CS4641 Group 117

Final Report

Introduction

The use of strategically deployed camera traps has allowed the collection of a vast amount of wildlife imagery which serves as an enabler for exploration in ecological research and conservation efforts. However, manually annotating and classifying such large datasets can be inefficient and prone to human error. Depending on the scale of the camera trap deployment and overall operation, it may even be unfeasible. For our ML group project, we chose to leverage ML in an attempt to automate the classification of wildlife species from camera trap images from a given dataset in order to mitigate this problem and support wildlife conservation efforts.

We are using a massive dataset containing approximately 2.65M sequences of camera trap images, totalling 7.1M images from the Snapshot Serengeti project [1], which can be accessed here. This dataset contains images from camera traps in the Serengeti National Park. Labels are provided for 61 categories of animal species, primarily distinguishing at the species level, but including broader and narrower classifications on the basis of frquency. For example, more common species are even subcategorized based on gender. The goal of our project is to use Machine Learning on this given labelled image data to classify different animal species in unseen images with a high degree of correctness.

The problem of animal species classification is a well researched one. Classical machine learning methods such as support vector machines and multi layer perceptrons have been applied to the task of animal classification with some degree of success [3][5]. More recently, Convolutional Neural Networks have emerged as ideal architectures for the purposes of image processing tasks, making them a promising approach for our purpose. Current research has applied the idea of Neural networks to datasets such as the Snapshot Serengeti to acheive impressive results [4]. Our aim in this project is to further explore both classical machine learning approaches such as K-nearest neighbours and Multi-layer perceptrons along with state of the art architectures such as Convolutional Neural Networks for the task of animal classification.

Problem Definition

The major problem with camera trap datasets such as the Snapshot Serengeti is that the sheer size and scale of the project as well as the resulting data makes it prohibitively expensive to manually label all classes of wildlife associated with a certain picture. However, this data is infinitely valuable when it comes to dedicated conservation efforts to track the patterns and distribution of various species across important ecological hotspots such as the Serengeti, coalescing wildlife conservation efforts in these areas and effectively allocating resources in large-scale wildlife conservation projects.

This is what motivates our project. The main idea is to use common approaches in classical and modern machine learning and apply them to the problem of species classification on a sample conservation project such as the Serengeti. For this purpose, we took data from Snapshot Serengeti that was already labelled with the correct species and attempted to apply machine learning techniques in order to train a model that could perform reasonably well on classification. We use various metrics to measure our performance and present our results in this report.

Methods

After brainstorming the different paths we could take to implement the model we narrowed down on using KNN, MLP and CNN, mainly because we wanted to use MLP as a baseline implementation before moving on to a refined CNN algorithm, and compare these advanced neural network based architectures with classical machine learning approaches such as K-nearest neighbours. Not only do we believe that CNN would be a better model for complex images (like we are dealing with in the data) but it would also provide a tangible visualization of different class-specific features learned through the kernels which would help us identify how CNN, an extension of the MLP algorithm is better for image processing and how the results fit within the context of varying performances accross different classes in the dataset. The following methods were used to implement our goal:

1. Data Processing

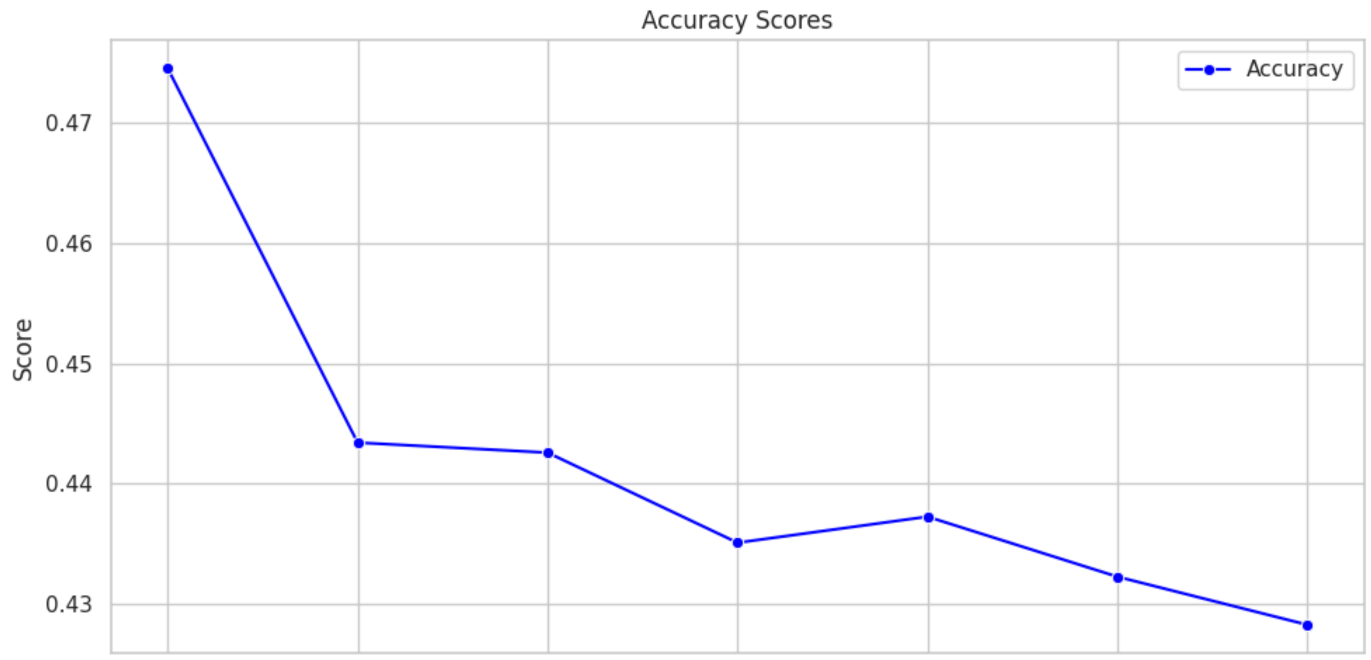
- Downsampling was employed to reduce the size of a dataset from the 7M images it has to about 30K in order to make processing and training possible. We ensured that we had around a thousand samples for each class, which helped us eliminate classes for which data was sparse.
- Average pooling was used for feature reduction (essentially resolution reduction on the images) as discussed in this paper [2]. The concept of average pooling is very old, but its application for feature reduction is something we believe to be novel.
- Data augmentation via random cropping had its share of pros and cons. Though it increased network robustness, especially in the case of CNN, it did not work as well as predicted because some of the crops after the augmentation did not contain the animals anymore.
- Data augmentation via image flipping enabled us to increase data diversity and expose the models to variations in image orientation, making them more robust and better classifiers overall.
- Data augmentation via greyscaling helped us in reducing any overfitting that might have been caused due to reliance on specific color patterns. Overall, we believe it helped the model rely more on shape and textures for classification instead.

