Comparative Analysis of Deep Learning Architectures for Fine-Grained Bird Classification

Fernanda Santiago Martinez, m.f.santiagomartinez@student.utwente.nl, University of Twente, The Netherlands

This study compares the performance and computational cost of various state-of-the-art deep learning models for fine-grained bird classification. We evaluate ResNet-50, VGG-16, Inception-v3, EfficientNet-B3, MobileNetv3, ViT, DeiT, and ConvNeXt on the CUB-200-2011 dataset. Each model was trained using the PyTorch library on a high-performance server, with early stopping employed to prevent overfitting. Our results indicate that DeiT achieves the highest accuracy (85.6%) and F1 score (85.6%), while MobileNetv3 shows the best computational efficiency. These findings offer valuable insights into selecting appropriate models for fine-grained classification tasks based on specific application requirements.

Additional Key Words and Phrases: deep-learning architecture, fine-grained bird classification.

1 INTRODUCTION

Fine-grained bird classification refers to distinguishing between bird species based on minor differences. Bird classification is relevant to various domains such as ornithology, ecology, and biology, as birds are commonly used to assess an environment's status and diversity due to the bird population being very susceptible to ecological changes and more accessible to observe than other species [17].

In computer vision, there is an essential distinction between fine-grained and general classification tasks. General classification involves distinguishing significantly different categories, the differences between which are usually evident and easy to identify. On the other hand, fine-grained classification requires distinguishing between visually similar classes that belong to the same broader category.

Bird classification is also a significant task in the computer vision field because of the major technical challenges implied in identifying the species of a bird. These challenges include high intraclass variance, which refers to images of birds of the same species usually presenting different postures or perspectives; low interclass variance, which means birds of different species have very similar characteristics; and finally, data biases due to the limited data there is for some rare bird species, making the datasets to be unbalanced [17].

To address the first two challenges, focusing on subtle differences in key features of birds, such as the head, tail, and wings, is needed for successful classification. Therefore, the fine-grained bird classification task usually consists of two stages [16]. The first is to localize those key feature regions on the image just mentioned and avoid irrelevant information as the background. The second stage is to perform label prediction based on the features extracted in the first stage.

Deep learning architectures have been widely used for image classification in recent years, outperforming other alternatives such as machine learning [12] and eliminating manual feature extraction, making the classification more efficient [10]. These architectures have been trained on immense datasets and, therefore, have already learned relevant image features and model weights that can be transferred for training a new classification model, decreasing the learning time and improving the accuracy of this new model [1], this is called transfer learning which along with fine tuning are necessary techniques for fine-grained classification to extract the detailed features. Also, for general classification, deep learning models work well alone. As a result, substantial research has been done to improve the performance of such models or develop new ones.

However, the increase in the diversity of deep learning architectures introduces the question of which architecture performs best for our specific task: bird image classification. Consequently, this research aims to evaluate and compare various available deep-learning architectures using the CUB-200-211 as a benchmark dataset. The approach involves assessing the models' performance using standard metrics and analyzing their computational costs. By comparing these aspects, the aim is to identify a relation and trade-offs that can guide the selection of appropriate models based on specific requirements and constraints.

The problem statement leads to the following three research questions:

1. How do various deep learning architectures perform in fine-grained bird classification in terms of relevant performance metrics as accuracy and f1-score?
2. What are the computational requirements of these models, including training duration, inference time and resource utilization?
3. What is the relationship between the computational cost of different deep learning models and their performance metrics?

The paper structure is as follows: the next section reviews related work in deep-learning architecture comparison. Section 3 provides information on different deep-learning architectures. The next section describes the research method, including the dataset, its preprocessing steps, and the training and evaluation process. Section 5 presents the

TScIT 41, July 5, 2022, Enschede, The Netherlands

classification results of each model's performance metrics and computational cost followed by detailed discussion in Section 6. Finally, a summary and suggestions for future research are given as a conclusion to this paper.

2 LITERATURE REVIEW

Fine-grained image classification has been challenging in computer vision in recent years. Numerous studies have explored using deep learning models to tackle this problem, evaluating different architectures to determine their effectiveness and precision. This section reviews prior research comparing deep learning architectures in fine-grained image classification.

