Sequence Information Channel Concatenation for Improving Camera Trap Image Burst Classification

Bhuvan Malladihalli Shashidhara
University of Washington
msbhuvan@uw.edu

Darshan Mehta
University of Washington
darshanm@uw.edu

Yash Kale
University of Washington
yashkale@uw.edu

Dan Morris
Microsoft Research
dan@microsoft.com

Megan Hazen
University of Washington
mh75@uw.edu

Abstract

Camera Traps are extensively used to observe wildlife in their natural habitat without disturbing the ecosystem. This could help in the early detection of natural or human threats to animals, and help towards ecological conservation. Currently, a massive number of such camera traps have been deployed at various ecological conservation areas around the world, collecting data for decades, thereby requiring automation to detect images containing animals. Existing systems perform classification to detect if images contain animals by considering a single image. However, due to challenging scenes with animals camouflaged in their natural habitat, it sometimes becomes difficult to identify the presence of animals from merely a single image. We hypothesize that a short burst of images instead of a single image, assuming that the animal moves, makes it much easier for a human as well as a machine to detect the presence of animals. In this work, we explore a variety of approaches, and measure the impact of using short image sequences (burst of 3 images) on improving the camera trap image classification. We show that concatenating masks containing sequence information and the images from the 3-image-burst across channels, improves the ROC AUC by 20% on a test-set from unseen camera-sites, as compared to an equivalent model that learns from a single image.

1. Motivation

Camera Traps are extensively used to observe wildlife in their natural habitat. A camera trap is a remotely activated camera that is equipped with a motion sensor or an infrared sensor, or uses a light beam as a trigger. Camera trapping is a method to capture wild animals on film without disturbing the ecosystem, and has been used in ecological research for decades. This could help in the early detection of natural or human threats to animals, and help towards ecological conservation.

Currently, a massive number of such camera traps have been deployed at various ecological conservation areas around the world, collecting data for decades. Wildlife researchers are interested in performing ecological studies like population estimation of different animal species across different seasons of a particular ecological conservation area. But, due to the vast size of data, it is often very difficult for researchers to go through all the images taken by the camera traps manually to perform such studies. Automating the process of labeling camera trap images has had a high impact on enabling wildlife research at such scales and as a result, it has been an active area of research in recent times.

The AI for Earth group at Microsoft has developed the MegaDetector, and the Microsoft AI for Earth Species Classification software, which uses deep learning to identify the animal species present in a given image. These models are trained on various open-source camera trap images and have achieved great results. These systems, however, perform classification on a single image. Due to challenging scenes with animals camouflaged in their natural habitat and due to dark scenes from night, it is very difficult to identify and classify animals from merely a single image. Moreover, a large number of instances in camera-trap data have such scenes. For instance, in many cases, it is very hard to spot small animals like birds using a still image, where it looks well-camouflaged within a thick forest scene, and we can only spot it when we observe its moving tail from a burst of images. This problem gets particularly challenging during night time, for instance - if a rabbit is found at night, we only see two bright dots (rabbit’s eyes) in a single image, but over a short sequence (burst) of images, we can see these bright dots move in a pattern which
enables us to identify it as a rabbit.

We hypothesize that on utilizing the sequence information present in a short burst of images, assuming that the animal moves, it becomes much easier for a human as well as a machine to detect the presence of an animal. Hence, in this research project, we explore and measure the impact of using image sequences on improving the camera trap image classification.

This paper initially provides a brief review of prior literature, then establishes the problem statement along with the dataset description. This is followed by a detailed description of the proposed methodology, and the discussion of results.

### 2. Literature Review

Wei et al. [18] built a tool named 'Zilong' that uses an image processing based approach to identify empty images in a camera trap dataset. They use pixel-level differences between frames to measure color change and the Sobel algorithm to identify edges. They have a slightly separate pipeline for foggy vs. non-foggy images. While this approach is extremely fast due to the simplicity of the computational steps, it performed poorly in scenarios where empty images contained swinging vegetation since these are usually associated with color intensity and edge change. Unlike Wei et al. [18], Yousif et al. [22] built a software tool named 'Animal Scanner' which uses both computer vision algorithms and deep learning to detect and classify moving objects. Similar to us, they mainly focus on aiding the removal of empty frames in camera trap images. Their predictions fall into three categories: animal, human, and background. They use the information available in the sequences for background subtraction thereby identifying regions that could potentially contain a moving object. This helps in removing focus from the heavy vegetation present in most scenes in the wild. They then reduce the false positive foreground proposals caused by the varying intensity levels within the same sequence. Finally, they use a Deep Convolutional Neural Network with a configurable input size and convolutional layers to classify the proposed regions.

