

NEURAL NETWORK PROGRAM

USER MANUAL

- 1. INTRODUCTION**
- 2. GENERAL FEATURES**
- 3. PREREQUISITE**
- 4. START PROGRAM**
- 5. SEARCH EXISTING ASSETS**
- 6. RETRAIN OUR MODEL**
- 7. HOW TO CHANGE DATABASE SERVER**
- 8. CONCLUSION**

1. INTRODUCTION

This document lets you know how to use our program step by step.

Welcome to the Neural Network Team!

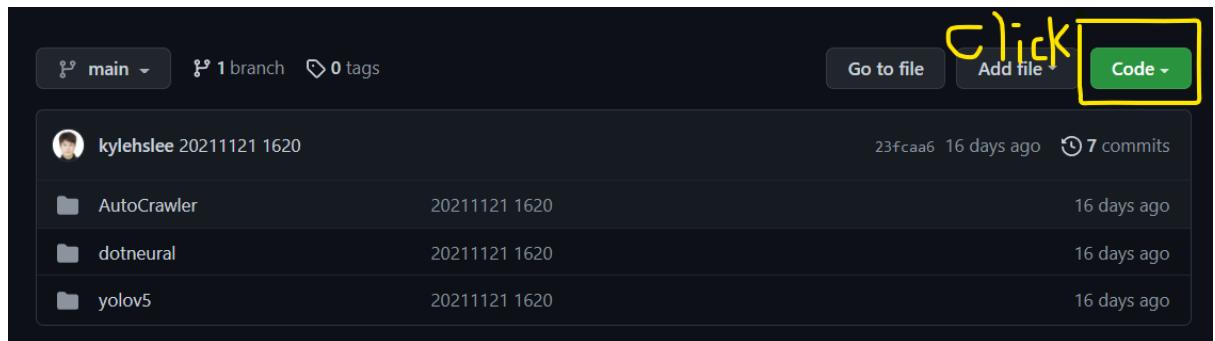
2. GENERAL FEATURES

On our program, You can

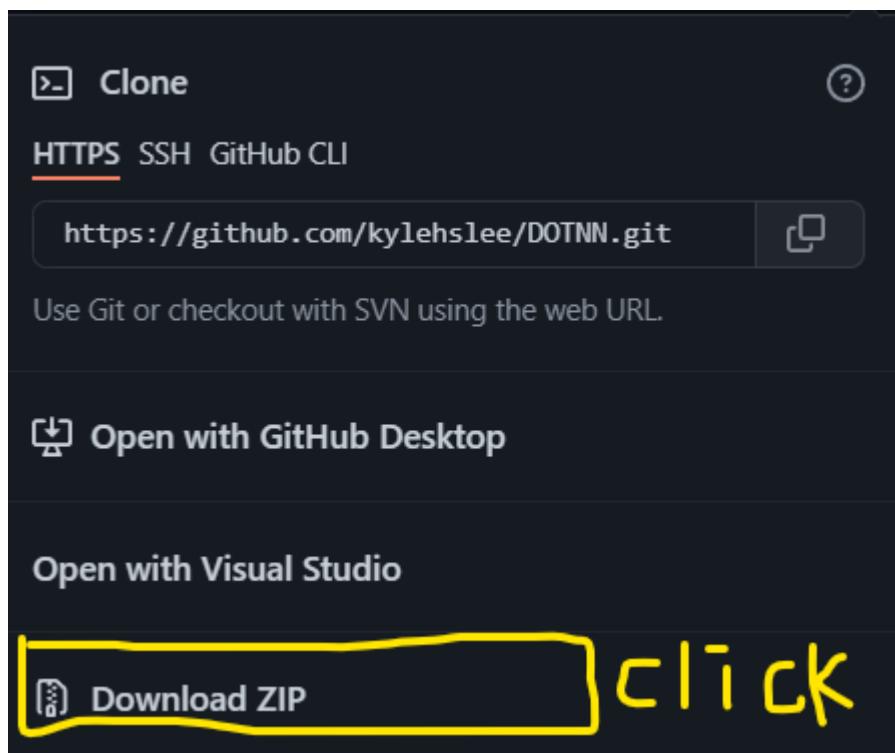
- Search assets, which are already stored on our database server.
- Retrain our model with new assets and detect those assets on street images if not on our server.

3. PREREQUISITE

1. Only Windows operations are supported. (No Macbooks, iMacs, Linux and so on)
2. Make sure you have installed Google Chrome on your computer. It will be needed for our autocrawler. <https://www.google.com/chrome/>
3. Go to <https://github.com/kylehslee/DOTNN>

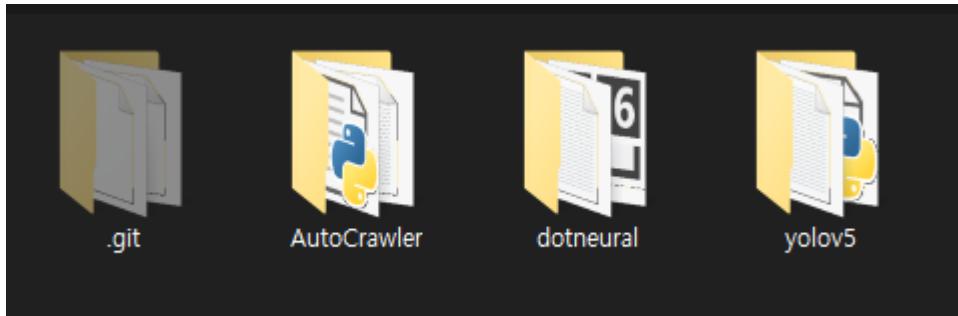


Click the “Code” green button.



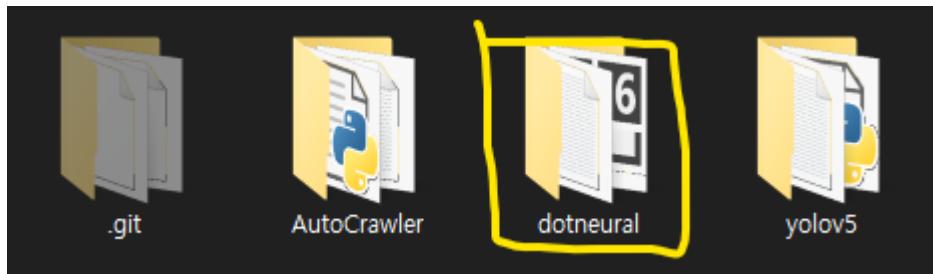
Click the “Download ZIP”. Then, you will successfully download our program.

