Team OverfitOwls - BDMA 7 Project: Image Classification Kaggle Competition

Muhammad Qasim Khan, Arijit Samal, Hareem Raza, Simon Coessens
CentraleSupélec
{muhammad-qasim.khan, arijit.samal, hareem.raza, simon.coessens}@student-cs.fr

Abstract

Fine-Grained Visual Classification (FGVC) is challenging due to subtle inter-class differences, occlusions, and background distractions. In this project, we tackle bird species classification using a subset of the Caltech-UCSD Birds-200-2011 dataset. To enhance accuracy, we integrate object detection with Transformer-based classification. YOLOv11X is used to detect and crop bird regions, reducing background interference. For classification, we experimented with CNNs and Vision Transformers, ultimately selecting EVA-02 due to its strong feature extraction capabilities. To further improve robustness, we apply data augmentation techniques, including TrivialAugment, CutMix, MixUp, and Random Erasing. Our final approach achieves a public test set accuracy of 92.5% on Kaggle, which demonstrates the effectiveness of combining object detection, vision transformers, and augmentation strategies for FGVC. Our code is available at github.com/QasimKhan5x/OverfitOwls-Bird-Classification

1. Introduction

Fine-Grained Visual Classification (FGVC) is a challenging problem in computer vision which involves models to distinguish between visually similar subcategories within a broader category [11]. Unlike general image classification tasks, where inter-class differences are usually more noticeable, FGVC involves subtle variations that may be difficult to identify, i.e., there is small inter-class and large intra-class variation. In this project, we focus on the task of bird species classification, a common FGVC problem, where species differ in subtle features like feather textures, beak structures, and color variations. The primary goal of this project is to optimize classification accuracy while adhering to competition constraints, such as limiting the use of external labeled datasets and ensuring computational efficiency

The dataset used in this competition is a subset of the Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset [10], which contains high-resolution images of various bird species .Our dataset is split into training, validation, and test sets, with each split containing a different number of images. Specifically, the dataset consists of 1082 training images, 103 validation images, and 400 test images, across a total of 20 bird species. Each image varies in resolution but typically falls under 500x500 pixels. The dataset presents unique challenges due to variations in lighting conditions, image backgrounds, and bird postures.

One of the primary difficulties in this dataset is the presence of occlusions, where birds are partially obscured by branches, leaves, or other objects, making feature extraction difficult. Additionally, certain species are extremely similar in appearance, with subtle distinctions in feather coloration or size, making classification a non-trivial task. Another significant challenge is the scale of birds in the images—some birds occupy only a small portion of the frame, making it difficult for models to focus on the relevant features. Furthermore, in some cases, birds are camouflaged against complex backgrounds, blending in with trees or water, increasing the risk of misclassification. Figure 1 illustrates some of these challenges with sample images from the dataset.

Figure 1. Examples of challenges in our bird classification dataset. (a) **Occlusion**: The bird is partially obscured by branches, making feature extraction difficult. (b) **Camouflage**: The bird blends into the background, increasing the risk of misclassification. (c) **Size Discrepancy**: The bird appears too small relative to the image, making it hard for the model to focus on relevant details. These challenges impact classification performance and require robust preprocessing techniques.

Given the challenges of fine-grained bird classification, we use a methodology combining object detection, state-of-the-art classification models, and data augmation. Accord-

ingly, this report is structured as follow. Section 2 discusses prior work on fine-grained visual classification (FGVC), focusing on bird species recognition with the Caltech-UCSD Birds-200-2011 (CUB) dataset. It examines deep learning advancements, including Vision Transformers (ViT) [1], EVA-02 [2] , and DINOv2 [6], as well as data augmentation techniques like automatic augmentation [5], CutMix [12], MixUp [13], and random erasing. Section 3 describes the design and architecture of our classification models. Section 4 presents our training methodology, including data preprocessing, augmentation techniques, and optimization strategies. Section 5 involves our experimental setup, evaluation metrics, and key results. Finally, Section 6 summarizes our findings and suggests potential improvements for future work.

2. Related Work

2.1. Fine-Grained Bird Classification and the CUB Dataset

FGVC presents unique challenges due to the high similarity between subcategories, requiring models to capture subtle differences in appearance while remaining robust to variations in lighting, occlusion, and background clutter [7]. Early FGVC approaches relied on hand-crafted features such as color histograms and keypoint descriptors. However, deep learning models, particularly Convolutional Neural Networks (CNNs), have since become the dominant approach. More recently, Transformer-based architectures have demonstrated superior performance in fine-grained classification by capturing long-range dependencies and global context [11].

