Report on EfficientNet and DINOv2 Models for Image Classification

Munieb Awad Elsheikhidris Abdelrahman
101233250

1. Methodology

1.1. Dataset

The dataset contains 200 unique bird species, making it a challenging, fine-grained classification task. To handle this complexity, both **transfer learning** and **regularization techniques** were applied to the models.

Data Preparation

All images were resized to 224x224 pixels to ensure compatibility with the input layer of the models. The dataset was organized into structured folders to maintain a clear separation between training and testing data, ensuring no data leakage during evaluation. Below is the folder structure used:

```
train/
  Black_footed_Albatross/
    Black_footed_Albatross_0001.jpg
    Black_footed_Albatross_0002.jpg
  Common_Yellowthroat/
    Common_Yellowthroat_0010.jpg
    Common_Yellowthroat_0011.jpg

test/
  Black_footed_Albatross/
    0000.jpg
    0001.jpg
  Common_Yellowthroat/
    0007.jpg
    0008.jpg
```

This structure ensured that images from each species were properly distributed across the train and test datasets to facilitate accurate performance evaluation.

Data Augmentation (EfficientNet Model Only)

To improve adaptability and avoid specialization, in the training process of the EfficientNet model for image data analysis purposes data augmentation methods were implemented solely on the training images, with transformations applied.

RandomRotation, with a deviation of 15% is used to mimic rotations to accommodate orientations.

RandomFlip() function is used to flip images to add variety and accommodate mirrored perspectives.

These adjustments mimic differences, in the bird pictures to aid the EfficientNet model in performing across different scenarios during assessment purposes. Regarding the DINOV model there haven't been any modifications applied as the transformer design underwent training, on the resized images directly.

1.2. Model Architectures

Two models were used: EfficientNet and DINOv2 Vision Transformer.

EfficientNet

EfficientNet utilizes a convolutional neural network (CNN) architecture with key components to enhance performance and prevent overfitting:

- **Depthwise Separable Convolutions:** Reduces the computational load by splitting filters across dimensions.
- **Squeeze-and-Excitation (SE) Blocks:** Dynamically adjusts feature importance, improving focus on relevant patterns.
- **Dropout (0.2):** Deactivates random neurons during training to combat overfitting.
- **Output Layer:** Uses softmax activation to predict probabilities across 200 classes.
- **Parameters:** Trainable ~2.18M, Total ~4.31M

DINOv2 Vision Transformer

DINOv2 builds on self-attention mechanisms to capture long-range dependencies:

- **Patch Embedding:** Converts the input image into 384-dimensional embeddings through a convolutional layer.
- **Multi-Head Self-Attention:** Computes relationships between patches.
- **SVM with L2 Regularization:** Uses support vector machines for robust classification, improving resistance to overfitting.
- **Layer Normalization and Residual Connections:** Stabilize gradients and enhance convergence during training.

This transformer-based architecture proved flexible, allowing selective fine-tuning of only the last few layers in later iterations.

1.3. Comparison of Methods for Overfitting and Underfitting

Both EfficientNet and DINOv2 incorporate strategies to manage overfitting and underfitting, ensuring better generalization on unseen data.

Similar Techniques:

- **Regularization:** Both models employ methods to minimize overfitting. EfficientNet uses **dropout (0.2)** to deactivate neurons randomly during training, while DINOv2 leverages **SVM with L2 regularization** to constrain complexity, improving stability across diverse classes.
- **Layer Normalization:** In DINOv2, **layer normalization** ensures smooth gradient flows, mirroring the effect of **batch normalization** in EfficientNet. Both techniques stabilize training by regulating parameter updates and preventing underfitting.

Distinct Techniques:

- **Data Augmentation:** EfficientNet applies **RandomRotation** and **RandomFlip** to diversify its training data, enhancing generalization and countering overfitting. In contrast, DINOv2 relies on the **self-attention mechanism** within its transformer layers, which inherently captures complex dependencies without requiring augmentation.
- **Fine-tuning Strategy:** EfficientNet unfreezes its **final 40 layers** during the second iteration, applying **different learning rates** to the dense classification layers and the convolutional layers. This approach refines feature extraction while preventing overfitting in the classifier. DINOv2, on the other hand, achieves fine-tuning by training only the **last 5 layers**, striking a balance between adaptability and model stability.

