

## Introduction

Catch & Release (C+R) is a collection of procedures that allow one to apply machine learning classification onto field videos. C+R's goal is to facilitate the creation of under-represented knowledge in machine learning in general, and experimental datasets for neural network image classification in particular. C+R allows anyone with a mobile phone to create viable datasets for image classification and to train state of the art convolutional neural networks with these datasets.

Furthermore, C+R can extract text from video. It can extract labels from video and use them as labels to generate image categories associated with the utterance.

This software and the bali-26 dataset are the basis for the 'Return to Bali' project that explores machine learning to support the representation of ethnobotanical knowledge and practices in Central Bali.

[http://www.realtechsupport.org/new\\_works/return2bali.html](http://www.realtechsupport.org/new_works/return2bali.html)

C+R can help create machine learning datasets or can serve as a companion while studying convolutional neural network systems. If you want to use it for specific research needs, for instructional purposes or if you would like to contribute toward building a viable alternative to ImageNet for machine learning training, contact the repository owner (marcbohlen@protonmail.com).

## License

Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

Cite this software project as follows: 'Catch&Release version1'

## Platform Information

C+R runs on Linux under Python3 and Flask with Chromium or Firefox. C+R uses the PyTorch framework to train and test image classifiers and connects to the Google Speech API (free) for speech processing. Library versions and dependencies are given in the requirements file.

C+R has been tested on a desktop (i7-4770 CPU with 16GB of memory) and a laptop (i7-3667 CPU with 8GB of memory) with Ubuntu 18.04 TLS under kernels 5.2.8 and 5.3.0 with images sourced from .mp4 and .webm video (HD [1920 x 1080] at 30f/s; .mp4 H.264 encoded) from multiple (android OS) mobile phones and GoPro Hero 6 action cameras.

Recommended browser: Chromium.

Install the free Classic Cache Killer:

<https://chrome.google.com/webstore/detail/classic-cache-killer/kkmknnnjiniefekpicbaaobdnjjikfp?hl=en>

## Software Installation

Clone the C+R repository on GitHub

Open a terminal window and type:

```
git clone https://github.com/realtechsupport/c-plus-r.git
Username for 'https://github.com': realtechsupport
Password for 'https://realtechsupport@github.com':
<enter the access token>
```

Cd to the c-plus-r directory and run the following commands to update your basic python environment:

```
chmod +x basics.sh
sudo sh basics.sh
```

(This script just updates your ubuntu installation and requires sudo to do so.)

Create a virtual environment:

```
python3 -m venv env
```

Activate the environment:

```
source ./env/bin/activate
```

Cd to to the c-plus-r directory again. Install Requirements and Dependencies.

(This may take about 30 minutes.)

```
pip3 install -r requirements.txt
```

### Generate an STT key (optional)

While there are multiple providers of Speech to Text services, the most effective offering with the widest range of languages is at this moment provided by Google. If you want to make use of the text from video extraction you should obtain an access key to the Google Speech API. Creation of this key is free of charge and you can use it in this software at no cost as C+R operates within free limits of the API. However, you do require a google account in order to create the key. If that is not palatable, skip the sections that make use of the Speech API.

Instructions to generate a key:

1. Navigate to the **APIs & Services**->Credentials panel in Cloud Platform Console.
2. Select **Create** credentials, then select **API key** from the dropdown menu.
3. Click the **Create** button. ...
4. Once you have the **API key**, download it and **create a JSON** file.
5. Save to the C+R project

Other Speech to Text engines are available, but are not yet integrated into C+R. The key barrier at this point is limited availability of high quality language corpora in less popular languages. This is a major concern beyond C+R.

### Launch C+R

Activate the virtual environment: `source ./env/bin/activate`

Start C+R (in the c-plus-r directory): `python3 main.py chrome no-debug`

To run in debug mode replace 'no-debug' with 'debug'. Firefox is also supported, but less stable.  
The terminal window will display comments. You should see the launch screen in a browser window:  
*Catch & Release*

ctrl + / ctrl - increase / decrease zoom factor.

## Stop C+R

Stop the app from the terminal:

ctrl-c

Exit environment at the terminal:

ctrl-d

If you see browser errors .. clear the browsing history:

ctrl-H

clear browsing data

clear data

## Description of modules in C+R

### Overview

There are three related paths within C+R. Option (A) leads from field videos through video preparation, annotation / labeling to a collection of labeled images for a classifier. Option (A) is a general approach and applicable to all field videos.