One of the most widely used datasets for evaluating FGVC models in bird classification is the **Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset** [10], which consists of 11,788 images across 200 bird species. It includes annotations such as bounding boxes and part locations, providing a valuable benchmark for fine-grained recognition [1]. In our project, we utilize a subset of the CUB dataset and refrain from using any additional external birds data or labels.

2.2. Vision Transformers for Image Classification

The adoption of Transformers in computer vision has led to improvements in FGVC performance. As opposed to CNNs such as EfficientNet [8] which rely on local feature extraction, Vision Transformers (ViTs) use a self-attention mechanism to model relationships between distant image regions, making them well-suited for recognizing subtle variations in bird species [9].

ViT models [1], have achieved strong results on large-scale classification tasks but require substantial labeled training data. To overcome this, **DINOv2** [6] applies **self-supervised learning** which enables ViTs to learn robust feature representations without requiring labeled data. This makes them ideal for FGVC tasks where annotations are limited.

Additionally, **EVA-02** is a Transformer-based model that incorporates CLIP pretraining and enhanced feature representations [2]. EVA-02 refines the ViT framework to improve fine-grained recognition while maintaining computational efficiency, demonstrating strong performance on datasets like ImageNet and CUB-200.

Based on the advancements in the literature, we use transformer-based models, self-supervised learning, and advanced augmentation techniques to maximize classification accuracy and improve model robustness in fine-grained bird classification.

3. Model Design

In this section, we describe the design of our image classification model and provide an high-level explanation of its underlying architecture. Then we will go into further detail on our motivation for this model.

Transformers, originally introduced in [9] for NLP tasks, have demonstrated their power by modeling long-range dependencies through attention mechanisms. This success motivated researchers to adapt transformer architectures for computer vision, culminating in the Vision Transformer as presented in [1].

3.1. Standard Vision Transformer Architecture

The Vision Transformer adapts the transformer framework to process images. An input image is represented as

$$\mathbf{X} \in \mathbb{R}^{H \times W \times C},$$

where H, W, and C denote the image height, width, and the number of channels (e.g., $C = 3$ for RGB images), respectively.

To manage computational complexity, the image is partitioned into fixed-size patches (analogous to tokens in NLP) instead of processing every pixel individually. Each patch is then flattened and mapped to a high-dimensional embedding via a learnable linear projection. To preserve the spatial structure of the image, we add a positional embedding to each patch embedding. This process can be summarized as follows:

- **Patch Extraction:** The input image is divided into a grid of patches.
- **Patch Embedding:** Each patch is flattened and passed through a linear projection.

- **Positional Encoding:** A learnable positional embedding is added to each patch embedding to encode spatial information.

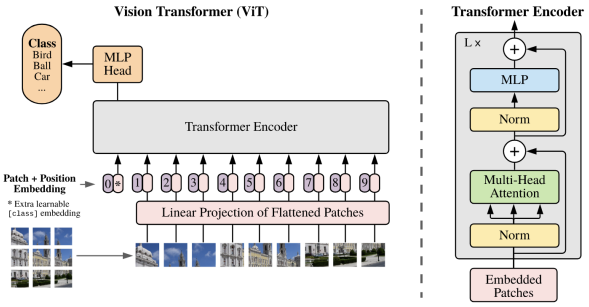


Figure 2. Overview of the Vision Transformer. The image is split into fixed-size patches, linearly embedded, and augmented with positional encodings. A learnable classification token is prepended to the sequence, and the resulting embeddings are processed by a transformer encoder. The illustration is taken from [1]

The heart of the Vision Transformer is the transformer encoder, which leverages the attention mechanism to iteratively refine the patch representations. By repeatedly applying the attention operation, the network captures both local and global relationships among image patches—much like how transformers build up semantic information from word tokens in NLP.

Motivation for Vision Transformer Why does a Vision Transformer model work well for image classification? As discussed in the Introduction, certain images contain birds that appear in only a very small region of the frame, making it essential for the model to focus on these critical areas. Transformer-based models achieve this through their self-attention mechanism, which dynamically emphasizes informative patches while downplaying less relevant background regions.

In contrast to Convolutional Neural Networks (CNNs), which depend on fixed local filters and pooling operations that may miss long-range dependencies, Vision Transformers enable interactions between all image patches. This global reasoning and adaptive feature weighting allow them to capture complex, dispersed features more effectively, leading to improved classification performance.

