For shuffler documentation see:

The ‘Functionality’ section here:

<https://github.com/kukuruza/shuffler>

And subcommands:

<https://github.com/kukuruza/shuffler/blob/master/doc/Subcommands.md>

#To get main commands from the terminal:

shuffler/shuffler.py -h

#To get help with subcommands for a specific main command :

shuffler/shuffler.py exportLabelme -h

# in -i and out -o databases go at the front of the command

**#Install Shuffler and Prepare Data**

**Resize Images**

#go to your stamps project folder

$cd path\_to\_stamps\_folder/stamps

#Use 1800x1200 images or resize original 6000x400 images using ImageMagick on Bridges

#Check image size:

$identify \*.jpg

#If you need to resize:

$mogrify -resize 1800x1200 images/\*.jpg

#Change your final image directory folder structure to:

labelme/Images/HistoricalDocuments-1800x1200/\*.jpg

labelme/Annotations/\*.xml

**Shuffler Installation**

#start an interact session on bridges

$interact --egress

#clone the shuffler repo

$git clone https://github.com/kukuruza/shuffler

#create virtual environment on Bridges

$module load anaconda3/5.1.0

$source activate

$conda create -n shuffler python=3

$conda activate shuffler

$rm -rf /home/myilmaz/.conda/pkgs/ffmpeg-4.0.2-ha0c5888\_2

$conda install -y -c conda-forge ffmpeg=4.0

$conda install -c menpo opencv3 && conda install -y imageio matplotlib lxml simplejson \ progressbar2 pillow scipy=1.1.0

$conda install -y pandas seaborn

**Shuffler Instructions for STAMPS**

# Import Labelme to Shuffler format.\*do this from the project folder (stamps)

$shuffler/shuffler.py -o stamps-1800x1200.db \

importLabelme \

--images\_dir labelme/Images/HistoricalDocuments-1800x1200/ \

--annotations\_dir labelme/Annotations

#Annotations are for image size of 6000x4000, but the images are now 1800x1200.

$shuffler/shuffler.py -i stamps-1800x1200.db -o stamps-1800x1200.db \

sql "UPDATE images SET width=6000, height=4000;" \| \

resizeAnnotations --target\_width 1800 --target\_height 1200

##### You can skip inspecting the results.

$shuffler/shuffler.py -i stamps-1800x1200.db examineObjects --shuffle --winsize 1000

######You can skip making a video

$mkdir results

$shuffler/shuffler.py -i stamps-1800x1200.db \

filterEmptyImages \| \

writeMedia --media video --image\_path results/stamps-1800x1200.avi \

--with\_imageid --with\_objects --overwrite

# This step rewrites all the stamp classes so they are all “stamp” since we are detecting all stamps.

$cp stamps-1800x1200.db stamps-1800x1200-1class.db  
$sqlite3 stamps-1800x1200-1class.db 'UPDATE objects SET name="stamp";'

####### You can skip recording a video

$mkdir -p results

$shuffler/shuffler.py -i stamps-1800x1200-1class.db \

filterEmptyImages \| \

writeMedia --media video --image\_path results/stamps-1800x1200-1class.avi \

--with\_imageid --with\_objects --overwrite

# Export to COCO format.

$shuffler/shuffler.py -i stamps-1800x1200-1class.db \

filterEmptyImages \| \

exportCoco --coco\_dir datasets/stamps-1800x1200-1class-2 --subset train2017 \

--copy\_images

# We need to create a validation dataset for the code to work.

# Since we don’t have enough data, just copy the training one for now.

$cp datasets/stamps-1800x1200-1class-2/annotations/instances\_train2017.json \

datasets/stamps-1800x1200-1class-2/annotations/instances\_val2017.json

$cd datasets/stamps-1800x1200-1class-2/images

$ln -s train2017 val2017

#Go back to your stamps project folder

$cd path\_to\_stamps\_folder/stamps

#Create database pointing to unannotated images

$shuffler/shuffler.py -i stamps-1800x1200-1class.db -o empty-only.db filterImagesSQL "SELECT DISTINCT(imagefile) FROM objects"