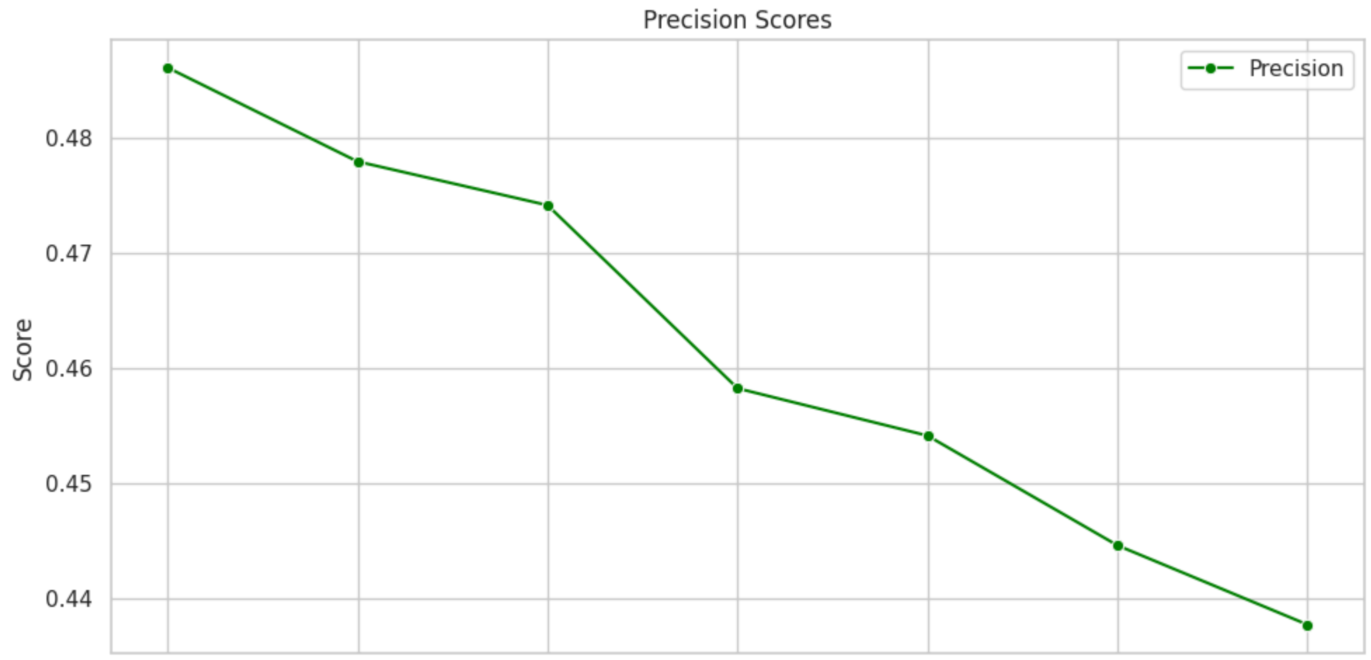
2. K-Nearest Neighbours Model (knn.ipynb)

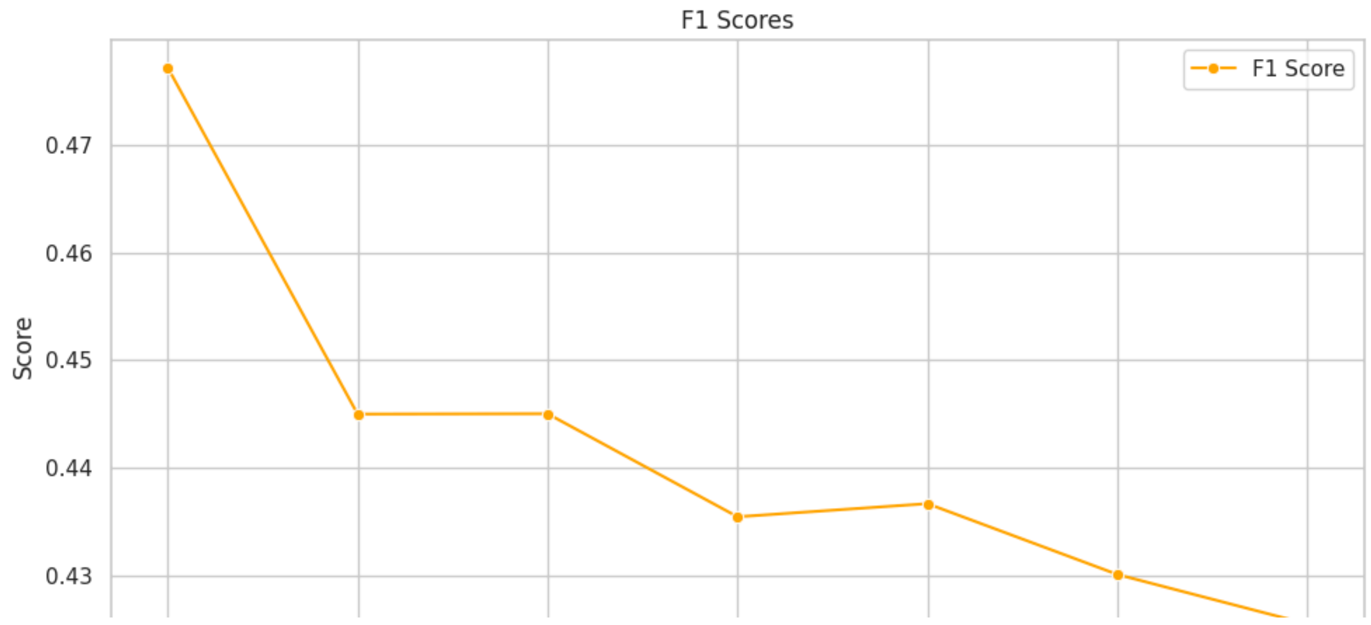
- This model loads the dataset and preprocesses it by converting the labels to indices.
- For choosing the value of k for our algorithm, we plotted testing accuracy and other quantitative metrics obtained with different k

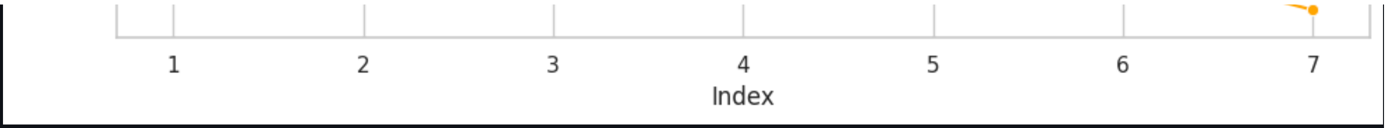
values to determine the optimal k value of 1. Please refer to the graphs below for a detailed presentation of the same.

- The evaluation section reports a summary of various metrics like accuracy, precision, recall, and F1 scores for the model.
- This model was chosen for its simplicity and computational inexpensiveness which enables us to experimnent with multiple values of k in order to enhance the effectiveness of the model [6]. Moreover, it acts as an example of a classical machine learning algorithm against which to compare more advanced neural network based models.


Accuracy Scores
Accuracy
Score
0.47
0.46
0.45
0.44
0.43


Precision Scores
Precision
Score
0.48
0.47
0.46
0.45
0.44


F1 Scores
F1 Score
Score
0.47
0.46
0.45
0.44
0.43



3. Multi Layer Perceptron Model (mlp.ipynb)

- This model loads the dataset and preprocesses it by normalizing the pixel values of the images and converting the labels to indices.
- For the MLP architecture, we have 3 hidden layers with 7500, 1500 and 300 neurons respectively and ReLU activation.
- The model is initialized with Xavier initialization and trained with the Adam optimizer and Cross Entropy Loss function.
- The evaluation section reports a summary of various metrics like accuracy, precision, recall, and F1 scores.
- This model was chosen for its demonstrated robustness on classification tasks [5] and to act as a baseline neural network model against which results from the CNN model could be compared.

