SWINBURNE UNIVERSITY OF TECHNOLOGY
SCHOOL OF SCIENCE, COMPUTING AND ENGINEERING TECHNOLOGIES

COS30082: Applied Machine Learning

Assignment – Bird Species Classification

Thanh Thao Bui
104170172
104170172@student.swin.edu.au

Lecturer
Mr. Minh Hoang

October 20th, 2024

Table of Contents

1 Introduction

Bird species classification is an important domain in ecology and wildlife conservation. To address a multi-class bird classification task with minimal human effort and intervention, this assignment report explores the application of Transfer Learning through feature extraction from pre-trained models, including VGG16, Inception-v3, and ResNet50 with TensorFlow and Keras. The dataset used in this assignment is the Caltech-UCSD Birds 200 (CUB-200) dataset, which comprises 4,829 training images of 200 predominantly North American bird species.

In particular, the training set is relatively small, with each species represented by only 16 to 31 images. Consequently, this necessitates data preprocessing techniques such as data augmentation to expand the training set. Furthermore, due to the limited dataset size, the models are prone to overfitting. To address this challenge and mitigate the associated risks, various techniques are employed, such as data augmentation, early stopping. Subsequently, the performance of three neural network models with softmax output for 200 classes, trained on feature extractions from pre-trained models, is thoroughly evaluated and compared. In this assessment, the evaluation metrics encompass top 1 accuracy, average accuracy per class, as well as a comprehensive classification report. These measures collectively provide a robust framework for analyzing the models' effectiveness in this specific bird classification task.

2 Related Work

2.1 Transfer Learning

Transfer learning is the process of leveraging knowledge from a pre-trained model on a previous dataset to solve a new problem without training a model from scratch. It's based on the idea that lower layers of neural networks learn general features useful for many tasks.

Two main strategies in transfer learning

1. **Feature extraction** consists of the following four steps: (i) taking the layers of a pre-trained model, (ii) freezing these layers to preserve the knowledge gained by the model during training, (iii) add- ing new trainable layers at the top of the frozen layers for adapting existing features to new data, and (iv) training these new layers with the dataset related to the selected task.
2. **Fine-tuning** is a process of unfreezing the entire model and retraining it from scratch with the new data. This iterative training allows the model to adjust its learned features and parameters to align more closely with the specific characteristics of the new task.

When to Use Transfer Learning?

- Insufficient data: When the new task has limited data for effective training.
- Limited computational resources: Reduces requirements for training complex models.
- Improved model quality: Enhances performance, especially with limited or specialized data.
- Quick adaptation: Allows rapid model adjustment for new tasks.
- Domain-specific applications: Particularly useful in fields like medicine or science where data collection is challenging and expensive.

Transfer learning significantly reduces training time and resource requirements while potentially improving model performance on new tasks.

2.2 State-of-Art Models

In this assignment, I have selected start-of-art models which are very well-known for image classification. Due to their high popularity, VGG16, Inception-v3, and ResNet50 models are used to address the bird classification task.

VGG16.
The Visual Geometry Group (VGG) from Oxford developed two convolutional neural network models one of 16 layers and the other of 19 layers. They are widely known as VGG16 and VGG19. VGG16 accepts the input of size 224X224 pixels with 3 channels for RGB, though the input size is fixed here, it can take any value greater than 224. The input image is passed through convolutional layer, the filters here have 3X3 receptive field with

stride 1. Pooling performed here is max-pooling with 2X2 filter and stride 2. The width of convolution layer is 64 in first layer and increases by factor of 2 till it reaches 512. All hidden layers use ReLU as its activation function. The stack of convolutional layer is followed by three fully connected layers. The final layer is the soft-max layer.