Jhony Heriberto et al. [6] proposed a multi-layer robust principal component analysis (multi-layer RPCA) approach for background subtraction. Their method computes sparse and low-rank images from a weighted sum of descriptors, using color and texture features as a case of study for camera-trap images segmentation. The segmentation algorithm is composed of histogram equalization or Gaussian filtering as pre-processing, and morphological filters with active contour as post-processing. They optimize the parameters of their multi-layer RPCA using an exhaustive search.

Marco Willi et al. [21] used Convolutional neural networks (CNNs) in their work to differentiate among images of different animal species, images of humans or vehicles, and empty images. They worked with a combination of many famous datasets and showed that with transfer learning, the performance improvement in classification is quite significant in this domain. Our interesting approach to use transfer learning to initialize non-standard networks (which vary in the size of the input layer) was motivated by their gain in performance.

To exploit the sequence information present in the bursts, Jeff Donahue et al. [4] used an LSTM layer by wrapping each image in a sequence in separate convolutional layers. Each image in the sequence would be wrapped inside a TimeDistributed layer, where each image would be treated as a single timestamp. Once the features were generated for each image inside a sequence, an LSTM layer would then be added after this to see if any information can be extracted from the features of the images. Jeff Donahue et al. [4] used a new method such that instead of taking the output of only the last cell from the LSTM layer, they took the output of each cell and average the predictions to get one final prediction. In this work, we explore this approach on our dataset.

In this section, we discussed several prior research studies that tried to utilize sequence information for image classification. In this research, we explore some of these methods along with a simple channel-concatenation approach on a considerably large camera-trap dataset, to determine if such sequence information is beneficial for camera-trap image classification.

### 3. Problem Statement

In this project, we focus on a binary classification task to predict whether a sequence of images (burst of 3 images), contains an animal or not. In order to determine if the sequence information helps, we consider the following two categories of models:

- **Baseline Models:** Single-image-based models that learn from a single-image input (single image from the burst of images), and does not utilize any sequence information. These models represent the currently adopted strategy in many systems.

- **Sequence Models:** Models that utilize sequence information from the burst or sequence of images in some way during the training and inference processes.

Our main aim is to determine if the addition of sequence-information that is present in the image sequences help in improving the image classification process. In other words, we aim to test if the Sequence Models perform better than the Baseline models on the test set. We use Area under ROC [20] as our primary evaluation metric for measuring the performance of the image classification models.
Further, we consider 2 scenarios for evaluation, based on the way we split the data for training and testing - uniform split and camera-site based split. Firstly, for the uniform split scenario, we consider a scenario where we assume that data from every camera-site is required during the training time, for which we split the data uniformly across all image sequences to form train, validation and test sets. However, this assumption would imply that we are required to retrain for every new camera-site installation by labeling a few image-sequences from the new camera site. In order to overcome this limitation, we consider the second scenario of camera-site based split, where we split the data based on the camera-sites, to form train, validation and test sets containing image sequences from mutually exclusive camera-sites.

3.1. Dataset Description

The Wellington dataset [1] comprises of 90150 image sequences (burst of images) collected from camera traps across 187 camera-sites in Wellington, New Zealand, and is open for public non-commercial research purposes. Each image sequence consists of 3 images leading to a total of 270450 images in the dataset consuming 190 GB disk space. The data contains image sequences (bursts) that are captured in daylight and at night. In most cases, it is very hard to visually spot an animal by just seeing a single image because the animal is either very small or is camouflaged against the background, but we can spot the animal by observing the burst of images. Few sample sequences from the dataset are shown in Figure 1.

Each of the sequences has been hand-labeled into one of the 17 classes: 15 classes corresponding to various animal species, the empty class for sequences that contain no animals, and the unclassifiable. However, for this study, we binarize the labels to whether the sequence contains an animal or not. This dataset is listed on Lila BC [2] which internally uses Microsoft Azure to host the data. All the image labels are maintained in COCO Camera Traps Format [11]. Labels in the dataset are given at a sequence level i.e., each image within a sequence has the same label. In other words, a burst or sequence of 3 images would be treated as a single data point and would have a corresponding label. Data processing steps and further details regarding the data is available in Section 4.