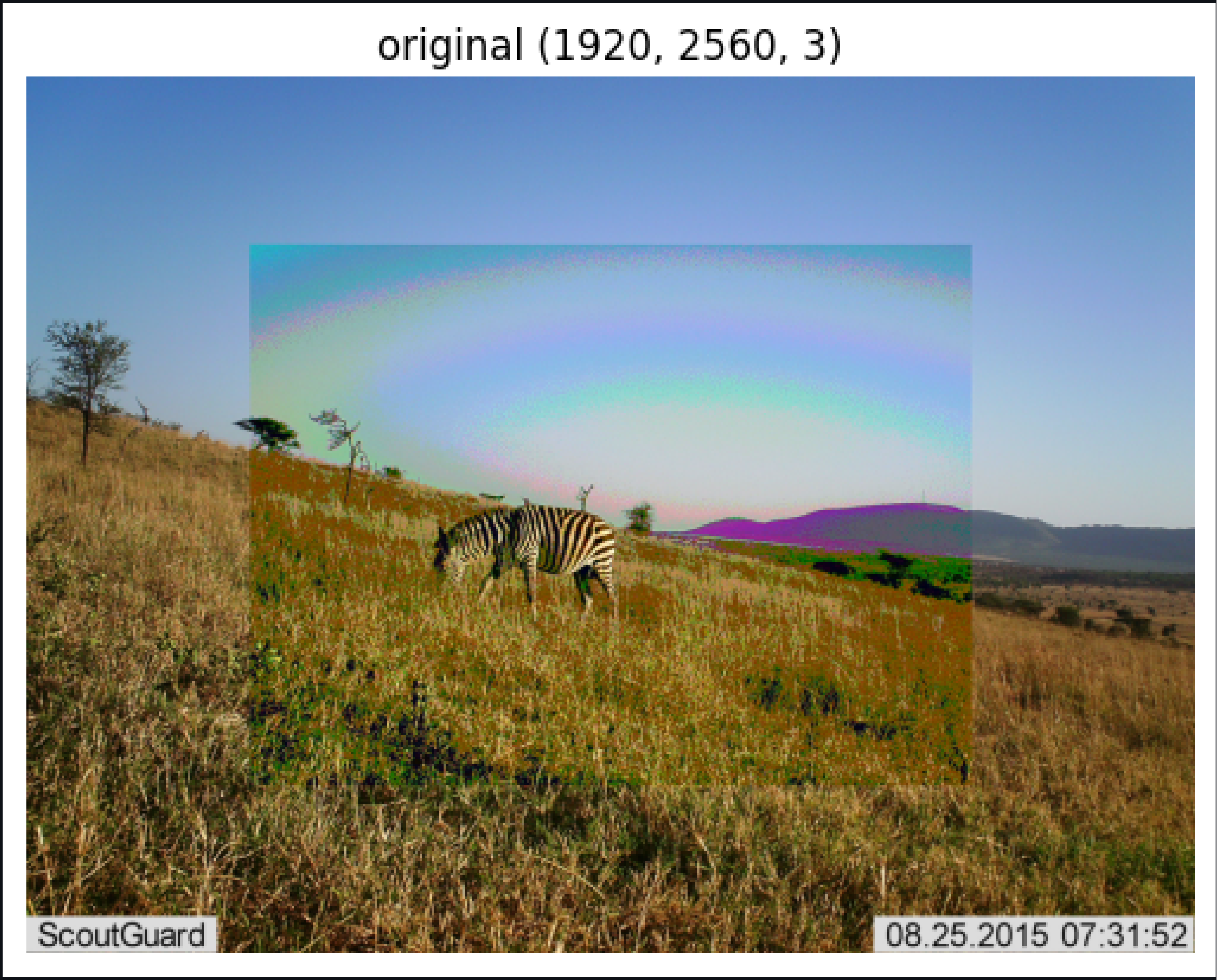
4. Convolutional Neural Network Model (cnn.ipynb)

- This model loads the dataset and preprocesses it by converting it to tensors using the PyTorch library and mapping labels to indices.
- We used a custom CNN architecture with 3 convolutional layers. Two convolutional layers with 32 and 64 3x3 kernels, followed by a MaxPool layer. This is essentially our feature extractor. This is followed by a 16 3x3 kernel layer that encodes the features into representations that are fed into a 3 hidden-layer dense network for classification.
- ReLU was chosen as the activation function due to its computational simplicity and widespread popularity.
- Similar to MLP, we used the Adam optimizer and cross-entropy loss function to train the model.
- We selected a batch size of 32 to meet our computational constraints.
- Similar to the MLP, evaluation comprises inspection of training loss and comparision of metrics like accuracy, precision, recall, and F1 scores.
- This model was chosen due to its ability to learn representations of spacial data like images [3] and as a natural advanced progression of the neural network paradigm

Results and Discussion

1. Data Processing

As discussed above, most of the images in the Snapshot Serengeti project were very high resolution images. A sample image is shown below. Performing average pooling on the image, along with random cropping gave us promising results that visually retained the qualities of the original image while reducing the number of features. The pooled and augmented version of the sample image is also shown for reference. We also present samples of the given image with our other data augmentations, namely image flipping and greyscaling to present the qualitative features of our final augmented dataset. Visual inspection suggests that the data preprocessing applied was successfull.



downsized (120, 160, 3)

Grayscale

Flipped

2. KNN Model

Our K Nearest Neightbours model does not perform quite as well as expected, as evidenced by the accuracy and Macro F1 scores. A more refined picture can be drawn by looking at the per class performances. A potential reason we see the KNN algorithm not perform so well is the high dimensionality of image data, leading to the curse of dimensionality problem with euclidian distance, where the distance between high dimensional vectors becomes so great that measures like closer and farther become less meaningful. These results seem to indicate that classical approaches may not be sophisticated enough to give good performance on image tasks such as ours and motivate the use of more advanced machine learning techniques such as MLP and CNN.

Overall Performance Metrics

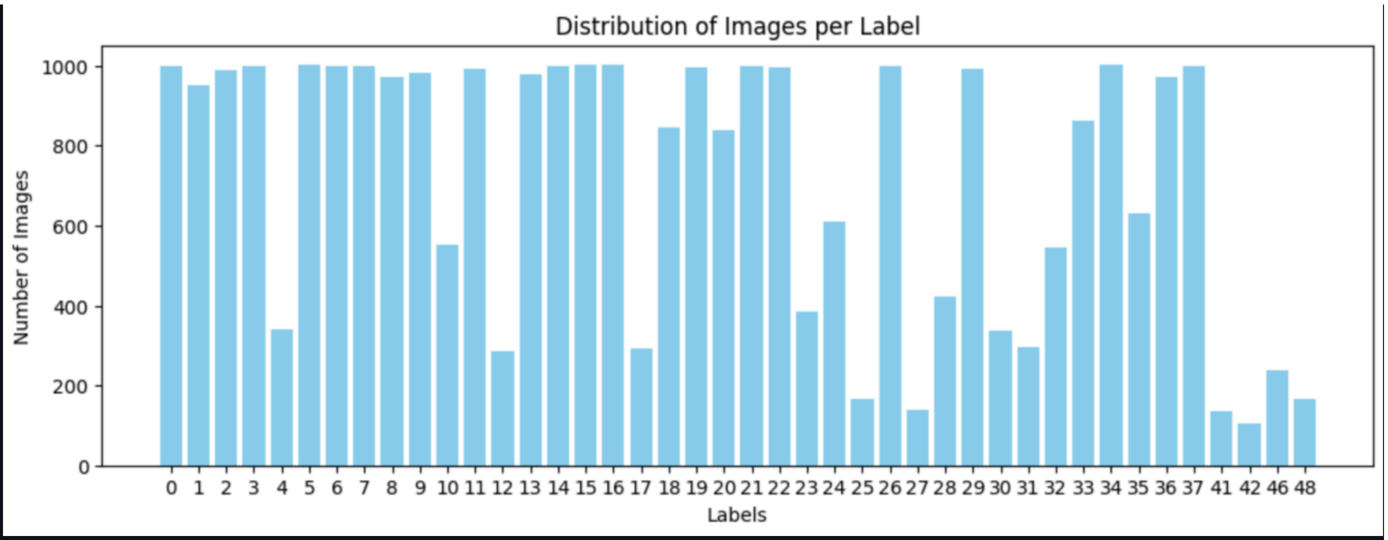
	Metric	Value
0	Accuracy	47.46%
1	Macro-F1 Score	0.48