2.1 Deep Learning Model in Fine-Grained Image Classification

[8] evaluated the performance of several deep learning architectures on the CUB- 200-2011 dataset, highlighting the strengths of models like AlexNet and VGG in capturing precise visual features helpful for distinguishing between bird species. However, their analysis was limited by the architectures available at the time, which more advanced models have outperformed.

[5] conducted a comprehensive study on fine-grained image classification, comparing various CNN architectures, including ResNet and DenseNet. They found that deeper networks with residual connections performed better as they could learn more complex representations without the disadvantage of the vanishing gradient problem. Their work emphasized the importance of architectural innovation in improving classification accuracy.

2.2 Transfer Learning and Fine-Tuning

Another line of research has focused on using transfer learning and fine-tuning pre-trained models for fine-grained classification. [2] proposed using a considerable amount of data from related domains to pre-train models and fine-tuning them later on smaller fine-grained datasets. This approach significantly improved the performance of models like Inception-v3 and ResNet-50.

Similarly, [13] investigated the effectiveness of fine-tuning. Their research demonstrates that when pre-trained on large datasets like ImageNet and fine-tuned on specific fine-grained datasets, models such as EfficientNet and MobileNetv3 achieved great accuracy with reduced computational costs.

3 DEEP-LEARNING ARCHITECTURES

The comparison base for this research consists of eight different deep-learning architectures. This section overviews the ResNet-50, VGG-16, Inception-v3, EfficientNet-B3, MobileNetv3, Vision Transformer (ViT), Data-efficient Image Transformer (DeiT), and ConvNeXt. Each model depicts a distinctive approach to image classification, with different architectural designs, strengths, and trade-offs in performance and computational cost. These models are broadly classified into four categories: Convolutional Neural Networks (CNNs), Modern CNNs and variants, Vision Transformers, and Hybrid models.

3.1 Convolutional Neural Networks (CNNs)

CNN architecture is composed of different layers which follow this structure [12]:

- In the input layer, the format is usually a tensor.
- The convolutional layer consists of filters used to find the features in the image, which are later used for classification.
- The non-linearity layer applies an activation function to model a response variable non-linearly with its explanatory variables.
- The pooling layer reduces the images and parameters by taking the maximum, minimum, or average value to preserve the relevant information.
- The fully connected layer translates images into labels with categories.
- Finally, the output layer presents the classification decision.

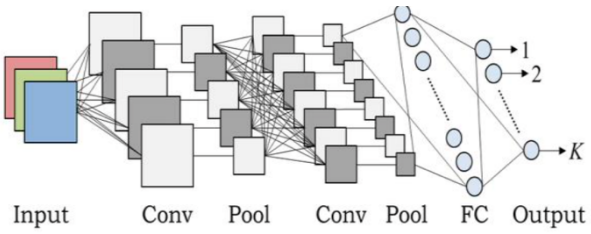

Figure 1. **CNN architecture. (Krishna & Kalluri, 2019)**

3.1.1 ResNet-50. ResNet-50, a 50-layer deep convolutional neural network, is part of the Residual Networks (ResNet) family introduced by He et al. in 2015. ResNet's key innovation is the addition of residual blocks to its convolutional layers that can skip connections and be directly added to the output of the last convolutional layer. These blocks make it easier to train deeper networks and avoid vanishing learning effects [6]. ResNet-50 is known for balancing depth and computational efficiency, making it a popular choice for image classification tasks.

3.1.2. VGG-16. VGG-16 is a deep convolutional network that uses 16 weight layers, consisting of 13 convolutional layers and three fully connected layers, developed by the Visual Geometry Group at the University of Oxford to measure the influence of network depth on classification results [13]. VGG-16 is characterized by its simplicity and uniform architecture, utilizing small 3x3 convolutional filters throughout the network, making it possible to extract multiple complex features at low cost [9]. As a result, this model has achieved high performance on image classification tasks. However, its large number of parameters and depth makes it computationally expensive.