These combined techniques allow both models to effectively handle the challenges of bird species classification, with strategies tailored to their respective architectures for optimal performance.

1.4. Hyperparameters

EfficientNet:

- **Learning Rate:** 0.0001
- **Batch Size:** 16 (1st iteration), 32 (2nd iteration)
- **Epochs:** 30 (1st iteration), 20 (2nd iteration)

DINOv2 Transformer:

- **Learning Rate:** Consistent initially, lower for the final 5 layers in the second iteration.
- **Classifier:** SVM with gamma = 'scale', L2 regularization.

1.5. Loss Function

Both EfficientNet and DINOv2 use **categorical cross-entropy loss** to measure the difference between predicted and actual class probabilities. This loss function is well-suited for multiclass classification, ensuring that the models learn to assign high probabilities to the correct class while penalizing incorrect predictions.

In EfficientNet, the loss backpropagates through the CNN layers, adjusting both feature extraction and classification weights. DINOv2, using embeddings passed through an **SVM classifier**, minimizes cross-entropy to optimize multi-class predictions. This combination ensures that both models effectively adapt to fine-grained bird species identification, balancing accuracy across complex classes.

2. Results and Discussion

Model	Iteration	Top-1 Accuracy (%)	Top-5 Accuracy (%)	Average Accuracy per Class (%)	Comments
EfficientNet	First	53.73	79.48	39.08	Overfitting observed; validation lagged behind training
	Second	53.73	79.48	39.08	Learning rate adjustments; limited improvement
DINOv2 Transformer	First	82.64	-	82.17	Minor overfitting noticed
	Second	82.64	-	82.18	Stable performance with only 5 layers unfrozen

Table 1: EfficientNet & DINOv2 Models Accuracies

Count of Classes	**85**	**1**	**18**	**13**	**15**	**10**	**13**	**9**	**1**	**8**	**8**	**7**	**1**	**5**	**3**	**1**	**1**	**1**
Unique Accuracies	1	0.875	0.857	0.833	0.8	0.75	0.714	0.667	0.625	0.6	0.571	0.5	0.429	0.4	0.333	0.2	0.167	0.143

Table 2: DINOv2 (2nd iteration): Number of classes that Share the same Accuracy

Total Number of classes with Accuracy of more than 70%	155
Total Number of classes with Accuracy of more than 60%	173
Total Number of classes with Accuracy of more than 50%	188

Table 3: DINOv2 (2nd iteration): number of classes with in a range

2.1. EfficientNet Performance

First Iteration:

- Achieved **53.73% Top-1** and **79.48% Top-5** accuracy, indicating reasonable performance on closely related species.
- Overfitting was observed: Validation accuracy lagged behind, limiting generalization to new data.

Second Iteration:

- Adjusted learning rates for the last 40 layers and the dense output layer, but accuracy improvements were marginal.
- Average class accuracy remained **39.08%**, revealing inconsistency across classes—some species were easily identified, while others proved challenging.

2.2. DINOv2 Transformer Performance

First Iteration:

- Achieved **82.64% Top-1** and **82.17% Average Class Accuracy**, with minor overfitting.
- All layers were trained with identical learning rates, providing good convergence across classes.

Second Iteration:

- By freezing all layers except the last 5, the model maintained **82.64% Top-1 Accuracy**.

- **188 out of 200 classes** achieved over 60% accuracy, showcasing robust performance. The SVM classifier further contributed by maintaining **class balance** through L2 regularization.

2.3. Comparison and Model Selection

The two models, EfficientNet and DINOv2 Vision Transformer, demonstrated distinct performance characteristics across iterations. Below is a summary of their strengths and limitations:

EfficientNet:

- Strengths:
    - **Top-5 Accuracy:** 79.48%, indicating it can effectively identify correct classes within multiple predictions.
    - Benefited from **data augmentation**, enhancing generalization during training.
- Weaknesses:
    - **Overfitting persisted** even after adjustments.
    - Inconsistent performance across classes, with an average accuracy of **39.08%**.