Per Class Performance

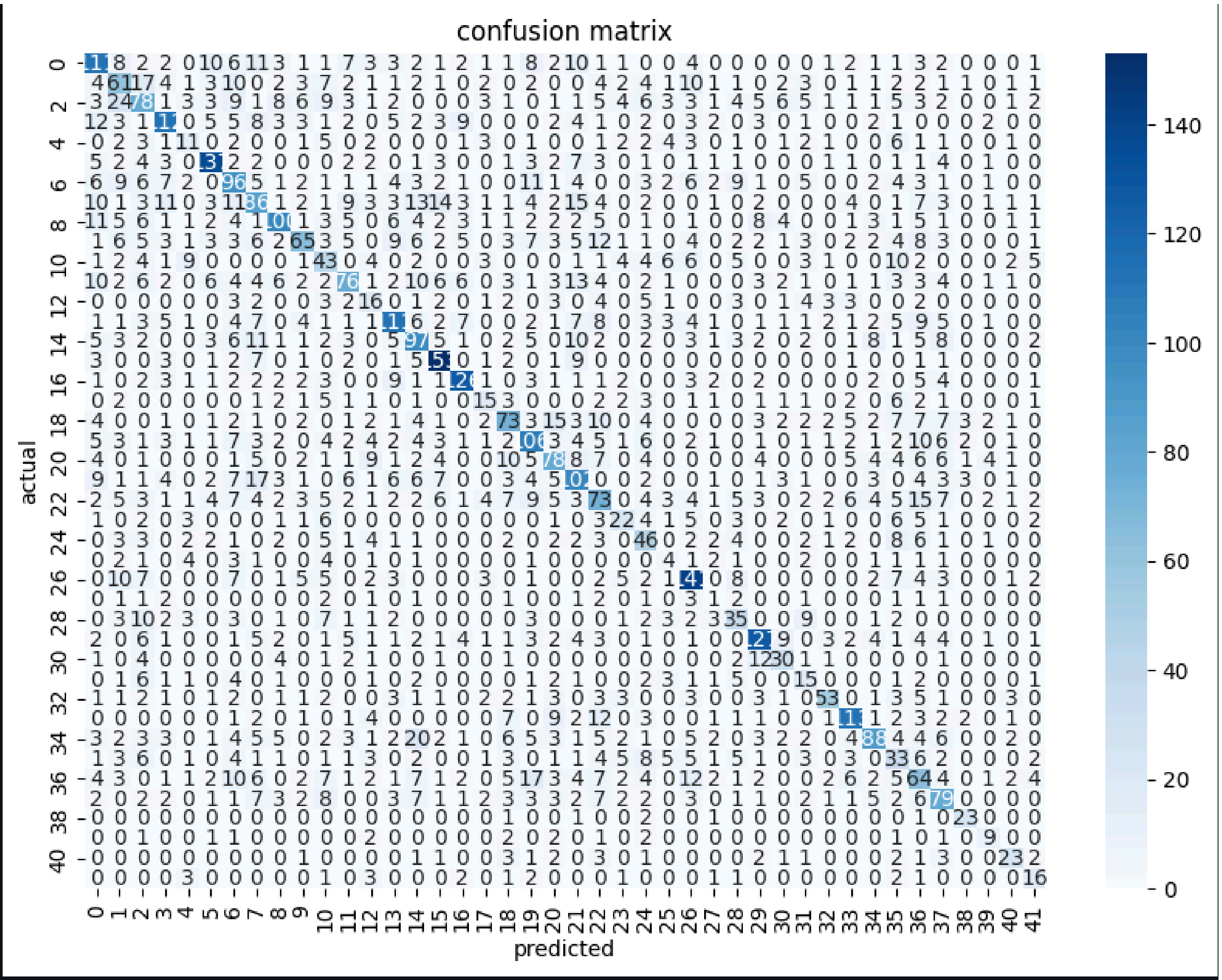
	Class Name	Precision	Recall	F1 Score
0	gazelleGrants	0.5000	0.5200	0.5100
1	reedbuck	0.3600	0.3800	0.3700
2	dikDik	0.3900	0.3600	0.3700
3	zebra	0.6200	0.5700	0.5900
4	porcupine	0.2200	0.1900	0.2000
5	gazelleThomsons	0.7100	0.7100	0.7100
6	hyenaSpotted	0.4100	0.4800	0.4400
7	warthog	0.4100	0.3800	0.4000
8	impala	0.6200	0.5100	0.5600
9	elephant	0.5600	0.3400	0.4200
10	aardvark	0.2800	0.3600	0.3200
11	giraffe	0.5100	0.4000	0.4500
12	mongoose	0.2000	0.2600	0.2300
13	buffalo	0.5900	0.5200	0.5500
14	hartebeest	0.4400	0.4800	0.4600
15	guineaFowl	0.6800	0.7900	0.7300
16	wildebeest	0.7200	0.6800	0.7000
17	leopard	0.3100	0.2700	0.2900
18	ostrich	0.4800	0.4200	0.4500
19	lionFemale	0.5000	0.5200	0.5100
20	koriBustard	0.4800	0.4400	0.4600
21	otherBird	0.4500	0.4900	0.4700
22	cheetah	0.3500	0.3300	0.3400
23	bushbuck	0.3500	0.3100	0.3300

	Class Name	Precision	Recall	F1 Score
24	jackal	0.3200	0.4200	0.3700
25	aardwolf	0.1000	0.1200	0.1100
26	hippopotamus	0.5800	0.6400	0.6100
27	hyenaStriped	0.0300	0.0400	0.0400
28	hare	0.3400	0.3700	0.3600
29	baboon	0.6500	0.6200	0.6400
30	vervetMonkey	0.4300	0.4800	0.4600
31	batEaredFox	0.2000	0.2900	0.2400
32	waterbuck	0.6800	0.5100	0.5800
33	secretaryBird	0.6600	0.6600	0.6600
34	topi	0.6100	0.4400	0.5100
35	serval	0.2200	0.2900	0.2500
36	lionMale	0.2900	0.3200	0.3100
37	eland	0.4400	0.4800	0.4600
38	reptiles	0.6600	0.7900	0.7200
39	insectSpider	0.3300	0.3900	0.3600
40	gazelleGrants	0.5100	0.4800	0.4900
41	hyenaSpotted	0.3300	0.4700	0.3900

An interesting aspect of the model to note is that the predictions made by the model seem to be biased towards those classes for which there exist more samples in the dataset, as can be confirmed by cross referencing the class frequencies shown below against classwise performace from the table above. This makes sense since having a larger number of samples for a particular class increases the probability of the model picking that class based on the neighbours of the test datapoint, and highlights the model sensitivity towards the distribution of training data.



Moreover, the performance of KNN can also be visualized with a confusion matrix that considers all the classes. We see a definite diagonal along the matrix, which implies that the model does have some classificationary power for the data, however, there are also off diagonal elements with non-insignificant element counts, indicating false predictions.



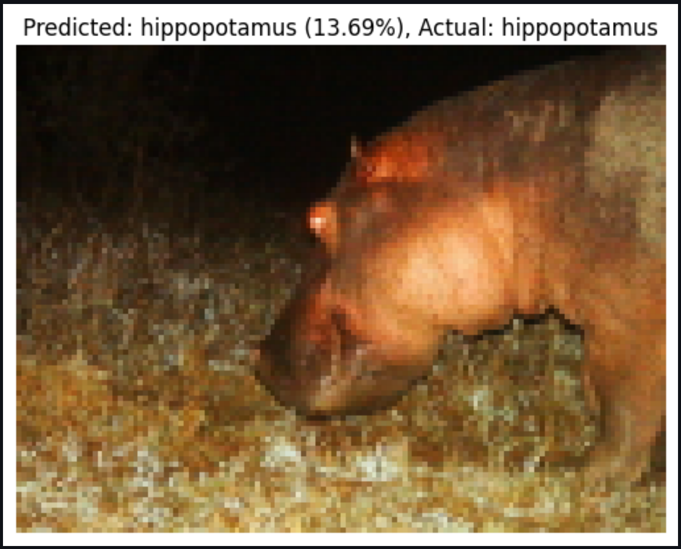
3. MLP Model

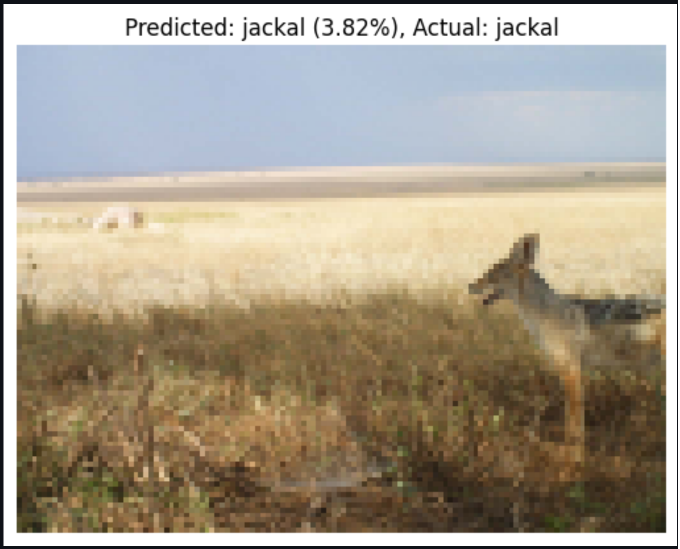
Our multi layer perceptron model showed reasonable performance on the dataset, with aggregate and classwise metrics as given below. All in all, the accuracy was as expected. Understandably, the MLP model struggles with the spacial nature of image data, which gets lost when the pixels are flattened to convert to features that are input to the MLP. In fact, it is this spacially significant nature of image data that motivates the use of CNNs in such scenarios, as detailed in the next section. Despite this limitation, our MLP displays significant performance on many classes.