Options (B) and (C) are specific to the ‘Return to Bali’ project as they work with a specific dataset.

Option (B) will train a low level neural net on several test datasets and graphically display the results based on the chosen inputs. Option (C) will test the ability of several deep neural networks trained on the bali-26 dataset to recognize images from the validation set of said collection.

When you run C+R for the first time, video samples, image samples and the trained models will be downloaded from pCloud automatically. You need these data files to run the examples.

```
A > prepare videos      > capture text
                        > video annotation
                          > label images from video    > check results
                              > bulk                      > remove outliers
                              > by audio label          > add labeled images to collection
```

```
B > train a classifier
    > train a low level neural network and display training performance and top-1 errors
```

```
C > test classifier
    > display images and choose input
    > classify input; show best guess with confidence level and top 3 choices
```

The next sections describe the individual modules.

## Prepare Field Videos

This module allows one to chunk a long field video into smaller parts for processing. Supported formats are .mp4 and .webm. Select the video and choose the chunk size. It is suggested that segments do not exceed 3 minutes.

Chunked segments are saved to the C+R /tmp directory, and are deleted if you go back to start or exit the program. You can also download some sample videos to process and experiment with.

When you have some data ready, continue either onto the text extraction or the video voice-over modules.

## Capture Text from Field Video

Use this module to extract text from a field video.

First load a field video to locate a section you want to extract text from, then reload to capture the text. This module requires an access code for the Google Speech API (key for capture text).

If you add a search term, text from video sections that contain that term will be listed separately. If the video is shorter than 1 minute, reduce chunk length.

Reducing the confidence level to below 0.9 increases detection chances and false positives. This module may take several minutes to complete. It is best to limit the difference between start and end times to a few minutes.

## Add Voice-over to Field Video

Use this module to replace the field audio with a new audio file. This can be helpful in the event an expert wishes to annotate a field video.

Select your field video and choose the microphone input.

The default mic is cardn = 0, devicen = 0. This is typically the default setting on a Linux box.

You can add your favorite external mic to the list of available microphones. For example, you may insert a (USB) microphone. After the USB microphone is connected, enter at the terminal the following command:

```
arecord -l -list-devices
```

You will see something like this (instead of 'USB Audio' you will see your microphone name):

```
card 3: Audio [Kmic USB], device 0: USB Audio [USB Audio]
```

Add this info (inside of the [ ] ) into the AnotateInputs class of the 'inputs.py' file (line 54ff) that contains the recording parameters.

Before:

```
mic = SelectField(label='select microphone', choices = [('0,0', 'default'),('AT2020', 'AT2020'),('UC02', 'UC02')],  
default=('default'), validators=[validators.InputRequired()])
```

After:

```
mic = SelectField(label='select microphone', choices = [('0,0', 'default'),('AT2020', 'AT2020'),('UC02', 'UC02'),  
('Kmic', 'Kmic')], default=('default'), validators=[validators.InputRequired()])
```

Save the file (ctr s) and refresh the audioannotate.html page. You should be able to select the USB mic you just added.

Then proceed as follows. Load the video (from samples, tmp or annoate folders) to identify the spot on which you want to add voice-over. Set start and end times to define the interval in which you will add a voice-over. Make sure the end time is greater than the start time and less than the length of the video.

Then segment the video. This can take a few minutes as the .mp4 file will be re-encoded. The terminal window shows the progress.

When the segmentation is complete, the silent segmented video will auto-play. Use the video controls to stop the video and rewind to the start (move the dot in the display timeline to the left).

Turn off your speakers.

Then click voice-over and comment on the video with appropriate key terms.

When you finished the voice-over (it can only be as long as the segmented video), go back to the start page. You can process this voice-over video (stored in the annotate folder) to labeled images with the next module.

## **Label images from Field Video**

Use this module to create labels from field videos or from your voice-over additions.  
Load the video to check, just in case.

There are two options:

A - Label images with key terms from audio track (label by audio).

In this case you will select a single key term and the number of images per utterance. Then set a confidence level for the speech to text API and the language spoken in the video. Select your key file to access the Speech API. If you want to search for multiple key terms, repeat the process above with a new term.

B - Label all images in the video with a given term (bulk label).

In this case, all images generated from a chosen video will be bulk labeled with a given category / folder name. Set the frame rate (number of images to be extracted per second).

When the process has completed, click 'check the results' to open the subsequent module.

