PyTorchWildlife Microsoft | AI for Good Lab

Megadetector maintenance & improvement pipeline

Andrés Hernández, Zhongqi Miao, Rahul Dodhia, Amrita Gupta, Md Nasir.

The purpose of this proposal is to outline a plan for the maintenance and improvement of Megadetector. The proposed plan is divided into four main stages: packaging, improvements, customer service and marketing. The development on these stages seeks to position Megadetector as a tool for animal detection for users with and without programming knowledge. Figure 1 shows a summary of the plan. Firstly, Megadetector will be converted into a Python package for ease of use and implementation in other pipelines. Then, several modifications to Megadetector's main architecture will be done to increase the performance in some animal categories, such as reptiles and aquatic species. Third, the customer service stage will address requests and recommendations through the development of a contact platform that does not rely on a live service. Finally, the fourth stage will tackle marketing and service promotion to extend the use of Megadetector worldwide and to position this model as the golden standard for animal detection. This work will be done by a multidisciplinary group from the Microsoft AI for Good Lab.

Figure 1. Stages for the management and improvement of Megadetector.

The following table shows an estimate timeline for the initial development of the four stages until the end of 2023.

Stage \ Month	Oct	Nov	Dec	Jan	Feb
Packaging (Core) 870 hours 1-2 Research scientists					
Improvements 570 hours 1-2 Research scientists					
Marketing 240 hours 1 Person with knowledge in customer service and marketing					
Customer Service 240 hours 1 Person with knowledge in customer service and marketing					

Project workload:

The estimated workload suggests the requirement of two research scientists for the Packaging and Improvement stages with 203 hours of work per month per scientist, as well as one person with experience in marketing and customer service with a workload of 120 hours of work per month.

First stage: Packaging

Estimated human effort: 1-2 full time scientists, 870 hours of work in total.

Following the approach of popular computer vision libraries that work hand to hand with Pytorch [1] such as Torchvision [2] and Torchgeo [3], the Megadetector repository will be converted into a fully integrated Python package named **PyTorchWildlife**. The development of the features inside the package is divided into five steps. Figure 2 shows the order of development for these stages. **The first step focuses on the core implementation of PyTorchWildlife**, where the current Megadetector pipeline will be available for the user. Then, **PyTorchWildlife** will be connected to camera trap datasets available through LILA for ease of access. Thirdly, PyTorchWildlife will offer classification models for the user to choose from. Then, a set of utility functions will be available for the user to perform preprocessing, data splitting and post-processing. Finally, the Accessibility step will focus on the development of a user interface to extend PyTorchWildlife's capabilities to a userbase that does not have programming knowledge.

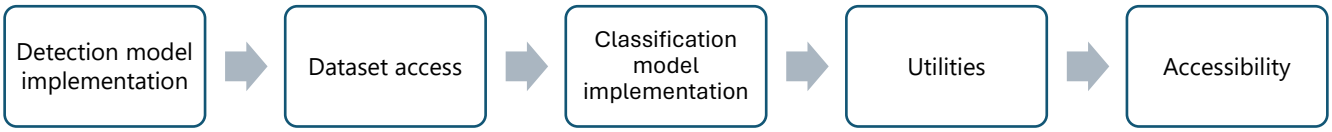


Figure 2. Steps for Megadetector's packaging stage.

1.0 Stage objectives and deliverables

The main objective of the first stage is to convert Megadetector into a Python package called **PyTorchWildlife**. The following list shows the deliverables and functions that will be developed for the launch of **PyTorchWildlife**:

- A fully integrated Python package for animal detection and classification in images and video.
- A feature to choose from two image detection models and one video detection model.
- A feature to choose from one image classification model and one video classification model.
- A feature to access and download datasets available on LILA.
- A feature to create custom datasets for finetuning and inference.
- A module to perform data splitting, data folder creation and image preprocessing and augmentation if needed.
- A user interface to perform animal detection and classification.
- A set of functions supporting the development of such interfaces.

1.1 Detection model implementation (core):

Being able to train and perform inferences on different detection models is essential to refine PyTorchWildlife to each of the user's needs. The following list shows the deliverables and features for this step. Each feature should come with a set of unit tests to assess good implementation.

Workload:

The estimated development time for this task is 250 hours of work for 1-2 full-time research scientists.

1.1.1 A modular and flexible model loading module that allows users to choose from a set of detection models.

- A feature to load Megadetector's YOLOv5 detection model.
- A feature to load Megadetector's pretrained weights in YOLOv5.

