

COS30082 Assignment

Bird Species Classification

Individual Assignment

This is for students who are aiming for a credit grade.

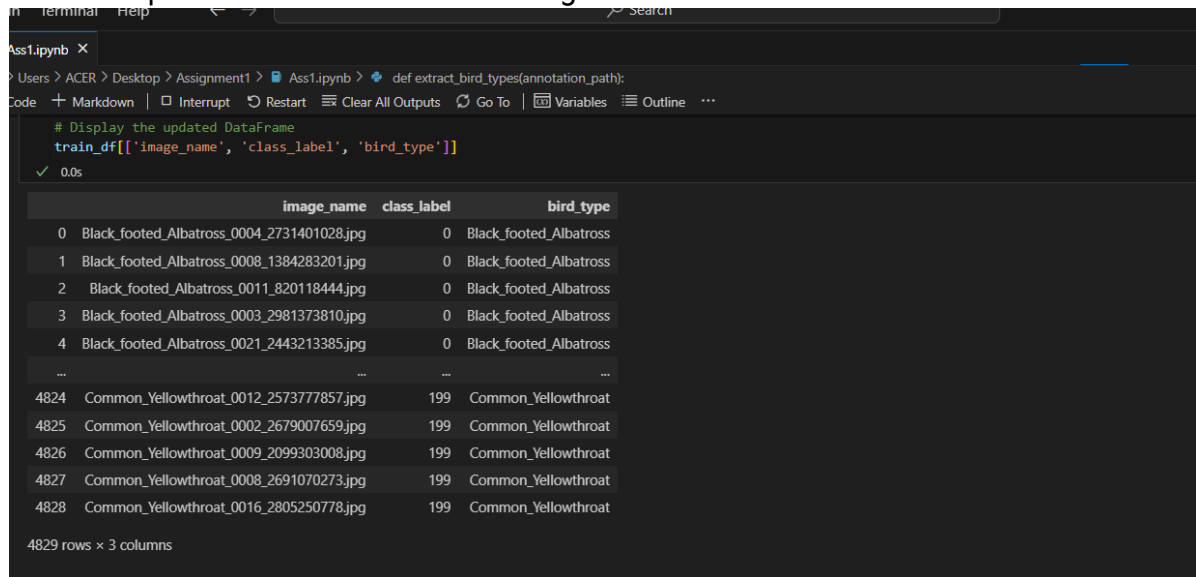
Truong Minh Son – 104221795

1. Methodology

The dataset is altech-UCSD Birds 200 (CUB-200), which contains images of 200 bird species primarily from North America. There are 4,829 images available for training along with the annotation train.txt. Of course there are testing images and the annotation file test.txt.

Data Preprocessing

I downloaded and understand the testing and training file , reading the file, I have to divide all the data to class_label and move images to the corresponding bird type folders inside the train folder, the similar process was done with the testing file.



The screenshot shows a Jupyter Notebook interface with a file named 'Ass1.ipynb'. The code cell contains a comment '# Display the updated DataFrame' followed by the command 'train_df[['image_name', 'class_label', 'bird_type']]'. The output is a DataFrame with 4829 rows and 3 columns. The first few rows are for 'Black_footed_Albatross' (class_label 0) and the last few rows are for 'Common_Yellowthroat' (class_label 199).

	image_name	class_label	bird_type
0	Black_footed_Albatross_0004_2731401028.jpg	0	Black_footed_Albatross
1	Black_footed_Albatross_0008_1384283201.jpg	0	Black_footed_Albatross
2	Black_footed_Albatross_0011_820118444.jpg	0	Black_footed_Albatross
3	Black_footed_Albatross_0003_2981373810.jpg	0	Black_footed_Albatross
4	Black_footed_Albatross_0021_2443213385.jpg	0	Black_footed_Albatross
...
4824	Common_Yellowthroat_0012_2573777857.jpg	199	Common_Yellowthroat
4825	Common_Yellowthroat_0002_2679007659.jpg	199	Common_Yellowthroat
4826	Common_Yellowthroat_0009_2099303008.jpg	199	Common_Yellowthroat
4827	Common_Yellowthroat_0008_2691070273.jpg	199	Common_Yellowthroat
4828	Common_Yellowthroat_0016_2805250778.jpg	199	Common_Yellowthroat