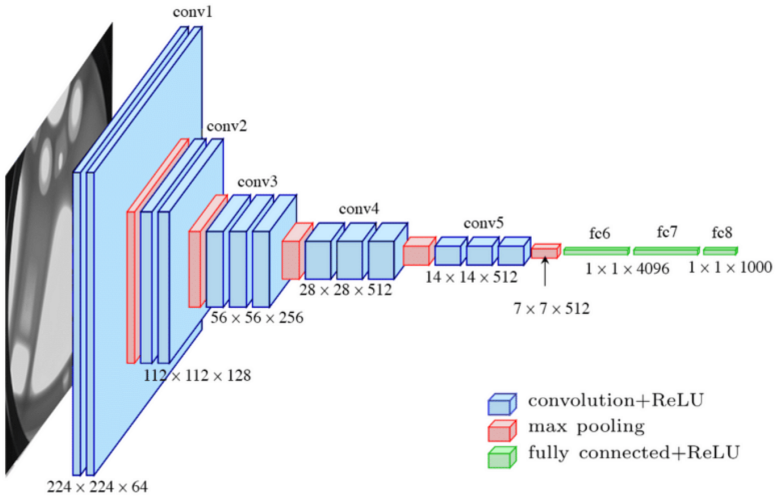

Fig. 1. The architecture of VGG16 model

InceptionV3

The InceptionV3 model is a deep convolutional neural network (CNN) architecture developed by Google researchers for image recognition tasks. The below figure shows the architecture of InceptionV3. It builds upon the original Inception architecture and is known for its high accuracy and computational efficiency. InceptionV3 utilizes inception modules, which incorporate different filter sizes to capture both local and global information while reducing the computational cost. Additional techniques like batch normalization, factorized convolution, and regularization contribute to improved performance and prevent overfitting. InceptionV3 has achieved impressive results in various image recognition challenges, including winning the ImageNet Large Scale Visual Recognition Challenge in 2015. Its balance between accuracy and efficiency has made it a popular choice in computer vision applications. Furthermore, InceptionV3 has influenced subsequent iterations of the Inception architecture, such as InceptionV4 and Inception-ResNet, which have introduced advanced techniques to further enhance performance.The model's impact on deep learning is substantial, inspiring researchers to explore more sophisticated and efficient CNN architectures for image recognition tasks.

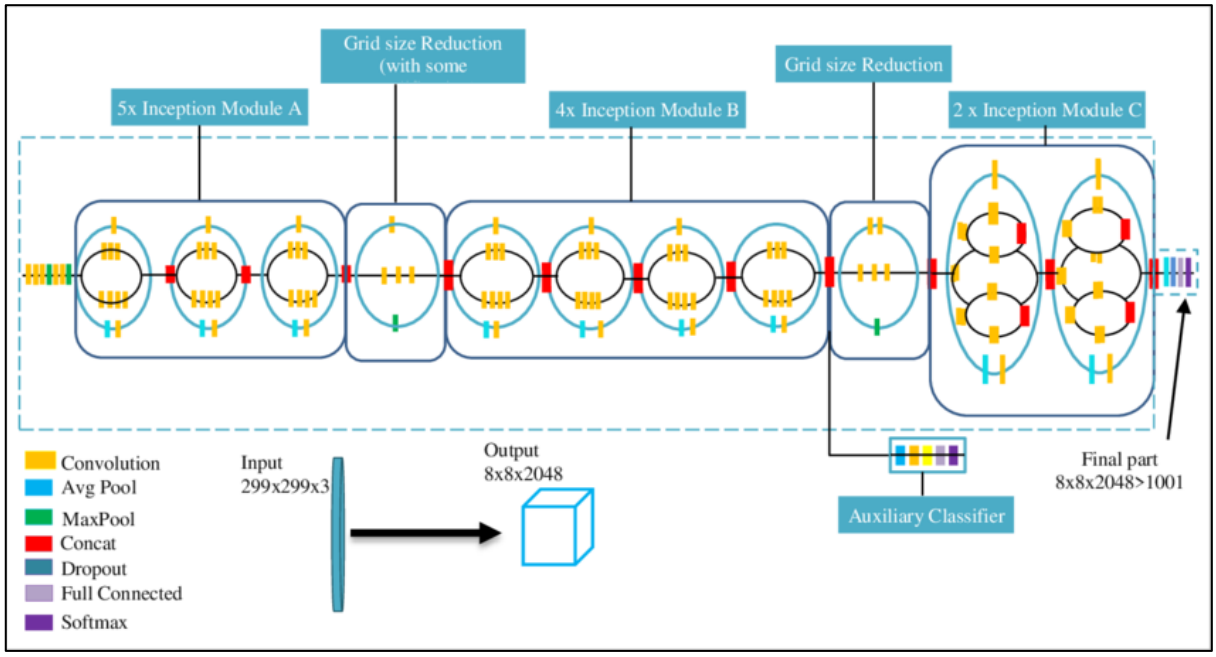

Fig. 2. The architecture of Inception-v3 model

ResNet50
ResNet50 is part of the ResNet (Residual Network) family, first introduced by Microsoft Research in 2015. The architecture was presented in the paper "Deep Residual Learning for Image Recognition" at CVPR 2016, one of the leading conferences in computer vision.