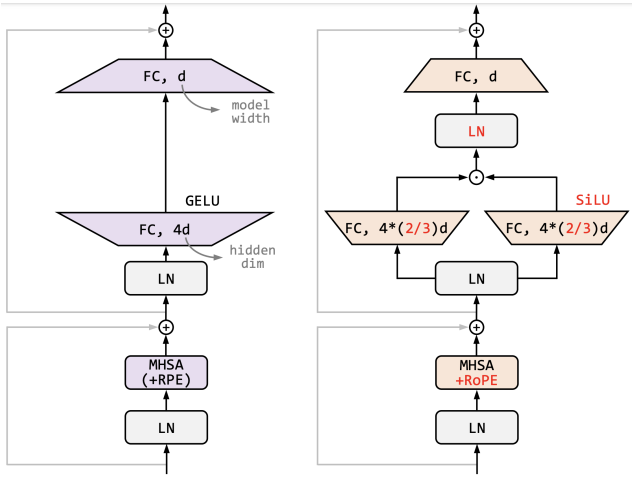
3.2. EVA-02 Architecture

To further enhance the performance of our image classification system, we incorporate the Eva02 architecture—a state-of-the-art variant within the Vision Transformer family. Inspired by [2], EVA-02: A Visual Representation for Neon Genesis presents a next-generation Transformer-based visual representation model that leverages an updated plain Transformer architecture, extensively pre-trained using a giant CLIP vision encoder. This design allows EVA-02 to reconstruct robust, language-aligned vision features via masked image modeling while utilizing significantly fewer parameters and compute budgets compared to prior state-of-the-art approaches. Notably, the EVA-02 variant with only 304M parameters achieves an impressive 90.0% fine-tuning top-1 accuracy on the ImageNet-1K validation set.

The efficacy of the EVA-02 architecture is underscored by its competitive performance on benchmark datasets. For example, according to the `timm` (PyTorch Image Models) Leaderboard, the model `eva02_large_patch14_448.mim_m38m_ft_in1k` achieves an average top-1 accuracy of approximately 84.97% (see Table 1).

Model	Img Size	avg_top1 (%)	Param Count (M)
eva02_large_patch14_448.mim_m38m_ft_in1k	448	84.97	305.08
eva_giant_patch14_336.clip_ft_in1k	336	84.89	1013.01
eva_giant_patch14_560.m30m_ft_in22k_in1k	560	83.69	1906.76

Table 1. Selected results from the timm Leaderboard.



(a) Vision Transformer (ViT) (b) Transform Vision (TrV)

Figure 3. An illustration of ViT and TrV blocks. TrV builds upon the original plain ViT architecture [41] and includes several enhancements: SwiGLU FFN, sub-LN, 2D RoPE, and xavier normal weight initialization. [2]

In Figure 3 the original ViT architecture is compared to the adapted eva02 architecture.

3.3. Classification Head

The final component of our image classification model is the classification head, which transforms the learned fea-

ture representation into a probability distribution over 20 classes. This head is implemented as a sequence of fully connected layers with intermediate batch normalization, non-linear activation (GELU), and dropout for regularization. The following code snippet illustrates the implementation:

Listing 1. Implementation of the Classification Head

```
model.head = nn.Sequential(
    nn.Linear(1024, 512),
    nn.BatchNorm1d(512),
    nn.GELU(),
    nn.Dropout(p=0.2),
    nn.Linear(512, 256),
    nn.BatchNorm1d(256),
    nn.GELU(),
    nn.Dropout(p=0.2),
    nn.Linear(256, 20)
)
```

In this design, the feature dimension is gradually reduced from 1024 to 20, which corresponds to the number of target classes. The intermediate layers—with batch normalization and GELU activation—help to stabilize training, while dropout layers mitigate overfitting by randomly deactivating neurons during training. The final linear layer produces the logits for each of the 20 classes, thereby completing the model's prediction pipeline.

4. Training and Inference Strategies

In this section, we present the training and inference strategies, along with data augmentation techniques, used to improve model generalization and robustness. Our approach integrates **regularization methods**, **data augmentations**, **training methodologies**, and **inference enhancements** to optimize performance.

4.1. Label Smoothing Regularization

To mitigate overconfidence and improve generalization, we applied label smoothing in our loss function. Label smoothing replaces one-hot labels with a weighted mixture of the true label and a uniform distribution over labels, effectively acting as a regularizer [7].

