North American Animals Classification

Scientific Report for DSPRO1 Course (HS24)

Coaches

Umberto Michelucci
Seraina Glaus
Derungs Curdin

Authors

Michelle Suter
Nadine Flück
Sevan Sherbetjian

Table of Contents

External Resources

This project has been completed with a variety of tools. Here is an overview of all services used:

GitHub Repository: https://github.com/michsute/North-American-Animals-Classification

Project Management: https://github.com/users/michsute/projects/1

Dashboard: https://north-american-animals-classification.streamlit.app/

We will happily provide access to these resources on request.

Problem and Vision

Problem Statement

Human activities such as agriculture and urban expansion, combined with climate change, are increasing habitat loss and disrupting ecosystems, putting many species at greater risk of extinction (Hanski, 2011). This emphasizes the importance of wildlife monitoring for nature conservation. Wildlife monitoring informs about the state and evolution of different species and their interaction and supports decision-making. However, collecting data is tedious.

Many animals avoid humans, are mostly active at night or live in remote, inaccessible areas, making them difficult to observe. For this reason, researchers rely on camera traps to collect photos and videos. These digital cameras contain an infrared sensor, which takes pictures every time a warm-blooded animal is detected (Wearn & Glover-Kapfer, 2017). The next step, which consists of manually analysing thousands of images is time-consuming, error-prone and resource-intensive

Vision

Our goal is to develop an automated system that uses machine learning to quickly and accurately identify animals in camera trap images. This system will process large amounts of data much faster than manual methods, reducing errors and saving time for researchers. By providing tools such as species trend analysis, activity visualization and alerts for unusual sightings, the system will help track endangered species, manage animal populations and avoid human-wildlife conflicts. The final product will be a user-friendly dashboard that supports conservation efforts by providing reliable data for informed decision-making.

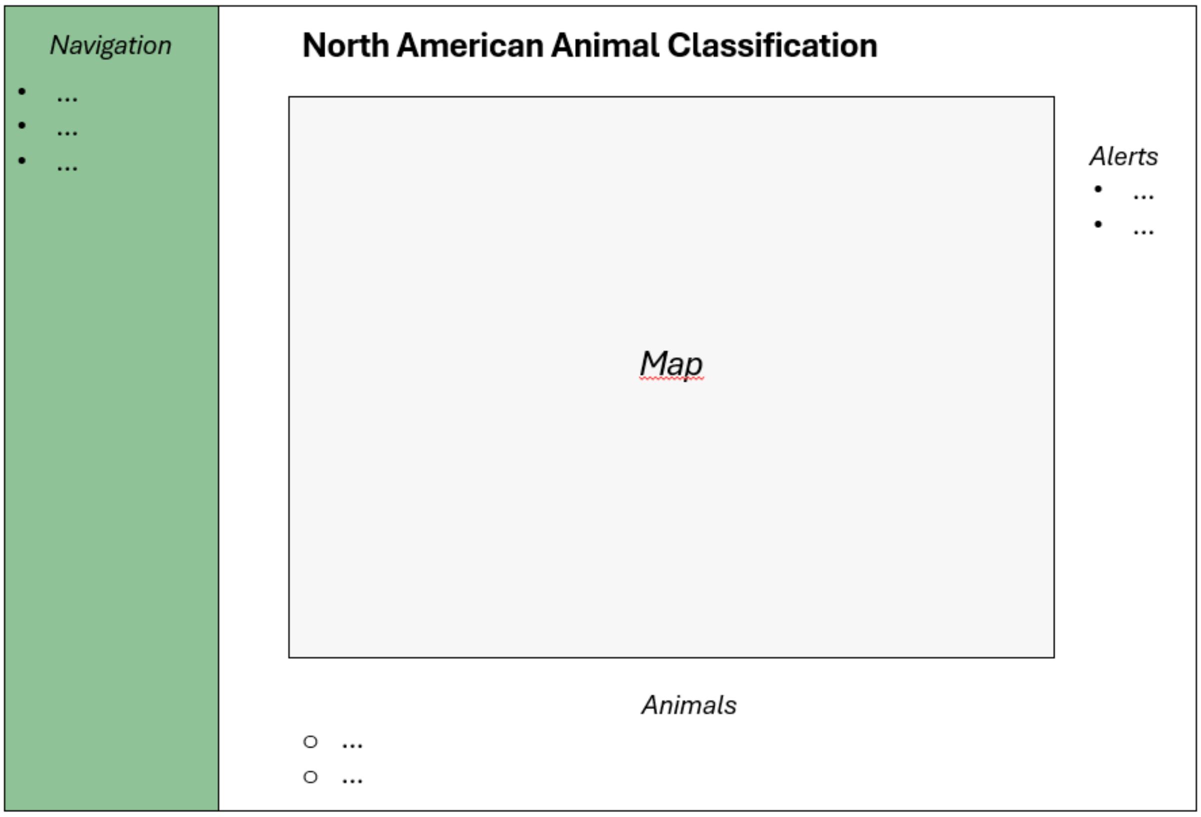


Figure 1: Dashboard vision

Business Motivation

Our project aims to create a dashboard that helps researchers and conservationists by automating the identification of animals in camera trap images. This saves time, reduces effort, and enables users to focus on data analysis and decision-making. The system's tools improve understanding of animal behaviour, population trends, and habitat use. By providing accurate insights quickly, the project supports better conservation strategies and helps protect endangered species. The dashboard makes wildlife research and conservation more efficient and impactful.

Target Audience

Wildlife Researchers and Biologists

Wildlife researchers study animal behaviour, movement, and habitats, often collecting large amounts of data through camera traps.

Our system simplifies their work by automating image analysis, saving time and enabling them to focus on analysing results, publishing findings, and planning conservation efforts. With more accurate datasets, researchers can better monitor long-term population changes and adapt strategies accordingly.

Conservation Organizations

Conservation groups protect animals and their habitats, working to prevent extinction and promote ecosystem health.

Our system helps them process large datasets efficiently, identifying species and tracking movements with minimal manual effort. These insights allow for quicker responses to habitat threats, reduced human-wildlife conflicts, and more effective awareness campaigns. The ability to demonstrate measurable results also helps organizations secure funding and support for their work.

Literature review

Background of Wildlife Population Monitoring

Monitoring wildlife populations involves collecting data over time on key environmental factors such as species diversity and population size. This process helps to assess the state of ecosystems and monitor changes in wildlife populations (Yoccoz, Nichols, & Boulinier, 2001).

Camera traps are very widely used to monitor the presence of animals and record their behaviour (Apps & McNutt, 2018). They offer different advantages such as non-invasive observation of species and monitor them without the need for human presence. This is especially useful for studying night-active, fearful and protected species, providing a more accurate understanding of wildlife presence and interactions, especially across large areas and difficult-to-access habitats. Advances in motion detectors and infrared technology have enabled more precise monitoring of animal behaviour (Apps & McNutt, 2018).

Background on Image Classification

Image classification involves categorizing and labelling groups of pixels or vectors within an image based on specific rules. Traditional methods relied on manual feature extraction. Earlier advancements in machine learning and deep learning have revolutionized the field (Krizhevsky, Sutskever, & Hinton, 2012). Convolutional neural networks (CNNs) are the most commonly used deep learning models for image classification. They are characterized by their ability to automatically learn spatial hierarchies of features in tasks such as object recognition and scene understanding (Simonyan & Zisserman, 2015).

Modern architectures like Vision Transformers (ViTs) and hybrid models combine CNNs with attention mechanisms to further improve accuracy, particularly for complex datasets (Dosovitskiy, et al., 2021). Transfer learning is also widely applied, where pre-trained models on large datasets like ImageNet are fine-tuned for specific tasks, reducing the need for extensive labelled data (Deng, et al., 2009). These advances have made image classification a fundamental element of computer vision applications in various fields, including wildlife monitoring.

Background on Animal Classification

Building on image classification techniques, animal classification focuses on identifying species from images, often captured through camera traps. Challenges in this domain include occlusion, intra-species variation, and imbalanced datasets. Techniques like transfer learning and fine-tuning pre-trained models are commonly employed to address these challenges (Russakovsky, et al., 2015).

Datasets specific to wildlife, such as camera trap images, require preprocessing steps like bounding box annotation and data augmentation to improve model performance (Beery, van Horn, & Perona, Recognition in Terra Incognita, 2018). Tools such as CNNs, EfficientNet, and ViTs have shown promise in accurately classifying animals, even in low-light or cluttered environments (Tan & Le, 2020).

State of the Art

Advances in deep learning and computer vision have significantly improved wildlife monitoring, especially in the automatic identification of animals in camera trap images. Many solutions have been developed to overcome challenges such as limited data, changing environments and uneven representation of species in datasets. Some of the most important methods and tools in this area are presented below.

MEWC: A user-friendly AI workflow for customised wildlife-image classification

MEWC (Mega-Efficient Wildlife Classifier) is a tool designed for the automatic classification of species in camera trap images. It utilizes the MegaDetector model, a pre-trained object recognition system capable of identifying animals, humans, and vehicles. This model helps by filtering out empty images, significantly reducing the time required to process large datasets (Brook, Buettel, & Aandahl, 2023).

In addition, MEWC also includes animal species recognition models specific to different regions, such as Sub-Saharan Africa and Europe, enabling accurate identification of animals across diverse environments. As an open-source and user-friendly system, MEWC provides significant value to researchers working with large-scale ecological data (Brook, Buettel, & Aandahl, 2023).

iWildCam Dataset and Challenges

The iWildCam 2020 challenge, part of the FGVCx competition, presents a significant dataset for wildlife species classification from camera trap images. The dataset, created by Beery, Cole, & Gjoka (2020), consists of millions of images captured by camera traps across diverse global locations. They offer a valuable resource for automated wildlife monitoring. The dataset includes three primary components: camera trap images, citizen science images from the iNaturalist platform, and multispectral remote sensing data for each camera location.

The dataset tests models on their ability to generalize across different geographic locations. This ensures that models are trained and evaluated on data from unseen environments. A baseline Inception-v3 model achieved an F1 score of 0.62 and an accuracy of 62%. This highlights the need for more robust techniques, such as transfer learning and advanced deep learning architectures, to handle imbalanced data and complex environments (Beery, Cole, & Gjoka, The iWildCam 2020 Competition Dataset, 2020).

MegaDetector: A Baseline for Wildlife Detection

The MegaDetector, developed by Microsoft AI for Earth, is a pre-trained object detection model designed to automate the monitoring of wildlife in camera trap images. The model has been optimized for the detection of animals, humans and vehicles and is able to filter out empty images. This capability makes it particularly useful for large-scale ecological research.

The MegaDetector has been optimized for wildlife monitoring and is effective in a variety of environments, from forests to deserts. While it performs well in detecting animals in clear images, there are still challenges in accurately identifying smaller or rare species and handling cluttered backgrounds.

The model is often used as a first step in wildlife monitoring pipelines, where it helps to detect the presence of animals before further species-specific classification takes place. Its scalability and efficiency make it a valuable tool for large-scale wildlife research. (Microsoft, 2024) (Microsoft, 2021)

Wildlife Insights: A Scalable Monitoring Platform

Wildlife Insights is an open-source platform designed to support large-scale wildlife monitoring projects. It helps researchers gather, analyse, and share data from camera traps. This platform offers tools for managing large datasets and automating aspects of wildlife detection.

Wildlife Insights allows researchers to combine data from camera traps, citizen science, and remote sensing to better understand wildlife populations. It also emphasizes collaboration by enabling global data sharing, which accelerates conservation efforts and advances the understanding of wildlife behaviour and ecology. (Wildlife Insights, n.d.)

CNN-Based Classification Models for Low-Light and Occluded Images

Recent advances in CNNs have significantly improved the classification of images in challenging conditions, such as low-light environments and occluded objects. CNNs are widely used due to their ability to learn complex hierarchical features but have difficulties when dealing with poorly lit images or partially occluded objects. To address these challenges, several innovative approaches have been developed to improve the performance of CNN models under such conditions.