Key features of ResNet50 include:

1. 50-layer deep architecture
2. Utilizes residual blocks and bottleneck blocks to mitigate the vanishing gradient problem
3. Employs skip connections to allow information to bypass layers, enabling training of very deep networks

The ResNet50 architecture consists of:

- An initial convolutional layer
- Four stages of residual blocks
- Global average pooling
- A fully connected layer for final classification

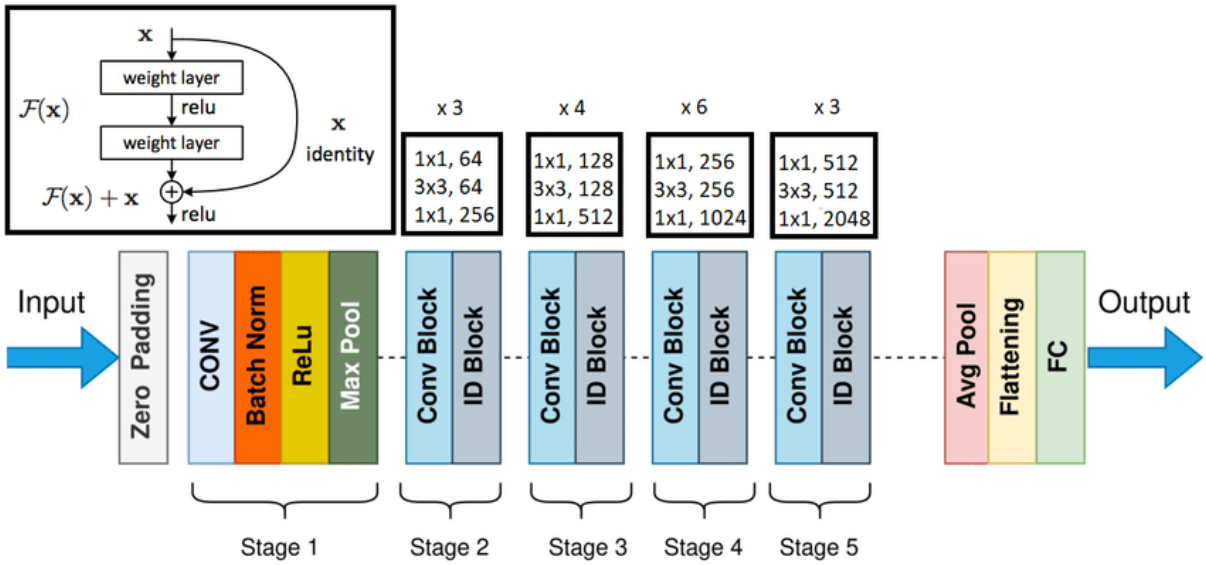

Fig. 3. The architecture of ResNet50 model

ResNet50 improved upon shallower networks like ResNet18 and ResNet34 by increasing depth to 50 layers, allowing for more complex feature representation. Compared to deeper variants like ResNet101 and ResNet152, it offers a good balance between depth and computational efficiency.

3 Methodology

3.1 Dataset

The dataset is used in this assignment is the Caltech-UCSD Bird-200 Dataset, which contains images of 200 bird species primarily from North America. There are 4,829 images available for training and 1,204 images for testing. The average image in training set corresponding to each bird species is 25 images (max: 31 images, min: 16 images). The dataset has 2 folders, including the 'Train' and 'Test' folder which contain images and 2 annotation files, including the 'train.txt' and 'test.txt' files.