1.1.2 A module for fine-tuning and training of pre-defined models from scratch using custom datasets.

- A trainer module for fine-tuning and training from scratch using custom datasets.

1.1.3 A set of Jupyter Notebooks that document and guide the user into how to define, load and perform inference on the predefined models.

- A Notebook to create, load and run inference on each of the implemented detection models.
- A Notebook to fine-tune or train a model using a dataset available in LILA.

1.2 Dataset access:

One key component to promote scientific research using PyTorchWildlife is to implement a module that enables the user to access popular LILA datasets with ease. HuggingFace currently offers a function to access LILA datasets. This function will be used as part of our dataset loading module. The following list shows the deliverables and features for this step. Each feature should come with a set of unit tests to assess good implementation.

Workload:

The estimated development time for this task is 40 hours of work for 1-2 full-time research scientists.

1.2.1 A dataset loading module that allows the user to access datasets available in LILA.

- A feature to download a LILA dataset.

- A feature to create a dataset class from a given LILA dataset that includes visualization functions.
- A feature to create a data loader for the dataset class.

1.2.2 A dataset module to create custom datasets.

- A function to define a custom dataset and dataloader.
- A set of visualization functions for the dataset.
- A set of pre-processing functions such as transformations.

1.2.3 A Jupyter Notebook to showcase how to use the dataset functions.

- A Notebook to show how to use a LILA dataset using the loading module.
- A Notebook to show how to create a custom dataset.

1.3 Classification model implementation:

One of the most important tasks in the assessment of biodiversity involves the classification of the species present in camera traps. Being able to train and perform inferences on different detection models is essential to refine PyTorchWildlife to each of the user's needs. The following list shows the deliverables and features for this step. Each feature should come with a set of unit tests to assess good implementation.

Workload:

The estimated development time for this task is 250 hours of work for 1-2 full-time research scientists.

1.3.1 A modular and flexible model loading module that allows users to choose from a set of classification models.

- A feature to load the classifier model from Megadetector.
- A feature to load the weights of Megadetector's classifier weights.

1.3.2 A module for fine-tuning and training of pre-defined models from scratch using custom datasets.

- A trainer module for fine-tuning and training from scratch using custom datasets.

1.3.3 A set of Jupyter Notebooks that document and guide the user into how to define, load and perform inference on the predefined models.

- A Notebook to create, load and run inference on each of the implemented detection models.
- A Notebook to fine-tune or train a model using a dataset available in LILA.

1.4 Utilities:

Data from camera traps come from a variety of ecosystems and environmental conditions. A utility module will help the users to address specific needs regarding processing and

organization of the data for custom datasets. The following list shows the deliverables and features for this step. Each feature should come with a set of unit tests to assess good implementation.

Workload:

The estimated development time for this task is 80 hours of work for 1-2 full-time research scientists.

1.4.1 A utilities module that allows the user to perform pre-processing, data splitting and post-processing if needed.

- A feature to perform pre-processing operations on the data.
- A feature to perform data splitting and creating folds from the dataset.
- A feature to perform post-processing on the predictions and the data.

1.4.2 A video detection pipeline for animal detection in videos.

- A function to split a video into frames based on the recording framerate.
- A function to draw bounding boxes on a set of frames and reconstruct the video.
- A set of visualization functions to sample frames on a dataset.

1.4.3 A Jupyter Notebook showcasing how to use the utility functions and the video detection pipeline.

- A Notebook to show how to perform pre-processing, data splitting and post-processing.
- A Notebook to show how to use the video detection pipeline.

1.5 Usability:

A significant portion of Megadetector's target audience is not proficient in programming. Implementing a user interface to use PyTorchWildlife services is a natural solution for this shortcoming. The following list shows the deliverables and features for this step. Each feature should come with a set of unit tests to assess good implementation.

Workload:

The estimated development time for this task is 250 hours of work for 1-2 full-time research scientists.

1.5.1 A user-friendly interface that allows non-programmers to interact with PyTorchWildlife easily.

- A feature to load images from a directory.
- A feature to split the images and create folders if needed.
- A feature to perform inference on the split data.

- A feature to visualize and save results.

1.5.2 A video detection framework for the user interface

- A feature to load videos from a directory.
- A feature to perform inference on the split data.
- A feature to visualize and save detection results.

1.5.3 A video classification framework for the user interface

- A feature to load videos from a directory.
- A feature to perform inference on the split data.
- A feature to visualize and save detection results.

1.5.4 A complete documentation on how to use the user interface for image and video detection.

- Documentation on how to use the user interface for data organization.
- Documentation on how to perform inference in images.
- Documentation on how to perform inference in videos.

