CS 464: Introduction to Machine Learning

Fall 2023-2024

Term Project
Final Progress Report

Group 3

Instructor:

Ayşegül Dündar Boral

Team Members:
Mennan Gök (22003074)
Ömer Tuğrul (22002723)
Selin Ataş (21902858)
Gökay Balcı (22003796)
Ömer Asım Doğan (21903042)

Contents

1 Introduction

Ornithology often requires the careful task of manually identifying bird species. This process is particularly challenging in diverse ecosystems due to the vast number of species and intra-species variations. The Bird Species Image Classification Project tackles this challenge by utilizing machine learning algorithms to automate identification.

These algorithms can process large image datasets and select complex patterns, significantly enhancing the efficiency of species identification. This technological advancement is crucial as biodiversity research and conservation depend on precise and rapid data collection. Advanced classification models of the project promise to transform avian biodiversity conservation by enabling rapid and accurate identification of various bird species, providing vital support for researchers and public officials in diverse environments.

The "Bird Species Image Classification Project" includes two different machine learning models, which are Multinomial Logistic Regression and Random Forest, and a Deep Learning model, namely a Convolutional Neural Network capable of accurately identifying and classifying different bird species based on images that are provided from [1]. We also utilized two different transfer learning models, EfficientNet-B0 and ResNet50, to leverage pre-trained models on our dataset, aiming to achieve high accuracy and improved performance.

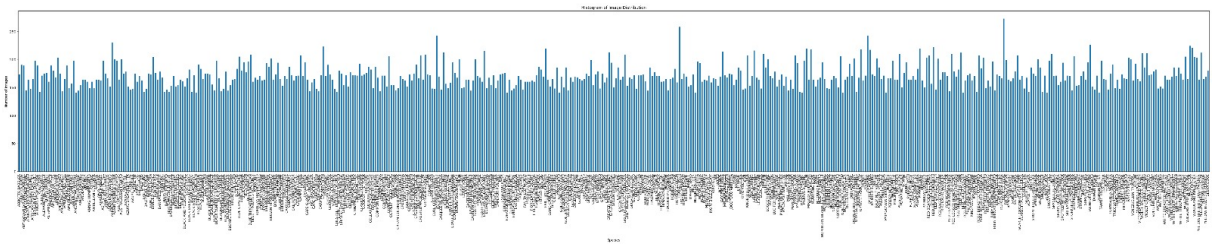


Figure 1: The Histogram of the Database of 525 Bird Species

2 Dataset Description

In this project, the dataset BIRDS 525 SPECIES-IMAGE CLASSIFICATION [1] is utilized. It has a collection of 84635 training images, 2625 test images (5 images per species), and 2625 validation images (5 images per species) of 525 different types of birds. The dataset includes high-quality images such that there is only one bird in each image and the bird typically takes up at least 50% of the image. All images are (224 X 224 X 3) color images in jpg format. The training set is not balanced, having a varying number of files per species. Therefore, we preprocessed the data to use balanced classes. Each species has at least 130 training image files. Also, the number of images for each species in the test and validation folders is few compared to the train images, so we needed to increase them. Furthermore, the data set also includes a file birds.csv. This csv file contains 5 columns which are named relative file path to an image file, bird species class name, and scientific name with class index and data subset (train, test, or valid).

3 Preprocessing the Data

The successful development and training of accurate machine learning models for bird species classification rely heavily on thoughtful data preparation. In this context, several crucial steps have been undertaken to ensure the dataset's balance, informativeness, and variability, optimizing the models' capacity to generalize across diverse avian species. From strategic data balancing and stratified sampling to the implementation of grayscale processing, image resizing, and data augmentation, each preprocessing step plays a pivotal role in enhancing the models' robustness. This comprehensive approach not only addresses challenges related to imbalanced class representation but also considers computational efficiency and the intricate nature of bird images. The following sections detail each preprocessing step, shedding light on the methodologies employed to create a well-structured dataset that lays a solid foundation for training

effective bird species classification models.

3.1 Data Balancing and Stratified Sampling

Ten bird species with the highest number of images were identified, and 100 images were selected from each of these species. The remaining images of these birds were added to the test and validation files to increase their quantity. Hence, we generated "RGB_train/test/val" folders. This step was essential as initially, only 5 images per species were allocated for testing and validation, which was insufficient. If some species are represented by a higher number of images than others, it can lead to a biased model that performs well only for those overrepresented species. By identifying the top 10 species and selecting an equal number of images for each, the dataset becomes more balanced, allowing the model to learn to identify each species with equal proficiency. Furthermore, we increased the efficiency of the models by reducing data samples.

3.2 Gray Scaling

To simplify computational cost and eliminate the color bias in the detection process, we created gray scaled folders "gray_train/test/val" from "RGB _train/test/val" folders. Reducing the images from three channels (red, green, blue) to a single channel allows the model to focus more on shapes and textures than color—a key feature in bird identification. This approach also reduces issues related to varying lighting conditions, which can alter color perception, and helps reduce the risk of overfitting, as the model has fewer features to process.

3.3 Image Resizing

The size of all images in the dataset was 224x224 pixels resolution. For ML models, we resized the images to 64x64 pixels resolution to reduce computational complexity; however, in CNN and Transfer Learning models we utilized the original sized images. The reason for this decision is that ML models may not efficiently handle higher-resolution images like CNNs. This resizing is sufficient for extracting key features needed for classification, offering a balanced compromise between detail and computational efficiency. Conversely, for CNNs and Transfer Learning models, images were retained at the original resolution of 224x224 pixels, benefiting from the advanced capability to process and extract finer details from high-resolution images of these models, which is crucial for accurate and nuanced classification tasks.

Parameter	**Value**
Total Classes	10
Total Train Images	1000
Total Valid Images	629
Total Test Images	634

Table 1: Dataset Information

3.4 Data Augmentation

The data augmentation code enhances the dataset through a series of transformations aimed at increasing the robustness of the model. It provides variability by applying random horizontal and vertical flips, random rotations of up to 10 or 15 degrees, and random crops that resize up to 224x224 pixels. Also, it performs arbitrary affine transformations with translations, adjusts brightness and contrast with color change, and lastly converts the augmented images into tensors suitable for PyTorch.