4. Extract the zip file.

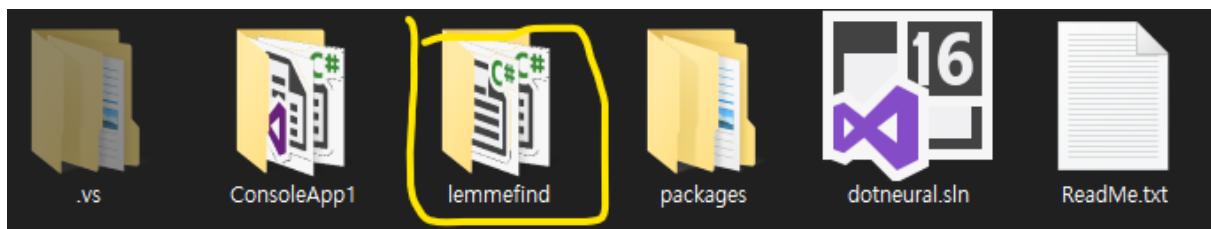


Now we are ready.

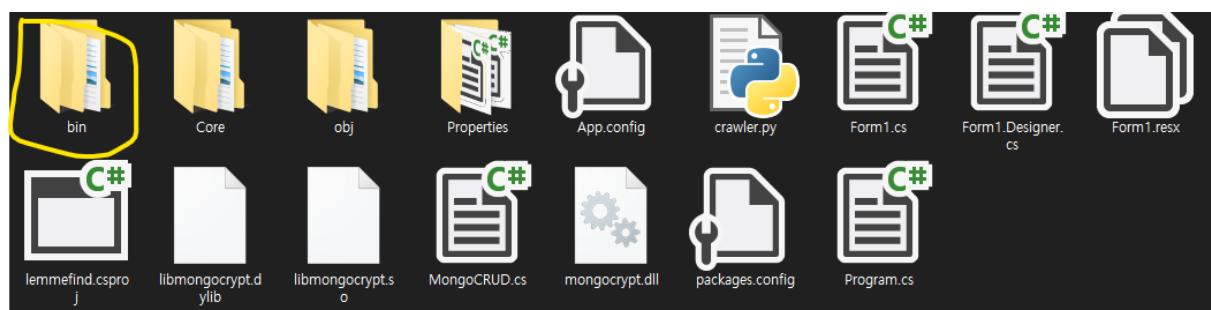
4. START PROGRAM



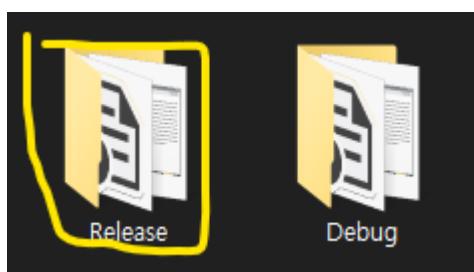
Click the “dotneural”



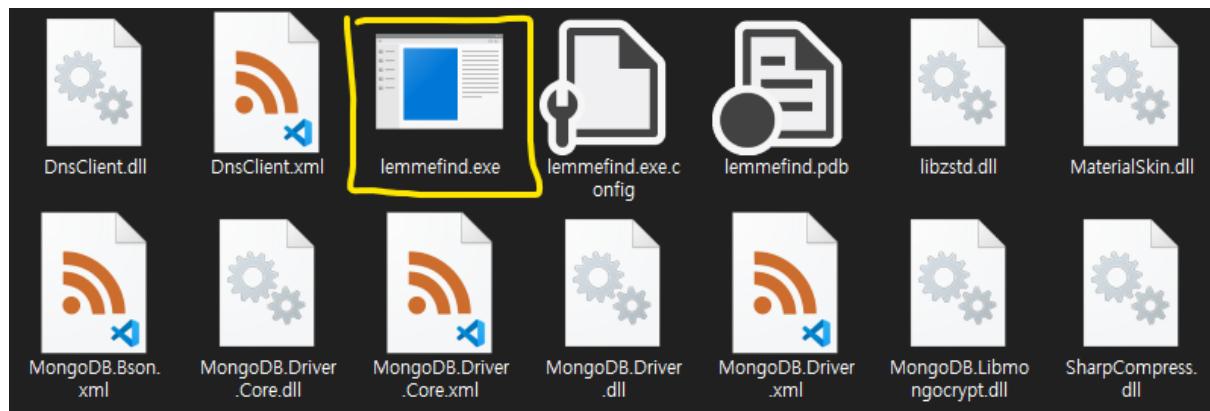
Click the “lemmefind”



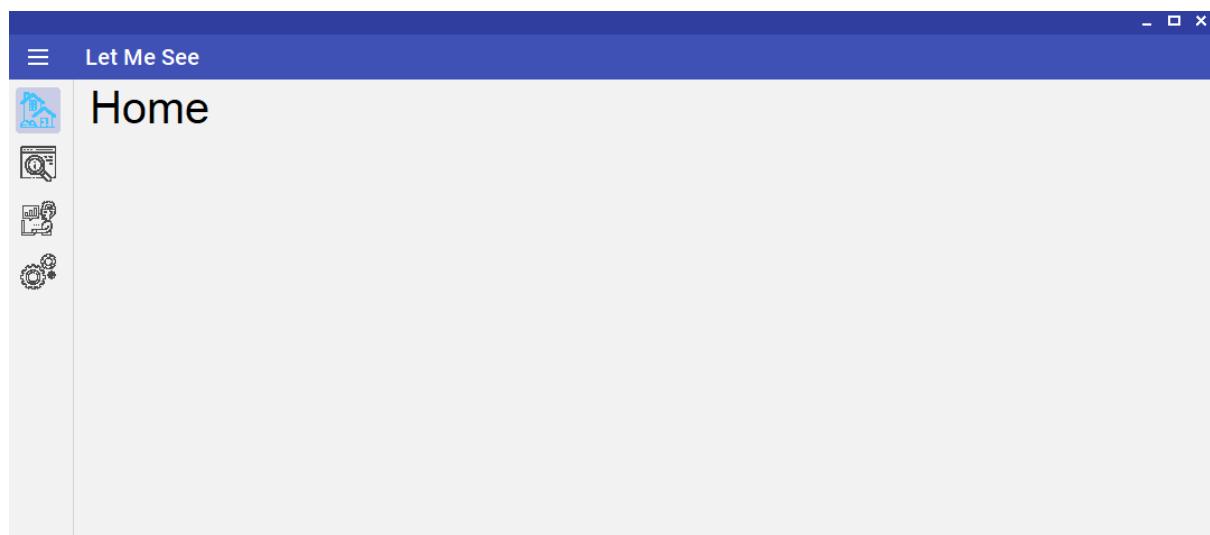
Click the “bin”