Overall Performance Metrics

	Metric	Value
0	Average Loss	1.4563

	Metric	Value
1	Accuracy	59.59%
2	Macro-F1 Score	0.56



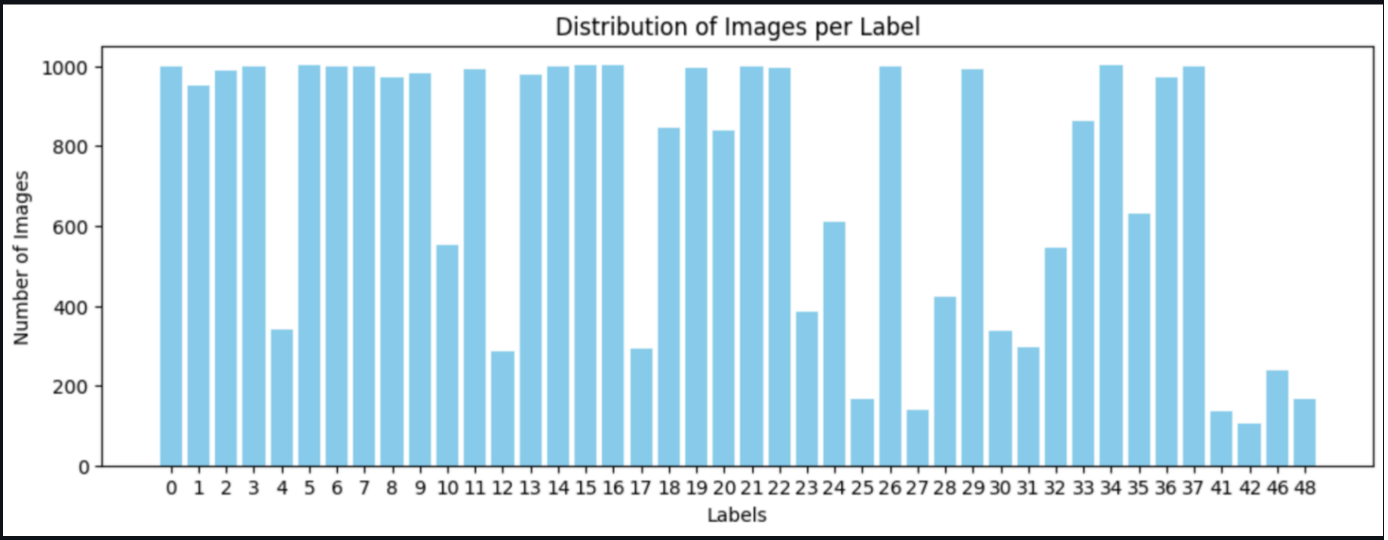


Per Class Performance

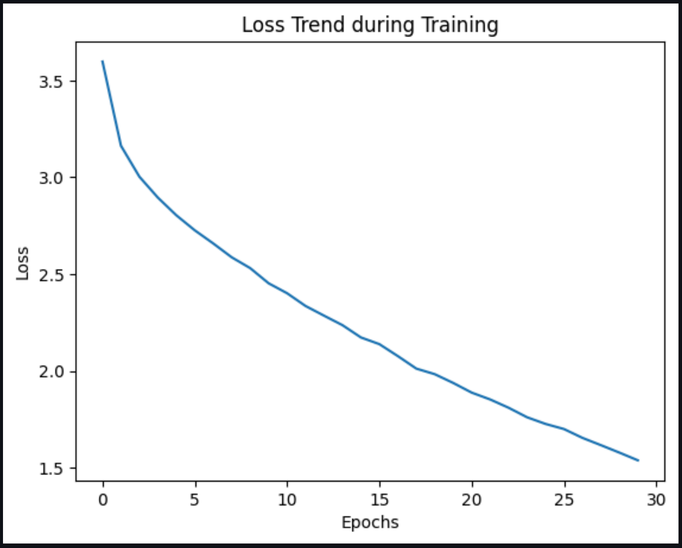
	Class Name	Precision	Recall	F1 Score
0	gazelleGrants	0.7000	0.4900	0.5800
1	reedbuck	0.6200	0.6500	0.6300
2	dikDik	0.3800	0.6400	0.4800
3	zebra	0.7500	0.6900	0.7200
4	porcupine	0.7200	0.4100	0.5200
5	gazelleThomsons	0.6700	0.9000	0.7700
6	hyenaSpotted	0.5000	0.6900	0.5800
7	warthog	0.3500	0.4100	0.3800
8	impala	0.7400	0.6400	0.6900
9	elephant	0.7400	0.5300	0.6200
10	aardvark	0.4600	0.6600	0.5400
11	giraffe	0.6800	0.4400	0.5300
12	mongoose	0.4500	0.0700	0.1200
13	buffalo	0.7000	0.7700	0.7300
14	hartebeest	0.9000	0.4000	0.5600

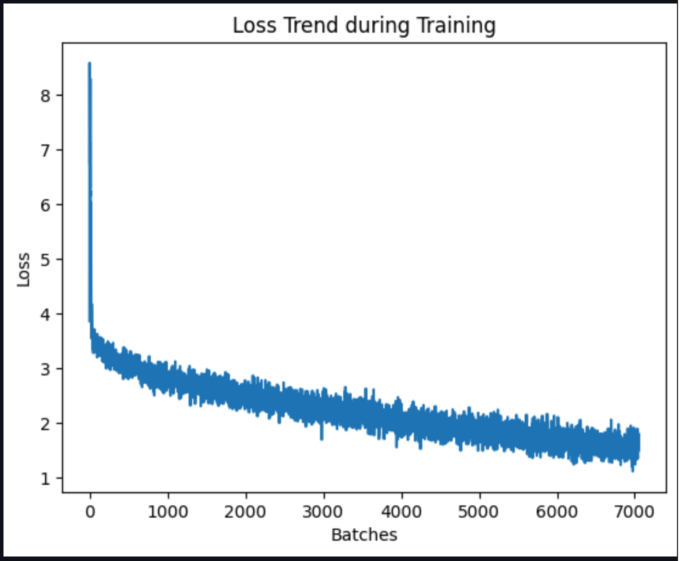
	Class Name	Precision	Recall	F1 Score
15	guineaFowl	0.3800	0.9200	0.5400
16	wildebeest	0.9300	0.8600	0.9000
17	leopard	0.7200	0.3700	0.4900
18	ostrich	0.4000	0.6400	0.4900
19	lionFemale	0.7300	0.6000	0.6600
20	koriBustard	0.5300	0.4300	0.4800
21	otherBird	0.6200	0.5100	0.5600
22	cheetah	0.6600	0.4400	0.5300
23	bushbuck	0.7000	0.6900	0.6900
24	jackal	0.5400	0.3200	0.4000
25	aardwolf	0.4000	0.1500	0.2200
26	hippopotamus	0.6300	0.8200	0.7100
27	hyenaStriped	1.0000	0.0400	0.0800
28	hare	0.4100	0.3600	0.3800
29	baboon	0.6100	0.7500	0.6700
30	vervetMonkey	0.7000	0.3900	0.5000
31	batEaredFox	0.5000	0.1000	0.1700
32	waterbuck	0.7700	0.7700	0.7700
33	secretaryBird	0.8300	0.6700	0.7400
34	topi	0.8400	0.6400	0.7300
35	serval	0.6900	0.4200	0.5200
36	lionMale	0.5100	0.5700	0.5400
37	eland	0.5300	0.6400	0.5800
38	reptiles	0.9100	0.8700	0.8900
39	insectSpider	0.2800	0.5300	0.3600
40	gazelleGrants	0.7700	0.5400	0.6300
41	hyenaSpotted	0.7900	0.6200	0.6900