Black_footed_Albatross_0019_416160254.jpg 0
Black_footed_Albatross_0005_2755588934.jpg 0
Laysan_Albatross_0014_174432783.jpg 1
Sooty_Albatross_0005_340127050.jpg 2

Fig. 4. Part of contents in train.txt files with 'image's name{space}class label' format

```
dataset/
    train/
        Black_footed_Albatross_0004_2731401028.jpg
        Black_footed_Albatross_0008_1384283201.jpg
        ...
    test/
        0000.jpg
        0001.jpg
        ...
    train.csv
    test.csv
```

Fig. 5. The structure of dataset folder

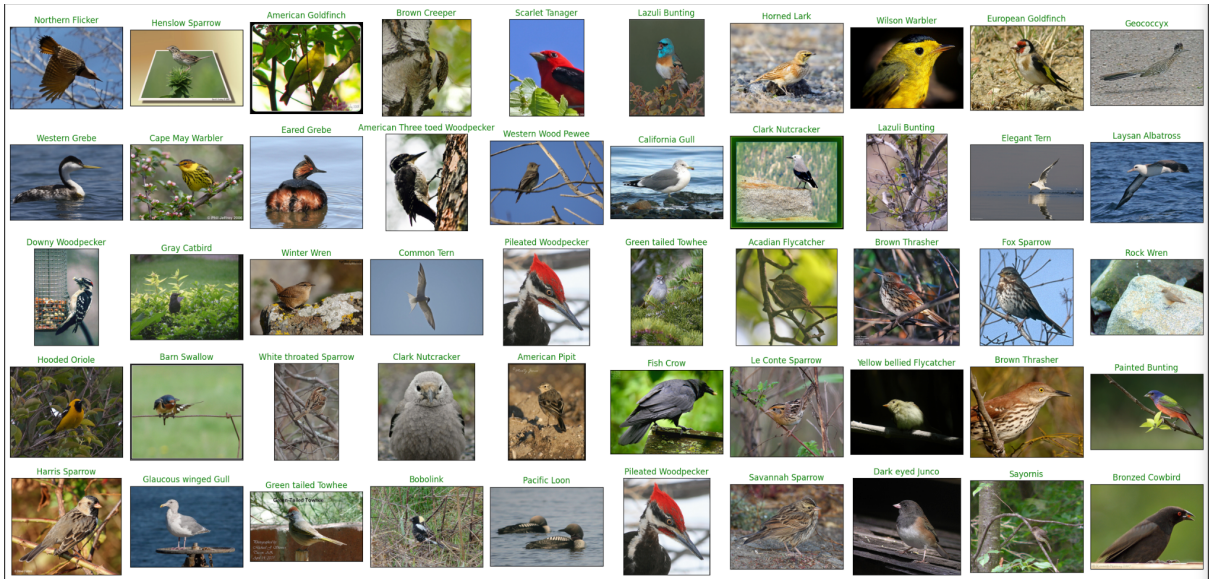

Fig. 6. Some images and their corresponding bird species name in training set

3.2 Transfer Deep Learning Models

Architecture of the model.
In this assignment, transfer learning is applied in the following way: First, pre-trained CNN models are chosen, and new custom layers are added to them. Then, these models are retrained with feature extraction. Finally, these models are compared and the one with the highest accuracy is considered the most effective.

The selected pre-trained three CNN models, namely VGG16, Inception-v3, and ResNet50, are subjected as feature extraction.

Table 1. Details of the pre-trained models used in this assignment.

Pre-trained	Depth	Parameters(Millions)
VGG16	16 layers	138
Inception-v3	48 layers	23.9
ResNet50	50 layers	23.9

The below image presents the architecture and layers of the proposed models for bird classification based on the VGG16, Inception-v3, and ResNet50 network architectures.

VGG16

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14,714,688
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 1950)	48,923,550
batch_normalization (BatchNormalization)	(None, 1950)	7,800
dense_1 (Dense)	(None, 200)	390,200

Total params: 64,036,238 (244.28 MB)
Trainable params: 49,317,650 (188.13 MB)
Non-trainable params: 14,718,588 (56.15 MB)

Inception-v3

Model: "sequential_1"

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 5, 5, 2048)	21,802,784
flatten_1 (Flatten)	(None, 51200)	0
dense_2 (Dense)	(None, 1950)	99,841,950
batch_normalization_95 (BatchNormalization)	(None, 1950)	7,800
dense_3 (Dense)	(None, 200)	390,200

Total params: 122,042,734 (465.56 MB)
Trainable params: 100,236,050 (382.37 MB)
Non-trainable params: 21,806,684 (83.19 MB)

ResNet50

Model: "sequential"

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
flatten (Flatten)	(None, 100352)	0
dense (Dense)	(None, 1950)	195,688,350
batch_normalization (BatchNormalization)	(None, 1950)	7,800
dense_1 (Dense)	(None, 200)	390,200

Total params: 219,674,062 (837.99 MB)
Trainable params: 196,082,450 (748.00 MB)
Non-trainable params: 23,591,612 (89.99 MB)

Fig. 7. The architecture of the species bird classification model is constructed using VGG16, Inception-v3, and ResNet50.

