AlexNet which provides a similar level of accuracy with 50x fewer parameters and 5x faster computation time [8]. These optimizations are achieved using 'Fire modules'. The some of the traditional 3x3 filters are replaced with 1x1 filters, and the number of input channels to the remaining 3x3 filters are decreased. The squeeze layers of 1x1 convolutions reduce dimensionality, and the expand layers of 1x1 and 3x3 convolutions extract features. Overall, SqueezeNet is able to maintain the accuracy levels of AlexNet with fewer resources.

ResNet (Residual Network), introduced in 2015, is a more modern architecture which addresses issues with increasing the number of layers in the neural networks [7] which causes VanishingExploding gradients. This occurs when gradients become 0 or too large which leads to an increase in the training error. To combat this issue, ResNet introduces Residual Blocks. This connects activations that can be separated by multiple layers, allowing some intermediate layers to be skipped. This connection is called a 'skip connection', and it is used to avoid layers that may lead to a degradation of performance.

Fig. 1. Skip Connection




### B. Implementation

*1) Dataset:* The PyTorch DataLoader [10] data primitive was used to manage creating the dataset used by all three architectures.

*2) ResNet:* For this dataset, a user of Kaggle was able to use ResNet34 to achieve approximately 80% accuracy [6]. Various techniques such as data augmentation, normalization, transformation, Adam optimizer, learning rate finder, triangular training policy, and batch normalization were used. The model consisted of 7 layers of ResNet34 along with 2 custom fully connected layers that include batch normalization. The learning rate finding implementation was based on fast.ai [4] method. Additionally, triangular training policy was used. This increases the learning rate for half a cycle, then decreases it when it results in 'super convergence' [6]. Finally, the implementer froze and unfroze layers to find the best learning rate. This notebook

was implemented to run alongside the other two architectures.

*3) AlexNet and SqueezeNet:* Generic tutorials were used as templates for the AlexNet [2] and SqueezeNet [5] implementation. These implementations by default did not include many of the accuracy improving techniques, but they were incrementally added to analyze how they contributed to increases in overall performance. The Adam optimizer was included in the SqueezeNet implementation.

### C. Testing

All of the hyperparameters of the ResNet implementations were unchanged from the original and held constant. For both AlexNet and SqueezeNet, the learning rates were remodeled until a successful value was found.

*1) Augmentation:* Both architectures were tested with and without augmentation.

*2) Batch Normalization:* Both architectures were tested with and without batch normalization.

## IV. RESULTS

The results of testing the three convolutional neural network (CNN) architectures—AlexNet, ResNet, and SqueezeNet—demonstrate clear differences in performance, training efficiency, and size. This section compares the architectures based on the inclusion of data augmentation, the use of variable learning rates, and their loss and accuracy metrics. Additionally, the resulting model sizes from each architecture will be analyzed.

### A. Model Size

The model size, presented in Table I, shows a significant variation in resource demands between the different architectures. AlexNet, with a size of 230,869 KB, is the largest by a wide margin, followed by ResNet at 84,729 KB and SqueezeNet at only 3,294 KB. The compact nature of SqueezeNet makes it especially appealing for deployment on resource-constrained devices, though this size advantage comes at a cost to accuracy. An ideal final model size would be somewhere between the size of ResNet and SqueezeNet. The size of the ResNet model at 84.7MB is not insignificant.

### B. ResNet

*1) Performance:* ResNet consistently outperformed the other architectures in accuracy, with a peak accuracy of 76% when data augmentation was enabled and a variable learning rate was used. Its loss during training was the lowest across all models at 0.03. These results highlight the advantages of ResNet's deeper architecture and skip connections. The inclusion of triangular training policy also influenced this result.

*2) Learning Rate Adaptability:* The use of variable learning rates in ResNet likely contributed to its high accuracy by enabling dynamic adjustment of weight updates. This adaptability appears to have supported finding a more optimal solution compared to static learning rates used in other architectures.

