README - Catch & Release version 1 spring 2020

Introduction

Catch & Release (C+R) is a collection of procedures that allow you to apply machine learning classification to 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 them.

Furthermore, C+R can extract text from video. It can extract labels from video and use them as labels for 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

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

License

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

Platform Information

C+R runs on Linux under Python3 and Flask with Chrome or Firefox. C+R uses the PyTorch framework to train and test the image classifiers. Library versions and dependencies are given in the requirements.txt file. C+R has been tested on Ubuntu 18.04 TLS with kernel 5.3.0 on images sourced from .mp4 and .webm video formats (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: Chrome.
Install the free Chrome Cache Killer:
https://chrome.google.com/webstore/detail/classic-cache-killer/kkmknnnjliniefekpicbaaobdnjjikfp?hl=en

Software Installation

At the command prompt, run the following commands:

sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get install python3-dev -y
sudo apt-get install build-essential python3-dev -y
sudo apt-get install python3-venv -y
sudo apt-get install ffmpeg -y
sudo apt-get install sox -y

Create a virtual environment:
python3 -m venv env

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

Clone the C+R repository on GitHub

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>

Install Requirements and Dependencies

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 Google. If you want to make use of the text from video extraction you should get an access key to the Google Speech API. Creation of the key is free and you can use it in this software at no cost as C+R operates within free limits of the API. However, you do need a google account to create the key. If that is not palatable, skip the sections that use 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 not yet integrated into C+R. The key barrier at this point are adequate and high quality language corpora in less popular languages. Big issue.

Launch C+R

Activate the virtual environment: *source ./env/bin/activate*
Start C+R: *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

crtl + / 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 (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.

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

B > train a classifier
- > t*rain 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 lets you chop a long field video into smaller parts for processing. Supported formats are .mp4 and .webm. Select the video and choose the chunk size. Suggested are about 3 minute segments.
Chunked segments are saved to the C+R /tmp directory, and deleted if you go back to start or exit the program. You can also download some sample videos to process and play around with.

When you have some data ready, continue either 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 find a section you want to get text from, then reload to capture the text.
This module requires an access code to 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 can take several minutes to complete. Best to limit the difference between start and end time 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 if an expert wants to annotate a field video. Pick your field video and select the microphone input. The default mic is cardn = 0, devicen = 0. That is usually the default on a Linux box.

You can add you favorite mic to the list of available microphones. Plug in a (USB) microphone.
At the command prompt in a terminal, type:

arecord –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 pick the USB mic you just added.

Then proceed as follows. Load the video (from samples, tmp or annoate folders) to find the spot you want to voice-over on. Set start and end times to define the interval you will 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 that 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 start. You can process this voice-over video (stored in the annotate folder) to labeled images in 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 select a single key term and the number of images per utterance. Also set a confidence level for the speech to text API and the language spoken in the video. Pick 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 goal of this module 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 cleaned up 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 viable1500 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 you 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. Changing the parameters will impact training time and classifier performance. For example, reduced epochs and fewer images per category shorten the training time but compromise accuracy.

Two networks available: a simple (vanilla) CNN and the Alexnet architecture.
Pre-trained option is available for the Alexnet architecture and usually improves performance – but you have no control over pre-training, and the logics with which your system is primed. This can 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 here, 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 your choices are made 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 v*env+pytorch+nvidia.txt* in the ‘remote’ folder for step by step instructions.

Copy the following 4files 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 on the 50’000 image bali-26 dataset, the deep Resnet152 and Resnext50 classifiers, the responses of which you can test 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 lets you 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.

Achtung: Click 'display image set' PRIOR TO 'classify image’ !
Left click to select a single image from the image gallery, then click ‘classify image’ to obtain the results. The gallery will only load once. To check more images, just left click to select an additional image, then click ‘classify image’ again, repeat.