Click the “release”

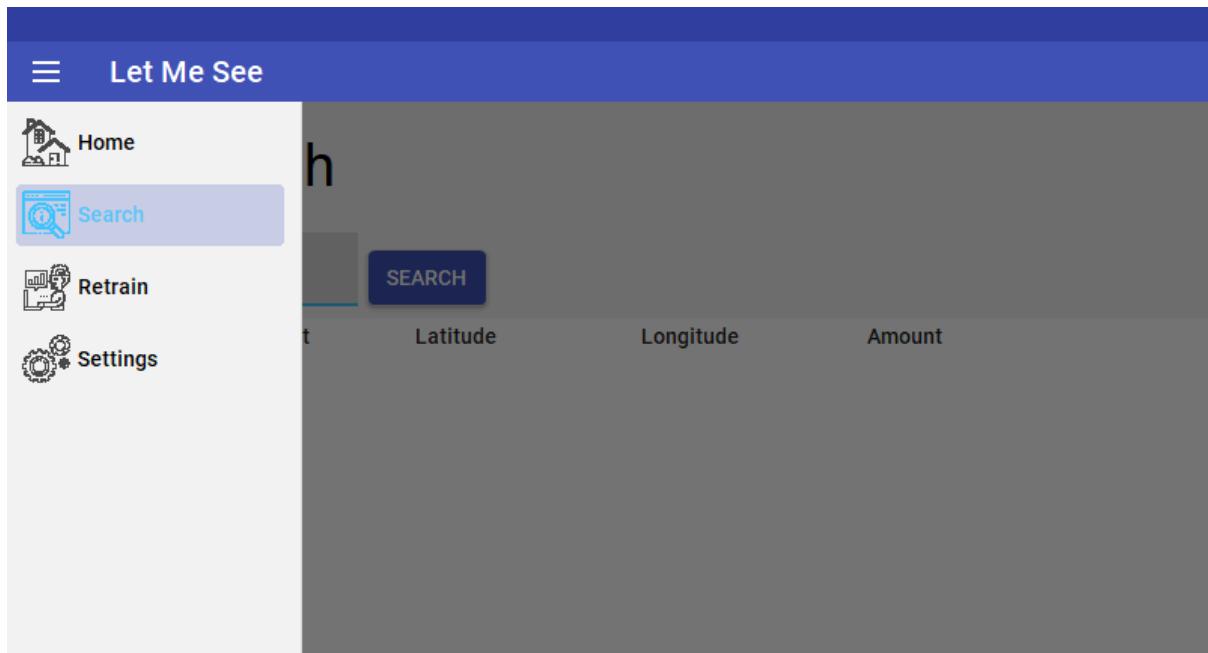


Launch "lemnemefind.exe".

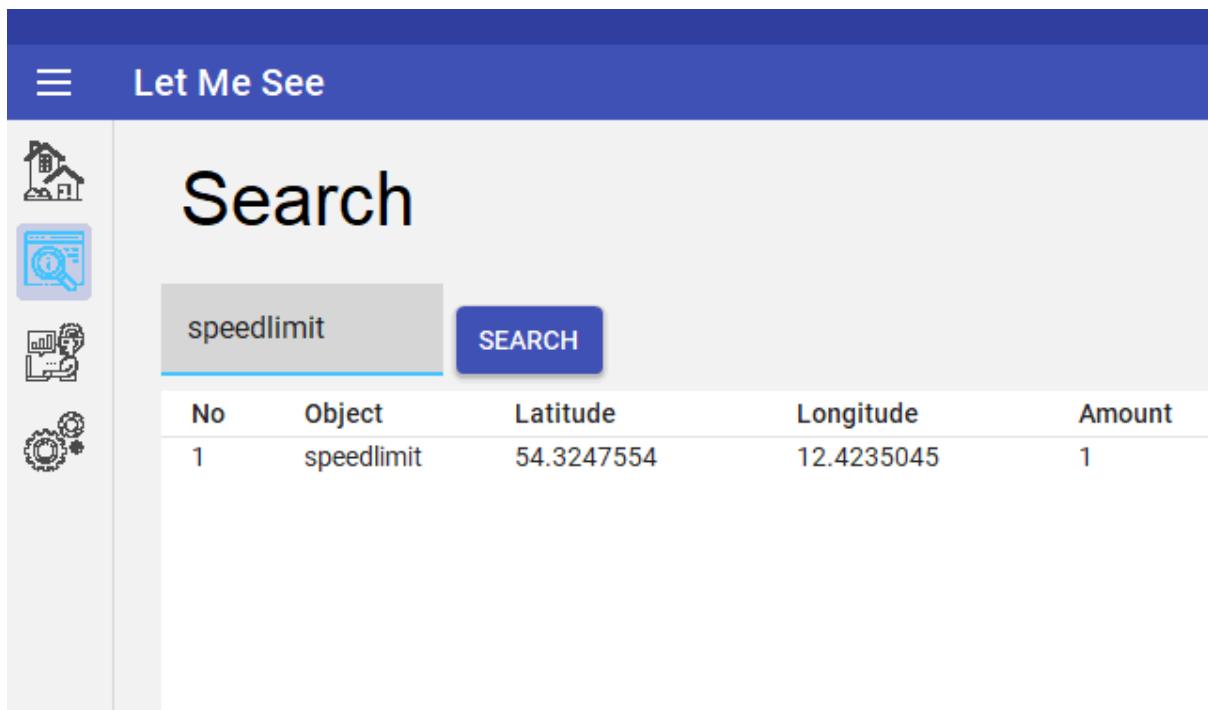


This is how you launch the program.