In conclusion, the preprocessing steps aim to create a balanced, informative, and varied dataset for training bird species classification models. The strategies employed address issues such as imbalanced class representation, computational efficiency, and the diverse nature of bird images. It means the models' ability to generalize and perform well on a variety of bird species will be better due to carefully preprocessed data.

```
train_transforms = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.RandomResizedCrop(224, scale=(0.9, 1.1)),
    transforms.ToTensor()
])
```

Listing 1: Transforms for Training

```
valid_test_transforms = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(15),
    transforms.ColorJitter(brightness=0.2, contrast=0.2),
    transforms.RandomResizedCrop(224, scale=(0.9, 1.1)),
    transforms.ToTensor()
])
```

Listing 2: Transforms for Validation and Testing

Parameter	**Value**
Total Classes	10
Total Train Images	5000
Total Valid Images	3145
Total Test Images	3170

Table 2: Dataset Information

3.5 Multinomial Logistic Regression

Although we proposed using SVM in our project, we changed this method to Multinomial Logistic Regression. The decision to transition from an SVM to an MLR model in the project was mainly due to the need for a more computationally efficient approach. Compared to SVMs, Logistic Regression is less demanding in terms of computational resources and handles large datasets more effectively, which is a key advantage for image classification tasks. Additionally, Logistic Regression is inherently better suited for tasks involving multiple classes, such as distinguishing various bird species. It also has the upside of being easier to interpret.

3.6 Random Forest

Random Forest is an ensemble learning method that combines the predictions of multiple decision trees. For this project, Random Forest is considered as a potential model due to its ability to handle diverse features and provide insights into feature importance. Image features, extracted using methods like color histograms or texture analysis, can be fed into the Random Forest model. The ensemble nature of Random Forest helps mitigate overfitting and enhances the generalization capabilities of the model.

3.7 Convolutional Neural Network

CNNs are a natural fit for image classification tasks due to their ability to automatically learn hierarchical features from images. In this project, a CNN architecture has been selected for its capacity to capture spatial hierarchies and patterns in bird images. The chosen CNN architecture involves convolutional layers for feature extraction, followed by fully connected layers for classification.

3.8 Transfer Learning

Lastly, we have generated Transfer Learning models to enhance the efficiency and accuracy of the bird species classification task. Transfer Learning allows us to utilize a pre-trained neural network, capitalizing on the rich feature extraction capabilities these networks have developed through training on extensive, diverse datasets. Thanks to Transfer Learning, we adapted these pre-existing neural networks to our specific classification task with minimal additional training. By doing so, we bypass the intensive computational costs typically associated with training deep learning models from scratch. The pre-trained models bring a wealth of nuanced image recognition capabilities, which, when fine-tuned with our bird image dataset, create a powerful tool for identifying species across various environments and conditions.

4 Training

Model trained with four different algorithms. We used Random Forest and Multinomial Logistic Regression from machine learning models and CNN and Transfer Learning from deep learning models. Every model was trained with 4 different kinds of data consisting of RGB, Augmented RGB, Greyscale, and Augmented Grayscale. While the RGB-trained models were tested on the RGB test set, Grayscale-trained ones were tested on the Grayscale test set. We compared the results of the trained models with each other and themselves.

4.1 Multinomial Logistic Regression

For this model, we first standardized the data using "Standart Scaler". This is one of the major steps before training the model since logistic regression is sensitive to the scale of input features. The solver is the SAGA for our model. It is one of the available algorithms in scikit-learn. The multinomial in the model indicates that the algorithm should use the multinomial logistic regression, which is suitable for more than two classes. Iteration number is "100" which indicates the number of iterations for the solver to converge. The "fit_transform" method computes the mean and standard deviation of each feature in the training set and then transforms the data accordingly.

Compared to the baseline accuracy of 0.1, we can say that our model is learning patterns in the data and performing better than a random guess.The highest accuracy of the model was **0.5425** when we trained with Augmented RGB data. The lowest was **0.2933** when we trained with the Augmented Greyscale data.

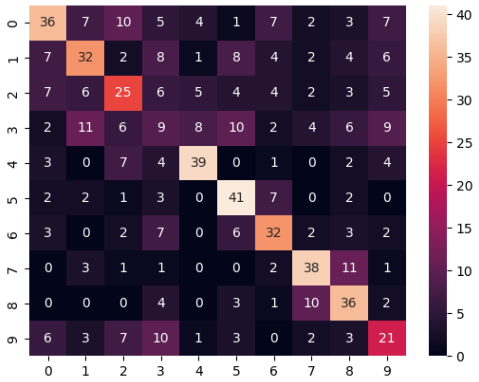
These results show that the MLR model is effectively capturing the patterns in the input data and making accurate predictions. The success of the model is indicative of effective feature discrimination and successful training.

Similar to the Random Forest Algorithm results, the confusion matrices show the model mostly mixed the first 4 classes with each other while classifying.

In conclusion, the MLR exceed our expectations with a 0.55 accuracy in image classification.

	Precision	Recall	F1-Score	Support
0	0.55	0.44	0.49	82
1	0.50	0.43	0.46	74
2	0.41	0.37	0.39	67
3	0.16	0.13	0.15	67
4	0.67	0.65	0.66	60
5	0.54	0.71	0.61	58
6	0.53	0.56	0.55	57
7	0.61	0.67	0.64	57
8	0.49	0.64	0.56	56
9	0.37	0.38	0.37	56
Accuracy			0.49	634
Macro Avg	0.48	0.50	0.49	634
Weighted Avg	0.48	0.49	0.48	634