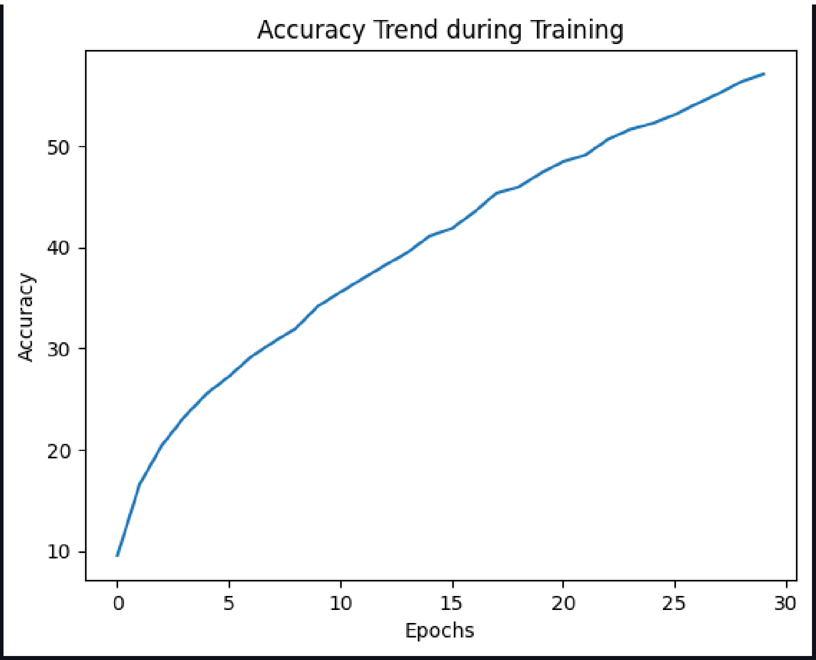
A notable point about the MLP model is that perfomance is on average worse for classes that we possess lesser sample images from, as can be seen by cross referencing the class frequencies shown below for convenience against classwise performace from the table above.



Moreover, the MLP model performance can also be quantified via the loss function progression as a function of epoch as well as number of batches. In both visualizations, we see a consistent trend of decreasing loss, indicating steady improvement in model progress.





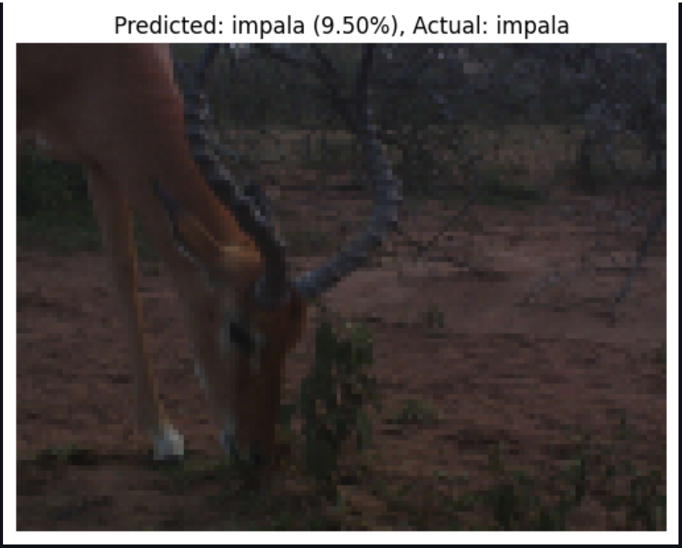


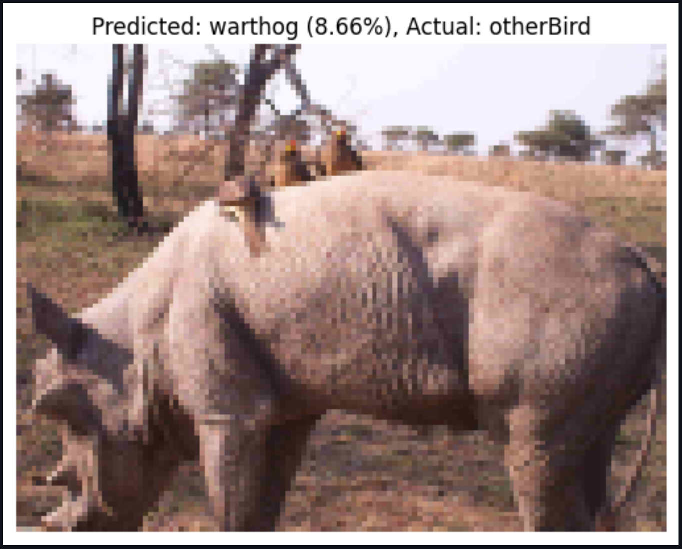
4. CNN Model

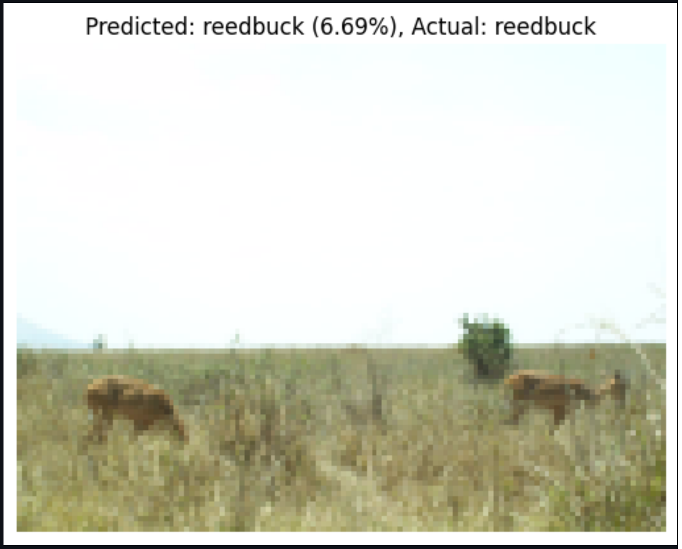
Our convolutional neural network model showed the best performance out of all three models we experimented with, giving the lowest average cross entropy loss, the highest accuracy and the best Macro F1 score. As predicted, the CNN has a much greater representational capacity when it comes to understanding image data that has a spacial component. However, this greater capacity to capture spacial features comes at the cost of computational expense and a longer training time than any of the other two models. Overall, the CNN takes the longest time to train because of the expensive process of computing convolutions, making convolutions both a strength as well as a drawback when it comes to our model.