The paper by Jincheng Li et al. (2024) introduces an innovative deep learning-based model that aims to improve the recognition and classification performance of CNNs in low-light conditions. The model uses an adaptive framework that focuses on noise reduction and contrast enhancement, significantly improving the CNN's ability to extract relevant features from low-light images (Li, Qiao, Xu, & Huang, 2024). This method is especially useful in wildlife monitoring, where many images are captured in suboptimal lighting conditions.

In the other hand, Jiahong Zhang et al. (2023) present a novel deep learning system that improves classification accuracy despite occlusions. By integrating a multi-task learning approach, the model can recognize animals even when parts of them are obscured by environmental elements such as trees or foliage (Zhang, Cao, Lai, Li, & Qin, 2023). This makes it a valuable tool for wildlife monitoring, where occlusions often occur due to the natural environment.

Additionally, Zixiang Wei et al. (2023) proposes a hybrid approach combining the strengths of both CNNs and transformers. The model leverages CNNs for local feature extraction and transformers for global contextual understanding, resulting in improved classification accuracy for low-light and occluded images (Wei, Wang, Sun, Vasilakos, & Wang, 2023).

Data

Data collection

Data source

For our project, we used the North American Camera Trap Images dataset. We found this dataset on Labeled Information Library of Alexandria: Biology and Conservation (LILA BC). This dataset includes 3.7 million camera trap images taken from five locations across the United States. These images capture a variety of animals and are labelled into 28 animal species. About 12% of the images are labelled as empty (no animals present). The dataset is provided in COCO Camera Traps .json format, which makes it easy to use for image analysis and machine learning tasks.

Data analysis

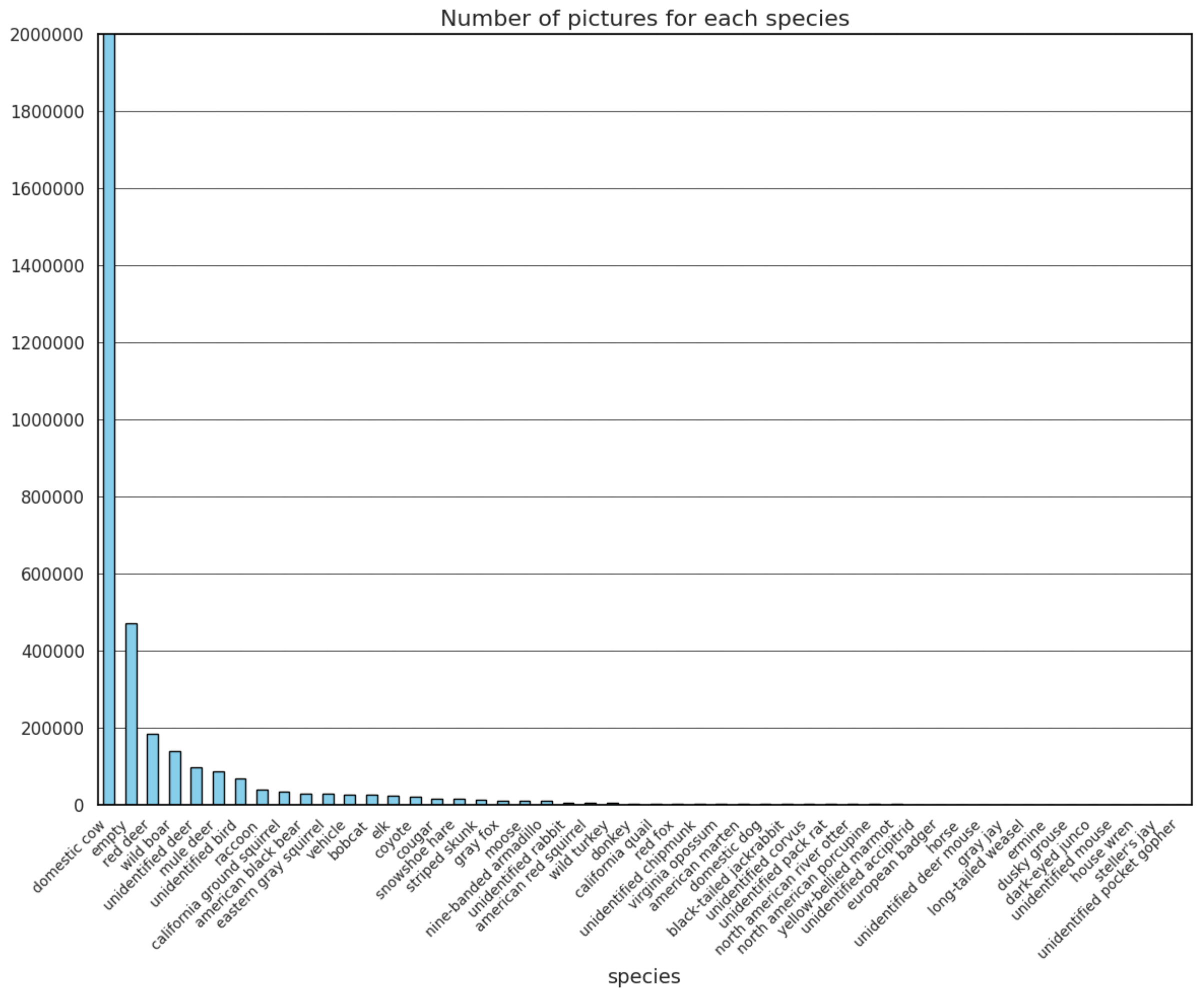


Figure 2: Original dataset with 28 labels

As seen in the visualization of the original dataset, the data appears to be very large and highly unbalanced. Some categories, such as “Domestic cow”, contain an extremely large number of images (2’019’009), while others have significantly fewer (17 categories contain less than 1’000 images). This unbalanced distribution of categories introduces a bias in machine learning models, as they can tend towards the more frequent labels. To address this issue, we decided to focus on 12 specific categories and select 3’000 images per category.

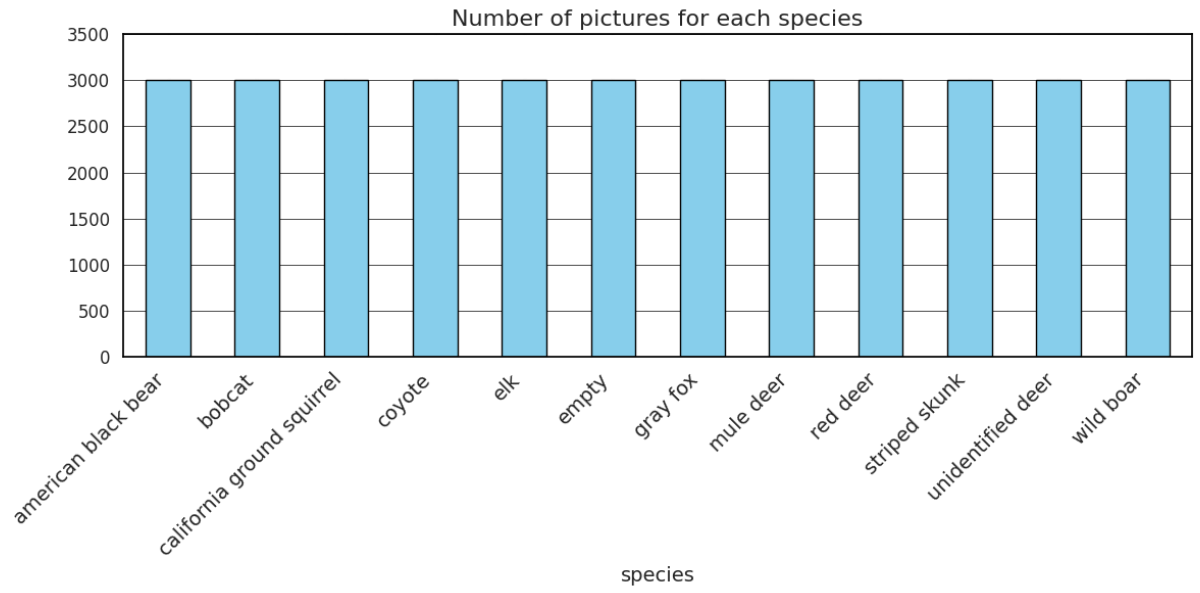


Figure 3: Dataset with 12 labels

This approach allowed us to create a balanced and more manageable dataset, suiting our image classification model. The balanced dataset not only improves fairness during training but also ensures that the model can perform consistently across all selected categories.

Class Diversity

The selected categories include a range of species’ types. This diversity allows the model to handle various ecological contexts.

Carnivores	bobcat, coyote, gray fox
Herbivores	elk, mule deer, unidentified deer
Omnivores	wild boar, striped skunk, American black bear
Small mammals	California ground squirrel
Other	empty

The inclusion of the “empty” class is important in our use case. Camera trap datasets often contain images triggered by environmental factors (e.g., wind, moving vegetation) rather than animals. Training the model on this class will help minimize false positives during deployment.

Sample images from the Dataset

To give a better overview of the animals and the types of images we worked with, here are a few images of each animal from our selected categories.