4. Methodology

This section describes the data pre-processing steps, the training and inference pipeline software framework implementation details, followed by baseline model approaches and sequence-model approaches.

4.1. Data Processing

The Wellington dataset [1] is available as two zip files, one containing all the images and the other containing the metadata. The metadata is a CSV file that specifies which image belongs to which sequence, and the label associated with it amongst other details.

During our initial exploration, we found that not all sequences had 3 images. We had to discard such sequences for the purpose of maintaining uniformity. We also excluded the sequences which were ‘Unclassifiable’ since we could not say what the true labels were. We binarized the label to whether an animal is present or not. Finally, we created a new tabular dataset by pivoting the existing one, so that each new row represents a sequence, and contains references to the images that belong to this sequence and the label associated with them. We opted for this structure in order to simplify the ingestion of data in the pipeline downstream. All images were resized to 224x224 in order to work with a manageable size for running the computations in a reasonable amount of time.

We create two sets of train, validation and test sets for the two scenarios described in the Section 3 - uniform split and camera-site based split. In both scenarios, we up-sampled the empty-sequences (sequences not containing any animal) to balance the class distribution in the training set, but we perform down-sampling (random removal) of sequences with animals in order to balance the validation and test sets. Firstly, for the uniform split scenario, the dataset into train, validation and test set in the ratio 0.70, 0.15 and 0.15 respectively, by uniform random sampling, resulting in 62165 sequences for training, out of which 51426 sequences contained an animal while 10738 sequences were empty. After balancing the class distribution as mentioned above, the validation and test sets contained 4279 and 4601 sequences respectively. Secondly, for the camera-site based split scenario, we considered 111, 36 and 35 camera-sites for training, validation and test sets respectively, by random selection. After balancing the class distribution as mentioned above, the validation and test sets contained 5969 and 6453 sequences respectively. Note that all the data splits in both scenarios have a balanced class distribution.

We built a highly-configurable data pipeline using the Dataset API provided by TensorFlow [16] that would work with sequences of any sizes. This pipeline handles loading, processing, batching and pre-fetching of images from the disk in order to speed up the training process. The pipeline performs random augmentations such as flipping, rotating, color augmentation, etc. on the sequence level. Since the input data is a sequence of three images, the output required from the pipeline varies for each type of model that we plan to work with. Some require only one of the images from the sequence, some require all three as separate data points and some require a single image with multiple channels. The
pipeline allows for all this flexibility by setting the 'mode' configuration while creating the object.

We faced a few computational challenges with respect to pre-processing and disk bottleneck. Working with gradient descent based optimizers entails making multiple passes over the dataset. Since our dataset is 190 GB on the disk, reading, holding in memory and processing this data takes up a lot of memory and time. In order to tackle this limitation, we need to reduce on-the-fly operations as much as we can. We do this by first performing the resizing operation before the training process begins. We pick the size of the image based on the input layer requirement of the pre-trained network weights that we plan to use. Then, we batch the dataset and perform random augmentations and other processing on the fly. Batching helps reduce the memory load on the GPU. If a mode requires a single image from a sequence, we avoid reading the other images of the sequences in order to speed up the training process. As explained in the previous process, reading, augmenting and generating data points is an expensive operation. The actual gradient calculation is, however, relatively faster making the loading operation a bottleneck. We tackle this issue by pre-fetching multiple subsequent in parallel batches and keeping them in memory for the GPU to use when it finishes processing the current batch.

4.2. Model Training and Inference Framework Implementation

We use Azure Blob storage and Azure computes to store and run our models respectively. In order to quickly load the data onto different Virtual Machines (VMs), we have stored the data on an Azure Disk, which we can easily mount to different VMs. We would be using TensorFlow 2.0 for implementing our data pipelines and model architectures due to its popularity and well-documented user guides. All of our code is in Python 3, can be found on our GitHub repository.

We designed and implemented a framework based on TensorFlow 2.0 which has an independent data pipeline, model architectures, training pipeline and inference (evaluation) pipeline. This allows us to work independently on different models, and easily integrate and test our solutions.