To be more detail, I will describe the architecture of the model which used VGG16 as pre-trained models. To implement the transfer learning, a pre-trained VGG16 model (trained on ImageNet) is utilized as a base. The base model's layers are frozen (vgg16_model.trainable = False) to prevent their weights from being updated during training. This pre-trained base is then followed by a sequential model that includes a flattening layer to convert the output of the convolutional layers into a 1D vector. This flattened output is then passed to a dense layer with 1950 units and ReLU activation, followed by a batch normalization layer. Finally, the output is fed to another dense layer with 200 units (representing the 200 bird species) and a softmax activation function to produce the classification probabilities. This architecture leverages the learned features from the pre-trained VGG16 model and adapts them for the bird species classification task. The other models which use Inception-v3 and ResNet50 uses the same structure with this model architecture, just use the different base model.

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.applications import VGG16

# Load VGG16 pretrained model
vgg16_model = VGG16(input_shape=(224, 224, 3),
                    include_top=False,
                    weights='imagenet')

# Freeze the base model
vgg16_model.trainable = False

# Build Sequential model
model_vgg16 = keras.Sequential([
    keras.Input(shape=(224, 224, 3)),
    vgg16_model,
    layers.Flatten(),
    layers.Dense(units=1950, activation='relu'),
    layers.BatchNormalization(),
    layers.Dense(units=200, activation='softmax')
])

model_vgg16.summary()
```

```
# Compile model
model_vgg16.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

```
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss',patience=10)

# Fit the model
history = model_vgg16.fit(
    train_generator,
    validation_data=test_generator,
    epochs=100,
    verbose=1,
    callbacks=[early_stop]
)
```

Fig. 8. My code for the model uses VGG16 as a pre-trained model

The below table represents an overview of the different hyperparameters and their corresponding assigned value in this assignment.

Table 2. Training parameters

No.	Factors	Selected value
1	Optimizer	Adam
2	Activation functions	ReLU, softmax
3	Loss function	categorical_crossentropy
4	Metrics	accuracy

Methods that used to address the overfitting.
In the process of doing the assignment, the models are overfitting because the dataset is so small, so I have used some methods such as data augmentation techniques and early stopping to reduce the overfitting rate.

Data Augmentation
Due to the limited number of images representing 200 bird species from the CUB-200 dataset, , the trained networks experienced overfitting, failing to generalize effectively. Several data augmentation techniques were employed to address this issue. Data augmentation expands a dataset by applying transformations, such as flipping, translation, and shear, to the existing data. In this assignment, rotation is done in range of 10. The width and height shift are applied within 0.1 range. The shear and zoom are applied within a range of 0.1. The horizontal flipping is also being used. The below image details the transformations used for data augmentation on the dataset.

```
[7]  # Rescale all images by 1/255 & generate some data augmentation techniques
     train_datagen = ImageDataGenerator(rescale=1.0/255,
                                        rotation_range=10,
                                        width_shift_range=0.1,
                                        height_shift_range=0.1,
                                        shear_range=0.1,
                                        zoom_range=0.1,
                                        horizontal_flip=True,
                                        fill_mode='nearest')

     test_datagen = ImageDataGenerator(rescale=1.0/255)

     # define parameters
     BATCH_SIZE = 32
     image_size = (224, 224)

     # Create data generators for training and testing data using a DataFrame

     train_generator = train_datagen.flow_from_dataframe(
         dataframe=train_df,
         directory=train_dir,
         x_col='filepath',
         y_col='class_name',
         target_size=image_size,
         batch_size=BATCH_SIZE,
         class_mode='categorical',
         color_mode='rgb'
     )

     test_generator = test_datagen.flow_from_dataframe(
         dataframe=test_df,
         directory=test_dir,
         x_col='filepath',
         y_col='class_name',
         target_size=image_size,
         batch_size=BATCH_SIZE,
         class_mode='categorical',
         color_mode='rgb',
         shuffle=False  # Important for evaluation to keep the order consistent
     )

     Found 4829 validated image filenames belonging to 200 classes.
     Found 1204 validated image filenames belonging to 200 classes.
```

Fig. 9. Overview of Data augmentation techniques applied in this assignment.