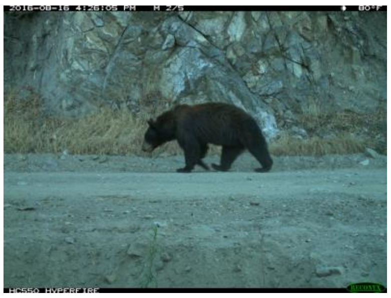


Figure 4: Image American Black Bear

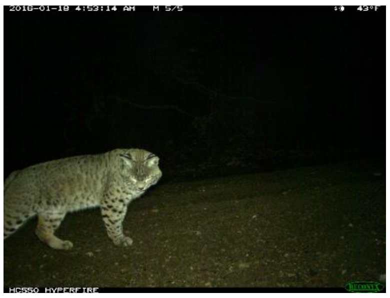


Figure 5: Image Bobcat

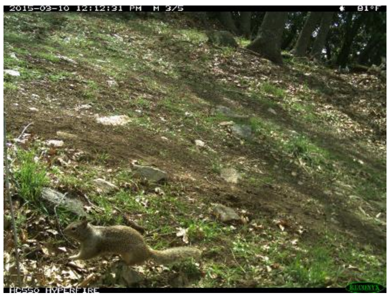


Figure 6: Image California Ground Squirrel

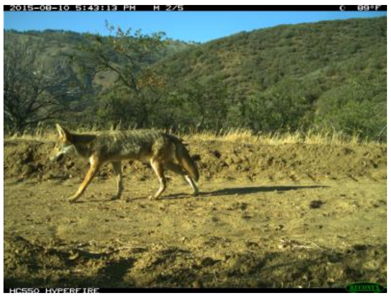


Figure 7: Image Coyote

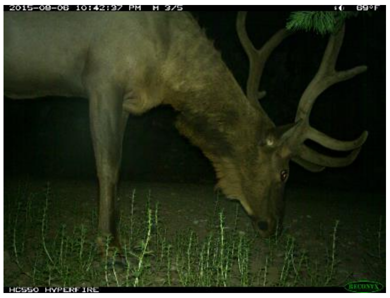


Figure 8: Image Elk

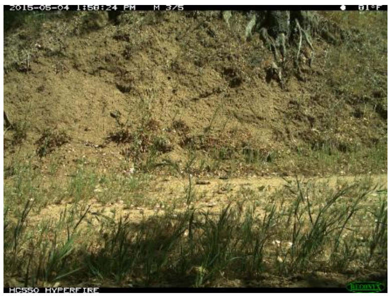


Figure 9: Image Empty

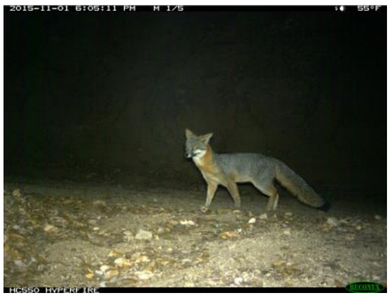


Figure 10: Image Gray Fox

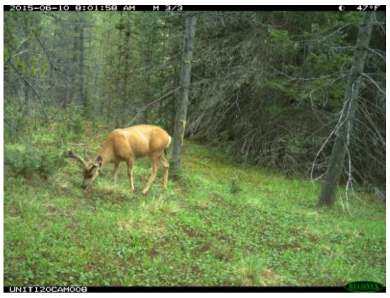


Figure 11: Image Mule Deer

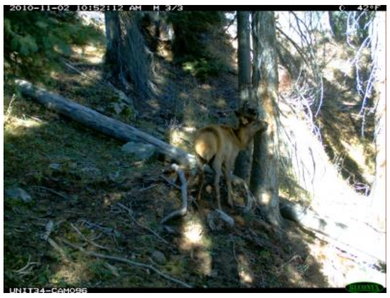


Figure 12: Image Red Deer

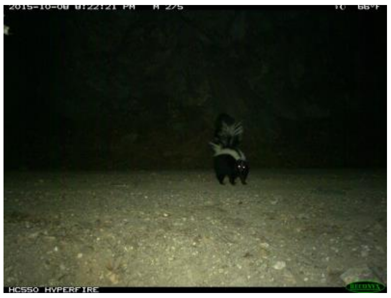


Figure 13: Image Striped Skunk

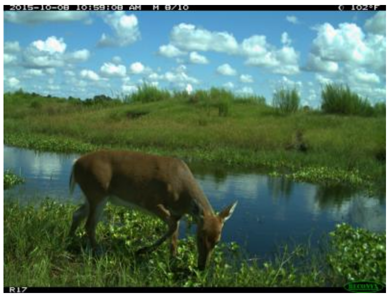


Figure 14: Image Unidentified Deer

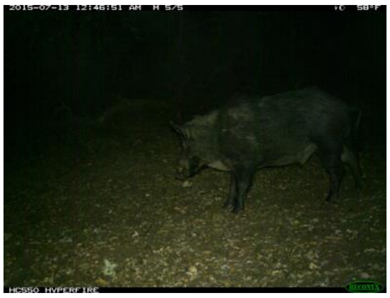


Figure 15: Image Wild Boar

Data quality

To test the quality of our selected classes, we manually checked about 40 images per category (about 480 in total). This confirmed that the animals in the images corresponded to the labelled class, and we observed that, in some cases, multiple animals of the same category appeared in a single image. This task helped us familiarize with the data and anticipate challenges (dark images, animal present in the background, etc.).

Data split

To divide our dataset, we used an 80/20 split, where 80% of the data was used for training the model, and the remaining 20% was set aside for testing the model. This means that we trained the model on 80% of the images, and after training, we tested it on the 20% that it hadn't seen before to check how well it performed.

To improve the training process, we split the training data further. Out of the 80% training data, we used 90% for training the model, and the remaining 10% was used as a validation set. The validation set allows us to assess the model's performance while it is still being trained. By testing the model on this smaller set of data during training, we could track its progress and adjust the model's settings to improve performance. This helps us avoid overfitting, ensuring that the model doesn't just memorize the training data but is capable of generalizing to new, unseen data.

	Pictures per class
Training	1920
Validation	480
Testing	600
Total	**3000**

Table 1: Datasplit per Categorie

Data preprocessing

For our project, we applied a series of preprocessing steps to prepare the images for model training. These steps were essential to ensure that the data fed into the model was standardized, while also introducing techniques to prevent overfitting and improve generalization.

Image Resizing

The first step in our preprocessing pipeline was resizing the images. All images were resized to a fixed size of 128x128 pixels. This is necessary because the models expect input images of consistent dimensions. By resizing the images to the same size, we ensure that the model can process them correctly without issues. Moreover, it accelerated training time.

Conversion to Tensors

Once the images were resized, we converted them into PyTorch tensors using *transforms.ToTensor()*. This step transforms the images from a format that is easy for humans to view (JPEG or PNG) into a tensor format that can be processed by neural networks. Tensors are multi-dimensional arrays and are the fundamental data structure used by PyTorch for model training.

Normalization

Next, we applied normalization to scale the pixel values of the images. This step adjusts the colour values so that they are within a range that the model can process more efficiently. Since our dataset had no unusual images (like medical scans), we used the mean and standard deviation values from ImageNet, on which Resnet was pre-trained. The values we used were:

Mean:	[0.485, 0.456, 0.406] (for Red, Green and Blue)
Standard Deviation:	[0.229, 0.224, 0.225]

Normalization helps the model converge faster during training. Without normalization, the model would struggle to learn effectively due to the differences in the scales of pixel values.

Data Augmentation

To prevent overfitting and help the model generalize better we applied data augmentation to the training images. Data augmentation techniques introduce variations in the images by applying random transformations, which helps the model learn more robust features. The augmentations we explored include:

Random Horizontal Flip:	This flips the image horizontally with a 50% chance. This technique helps the model learn to recognize objects regardless of their orientation.
Random Rotation:	We randomly rotated the images by up to 15 degrees. This introduces variability in how objects may appear in different orientations.
Colour Jitter:	We applied slight variations to the image’s brightness, contrast, saturation, and hue. This helps the model become more robust to different lighting conditions in real-world scenarios.

Denormalization (for visualization)

While training, we used normalized data, but for visualizing the images during experimentation, we applied denormalization. Denormalization reverses the normalization process and restores the image’s original colour scale. This makes it easier to inspect the original images versus their transformed version, as they would appear in their natural state. We did this by applying the inverse of the mean and standard deviation values which we used during normalization.

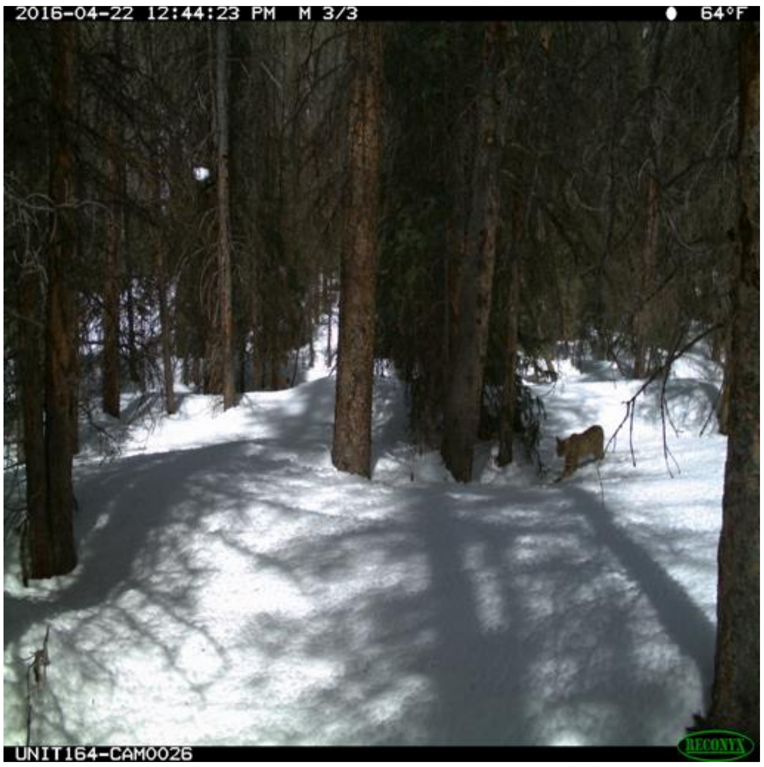


Figure 16: Example of an original image of the bobcat class

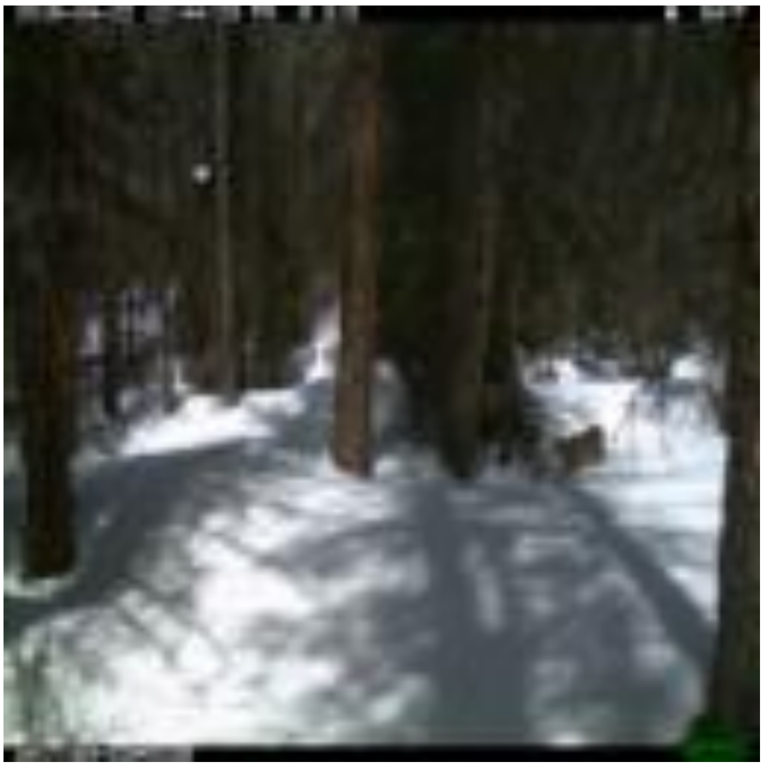


Figure 17: same picture after resizing

Figure 18: same picture after data augmentation

Methods

Framework

We used the CRISP-DM (Cross Industry Standard Process for Data Mining) framework to guide our project. This framework is well-suited for our project, as it provides clear guidelines through every stage of the process, from understanding the business problem to evaluating model performance.

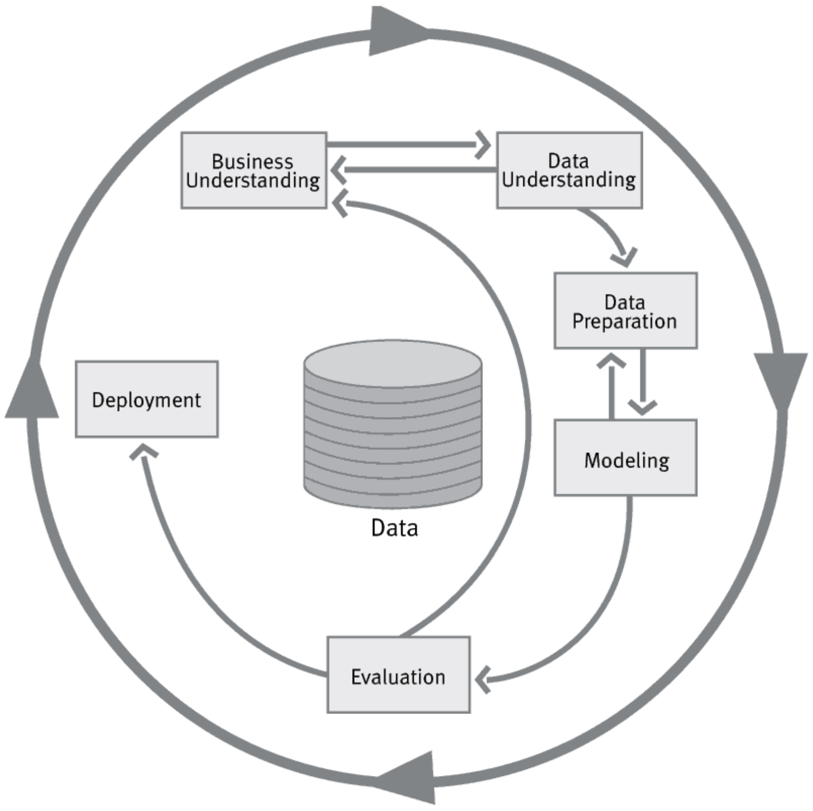


Figure 19: CRISP-DM framework

Phases of CRISP-DM

1. Business Understanding

   We begin by identifying the key objectives for wildlife conservation. The goal of the project is to classify images of North American wildlife to assist researchers and conservationists in monitoring animal populations and their habitats efficiently. Understanding the importance of accurate species identification helps us align the project outcomes with wildlife conservation goals.

2. Data Understanding

   In this phase, we analyse the North American Camera Trap Images dataset from LILA BC. We examine the structure of the data, evaluate its quality, and assess potential challenges such as class imbalance, mislabelling, or unclear images.

3. Data Preparation

   The data preparation phase involves cleaning and preprocessing the images. This includes resizing images to ensure consistency, augmenting data to address imbalances, and splitting the dataset into training, validation, and test sets. Ensuring a high-quality, well-organized dataset is critical for building effective models.