The following table shows an estimate timeline and work intensity for the development of these features.

PyTorchWildlife Microsoft | AI for Good Lab

Stage \ Month	Oct	Nov	Dec	Jan	Feb
Detection Model implementation (Core) 250 hours 1-2 research scientists	■	■	■		
Dataset access 40 hours 1-2 research scientists	■	■			
Classification Model implementation 250 hours 1-2 research scientists			■	■	■
Utilities 80 hours 1-2 research scientists			■	■	
Accessibility 250 hours 1-2 research scientists				■	■

Second stage: Improvements

Estimated human effort: 2 full time research scientists for 570 hours.

Improving the performance of PyTorchWildlife is essential to keep it as a golden standard in animal detection. Figure 3 shows the current improvement pipeline. There are currently two main steps for this stage: First, data-driven improvements are related to the acquisition of new data through alliances and partnerships. Second, model-driven improvements are associated with the development of new deep learning architectures for animal detection through internal

and external research. *This stage is open to new thoughts and ideas on how to improve torch Wildlife.*

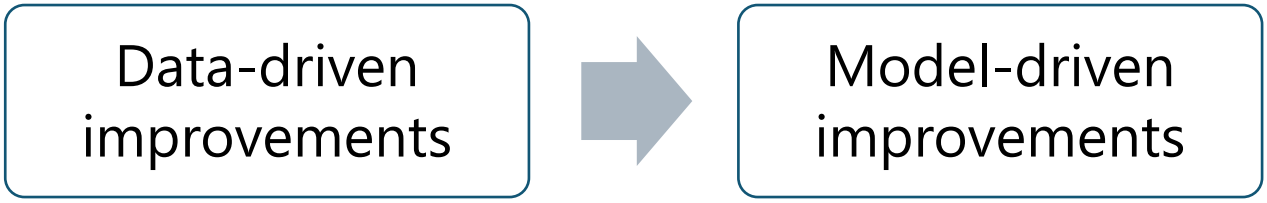


Figure 3. Steps for PyTorchWildlife's improvements stage.

2.1 Data-driven improvements:

The performance of a model is directly related to the amount of data it was trained with. The acquisition of new data through alliances and partnerships is essential to help PyTorchWildlife perform in a wide variety of ecosystems and regions. The following list shows the current objectives for data-driven improvements.

Workload:

The estimated development time for this task is 250 hours of work for 1-2 full-time research scientists.

- 2.1.1 Gathering data to improve detection in reptiles in the state of Florida.
  - Collaboration with USGS.
- 2.1.2 Gathering data to improve detection in reptiles in Australia.
  - Collaboration partner pending.
- 2.1.3 Gathering data to improve detection in subaquatic species.
  - Collaboration partner pending.

2.2 Model-driven improvements:

Keeping with the state-of-the-art in object detection and classification is a key for Torch Wildlife's success. Model-driven improvements are associated with the use of improved deep learning architectures that are a product of internal or external research. The end goal after finishing the implementation of the codebase currently available in Megadetector is to develop machine learning models that achieve state-of-the-art results in animal detection and classification as the product of internal research, as well as expanding PyTorchWildlife to include Natural Language Processing and animal segmentation.

Workload:

The estimated development time for this task is 320 hours of work for 1-2 full-time research scientists.

2.2.1 Implement and train the Deformable DETR detection model on the Megadetector dataset.

- Add the Deformable DETR detection models.
- Add the Deformable DETR pretrained weights on the Megadetector dataset.

2.2.2 Implement and train the Swin Transfomer classification model on the Megadetector dataset.

- Add the Swin Transformer classification models.
- Add the Swin Transformer pretrained weights on the Megadetector dataset.

2.2.3 Implement an end-to-end video detection pipeline using the DINOv2 architecture.

- Implement DINOv2 for end-to-end video detection.
- Add DINOv2 pretrained weights.

2.2.4 Implement an end-to-end video classification pipeline.

- Design a novel pipeline for animal video classification.
- Add the pretrained weights of the classification model.

2.2.5 Design, train and develop a state-of-the-art model for animal segmentation and classification.

- Design a novel architecture for animal segmentation and classification.
- Implement the novel detection model developed at AI4G.
- Add the pretrained weights of the detection model.

2.2.6 Develop Jupyter notebooks that explain the new models and pipelines.

- A notebook to show how to use the video classification pipeline.
- A notebook to show how to use the state-of-the-art model for animal segmentation and classification.