4.2. Data Augmentation

Mixup and CutMix To improve generalization and robustness, we incorporated *Mixup* and *CutMix* augmentation techniques (illustrated in Figure 4):

- **Mixup**: Generates synthetic training samples by performing convex combinations of image pairs and their labels [13].
- **CutMix**: Pastes a randomly cropped patch from one image into another, assigning labels proportionally to the patch area [12].

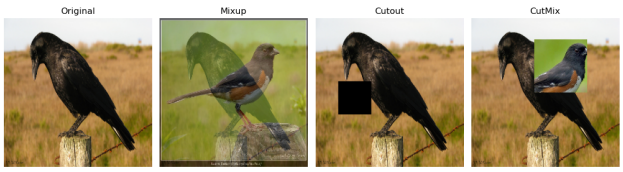


Figure 4. Examples of Mixup and CutMix data augmentations.

TrivialAugment Wide To introduce diverse augmentations without extensive tuning, we employed *TrivialAugment Wide* [4]. This method applies a set of simple transformations randomly.

Random Erasing To simulate occlusions and encourage reliance on less discriminative features, we applied *Random Erasing* [14]. This technique randomly removes rectangular regions from an image and replaces them with either a constant value or random noise. We use a probability of 0.3 for random erasing. Figure 5 shows an example.

Figure 5. Example of Random Erasing augmentation.

4.3. Training Strategies

Learning Rate Warmup During early training, we gradually increase the learning rate from a small value to its target value. This *warmup phase* stabilizes training by allowing adaptive optimizers (e.g., Adam, RMSProp) to compute reliable gradient statistics before making large updates [3]. Warmup also enables the use of larger learning rates, improving convergence.

Layer Freezing and Two-Stage Training When leveraging pre-trained, self-supervised models, it is often advantageous to adopt a two-stage training strategy. In the first stage, the backbone is frozen and only the classifier head is trained. This allows the model to quickly adapt using the robust, pre-learned features. In the second stage, the entire network is fine-tuned to further improve performance.

4.4. Inference Enhancements

Test-Time Augmentation Test-Time Augmentation (TTA) improves prediction robustness by averaging pre-

dictions over multiple augmented variants of each input image. During inference, various transformations (e.g., flips, crops, rotations) are applied to the input image, and the final prediction is obtained by aggregating the outputs.

Utilization of Validation Data for Training In this challenge, we partition the dataset into training and validation subsets, leveraging the validation set for hyperparameter selection. Once optimal hyperparameters are identified, we retrain the model using the entire dataset. The final trained model is then submitted to the Kaggle competition for evaluation on the test set.

4.5. Object Detection-Based Cropping

Object detection was incorporated into this study as a preprocessing step to improve classification performance by isolating the regions of interest containing birds. The dataset consisted of images with complex backgrounds, varying lighting conditions, and multiple objects, making it difficult for classification models to focus solely on the birds. By detecting and cropping the regions where birds were present, we aimed to remove irrelevant information and help the classifier learn better representations. However, object detection introduced challenges such as occlusion, where parts of the birds were hidden behind branches or leaves, and camouflage, where birds blended into their surroundings, making them harder to distinguish.

To improve the robustness of the object detection model and increase the overall dataset size, we applied a range of data augmentation techniques. These included flipping, rotation, shear transformations, grayscale conversion, and hue adjustments. Augmentations helped introduce variations in bird orientations, lighting conditions, and background complexity, allowing the model to generalize better to real-world scenarios. This augmentation strategy significantly enhanced the diversity of the training set, making the model more resilient to occlusion and environmental variations. Moreover, we did not use augmentations like cutmix, mixup, and random erase during the training of the ViT when the cropped images are given as an input because the regions of the image that are required for the classification prediction are already cropped using the best YOLOv11x model.

We manually labeled the dataset using the LabelImg tool to perform object detection, creating precise annotations for all training, validation, and test images. The YOLOv11X model, the most advanced version in the YOLOv11 family, was chosen due to its high accuracy and robust object localization capabilities. YOLOv11X is a large-scale detection model with 56.9 million parameters, integrating transformer-based attention mechanisms and advanced feature fusion techniques. Its architecture allows it to detect small objects with high precision, making it particularly useful for fine-grained tasks such as bird detection.