4829 rows x 3 columns

Figure 1: Understanding the Data

As a result, I got the desired divided preprocessing data which is ready for evaluation as well as applying the ML models.

```
[10] ✓ 0.2s
... Found 4829 images belonging to 200 classes.
    Found 1204 images belonging to 200 classes.
```

Now I got the folder for testing and training

Model Architecture

Convolutional Neural Network (CNN)

A standard CNN model for image classification typically consists of several layers. In this case:

- The CNN starts with three convolutional layers (with filters of 32, 64, and 128 respectively), each followed by a max-pooling layer to reduce spatial dimensions.
- After the convolutional layers, the output is flattened and passed to a fully connected layer with 512 neurons, using the ReLU activation function for non-linearity.
- The output layer consists of 200 neurons (for 200 bird species) with a softmax activation function to predict the probability of each class.

```
# CNN Model Definition
cnn_model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(200, activation='softmax') # Assuming 200 bird classes
])

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

# Train the CNN model
cnn_model.fit(train_generator, epochs=10, validation_data=test_generator)

# Evaluate the CNN model
test_loss, test_accuracy = cnn_model.evaluate(test_generator)
print(f"Test Accuracy (CNN): {test_accuracy}")

# Getting predictions and calculating Top-1 Accuracy
test_generator.reset() # Reset the generator before making predictions
preds = cnn_model.predict(test_generator, verbose=1)

# Get true labels and predicted labels
y_true = test_generator.classes
y_pred = preds.argmax(axis=1)

# Calculate accuracy score and classification report
accuracy = accuracy_score(y_true, y_pred)
print(f"Top-1 Accuracy: {accuracy}")

# Detailed classification report
report = classification_report(y_true, y_pred, target_names=list(test_generator.class_indices.keys()))
print(report)
```

Loss Function and Optimizer:

- Loss function: Categorical Cross-Entropy, which is standard for multi-class classification problems.
- Optimizer: Adam optimizer was chosen for its adaptive learning rate and efficient gradient-based optimization.

Hyperparameters:

- Learning rate: 0.001 (default for Adam)
- Batch size: 32
- Epochs: 10
- Target image size: 224x224

Regularization Techniques:

- Data Augmentation: Rotation, width/height shift, zoom, and horizontal flip to introduce variability in the data and prevent overfitting.

Training Process:

The model was trained over 10 epochs using the ImageDataGenerator for data loading and augmentation. Validation was performed using a separate test set. Accuracy and loss metrics were recorded during training to monitor performance.

```
self.warn_if_super_not_called()
Epoch 1/10
151/151 ----- 235s 2s/step - accuracy: 0.0042 - loss: 5.3962 - val_accuracy: 0.0050 - val_loss: 5.3100
Epoch 2/10
151/151 ----- 199s 1s/step - accuracy: 0.0104 - loss: 5.2913 - val_accuracy: 0.0174 - val_loss: 5.2129
Epoch 3/10
151/151 ----- 195s 1s/step - accuracy: 0.0971 - loss: 4.7278 - val_accuracy: 0.0266 - val_loss: 5.4092
Epoch 4/10
151/151 ----- 204s 1s/step - accuracy: 0.5271 - loss: 2.4035 - val_accuracy: 0.0257 - val_loss: 8.7995
Epoch 5/10
151/151 ----- 195s 1s/step - accuracy: 0.8669 - loss: 0.7752 - val_accuracy: 0.0307 - val_loss: 12.6471
Epoch 6/10
151/151 ----- 236s 2s/step - accuracy: 0.9555 - loss: 0.4159 - val_accuracy: 0.0299 - val_loss: 14.9106
Epoch 7/10
151/151 ----- 215s 1s/step - accuracy: 0.9809 - loss: 0.1846 - val_accuracy: 0.0291 - val_loss: 14.5671
Epoch 8/10
151/151 ----- 226s 1s/step - accuracy: 0.9906 - loss: 0.0556 - val_accuracy: 0.0341 - val_loss: 15.7848
Epoch 9/10
151/151 ----- 211s 1s/step - accuracy: 0.9959 - loss: 0.0549 - val_accuracy: 0.0316 - val_loss: 14.9261
Epoch 10/10
151/151 ----- 214s 1s/step - accuracy: 0.9956 - loss: 0.0358 - val_accuracy: 0.0332 - val_loss: 15.6539
38/38 ----- 11s 287ms/step - accuracy: 0.0459 - loss: 15.2668
Test Accuracy (CNN): 0.03322589641809464
```