3.1.3. Inception-v3. Inception-v3, introduced by [14], incorporates several advanced techniques, such as factorized convolutions, batch normalization, and auxiliary classifiers. The Inception architecture can capture various spatial hierarchies in the input image by employing multiple convolutional filter sizes at each layer. It was designed to perform well under strict memory and computational budget constraints. Therefore, it is known for its high accuracy and

efficiency, as its computational cost is much lower than that of previous higher-performing models [14].

3.2 Moder CNNs and variants

3.2.1 EfficientNet-B3. EfficientNet-B3, proposed by Tan and Le (2019), evenly scales up the network dimensions (depth, width, resolution) to achieve a good trade-off between accuracy and computational efficiency [4]. It uses mobile inverted bottleneck convolutions and squeeze-and-excitation optimization, making it highly efficient in parameter count and FLOPs (floating-point operations per second).

3.2.2 MobileNetv3. MobileNetv3, developed by [7], is designed for mobile and embedded vision applications. It combines depthwise separable convolutions with a modern architecture search method to optimize accuracy and efficiency. Depthwise separable convolutions are made up of two layers: depthwise to apply a single filter per each input channel, and then pointwise is used to create a linear combination of the output of the depthwise layer [7]. The model's small size and low latency make it an excellent alternative where resources are limited.

3.3 Vision Transformers

3.3.1 Vision Transformer (ViT). ViT, introduced by [3], is based on the transformer architecture, which has been broadly used successfully in natural language processing tasks. It treats an image as a sequence of patches and applies a standard transformer encoder to process this sequence. ViT has shown competitive performance on image classification tasks when pre-trained on large amounts of data. It attains excellent results compared to CNNs, requiring fewer computational resources to train [3].

3.3.2 Data-efficient Image Transformer (DeiT). DeiT, proposed by [15], is a ViT variant designed to be more data efficient. It incorporates knowledge distillation to enhance training efficiency and performance on smaller datasets, making it a compelling choice for scenarios with limited training data.

3.4 Hybrid models

3.4.1 ConvNeXt. Introduced by Liu et al., ConvNeXt combines elements from traditional CNNs and modern transformer models, offering strong performance and improved computational efficiency. It features the Swin transformer's layer structure, downsample method, activation function, data-processing method, inverted bottleneck, and depthwise convolution. ConvNeXt-Tiny has low computational requirements for training and excellent feature extraction abilities with few parameters [10].

4 METHOD

4.1 Data preparation

The Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset is a widely used benchmark for fine-grained image classification tasks. This dataset comprises 11,788 images across 200 bird species, each with approximately 60 images. The images are annotated with one bounding box, 312 Binary Attributes, and 15-part locations, providing a broad set of features for model training and evaluation. For this research, the data was divided into train and test sets based on the train_test_split.txt file provided within the dataset to ensure consistency and comparability with existing studies. The train set has 5994 images, and the test set has 5794.

We apply several data augmentation techniques to enhance the model's generalization ability. These techniques manipulate the training images to differ to some degree on each training iteration (epoch), preventing overfitting and improving the model's efficiency. The augmentation differs slightly between the training and testing sets to align with best practices in deep learning.

Training Data Augmentation:

- RandomResizedCrop: crops a random portion of the image and resizes it to the target size of pixels, which varies depending on the model.
- RandomHorizontalFlip: The image is horizontally flipped with a probability of 0.5.
- ToTensor: Converts the image to a tensor, the required format for PyTorch models.
- Normalize: It normalizes the image tensor by subtracting the mean and dividing by the standard deviation based on the statistics of the ImageNet dataset, a common practice for models pre-trained on ImageNet.

Test Data Augmentation:

- Resize: It resizes the shorter side of the image while maintaining the aspect ratio.
- CenterCrop: It crops the central part of the image to the required size, keeping the most relevant part of the image for evaluation.
- ToTensor
- Normalize

These data augmentation techniques are applied evenly across all models in the study, to ensure differences in performance can be attributed to the model architectures rather than variations in the preprocessing.