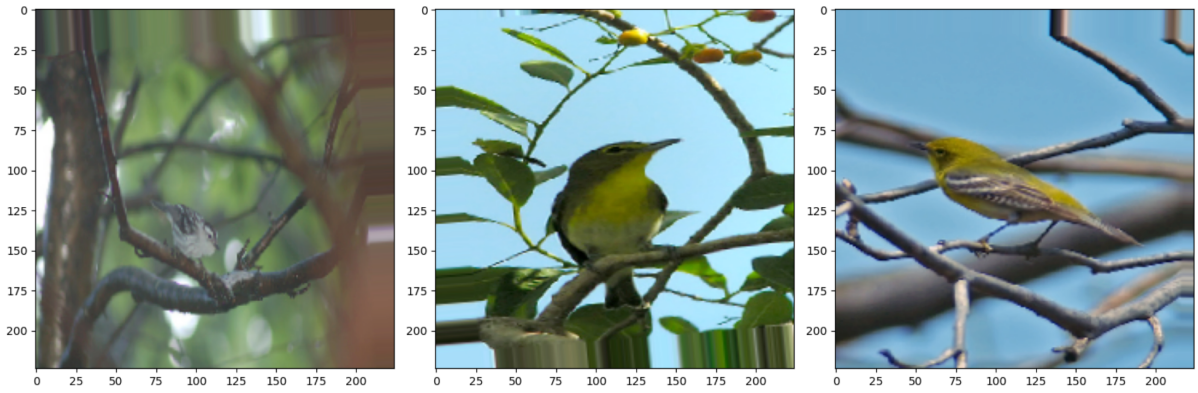

Fig. 10. 3 sample images are generated from the data augmentation techniques

Early Stopping

I used tensorflow's EarlyStopping, which is used to stop training of model if there is no improvement of loss over each epoch, to prevent overfitting during the training of your neural network. In this assignment, the early stopping technique works by monitoring the validation loss (val_loss) after each epoch. If the validation loss does not improve (decrease) for 10 consecutive epochs (patience=10), the training process is stopped automatically. This helps to prevent the model from learning noise and irrelevant patterns in the training data, which could lead to poor performance on unseen data. By stopping the training early, it helps find the optimal point where the model generalizes well to new data, thus mitigating overfitting.

```
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss',patience=10)
```

Fig. 11. Early stopping is used to trained 3 models in this assignment

4 Results & Disscussion

4.1 Performance Evaluation Metric

In this assignment, two evaluation metrics, namely Top-1 accuracy and Average accuracy per class are mainly used to evaluate the models. Besides using the 2 above two evaluation metrics, I also used the accuracy and loss values of models to evaluate models.

The Top-1 accuracy

It is used to evaluate the overall classification performance of the models.

$$\text{Top-1 accuracy} = \frac{1}{N}\sum_{k=1}^{N} 1\{argmax(y) == groudtruth\}$$

where N is the total number of testing images, y is the output probabilities of 200 classes, and argmax(y) is the class label corresponding to the highest predicted probability by the model. In other words, the model answer (the one with highest probability) must be exactly the expected answer. For example, a picture of a cat is shown, and these are the outputs of the neural network: Tiger: 0.4; Dog: 0.3; Cat: 0.1; Lion: 0.08. Using top-1 accuracy, we count this output as wrong, because it predicted a tiger.

The Average accuracy per class

It is used to assess the model's performance across each individual class.

$$\text{Ave} = \frac{1}{C}\sum_{i=1}^{C} T_i$$

where T_i is the accuracy calculated for all test images of the class C_i, and C is the total number of classes.

4.2 Model Performance Discrepencies and Justification for which one yielded the best results

Overall, based on the results, the models in this assignment are overfitted, as they perform well on the training set but not as accurately on the test set. Using various performance evaluation metrics, the model that uses Inception-v3 as a pre-trained model achieves the best results. While the model using VGG16 has higher training accuracy and lower training loss, its performance on the test set is still not better than the one using Inception-v3.