5. Experiments

5.1. Environment Configuration

For our experiments, we adapted the image classification training script provided by PyTorch[1]. For every model, we first freeze the entire backbone and add a trainable classification head, which we train for 200 epochs. Then, we reinitialize the weights using the best checkpoint of the previous 200 epochs, unfreeze the entire model, and retrain it for an additional 200 epochs. We used the AdamW optimizer with 0.9 momentum. For training only the classification head, the batch size, learning rate, and weight decay were kept at 32, 0.001, and 0.0001, respectively. For fine-tuning the entire model, we decreased these hyperparameters to 8, 0.00001, and 0.00001, respectively. We used a the Cosine Annealing learning rate scheduler and a linear learning rate warmup schedule for 10 epochs. We center cropped all images to 448x448 because we can benefit from a larger resolution of images since the birds are small and the most of the images are 500x500 in the dataset. Training only the classification head takes 30 minutes, whereas training the entire model takes 1 hour.

5.2. EfficientNet

The Efficient CNN model, despite its popularity for its computational efficiency, performed very poorly in our bird classification experiments. As evidenced in Table 2, even after extensive fine-tuning, the model achieved low training accuracy and failed to generalize effectively to the challenging visual features of our dataset. Its architectural design, which relies solely on convolutional operations, is inherently limited when it comes to capturing global contextual information and fine-grained details—critical aspects for distinguishing between visually similar bird species.

The primary shortcoming of the Efficient CNN lies in its restricted receptive field, which hinders its ability to learn and integrate long-range dependencies across an image. This limitation is particularly detrimental for bird classification tasks where subtle variations in texture, shape, and color are crucial for accurate species identification. Furthermore, the lack of advanced attention mechanisms means that the model struggles to focus on the most relevant regions, even when provided with high-quality cropped images from the object detection step. As a result, the training accuracy remains quite low, reflecting the model's inability to effectively extract and utilize discriminative features from the complex backgrounds.

In contrast to more sophisticated architectures like EVA02 and DINOv2, which incorporate hybrid designs and

[1] https://github.com/pytorch/vision/tree/main/references/classification

Model	Method	Train Accuracy	Valid Accuracy	Kaggle Accuracy
EfficientNet v2	Finetuned	92.10	87.50	78.50
VIT L/16	Pretrained	95.65	91.26	78.0
VIT L/16	Finetuned	98.61	95.15	78.50
VIT L/14	DINOv2 SSL and Finetuned	95.54	96.14	89.0
VIT L/14	DINOv2 SSL and Finetuned after OD	99.80	97.50	90.50
EVA02 L/14	Finetuned	100.0	98.06	92.00
EVA02 L/14	**Finetuned after OD**	**100.0**	**99.02**	**92.50**

Table 2. Train and validation accuracies along with Kaggle Accuracy (50% of test set) for all model and method combinations used in the experimentation. The best-performing model, determined by the highest Kaggle score, train accuracy, and validation accuracy, is highlighted in **bold**. *Note:* SSL refers to Self-Supervised Learning and OD refers to Object Detection.

refined attention mechanisms to better handle such complexities, the Efficient CNN's straightforward design proves to be a significant demerit. Its failure to generalize and adapt to the nuances of our bird classification task confirms that while it may offer advantages in computational speed, it is not a suitable model for scenarios demanding detailed feature extraction and robust performance.

5.3. Vision Transformers

5.3.1 VIT L/16

ViT L/16, the large vision transformer from Google, served as the baseline model for using ViT for bird classification. Although this model has proven effective on several standard benchmarks, its relatively basic architecture did not suffice for the complexities of our dataset. Relying solely on standard transformer layers, ViT L/16 lacks the advanced hybrid features and enhanced attention mechanisms that are critical for capturing fine-grained details in challenging visual environments. Even after applying extensive fine-tuning and leveraging high-quality cropped images from the object detection stage, the model struggled to extract the subtle texture differences and intricate local patterns necessary for accurately distinguishing between visually similar bird species.

As indicated in Table 2, the performance of ViT L/16 was noticeably lower compared to more advanced models such as EVA02 and DINOv2. Its straightforward design proved less effective at handling conditions like occlusion, varying lighting, and complex backgrounds, which are prevalent in our bird dataset. The insights gained from using ViT L/16 as our starting point highlighted the limitations of a basic vision transformer architecture and underscored the need for models with enhanced feature extraction capabilities. This realization guided our subsequent exploration of more sophisticated architectures.

5.3.2 DINOV2

The DINOv2 method has demonstrated robust performance in our bird classification task, delivering competitive results that underscore its strong ability to extract high-level semantic features from complex images. Its self-supervised pre-training strategy enables the model to learn rich and transferable representations from large-scale data, allowing it to effectively capture subtle visual cues such as variations in texture, color, and structure.