Overall Performance Metrics

	Metric	Value
0	Average Loss	0.2348
1	Accuracy	92.97%
2	Macro-F1 Score	0.92







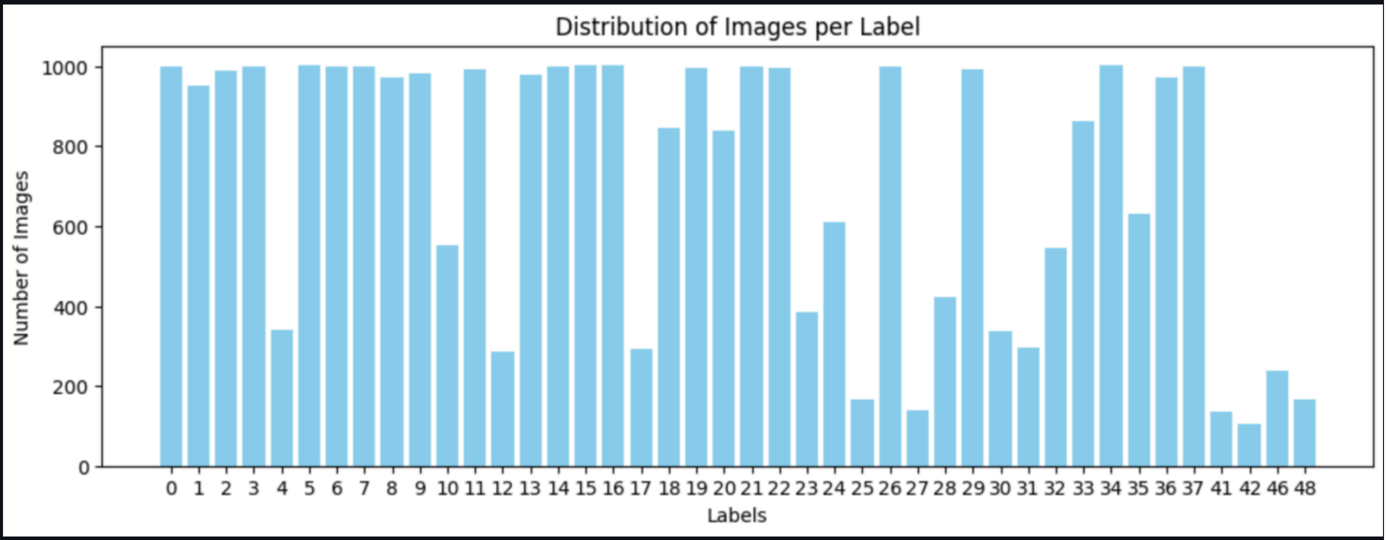
Per Class Performance

	Class Name	Precision	Recall	F1 Score
0	gazelleGrants	0.9200	0.9000	0.9100
1	reedbuck	0.8000	0.9400	0.8700
2	dikDik	0.7600	0.9200	0.8400
3	zebra	0.8900	0.9700	0.9300
4	porcupine	0.9900	0.8100	0.8900
5	gazelleThomsons	0.9200	0.9400	0.9300
6	hyenaSpotted	0.9600	0.9100	0.9300
7	warthog	0.9500	0.8700	0.9100
8	impala	0.9400	0.9400	0.9400
9	elephant	0.9800	0.9700	0.9700

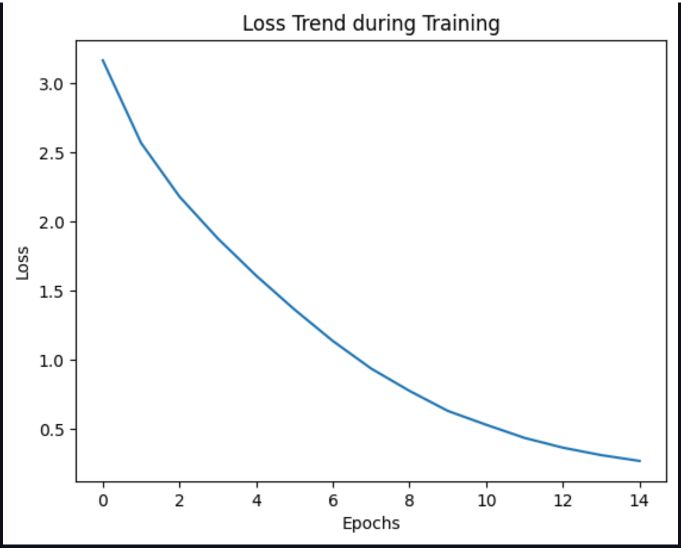
	Class Name	Precision	Recall	F1 Score
10	aardvark	0.9800	0.8400	0.9100
11	giraffe	0.9500	0.9300	0.9400
12	mongoose	0.8200	0.8200	0.8200
13	buffalo	0.9700	0.9700	0.9700
14	hartebeest	0.8900	0.9700	0.9300
15	guineaFowl	0.9400	0.9600	0.9500
16	wildebeest	1.0000	0.9400	0.9700
17	leopard	0.9400	0.9500	0.9400
18	ostrich	0.9800	0.9500	0.9700
19	lionFemale	0.9600	0.9100	0.9400
20	koriBustard	0.9500	0.9300	0.9400
21	otherBird	0.9000	0.8700	0.8900
22	cheetah	0.9700	0.9600	0.9700
23	bushbuck	0.9600	0.9300	0.9500
24	jackal	0.8500	0.9400	0.8900
25	aardwolf	0.9400	0.7200	0.8200
26	hippopotamus	0.9500	0.9700	0.9600
27	hyenaStriped	0.8200	0.6800	0.7400
28	hare	0.9500	0.7000	0.8100
29	baboon	0.9500	0.9700	0.9600
30	vervetMonkey	0.9400	0.9200	0.9300
31	batEaredFox	0.7300	0.7300	0.7300
32	waterbuck	0.9800	0.9900	0.9800
33	secretaryBird	0.9600	0.9900	0.9700
34	topi	0.9500	0.9500	0.9500
35	serval	0.9900	0.8400	0.9100
36	lionMale	0.9300	0.9600	0.9400
37	eland	0.9600	0.9900	0.9800
38	reptiles	0.9200	0.9700	0.9500

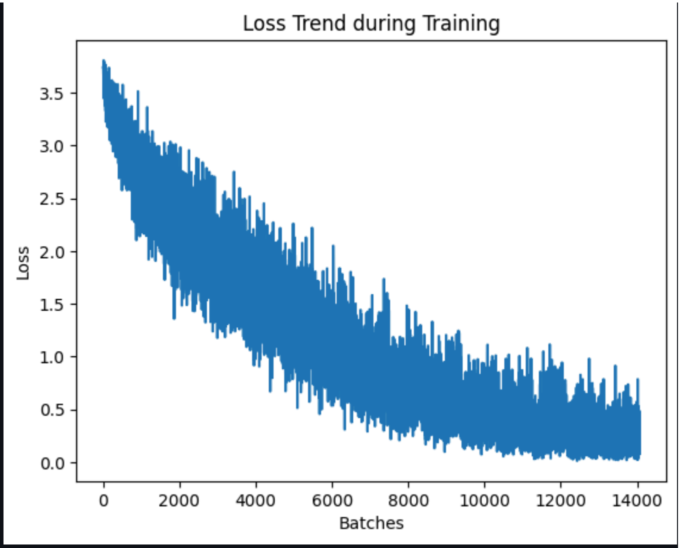
	Class Name	Precision	Recall	F1 Score
39	insectSpider	0.9900	0.9300	0.9600
40	gazelleGrants	1.0000	1.0000	1.0000
41	hyenaSpotted	0.9700	0.9900	0.9800

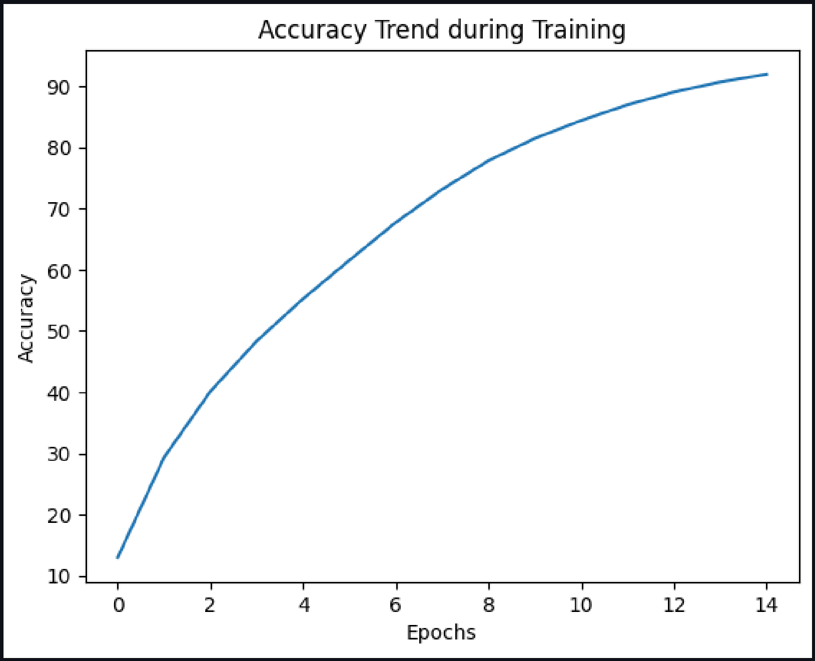
Once again, we see that perfomance is on average worse for classes that we possess lesser sample images from, as can be seen by cross referencing the class frequencies shown below for convenience against classwise performace from the table above.