ResNet (ResNet50 Pre-trained Model)

ResNet50 is a deep pre-trained model with 50 layers, designed for image classification tasks. It uses residual blocks to allow gradients to flow more easily, solving the vanishing gradient problem in deep networks. This was followed by a custom fully connected layer (512 neurons with ReLU activation) and a softmax output layer with 200 classes.

Loss Function and Optimizer:

- Loss function: Categorical Cross-Entropy.
- Optimizer: Adam optimizer.

Hyperparameters:

- Batch size: 32
- Learning rate: 0.001
- Epochs: 10
- Target image size: 224x224

Regularization Techniques:

- Transfer Learning: Pre-trained weights from the ImageNet dataset were used to leverage ResNet50's feature extraction capabilities.
- Freezing layers: All layers from the pre-trained ResNet were frozen, except for the new top layers, to prevent overfitting.
- Data Augmentation: The same augmentation strategy as the CNN was applied, including random transformations to make the model more robust.

Training Process:

The model was trained using the augmented data with 10 epochs. Since the base ResNet layers were frozen, only the new fully connected layers were updated during training. Validation was done on a separate test set, and top-1 accuracy was recorded.

InceptionV3 (Pre-trained Model)

InceptionV3 is another deep learning architecture designed for image classification. It employs multiple filter sizes in each convolutional block, allowing the model to capture features at different scales.

- For this task, the InceptionV3 model was loaded without the top layers, and the output was fed into a global average pooling layer. A dense layer with 512 neurons (ReLU activation) was added, followed by a softmax output layer with 200 neurons for the bird species classification.

Loss Function and Optimizer:

- Loss function: Categorical Cross-Entropy, standard for classification tasks.
- Optimizer: Adam optimizer for its efficiency in handling large datasets and deep models.

Hyperparameters:

- Batch size: 32
- Learning rate: 0.001
- Epochs: 10
- Target image size: 224x224

```

# Set the paths to your training and testing directories
train_dir = 'C:/Users/ACER/Downloads/bird_class/train'
test_dir = 'C:/Users/ACER/Downloads/bird_class/test'

# Data generators for loading images (assuming 299x299 image size for InceptionV3)
datagen = ImageDataGenerator(rescale=1./255)

train_generator = datagen.flow_from_directory(
    train_dir,
    target_size=(299, 299), # InceptionV3 expects 299x299 images
    batch_size=32,
    class_mode='categorical'
)

test_generator = datagen.flow_from_directory(
    test_dir,
    target_size=(299, 299),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

# Load the pre-trained InceptionV3 model without the top layer
inception_base = InceptionV3(weights='imagenet', include_top=False, input_shape=(299, 299, 3))

# Freeze the layers of InceptionV3 to prevent them from being updated during training
inception_base.trainable = False

# Build the complete model by adding custom layers on top of InceptionV3
inception_model = models.Sequential([
    inception_base,
    layers.GlobalAveragePooling2D(), # Pool the features from InceptionV3
    layers.Dense(512, activation='relu'),
    layers.Dense(200, activation='softmax') # Assuming 200 bird classes
])

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

# Train the model
inception_model.fit(train_generator, epochs=10, validation_data=test_generator)

# Evaluate the model
test_loss, test_accuracy = inception_model.evaluate(test_generator)
print(f"Test Accuracy (InceptionV3): {test_accuracy}")

# Get predictions and calculate Top-1 Accuracy
test_generator.reset()
preds = inception_model.predict(test_generator, verbose=1)

# Get true labels and predicted labels
y_true = test_generator.classes
y_pred = preds.argmax(axis=1)