Despite these strengths, DINOv2 does not perform as well as EVA02 in our experiments. One of the key differences lies in the architectural design. While the VIT L/14 used with DINOv2 excels at learning global representations, its architecture does not incorporate the hybrid integration of transformer and convolutional operations that characterizes EVA02. This hybrid approach in EVA02 enables more effective extraction of fine-grained local details—details that are critical for distinguishing between visually similar bird species. Moreover, although DINOv2 utilizes powerful attention mechanisms, these are not as finely tuned for the specific challenges of our dataset, such as handling varying lighting conditions, occlusions, and subtle texture differences, as those implemented in EVA02.

Its ability to generalize well across diverse image conditions and learn discriminative features from unlabeled data makes it a valuable model, as reflected in the comparable accuracies documented in Table 2. The focused inputs from the object detection model have been instrumental in increasing the DINOv2's performance as shown in the table.

5.3.3 EVA02

EVA02 not only benefits from the refined inputs provided by the object detection step but also capitalizes on its own architectural advantages to deliver consistently high classification scores, as evidenced in Table 2. Its ability to scale up to a large number of parameters further allows it to learn complex, discriminative features that set it apart from other vision transformers.

The training process for EVA02 was carried out in two stages: initially by training the classification head and subsequently by fine-tuning the entire network. Figures 6 and 7 show the evolution of both loss and accuracy during these stages. The first figure demonstrates a smooth convergence of training and validation losses with a steady increase in accuracy and minimal generalization gap, indicating that the initial feature extraction layers are well-calibrated and not overfitting. In the second figure, the continued decline in loss and further improvement in accuracy during full model training underscore the effectiveness of our training strategy and the model's strong generalization capabilities.

Additional insights into the model's performance are provided by the confusion matrices in Figures 8 and 9. The training set confusion matrix reveals that EVA02 correctly classifies all bird species, while the validation set matrix, despite a few minor misclassifications, confirms the robust performance and high reliability of the model. Together, these visualizations reinforce the conclusion that EVA02's advanced and uniquely tailored architecture, combined with focused, high-quality inputs from the object detection stage, leads to superior accuracy and performance in our bird classification task.

Figure 6. Loss and accuracy curves during training of the classification head.

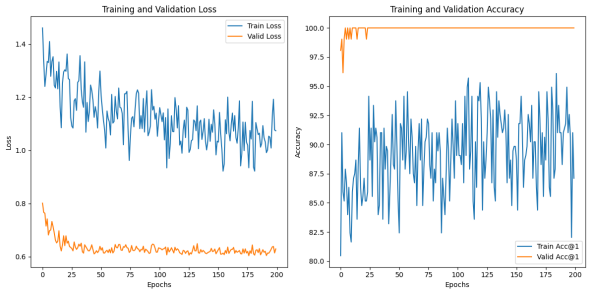


Figure 7. Loss and accuracy curves during full model training.

To better understand why the EVA02 model delivered the best results, we analyzed its attention heatmaps using various images. This allowed us to interpret which parts of the input the model prioritized for image classification.

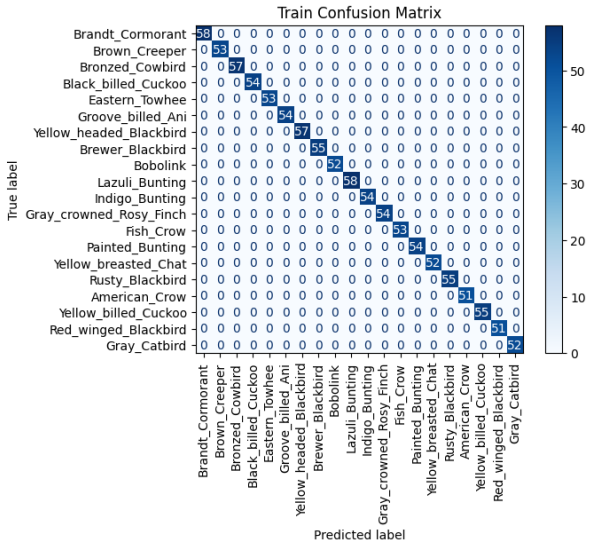


Figure 8. Confusion matrix for the training set.