4.3. Baseline Model

As introduced in Section 3, we consider single-image based models (which do not utilize any form of sequence information) as baseline models. During training time, all images from a sequence are considered as independent data-points having the same label as the one associated with the sequence, and therefore, all images from all training sequences are used independently to train the baseline models. However, during inference time, only the first-image of each sequence in the test-set is used to obtain the predicted labels, in order to maximize the chance of an animal being present in the frame since this image is captured as soon as the sensor on the camera gets triggered.

We experimented with various standard CNN architectures like VGG-16 [14], InceptionResNet-V2 [15], and variations of ResNet [7] like ResNet-50, ResNet-101, ResNet-152, and ResNet-152-V2 [8]. These architectures were chosen as they have been proven to perform well on ImageNet challenge. We initialized the network with pre-trained weights from ImageNet prior to training to achieve faster and better convergence.
4.4. Proposed Sequence Models

As described in Section 4.4.1, ResNet-152 architecture performs well as a baseline (single-image classifier) compared to other network architectures. Therefore, ResNet-152 architecture is used as the backbone feature-extractor (base CNN-architecture) for the sequence models. We propose Sequence Information Channel Concatenation approach described in Section 4.4.1 and compare it to the Recurrent Neural Architecture approach described in Section 4.4.2. The Figure 2 shows a pictorial overview of the sequence modeling approaches.

4.4.1 Sequence Information Channel Concatenation

The sequence information is extracted as a mask from the three images of the burst using either of the following two methods. The below methods were considered due to their proven effectiveness in motion analysis research. The resulting masks are concatenated to the first-image or all three images of the sequence across channels, and is considered as the input-layer for the ResNet-152 model.

- **MOG2 Mask**: This is an OpenCV implementation of a Gaussian Mixture-based Background/Foreground Segmentation Algorithm. It is based on two papers by Z.Zivkovic. One important feature of this algorithm is that it selects the appropriate number of Gaussian distributions for each pixel and hence, it provides better adaptability to varying scenes due to illumination changes etc. We create a background subtractor object by passing all the frames of the sequence to the function call. This then returns us a grayscale mask that differentiates between the background, foreground and the shadows.

- **Optical Flow RGB Mask**: Dense Optical Flow using Gunnar Faurnebbs algorithm results in an RGB mask (colors represent direction of motion). This method takes 2 images in a sequence and produces a mask. Since we have 3 images in our sequence, we get two Optical Flow masks - Optical Flow RGB Mask-1-2 computed between the first and the second image, and Optical Flow RGB Mask-2-3 computed between the second and the third image. However, for most of our experiments, we take a simple average of these two masks which is referred in the rest of the paper as Optical Flow RGB Mask.

The comparison of the masks produced by MOG2 and Dense Optical Flow for a few sequences has been shown in Figure 3, which provides insight on how the sequence information is being highlighted.

We experiment with a variety of channel-wise concatenation possibilities, by modifying the first (input) layer of the model. Note that we initialize weights for the channels corresponding to the original image (the first image or all three images from the sequence), and the channels corresponding to the optical-flow output (RGB channels) with the ImageNet pre-trained weights, but use a random-normal initialization for the MOG2 grayscale mask. Based on this approach, we experimented with the following models having variations in the input layer. Image1, Image2 and Image3 refer to the first, second and third images of the sequence respectively. Figure 4 shows this approach pictorially for the Hybrid-13-Channel model described below.

- **MOG2-4-Channel**: Image1 (3) stacked with MOG2 grayscale mask (1) across channels to form 4-Channel input layer.

- **MOG2-10-Channel**: Image1 (3), Image2 (3) and Image2 (3) stacked across channels along with MOG2 grayscale mask (1) to form 10-Channel input layer.

- **OpticalFlow-6-Channel**: Image1 (3) stacked with Optical Flow RGB mask (3) across channels to form 6-Channel input layer.

- **OpticalFlow-15-Channel**: Image1 (3), Image2 (3) and Image2 (3) stacked across channels along with Optical Flow RGB mask-1-2 (3) and Optical Flow RGB mask-2-3 (3) to form 15-Channel input layer.

