BDMA Project: Image Classification

Dilbar Isakova
Melissa Medjahed
Sihem Boutebal
Aimene Hammani

Our Github repository

Abstract

Fine-grained bird species classification presents significant challenges for us due to the subtle morphological differences between species. This research investigates advanced deep learning architectures to develop a robust bird image classification model. Our study systematically explores multiple neural network approaches, starting with traditional Convolutional Neural Networks (CNNs) and progressively advancing to state-of-the-art models like EfficientNet and Vision Transformers (ViT).

Initial experiments with baseline CNN and ResNet-18 architectures revealed limitations in capturing intricate bird species characteristics, achieving unsatisfying validation accuracies. record performances were realized with EfficientNet models, particularly EfficientNet-B3, which demonstrated remarkable results. Advanced techniques such as object detection preprocessing, label smoothing, and Mixup data augmentation were strategically implemented to enhance model generalization.

The research introduced innovative preprocessing strategies, including object detection using DETR ResNet-101 to isolate and focus on bird-specific features. By implementing techniques like half-precision training and sophisticated data augmentation, we developed a classification framework that effectively mitigates overfitting and improves model robustness.

Our findings underscore the significance of architectural sophistication and preprocessing techniques in fine-grained image classification.

1. Introduction

The objective of this project was to develop a high-performing deep learning model for **bird species classification** [18], a task that presents significant challenges due to the fine-grained nature of differences between species. The dataset contained images of various bird species, requiring a model capable of distinguishing subtle differences in features such as color, texture, and shape. Given the complexity of the problem, multiple architectures were explored, starting with **CNN** [4, 17] and **ResNet-18** [6] as a baseline model and progressively moving towards **EfficientNet** [12], **ConvNeXt** [9], **EfficientNet-B4/B5** [8] and **Vision Transformers (ViT-B16, Swin-Tiny)** [13] architectures.

2. Models of Initial Approach

This section provides a **detailed examination** of the models explored in the initial phase of the project. It outlines the **rationale behind choosing each architecture, modifications made to optimize performance, training methodologies, and final results**. The performance of each model is analyzed, and conclusions are drawn to guide further improvements. The progression of experiments highlights the transition from **traditional CNNs to more advanced architectures**, culminating in the selection of **EfficientNet-B3 as the most effective model in this phase**.

2.1. Convolutional neural networks (CNN)

Convolutional Neural Networks (CNNs) [4] are widely used in the field of image recognition due to their ability to detect patterns and features in images effectively. This report details the implementation, hyperparameters, and performance of a CNN model trained to classify [17]images of bird species.

2.1.1 CNN Method

Convolutional Neural Networks (CNNs) are designed to process data in the form of multiple arrays, capturing complex patterns such as textures, colors, and edges. A typical CNN architecture consists of several key components. **Convolutional layers** apply filters to transform input images into feature maps, extracting essential features. These are followed by **activation functions**, such as ReLU, which introduce non-linearity into the network, enabling it to learn complex patterns. [17] **Pooling layers** are used to reduce

the spatial dimensions of the feature maps, thereby decreasing computational load and helping to control overfitting. The extracted features are then passed to **fully connected layers**, which produce predictions for each class. Finally, **dropout** is employed as a regularization technique to prevent overfitting by randomly dropping units during the training phase. Together, these components form the backbone of the CNN architecture, allowing it to effectively capture and process the intricate features present in image data. [4]

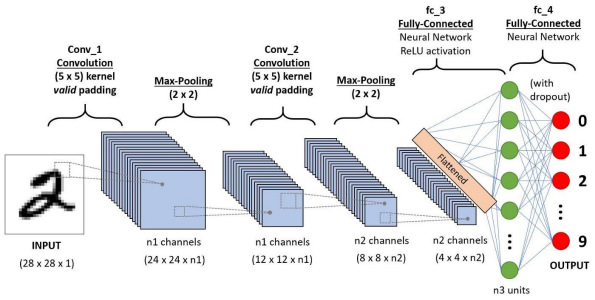


Figure 1. CNN Architecture [4]

2.1.2 Hyperparameters

The model was configured with a specific set of hyperparameters to optimize its performance. It consisted of three convolutional layers with filter sizes of 64, 128, and 256, each using a kernel size of 3×3 and padding set to 1 to maintain the spatial dimensions of the feature maps. MaxPooling layers with a 2×2 pooling size were used to downsample the feature maps. The activation function throughout the network was ReLU, and a dropout rate of 0.5 was applied to prevent overfitting. The fully connected layers included two dense layers, each with 512 units, followed by the output layer corresponding to the number of classes. The optimizer used was Adam with a learning rate of 0.001, and learning rate scheduling was handled by ReduceLROnPlateau. The model was trained with a batch size of 32 for 100 epochs, ensuring sufficient iterations to learn the dataset effectively.