### C. AlexNet

*1) Performance Without Augmentation:* AlexNet, despite it once being the highest performing model available, showed limited performance in this experiment. When trained without augmentation, its accuracy was a mere 34%, with a loss of 1.5 after 20 epochs.

*2) Impact of Augmentation:* Enabling data augmentation and extending the training to 400 epochs improved accuracy to 47%, but this performance was still significantly lower than ResNet's. Interestingly, using a variable learning rate with AlexNet offered only marginal improvements over a static rate of 0.005, with accuracy increasing by just 1% (from 46% to 47%). This suggests that while data augmentation moderately enhances AlexNet's generalization ability, its architectural limitations limit its performance on complex tasks like bird species identification.

*3) Resource Constraints:* The large size of AlexNet also poses challenges for use as an application as the model itself requires 230MB of storage space. This eliminates its consideration for use in an application due to the impracticality of the model size.

### D. SqueezeNet

*1) Performance Without Augmentation:* SqueezeNet demonstrated mixed results. Its lightweight design, with a model size of only

3,294 KB, is would be the preferred model for mobile applications. However, this compact size resulted in compromised performance. Without augmentation, SqueezeNet achieved only 24% accuracy with a loss of 2.4, the worst among all configurations.

*2) Impact of Augmentation:* Data augmentation markedly improved SqueezeNet's performance, increasing its accuracy to 38% and reducing the loss to 0.14. However, even with augmentation, its accuracy was well below that of both ResNet and AlexNet.

### *E. Impact of Data Augmentation*

Across all models, the inclusion of data augmentation generally improved accuracy, in SqueezeNet (from 24% to 38%) and AlexNet (from 34% to 47%). Augmentation likely enhanced the models' ability to generalize by exposing them to a broader range of data variations.

### *F. Effect of Variable Learning Rates*

Variable learning rates demonstrated clear benefits, particularly for ResNet, which achieved its highest accuracy (76%) with this approach. In contrast, AlexNet showed only marginal improvements with variable rates, indicating that architectural factors may limit the benefits of dynamic learning rates. SqueezeNet was not tested with variable rates, leaving open the possibility of performance improvements through this method.

### *G. Summary of Findings*

ResNet emerged as the most effective architecture, having balance between accuracy (76%), low loss (0.03), and moderate size (84,729 KB). SqueezeNet's minimal size is ideal for constrained environments but compromises on accuracy significantly. AlexNet, while historically influential, struggled in both accuracy and efficiency compared to ResNet. These results show the impact of modern architectural innovations like skip connections and adaptive learning strategies for achieving high performance in complex classification tasks.

## V. DISCUSSION

Overall, the performance of the ResNet implementation were not able to be replicated by AlexNet and SqueezeNet. Even with the implementation of accuracy improving features, the benefits of the ResNet model with learning rate scheduling and triangular training policy were more significant. There are still more features that could be made to the AlexNet and SqueezeNet implementations that could lead to more improvements, but I do not think this will lead to comparable performance to ResNet. These features include more refined learning rate scheduling, gradient clipping, and triangular training policy. The 2 experimental architectures are limited by the lack of these features.

| Architecture | Size (KB) |
|---|---|
| AlexNet | 230,869 |
| ResNet | 84,729 |
| SqueezeNet | 3,294 |

TABLE I
MODEL SIZES

This leads me to conclude that ResNet is the better choice to solve the problem of classifying bird species. However, I still think there is room for improvement that could lead to a better solution. The current implementation does not include gradient clipping, and I have not analyzed the distribution of incorrect classifications. Additionally, I would add a 'top 5' feature that returns if the correct label is in the top 5 classes that match the image. This would not only result in an increased accuracy, but it would also give a potential user a more helpful output. Instead of returning only 1 class with less confidence, an application can return 5 classes (with corresponding images) that is more likely to include the correct classification, and the user can make the final decision.

The model training is also limited by the size of the dataset. Currently, there are only approximately 58 images per class. I believe that the model would be accurate if there were at least 100 images per class. With 200 total classes and so many similarities between many bird species, achieving over 90% accuracy would be a difficult task.