Table 3. Model Comparison – top-1 accuracy, average accuracy per class, train accuracy, train loss, test accuracy, test loss, precision, recall, f1-score

Models	Top-1 accuracy	Average accuracy per class	Train accuracy	Train loss	Test accuracy	Test loss	Precision	Recall	F1-score
VGG16	0.2467	0.3650	**0.9211**	**0.2851**	0.2467	5.5166	0.37	0.35	0.36
Inception-v3	**0.4493**	**0.4423**	0.9086	0.3049	**0.4493**	**2.9838**	**0.53**	**0.45**	**0.44**
ResNet50	0.0623	0.0574	0.2296	3.4522	0.0623	7.3188	0.09	0.06	0.05

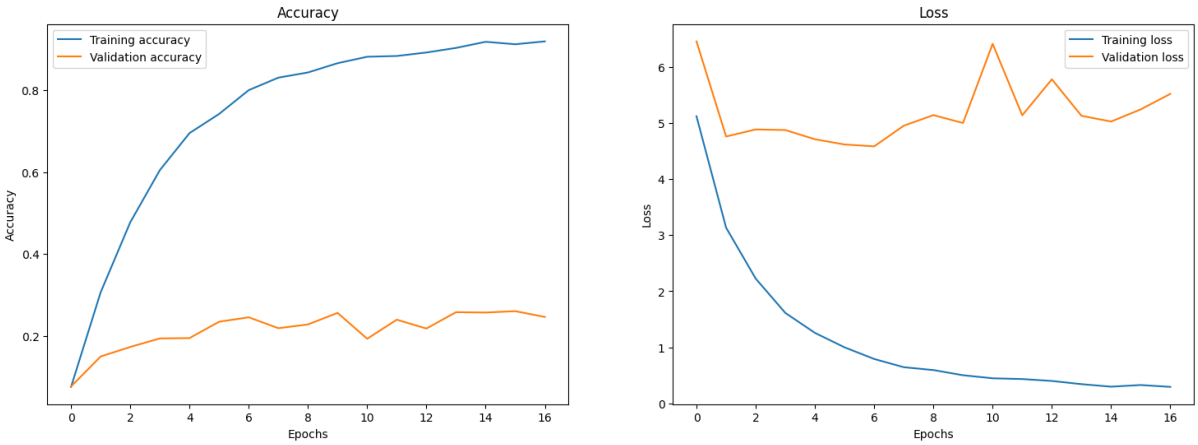

Fig. 12. The accuracy and loss values of the **VGG16** model using Adam optimizers

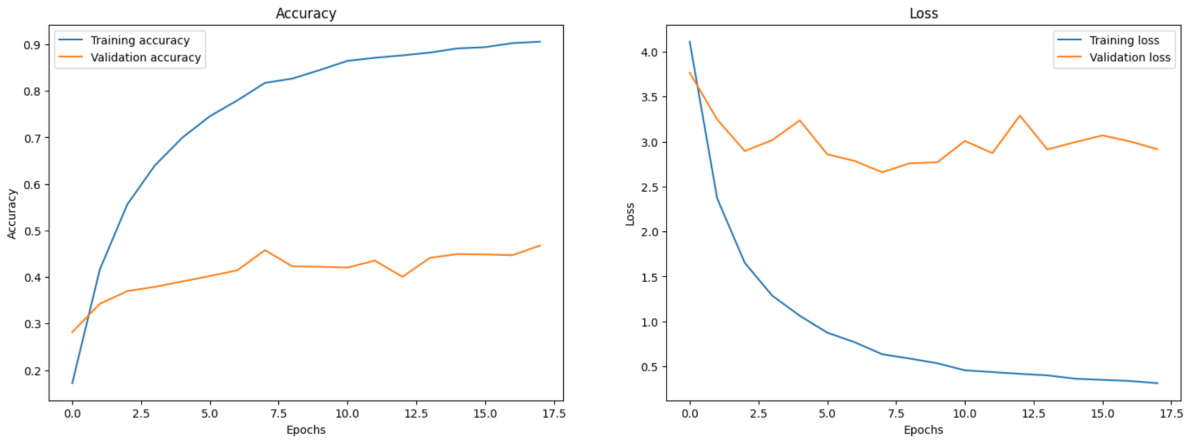

Fig. 13. The accuracy and loss values of the **Inception-v3** model using Adam optimizers