The best obtained accuracy on the validation set was:

Metric	Value
Model	CNN
Train Loss	0.2461
Validation Loss	0.1754
Validation Accuracy (%)	42.72

Table 1. Performance metrics of a basic CNN model.

The CNN model demonstrated a consistent decrease in loss, suggesting effective learning. However, the maximum validation accuracy reached was 43.69%, indicating potential limitations in the model's ability to generalize. This might be due to overfitting or insufficient feature extraction capabilities of the CNN.

2.2. ResNet-18: Baseline Model

The first model implemented after experiement with CNN was **ResNet-18**, a well-established convolutional neural network (CNN) known for its efficiency and relatively small number of parameters. The ResNet architecture introduced **residual connections**, which help mitigate the The vanishing gradient problem [7] which occurs when gradients become extremely small during backpropagation, making it difficult for deep neural networks to learn effectively.

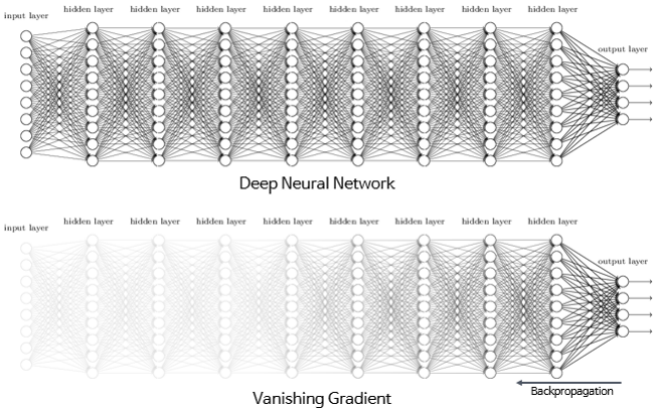


Figure 2. Vanishing gradient illustration in a neural network [1]

ResNet-18, with its **18 layers**, was chosen as the **baseline model** due to its computational efficiency and its ability to learn hierarchical features through **pretrained weights from ImageNet**. [19] The assumption was that its prelearned feature maps would provide a **good starting point** for bird classification.

In its original form, **ResNet-18 was designed for classifying 1,000 ImageNet categories**. To adapt the model for this task, the **final fully connected (FC) layer was modified** to match the number of bird species in the dataset. This modification ensured that the model's outputs corresponded to the correct number of classes. The training images were resized to **224×224 pixels**, a standard input size for ResNet-based models, and normalized using ImageNet mean and standard deviation.

The model was trained using **CrossEntropyLoss** as the loss function and **Adam optimizer** with a learning rate of **0.001**. Data augmentation techniques such as **random horizontal flipping and random rotation** were applied to introduce variability and improve generalization.

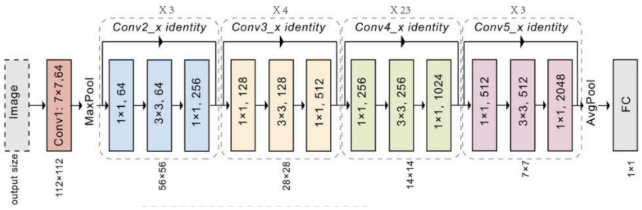


Figure 3. An example of a Resnet18 architecture [16]

Metric	Value
Model	ResNet-18 (Baseline)
Train Loss	0.3258
Validation Loss	1.0388
Validation Accuracy (%)	64.08

Table 2. Performance metrics of ResNet-18 (Baseline) model.

Accodrdiong to the Table 2. above, after training, ResNet-18 achieved a **validation accuracy of 64.08%**, with a validation loss of **1.0388**. Although this provided a reasonable starting point, the model exhibited clear **overfitting**, as evidenced by the significant gap between training and validation losses. The inability of ResNet-18 to capture fine-grained differences between bird species suggested that **a more powerful feature extractor was necessary**.

2.3. Improving ResNet-18

To address the limitations observed in the baseline ResNet-18 model, several improvements were introduced. First, **dropout regularization (0.5)** was added to the **final classification layer** to reduce overfitting by preventing neurons from relying too heavily on specific patterns. Additionally, **learning rate scheduling (ReduceLROnPlateau)** was incorporated to **adapt the learning rate dynamically**, helping the model stabilize and improve generalization.

To further enhance variability in training, the data augmentation pipeline was expanded to include **color jitter (brightness, contrast, and saturation adjustments)** and **random affine transformations**. These transformations aimed to introduce more natural variations in images, allowing the model to learn robust features that generalize better to unseen data.