4. Modelling

   In this step, we build and fine-tune machine learning models to classify the animal species. We experiment with different architectures, such as ViTs, ResNet, and EfficientNet, to determine the most effective model for this classification task. After evaluating the performance of each model on the training set, we select one and focus on refining it for further training and optimization. This allows us to concentrate on improving a single model and ensure it is well-tuned for classification task.

5. Evaluation

   The selected model is thoroughly evaluated using metrics such as accuracy, precision, recall and F1 score to ensure it meets the project objectives.

6. Deployment

   Finally, we deploy the best-performing model. A dashboard is created to present the results, providing an intuitive interface for visualizing the model's predictions and performance metrics. This allows users to easily interact with the model's output and gain insights into its classification capabilities.

Team Management

Our team utilizes GitHub as the primary platform for collaboration and project management. To ensure seamless teamwork, we employ GitHub's version control system to manage our codebase. All team members contribute via branches and submit pull requests.

To organize tasks and track progress, we use the Kanban board on GitHub. The tasks are categorized into “To Do”, “In Progress” and “Done”. This provides a clear overview of the project’s status at any given time.

In addition to GitHub, we frequently communicate with each other over Discord. This platform allows us to discuss project updates, share insights, ask questions, and resolve issues in real time.

Selected Models

To achieve the best possible results for our wildlife classification project, we decided to train three different models, each chosen for its strengths in image classification tasks. Our selected models are well-established in the machine learning community, and each brings unique advantages to the table.

EfficientNet

EfficientNet is a family of CNNs. It is known for its efficiency in terms of both accuracy and computational resources. By optimizing the depth, width and resolution of the network, EfficientNet achieves high performance without requiring excessive computing power. It is designed to scale effectively. This makes it a good candidate for our first project in the field of machine learning- wildlife image classification. The model is particularly suitable for situations where computing resources or processing time are limited.

ResNet

ResNet (Residual Networks) is a deep learning architecture that introduces the concept of residual learning. The main innovation of ResNet is the use of skip connections, which allow gradients to flow more easily through very deep networks. This helps to mitigate the problem of vanishing gradients and allows the network to become much deeper without sacrificing performance. ResNet is known for its strong performance in image classification tasks.

Vision Transformer

ViT is a newer approach to image classification in which transformer models originally developed for natural language processing (NLP) are used for visual data. In contrast to CNNs, which use local receptive fields, ViT divides images into regions and treats them as sequences, similar to how words are treated in NLP. This allows the model to capture wide-ranging dependencies and global information, which can be particularly useful for complex patterns and relationships in images from the animal world. ViT has shown top performance in various benchmarks and is an interesting choice for our classification task.

Libraries

In this project, we utilized several libraries. These libraries help with various aspects of the project, including data preprocessing and training.

Libraries and Packages Used	Motivation	Usage in code
Torch	Core library for tensor operations and GPU acceleration.	Used for model training, tensor operations, and managing the device.
Torch.optim	For optimization algorithms such as Adam.	Used to define the Adam optimizer for updating the model's weights.
Torch.nn	For defining neural networks and loss functions.	Used for defining the cross-entropy loss function.
Torch.utils.data DataLoader	Efficient batch loading and parallel data processing.	Used to create data loaders for training and validation datasets.
Torch.utils.data random_split	To split the dataset into training and validation sets.	Used to split the dataset into training and validation subsets.

Table 2: Libraries Used Across Models

EfficientNet

Libraries and Packages Used	Motivation	Usage in code
Torchvision datasets	To handle image loading and transformation tasks.	Used for loading and transforming images, such as resizing and normalization.
Torchvision transforms	Image transformations (resize, normalization, tensor conversion).	Applied transformations to the images before feeding them into the model.

Torchvision models	To access pre-trained models such as EfficientNet.	Used to load the pre-trained EfficientNet-B0 model.
Torch.optim.lr_scheduler	For controlling the learning rate during training.	Used to implement a learning rate scheduler.
Tqdm	To show progress bars for loops, making the training process more user-friendly.	Used for providing progress bars during the training and evaluation loops.

Table 3: Libraries EfficientNet

ResNet

Libraries and Packages Used	Motivation	Usage in code
Torchvision datasets	To handle image loading and transformation tasks.	Used for loading and transforming images, such as resizing and normalization.
Torchvision transforms	Image transformations (resize, normalization, tensor conversion).	Applied transformations to the images before feeding them into the model.
Torchvision models	To access pre-trained models such as ResNet.	Used to load and modify the pre-trained Resnet50 model.
Torch.optim.lr_scheduler	For controlling the learning rate during training.	Used to implement a learning rate scheduler.
sklearn.metrics	Used to quantify the quality of predictions.	Used to calculate f1_score, confusion_matrix
Datetime	Retrieve current date and time	Used in the filename of saved data (csv and pth).
PIL	Manipulating images	Used to display images
Matplotlib	To visualize training/validation loss and accuracy during training.	It is used for plotting loss and accuracy graphs to monitor model performance.
Numpy	Used for numerical computations, particularly for handling array operations.	Numpy is used for array manipulations and computations, especially in the plotting of results.
Seaborn	Used for statistical data visualization.	Used to plot the confusion matrices.

Pandas	Used for data analysis. Provide user-friendly dataframes.	Used to read and write csv files.
Os	Used to navigate in the operating system.	Used to read the path and pictures to test the original Resnet model (without training).
Transformers AutoImageProcessor	This library is used to handle image preprocessing for Vision Transformer models.	It is used to initialize the image processor for transforming input images.
Transformers ResNetForImageClassification	To load the model.	Used to load the model ResNet50.
Torch.nn.functional	Contains functions such as pooling and activation functions.	Used with Softmax to retrieve probabilities of the original model.

Table 4: Libraries ResNet

Vision Transformer

Libraries and Packages Used	**Motivation**	**Usage in code**
Transformers AutoImageProcessor	This library is used to handle image preprocessing for Vision Transformer models.	Used to initialize the image processor for transforming input images.
Transformers ViTForImageClassification	To use the pre-trained ViT model for image classification tasks.	Used to load the pre-trained Vision Transformer model.
Matplotlib	To visualize training / validation loss and accuracy during training.	Used for plotting loss and accuracy graphs to monitor model performance.
Numpy	Used for numerical computations, particularly for handling array operations.	Numpy is used for array manipulations and computations, especially in the plotting of training / validation results.
Datasets load_dataset	Used to load image datasets in a structured format for training.	Used to load the image dataset from our directory.
Tqdm	To provide a progress bar for loops, making training and evaluation more user-friendly.	Used to display progress bars during training and validation phases.

Table 5: Libraries Vision Transformer

Transfer learning (ResNet)

Fully connected layer

To adapt the existing ResNet model to our use case, the final linear layer is replaced by a fully connected layer. The latter leverages the knowledge of the trained ResNet model and adapts it to a new task: classify 11 animal species and recognize empty images. Its output is thus 12 and no longer 1’000.

Additional layers

A fully connected feedforward neural network was added at the end of the existing ResNet model, instead of the fully connected layer. It contains two linear layers. These layers enhance the learning of complex patterns. The last layer is also linear and classifies the feature into the twelve classes. The Rectified Linear Unit (ReLU) activation function and a dropout function are applied after the first and second layers. The ReLU introduces Alleviates the vanishing gradients problem. It is a default activation function for many neural networks. The dropout sets some input units to zero to prevent overfitting and encourage generalization.

Gradient Descent

Cost function

The cost function (also called the "loss function") helps us measure how well our model’s predictions match the actual values (true labels). In classification tasks like ours, we want to minimize this difference to make the model more accurate. We used cross-entropy as the cost function because it works well for classification tasks, and it helps the model focus on minimizing the difference between predicted and true labels. A lower cross-entropy value means better performance.

Optimizer

During the training phase, the optimizer computes the gradient for each parameter. Adam optimizer (Adaptive Moment Estimation) was used in the model, as it is a corresponds to our use-case and is not very sensitive to the parameters’ scaling (Stevens, Antiga, & Viehmann, 2020). It combines the momentum, and the Root Mean Square Propagation (GeeksforGeeks, 2024).

The learning rate is a key parameter that controls how much the weights should be changed. If the learning rate is too high, the model might make big, unstable changes. If it's too low, training will be slow. We started with a learning rate of 0.001, which is a common starting point. Smaller learning rates help prevent the model from jumping around too much and ensure more stable progress.

We also used weight decay, which is a form of L2 regularization (or "Lasso regularization"). This adds a penalty to the loss function for having large weights. It helps prevent the model from fitting the noise in the training data, making the model more general and better at handling new, unseen data.

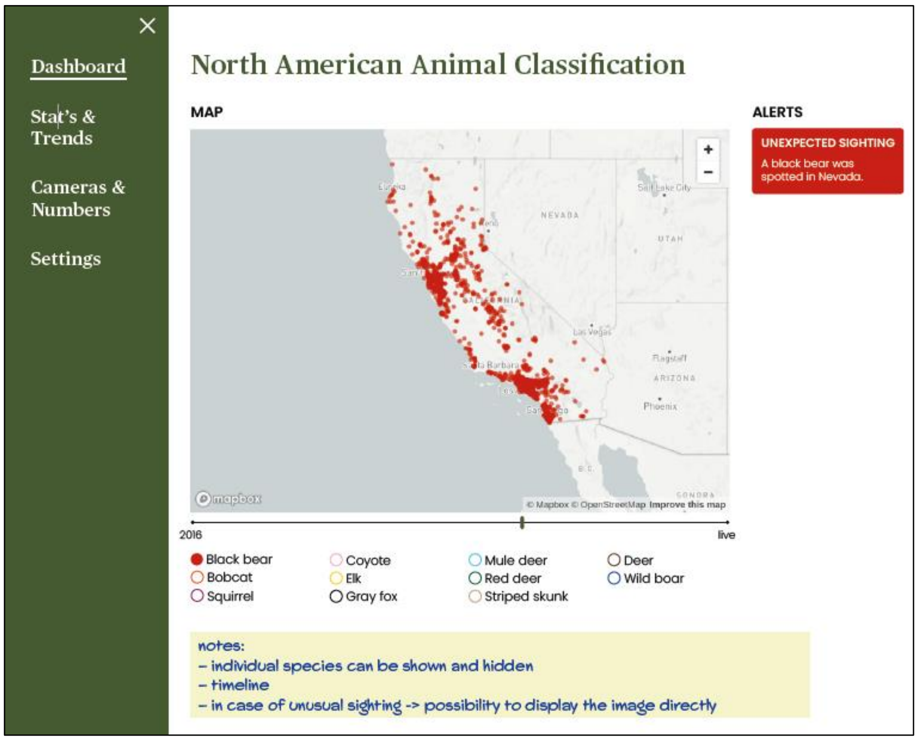
Scheduler

The scheduler dynamically adjusts the learning rate during the training phase. In our case, the scheduler reduces the learning rate by a factor of 0.1 (parameter “gamma”) every 5 epoch (parameter step size). It works together with the optimizer to improve training stability and speed.

Dashboard

Our final product is a dashboard designed to visualize and analyse wildlife data effectively. It displays a map as a core feature. This map shows the number of animals observed across different locations and allows users to filter data by species or date. An alert system is integrated to notify users of unexpected events such as an animal appearing in a specific location. Additionally, statistical data such as trends, co-occurrence of species, and seasonal movements are accessible.

