COS30082

BIRD SPECIES CLASSIFICATION

Video: https://youtu.be/AOtr6FbZTAE
Hugging Face: https://huggingface.co/spaces/brxerq/Bird_Classification

ASSIGNMENT 1

SYED MUHAMMAD HASSAAN BIN GHAYAS
101231186

Background:

Multi-class classification serves as a foundation in both machine learning and computer vision. It aims to categorize incoming data into one of several pre-established classes. This report focuses on employing diverse strategies to tackle a multi-class classification task using the Caltech-UCSD Birds 200 (CUB-200) dataset. The training images consist of 4829 number of pictures each of them is divided into 200 classes of bird species. Given the small number of samples per class, the propensity for overfitting is high. To mitigate this, the report delves into a range of models and techniques aimed at minimizing overfitting.

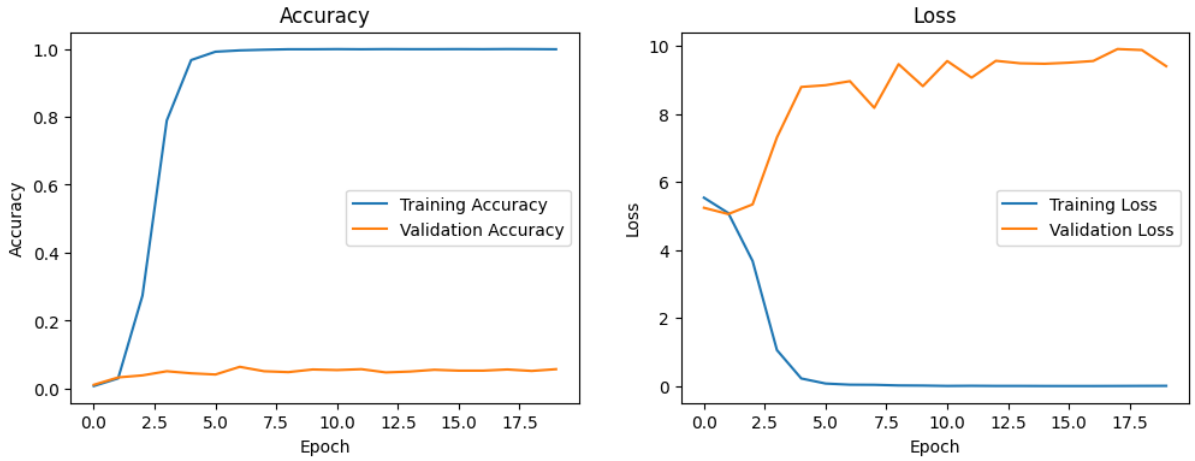
Methodology

I will implement a basic CNN model to set a foundational benchmark. Observing its performance, I will transition to the MobileNetV3 architecture, anticipating improved test accuracy. I will introduce data augmentation techniques to address overfitting and enhance the model's generalisation. I will also integrate learning rate scheduling to ensure optimal convergence during training. Finally, I aim to fine-tune the MobileNetV3 model by making its layers trainable and adjusting the learning rate to achieve a balance between training and validation outcomes.

Data preparation used the CUB-200 dataset containing 4829 training images of 200(classes) bird species. Images were loaded from a zip file and extracted to a temp location. Preprocessing Images were resized to 224x224 (can extract more features) and normalized ([0,1]).