4.1.1 Data Loading. The DataLoader utility from PyTorch was used to load the images from the dataset and provide them to the model during the training and evaluation processes. Data loaders iterate over the dataset, splitting it into smaller batches of a specified batch size, randomizing the order of the data, and using the specified number of worker (*num_workers)* processes to load the data in parallel.

For training, data loaders were configured with a batch size of 16. The *num_workers* parameter was set to 2 or 1 depending on the specific model's GPU memory and computation requirements.

For evaluation, data loaders used a batch size of 4, and 4 workers in all models to facilitate faster evaluation while considering memory constraints.

4.2 Model Training

All eight deep learning architectures were trained on the CUB-200-2011 dataset using the Python programming

language with the PyTorch library on Jupyter Notebook, leveraging a server equipped with an Nvidia Tesla T4 GPU, 48-core/96-thread CPU, and 256 GB of memory.

4.2.1 Training Procedure. Each model was trained for a maximum of 20 epochs (iterations). However, early stopping was employed to prevent overfitting and optimize training time. Training was terminated if the validation loss did not improve for five consecutive epochs.

4.2.2 Loss Function and Optimization. All models were trained using the cross-entropy loss function, which is well-suited for classification tasks. The loss function measures the model's performance by comparing the predicted class probabilities with the actual class labels.

The Stochastic Gradient Descent (SGD) optimizer was utilized for training, with momentum to accelerate convergence and improve stability.

A learning rate scheduler was employed to enhance the training process further, decreasing the learning rate by 0.1 every seven epochs which helps fine-tune the model by reducing the learning rate as training progresses, allowing the model to converge more smoothly and avoid overshooting the optimal solution.

The learning rate controls the step size at each epoch while the training process moves toward a minimum of the loss function.

4.2.3 Hyperparameters. The learning rate and number of workers were the only parameters used in the training process that varied among the models due to differences in their architectural complexity and the amount of GPU memory required. Table 1 shows each model's hyperparameters.

Table 1. Model's training hyperparameters

Model	Num workers	Learning rate
ResNet-50	2	0.001
VGG16	2	0.001
InceptionV3	1	0.010
EfficientNet-B3	1	0.010
MobileNetv3	2	0.010
ViT	2	0.010
DeiT	2	0.010
ConvNext	2	0.005

The training function included a validation step at the end of each epoch to monitor the validation loss. The model with the lowest validation loss was saved as the best model for each architecture.

This standardized training setup allows a fair comparison of the different deep-learning models.

4.3 Evaluation

The models saved during the training process were evaluated on the test set for testing. Performance metrics were computed for each model to assess its classification capabilities, and computational profiling was monitored during each model's training process.

The performance metrics used in this study are:

- Accuracy: Measures the proportion of correctly classified images out of the total number of images.
- Precision: Calculates the ratio of true positive predictions to the total number of positive predictions.
- Recall: Measures the ratio of true positive predictions to the total number of actual positive instances.
- F1-Score: The weighted harmonic mean of precision and recall.

For evaluating the computational cost, the following information was measured:

- Training Time: The total time to train each model, including all epochs.
- Inference Time: The average time it takes for each model to predict the label of a single image in the test set.
- Memory Usage: The amount of allocated GPU memory consumed by each model during training.

5 RESULTS

This section presents the results of our experiments, comparing the performance and computational cost of eight different deep learning architectures—ResNet-50, VGG-16, Inception-v3, EfficientNet-B3, MobileNetv3, ViT, DeiT, and ConvNeXt—on the CUB-200-2011 fine-grained bird classification task.

5.1 Performance metrics

Table 2 summarizes the accuracy, precision, recall, and F1-score for each model on the test set.

Table 2. Model's performance metrics

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
ResNet-50	79.72	80.56	79.84	79.53
VGG16	77.72	78.20	77.85	77.69
InceptionV3	74.75	75.79	74.95	74.35
EfficientNet-B3	73.17	75.10	73.28	72.62
MobileNetv3	77.96	78.50	78.10	77.83
ViT	85.16	85.69	85.13	85.08
DeiT	85.47	85.90	85.70	85.60
ConvNext	75.28	75.54	75.40	74.86

ViT and DeiT achieved the highest performance metrics, outperforming ResNet-50, the third-best model, by around 5% in all metrics. VGG-16 and MobileNetv3 also had strong performance, while EfficientNet-B3 and Inceptionv3 had lower but still competitive results.