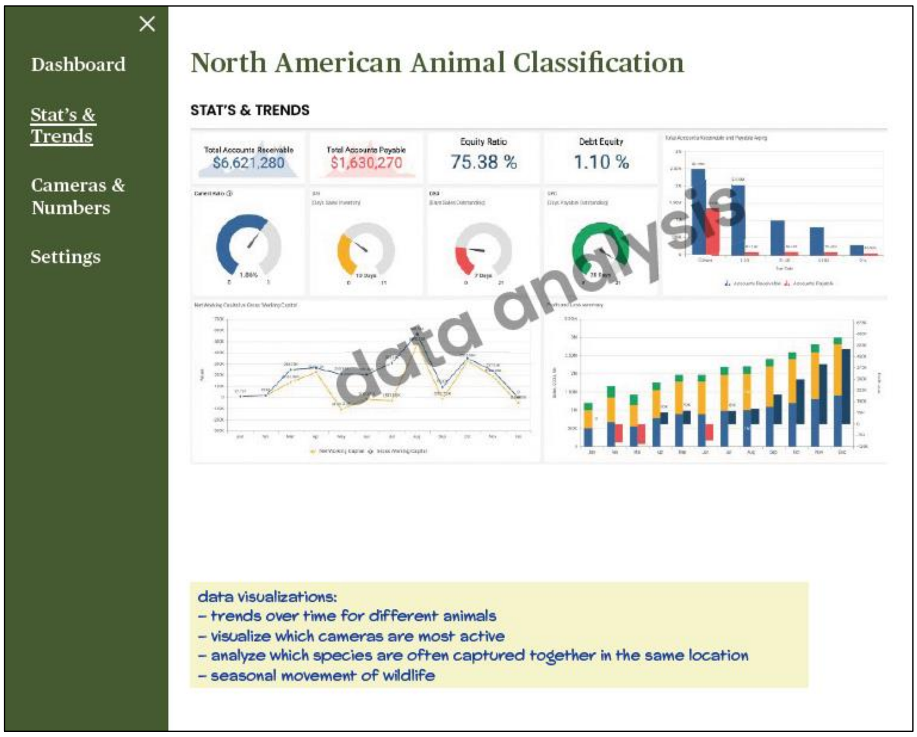
With this in mind, we first created a visualization of our dashboard. The finished result can be seen in the next chapter "Streamlit".



A black bear was spotted in Nevada.

notes:
- individual species can be shown and hidden
- timeline
- in case of unusual sighting -> possibility to display the image directly

Figure 20: Dashboard visualization



Dashboard

Stat's & Trends

Cameras & Numbers

Settings

North American Animal Classification

STAT'S & TRENDS

Total Accounts Receivable $6,621,280

Total Accounts Payable $1,630,270

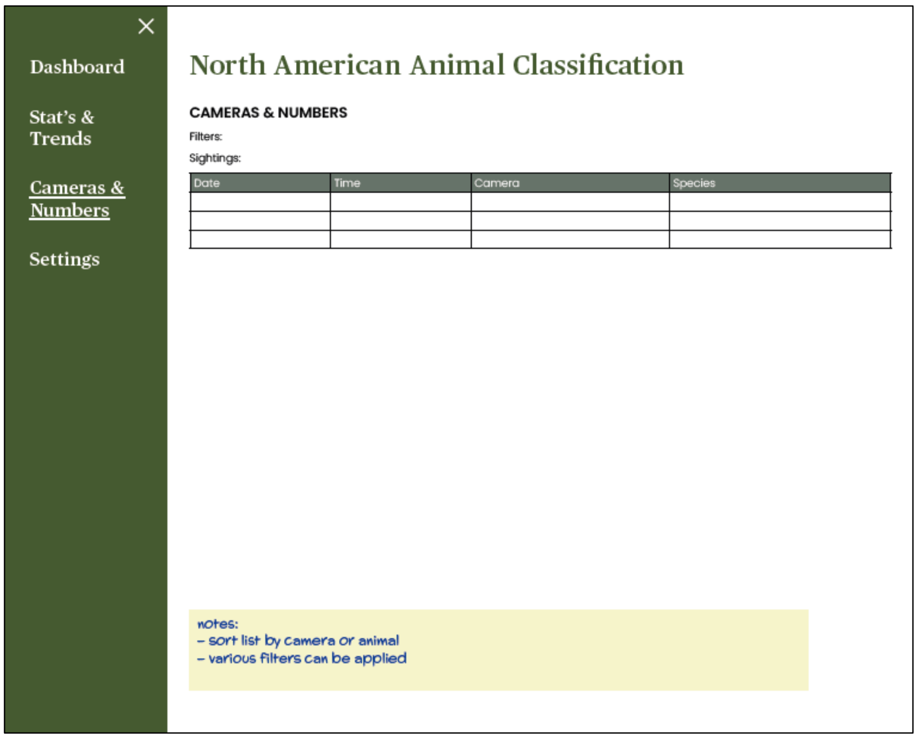
Equity Ratio 75.38 %

Debt Equity 1.10 %

data analysis

data visualizations:
- trends over time for different animals
- visualize which cameras are most active
- analyze which species are often captured together in the same location
- seasonal movement of wildlife

Figure 21: Stat's & Trends visualization



Dashboard

Stat's & Trends

Cameras & Numbers

Settings

North American Animal Classification

CAMERAS & NUMBERS

Filters:

Sightings:

Date	Time	Camera	Species

notes:
- sort list by camera or animal
- various filters can be applied

Figure 22: Camera & Numbers visualization

Key Features

Interactive Map	The map is the central feature of the dashboard. It displays animal sightings at their observation locations. Users can apply filters to toggle visibility by species or date. This way, it is easier to focus on specific data point. Beside this, there is an alert system, that notifies users of unusual sightings, such as “a black bear appearing in Nevada”.
Stat’s & Trends	This section provides in-depth analysis of wildlife data. It shows temporal trends, which helps users to understand, how species populations have changed over time. It also includes a co-occurrence analysis to reveal which species are frequently observed together. You can also see, which cameras are most active. Seasonal wildlife movement patterns are also visualized, offering insights into migration and behaviour.
Cameras & Numbers	Users can explore a detailed, sortable table that organizes data by camera, species, date and time. The filters allow for specific searches, such as focusing on a single camera or species.
Settings	The dashboard offers flexibility through its settings. Among other things, notifications can be configured to alert users of significant changes. These notifications can be delivered through various channels, including dashboard alerts, email updates, or push notifications. This ensures that the users are promptly informed.

Streamlit

We decided to create our dashboard with Streamlit. Streamlit is an open-source framework that simplifies the process of building interactive web applications for data analysis. It is well-suited for rapid prototyping and development. This enables us to focus on the core features of our dashboard without getting stuck in the complexity of web development.

With Streamlit, we can easily create dynamic and interactive visualizations such as maps, charts, and tables. It supports integration with popular Python libraries like pandas, NumPy, and Plotly, which makes it perfect for visualizing complex wildlife data. Streamlit's ability to handle real-time updates and notifications enhances the functionality of our dashboard and ensures that users stay informed of important changes in animal sightings.

It's simple and user-friendly interface allows us to quickly create and revise our dashboard. It makes it an ideal tool for creating an engaging and functional data visualization platform.

Design

The design of our dashboard focuses on usability and clarity. It should be easy for users to access and analyse wildlife data. The layout is kept minimalistic, with a clean sidebar navigation that allows users to switch easily between sections. This structure ensures that information is well-organized and easy to find.

The design of the dashboard includes green as one of the primary colours. Green is often linked to nature, growth, and balance, which aligns perfectly with the theme of wildlife conservation and ecological awareness. By using green in key elements such as headers, buttons, and charts, the dashboard highlights its connection to nature and wildlife.

To further improve the user experience, the dashboard uses a consistent and intuitive visual language. Interactive elements such as buttons, drop-down lists and filters are clearly labelled so that users can easily navigate through the platform.

Model Integration

Our project aimed to classify animals using live images and generate a map and statistics based on the results. However, due to the absence of a real-time data stream, we decided to structure our dashboard as follows: the map and visualized statistics are displayed using a test dataset.

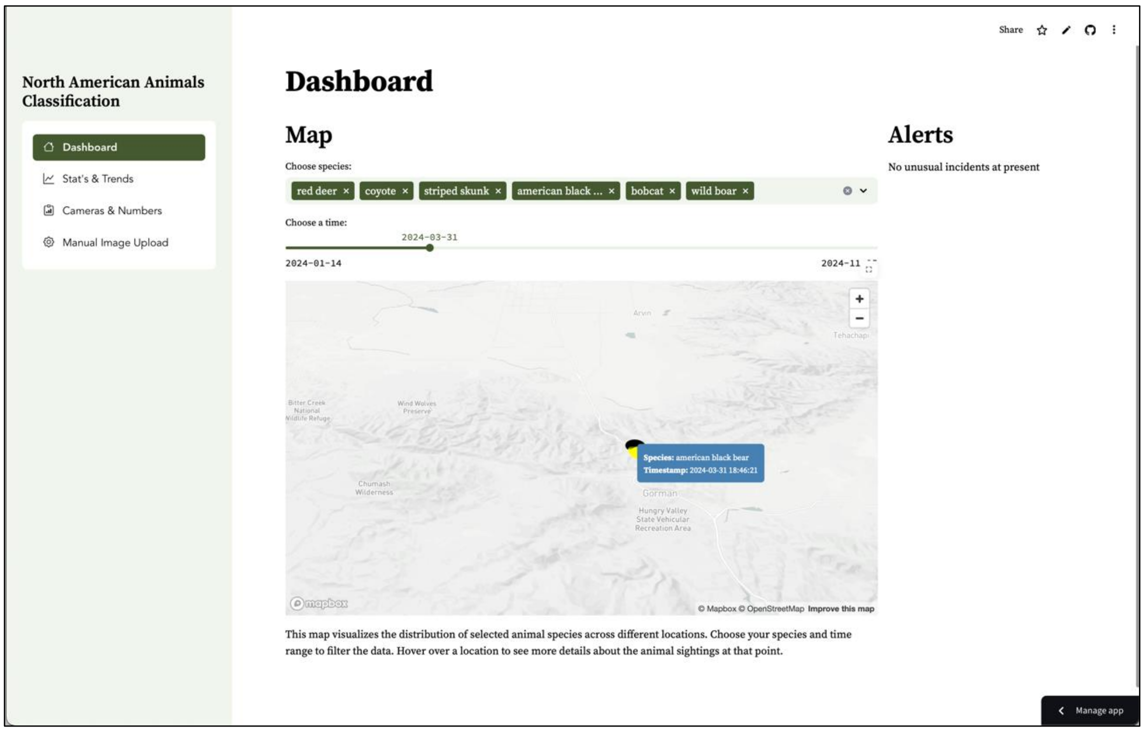
To demonstrate the functionality of our model, we added an additional page titled "manual image upload". On this page, users can manually upload images, and the dashboard will display the predicted species along with the corresponding probabilities as percentages.

After training our model, we saved its parameters using torch.save. This file was later loaded into our dashboard to utilize the trained model. Before classifying images with the “predict” function, the same preprocessing steps used during training are applied. These include resizing, cropping and normalization. Additionally, the model utilizes the 12 labels representing the target classes.

This approach allows us to showcase both the statistical insights derived from the test dataset and the real-time predictive capabilities of our model.

Our dashboard can be accessed via the following link: https://north-american-animals-classification.streamlit.app/

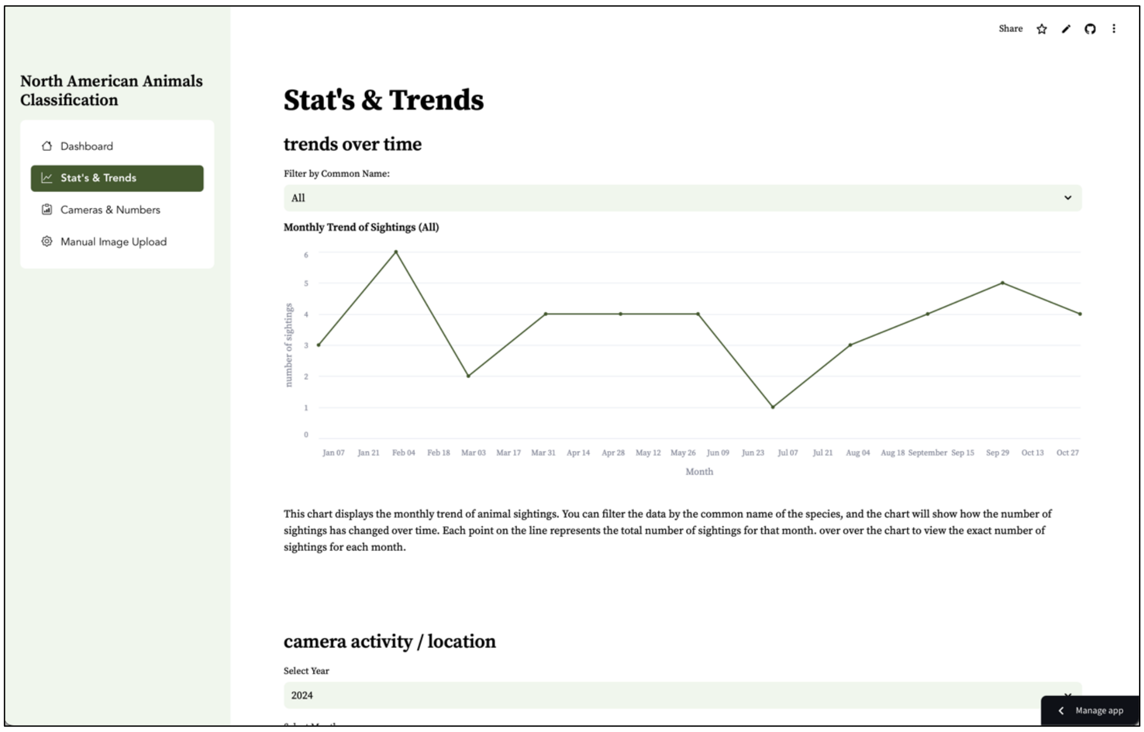
Below are screenshots of our final implementation with Streamlit.