5. SEARCH EXISTING ASSETS



Go to the “Search” tap.



No	Object	Latitude	Longitude	Amount
1	speedlimit	54.3247554	12.4235045	1

You can search assets which are already stored on our database. It seems like there is only one data for a speed limit sign. You can get latitudes and longitudes of assets. “Amount” column lets you know how many objects there are on that latitude and longitude.



Search

trafficlight

SEARCH

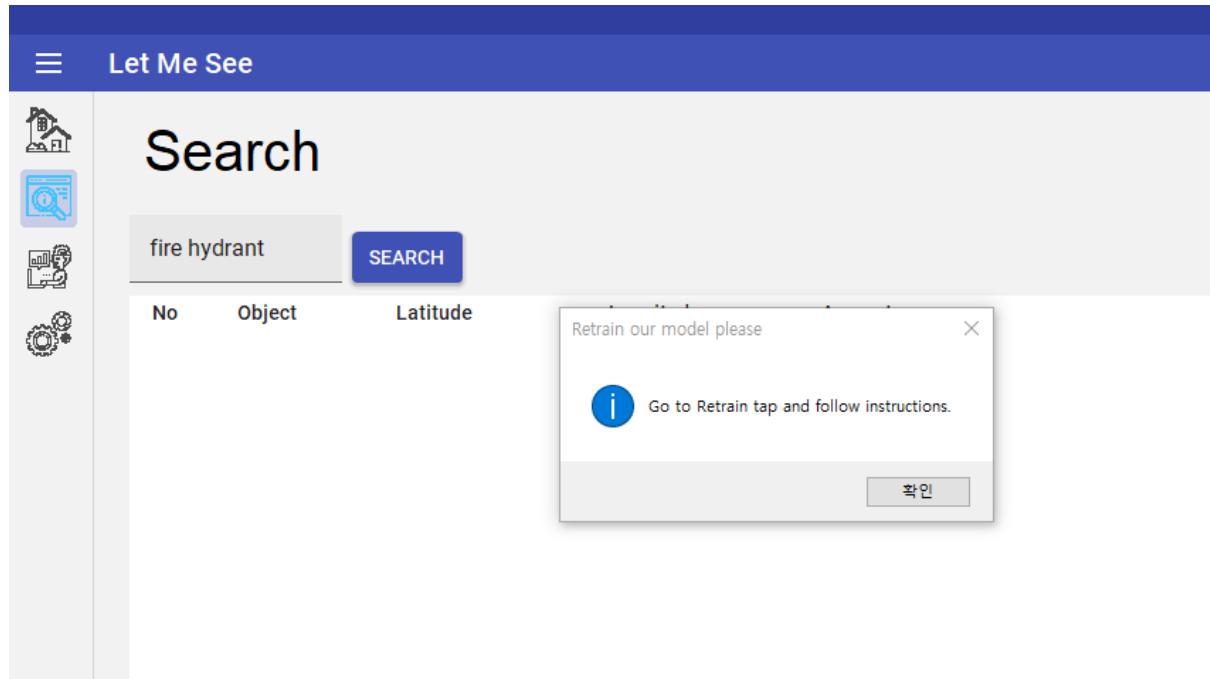
No	Object	Latitude	Longitude	Amount
1	trafficlight	13.6523754	18.5464578	1
2	trafficlight	13.6528757	43.5464578	3
3	trafficlight	13.6523754	86.5464578	2

The results which are shown on the result box are demo data, so it might look weird. What I would like to tell you here is that when you see No 2 traffic light, its amount is 3, which means on that location there are three traffic lights detected.

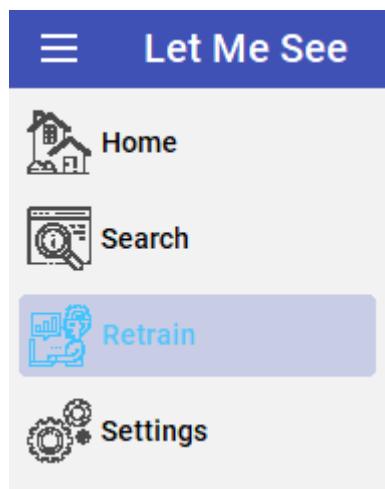
6. RETRAIN OUR MODEL

if there was not data that you were looking for, you can retrain our model with objects you like to find on the street and detect them on real street images.

Let's say you would like to find information on "fire hydrants". You might type like this.



However, you would not see data because our database does not have such data. Don't worry.



RETRAIN

<p>Step 1</p> <p>Type a keyword of an image you like to download</p> <hr/> <p>Choose the number of images you like to download</p> <p>50 ▾</p> <p>DOWNLOAD</p>	<p>Step 2</p> <p>Follow our user documentaion</p> <p>After following our user manual, press button below</p> <p>DETECT</p>
---	---

You will end up being here. From now on, it might be a little bit complicated to follow, but I will do my best to explain specifically.