The following table shows an estimate timeline and work intensity for the development of the features.

Stage \ Month	Oct	Nov	Dec	Jan	Feb
Data-driven improvements 250 hours 1-2 research scientists					
Model-driven improvements 320 hours 1-2 research scientists					

Third stage: Customer Service

Estimated human effort: 1 person with experience in marketing and customer service for 240 hours.

Developing strategies to contact existing partners and to handle user requests is key. This *stage is in development, and it is open to new thoughts and ideas on how to improve customer service in torch Wildlife.*

Figure 4. Steps for PyTorchWildlife's customer service stage.

3.1 Retention of existing partners:

One method to strengthen relationships with existing partners is to establish direct communication channels and to offer personalized assistance. The following list shows the current objectives for this stage.

Workload:

The estimated development time for this task is 80 hours of work for one person with experience in customer service and marketing.

3.1.1 To establish direct communication channels with existing partners.

- Use information from the existing CameraTrap repo to implement a "contact" section in PyTorchWildlife with contact information.
- To communicate with existing partners about the new communication channels.

3.2 Addressing requests:

There is a need to develop a pipeline to efficiently address user reports on the GitHub repository and through direct contact with the least amount of human effort.

Workload:

The estimated development time for this task is 120 hours of work for one person with experience in customer service and marketing.

3.2.1 Address issues from the GitHub repository.

3.3 Addressing suggestions:

There is a need to develop a system that can receive user suggestions to add them to the improvement pipeline.

Workload:

The estimated development time for this task is 40 hours of work for one person with experience in customer service and marketing.

3.3.1 Add a suggestion channel for new features

3.3.2 Add suggestions to PyTorchWildlife's timeline.

The following table shows an estimate timeline and work intensity for the development of the features.

PyTorchWildlife Microsoft | AI for Good Lab

Stage \ Month	Oct	Nov	Dec	Jan	Feb
Retention of existing partners 80 hours 1 person with marketing and customer service knowledge					
Addressing requests 120 hours 1 person with marketing and customer service knowledge					
Addressing suggestions 40 hours 1 person with marketing and customer service knowledge					

Fourth stage: Marketing

Estimated human effort: 1 person with experience in marketing and customer service for 240 hours.

Promoting the use of PyTorchWildlife is the last component to keep this package as a golden standard in animal detection. *This stage is in development and it is open to new thoughts and ideas on how to promote torch Wildlife.*

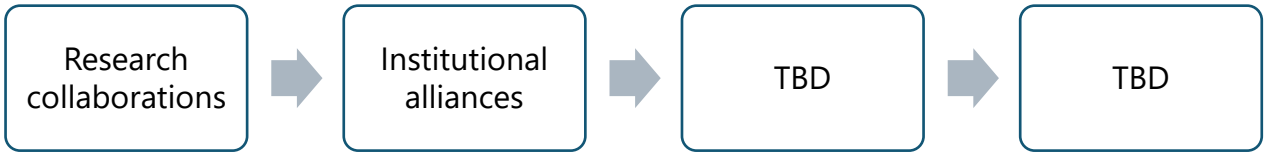


Figure 5. Steps for Megadetector's marketing stage.

4.1 Research collaborations:

Showcasing the work of PyTorchWildlife to partners and collaborators is essential to establishing new communication channels and to extend PyTorchWildlife's use.

Workload:

The estimated development time for this task is 120 hours of work for one person with experience in customer service and marketing.

- 4.1.1 A collaboration initiative with renowned researchers in the animal conservation and biodiversity field.
- 4.1.2 Reopen the AI conservation workshops and summer schools (maybe in collaboration with academic institutions like CalTech and CCAI, etc.) to bring more people into the area of AI Conservation and teach them how to use MegaDetector.

4.2 Institutional alliances:

One method that could help to ensure the use of PyTorchWildlife is to establish partnerships with renowned conservation institutions worldwide.

Workload:

The estimated development time for this task is 120 hours of work for one person with experience in customer service and marketing.

- 4.1.3 To develop a marketing campaign to establish partnerships with renowned conservation institutions worldwide (e.g., Humboldt Institute).

The following table shows an estimate timeline and work intensity for the development of the features.

PyTorchWildlife Microsoft | AI for Good Lab

Stage \ Month	Oct	Nov	Dec	Jan	Feb
Research Collaborations 120 hours					
Institutional Alliances 120 hours					

PyTorchWildlife

Microsoft | AI for Good Lab

Technical timeline

Microsoft | AI for Good Lab

TORCH WILDLIFE

Technical Timeline

PACKAGING

- 1.1.1 Add the YOLOv5 detection model.
- 1.1.3 Add a Notebook to create, load and run inference on the YOLOv5 detection model using pretrained weights.
- 1.1.1 Add the YOLOv5 pretrained weights.
- 1.2.1 Add a Wildlife dataset module to download LILA subsets.