No unusual incidents at present

This map visualizes the distribution of selected animal species across different locations. Choose your species and time range to filter the data. Hover over a location to see more details about the animal sightings at that point.

Figure 23: Dashboard final implementation



Stat's & Trends

trends over time

Filter by Common Name:

All

Monthly Trend of Sightings (All)

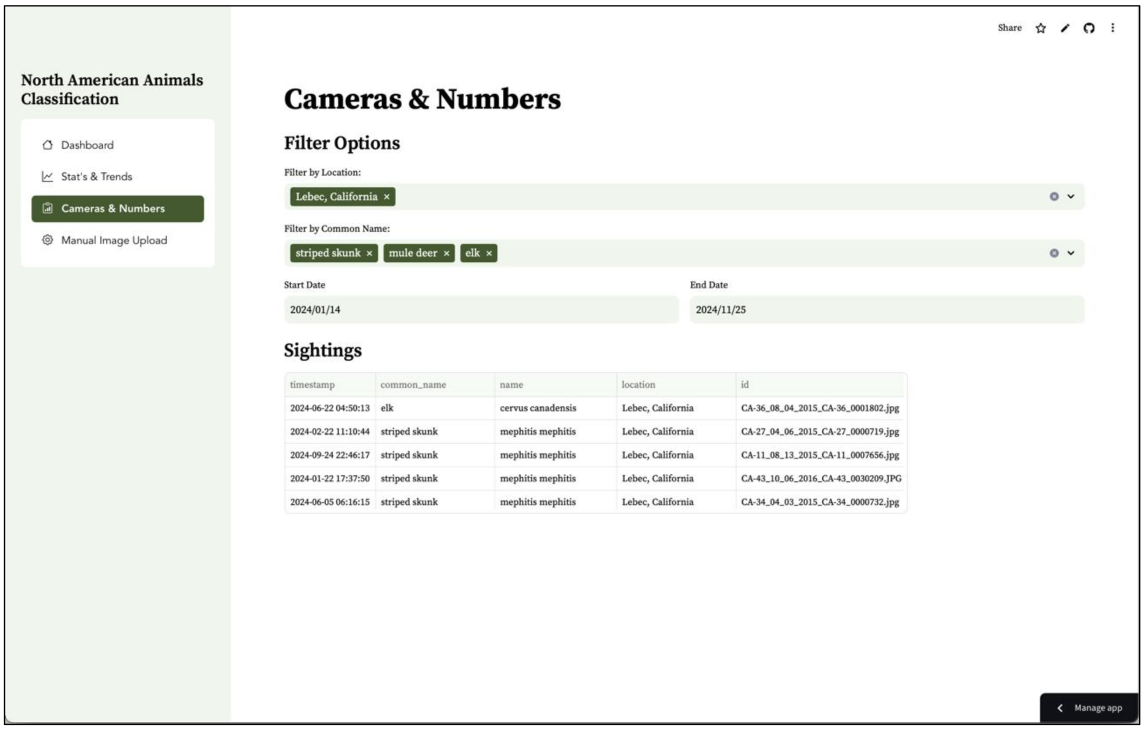
This chart displays the monthly trend of animal sightings. You can filter the data by the common name of the species, and the chart will show how the number of sightings has changed over time. Each point on the line represents the total number of sightings for that month. over over the chart to view the exact number of sightings for each month.

camera activity / location

Select Year

2024

Figure 24: Stat's & Trends final implementation



Cameras & Numbers

Filter Options

Filter by Location:

Lebec, California ×

Filter by Common Name:

striped skunk × mule deer × elk ×

Start Date

2024/01/14

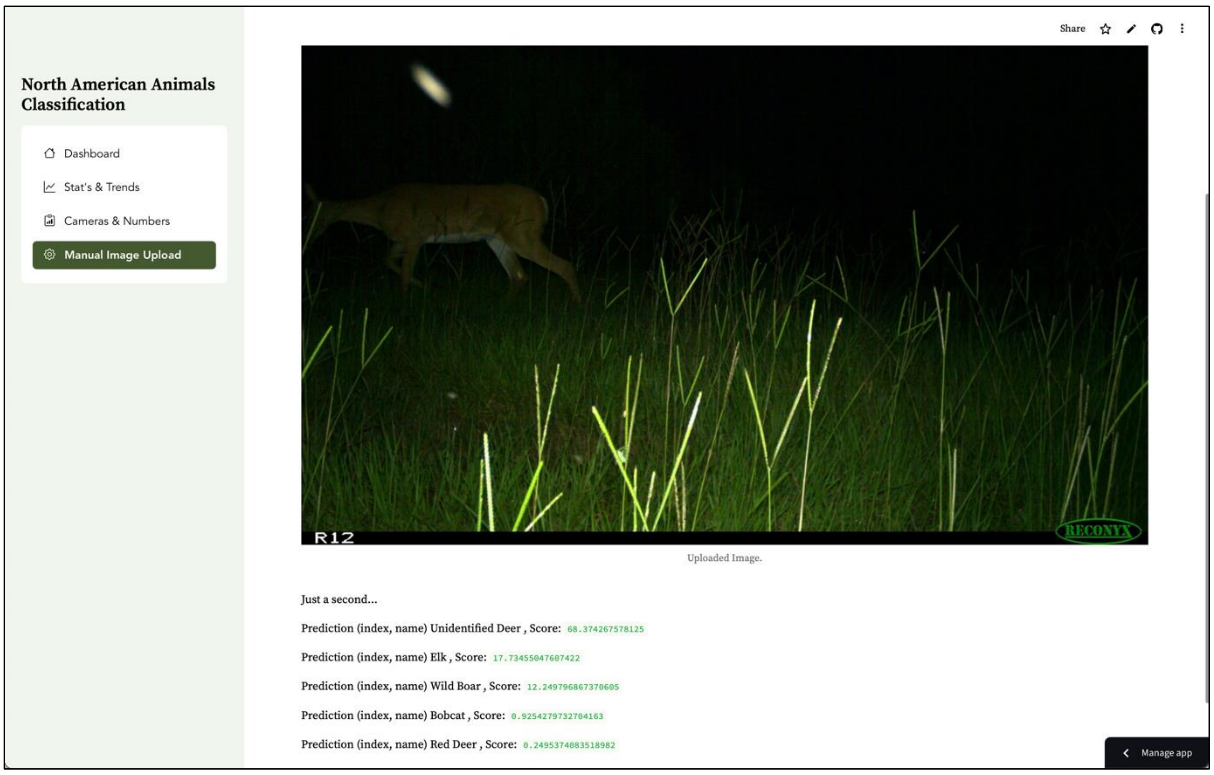
End Date

2024/11/25

Sightings

timestamp	common_name	name	location	id
2024-06-22 04:50:13	elk	cervus canadensis	Lebec, California	CA-36_08_04_2015_CA-36_0001802.jpg
2024-02-22 11:10:44	striped skunk	mephitis mephitis	Lebec, California	CA-27_04_06_2015_CA-27_0000719.jpg
2024-09-24 22:46:17	striped skunk	mephitis mephitis	Lebec, California	CA-11_08_13_2015_CA-11_0007656.jpg
2024-01-22 17:37:50	striped skunk	mephitis mephitis	Lebec, California	CA-43_10_06_2016_CA-43_0030209.JPG
2024-06-05 06:16:15	striped skunk	mephitis mephitis	Lebec, California	CA-34_04_03_2015_CA-34_0000732.jpg

Figure 25: Cameras & Numbers final implementation



```
Just a second...

Prediction (index, name) Unidentified Deer , Score: 68.374267578125

Prediction (index, name) Elk , Score: 17.73455047607422

Prediction (index, name) Wild Boar , Score: 12.249796867370605

Prediction (index, name) Bobcat , Score: 0.9254279732704163

Prediction (index, name) Red Deer , Score: 0.2495374083518982
```

Figure 26: Manual Image Upload final implementation

Results

Initial Model Performance Comparison

The performance of the three selected model architectures was evaluated to determine the best approach for our project. All models were tested on the same dataset, with accuracy as the primary evaluation metric. While we are aware that other metrics such as precision, recall, and F1-score could also be used, we chose to focus on accuracy for this initial comparison. Our goal was to identify the best-performing model as quickly as possible and proceed with further optimization. The training results are as follows:

Model	**Accuracy on validation set**
EfficientNet	89%
ResNet	85%
Vision Transformer	68%

Table 6: Model Accuracy Comparison

We decided to continue with ResNet for further optimization based on its strong performance in the initial evaluation. While EfficientNet achieved slightly higher accuracy (89%) compared to ResNet (85%), the difference wasn't significant enough to make a final choice. ResNet also trained relatively quickly which allows for faster experimentation and adjustments. It is important to note that we do not know if any of the models have overfitted the training data. Based on these results and our experience with the models, we decided to proceed with ResNet for further optimization.

ResNet without training

Since ResNet originally has 1'000 classes, including some animals of our dataset, we ran it without training on 50 pictures per category. The following classes are included in ResNet: American black bear, gray fox, coyote and wild boar. Bear, squirrel and skunks are also present but for different types. Only the highest probability was considered.

The three included classes were able to identify some pictures correctly but failed on most of them. In addition, the model was not consistent and classified each category as many different classes. Between 9 and 29 categories were recognized for a single class of our dataset. An addition analysis showed that "American Black Bear" was mistaken 4 times with a different type of bear. Squirrel and skunk were never mistaken with other types.

Class	Accuracy (total=50)	Accuracy [%]	Total number of classes
American Black Bear	16 4 as other bear types	32	26
Bobcat	0	0	20
California Ground Squirrel	0	0	28
Coyote	14	28	22
Elk	0	0	25
Empty	0	0	32
Gray Fox	0	0	9
Mule Deer	0	0	21
Red Deer	0	0	29
Striped Skunk	0	0	10
Unidentified Deer	0	0	20
Wild Boar	18	36	24

Table 7: ResNet Accuracy without training

ResNet Optimization

After selecting ResNet, we proceeded with additional optimization steps to further enhance its performance. We attempted to add random rotations, horizontal flips and color jitter to avoid overfitting and reduce the gap between the training and validation result (94% and 85% respectively). This led to 86% of accuracy on the training set. While the validation and testing accuracies were closer to the training one, they cannot be considered. Due to a mistake, the data augmentation was also performed on the validation and testing set, which biased the results. However, the results, while supposed to be overestimated due to the bias, remained below the results without data augmentation.

So far, the model was trained with only one fully connected layer replacing the output layer. Attempts were also made to add more layers. Unfortunately, the same mistake was done with the data augmentation. However, the outcome remained lower than the models with a single fully connected layer: (-3%) on training and (-6%) on biased validation. The model learnt longer, for 16 epochs. Since the results were less promising, the model was not tested.

The best tested results come from the model using no data augmentation; just resizing (128x128 pixels) and normalization with ImageNet values. The model stops learning after 13 epochs, where it obtains its best total accuracy (86.98%). It drops out five epochs later, following the implemented patience. We can observe the training curve reaching almost 100%, and the gap with the validation remaining until the end.