- **Hybrid-13-Channel**: Image1 (3), Image2 (3) and Image2 (3) stacked across channels along with Optical Flow RGB mask (3) and MOG2 grayscale mask (1) to form 13-Channel input layer.
Figure 3. Comparison of MOG2 grayscale mask and Optical Flow RGB mask. The figure shows the first image in the sequence, MOG2 mask and Optical Flow RGB mask for the four sequences shown in Figure[1]. The sequences shown on left-top (bird found in full daylight), left-bottom (bird found in low daylight) and right-top (hedgehog found at night time), contain a small animal, whose slight directional movement is represented in MOG2 mask (direction of motion is not captured) and Optical Flow RGB mask (the direction of motion gets represented by color). The sequence on the right-bottom is empty (does not contain any animal), but the slight arbitrary movement of leaves due to wind causes random noises in MOG2 mask and Optical Flow RGB mask.

Figure 4. Concatenated input layer along with the backbone CNN architecture (ResNet-152) shown for the Hybrid-13-Channel model. The other sequence information channel concatenation models described in Section 4.4.1 use a similar approach.

- **OpticalFlow-Only (6-Channel)**: Optical Flow RGB mask-1-2 (3) and Optical Flow RGB mask-2-3 (3) stacked across channels to form 6-Channel input layer.

- **OpticalFlow-MOG2-only (7-Channel)**: Optical Flow RGB mask-1-2 (3), Optical Flow RGB mask-2-3 (3) and MOG2 grayscale mask (1) stacked across channels to form 7-Channel input layer.

### 4.4.2 Recurrent Neural Architecture

LSTM [9] network architecture is proven to work well in learning sequence information, especially in the Natural Language Processing domain. Therefore, we considered this method of learning the sequence information present in our short-sequences of images, for a comparative study. This model is referred to as 'LSTM' model in the rest of the paper.

In this model, each frame is considered as a time step for the LSTM layer. A TimeDistributed wrapper in TensorFlow [12] is used over each image in the sequence, such that the base CNN-architecture is used for each image in the sequence. Thus, each image in the sequence passes through the base-CNN to extract features. The features of all the images in the sequence are then considered as timestamps and passed onto the LSTM layer. Note that for LSTM model, the backbone architecture used was ResNet-50 instead of ResNet-152.

All models were trained up to 10 epochs (as we have a large training set), with a learning rate of $10^{-4}$ with Adam optimizer. Checkpoints were taken at each quarter of an epoch, and the inference was run on the validation set to determine the validation ROC AUC at that checkpoint. The models were early stopped by monitoring validation ROC AUC at each checkpoint with a patience of 3 checkpoints. The batch-size was based on GPU capacity of 12 GB (Tesla K80), and was set to 32 or 16 based on the model (we had to reduce the batch-size for models inputting images with multiple channels to accommodate it within the GPU memory). The best model was saved at each checkpoint when the model resulted in an improved validation ROC AUC. Under this setting, the Baseline model took about 13 hours, and the sequence models took about 6 hours to converge.

Each of the above configurations is evaluated on the held-out test-sets of both the uniform split scenario and the camera-site based split scenario. The ROC AUC on the test-sets is compared across the baseline model and the different types of sequence-models, in order to determine if the sequence information can effectively be used to improve the classification performance.
5. Results and Discussion

In this section, we describe the selection of the best performing baseline-model, which would henceforth be used as a backbone CNN architecture for the sequence models described in Section 4.4. Further, we compare the performance of different sequence-models to the baseline model, followed by the discussion to determine whether the sequence-information is helpful to our problem.

The comparison of the performance of the baseline-model architectures on the test-set of uniform split scenario has been summarized in Table 1. We observe that the ResNet-50 and ResNet-152 architecture provide the highest ROC AUC of 0.85. Therefore, we consider ResNet-152 as the backbone CNN architecture for sequence-models, as it has more parameters than ResNet-50 which may be helpful for sequence-models. This also implies that ResNet-152 architecture is a good feature extractor from single-images. If any of the sequence models can achieve better performance on the test-set on both the scenarios, then it would imply that sequence information is useful for our problem.

The Table 2 shows the performance of all the sequence-models compared to the baseline model on the held-out test-sets of both the uniform split scenario and the camera-site based split scenario. We observe a drop in the performance on all the models in the camera-site based split scenario as compared to the uniform split scenario, which can be expected due to the presence of new scenes in the test set from unseen camera-sites. However, in both the scenarios, we observe that the channel concatenated sequence models (described in Section 4.4.1) significantly outperform the corresponding Baseline models, as Hybrid-13-Channel model performs 11% and 20% better than the Baseline on the test sets of uniform split scenario and camera-site based split scenario, respectively. Note that the channel concatenated sequence models share identical feature extraction layers with Baseline models i.e., the same backbone ResNet-152 architecture, but still, they result in significantly better performance. This implies that the sequence information captured in Optical Flow and MOG2 masks are effectively utilized by concatenating across the channels in the input-layer, to achieve a significantly better classification of 3-image-sequence instances (bursts of 3-images) of the camera-trap data.