Despite these enhancements, the **improved ResNet-18 model performed worse**, with a **validation accuracy of only 58.25%** and a validation loss of **1.9238**. The addition of regularization appeared to have **over-restricted the model's learning capacity**, resulting in **underfitting**. These results confirmed that **ResNet-18 was fundamentally insufficient for this classification task**, and a more advanced architecture was required.

Metric	Value
Model	ResNet-18 (Improved)
Train Loss	0.6923
Validation Loss	1.9238
Validation Accuracy (%)	58.25

Table 3. Performance metrics of ResNet-18 (Improved) model.

2.4. EfficientNet-B0

Recognizing the limitations of ResNet-18, the next approach was to explore **EfficientNet**,

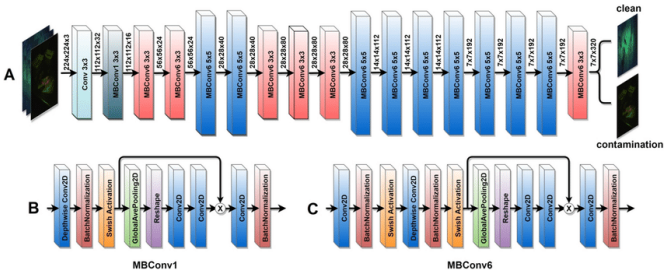


Figure 4. A concise representation of the EfficientNet-B0 model [15]

a family of architectures designed to balance **depth, width, and resolution** efficiently. [12] EfficientNet-B0 was chosen as the next candidate because of its **higher accuracy-to-parameter ratio**, making it ideal for **fine-grained classification tasks** like bird species identification.

EfficientNet introduces **compound scaling**, which optimally expands the network's depth (number of layers), width (number of channels per layer), and resolution (input image size) in a balanced manner. This enables it to **achieve higher accuracy with fewer parameters** compared to traditional CNN architectures. [14]

For this model, the **fully connected classification layer was replaced** to match the number of bird species in the dataset. To prevent overfitting, **weight decay (1e-4) was applied**, encouraging better generalization. The optimizer remained **Adam**, but additional **data augmentations, including random affine transformations and color jitter**, were introduced.

EfficientNet-B0 **significantly outperformed ResNet-18**, achieving a **validation accuracy of 86.41%**, with a **low validation loss of 0.2925**. The improvement was drastic, confirming that **EfficientNet-B0 was well-suited for this dataset**. Unlike ResNet-18, this model was capable of extracting fine-grained details from the images, leading to better classification performance.

Metric	Value
Model	EfficientNet-B0
Train Loss	0.0397
Validation Loss	0.2925
Validation Accuracy (%)	86.41

Table 4. Performance metrics of EfficientNet-B0 model.

2.5. EfficientNet-B3

Given the success of EfficientNet-B0, the next logical step was to experiment with **EfficientNet-B3**, a deeper and more powerful version of the model. EfficientNet-B3 features **more layers and parameters**, allowing for **better feature extraction** while maintaining computational efficiency. [8]

To further enhance learning, **class weights were computed and incorporated into the loss function** to address class imbalance, ensuring that the model learned equally from all species. Additionally, **label smoothing (0.1)** was introduced to prevent the model from becoming overconfident in its predictions.

EfficientNet-B3 yielded the **highest accuracy of all tested models, reaching 87.38% validation accuracy**, with a **more stable training process compared to EfficientNet-B0**. This confirmed that **EfficientNet-B3 was the most effective model in the initial phase of development**.

Metric	Value
Model	EfficientNet-B3
Train Loss	0.6874
Validation Loss	0.8975
Validation Accuracy (%)	87.38

Table 5. Performance metrics of EfficientNet-B3 model.

The experiments conducted in this phase demonstrated the **importance of selecting an architecture well-suited for fine-grained classification tasks**. ResNet-18, despite being a strong general-purpose model, proved to be **insufficient**, while EfficientNet-B0 brought a **major breakthrough in performance**. The further refinement with EfficientNet-B3 led to **the best accuracy achieved in this phase (87.38%)**.

Future work should explore **EfficientNet-B4/B5 and Vision Transformers (ViT-B16, Swin-Tiny)** to push accuracy even further.

3. Advanced Techniques for Model Improvement

3.1. Visual Transformers (ViT)

Visual Transformers (ViTs) are a class of deep learning models that apply the Transformer architecture to computer vision tasks (originally designed for natural language processing (NLP)).

Unlike Convolutional Neural Networks (CNNs) that rely on convolutional filters to extract local features from images, ViTs treat an image as a sequence of patches and apply self-attention mechanisms to capture global relationships across different parts of the image.