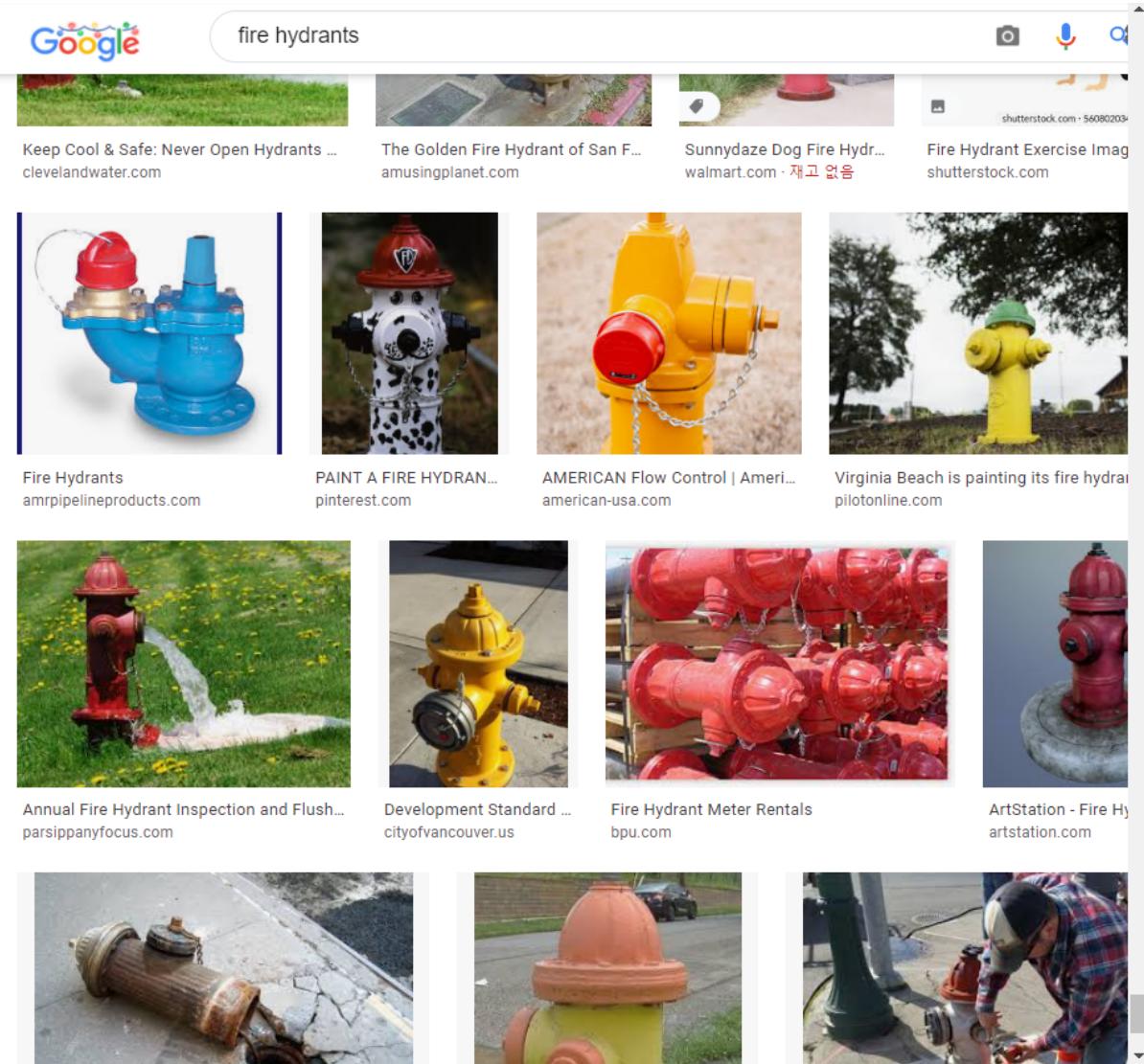
What you are looking for is a “fire hydrant”. If you have images of a fire hydrant, then you are good. However, if you do not, then you need to download images of it using our crawler. How? just type a keyword like below.

<p>Step 1</p> <p>Type a keyword of an image you like to download</p> <hr/> <p>fire hydrant</p> <p>Choose the number of images you like to download</p> <p>50 ▾</p> <p>DOWNLOAD</p>

That's it. Click the “download” button, then wait for about 1 minute.

Images will be downloaded in the “ AutoCrawler/download ” folder.

While our program downloads images, two things will come up on your screen.



The first thing is google chrome. It will look like it is doing something alone automatically.

```
C:\Windows\py.exe

=====
WebDriver manager =====
Current google-chrome version is 96.0.4664
Get LATEST driver version for 96.0.4664
Driver [C:\Users\Kyle\wdm\drivers\chromedriver\win32\96.0.4664.45\chromedriver.exe] found in cache

DevTools Listening on ws://127.0.0.1:54615/devtools/browser/b271ab05-0c4e-41d8-b7f9-25af1f63da31
[12816:18552:1207/095348.282:ERROR:chrome_browser_main_extra_parts_metrics.cc(226)] crbug.com/1216328: Checking Bluetooth availability started. Please report if there is no report that this ends.
[12816:18552:1207/095348.283:ERROR:chrome_browser_main_extra_parts_metrics.cc(229)] crbug.com/1216328: Checking Bluetooth availability ended.
[12816:18552:1207/095348.283:ERROR:chrome_browser_main_extra_parts_metrics.cc(232)] crbug.com/1216328: Checking default browser status started. Please report if there is no report that this ends.
[12816:1388:1207/095348.296:ERROR:device_event_log_impl.cc(214)] [09:53:48.296] Bluetooth: bluetooth_adapter_winrt.cc:1075 Getting Default Adapter failed.
[12816:18552:1207/095348.318:ERROR:chrome_browser_main_extra_parts_metrics.cc(236)] crbug.com/1216328: Checking default browser status ended.
```

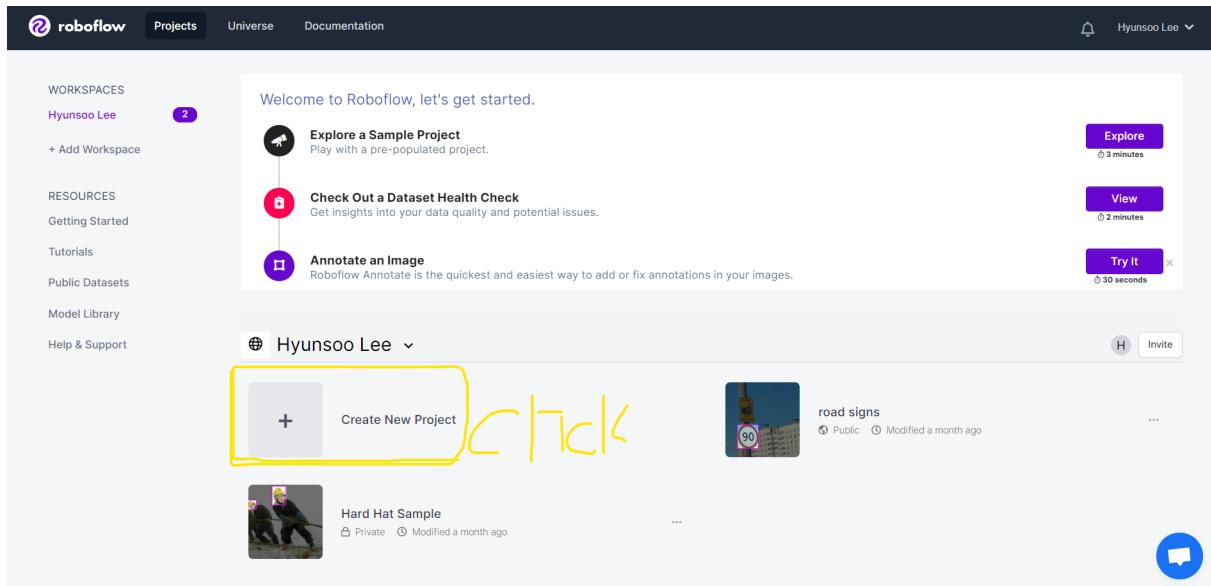
The second thing is a command prompt.