Qualitatively, as observed in Figure 3, we observe that the Optical Flow RGB mask is much better than the MOG2 mask as it contains the information about the direction of motion in the sequence. However, quantitatively from Table 2, we do not observe a considerable difference in the test ROC AUC (in both scenarios) between the MOG2 mask based sequence models (MOG2-4-Channel, MOG2-10-Channel) and Optical Flow RGB mask based sequence models (OpticalFlow-6-Channel, OpticalFlow-15-Channel) when concatenated across channels with the first image (or all images) from the sequence. Further, we observe that the OpticalFlow-Only (6-Channel) and OpticalFlow-MOG2-Only (7-Channel) models perform better than the corresponding Baseline models in both scenarios, though they do not utilize any original image from the sequence, and their input layers contain just the Optical Flow and MOG2 masks. Also, we observe that OpticalFlow RGB mask and MOG2 mask complement each other to provide better performance when both are concatenated together across the channels in the input layer, as seen in the Hybrid-13-Channel model and the OpticalFlow-MOG2-Only (7-Channel) model. The LSTM models provide a small improvement over the Baseline models in both scenarios, and they are less effective compared to the channel-concatenated sequence models. This could imply that the LSTM models are less effective for short sequences, however, they may be more effective for sequences of longer length like the ones that are usually found in the Natural Language Processing domain.

Practically, the camera-site based split scenario is more suited for deployment as we can directly utilize the models for unseen camera-sites. Moreover, we observed a significant improvement of about 20% in test ROC AUC compared to its Baseline model in this scenario, indicating that sequence-information through channel-concatenation models is beneficial for image classification from unseen camera-site deployments. Though OpticalFlow-MOG2-only model provided the highest test ROC AUC of 0.93 in this scenario, we prefer to utilize the Hybrid-13-Channel model which provided an ROC AUC of 0.92, because this is a very small performance difference, and it has a higher potential as it uses an original image from the sequence as well.

Overall, we think that the channel concatenation configuration in which both the OpticalFlow and MOG2 masks are concatenated with an original image from the sequence,
Table 2. Comparison of all the model performances on the held-out test-sets of both the **uniform split** scenario and the **camera-site based split** scenario. We find that sequence-models perform significantly better over the Baseline model in both scenarios. We observe that the **Hybrid-13-Channel** model results in high performance in both scenarios. Also, the **OpticalFlow-Only** and the **OpticalFlow-MOG2-only** models (which do not have any original image from the sequence), outperform the Baseline model, which implies that the OpticalFlow and MOG2 masks contain helpful sequence information.

| Model Name                  | Model Description                                                                 | Test ROC AUC Uniform Split | Test ROC AUC Camera-site based Split |
|-----------------------------|-----------------------------------------------------------------------------------|-----------------------------|-------------------------------------|
| Baseline                    | Single-image based ResNet-152                                                      | 0.85                        | 0.72                                |
| LSTM                        | LSTM cells over Baseline feature maps                                              | 0.88                        | 0.76                                |
| MOG2-4-Channel              | Image1 + MOG2 mask                                                                | 0.93                        | 0.90                                |
| MOG2-10-Channel             | Image1 + Image2 + Image3 + MOG2 mask                                               | 0.95                        | 0.92                                |
| OpticalFlow-6-Channel       | Image1 + Optical Flow RGB mask                                                    | 0.95                        | 0.92                                |
| OpticalFlow-15-Channel      | Image1 + Image2 + Image3 + Optical Flow RGB mask-1-2 + Optical Flow RGB mask-2-3  | 0.95                        | 0.91                                |
| Hybrid-13-Channel           | Image1 + Image2 + Image3 + Optical Flow RGB mask + MOG2 mask                      | 0.96                        | 0.92                                |
| OpticalFlow-Only (6-Channel)| Optical Flow RGB mask-1-2 + Optical Flow RGB mask-2-3                             | 0.90                        | 0.92                                |
| OpticalFlow-MOG2-only (7-Channel)| Optical Flow RGB mask-1-2 + Optical Flow RGB mask-2-3 + MOG2 mask               | 0.91                        | 0.93                                |

like in the **Hybrid-13-Channel** model is best suited for our problem, as it uses different forms of information (original image along with Optical Flow and MOG2 masks), and because it provides high ROC AUC on the test sets of both scenarios, and especially in the **uniform split** scenario where it provides best ROC AUC of 0.96.