The main idea behind ViTs is to use **self-attention** to learn visual representations, allowing them to be more flexible and scalable compared to traditional CNNs.

Self-attention: Self-attention is widely used in computer vision to model long-range dependencies and improve feature representations. Unlike convolutional neural networks (CNNs), which rely on local features, self-attention enables a global view of the image, making it more effective for complex vision tasks like classification, segmentation, and object detection.

3.2. Architecture

Instead of using convolutional layers like CNNs, ViT splits an input image into small patches. Each patch is then flattened into a 1D vector (similar to how words are tokenized in NLP). These patches are projected linearly into a higher-dimensional embedding space.

Since Transformers do not have a built-in spatial understanding like CNNs, **positional embeddings** are added to the patch embeddings. This helps the model retain spatial relationships between patches.

A special **[CLS] token** (similar to BERT) is added to represent the entire image.

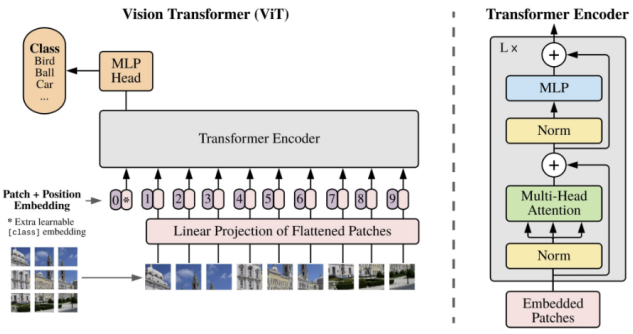


Figure 5. Vision Transformer (ViT) Architecture [2]

All embedded patches are passed into a **Transformer**

Encoder, which consists of multiple layers. Each layer contains:

- **Multi-Head Self-Attention (MHSA):** Each patch attends to all other patches in the image. This helps capture long-range dependencies between different regions.
- **Layer Normalization (Norm):** Helps stabilize training.
- **MLP (Feedforward Network):** Processes the attended features.
- **Skip (Residual) Connections:** Improves gradient flow and stabilizes training.

3.3. Object Detection Task

In this study, we implemented a Vision Transformer (ViT) for image classification. However, instead of feeding entire images directly into the ViT model, we first applied object detection as a preprocessing step. This approach was chosen to enhance model efficiency and accuracy, particularly in images containing multiple objects or background noise.

Object detection serves as a crucial preprocessing step for several reasons:

- **Eliminating Unnecessary Background:** Many images contain irrelevant background information that does not contribute to classification. By detecting and cropping objects, we ensure that the model focuses on the most relevant features.
- **Handling Multiple Objects in One Image:** Some test images contain more than one object (e.g., tree, human hand). If we feed the whole image into ViT, it may struggle to associate the correct label. By applying object detection first, we can crop the image around the object of interest (birds), allowing ViT to process the bird image only and improve classification accuracy.
- **Reducing Overfitting and Improving Generalization:** If ViT processes full images, it might learn irrelevant correlations. Object detection eliminates such biases by isolating objects, improving generalization to unseen data.

For the object detection task, we used the **DEtection TRansformer (DETR) ResNet-101** model, developed by Facebook AI. Unlike traditional object detection models such as Faster R-CNN, DETR leverages Transformers for direct object detection.

Key aspects of DETR ResNet-101:

- **End-to-End Object Detection:** It takes an image as input and directly predicts objects and their locations (bounding boxes) without using region proposal networks.
- **ResNet-101 Backbone:** ResNet-101 is used as a feature extractor to help the model recognize important parts of the image.

3.4. Label Smoothing

Label smoothing is a regularization technique used in classification tasks to prevent the model from becoming overconfident in its predictions. Instead of assigning a hard 100% probability (1.0) to the correct class, we soften the labels by distributing a small portion of the probability mass to the other classes. This technique helps in improving generalization and reducing overfitting in neural networks.

In a multi-class classification task with a ground truth label y and C total classes, the usual one-hot encoded target is:

- If the correct class is y, its probability is 1.0.
- All other classes have a probability of 0.0.

However, with label smoothing, instead of assigning 100% confidence (1.0) to the correct class, we spread some probability to the other classes using a smoothing factor α. The new probability distribution is given by:

$$\tilde{y}_i = \begin{cases} 1 - \alpha + \frac{\alpha}{C} & \text{if } i = y \text{ (correct class)} \\ \frac{\alpha}{C} & \text{if } i \neq y \text{ (other classes)} \end{cases}$$

Label Smoothing is Helpful in Multi-Class Classification:

- **Prevents Overconfidence:** Without label smoothing, a model may become too confident and assign a probability of 1.0 to a class, which can lead to overfitting. Label smoothing softens predictions, preventing extreme confidence.
- **Improves Generalization:** It reduces the model's reliance on memorizing training data, leading to better performance on unseen test data.