All you need to do is just wait for it to finish downloading. It will take about 1 minute for 50 images. After it finishes, those two screens will disappear automatically.

6.1. ADD ANNOTATIONS ON IMAGES

Go to the Roboflow website <https://roboflow.com> and sign in if you have an account, else you need to sign up. It is for free.

After you sign in,



Create new project

Create Project X

Hyunsoo Lee / New Public Project

Project Name

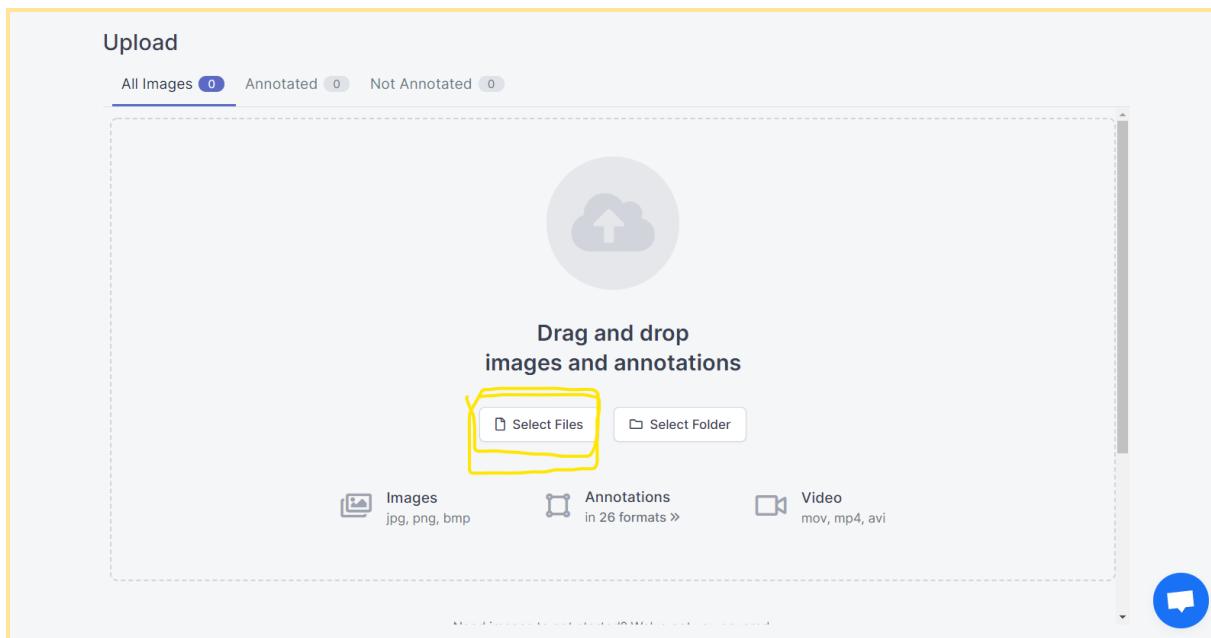
License

Project Type

Annotation Group ?

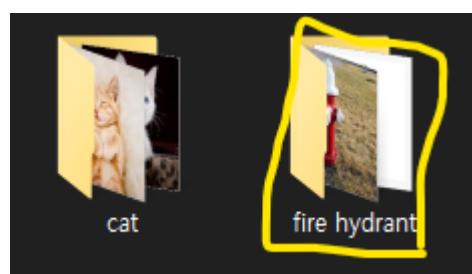
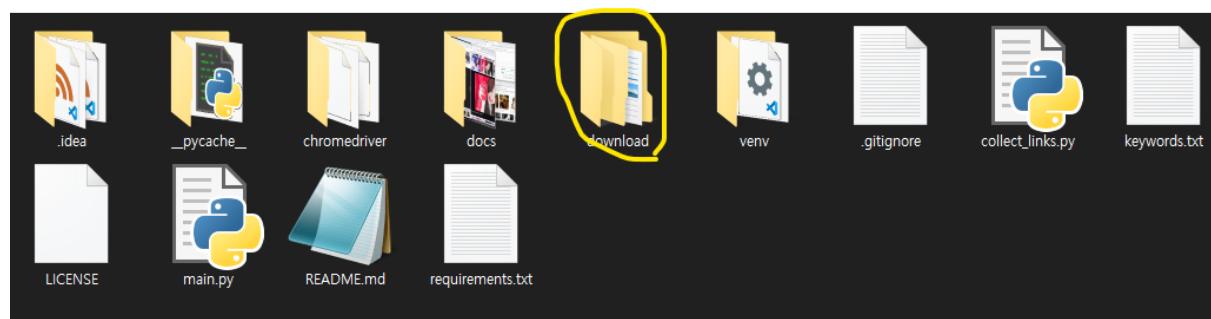
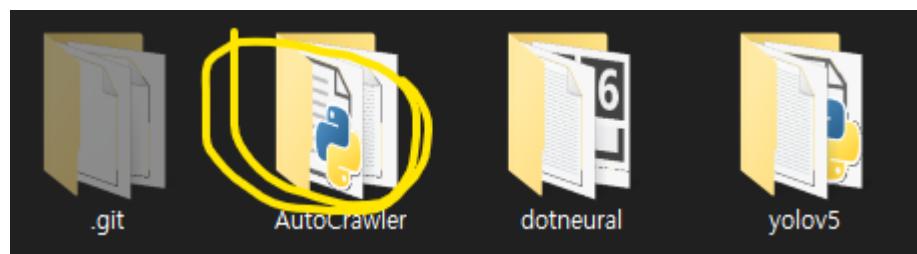
Cancel Create Public Project