### 6. Conclusion and Future Work

We observed that sequence-information concatenated across channels with the input image provides a significant improvement of about 11% and 20% in the test ROC AUC over baseline single-image model with the same base CNN architecture, in the **uniform split** scenario and the **camera-site based split** scenario, respectively. Both Optical Flow RGB mask and MOG2 mask seem to be almost equally effective, but the Optical Flow RGB mask is qualitatively slightly better than the MOG2 mask as it can indicate the direction of motion as well. The **Hybrid-13-Channel** model listed in Table 2 can be considered as the best sequence-model as it provides the high test ROC AUC in both scenarios, and it uses different forms of information (original image along with Optical Flow and MOG2 masks). As our primary contribution in this paper, we demonstrate that the sequence information present in 3-image-bursts can be exploited with a simple channel-concatenation approach to significantly improve the classification of camera-trap images data, which was hypothesized based on observing that the animals may be hard to spot in individual images, but are visible when observed in a short-burst of images. Therefore, as a takeaway, we recommend future camera trap deployments to consider collecting bursts of images instead of single images.

Future research can expand this work to test whether similar improvement can be observed in a multi-class species classification setting. Future work can also consider determining the optimal length of the sequence (number of images) in the burst, by experimenting with sequences of different length, which might be helpful for appropriate camera trap deployments. Further, this work could be expanded to determine if this approach of concatenating sequence-information channels helps in a detector-based setting where models utilize strong-labels in terms of bounding box or segmentation information.

### 7. Acknowledgment

We sincerely thank the Microsoft Azure grants program in association with Microsoft AI for Earth, for supporting this research project by providing us sufficient cloud credits to perform our experimentation. Also, we would like to thank the Master of Science in Data Science program at the University of Washington for supporting us through the capstone project.
References

[1] V. Anton, S. Hartley, A. Geldenhuis, and H. U. Wittmer. Monitoring the mammalian fauna of urban areas using remote cameras and citizen science. *Journal of Urban Ecology*, 4(1):juy002, 2018.

[2] L. BC. Labeled information library of alexandria: Biology and conservation, 2019.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[4] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[5] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.

[6] J.-H. Giraldo-Zuluaga, A. Salazar, A. Gomez, and A. Diaz-Pulido. Camera-trap images segmentation using multi-layer robust principal component analysis. *The Visual Computer*, 35(3):335–347, 2019.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Microsoft. Camera traps, 2019.

[11] Microsoft. Coco camera traps format, 2019.

[12] Microsoft. Microsoft ai for earth, 2019.

[13] Microsoft. Species classification, 2019.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[15] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[16] TensorFlow. A collection of datasets ready to use with tensorflow., 2019.

[17] TensorFlow. Tensorflow keras layers timedistributed., 2020.

[18] W. Wei, G. Luo, J. Ran, and J. Li. Zilong: A tool to identify empty images in camera-trap data. *Ecological Informatics*, 55:101021, 2020.

[19] Wikipedia contributors. Camera trap, 2019. [Online; accessed 20-Nov-2019].

[20] Wikipedia contributors. Receiver operating characteristic, 2019. [Online; accessed 12-Dec-2019].

[21] M. Willi, R. T. Pitman, A. W. Cardoso, C. Locke, A. Swanson, A. Boyer, M. Veldhuis, and L. Fortson. Identifying animal species in camera trap images using deep learning and citizen science. *Methods in Ecology and Evolution*, 10(1):80–91, 2019.

[22] H. Yousif, J. Yuan, R. Kays, and Z. He. Animal scanner: Software for classifying humans, animals, and empty frames in camera trap images. *Ecology and Evolution*, 9(4):1578–1589, 2019.

[23] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition*, 2004. ICPR 2004., volume 2, pages 28–31. IEEE, 2004.

[24] Z. Zivkovic and F. Van Der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780, 2006.