(a) Multinomial Logistic Regression Classifier Evaluation Metrics with RGB data

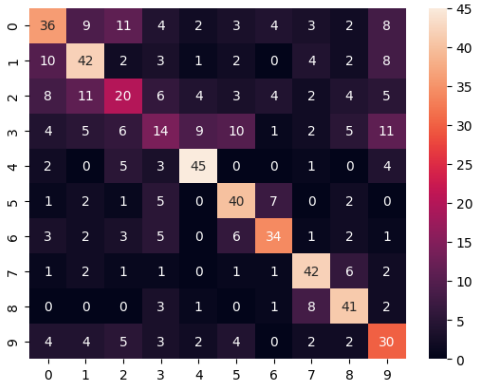


(b) Confusion Matrix with RGB data

Figure 2: Multinomial Logistic Regression Classifier Evaluation with RGB

	Precision	Recall	F1-Score	Support
0	0.52	0.44	0.48	82
1	0.55	0.57	0.56	74
2	0.37	0.30	0.33	67
3	0.30	0.21	0.25	67
4	0.70	0.75	0.73	60
5	0.58	0.69	0.63	58
6	0.65	0.60	0.62	57
7	0.65	0.74	0.69	57
8	0.62	0.73	0.67	56
9	0.42	0.54	0.47	56
Accuracy			0.54	634
Macro Avg	0.54	0.56	0.54	634
Weighted Avg	0.53	0.54	0.53	634

(a) Multinomial Logistic Regression Classifier Evaluation Metrics with Augmented RGB data

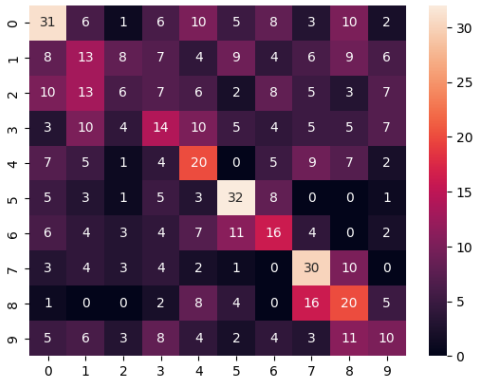


(b) Confusion Matrix with Augemented RGB data

Figure 3: Multinomial Logistic Regression Classifier Evaluation

	Precision	Recall	F1-Score	Support
0	0.39	0.38	0.39	82
1	0.20	0.18	0.19	74
2	0.20	0.09	0.12	67
3	0.23	0.21	0.22	67
4	0.27	0.33	0.30	60
5	0.45	0.55	0.50	58
6	0.28	0.28	0.28	57
7	0.37	0.53	0.43	57
8	0.27	0.36	0.31	56
9	0.24	0.18	0.20	56
Accuracy			0.30	634
Macro Avg	0.29	0.31	0.29	634
Weighted Avg	0.29	0.30	0.29	634

(a) Multinomial Logistic Regression Classifier Evaluation Metrics with Gray scale data

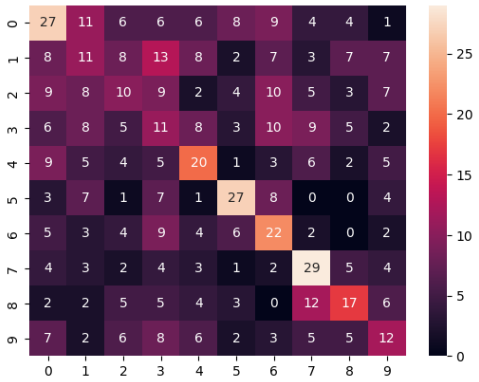


(b) Confusion Matrix with gray scale data

Figure 4: Multinomial Logistic Regression Classifier Evaluation with Gray scale data

	Precision	Recall	F1-Score	Support
0	0.34	0.33	0.33	82
1	0.18	0.15	0.16	74
2	0.20	0.15	0.17	67
3	0.14	0.16	0.15	67
4	0.32	0.33	0.33	60
5	0.47	0.47	0.47	58
6	0.30	0.39	0.34	57
7	0.39	0.51	0.44	57
8	0.35	0.30	0.33	56
9	0.24	0.21	0.23	56
Accuracy			0.29	634
Macro Avg	0.29	0.30	0.29	634
Weighted Avg	0.29	0.29	0.29	634

(a) Multinomial Logistic Regression Classifier Evaluation Metrics with Augmented Gray scale data



(b) Confusion Matrix with Augmented Gray scale data

Figure 5: Multinomial Logistic Regression Classifier Evaluation with Augmented Gray scale data

In addition to these different datasets, Principal Component Analysis is applied to RGB dataset and it is used to train MLR model. Evaluation results are given on the table below:

	Precision	Recall	F1-Score	Support
0	0.56	0.43	0.48	82
1	0.53	0.45	0.49	74
2	0.31	0.31	0.31	67
3	0.17	0.15	0.16	67
4	0.62	0.63	0.63	60
5	0.52	0.71	0.60	58
6	0.54	0.54	0.54	57
7	0.59	0.63	0.61	57
8	0.51	0.62	0.56	56
9	0.36	0.36	0.36	56
Accuracy			0.47	634
Macro Avg	0.47	0.48	0.47	634
Weighted Avg	0.47	0.47	0.47	634

Table 3: Multinomial Logistic Regression Classifier with PCA Evaluation Metrics

4.2 Random Forest

```
1  # Train Random Forest Classifier
2  rf_classifier_augrgb = RandomForestClassifier(n_estimators=10,
       random_state=464)
3  rf_classifier_augrgb.fit(X_train, y_train)
```

Listing 3: Train Random Forest Classifier

We used 10 decision trees to classify data into 10 different classes for the Random Forest algorithm. This helped us handle the complexity of sorting data while making sure our model does not get too specific (overfitting).

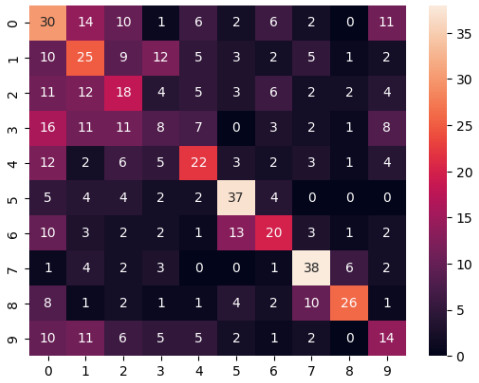
Compared to the baseline accuracy of 0.1, we can say that the Random Forest model is learning patterns in the data and performing better than a random guess.

The highest accuracy of the model was **0.3754** when we trained with Augmented RGB data. The lowest was **0.2555** when we trained with the Augmented Greyscale data. We can infer from these results that the Random Forest performing poorly on the dataset. The dataset might be challenging for this model, containing complex patterns or noise that the model is struggling to learn. Also, features used for classification may not be informative enough or may not adequately represent the differences between classes. The results of the Random Forest were the worst among all models.

