Welcome!

Welcome to the Complete Guide to Creating COCO Datasets, where you'll learn to build your own image datasets automatically with Python. I'm so excited that you've decided to take this course because I really believe that synthetic datasets are the future of training neural networks, especially image recognition. Understanding how to create a synthetic image dataset has been extremely valuable both at my professional job and in my personal projects, so I'm confident that it can provide value to you as well.

Feel free to ask for help if you have trouble with anything in the course! There's value in trying to solve an issue on your own, but not in quitting because you're completely stuck.

Don't just watch, follow along! I know I do my best learning when I'm actually *doing* things. Try to create your own, unique image dataset as we go. Seems obvious, but a lot of people don't do this.

How to connect with me

You can find all of the ways to connect with me at http://www.immersivelimit.com/connect. I love connecting with like-minded people and getting feedback on the content I'm creating.

About this document

This document contains all the links and extra information you need to get the most out of this course.

If you find any broken or missing links, or think a section needs more clarification, please let me know and I'll fix it as quickly as I can.

Most recent update: March 29, 2019

Section 1: Course Introduction

Setup Resources

Feel free to download these resources all up front or as you progress through the course.

CocoSynth Git Repo

https://github.com/akTwelve/cocosynth

The CocoSynth git repo contains all code specific to this course. You can clone it using git, or just download the zip file from GitHub.

Action items:

- Clone the repo to your computer
- Check out the docs inside the repo for initial setup instructions

Anaconda

https://www.anaconda.com/distribution/

Anaconda allows us to keep our Python environments organized. Unless you know of some reason not to use it, you probably should. It will make things easier, especially on Windows.

To set up your Anaconda environment for this course, you can use these commands:

```
conda create -n cocosynth python=3.6

conda activate cocosynth

conda install -c conda-forge shapely

pip install -r requirements.txt
```

Note: I suggest python 3.6 because tensorflow-gpu doesn't seem to like 3.7 yet.

Action items:

- Download and install Anaconda
- Set up your "cocosynth" conda environment using the commands above

Jupyter Notebook

Hopefully you're already familiar with Jupyter Notebooks since most Deep Learning courses use them. If not, fear not, they're easy to use. You might want to search YouTube for a beginner tutorial.
Installation Instructions: https://jupyter.readthedocs.io/en/latest/install.html
Running a Notebook: https://jupyter.readthedocs.io/en/latest/running.html

Note: The folder you launch your Jupyter Notebook from will be the root directory, so I suggest you change directories to the directory containing your cocosynth repo so that you can easily access another repo.

Action items:

- Run Jupyter Notebook from inside your cocosynth Anaconda environment

Visual Studio Code (Optional)

https://code.visualstudio.com/
I prefer Visual Studio Code for my development and that's what you'll see me using when I do the code walkthroughs. It provides Python syntax highlighting, git integration, markdown previewing, and an ever expanding list of features and addons. It works on Windows, Linux, and Mac. That said, feel free to use another code editor if you prefer.

Action items:

- Make sure you have a good code editor. VS Code is a great choice.

GIMP

https://www.gimp.org/downloads/
We'll use GIMP as our image editor in this course. I downloaded the most recent version before recording videos for the course, but I think some versions of Linux come with an older version of GIMP. If your interface looks different, that's probably why.

Action items:

- Download and install GIMP

COCO Dataset (Optional)

http://cocodataset.org/#download

If you're interested, download the 2017 Validation images and the Instances Train/Val Annotations and try to view images with the COCO Image Viewer.

Course Datasets

I'm including the dataset .zip files in the first lecture of relevant sections of the Udemy course.

Section 3 & 4 downloadable resources:

- box_dataset_by_hand
    - Contains sample images you can use to follow along
- box_dataset_by_hand_complete
    - My resulting images, masks, and json files after completing Section 4

Section 6 downloadable resources:

- box_dataset_synthetic
    - Sample input foregrounds and backgrounds you can use to follow along
- box_dataset_synthetic_complete
    - My resulting images, masks, and json files after completing Section 6

Action items:

- Download the datasets as you get to the relevant sections

Tensorflow GPU

https://www.tensorflow.org/install/gpu

Tensorflow GPU is (I think) required for Mask R-CNN. It speeds up neural network training and inference by a factor of at least 8 in my experience. As I mention in the course, it can be a bit of a pain to install CUDA and CUDNN to make it work, but the result is absolutely worth it. Just read the instructions carefully.

In the past, I've had difficulty getting tensorflow-gpu to work without ipykernel. You can set it up in your cocosynth Anaconda environment with this command:

```
conda install ipykernel

python -m ipykernel install --user --name cocosynth --display-name "Python (cocosynth)"
```

Action items:

- Install tensorflow-gpu using the instructions at the link provided
- Install ipykernel for your Anaconda environment
- Set the new kernel inside your Jupyter Notebook using the top menu: Kernel > Change Kernel > Python (cocosynth)

Matterport Mask R-CNN

https://github.com/matterport/Mask_RCNN

This code is loaded in the "train_mask_rcnn.ipynb" Jupyter notebook. I recommend trying out some of their examples first to make sure it's working properly and get familiar with it. It's an awesome open-source project. I even contributed a bug-fix to it. :)

If you clone it to your computer right next to the cocosynth repo directory and call it "matterport_mask_rcnn", it will be easier to use in this course.