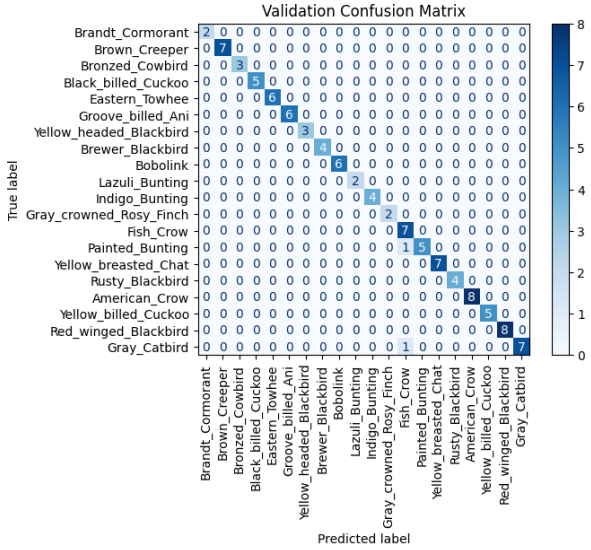
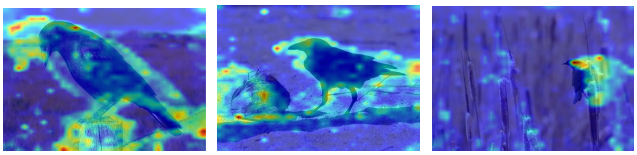


Figure 9. Confusion matrix for the validation set.

Specifically, we multiplied the `Q` and `K` matrices from the 5th block of the ViT layers, averaged the attention weights across all heads, and extracted the attention corresponding to the `[CLS]` token. The resulting 14×14 attention map was then interpolated to match the original image dimensions. The results are shown in Figure Fig. 10.

The green and red regions in the heatmaps indicate areas where the model assigned higher importance. Across all three images, the heatmaps consistently focus on the bird's body, suggesting that the model relies on the bird's body structure as a key feature to distinguish image classes in its 5th layer.



(a) American Crow (b) Fish Crow (c) Test Set Image

Figure 10. EVA02 Attention Heatmaps

5.4. Object Detection Results

During training, we monitored the performance using multiple loss functions: Box Loss, Class Loss, and Distribution Focal Loss (DFL). Box Loss measures the difference between the predicted and ground-truth bounding box coordinates, ensuring that detected regions closely match the actual bird locations. Class Loss evaluates how well the model classifies detected objects into the correct category, while DFL helps refine the localization of bounding box edges by improving the probability distribution of anchor points. The training loss curves, shown in Fig. 11, illustrate the convergence of these loss components over multiple epochs. A steady decrease in loss values indicates that the model was learning effectively, though small fluctuations suggest challenges in detecting birds under certain conditions.

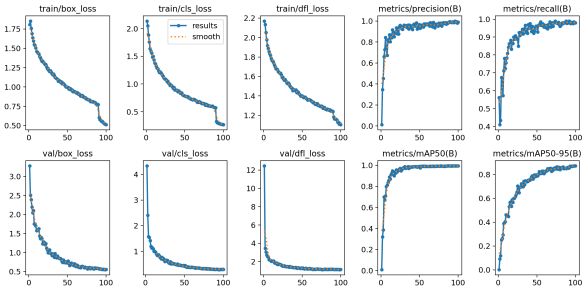


Figure 11. Loss. precision, recall and mAP curves for the YOLOv11X model

To evaluate the performance of the object detection model, we used the Mean Average Precision (mAP) metric. mAP is a widely used evaluation metric in object detection that measures the precision-recall trade-off by calculating the area under the Precision-Recall (PR) curve. In our case, the model achieved an mAP@0.5 score of 99.5, which indicates that the detected bounding boxes closely matched the ground-truth labels with high confidence. The PR curve, presented in Fig. 12, demonstrates the relationship between precision and recall, where a higher area under the curve signifies better detection performance. High precision ensures that the detected objects are indeed birds, while high recall ensures that most birds present in the image are detected. However, in some instances, the confidence scores for detected birds were low, leading to incorrect bounding boxes.