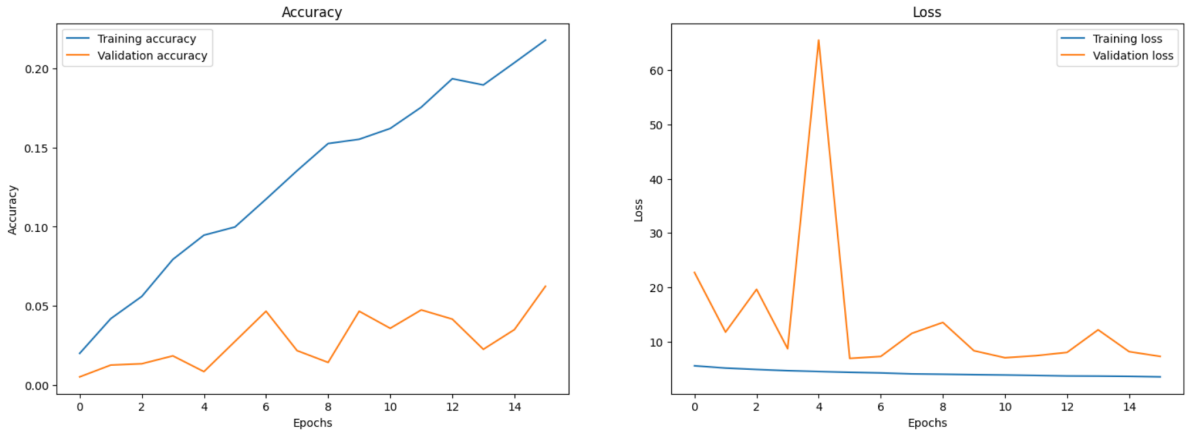

Fig. 14. The accuracy and loss values of the **ResNet50** model using Adam optimizers

5 Conclusion

Although the models' training accuracy is high and the training loss is low, the accuracy and loss on the test set indicate poor generalization. This shows that the models are overfitting, so other techniques like data augmentation, dropout, and regularization should be used to reduce overfitting in futher works. Due to time and resource constraints, particularly the limited GPU usage on Google Colab, these results reflect my best effort within the scope of this assignment.

References

Cheng, B. (2023, June 24). Birds classification with pre-trained convolutional neural networks and CAM heatmap visualisation. *Medium*. https://medium.com/@bobbycxy/birds-classification-with-pre-trained-convolutional-neural-networks-and-cam-heatmap-visualization-3d424423aaa4

Desai, C. (2021). Image Classification Using Transfer Learning and Deep Learning. *International Journal of Engineering and Computer Science*, *10*(9), 25394–25398. https://doi.org/10.18535/ijecs/v10i9.4622

Gulzar, Y., Ünal, Z., Aktaş, H., & Mir, M. S. (2023). Harnessing the Power of transfer learning in sunflower Disease Detection: A comparative study. *Agriculture*, *13*(8), 1479. https://doi.org/10.3390/agriculture13081479

Ikkiocean. (2024, October 3). *Bird Species Classification using DL*. Kaggle. https://www.kaggle.com/code/ikkiocean/bird-species-classification-using-dl/notebook

Jvkchaitanya. (2023, May 16). *Bird species recognition using EfficientNetB0*. Kaggle. https://www.kaggle.com/code/jvkchaitanya410/bird-species-recognition-using-efficientnetb0/notebook#Evaluate-Performance-of-model

Kumar, S. V., & Kondaveeti, H. K. (2024). Bird species recognition using transfer learning with a hybrid hyperparameter optimization scheme (HHOS). *Ecological Informatics*, *80*, 102510. https://doi.org/10.1016/j.ecoinf.2024.102510

SuNT. (n.d.). *SUNT's blog | AI in Practical*. https://tiensu.github.io/blog/33_dataaugmentation_modelcheckpoint_cnn/

Vo, Hoang-Tu & Thien, Nhon & Mui, Kheo. (2023). Bird Detection and Species Classification: Using YOLOv5 and Deep Transfer Learning Models. International Journal of Advanced Computer Science and Applications. 14. 10.14569/IJACSA.2023.01407102.

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., & Perona, P. (2011). Caltech-UCSD Birds 200 (Published). California Institute of Technology.