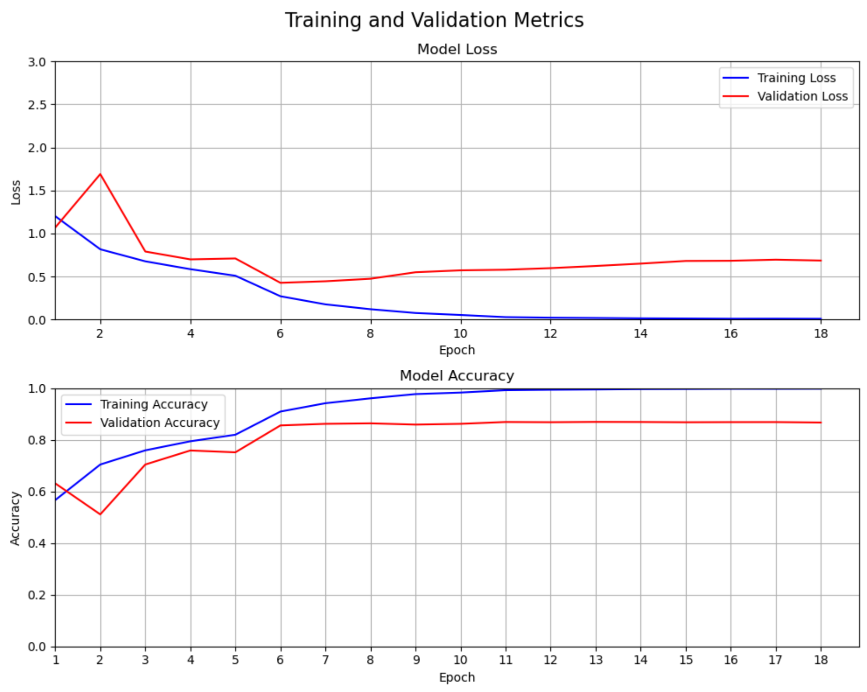


Figure 27: ResNet training and validation metrics

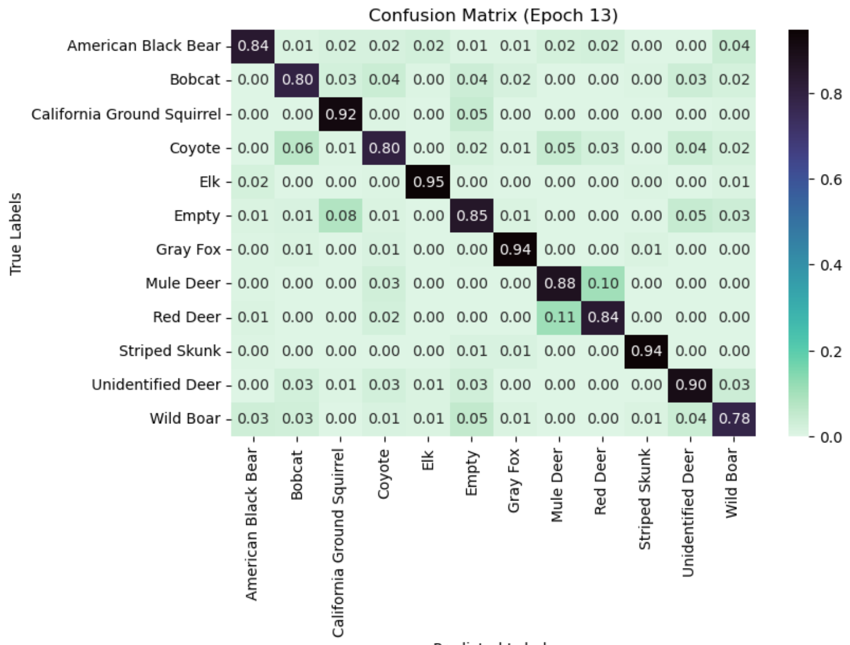
Additional Performance Metrics

The performance metrics show consistency between the validation and testing results. In addition, precision and recall indicate that the model makes a balanced number of false positive and false negative, relative to true positives.

ResNet	Validation	Testing
Accuracy	86.98%	86.57%
Precision	0.8696	0.8659
Recall	0.8697	0.8657
F1-Score	0.8694	0.8654

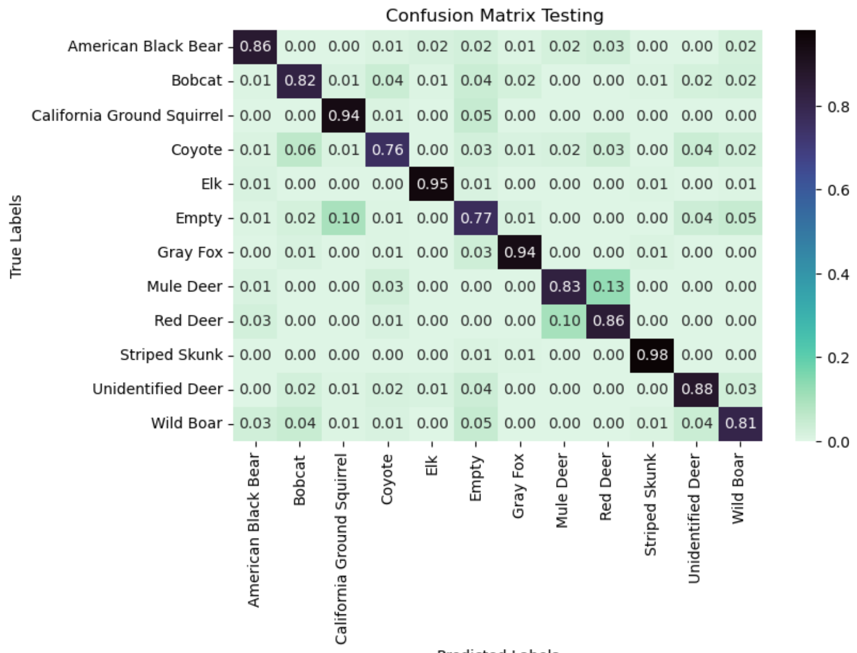
Table 8: ResNet Additional Performance Metrics

Confusion Matrices



True \ Predicted	American Black Bear	Bobcat	California Ground Squirrel	Coyote	Elk	Empty	Gray Fox	Mule Deer	Red Deer	Striped Skunk	Unidentified Deer	Wild Boar
American Black Bear	0.84	0.01	0.02	0.02	0.02	0.01	0.01	0.02	0.02	0.00	0.00	0.04
Bobcat	0.00	0.80	0.03	0.04	0.00	0.04	0.02	0.00	0.00	0.00	0.03	0.02
California Ground Squirrel	0.00	0.00	0.92	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00
Coyote	0.00	0.06	0.01	0.80	0.00	0.02	0.01	0.05	0.03	0.00	0.04	0.02
Elk	0.02	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.01
Empty	0.01	0.01	0.08	0.01	0.00	0.85	0.01	0.00	0.00	0.00	0.05	0.03
Gray Fox	0.00	0.01	0.00	0.01	0.00	0.00	0.94	0.00	0.00	0.01	0.00	0.00
Mule Deer	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.88	0.10	0.00	0.00	0.00
Red Deer	0.01	0.00	0.00	0.02	0.00	0.00	0.00	0.11	0.84	0.00	0.00	0.00
Striped Skunk	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.94	0.00	0.00
Unidentified Deer	0.00	0.03	0.01	0.03	0.01	0.03	0.00	0.00	0.00	0.00	0.90	0.03
Wild Boar	0.03	0.03	0.00	0.01	0.01	0.05	0.01	0.00	0.00	0.01	0.04	0.78

Figure 28: Results validation



True \ Predicted	American Black Bear	Bobcat	California Ground Squirrel	Coyote	Elk	Empty	Gray Fox	Mule Deer	Red Deer	Striped Skunk	Unidentified Deer	Wild Boar
American Black Bear	0.86	0.00	0.00	0.01	0.02	0.02	0.01	0.02	0.03	0.00	0.00	0.02
Bobcat	0.01	0.82	0.01	0.04	0.01	0.04	0.02	0.00	0.00	0.01	0.02	0.02
California Ground Squirrel	0.00	0.00	0.94	0.01	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00
Coyote	0.01	0.06	0.01	0.76	0.00	0.03	0.01	0.02	0.03	0.00	0.04	0.02
Elk	0.01	0.00	0.00	0.00	0.95	0.01	0.00	0.00	0.00	0.01	0.00	0.01
Empty	0.01	0.02	0.10	0.01	0.00	0.77	0.01	0.00	0.00	0.00	0.04	0.05
Gray Fox	0.00	0.01	0.00	0.01	0.00	0.03	0.94	0.00	0.00	0.01	0.00	0.00
Mule Deer	0.01	0.00	0.00	0.03	0.00	0.00	0.00	0.83	0.13	0.00	0.00	0.00
Red Deer	0.03	0.00	0.00	0.01	0.00	0.00	0.00	0.10	0.86	0.00	0.00	0.00
Striped Skunk	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.98	0.00	0.00
Unidentified Deer	0.00	0.02	0.01	0.02	0.01	0.04	0.00	0.00	0.00	0.00	0.88	0.03
Wild Boar	0.03	0.04	0.01	0.01	0.00	0.05	0.00	0.00	0.00	0.01	0.04	0.81

Figure 29: Results test set

These results indicate a good performance overall and between classes. Accuracy per class is between 76% and 98%. The Mule and Red Deer were mostly mistaken within another. The specificity of their family can be difficult for the model to identify, especially if the represented animal is far away, in the dark or only partially visible. Similarly, the California Ground Squirrel was mostly mistaken with an empty picture. This small size of the animal explains the confusion. Overall, the confused classes trigger no surprise and are easily explained. While these results are promising, they outperform the baseline by only 1% (85% on validation set).

To improve the model further, some exploration was made with bigger input pictures (256 x 256 pixels). As a result, the models learn for more epochs, both for the model with a single additional layer and model with additional layers.

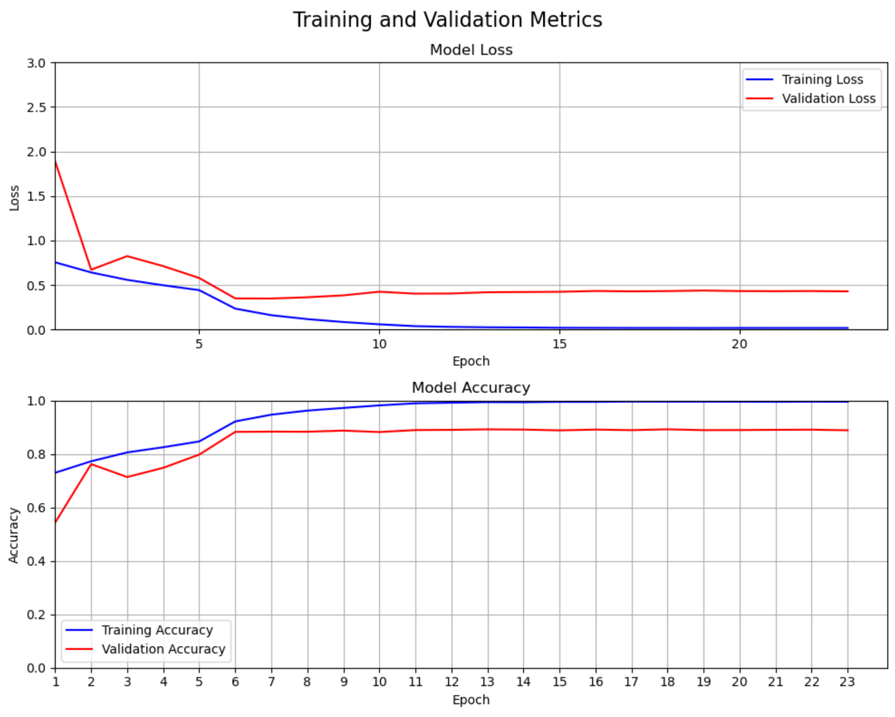


Figure 30: training curve of our best model with input images of 256x256 instead of 128x128 pixels.