0.1 **OCTOBER**

MARKETING

- 4.1.1 Define marketing campaigns to establish partnerships with renowned conservation institutions.

CUSTOMER SERVICE

- 3.1.1 Add a contact section in the TorchWildlife GitHub repository.

Microsoft | AI for Good Lab

TORCH WILDLIFE

Technical Timeline

PACKAGING

- 1.1.2 Add a module for fine-tuning and training from scratch.
- 1.2.2 Add a module for custom datasets.
- 1.2.1 Add visualization functions for LILA subsets.
- 1.2.3 Add a Notebook to fine-tune or train a model using a custom dataset.

0.2 **NOVEMBER**

MARKETING

- 4.1.1 Collaboration with Humboldt Institute for animal detection in the Amazon rainforest.
- 4.1.1 Start contact with other research institutions.

CUSTOMER SERVICE

- 3.1.1 Communicate with existing partners for the launch of TorchWildlife.

PyTorchWildlife

Microsoft | AI for Good Lab

Microsoft | AI for Good Lab

TORCH WILDLIFE

Technical Timeline

PACKAGING

- 1.3.1-1.3.2 Add an animal species classification model and pretrained weights.
- 1.4.2-1.4.3 Add a documented animal video detection pipeline.
- 1.3.3 Add a Jupyter notebook for the video detection

IMPROVEMENTS

- 2.1.1 Implement USGS' Florida Reptile dataset.
- 2.2.1 Add the Deformable DETR detection model.
- 2.2.2 Add the Swin Transformer classification model.

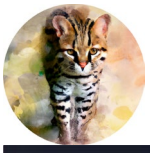
0.3 **DECEMBER**

MARKETING

- 4.1.1 Establish collaborations with other research institutions.
- 4.1.2 Reopen the AI conservation workshops and summer schools.

CUSTOMER SERVICE

- 3.1.1 Communicate with existing partners for the launch of TorchWildlife.
- 3.2.1 Address issues from the GitHub repository.



PyTorchWildlife

Microsoft | AI for Good Lab

Microsoft | AI for Good Lab

TORCH WILDLIFE

Technical Timeline

PACKAGING

- 1.4.1 Add a utilities module for data pre-processing and post-processing.
- 1.5.1 Add a user-friendly interface for animal detection.
- 1.5.2 Add the video detection pipeline to the user interface.

IMPROVEMENTS

- 2.1.2 Implement Australia's reptile dataset.
- 2.2.3 Implement DINOv2 for end-to-end video detection
- 2.2.4 Implement end-to-end video classification pipeline.

0.4

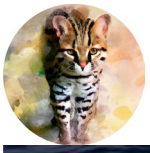
JANUARY

MARKETING

- 4.1.1 Establish collaborations with other research institutions.
- 4.1.2 Reopen the AI conservation workshops and summer schools.

CUSTOMER SERVICE

- 3.2.1 Address issues from the GitHub repository.
- 3.3.1 Add a suggestion channel for new features.



PyTorchWildlife

Microsoft | AI for Good Lab

Microsoft | AI for Good Lab

TORCH WILDLIFE

Technical Timeline

PACKAGING

- 1.5.3 Add animal classification to the user interface.
- 1.5.4 Complete documentation on how to use the user-interface.

IMPROVEMENTS

- 2.1.3 Implement a subaquatic species dataset.
- 2.2.5 Add a state-of-the-art segmentation and classification model developed at AI4G.
- 2.2.6 Add full documentation on the new models.

0.5

FEBRUARY

MARKETING

- 4.1.1 Establish collaborations with other research institutions.
- 4.1.2 Reopen the AI conservation workshops and summer schools.

CUSTOMER SERVICE

- 3.2.1 Address issues from the GitHub repository.
- 3.3.2 Add suggestions to the TorchWildlife timeline.

References

[1] Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." *Advances in neural information processing systems* 32 (2019).

[2] Marcel, Sébastien, and Yann Rodriguez. "Torchvision the machine-vision package of torch." *Proceedings of the 18th ACM international conference on Multimedia*. 2010.

[3] Stewart, Adam J., et al. "Torchgeo: deep learning with geospatial data." *Proceedings of the 30th international conference on advances in geographic information systems*. 2022.