5.2 Computational Cost

Table 3 details the computational costs, including training time, inference time per image, model size, memory usage for each model.

Table 3. Model's computational cost

Model	Training (min)	Inference (ms)	Memory (MB)	Size (MB)
ResNet-50	16.13	2.82	379	96
VGG16	20.23	3.01	2068	540
InceptionV3	48.33	4.56	261	89
EfficientNet-B3	43.23	8.36	137	44
MobileNetv3	10.92	2.90	52	18
ViT	81.80	11.54	992	344
DeiT	82.88	11.74	991	344
ConvNext	28.68	6.10	324	112

MobileNetv3 and ResNet-50 exhibited the lowest training and inference times. However, MobileNetv3 also had the smallest memory usage (52 MB), making it the most computationally efficient model. In contrast, DeiT and ViT both had the longest training and inference times, over 80 minutes and 11 milliseconds per images and VGG-16 has the highest memory usage.

6 DISCUSSIONS

This section analyzes and compares the results shown in the previous section. The discussion focuses on explaining the technical aspects that might be behind the performance of each model and providing a way to rank the models based on all the metrics measured: performance and computational cost.

6.1 Analysis of Results per Architecture

6.1.1 ResNet-50. In the experiment, ResNet-50 achieved outstanding results in terms of performance metrics and computational costs. The residual blocks part of its architecture helps make the training process more efficient and shorten the inference time while achieving high accuracy as residua blocks avoid the vanishing gradient problem common on deep networks.

6.1.2 VGG-16. The uniform architecture of VGG-16, consisting of small 3x3 convolutional networks, makes it a good model for capturing detailed features at a relatively low computational cost, as demonstrated in the experiment results where the training and inference times were the third fastest, and the accuracy was just below the one obtained with the ResNet model. However, the results also showed that it is the model with the largest size and memory usage due to its architecture; therefore, it is not ideal for environments with resource constraints.

6.1.3 Inception-v3. Inception-v3 was designed to be efficient under memory and computational budget constraints, resulting in the smallest size and memory usage of all CNN architectures in this study. However, as it incorporates advanced techniques to capture features effectively, its training and inference times were higher than the other CNN models. Regarding performance metrics, the results achieved were not as good as the other models, being between the bottom two models.

6.1.4 EfficientNet-B3. EfficientNet-B3 evenly scales up network dimensions to balance accuracy and computational efficiency by optimizing parameter count and floating-point operations per second (FLOPs). This approach contributes to EfficientNet-B3, showing an accuracy similar to the one obtained with Inception-v3 but with just half its memory usage and half its size. However, this also contributes to its higher training and inference times than the other CNN architectures.

6.1.5 MobileNetv3. MobileNetv3 is designed for mobile and embedded vision applications, as shown in the results; it exhibited the lowest training and inference times, memory usage, and model size, highlighting its computational efficiency in fine-grained bird classification. This model combines depth-wise separable convolutions with an architecture that aims to optimize its accuracy, making MobileNet the second-highest CNN model regarding performance metrics.

6.1.6 ViT and DeiT. Based on the transformer architecture, both ViT and DeiT treat images as sequences of patches, and ViT applies transformers for processing, while DeiT incorporates knowledge distillation. DeiT is a ViT variant explicitly designed to achieve good efficiency on small datasets. However, both models perform likewise in all metrics, probably due to the large dataset size used. Both models showed outstanding performance metrics in this study, outperforming all other models. Nonetheless, as a result of using the transformer architecture, both models' computational efficiency was inferior, as the training and inference times were significantly longer than the ones of any other model; besides, their memory usage and size were extensive, only smaller than VGG-16.