Action items:

- Clone the Matterport Mask_RCNN repo to your computer
- Try out some examples from their repo to make sure everything is working correctly

Other Links

Cigarette Butt Project

http://www.immersivelimit.com/blog/training-an-ai-to-recognize-cigarette-butts

Here's a link to my completed cigarette butt project on my blog.

GANs Research Paper

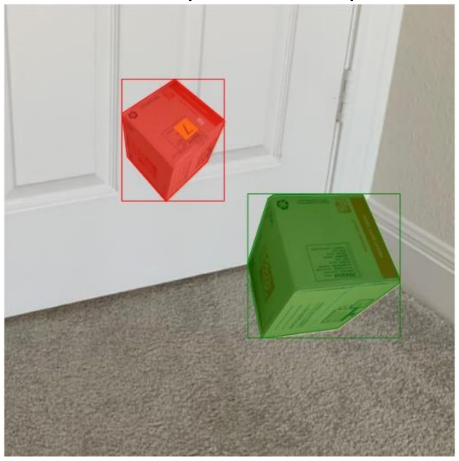
https://arxiv.org/pdf/1711.11585.pdf

High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs

Section 2: COCO Image Viewer

In this section, we work through the COCO Image Viewer Jupyter notebook, which you will find in the "notebooks" directory of the cocosynth repo. We'll use it to preview images and instance segmentations in COCO datasets. We'll walk through the code and understand how to display bounding boxes, polygonal overlays, and RLE (run length encoded) overlays. By the end of the notebook, you should have a good understanding of how COCO instance annotations work and have code you can use or modify as desired.

Here's an example of the output of the COCO Image Viewer:



Recommended Additional Material

COCO Dataset Format - Complete Walkthrough

https://www.youtube.com/watch?v=h6s61a_pqfM

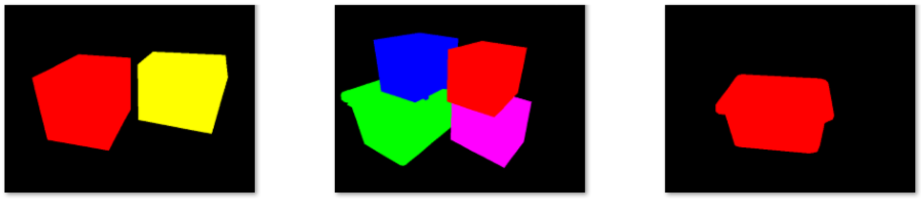
Important Note RLE is actually described incorrectly in this YouTube video. I had the x and y axes flipped, so just go with what is taught in the course. One of the downsides of YouTube is that I can't go back and fix it!

Section 3: Dataset Creation with GIMP

In this section, we are going to create a small image dataset! In order to do that, we'll need some images. You can use the images from the "box_dataset_by_hand" dataset linked above, but I'd suggest you use your own images. You can take images with your phone, camera, or just find some on the internet. Be aware that the more complex/intricate the outline of your object, the more work you'll have to do.

For each image, we're going to make a black background and then create a solid colored layer for each distinct object in the image. We'll also create a mask_definitions.json file that will define what each color means in each image. The JSON file will be used later in the course.

Here are some examples of masks:



Section 4: COCO JSON Utils

In this section, we'll walk through the code and usage of coco_json_utils.py, which can be found in the "python" directory of the cocosynth repo. This tool converts our images, masks and mask_definitions.json into actual COCO annotations/instances.

By the end of this lecture, we will convert our small dataset of images and masks into actual COCO Annotations.

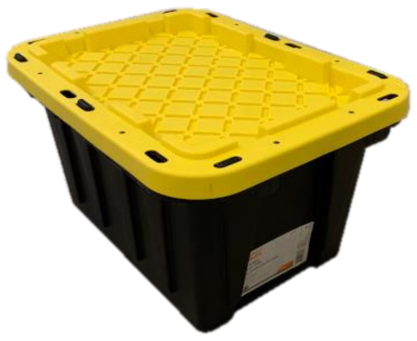
Example terminal command (inside the cocosynth directory):

```
python ./python/coco_json_utils.py -md
./datasets/box_dataset_by_hand/mask_definitions.json -di
./datasets/box_dataset_by_hand/dataset_info.json
```

Section 5: Foreground Cutouts with GIMP

In this section we are going to start creating our synthetic dataset using GIMP. To do so, we will need image foregrounds. Image foregrounds are clear pictures of each object you want to detect taken from various angles on a completely transparent background. This is necessary because we will be "pasting" the foreground images onto many different backgrounds.