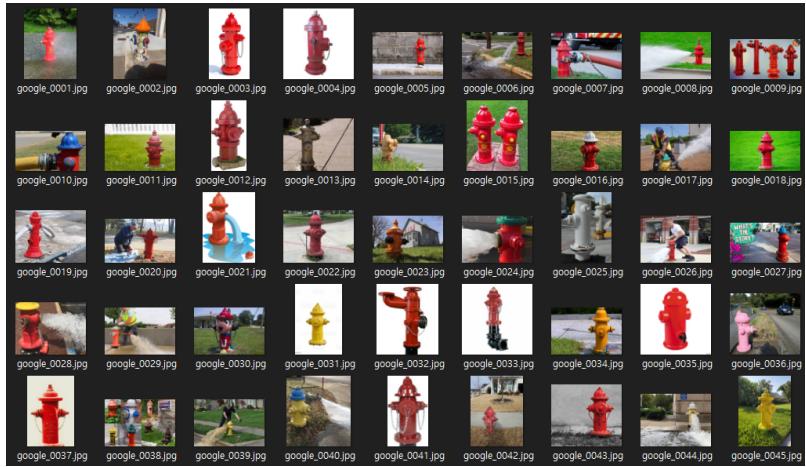
Set project name and annotation group.



Click the “Selected Files” to load our fire hydrant images

Go to “AutoCrawler/download/ WHATEVER YOU DOWNLOADED FOLDER “

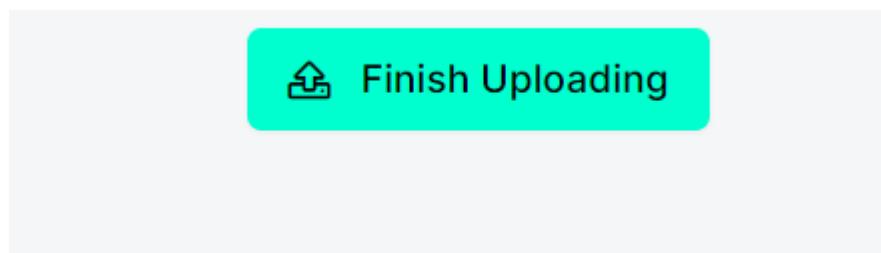




Select downloaded images.

The screenshot shows the Roboflow web interface. At the top, there's a navigation bar with the Roboflow logo, 'Projects', 'Universe', and 'Documentation'. On the far right, it says 'Hyunsoo Lee'. The main content area is titled 'Upload' and shows a grid of 50 images of fire hydrants. Above the grid, there are buttons for 'Select Files' and 'Select Folder'. At the top right of the upload area, there's a green button labeled 'Finish Uploading'.

Now, we uploaded our images on Roboflow.



Click the Finish Uploading

How should we split these images? X

Choose

Split Images Between Train/Valid/Test

Train
85%

Valid
10%

Test
5%



Not sure what this is? [Learn more on our blog.](#)

[Cancel](#)

[Continue](#)

Then, click the continue.

Images

All 50 Images

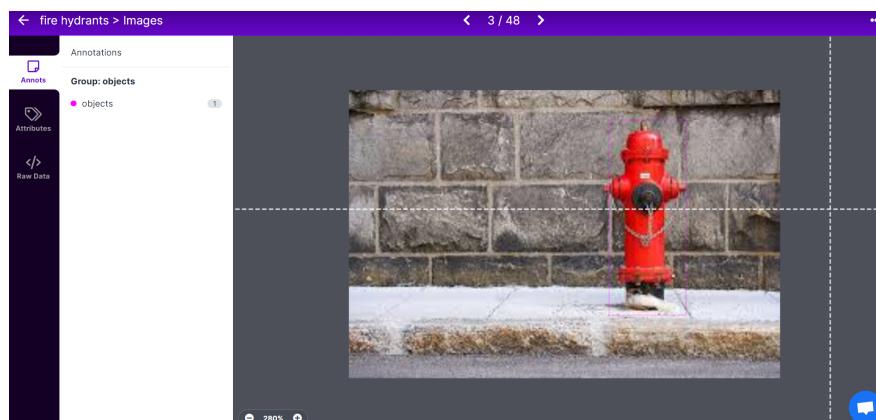
Training Set 42 Validation Set 5 Testing Set 3 Unannotated 48

+ Add Images Generate New Version >

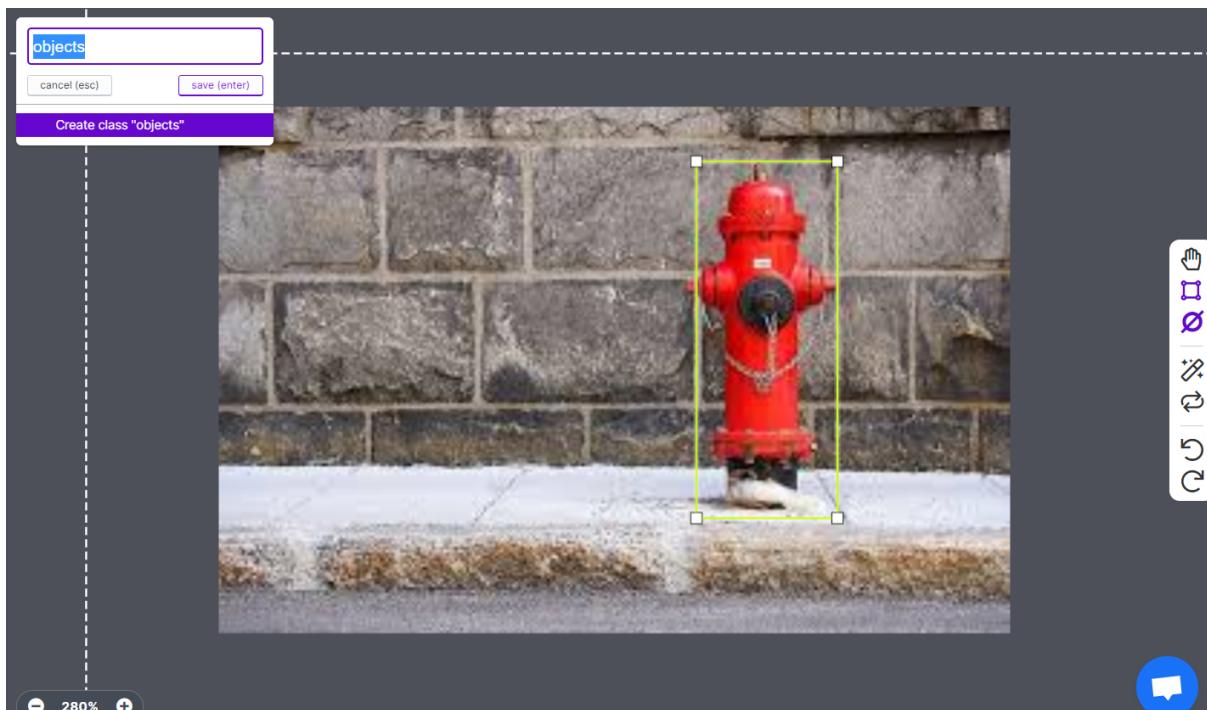
Learn how to label your images along with common best practices in our Roboflow Annotate Guide. [View Guide](#)