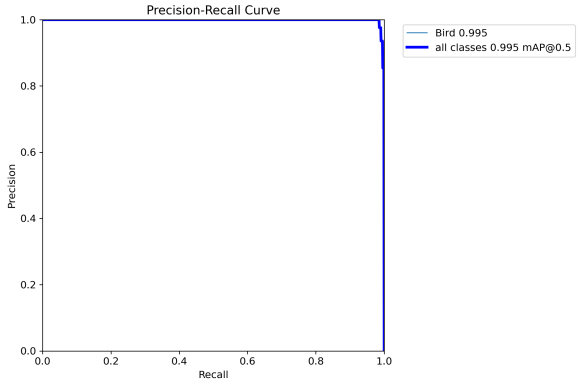


Figure 12. Precision-Recall curve for the YOLOv11X model

Fig. 13 illustrates some sample predictions made by the YOLOv11X model, showing bounding boxes around detected birds.

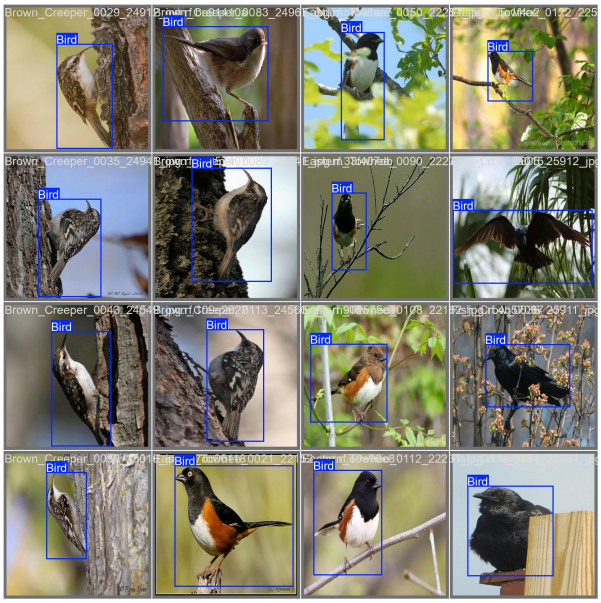


Figure 13. Sample images with predicted bounding boxes using the best YOLOv11X weights

Using the trained YOLOv11X model, we extracted the predicted bounding boxes with the highest confidence scores to crop the bird regions from the original images. These cropped images were then resized to match the input dimensions of the classification models. Since models like EVA02 and DINOv2 were trained on specific image sizes, we ensured that all cropped images were resized accordingly before feeding them into the classification network. However, due to low-confidence detections, some bounding boxes did not contain birds, resulting in irrele-

vant crops. Fig. 14 illustrates some examples where the detected bounding box is not the best. Despite these, it performed better than other classification models. Moreover, using the cropped images also fastened the training process of the ViTs and it converged faster to the best results.

Figure 14. Examples of incorrect detections where no bird was present in the predicted BBOX. These errors negatively impacted classification performance.

The cropped images were used to fine-tune two classification models: ViT L/14 (DINOv2) and EVA02. The results of these classification experiments are summarized in Table 2. The fine-tuned EVA02 model with object detection achieved the highest accuracy, benefiting from the combination of high-quality cropped inputs and its strong feature extraction capabilities.

6. Conclusion

Our final approach combined object detection with transformer-based classification to improve fine-grained bird species recognition. We first applied YOLOv11X to detect and crop bird regions, reducing background noise and focusing classification on relevant features. For classification, we experimented with various architectures, including CNNs and Vision Transformers, ultimately selecting EVA-02 and DINOv2 due to their strong performance. Data augmentation techniques, including TrivialAugment, CutMix, MixUp, and Random Erasing, further improved model robustness. This approach led to a test accuracy of 92.5%, demonstrating the effectiveness of integrating object detection, self-supervised learning, and augmentation for fine-grained classification.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 2, 3

[2] Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *Image and Vision Computing*, page 105171, 2024. 2, 3

[3] Dayal Singh Kalra and Maissam Barkeshli. Why warmup the learning rate? underlying mechanisms and improvements. In *OpenReview*, 2024. Available at https://openreview.net/forum?id=NVl4SAmz5c. 4

[4] Rafael Müller, Jonas Köhler, and Joachim Denzler. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Advances in Neural Information Processing Systems*, 2021. 4

[5] Samuel G. Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 774–782, October 2021. 2

[6] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 2

[7] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. 2, 4

[8] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 2

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. 2

[10] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 1, 2

[11] Xiu-Shen Wei, Yi-Zhe Song, Oisin Mac Aodha, Jianxin Wu, Yuxin Peng, Jinhui Tang, Jian Yang, and Serge J. Belongie. Fine-grained image analysis with deep learning: A survey. *CoRR*, abs/2111.06119, 2021. 1, 2

[12] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable fea-

tures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2, 4

[13] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018. 2, 4

[14] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 4