The confusion matrices show the model mostly mixed the first 4 classes with each other while classifying birds.

	Precision	Recall	F1-Score	Support
0	0.27	0.37	0.31	82
1	0.29	0.34	0.31	74
2	0.26	0.27	0.26	67
3	0.19	0.12	0.15	67
4	0.41	0.37	0.39	60
5	0.55	0.64	0.59	58
6	0.43	0.35	0.38	57
7	0.57	0.67	0.61	57
8	0.68	0.46	0.55	56
9	0.29	0.25	0.27	56
Accuracy			0.38	634
Macro Avg	0.39	0.38	0.38	634
Weighted Avg	0.38	0.38	0.37	634

(a) Random Forest Classifier Evaluation Metrics with RGB data

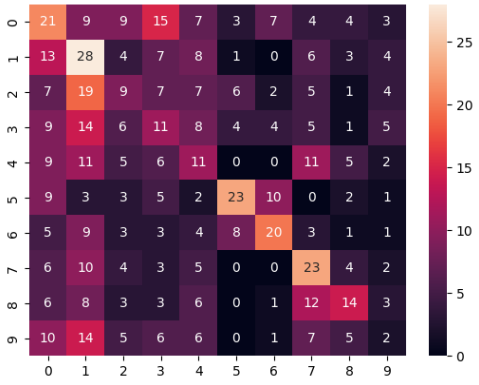


(b) Confusion Matrix with RGB data

Figure 6: Random Forest Classifier Evaluation with RGB data

	Precision	Recall	F1-Score	Support
0	0.22	0.26	0.24	82
1	0.22	0.38	0.28	74
2	0.18	0.13	0.15	67
3	0.17	0.16	0.17	67
4	0.17	0.18	0.18	60
5	0.51	0.40	0.45	58
6	0.44	0.35	0.39	57
7	0.30	0.40	0.35	57
8	0.35	0.25	0.29	56
9	0.07	0.04	0.05	56
Accuracy			0.26	634
Macro Avg	0.26	0.26	0.25	634
Weighted Avg	0.26	0.26	0.25	634

(a) Random Forest Classifier Evaluation Metrics

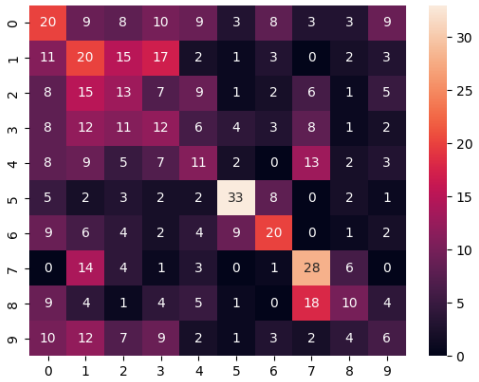


(b) Confusion Matrix with grayscale data

Figure 7: Random Forest Classifier Evaluation with Gray Data

	Precision	Recall	F1-Score	Support
0	0.23	0.24	0.24	82
1	0.19	0.27	0.23	74
2	0.18	0.19	0.19	67
3	0.17	0.18	0.17	67
4	0.21	0.18	0.19	60
5	0.60	0.57	0.58	58
6	0.42	0.35	0.38	57
7	0.36	0.49	0.41	57
8	0.31	0.18	0.23	56
9	0.17	0.11	0.13	56
Accuracy			0.27	634
Macro Avg	0.28	0.28	0.28	634
Weighted Avg	0.28	0.27	0.27	634

(a) Random Forest Classifier Evaluation Metrics



(b) Confusion Matrix with Augmented Gray scale

Figure 8: Random Forest Classifier Evaluation with augmented Gray scale data

In addition to these different datasets, Principal Component Analysis is applied to RGB dataset and it is used to train MLR model. Evaluation results are given on the table below:

	Precision	Recall	F1-Score	Support
0	0.21	0.17	0.19	82
1	0.20	0.24	0.22	74
2	0.18	0.24	0.21	67
3	0.10	0.10	0.10	67
4	0.32	0.38	0.35	60
5	0.31	0.26	0.28	58
6	0.08	0.05	0.06	57
7	0.37	0.35	0.36	57
8	0.44	0.43	0.43	56
9	0.16	0.14	0.15	56
Accuracy			0.23	634
Macro Avg	0.24	0.24	0.24	634
Weighted Avg	0.23	0.23	0.23	634

Table 4: Random Forest Classifier with PCA Evaluation Metrics

4.3 Convolutional Neural Network

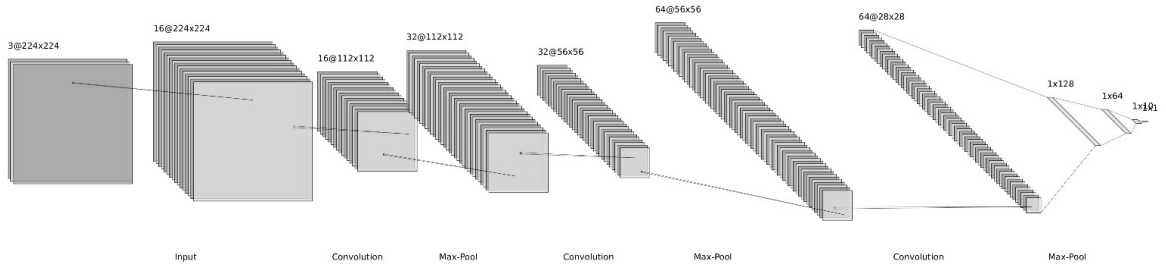


Figure 9: Architecture of the main CNN model.