1. CNN Model:

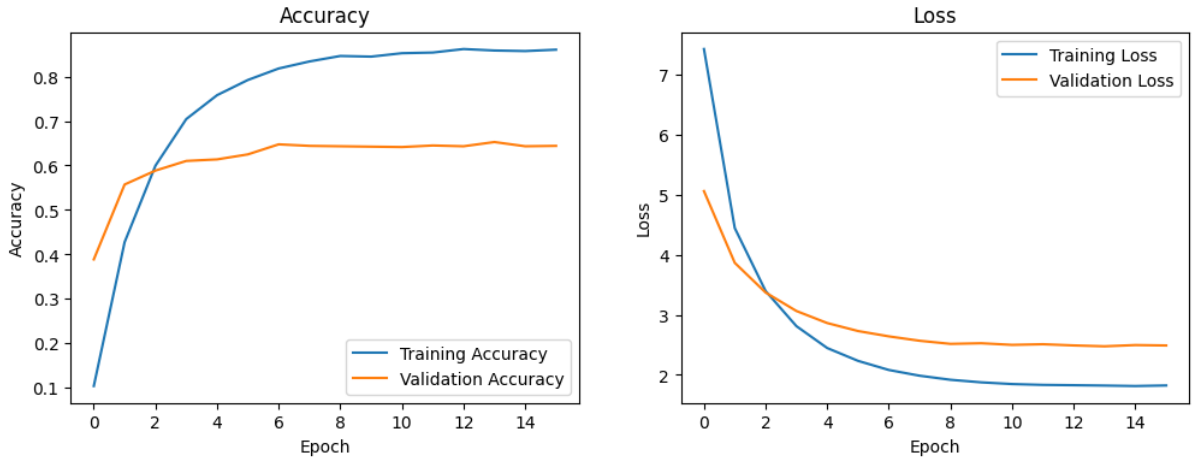
In my model, I've implemented a simple Convolutional Neural Network (CNN) with key layers: two Conv2D layers (32 and 64 filters, 3x3 kernel, ReLU), MaxPooling2D layers (2x2 pool size), Flatten, and two Dense layers (64 units, ReLU, and 200 units with softmax for 200 bird species in the dataset).



In analysing the base model results, I observed that the training accuracy reaches almost 71%, showcasing a strong learning capability. However, the validation accuracy plateaus at around 1.91%, highlighting a potential overfitting issue. The model has learned the training data complications but struggles to generalize effectively on unseen data.

2. MobileNetV3:

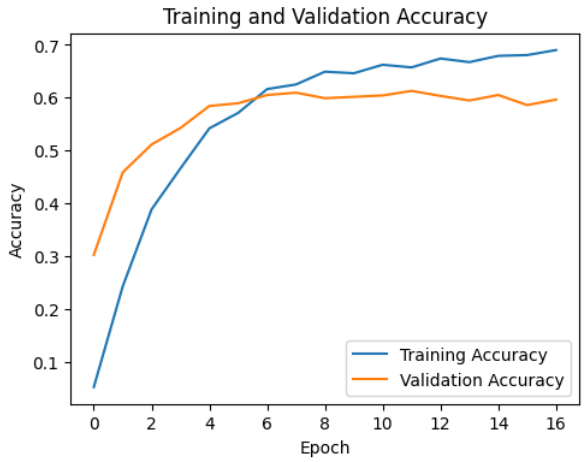
I use the MobileNetV3 architecture for the second, powerful model pre-trained on the ImageNet dataset. Instead of training the entire architecture, I imported its weights from TensorFlow Hub and set them as non-trainable, allowing me to harness the pre-learned features and adapt them to my specific task. I refined the architecture by adding a dropout layer with a rate of 0.5 to reduce overfitting, followed by a dense output layer containing 200 classes. These outputs layer also employs L2 regularization for added stability. I used the Adam optimizer with a learning rate 0.001 to train the model and used categorical cross-entropy as the loss function.

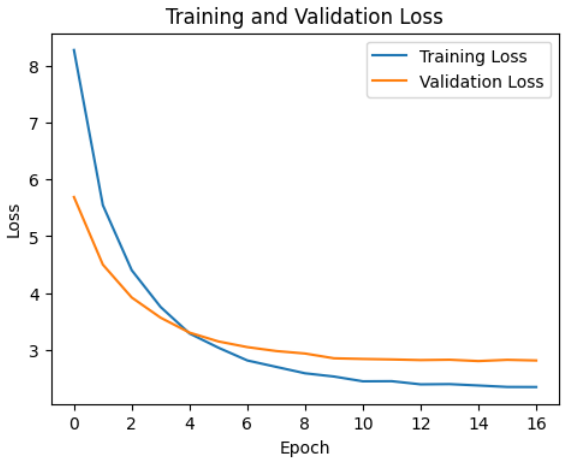


Analyzing the results of my second model, I observed the following: The training accuracy reached an impressive 81.21%, which aligns with a training loss of approximately 2.16. When I evaluated the model using the test data, the accuracy dropped significantly to 59.55%, and the loss increased to approximately 2.83. The graphs further solidify these observations. In graph, the training loss sharply decreases but the validation loss flattens, indicating potential overfitting.