For further details, refer to the original paper by Szegedy et al. [11].

3.5. Using Half Precision for GPU Memory Efficiency and Larger Batch Sizes in Computer Vision

Deep learning models typically use 32-bit floating point (FP32) for computations. However, in half precision (FP16), numbers are stored using 16 bits instead of 32 bits, reducing memory usage by 50%.

This technique is primarily used for two reasons:

- **Reduce GPU Memory Usage:** Switching from FP32 to FP16 cuts memory usage in half, allowing us to fit larger models on limited GPU memory (such as on Kaggle).
- **Increase Batch Size:** Larger batch sizes allow the model to process more images at once, leading to more stable training and faster convergence.

3.6. Mixup Data Augmentation

Mixup is a data augmentation technique where two training images and their labels are blended together to create a new synthetic training example. Instead of training the model on a single clear image with one label, we train it on a mixture of two images and their labels. The model learns to predict "soft labels," making it more robust and less likely to overfit. Given two images x_1 and x_2 with their corresponding one-hot labels y_1 and y_2, Mixup creates a new image-label pair as follows:

- **Create a new image by blending two images:**

$$x' = \lambda x_1 + (1 - \lambda)x_2$$

where λ is a random value sampled from a Beta distribution:

$$\lambda \sim \text{Beta}(\alpha, \alpha)$$

with α typically set between 0.1 and 0.4.

- **Create a new label by blending the labels:**

$$y' = \lambda y_1 + (1 - \lambda)y_2$$

This means that the new label is a weighted combination of the original labels.

Benefits of Mixup:

- **Reduces Overfitting:** Since images and labels are blended, the model can't simply memorize individual samples. This forces it to generalize better to unseen data.
- **Increases Robustness:** Helps the model become more tolerant to image variations.
- **Encourages Linearity in Predictions:** The model learns to make smooth, linear transitions between classes. Instead of making sharp, binary decisions, the model can better handle ambiguous cases.

For further details, refer to the original paper by Müller et al. [10].

3.7. Hyperparameters Used

3.7.1 Data Processing

- **Image Size:** $(224, 224)$ to ensure compatibility with the pre-trained ViT.
- **Data Augmentation Techniques:**
  - `transforms.ColorJitter()`: Introduces brightness/contrast variations $\rightarrow$ Improves robustness.
  - `transforms.RandomHorizontalFlip()`: Helps with symmetry-based variations.
  - `transforms.RandomRotation(90)`: Allows the model to recognize objects at different angles.
  - `transforms.RandomZoomOut(fill=0, side_range=(2.0, 4.0))`: Helps the model handle different scales of objects.

3.7.2 Training Hyperparameters

- **Batch Size:** 64
- **Number of Epochs:** 100
- **Max Learning Rate:** 1×10^{-5} (to prevent catastrophic forgetting)
- **Optimizer:**

  optim.Adam(model.parameters(), lr=max_lr, eps=1e-4)

- **Learning Rate Scheduler:** We used the OneCycleLR, a dynamic learning rate adjustment technique that follows the **1-cycle policy**, consisting of:
  - **Warm-up Phase:** Gradually increases the learning rate from a lower bound to the peak value (`max_lr`).
  - **Decay Phase:** Smoothly decreases the learning rate back to a minimum.
  - **Final Annealing:** Optionally reduces the learning rate further near the end of training to improve convergence.
- **Loss Function:**

  $\mathcal{L} = \alpha \cdot \text{soft\_cross\_entropy} + (1-\alpha) \cdot \text{CrossEntropyLoss}$

  where α represents the **degree of smoothing**. A higher α increases the impact of label smoothing, reducing overconfidence in predictions by assigning small probabilities to non-ground-truth classes. Conversely, when $\alpha = 0$, the loss function behaves as standard Cross Entropy Loss.

3.8. Results

3.8.1 Approach and Data Preprocessing

Table 6 presents the validation and test accuracy for different approaches, including the use of Vision Transformer (ViT), object detection, label smoothing, and Mixup.

Approach	Val Accuracy	Test Accuracy
A1	0.95	0.80
A2	0.98	0.90
A3	0.96 (very stable)	0.86

Table 6. Comparison of Different Approaches on Validation and Test Accuracy

Legend:

- **A1**: Using ViT on original images + data augmentation.
- **A2**: Perform object detection task on images, use ViT on cropped images and label smoothing trick.
- **A3**: Perform object detection task on images, use ViT on cropped images and apply label smoothing + Mixup tricks.