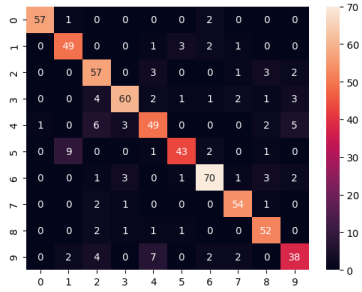
The structure of the main CNN model is explained as given below:

- Inherits from `nn.Module`, a base class for all neural network modules in PyTorch.
- Consists of 3 convolutional blocks each consisting of:
  - A convolutional layer (`nn.Conv2d`) that extracts features from the images.
  - Batch normalization (`nn.BatchNorm2d`) to stabilize and accelerate training.
  - A ReLU activation function for non-linearity.
  - A max-pooling layer (`nn.MaxPool2d`) to reduce the spatial dimensions of the output.
- `self.flatten` flattens the output of the last convolutional block to feed it into fully connected layers.
- Fully connected layers (`self.fc_layers`) are a series of linear layers and ReLU activations to perform the final classification. The last layer outputs 10 values, corresponding to the number of bird species.

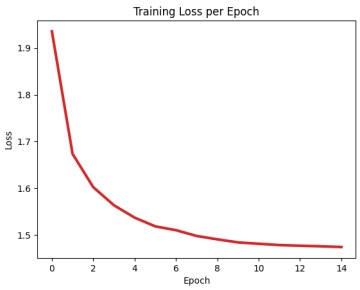
- A sigmoid activation function is applied to the output.

Model 1 in the following section is based on the architecture described above. Now, three different CNN models are trained on RGB datasets with augmentation and without augmentation. Validation evaluation metric and confusion matrix for test data set are given above. Additionally model descriptions can be found in Table 3 and Table 4.

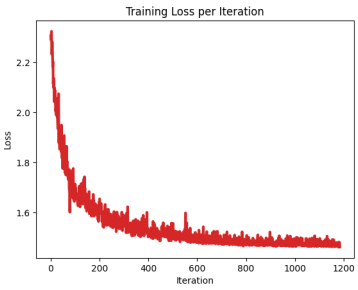
4.3.1 Model 1



(a) Confusion matrix (%84.10 Accuracy)

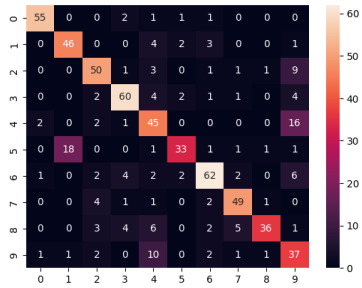


(b) Training Loss per Epoch

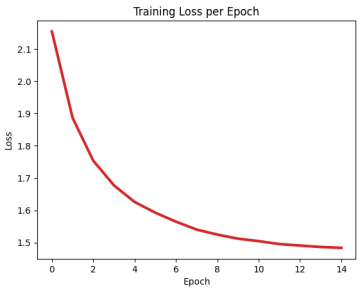


(c) Training Loss per Iteration

Figure 10: Validation results for Model 1 with augmented data



(a) Confusion matrix (%75.20 Accuracy)

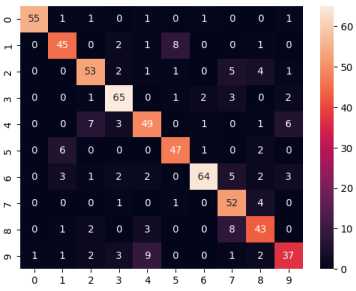


(b) Training Loss per Epoch

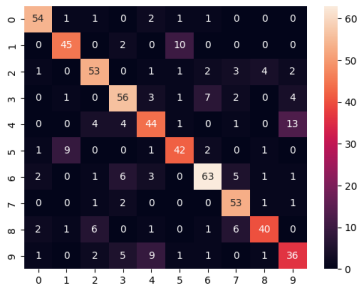
(c) Training Loss per Iteration

Figure 11: Validation results for Model 1 with non-augmented data

Test results of the model are given here for each dataset:



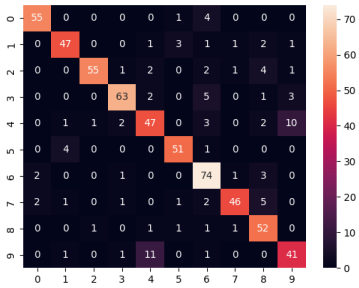
(a) Confusion matrix (%80.44 Accuracy)



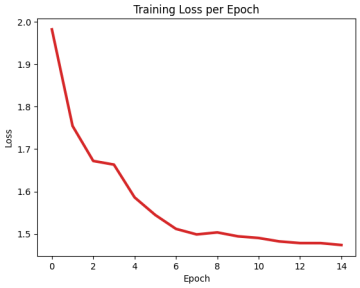
(b) Confusion matrix (%76.66 Accuracy)

Figure 12: Test results for Model 1 with non-augmented data

4.3.2 Model 2



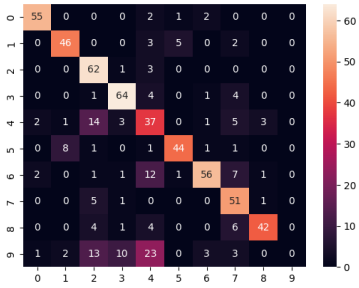
(a) Confusion matrix (%84.42 Accuracy)



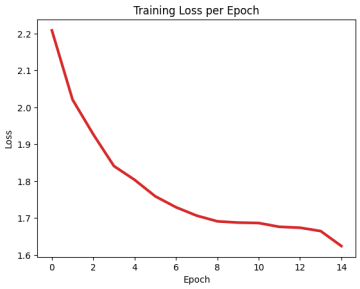
(b) Training Loss per Epoch

(c) Training Loss per Iteration

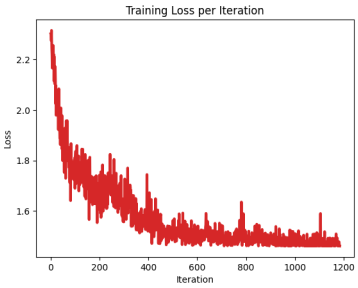
Figure 13: Validation results for Model 2 with augmented data



(a) Confusion matrix (%72.66 Accuracy)