```

Regularization Techniques:

- **Transfer Learning:** InceptionV3's pre-trained layers from the ImageNet dataset were used, with the top layers customized for the bird classification task.
- **Layer Freezing:** The base layers from the pre-trained model were frozen, only allowing the custom top layers to be trained.
- **Data Augmentation:** The same data augmentation strategy as in CNN and ResNet was applied to prevent overfitting and to make the model more robust.

Training Process:

InceptionV3 was trained using the pre-processed dataset over 10 epochs. The model's performance was validated using the test set, and metrics like top-1 accuracy and top-5 accuracy were computed.

2. Results and Discussion

. Model Performance Metrics

- **Top-1 Accuracy:**
 - The Top-1 accuracy of the model, which indicates the percentage of correctly classified test images when considering only the highest predicted class, was

calculated based on the test set. This shows how well the model can predict the correct bird species out of the 200 possible classes.

- **Average Accuracy per Class:**
 - The model's average accuracy per class, calculated to evaluate performance across individual bird species classes.

However, all the models are not good to go, I spent hours and period for training, even for image distribution but I still got really bad performance on every models. Having less than **1% accuracy** indicates that the model is struggling significantly with the bird species classification task. Compared to the other models, Inception V3 tends to have the best result.

Challenges and Potential Improvements

- One challenge encountered was the significant class imbalance in the dataset. Certain bird species had far more images than others, which likely led to the model performing better on those well-represented species. Techniques such as oversampling the minority classes or using class weights could further improve the model's fairness across species.
- Another area for improvement is hyperparameter tuning. Grid search or random search for hyperparameters such as learning rate, dropout rate, and batch size might yield better results by fine-tuning the model to achieve optimal performance.
- Exploring other architectures, like InceptionV3 or EfficientNet, could provide additional insights into model performance, as these architectures are known for performing well on image classification tasks with complex datasets.

The main problem of this bird species classification task lies in the complicated dataset composed of 200 bird species with high intra-class similarity and low inter-class variation, which makes it hard for a model to learn distinctive patterns for each species when the number of training samples is limited in each class. Another very likely problem is overfitting, in that it can memorize the data without generalizing to new and unseen data that may have poor performances. The wrong hyperparameters can make this task worse, such as a high learning rate or fewer epochs that prevent the model from learning effectively. It will underperform even with a more complex architecture such as ResNet50 or InceptionV3 because these models require very large datasets to exploit their capacity.

With these challenges in mind, several possible solutions can be implemented. First, the dataset should be properly labeled and balanced—for example, techniques concerning oversampling or weighting of classes—so the model can learn better. This can potentially be improved by reducing the learning rate, increasing the number of epochs, or using a less complex model architecture. Data augmentation should be performed with care, increasing diversity while preserving relevant features. Possible regularization techniques against overfitting, such as dropout or early stopping, can be used so that the model generalizes better across different species.

In the future, I may set up the tasks and learn more about multi-class classification. After doing this assignment portfolio, I gained knowledge how to preprocess the data class, label the class, and of course set up the models, of course training and evaluating process. And the most important thing I have learnt is consideration of the results. What a challenge and long way to go.

3. My source code

I have given the ipynb with this report , please have a look, and thank you so much for reading this report, sincerely .