DINOv2 Transformer:

- Strengths:
    - **Consistent Top-1 Accuracy:** 82.64% across iterations, demonstrating superior class prediction precision.
    - **Balanced Performance:** 82.18% average class accuracy, indicating stable generalization across species.
    - **Effective Fine-tuning:** Training only 5 layers while maintaining high performance illustrates architectural efficiency.
- Weaknesses:
    - Higher computational cost due to the **transformer architecture**.

Model Selection

The DINOv2 Vision Transformer is selected as the preferred model for the following reasons:

1. **Higher Top-1 Accuracy:** DINOv2 consistently outperforms EfficientNet with a Top-1 Accuracy of 82.64%, indicating its superior ability to predict the correct class with a single prediction.
2. **More Stable Class-wise Accuracy:** The Average Class Accuracy of 82.18% in the second iteration demonstrates that DINOv2 handles the variability in class distribution better, compared to EfficientNet's 39.08%.
3. **SVM Regularization:** The use of SVM with L2 regularization contributed to the model's stability, ensuring better performance across diverse classes without significant overfitting.
4. **Training Efficiency:** The transformer's ability to generalize well, even when only the last few layers were fine-tuned, highlights the effectiveness of its architectural design, making it better suited for datasets with diverse and fine-grained categories, such as bird species.

In comparison, EfficientNet's lower performance and tendency to overfit, even with data augmentation, suggest that it is less suitable for the specific challenges posed by this dataset. Thus, DINOv2 is recommended for practical bird species identification tasks.

3. Conclusion

This study evaluated the performance of two advanced image classifiers: EfficientNet and DINOv2 Vision Transformer, applied to a dataset of 200 bird species. Both models were explored across two iterations, with adjustments to their architectures and training strategies.

While EfficientNet demonstrated good performance in Top-5 accuracy (79.48%), it struggled with Top-1 accuracy and suffered from overfitting, limiting its generalization capabilities. In contrast, the DINOv2 Vision Transformer consistently achieved 82.64% Top-1 Accuracy and 82.18% Average Class Accuracy, demonstrating stability across iterations. The ability to fine-tune only the last 5 layers while maintaining high performance further highlights the transformer's efficiency.

The DINOv2 model was selected as the preferred solution due to its superior accuracy, class-wise balance, and the benefits derived from SVM regularization. These attributes make DINOv2 a more suitable choice for real-world bird species identification tasks, where precision and consistent performance are critical.

4. References

1. Chutani, T. (n.d.). DINOv2 TensorFlow 2.x Implementation. GitHub. Retrieved from https://github.com/TanyaChutani/DINO_Tf2.x/blob/main/README.md

2. Facebook AI Research. (n.d.). DINOv2: Self-Supervised Vision Transformers. GitHub. Retrieved from https://github.com/facebookresearch/dinov2

3. Meta AI. (n.d.). DINOv2 Vision Transformer Demo. Retrieved from https://dinov2.metademolab.com/

4. Hugging Face. (n.d.). DINOv2 Model Documentation. Retrieved from https://huggingface.co/docs/transformers/en/model_doc/dinov2

5. Keras. (n.d.). EfficientNet API Documentation. Retrieved from https://keras.io/api/applications/efficientnet/

6. Keras. (n.d.). Image Classification with EfficientNet: Fine-Tuning Example. Retrieved from https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/

7. PMC. (n.d.). Advanced Techniques in Neural Networks and Overfitting Solutions. Retrieved from https://pmc.ncbi.nlm.nih.gov/articles/PMC11318802/

5. Links

Hugging Face:

- https://huggingface.co/spaces/MuniebAbdelrahman/Bird-image-classifier-dinov2

GitHub:

- https://github.com/MuniebA/Bird-image-classifier

Presentation Video:

- https://youtu.be/mRzXZnCFQdY
- https://youtu.be/mRzXZnCFQdY?si=NIe5taYYjC1mQCGt