3.8.2 Interpretation

Here, we present the evolution of validation loss for three different training approaches:

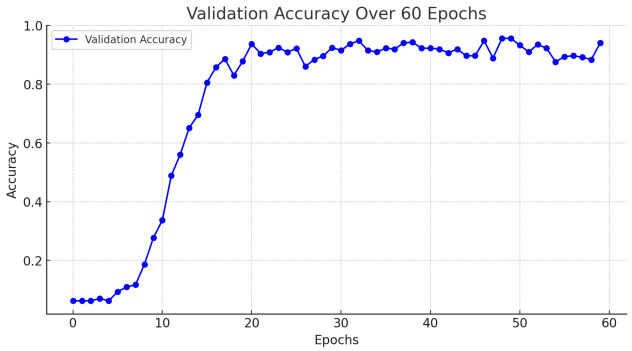
- Training Vision Transformer (ViT) on original images with label smoothing.
- Training Vision Transformer (ViT) on cropped bird images with label smoothing.
- Training Vision Transformer (ViT) on cropped bird images with label smoothing and Mixup.

These results emphasize the impact of different preprocessing techniques on the model's performance.

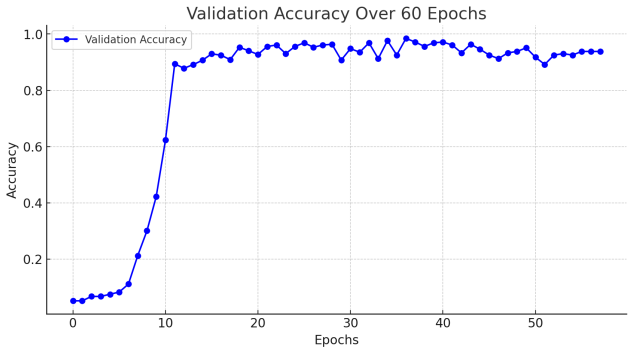
The results show that cropping images before training significantly improves accuracy, as seen in A2 (ViT on cropped bird images with label smoothing), which achieves the highest validation (0.98) and test accuracy (0.90). This suggests that removing background noise allows the model to focus on relevant features, leading to better classification. In contrast, A1 (ViT on full images with label smoothing) struggles slightly more, with lower validation (0.95) and test accuracy (0.80), likely due to the presence of distracting background elements.

Interestingly, A3 (ViT on cropped images with label smoothing and Mixup) results in a more stable validation curve but a lower test accuracy (0.86) compared to A2. This suggests that while Mixup helps stabilize training, it does not necessarily improve generalization in this case. The slightly lower test accuracy could indicate that Mixup makes the model learn more blended representations, which may not always translate well to real-world unseen data.

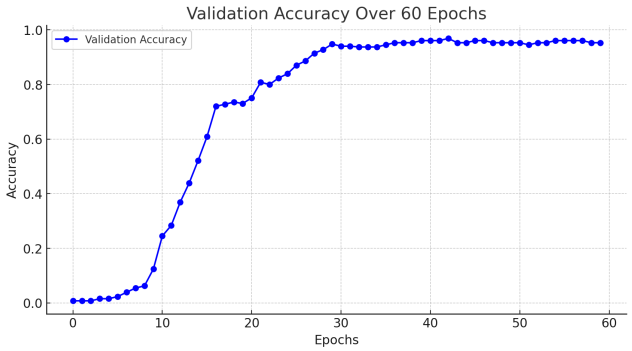
Overall, the results indicate that cropping images is the most effective preprocessing step for improving accuracy, while Mixup provides stability but might not always help with generalization.



(a) ViT on original images with label smoothing



(b) ViT on cropped bird images with label smoothing



(c) ViT on cropped bird images with label smoothing and Mixup

Figure 6. Comparison of validation loss evolution for different training approaches.

4. Other approaches

This section provides a comprehensive examination of additional models explored in subsequent phases of the project. It details the rationale behind the selection of each architecture, including the **Inception V3, VAE-based classifier, and VGG16** models, and describes the specific modifications implemented to enhance their performance. Each model's training methodology and the results obtained are thoroughly analyzed.

4.1. Inception V3 Model

The Inception V3 model [5] is a sophisticated convolutional neural network from the Inception family known for achieving high accuracy with relatively low computational cost. It features multiple-sized convolutional filters within each block, which allows it to capture a wide range of features at various scales and efficiently adapt to the complexity of the dataset. The model incorporates advanced techniques such as factorized convolutions and batch normalization to enhance training speed and overall performance.