Moreover, the CNN model performance can also be quantified via the loss function progression as a function of epoch as well as number of batches. In both visualizations, we see a consistent trend of decreasing loss, indicating steady improvement in model progress, along with a convergence to a lower final loss value than the MLP, indicating better learning.







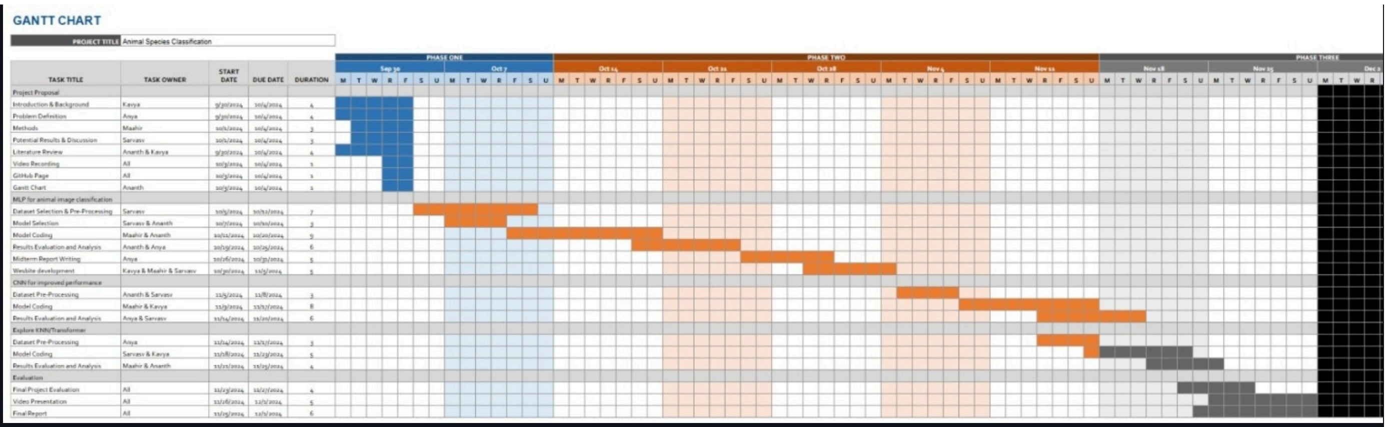
4. Model Comparisions and Conclusion:

In summary, we were able to use three different machine learning models in order to attempt to tackle the task of animal image classification, namely KNN, MLP and CNN. KNN was the least computationally expensive model, which enabled us to effectively tune hyperparameters and make adjustment without great computational cost, this flexibility came at the cost of performance, as evidenced by our metrics. All in all, it appears that the model was not sophisticated enough to handle image data, especially as compared to the other models, owing to the curse of dimensionality. The MLP in comparision, was a much more sophisticated learning approach that was able to achieve significantly better performance than the KNN approach, albeit at the cost of computation time. However, even a neural network based model such as MLP proved to not be capable of handling the spacial context associated with image data. It was this that made us turn to CNN, which enabled us to leverage the powerful representational capacity of convolutions to achieve the best performance overall. However, this performance came at the cost of significant computational resources owing to the expensive nature of convolutions. Thus, the three models used lie on a spectrum that goes from low to high performance while at the same time going from low to high computational cost, and therefore from high to low flexibility. Ultimately, there is no one right model for the task of animal image classification and the model we pick depends on our tolerance for errors and on the computational resources we have at hand. Nonetheless, all models represent an avenue to potentially automate the time-intensive task of image classification in large scale conservation projects and promise to help conservation efforts in areas such as the Serengenti and as such, we would consider this project to be a success.

5. Next steps:

Transformer-based architectures have been making great strides in the field of machine learning and have shown to have enormous representational power when it comes to tackling complex learning problems. Thus, our team believes that the natural next step toward improving our model in order to attain even better performance on the animal classification task would be to try and leverage the state of the art machine representation power offered by such architectures and attempt to utilize them as a foundational model for representation (essentially encoding the important features of the images) and feeding that representation through a classification framework such as an MLP while finetuning it in order to create a model that perform even better. We believe this is an exciting area of reseach and could contribute significantly to future wildlife conservation efforts in ecological hotspots like the Serengeti, offering a unique intersetion between machine learning and conservation.

Gantt Chart


GANTT CHART

PROJECT TITLE Animal Species Classification

TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION
Project Proposal				
Introduction & Background	Kavya	9/30/2024	10/4/2024	4
Problem Definition	Anya	9/30/2024	10/4/2024	4
Methods	Maahir	10/1/2024	10/4/2024	3
Potential Results & Discussion	Sarvasv	10/1/2024	10/4/2024	3
Literature Review	Ananth & Kavya	9/30/2024	10/4/2024	4
Video Recording	All	10/3/2024	10/4/2024	1
GitHub Page	All	10/3/2024	10/4/2024	1
Gantt Chart	Ananth	10/3/2024	10/4/2024	1
MLP for animal image classification				
Dataset Selection & Pre-Processing	Sarvasv	10/5/2024	10/12/2024	7
Model Selection	Sarvasv & Ananth	10/7/2024	10/10/2024	3
Model Coding	Maahir & Ananth	10/11/2024	10/20/2024	9
Results Evaluation and Analysis	Ananth & Anya	10/19/2024	10/25/2024	6
Midterm Report Writing	Anya	10/26/2024	10/31/2024	5
Website development	Kavya & Maahir & Sarvasv	10/30/2024	11/5/2024	5
CNN for improved performance				
Dataset Pre-Processing	Ananth & Sarvasv	11/5/2024	11/8/2024	3
Model Coding	Maahir & Kavya	11/9/2024	11/17/2024	8
Results Evaluation and Analysis	Anya & Sarvasv	11/14/2024	11/20/2024	6
Explore KNN/Transformer				
Dataset Pre-Processing	Anya	11/14/2024	11/17/2024	3
Model Coding	Sarvasv & Kavya	11/18/2024	11/23/2024	5
Results Evaluation and Analysis	Maahir & Ananth	11/21/2024	11/25/2024	4
Evaluation				
Final Project Evaluation	All	11/23/2024	11/27/2024	4
Video Presentation	All	11/26/2024	12/1/2024	5
Final Report	All	11/25/2024	12/1/2024	6

Contributions

Anantharaman Iyer	MLP Implementation
Anya Kathpalia	Model Visualization
Kavya Venkatesh	KNN Implementation
Sarvasv Barara	Data Preprocessing
Maahir Jain	CNN Implementation

Pivots and Challenges

We chose a different direction for our project during the start of the semester. Initially, we wanted to implement a model which would convert ASL into speech. However, throughout the first week of working we faced various hurdles in the implementation, a major one being that there were no well defined ASL image dataset which could be used for the model. In addition to that, we realized that the video data that we were planning to process would be too computationally expensive to process. Even though we were very passionate about our idea and its impact, we realized that it inhibited us from exploring a lot of areas in ML. This led us to pivot to our idea for animal image classification for which we had a better defined dataset. We think it was a good trade off since we value having a well defined dataset which would allow us to explore ML algorithms more effectively and apply what we had learnt in this class to solve another very real problem of animal species detection for the purpose of wildlife conservation.

References

[1] https://lila.science/datasets/snapshot-serengeti

[2] https://arxiv.org/abs/1312.4400

[3] https://arxiv.org/abs/1910.09716

[4] https://www.scirp.org/journal/paperinformation?paperid=62800

[5] https://www.sciencedirect.com/science/article/pii/S0003347216303360

[6] https://ieeexplore.ieee.org/document/9065747