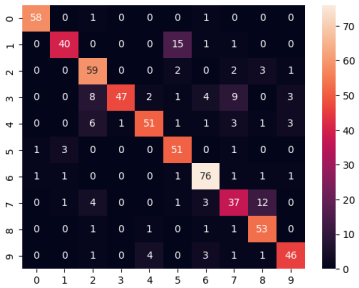
(b) Training Loss per Epoch



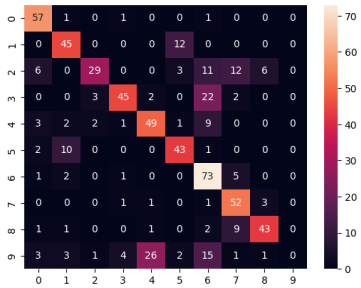
(c) Training Loss per Iteration

Figure 14: Validation results for Model 2 with non-augmented data

Test results of the model are given here for each dataset:



(a) Confusion matrix (%81.70 Accuracy)



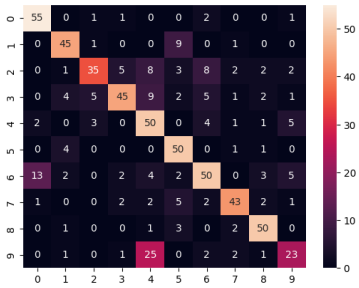
(b) Confusion matrix (%68.77 Accuracy)

Figure 15: Test results for Model 2 with non-augmented data

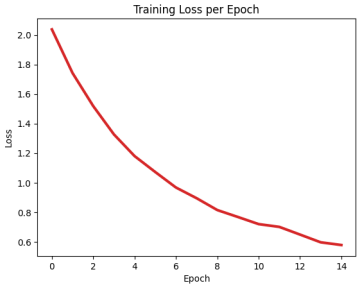
Layer Type	**Details**
Input Layer	Three input channels for RGB images.
Convolutional Block 1	• Conv: 32 channels, 3x3 kernel, padding 1 • Batch Norm • ReLU • Max-Pool: 2x2 kernel, stride 2
Convolutional Block 2	• Conv: 64 channels, 3x3 kernel, padding 1 • Batch Norm ReLU • Max-Pool: 2x2 kernel, stride 2
Convolutional Block 3	• Conv: 128 channels, 3x3 kernel, padding 1 • Batch Norm ReLU • Max-Pool: 2x2 kernel, stride 2
Convolutional Block 4	• Conv: 256 channels, 3x3 kernel, padding 1 • Batch Norm ReLU • Max-Pool: 2x2 kernel, stride 2
Flatten Layer	Flattens the output for fully connected layers.
Fully Connected Layers	• Linear 1: 256 * 14 * 14 input features, 512 output features, ReLU • Linear 2: 512 input, 256 output, ReLU • Linear 3: 256 input, 128 output, ReLU • Linear 4: 128 input, 10 output
Activation Function	Softmax applied to the final linear layer's output.
Output	Final output tensor with 10 values (assuming 10 classes) representing class probabilities.

Table 5: Summary of CNN Model 2 Architecture

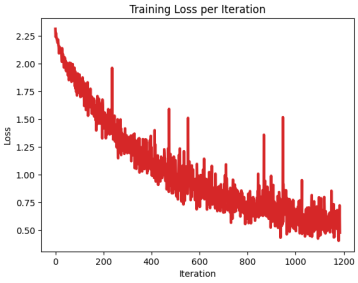
4.3.3 Model 3



(a) Confusion matrix (%70.91 Accuracy)

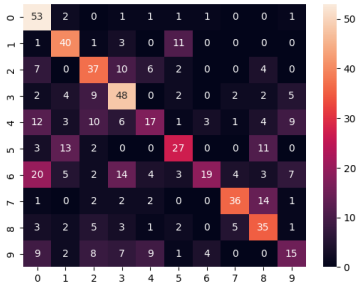


(b) Training Loss per Epoch

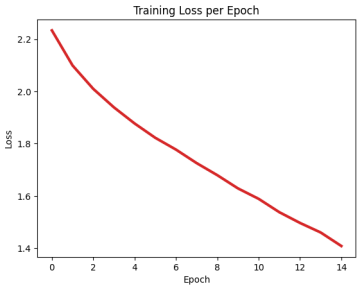


(c) Training Loss per Iteration

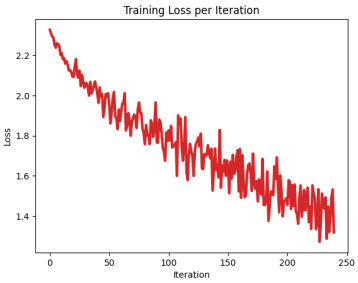
Figure 16: Validation results for Model 3 with augmented data



(a) Confusion matrix (%51.99 Accuracy)



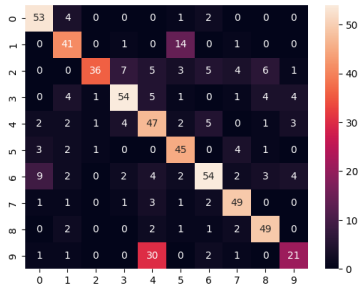
(b) Training Loss per Epoch



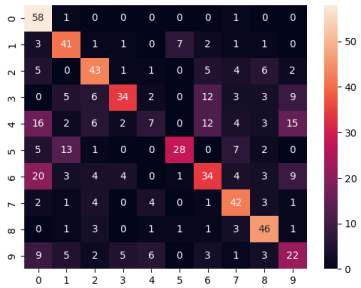
(c) Training Loss per Iteration

Figure 17: Validation results for Model 3 with non-augmented data

Test results of the model are given here for each dataset:



(a) Confusion matrix (%70.82 Accuracy)



(b) Confusion matrix (%55.99 Accuracy)

Figure 18: Test results for Model 3 with non-augmented data

Layer Type	Details
Input Layer	Three input channels for RGB images.
Convolutional Block 1	• Conv: 16 channels, 3x3 kernel, padding 1 • Batch Norm ReLU • Max-Pool: 2x2 kernel, stride 2
Convolutional Block 2	• Conv: 32 channels, 3x3 kernel, padding 1 • Batch Norm ReLU • Max-Pool: 2x2 kernel, stride 2
Convolutional Block 3	• Conv: 64 channels, 3x3 kernel, padding 1 • Batch Norm ReLU • Max-Pool: 2x2 kernel, stride 2
Convolutional Block 4	• Conv: 128 channels, 3x3 kernel, padding 1 • Batch Norm ReLU • Max-Pool: 2x2 kernel, stride 2
Global Average Pooling	Applied using nn.AdaptiveAvgPool2d(1), reducing spatial dimensions to 1x1
Flatten Layer	Output of global average pooling is reshaped into a 1D tensor for fully connected layers
Fully Connected Layers	• Linear 1: 128 input features, 64 output features, ReLU • Linear 2: 64 input features, 10 output features
Output	Final output tensor with 10 values (assuming 10 classes) representing class probabilities