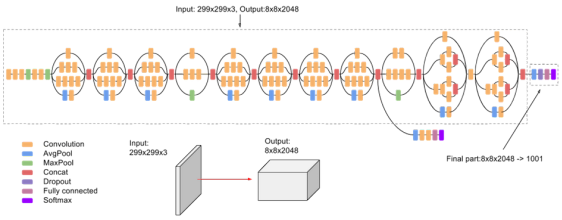


Figure 7. Inception V3 Architecture

4.1.1 Hyperparameters and Training Steps for Inception V3

The Inception V3 model was configured and trained using the following key settings and hyperparameters:

- **Pre-trained Weights:** Initialized with weights pre-trained on ImageNet to leverage prior learning for enhanced feature extraction.
- **Classifier Modification:** The final fully connected layer has been modified to classify images into 20 distinct bird species, tailored to the specific requirements of our dataset.
- **Optimizer:** Adam optimizer with a learning rate of 0.001, chosen for its effectiveness in achieving fast convergence and stability in training.
- **Loss Function:** CrossEntropyLoss, ideal for handling multi-class classification problems.
- **Batch Size:** Set at 32 to optimize the balance between resource utilization and model performance.
- **Epochs:** The model was trained over 100 epochs to ensure sufficient learning without overfitting.

4.1.2 Results

The best obtained accuracy on the validation set was:

Metric	Value
Model	Inception V3
Train Loss	0.4316
Validation Loss	0.6310
Validation Accuracy (%)	84.46

Table 7. Performance metrics of Inception V3 model.

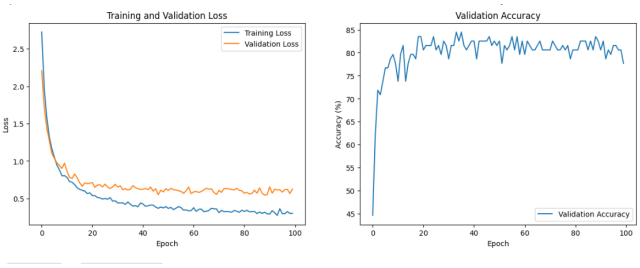


Figure 8. Inception V3 Training and Validation Results

4.1.3 Interpretation

The training process for the Inception V3 model demonstrated consistent and significant improvements in performance metrics:

- **Training Loss:** Starting at 2.705 and reducing to 0.290 by the 100th epoch, showing effective learning and adaptation by the model.
- **Validation Loss:** Exhibited a decreasing trend from 2.293 to 0.580, indicative of the model's increasing ability to generalize well to unseen data.
- **Validation Accuracy:** Began at 53.4% and impressively rose to a peak of 83.5%, although there were slight fluctuations, it stabilized around 80-82%, confirming the model's robustness.

4.2. A Enhanced VAE-based Classifier

The VAE-based classifier [3] utilizes a Variational Autoencoder as a feature extractor, enhancing the classification of bird species. By leveraging the VAE's encoder, im-

ages are transformed into a compact and expressive lower-dimensional latent space. This approach exploits the unsupervised learning capabilities of the VAE to improve the generalization of the classifier in a supervised setting.

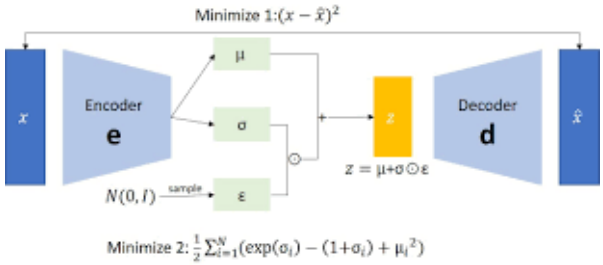


Figure 9. VAE-based Classifier Architecture

4.2.1 Hyperparameters and Model Architecture Details

VAE Model Configuration

- **Latent Dimension:** 128, dictating the size of the latent space.
- **Hidden Layers in Encoder:** Includes layers with 32, 64, 128, 256, 512, 512 filters, with strides of 2 and kernel size of 3.
- **Activation Functions:** LeakyReLU, enhancing non-linear processing capabilities.

Classifier Configuration

- **Neural Network Architecture:** Consists of two fully connected layers. The first layer maps the latent representation to 512 nodes, followed by a ReLU activation and dropout (rate: 0.5). The second layer outputs to the number of classes (20 bird species).
- **Output Activation:** Softmax, utilized for multi-class classification.

Training Configuration

- **Optimizer:** Adam with a learning rate of 0.001.
- **Loss Function:** CrossEntropyLoss, suitable for multi-class tasks.
- **Epochs:** 30.
- **Batch Size:** 32.

Metric	Value
Model	Vae Classifier
Train Loss	0.9913
Validation Loss	1.8595
Validation Accuracy (%)	53.40

Table 8. Performance metrics of Vae based classifier model.