6.1.7 ConvNext. ConvNeXt balances computational requirements and feature extraction abilities by utilizing the Swin transformer's structure. This structure combines elements from traditional CNNs and transformer models. As a result, this model achieved higher performance metrics and lower training time than some CNN architectures (Inception and EfficientNet) while requiring around three times less memory and size than the transformer models evaluated (ViT and DeiT).

6.2 Composite Score

While accuracy and f1-score are fundamental measures of a model's predictive capability in fine-grained bird classification, they do not account for real-world applications' functional constraints and requirements. Training and inference times, memory usage, and model size are also crucial for determining the suitability of a particular model in environments with different computational resources, such as mobile devices with limited resources. We integrated these metrics into a single composite score through a weighted sum approach to assess the trade-offs between performance metrics, computational efficiency, and resource usage. This evaluation facilitates a more balanced comparison of models, which helps guide the selection of the most appropriate model that meets performance and

efficiency criteria in a particular deployment context or for a specific task.

For creating the composite score, each metric was normalized to a scale of 0 to 1 using min-max normalization. The normalization of metric measuring computational was inverted, as these should be minimized. Then, each metric was assigned a weight based on its relative importance. The total weight was distributed evenly between performance and computational cost metrics, resulting in accuracy and f1-score getting 25% each, and training time, inference time, mode size, and memory usage were given a weight of 12.5% each. Finally, the composite score is calculated as the weighted sum of the normalized metrics.

Table 4. Composite Score

Model	Accuracy	F1 Score	Training (min)	Inference (ms)	Memory (MB)	Size (MB)	Score
ResNet-50	79.72	79.53	16.13	16.34	379	96	71.82
MobileNetv3	77.96	77.83	10.92	16.79	52	18	69.66
DeiT	85.47	85.60	82.88	68.03	991	344	61.37
ViT	85.16	85.08	81.80	66.89	992	344	60.20
ConvNext	75.28	74.86	28.68	35.34	324	112	46.99
InceptionV3	74.75	74.35	48.33	26.40	261	89	44.62
VGG16	77.72	77.69	20.23	17.46	2068	540	42.12
EfficientNet-B3	73.17	72.62	43.23	48.43	137	44	35.48

Table 4 shows the model's ranking based on their composite score, arranged in descending order. The table shows ResNet-50 and MobileNetv3 as the top 2 models that balance performance metrics and computational cost.

7 CONCLUSIONS

In this research, a thorough comparison was conducted of eight deep learning architectures—ResNet-50, VGG-16, Inception-v3, EfficientNet-B3, MobileNetv3, Vision Transformer (ViT), Data-efficient Image Transformer (DeiT), and ConvNeXt—on the fine-grained bird classification task using the CUB-200-2011 dataset as a benchmark. This study aimed to evaluate these models in terms of their performance metrics and computational costs and to understand the relationship between these two aspects.

The results of the comparison performed provide valuable insights into the research questions formulated in the introduction of this paper that can be summarized as follows:

1. Performance Metrics: Among the models, ViT and DeiT achieved the highest accuracy, precision, recall, and F1 scores, demonstrating a superior ability of transformers to capture subtle features in bird images. ResNet-50 and MobileNetv3 also performed exceptionally well in these metrics.
2. Computational Cost: The computational cost analysis revealed substantial differences among the models. MobileNetv3 was the most efficient, with the lowest training and inference times and memory usage, making it ideal for resource-constrained environments. ResNet-50 also had low computational costs. On the other hand, DeiT and ViT required significantly more computational resources.
3. Relation Between Computational Cost and Performance Metrics: This study highlighted a clear trade-off between computational cost and performance metrics in models with a transformer architecture. DeiT and ViT, which have the highest computational costs, delivered superior performance, whereas the ConvNext model, with lower computational costs, still provided adequate but lower performance. However, based on the results obtained for CNN architectures, there is no trade-off, as the top-performing models, ResNet-50 and MobileNetv3, also had the lowest computational costs. The only exception is VGG-16, which delivers high performance with reasonable computational costs but at the cost of high resource usage.