#exit interact

$exit

#exit virtual environment

$conda deactivate

**# Keras-Retinanet Installation.**

#start an interact session

$interact -p GPU-small -t 2:00:00 --egress --gres=gpu:p100:1

# Clone Evgeny’s forked Keras-Retinanet repository (contains detect.py)

$git clone <https://github.com/kukuruza/keras-retinanet>

#Original repository for reference: https://github.com/fizyr/keras-retinanet

#go into the repository

$cd stamps/keras-retinanet

# Make a new Anaconda environment and activate it

$conda create -n keras-retinanet python=3

$conda activate keras-retinanet

$conda install numpy tensorflow-gpu==1.15 keras cython opencv=3.4.2 progressbar2 pillow imageio

$pip install imageio-ffmpeg

$conda install pycocotools -c conda-forge

$pip install keras-resnet

$python setup.py build\_ext --inplace

$pip install keras\_resnet

$conda uninstall keras && pip install keras -U

# Increase keras version to keras==2.3 if there is a warning while running this step (min version #required is 2.3 #but conda has 2.2.4)

#Make sure you’re in stamps/keras-retinanet

# Download weights of trained COCO model to directory "snapshots".

$wget <https://github.com/fizyr/keras-retinanet/releases/download/0.5.1/resnet50_coco_best_v2.1.0.h5> \

--directory-prefix=snapshots

#make tensorboard folder

$cd stamps

$mkdir tensorboard

**Keras-Retinanet Instructions for STAMPS**

#train model

$cd keras-retinanet

# Train for 10 epochs, 1000 steps each.

$keras\_retinanet/bin/train.py --epochs=10 --steps=1000 \

--tensorboard-dir=../tensorboard/stamps-1800x1200-1class\

--weights snapshots/resnet50\_coco\_best\_v2.1.0.h5 \

--snapshot-path snapshots/from-coco-weights \

coco ~/scratch/stamps/datasets/stamps-1800x1200-1class-2

# Will evaluate on epoch:

epoch=10

# Evaluate on COCO

$keras\_retinanet/bin/evaluate.py \

--convert-model \

coco ../datasets/stamps-1800x1200-1class-2 \

snapshots/from-coco-weights/resnet50\_coco\_10.h5

# Detect stamps on unlabelled images. Results are written as images and as a Shuffler db.

$mkdir project\_folder/keras-retinanet/examples/detected/epoch10-test

#This is the original command without the epoch substituted:

$SHUFFLER\_DIR=~/scratch/stamps/shuffler CUDA\_VISIBLE\_DEVICES=0 python3 keras\_retinanet/bin/detect.py \

-i ~/scratch/stamps/stamps-1800x1200-empty-1class.db \

--rootdir ~/scratch/stamps \

-o examples/detected/epoch${epoch}-test.db \

--model\_path snapshots/from-coco-weights/resnet50\_coco\_${epoch}.h5 \

--out\_dir examples/detected/epoch${epoch}-test

#THIS IS ALL ONE LINE with the epoch substituted:

$SHUFFLER\_DIR=~/scratch/stamps/shuffler CUDA\_VISIBLE\_DEVICES=0 python3 keras\_retinanet/bin/detect.py \

-i ~/scratch/stamps/stamps-1800x1200-empty-1class.db \

--rootdir ~/scratch/stamps \

-o examples/detected/epoch50-test.db \

--model\_path snapshots/from-coco-weights/resnet50\_coco\_50.h5 \

--out\_dir examples/detected/epoch50-test

#make folder for model-annotated images

$mkdir project\_folder/newLabelme

#detect stamps and save new images with annotated stamps

$shuffler/shuffler.py -o newLabelme/0123export.db \ -i ~/scratch/stamps/keras-retinanet/examples/detected/epoch50-test.db exportLabelme \ --images\_dir newLabelme/images --annotations\_dir newLabelme/annotations

#File examples/detected/epoch10-test.db contains all the detection, folder #examples/detected/epoch10-test has the images with detections drawn on top.

#Try Shuffler subcommands from “info” and “GUI” sections for analyzing detection results and plotting histograms:

<https://github.com/kukuruza/shuffler/blob/master/doc/Subcommands.md>

#For testing with train dataset.

$SHUFFLER\_DIR=~/scratch/stamps/shuffler CUDA\_VISIBLE\_DEVICES=0 python3 keras\_retinanet/bin/detect.py \

-i ~/scratch/stamps/stamps-1800x1200-1class\_no\_objects.db \

--rootdir ~/scratch/stamps \

-o examples/detected/epoch10-test.db \

--model\_path snapshots/from-coco-weights/resnet50\_coco\_10.h5 \

--out\_dir examples/detected/epoch10-test