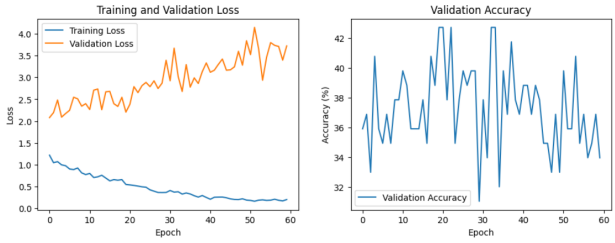


Figure 10. Training and Validation Results of the VAE-based Classifier

4.2.2 Results

4.2.3 Interpretation

The VAE-based classifier demonstrated continuous improvement in validation accuracy, peaking at 53.40% after 60 epochs. Despite this progress, it underperformed compared to other methodologies, indicating challenges in effective generalization. This suggests a potential need for further model refinement or exploration of alternative architectures to enhance performance.

5. Conclusion

This research embarked on a comprehensive exploration of deep learning architectures for fine-grained bird species classification, systematically advancing from traditional Convolutional Neural Networks to more sophisticated models. Our journey revealed critical insights into the challenges and potential solutions in complex image classification tasks.

5.1. Key Findings

The progression of model architectures demonstrated the critical importance of network design in computer vision:

- **Traditional CNNs** proved insufficient for capturing the nuanced features distinguishing bird species, with initial models achieving validation accuracies below 50%.
- **ResNet-18**, while an improvement, still struggled to generalize effectively, highlighting the limitations of traditional residual networks in fine-grained classification.

- **EfficientNet models**, particularly EfficientNet-B3, emerged as the most promising architecture, achieving a remarkable validation accuracy of 87.38%.

5.2. Innovative Preprocessing Techniques

Our research introduced several advanced preprocessing strategies that significantly enhanced model performance:

- **Object Detection Preprocessing:** Utilizing DETR ResNet-101 to isolate and focus on bird-specific features dramatically improved feature extraction.
- **Label Smoothing:** This technique effectively reduced model overconfidence and improved generalization.
- **Mixup Data Augmentation:** Provided a robust method for creating synthetic training examples, enhancing model robustness.

5.3. Limitations

Despite the significant progress, several limitations warrant future investigation:

- The current approach may still struggle with extremely similar bird species or low-quality images.
- Further exploration of more advanced architectures and advanced Vision Transformers could potentially push accuracy even higher.

5.4. Concluding Remarks

This research demonstrates the power of sophisticated deep learning architectures and preprocessing techniques in solving complex image classification problems. By systematically exploring and refining our approach, we have developed a robust framework for bird species identification that can be improved upon in the future provided a broader dataset.

References

[1] Md. Amanatulla. Vanishing gradient problem in deep learning: Understanding, intuition, and solutions. Online, Accessed 2025. Available at: https://medium.com/@amanatulla1606/vanishing-gradient-problem-in-deep-learning-understanding-intuition-and-solutions-da90ef4ecb54. 2

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations (ICLR)*, 2021. 4

[3] Anbang Ye et.al. A yolo-based neural network with vae for intelligent garbage detection and classification, 2021. Supplied as supplemental material `tr.pdf`. 8

[4] Jinzhu Lu et.al. Review on convolutional neural network (cnn) applied to plant leaf disease classification. 2021. 1, 2

[5] Xiaoling Xia et.al. Inception-v3 for flower classification. 2017. 8

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

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[8] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Efficientnetv2: Advanced scaling and architectures for vision models. *arXiv preprint arXiv:2104.00298*, 2021. 1, 4

[9] Ze Liu, Hanxiao Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11976–11986, 2022. 1

[10] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. Do we need some distraction to solve hard problems? *arXiv preprint arXiv:1906.02629*, 2019. 6

[11] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon 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, 2016. 5

[12] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6105–6114, 2019. 1, 3

[13] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 10096–10106, 2021. 1

[14] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10781–10790, 2020. 3

[15] Unknown. A concise representation of the efficientnet-b0 model. Online, Accessed 2025. Available at: https://www.researchgate.net/figure/A-A-concise-representation-of-the-EfficientNet-B0-model-B-The-building-blocks-of_fig4_358585113. 3

[16] Unknown. Original resnet-18 architecture. Online, Accessed 2025. Available at: https://www.researchgate.net/figure/Original-ResNet-18-Architecture_fig1_336642248. 3

[17] Imen C. Mohamed A Wadhah A., Wajdi E. Deep cnn for brain tumor classification. 53, 2021. 1

[18] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *Technical Report CNS-TR-2011-001*, 2011. http://www.vision.caltech.edu/visipedia/CUB-200-2011.html. 1

[19] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12, 2016. 2