3. MobileNetV3 using Data Augmentation:

In the third model, data augmentation techniques, including flips, rotations, and translations, are incorporated to enhance generalization. Built on the MobileNetV3 base, the model includes a dropout layer and a regularized dense output layer with 200 units. It's compiled with the Adam optimizer and categorical cross-entropy loss and trained using early stopping based on validation performance.

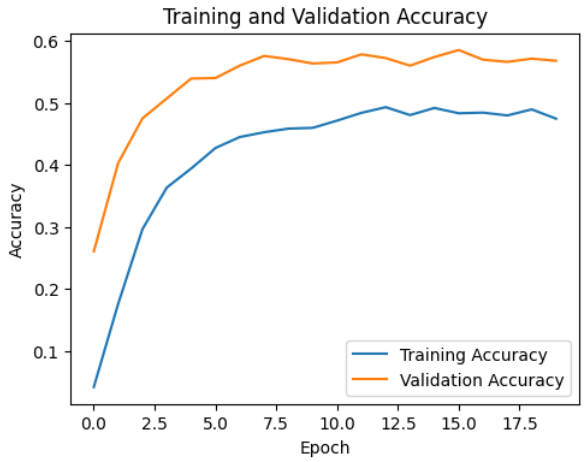


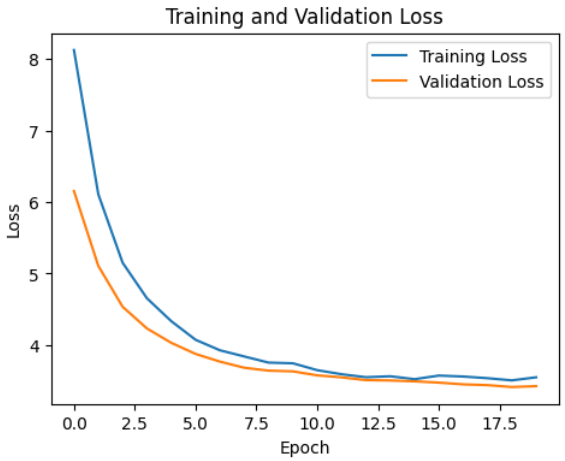


Based on the data I've observed, it's clear that the model is overfitting. The training accuracy I achieved is 78.54%, substantially higher than the validation accuracy of 57.38%. Furthermore, my training loss of 2.88 is much lower than the validation loss of 3.44. The difference is also evident in the graphs, where a growing gap between training and validation metrics is noticeable as training progresses.

4. MobileNetV3 (Data Augmentation and Learning Rate Scheduling):

In the fourth model, I merged my old data augmentation with a learning rate schedule that exponentially declines the learning rate over epochs. Furthermore, I increased the amount of dropout layers in the model. The model was compiled with the Adam optimiser, a categorical cross-entropy loss function, and metrics including top-1 accuracy. Adding regularisation techniques and data augmentation enhances the model's generalisation to validate data.