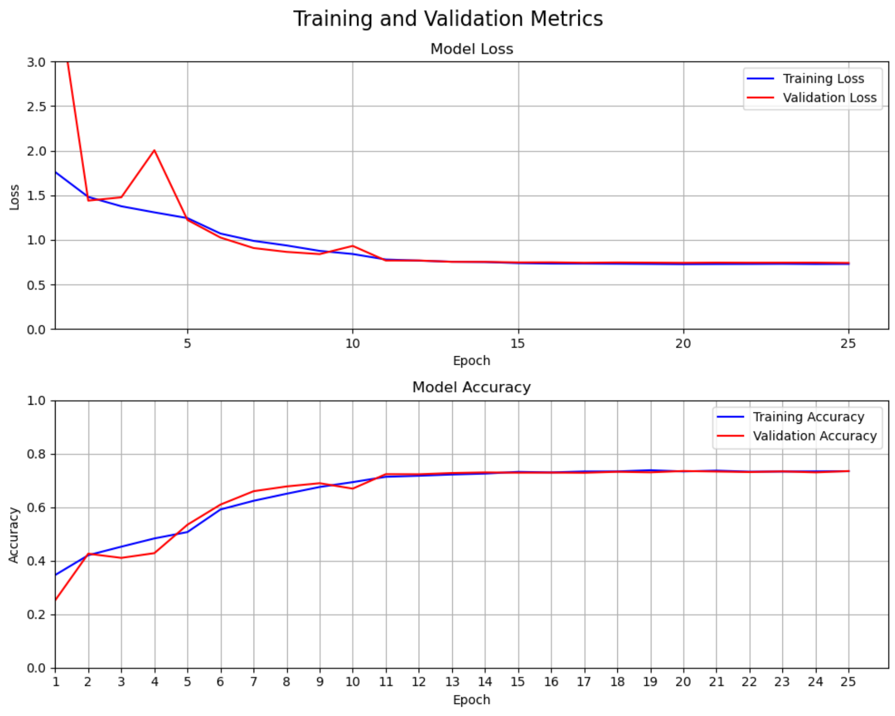


Figure 31: Training curve of the model with additional layers using 256x256 pictures as input

Discussion

Limitations

We must point out some limitations of our project. By choosing to work with only one model and focusing mainly on accuracy, we may have overlooked other important factors such as precision, recall and F1 score. These metrics are beside accuracy also important to understand if the model performs well on new, unknown data.

When we compare our results with other existing projects, we see some similarities in how the models performed, but also some differences that need to be investigated further. For example, the ViT didn't do better than ResNet or EfficientNet in our tests, which could be because of differences in our data or how we trained the models.

Future work

To enhance the robustness and accuracy of our classification system, we would continue with all three models and optimize them individually. This would allow us to better understand their performance on our dataset and identify their strengths and weaknesses in more detail. By fine-tuning each model and experimenting with different hyperparameters, we could improve their performance. It would be worth exploring their respective parameter (e.g. optimizer, early stop with regularization) to find the right fit. This would give us a more comprehensive view of which model is best suited for our project and how each model can be optimized to achieve its best performance.

Since we faced overfitting issues, expanding our dataset could improve the model's ability to generalize. More images would allow models to become familiar with different poses, environments and lighting conditions. This improves their ability to accurately classify animals in different situations.

Incorporating MLOps (Machine Learning Operation) best practices could improve our workflow by automating model training and validation processes. This would allow for faster iterations and easier updates.

Finally, we could extend the model to more species, such as rabbits, badgers, wolves, and others, the models could learn form a wider variety of animals. This would help them make predictions more appropriate to real-life applications.

Summary

This project focused on using machine learning to make wildlife monitoring faster and more accurate. Using images from camera traps, our team built a system that identifies animal species automatically. This saves researchers time and effort compared to manually analyzing thousands of images.

A carefully selected and balanced dataset ensured fairness during training, which helped the models perform more reliably. Among the models tested, ResNet seemed the most promising option for optimization. Through optimization, we were able to improve ResNet's performance by 1%.

To make the results accessible and practical, a user-friendly dashboard has been created. This dashboard displays animal trends, maps their locations and sends alerts for unusual sightings. This makes it easy to analyse and understand the data. By automating animal identification, this system supports researchers and conservationists in their efforts to protect wildlife. It speeds up the monitoring process, reduces errors and contributes to better decision-making for conservation strategies.

Conclusions

In this project, we successfully developed a machine learning-based system for classifying North American wildlife species from camera trap images.

Of the three models we tested—ResNet, EfficientNet, and ViT—we chose to continue with ResNet. It achieved a validation accuracy of 85% on our first attempt, and we saw great potential to improve it further through optimization.

EfficientNet performed slightly better, with an initial validation accuracy of 89%. On the other hand, the ViT underperformed, with a validation accuracy of just 68%. This was likely because our dataset of 36’000 images across 12 categories was too small. ViTs generally need much larger datasets to learn effectively.

After optimizing the ResNet model, we achieved a test set accuracy of 87%.

Expectations

Did our results meet expectations?

Overall, the results aligned with our expectations. We initially thought it would be more challenging to achieve good results, but since we only used powerful models, we were able to achieve promising performance right from the start. The performance of the ViT model was a bit disappointing, but it was also to be expected, given that ViTs generally require much larger datasets to train effectively. Our dataset wasn’t large enough for ViT to perform optimally.

While we expected ResNet to perform well, we had hoped to further optimize it to achieve at least 90% accuracy after optimization. Perhaps if we tried a different optimization approach, we could have reached our goal of at least 90%.

Problem Insights

Traditionally, researchers manually go through thousands of camera trap images to identify animals. This process is not only very slow but also prone to mistakes, as humans can easily overlook or misidentify animals, especially after hours of reviewing images.

With our project, we aimed to make this process easier and faster. By automating the image analysis, the system can scan through huge amounts of data in a fraction of the time it would take a human. What used to take days or even weeks now only takes hours. This means researchers can save a lot of time, reduce mistakes, and focus on more important work, like studying animal behaviour or planning conservation strategies. It’s a big step towards making wildlife monitoring more efficient and effective.

Learnings

At the start of this project, our team had no prior experience with machine learning projects and GitHub. We were uncertain about where to start and how the entire process worked. We often didn't know if what we were doing was right or wrong. Coaching, reading and exchanges with other students helped us to be more confident and implement tools to assess our work easily (e.g. metrics and csv files).

Over the course of the project, our team gained valuable insights and experiences that have contributed to our professional and personal growth. We are now more confident and excited about our future machine learning challenges.

Technical Skills

As this was our first project in the field of machine learning, we were moving into unknown territory. We had to quickly familiarize ourselves with the basic concepts and tools, often learning by trial and error. Our first big challenge was finding and handling our dataset. We needed to figure out, what data was good and which images were useful.

Next, we needed to find some models. We didn't know which models were good in the task of image classification. While we were familiar with famous model, we were uncertain which one suited our case and how to find it out. Through research and experimentation, we explored different models such as EfficientNet, ResNet, and ViTs. As each of us tried to start and work with one of these models, we all gained important insights. Even when we decided to only focus on one model and to improve it, we discussed and helped each other.

Working with GitHub was an essential aspect of our project. As it was our first time using GitHub, we quickly learned the importance of effective branching and commit practices. Committing PyTorch files created issue was a great way to learn not to commit heavy files. We also realized the value of frequent, smaller commits.

Collaborative Teamwork

Effective communication and time management was key to our project's success. Our team relied heavily on Discord for real-time exchange. As well as solving problems, Discord was valuable for brainstorming ideas and making important decisions together.

Personal and Team Development

On a personal level, this project taught us the importance of just starting, trying things out and seeing failure as an important part of the learning process. We quickly realized that it is better to try and fail than to delay progress trying to do everything correctly. Every setback became an opportunity to learn, adapt and improve. Moving forward, we know that taking action, even when we are unsure, is an essential part of growing. These lessons will be priceless as we approach future projects in machine learning and other fields.

Usage of ChatGPT

This document was created with the assistance of ChatGPT, an AI language model developed by OpenAI. The model supported this project by guiding us through the process, providing suggestions, and helping refine the language throughout. While efforts have been made to ensure the accuracy and quality of the information, the author(s) remain fully responsible for the final content.

References

Apps, P., & McNutt, J. W. (2018, November 29). *How camera traps work and how to work them*. Retrieved from WILEY Online library: https://onlinelibrary.wiley.com/doi/10.1111/aje.12563

Beery, S., Cole, E., & Gjoka, A. (2020, April 21). *The iWildCam 2020 Competition Dataset*. Retrieved from arXiv: https://arxiv.org/abs/2004.10340

Beery, S., van Horn, G., & Perona, P. (2018, July 25). *Recognition in Terra Incognita*. Retrieved from arXiv: https://arxiv.org/abs/1807.04975

Brook, B., Buettel, J., & Aandahl, Z. (2023, December 12). *MEWC: A user-friendly AI workflow for customised wildlife-image classification*. Retrieved from EcoEvoRxiv: https://ecoevorxiv.org/repository/view/6405/

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009, June). *ImageNet: A Large-Scale Hierarchical Image Database*. Retrieved from ResearchGate: https://www.researchgate.net/publication/221361415_ImageNet_a_Large-Scale_Hierarchical_Image_Database

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., . . . Houlsby, N. (2021, June 3). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. Retrieved from arXiv: https://arxiv.org/abs/2010.11929

GeeksforGeeks. (2024, March 20). *What is Adam Optimizer?* Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/adam-optimizer/

Hanski, I. (2011, March 18). *Habitat Loss, the Dynamics of Biodiversity, and a Perspective on Conservation*. Retrieved from Springer Nature Link: https://link.springer.com/article/10.1007/s13280-011-0147-3

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). *ImageNet Classification with Deep Convolutional Neural Networks*. Retrieved from NeirIPS: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Li, J., Qiao, Y., Xu, H., & Huang, S. (2024, June 14). *RSEND: Retinex-based Squeeze and Excitation Network with Dark Region Detection for Efficient Low Light Image Enhancement*. Retrieved from arXiv: https://arxiv.org/abs/2406.09656?

Microsoft. (2021, June 21). *Wildlife Protection Solutions helps protect the wildest places with Microsoft AI for Earth*. Retrieved from Microsoft: https://www.microsoft.com/en/customers/story/1384184517929343083-wildlife-protection-solutions-nonprofit-ai-for-earth

Microsoft. (2024, December 18). *CameraTraps*. Retrieved from GitHub: https://github.com/microsoft/CameraTraps?tab=readme-ov-file

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., . . . Fei-Fei, L. (2015, January 30). *ImageNet Large Scale Visual Recognition Challenge*. Retrieved from arXiv: https://arxiv.org/abs/1409.0575

Simonyan, K., & Zisserman, A. (2015, April 10). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Retrieved from arXiv: https://arxiv.org/abs/1409.1556

Stevens, E., Antiga, L., & Viehmann, T. (2020, July). *Deep Learning with PyTorch* . Retrieved from MANNING: https://www.manning.com/books/deep-learning-with-pytorch

Tan, M., & Le, Q. (2020, September 11). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. Retrieved from arXiv: https://arxiv.org/abs/1905.11946

Wearn, O., & Glover-Kapfer, P. (2017). *Conservation Technology Camera-Trapping.* Retrieved from WWF: www.wwf.org.uk/sites/default/files/2019-04/CameraTraps-WWF-guidelines.pdf

Wei, Z., Wang, Y., Sun, L., Vasilakos, A., & Wang, L. (2023, December 20). *ClassLIE: Structure- and Illumination-Adaptive Classification for Low-Light Image Enhancement*. Retrieved from arXiv: https://arxiv.org/abs/2312.13265?

Wildlife Insights. (n.d.). *Wildlife Insights*. Retrieved from Wildlife Insights: https://www.wildlifeinsights.org/standards

Yoccoz, N., Nichols, J., & Boulinier, T. (2001, August 01). *Monitoring of biological diversity in space and time*. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/pii/S0169534701022054

Zhang, J., Cao, L., Lai, Q., Li, B., & Qin, Y. (2023, March 2). *BIFRNet: A Brain-Inspired Feature Restoration DNN for Partially Occluded Image Recognition*. Retrieved from arXiv: https://arxiv.org/abs/2303.01309?

List of Figures

List of Tables