Table 6: Summary of CNN Model 3 Architecture

4.4 Transfer Learning

Transfer learning is using a pre-trained neural network on one task and adapting it for a new task. This approach is particularly useful when the amount of labeled data available for the target task is limited. The basic idea behind transfer learning is that the knowledge acquired by a model from solving one problem can be beneficial for solving another related problem. Instead of training a neural network from scratch on the target task, transfer learning allows the model to inherit and fine-tune the features and representations learned during the training on an often larger dataset. In our project, we have developed two models with transfer learning using EfficientNet-B0 and ResNet50 as our base models.

EfficientNet-B0 is the baseline pre-trained model of the EfficientNet family which are known for their efficiency in terms of both computational resources and model size while achieving state-of-the-art performance in various computer vision tasks, such as image classification and object detection [3]. In order to adopt the pre-trained EfficientNet-B0 to our image classification task with 10 classes, we send the output from the pre-trained model to a GlobalAveragePooling2D Layer and a Linear Layer (Dense)

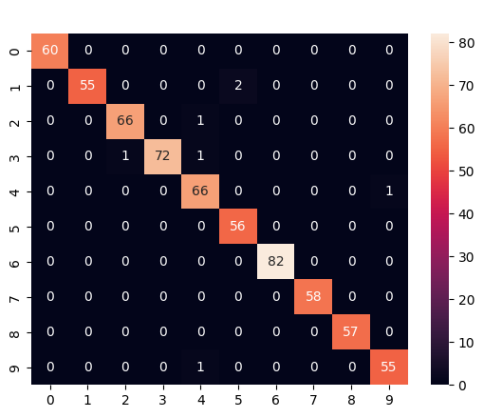
with 10 units.

ResNet50 is the middle-ground model of the Residual Networks family, in terms of number of layers within its structure, 50. The term "Residual" refers to the use of residual blocks, which are designed to address the vanishing gradient problem in deep neural networks. Residual blocks introduce skip connections, also known as shortcut connections or identity mappings, that allow the network to learn residual functions. This enables the training of very deep networks without degradation in performance.[2]

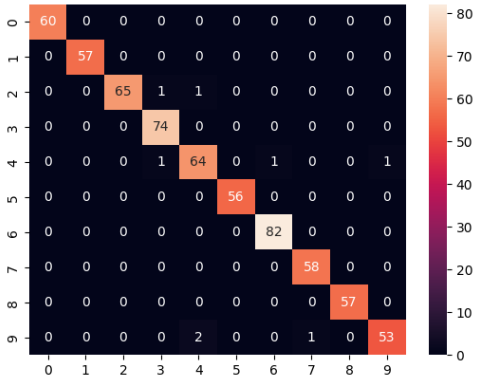
To adapt the pre-trained ResNet50 to our image classification task, we send the output from the pre-trained model to a GlobalAveragePoolin2D Layer, a Dense Layer with 1024 units, and a last Dense Layer for output with 10 units.

For both EfficientNet-B0 and ResNet50 models, we have used early stoppings to ensure that we maintain the best model in the case of an overfitting. For most cases, this approach helped us to prevent overfitting.

We have experimented with 8 different models, 4 for EfficientNetB0 and 4 for ResNet50, and each architecture with RGB, Grayscale, RGB Augmented, and Grayscale Augmented Dataset. Here is the confusion matrix and test accuracies for this experiment:

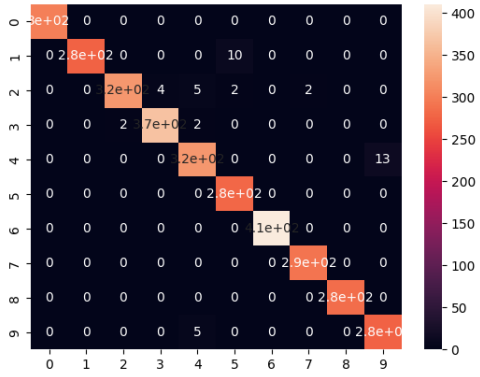


(a) EfficientNetB0 Confusion Matrix with RGB Trained

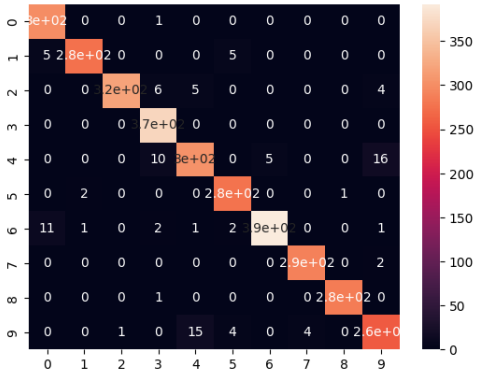


(b) EfficientNetB0 Confusion Matrix with Augmented RGB Trained

Figure 19: EfficientNetB0 Confusion Matrices

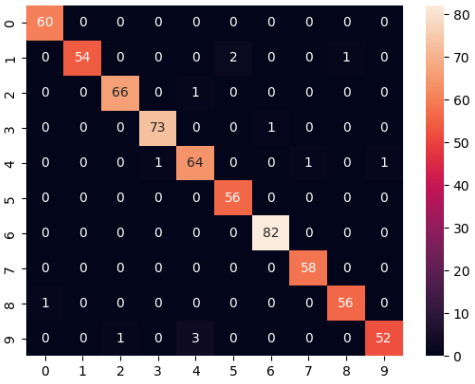


(a) EfficientNetB0 Confusion Matrix with Grayscale Trained

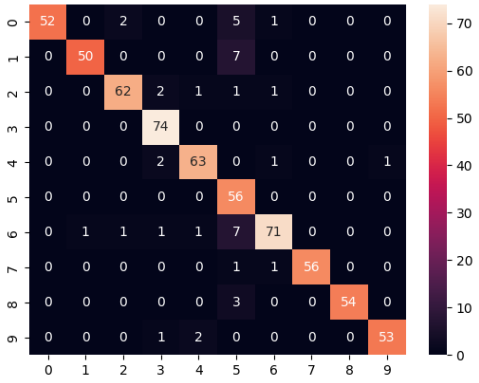


(b) EfficientNetB0 Confusion Matrix with Augmented Grayscale Trained

Figure 20: EfficientNetB0 Confusion Matrices (continued)