Here's an example foreground:



Again, if you are using complex or intricate objects this part will take more work, but sometimes that can't be avoided. The more examples you have, the better your dataset will be. That said, you will need to cut out every detail of each object and that can be extremely time consuming. To keep things simple, I'm demonstrating with relatively simple cardboard and plastic boxes.

We are going to walk through two different methods to cut out our images. It's encouraged that you try both methods to see which one will work best for your images. One method uses GIMP's foreground select tool to automatically create edges and another is simply cutting the image out manually with the select tool.

At the end of this section, we will have a number of different foregrounds to use to create a large, diverse, synthetic image dataset.

Section 6: Image Composition

In this section we are using image_composition.py to do image composition. We will take the foregrounds that we created in the previous section and paste them randomly on top of backgrounds. Because those foreground images have a transparent background we can place them onto backgrounds and end up with photos that are realistic enough to train our neural network with.

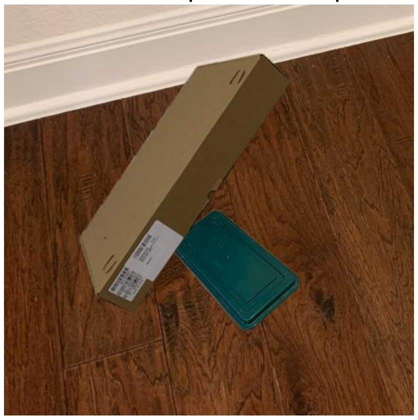
We are going to walk through the code and see how foregrounds with transparent backgrounds are "pasted" on top of different background images. Because we're pasting, we know exactly which pixels belong to the object and can create pixels masks automatically and store this information in a mask_definitions.json file.

At the end of this section you will understand how this script can be used to automatically generate thousands of images and segmentation masks very quickly.

Example terminal command (inside the cocosynth directory):

```
python ./python/image_composition.py --input_dir
./datasets/box_dataset_synthetic/input --output_dir
./datasets/box_dataset_synthetic/output --count 10 --width 512 --height 512
```

Here's an example of a composed image:



Section 7: Training Mask R-CNN

In this section, we are going to use the "train_mask_rcnn.ipynb" Jupyter notebook to train a neural network to spot our objects in images and tell us exactly which pixels belong to each object. Using the tools that we've learned throughout this course, you should:

Create two datasets

- a couple thousand training images
- a few hundred validation images.

Example terminal commands (inside the cocosynth directory):

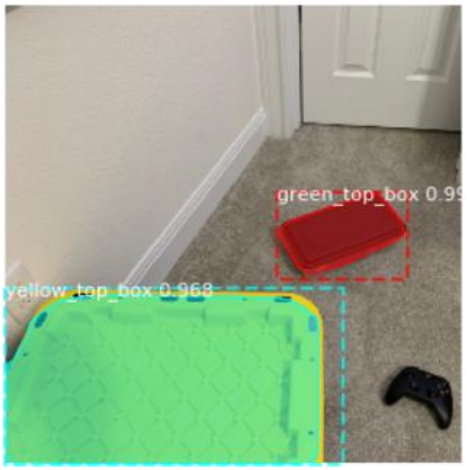
```
python ./python/image_composition.py --input_dir
./datasets/box_dataset_synthetic/input --output_dir
./datasets/box_dataset_synthetic/train --count 2000 --width 512 --height 512
```

```
python ./python/image_composition.py --input_dir
./datasets/box_dataset_synthetic/input --output_dir
./datasets/box_dataset_synthetic/val --count 300 --width 512 --height 512
```

Run Mask R-CNN

We will run these through the Mask R-CNN and by the end, you should have a trained neural network that can spot your objects in photos. YAY!

Here's an example of objects detected by Mask R-CNN:



Troubleshooting

Due to the complexity of Mask R-CNN, it may fail in different ways on different system setups. I recommend checking Google first, but you can also reach out to me and I'll try to help. Below are some issues I've seen.

Failed to get convolution algorithm

If you hit an error like the following:
UnknownError: Failed to get convolution algorithm. This is probably because cuDNN failed to initialize, so try looking to see if a warning log message was printed above.

Try restarting Jupyter notebook and running it again from the top. If that doesn't work, try restarting your computer. I've had this issue resolve itself without any other changes.

Section 8: Course Wrap

Congrats on making it to the end of the course. Hopefully you've had success in creating your own dataset and using it to train a Mask R-CNN. I'd love to see what you've accomplished, so let's connect on LinkedIn, Twitter, or Facebook and you can show me what you've made!
http://www.immersivelimit.com/connect/

It would be so helpful if you could provide feedback in the form of a Udemy review and if you got value from the course, it would mean the world to me if you shared it with someone else! Good luck, and I hope to hear from you in the future.

--Adam Kelly