Future research directions could focus on the following aspects:

- Extending the comparison to other fine-grained image classification datasets to generalize the findings and confirm the robustness of the conclusions
- Applying advanced optimization techniques and regularization methods to reduce the computational cost of the models without compromising their performance
- Investigating advanced data augmentation strategies to enhance the generalization ability of deep learning models and mitigate overfitting challenges in fine-grained image classification

REFERENCES

[1] Bressem, K. K., Adams, L., Erxleben, C., Hamm, B., Niehues, S. M., & Vahldiek, J. L. (2020). Comparing different deep learning architectures for classification of chest radiographs. Scientific Reports, 10(1). https://doi.org/10.1038/s41598-020-70479-Z

[2] Cui, Y., Song, Y., Sun, C., Howard, A., & Belongie, S. (2018). Large scale fine-grained categorization and domain-specific transfer learning. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4109-4118. https://doi.org/10.1109/CVPR.2018.00433

[3] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR 2021. https://openreview.net/pdf?id=YicbFdNTTy

[4] Ebenezer, A. S., Kanmani, S. D., Sivakumar, M., & Priya, S. J. (2022). Effect of image transformation on EfficientNet model for COVID-19 CT image classification. Materials Today: Proceedings, 51, 2512–2519. https://doi.org/10.1016/j.matpr.2021.12.121

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778. https://doi.org/10.1109/CVPR.2016.90

[6] Heinrich, Kai; Möller, Björn; Janiesch, Christian; and Zschech, Patrick, "Is Bigger Always Better? Lessons Learnt from the Evolution of Deep Learning Architectures for Image Classification" (2019). Proceedings of the 2019 Pre-ICIS SIGDSA Symposium. 20. https://aisel.aisnet.org/sigdsa2019/20

[7] Howard A G, Zhu M, Chen B, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications[C]. IEEE Conference on Computer Vision & Pattern Recognition, CVPR, 2017, 4510–4520.

[8] Krause, J., Jin, H., Yang, J., & Fei-Fei, L. (2016). Fine-grained recognition without part annotations. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 5546-5555. https://doi.org/10.1109/CVPR.2015.7299150

[9] Krishna, S., Kalluri, H. (February 2019). Deep Learning and Transfer Learning Approaches for Image Classification. International Journal of Recent Technology and Engineering (IJRTE). ISSN: 2277-3878, Volume-7, Issue-5S4.

[10] Li, Z.; Gu, T.; Li, B.; Xu, W.; He, X.; Hui, X. (2022). ConvNeXt-Based: Fine-Grained Image Classification and Bilinear Attention Mechanism Model. Appl. Sci. 12, 9016. https://doi.org/10.3390/ app12189016

[11] Morris, E. (2022, March 30). Image Classification of bird species using deep learning with PyTorch, Captum and ONNX. Medium. https://towardsdatascience.com/adventures-in-pytorch-image-classification-with-caltech birds-200-part-1-the-dataset-6e5433e9897c

[12] Saiharsha B, Abel Lesle A, B Diwakar, Karthika R, Ganesan M. (2020). Evaluating Performance of Deep Learning Architectures for Image Classification. IEEE Conference Record # 48766; IEEE Xplore ISBN: 978-1-7281-5371-1.

[13] Simon, M., Rodner, E., & Denzler, J. (2019). A few more examples may improve cross-domain generalization. Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), 1484-1492. https://doi.org/10.1109/WACV.2019.00162

[14] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. Cornell University. https://doi.org/10.1109/cvpr.2016.308

[15] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jegou, H., Sorbonne University, & Facebook AI. (2021). Training data-efficient image transformers & distillation through attention. Proceedings of the 38th International Conference on Machine Learning, 10347–10357. https://arxiv.org/pdf/2012.12877

[16] Yang, K., & Song, Z. (2021). Deep Learning-Based Object Detection Improvement for Fine-Grained Birds. IEEE Access, 9, 67901–67915. https://doi.org/10.1109/access.2021.3076429

[17] Wang, K., Yang, F., Chen, Z., Chen, Y., & Zhang, Y. (2023). A Fine-Grained Bird classification method based on attention and decoupled knowledge distillation. Animals, 13(2), 264. https://doi.org/10.3390/ani13020264