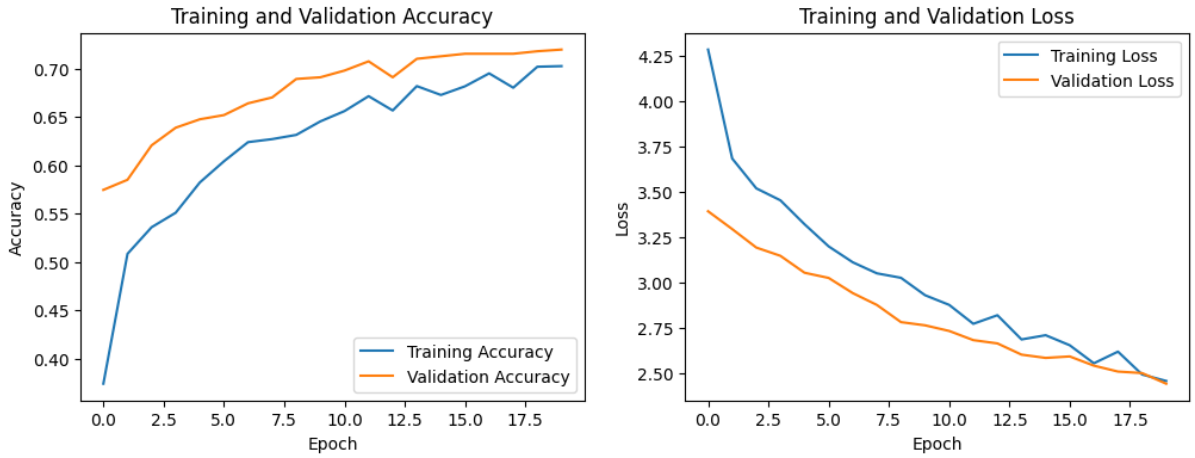




The graphs and results for the fourth model show that the training accuracy reached 78.54%, and the validation accuracy is 57.38%, indicating a discrepancy between training and validation performance. This suggests the model might still be overfitting, even though the gap is smaller than in some previous models. Comparing this with the third model's performance, the fourth model has countered the overfitting issue to an extent, but not entirely.

5. Fine-tune the 4th Model:

I decided to fine-tune the fourth model by setting the MobileNetV3 layers to be trainable. By allowing these layers to be adjusted during training, I hoped to get a better performance from the model. Using the Adam optimiser, I recompiled the model with a much lower learning rate of 0.0001. This lower learning rate would ensure that the previously learned features aren't drastically changed but are fine-tuned for better performance. I trained the model again with the same settings but with the added flexibility of trainable MobileNetV3 layers. Early stopping ensured the model wouldn't continue training if no significant improvements were observed in validation metrics over two consecutive epochs, preventing potential overfitting and saving computational resources.



Results after fine-tuning, I observed a remarkable improvement in training accuracy, reaching 96.81%. This indicates that the model has learned the training data exceptionally well. The training loss also dropped to 1.4056, a positive sign of model optimisation. The accuracy improved to 64.41% on the validation side, a notable enhancement compared to the previous results. However, there remains a significant gap between training and validation metrics, indicating potential overfitting, although the gap has reduced compared to earlier. The fine-tuning has indeed boosted the model's performance.

Results and Comparison:

Model Number	Model Description	Training Loss	Training Accuracy	Training Top K Accuracy	Test Loss	Test Accuracy	Test Top K Accuracy
1	**CNN Model**	1.16	70.71%	70.71%	16.00	1.91%	1.91%
2	**MobileNetV3**	1.57	94.71%	94.71%	2.50	64.32%	64.32%
3	**MobileNetV3 using Data Augmentation**	2.17	81.21%	81.21%	2.83	59.55%	59.55%
4	**MobileNetV3 (Data Augmentation and Learning Rate Scheduling)**	2.88	78.54%	78.54%	3.44	57.38%	57.38%
5	**MobileNetV3 (Data Augmentation, Rate Scheduling) with Finetuning**	1.45	96.81%	96.81%	2.55	64.41%	64.41%

In the presented graph, I've visualised the performance of deep learning models by plotting their training and test accuracies. The graph contrasts the results from a basic CNN Model to multiple versions of MobileNetV3, highlighting the progression of accuracies with different techniques like data augmentation, learning rate scheduling, and fine-tuning.

From the table, it's evident that the CNN model showed high accuracy on training data but performed poorly on the test data, indicating a clear case of overfitting. On the other hand, the fourth model achieved high training and test accuracies, suggesting effective regularisation and generalisation.

Average Accuracy per Class:

Since I forgot about the average accuracy per class in every model then I used it and updated it in few models' results are shown in the chart. Model 5 was the best in terms of accuracy but the model that performed well overall and was able to mitigate the overfitting problem was model 4.

Model Number	Model Description	Average Accuracy Per Class (Training)	Average Accuracy Per Class (Validation)	Difference
1	**CNN Model**	69.41%	1.73%	67.68%
4	**MobileNetV3 (Data Augmentation and Learning Rate Scheduling)**	78.21%	56.36%	21.85%
5	**MobileNetV3 (Data Augmentation, Rate Scheduling) with Finetuning**	96.81%	64.41%	32.4%