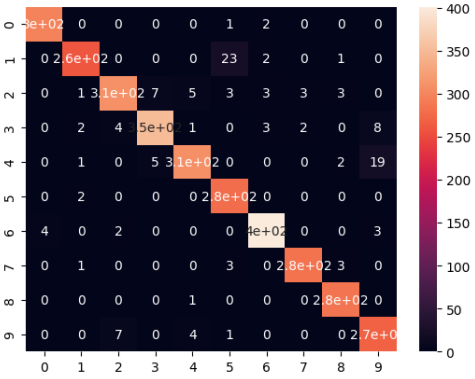


(a) ResNet50 Confusion Matrix with RGB Trained

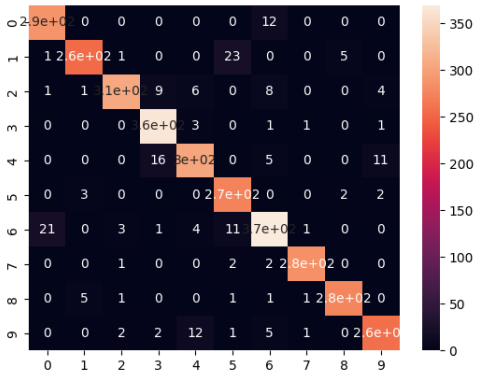


(b) ResNet50 Confusion Matrix with Augmented RGB Trained

Figure 21: ResNet50 Confusion Matrices



(a) ResNet50 Confusion Matrix with Grayscale Trained



(b) ResNet50 Confusion Matrix with Augmented Grayscale Trained

Figure 22: ResNet50 Confusion Matrices (continued)

Additionally, performance summary of the ResNet50 and EfficientNetB0 models are given below:

	RGB	Grayscale	RGB - Augmented	Grayscale - Augmented
EfficientNetB0	0.9826	0.9810	0.9810	0.9653
ResNet50	0.9763	0.9432	0.9621	0.9416

Figure 23: Summary of ResNet50 and EfficientNetB0 models

5 Challanges Encountered

During the course of the project, several challenges were encountered, reflecting the inherent complexities of bird species image classification. These challenges included addressing issues related to dataset variability, imbalanced class distribution, fine-grained distinctions between visually similar species, and optimizing model performance for diverse bird images captured under different environmental conditions. Additionally, the incorporation of advanced techniques such as transfer learning and data augmentation posed both opportunities and challenges, requiring careful consideration and experimentation to achieve optimal results in this intricate task of avian image classification.

6 Coding Environment

The project was conducted in a Python environment, leveraging a wide range of libraries for data management, machine learning, and deep learning tasks. The key libraries and their uses are as follows:

- **os and pathlib:** For interacting with the operating system, facilitating file and directory operations.
- **cv2 (OpenCV):** Employed for image reading, processing, and augmentation tasks.
- **numpy (np):** Utilized for efficient numerical operations and array manipulations.
- **pandas (pd):** Used for data manipulation and analysis since it offers advanced data structures.
- **matplotlib (plt) and seaborn (sns):** For creating visualizations, including plots and graphs.
- **PIL (Python Imaging Library):** Employed for opening, manipulating, and saving images.
- **torch and torchvision:** Central libraries for building and training deep learning models, including Convolutional Neural Networks (CNNs), and for data augmentation and transformations.
- **sklearn (Scikit-learn):** Leveraged for machine learning tasks, including model training, evaluation, and preprocessing.
- **PyTorch models (torchvision.models):** Used for importing pre-built model ResNet50.
- **Keras (part of TensorFlow):** Employed for building and training deep learning models, especially for transfer learning models, EfficientNet-B0 and ResNet50.
- **ImageDataGenerator (part of TensorFlow):** Used for real-time data augmentation during model training.

In addition to these libraries, we integrated functionalities for connecting Kaggle and Google Drive, enabling efficient data retrieval and storage.

7 Workload Distribution of Team Members

Member	Contributions
Ömer Asım Doğan	• Wrote the progress report. • Helped with the slides. • Wrote the Final Report
Gökay Balcı	• Data preprocessing. • Progress report writing and presentation. • CNN model experiments and visualization. • Final Presentation assistance.
Selin Ataş	• Data preprocessing. • Enhanced train-validation-test split. • Troubleshooted preprocessing issues. • Regenerated the CNN model. • Final report writing and presentation.
Mennan Gök	• SVM, Random Forest, and part of CNN model generation. • Progress report assistance. • Enhanced train-validation-test split. • Troubleshooted preprocessing issues. • Transfer Learning models for EfficientNet-B0 and ResNet50.
Ömer Tuğrul	• Progress report assistance. • Prepared progress presentation. • SVM, Random Forest, and part of CNN model generation. • Troubleshooted preprocessing issues. • Added PCA algorithm and data augmentation. • Final report assistance.

Table 7: Team Members' Contributions

References

References

[1] GPIosenka. "Birds 525 Species- Image Classification." Kaggle. Available at: https://www.kaggle.com/datasets/gpiosenka/100-bird-species.

[2] Shafiq, Muhammad, and Zhaoquan Gu. "Deep Residual Learning for Image Recognition: A Survey." Applied Sciences 12, no. 18 (September 7, 2022): 8972. https://doi.org/10.3390/app12188972.

[3] Wang, Liyuan, Yulong Chen, Xiaoye Wang, Ruixing Wang, Hao Chen, and Yinhai Zhu. "Research on Remote Sensing Image Classification Based on Transfer Learning and Data Augmentation." Knowledge Science, Engineering and Management, 2023, 99–111. https://doi.org/10.1007/978-3-031-40292-0_9.