## **Quality Control, Archiving, Sharing**

This module allows you to control the quality of the images created from the field videos (both images created by bulk labeling and by audio label). The purpose of this module is to combine automated and human quality control, to remove out of context and low quality images and retain only high quality images for subsequent

classification. High quality images will enable better classifier training and performance. The degree to which aesthetics matter for the classifier is not entirely clear. Sharpness is important, but poorly chosen backgrounds and offensive content might not matter for the classifier. The human image organizer plays a significant role in the compilation of these image sets. This is a new field of design.

The following options are available in this module:

Remove-selected:

Manually select images for removal. Select (multiple) images with a left click, confirm and then click 'remove-selected'

Remove-divergent:

Select a single image as reference with left click, confirm, and then click 'remove-selected'.

Hit <enter> to update the page after the removal process has completed.

Images that deviate from this selected reference in luminosity beyond the set min /max levels (under and overexposure) will be deleted. Other images that deviate structurally (different visual contexts or blurry images) beyond the set similarity measure will be deleted.

Once the image set has been reviewed you can add the resultant set to the collection. Once you have several collections / categories, you can archive the collection (compress the data sets) or, if something is amiss, delete everything and start again.

The final archived (compressed) collection will be the input to the classification procedures as described in the next module.

Aside -

Each image category should have at least 1500 viable images in order to offer enough information to neural net classifiers. Visually complex categories require substantially more than that. Collections very many categories have higher collection size requirements. More information on this issue forthcoming.

## **Train A Classifier With An Image Set**

This module allows one to create a fully functional – albeit low dimensional - convolutional neural network from one of several sample image sets. This module offers some insights into the impact of parameter settings on network training and performance. Altering the parameters will impact training time and classifier performance. For example, reduced epochs and fewer images per category will shorten the training time but may compromise accuracy.

Two networks available: a simple (vanilla) CNN and the Alexnet architecture.

The ore-trained option is available for the Alexnet architecture and usually improves performance – but does not allow control over pre-training and thus the logics with which your system is primed. This may be undesirable when networks are pre-trained on ImageNet or other biased image collections.

You can select the normalization. A set of image normalization parameters have been calculated from the bali-26 image collection (support local image logic!) and work well with the samples provided, which include:

bali-3	Cacao, Passiflora and Aroid
bali-3C	Durian, Nilam and Snakefruit
bali-3D	Coffee Arabica, Papaya, Sugarpalm

Unless you have a GPU enabled computer, training on your home device is a slow process. (55 min for 3 categories with 1200 images trained for 20 epochs with Alexnet model on an Intel i7-4770 CPU @3.4GHz with 16GB RAM).

Click 'train-network' once you make your choices and go for a walk. Training on generic computers will take an hour or so. When all is done, the training results and Top-1 errors per category are diagrammed. If the images do not reflect the choices you made, you probably have a caching issue. Prevent page update snafus with Cache Killer for Chrome. Check options to enable on start. (The Cach Killer icon above the bookmarks bar is 'green' when active, 'gray' when inactive).

Before you try different settings and retrain a network, click 'back to start' and then 'train classifier' again to clean up previous records.

*Aside -*

You can in fact train a deep network on a large data set with the code inside C+R if you move the internal routines to a remote GPU computer. The GPU version requires the installation of nvidia drivers. See the *venv+pytorch+nvidia.txt* file in the 'remote' folder for step-by-step instructions.

Copy the following 4 files from the 'remote' directory of the repository to a directory on the server:

*pyt\_trainandsave.py    pyt\_loadandeval.py    pyt\_utilities.py    create\_norms.py*

Place your data collection produced by C+R on the server as well and adjust the paths in the 4 python files. The *pyt\_trainandsave* program will train your chosen network on the data collection; *pyt\_loadandeval* will evaluate its performance. If you want to adjust the image normalization to your own image set, run 'create\_norms.py' and replace the results with the bali-norms used here.

These routines were used to train the deep Resnet152 and Resnext50 classifiers on the 50'000 images of the bali-26 collection. The results can be evaluated in the next module.

## **Test Sample Images On Trained Deep Networks**

This module loads one of several deep neural net classifiers trained on the bali-26 dataset and allows you to select an image from the validation set to test the selected classifier performance. The best prediction and the confidence level are listed as are the top 3 possibilities (two alternates to the first choice). These results are calculated in real time on your local computer.

Left click to select a single image from the image gallery, then click 'classify image' to obtain the results. The first time you run C+R, the selected trained model will be downloaded from pCloud automatically. The gallery will only load once. To inspect additional images, left click to select a new image, then click 'classify image' again, and repeat.