The dataset includes one feature that could be improve the model performance: bounding boxes. For each image, they had human participants place bounding boxes around the bird in each image.

| Architecture | Augmentation | Normalization | Bath Normalization | Learning Rate | Epoch | Loss | Accuracy |
|---|---|---|---|---|---|---|---|
| ResNet | Yes | Yes | Yes | Variable | N/A | 0.03 | 76% |
| AlexNet | No | Yes | Yes | 0.005 | 20 | 1.5 | 34% |
| | Yes | Yes | Yes | Variable | 400 | 0.16 | 47% |
| | Yes | Yes | Yes | 0.005 | 400 | 0.16 | 46% |
| SqueezeNet | No | Yes | Yes | 0.0001 | 275 | 2.4 | 24% |
| | Yes | Yes | Yes | 0.0001 | 275 | 0.14 | 38% |

TABLE II

TESTING RESULTS WITH CHANGES IN AUGMENTATION AND VARIABLES LEARNING RATE

If I cropped the images down to those bounding boxes and added padding, this could improve the accuracy of the model by reducing the amount of unnecessary information in the image.

I plan to continue refining the ResNet model and tailoring the output to be most useful for users who would like to identify bird species. This would also include working with user provided images. Unfortunately, AlexNet and SqueezeNet have proved ineffective for solving the problem, but the experience of working with and tailoring the models helped me better understand CNNs and DNNs.

## REFERENCES


[1] Zheng Cao et al. "Marine animal classification using combined CNN and hand-designed image features". In: *OCEANS 2015 - MTS/IEEE Washington*. 2015, pp. 1–6. DOI: 10.23919/OCEANS.2015.7404375.

[2] DigitalOcean. *Writing AlexNet from Scratch in PyTorch — DigitalOcean*. en. 2024. URL: https://www.digitalocean.com/community/tutorials/alexnet-pytorch.

[3] S. Divya Meena and L. Agilandeeswari. "An Efficient Framework for Animal Breeds Classification Using Semi-Supervised Learning and Multi-Part Convolutional Neural Network (MP-CNN)". In: *IEEE Access* 7 (2019), pp. 151783–151802. DOI: 10.1109/ACCESS.2019.2947717.

[4] fast.ai. *fast.ai—Making neural nets uncool again*. en. 2024. URL: https://www.fast.ai/.

[5] *GitHub - Lornatang/SqueezeNet-PyTorch: PyTorch implements 'SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size' paper. — github.com*. https://github.com/Lornatang/SqueezeNet-PyTorch. [Accessed 19-11-2024].

[6] MohammadHossein Givkashi. *CUB image classification with ResNet34*. en. 2022. URL: https://kaggle.com/code/givkashi/cub-image-classification-with-resnet34.

[7] Kaiming He et al. *Deep Residual Learning for Image Recognition*. arXiv:1512.03385 [cs]. Dec. 2015. URL: http://arxiv.org/abs/1512.03385 (visited on 11/19/2024).

[8] Forrest N. Iandola et al. *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size*. arXiv:1602.07360 [cs]. Nov. 2016. URL: http://arxiv.org/abs/1602.07360 (visited on 11/19/2024).

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[10] PyTorch. *Datasets DataLoaders — PyTorch Tutorials*. 2024. URL: https://pytorch.org/tutorials/

beginner / basics / data _ tutorial.html%7D.

[11] Michael A. Tabak et al. "Machine learning to classify animal species in camera trap images: Applications in ecology". en. In: *Methods in Ecology and Evolution* 10.4 (Apr. 2019). Ed. by Theoni Photopoulou, pp. 585–590. ISSN: 2041-210X, 2041-210X. DOI: 10.1111/2041-210X.13120. URL: https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210X.13120 (visited on 11/20/2024).

[12] C. Wah et al. *Caltech-UCSD Birds-200-2011*. Tech. rep. CNS-TR-2011-001. California Institute of Technology, 2011.

[13] *What are Convolutional Neural Networks? — IBM*. en. Oct. 2021. URL: https://www.ibm.com/topics/convolutional-neural-networks (visited on 11/20/2024).