A grid of 50 image thumbnails showing various fire hydrants and related scenes. One thumbnail in the top-left corner has a yellow arrow pointing to it from the left side of the screen, indicating which one to click.

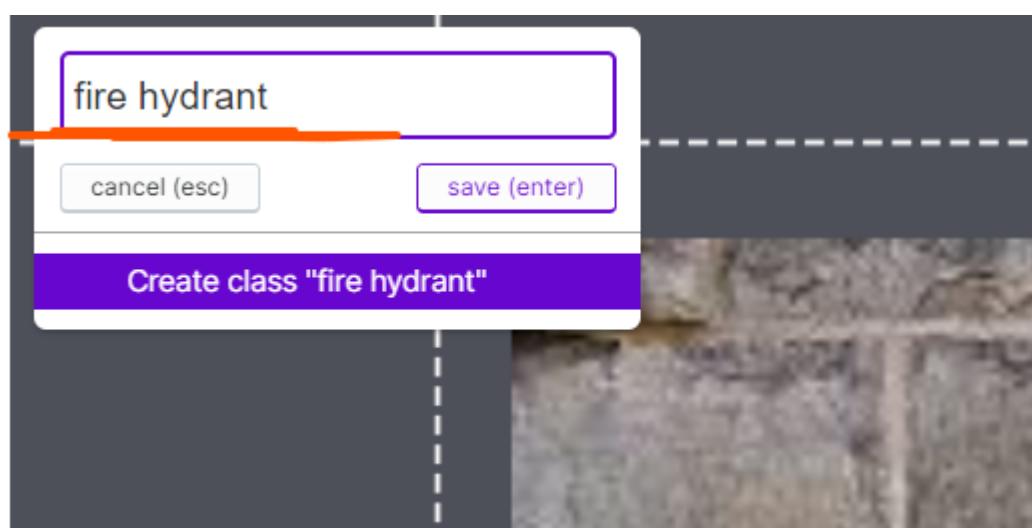
Click the first image



You will see this come up.



Now, drag only the “fire hydrant” part.



Then, save that bounding box with a name “fire hydrant” or whatever you like to call.



 Do the same thing on all

other images. The reason why we do this is because we only need to know what fire hydrants look like, but other parts like grass or cars.

Once you add bounding boxes on all images, there will be nothing on the “Unannotated” tap.

Images

+ Add Images Generate New Version >

12 images uploaded on Oct 0, 2021 11:05pm

Training Set 9 Validation Set 2 Testing Set 1 Unannotated 0

All Images Annotated

All of the images you uploaded have been annotated. Now you're ready to generate a new version of your dataset and train a model.

Generate New Version >

Click the Generate New Version.

The screenshot shows the Roboflow interface for generating a new dataset version. At the top, there's a purple button labeled "Generate New Version". Below it, a sidebar titled "VERSIONS" shows a single entry: "2021-12-07 10:28am v1 Dec 7, 2021". The main area is titled "Generating New Version" with a sub-instruction: "Prepare your images and data for training by compiling them into a version. Experiment with different configurations to ac...".

The process is divided into five steps:

- Source Images**: Shows 50 Images, 1 Class, and 0 Unannotated.
- Train/Test Split**: Shows Training Set: 42 images, Validation Set: 5 images, and Testing Set: 3 images.
- Preprocessing**: Step 3. It lists "Auto-Orient" and "Resize" (Stretch to 416x416), each with an "Edit" button and an "x" to remove. There's also a "+ Add Preprocessing Step" button.
- Augmentation**: Step 4.
- Generate**: Step 5.

A green "Continue" button is located at the bottom of the preprocessing section. A watermark "YOLOv5" is visible in the bottom right corner of the interface.

After you click “Generate new version”, you will see this. Just click continue on step 3, 4 and 5.
If you would like to know more about preprocessing and augmentation, here’s a link.

[▶ YOLOv5 + Roboflow Custom Training Tutorial](#)

fire hydrants Dataset

Generate New Version

VERSIONS

2021-12-07 10:31am v2 Dec 7, 2021

2021-12-07 10:28am v1 Dec 7, 2021

2021-12-07 10:31am Save Name

Version 2 Generated Dec 7, 2021

TRAINING OPTIONS

Use Roboflow Train

Let us train your model and get results within 24 hours along with a hosted API endpoint for making predictions. [Learn More >](#)

Start Training Available Credits: 0

IMAGES

50 images [View All Images >](#)

TRAIN / TEST SPLIT

Training Set 84% 42 images	Validation Set 10% 5 images	Testing Set 6% 3 images
-------------------------------	--------------------------------	----------------------------

PREPROCESSING

Auto-Orient: Applied
Resize: Stretch to 416x416

Export More :

After it's generated, you need to click export.

Export



Format

YOLO v5 PyTorch



TXT annotations and YAML config used with [YOLOv5](#).

download zip to computer show download code

[Cancel](#)

[Continue](#)

Choose the format "YOLO v5 PyTorch" and "show download code"

Your Download Code

X

 Jupyter  Terminal  Raw URL

Paste this snippet into [a notebook from our model library](#) to download and unzip [your dataset](#):

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="XXXXXXXXXXXXXX")
project = rf.workspace().project("fire-hydrants")
dataset = project.version(2).download("yolov5")
```

 **Warning:** Do not share this snippet beyond your team, it contains a private key that is tied to your Roboflow account. Acceptable use policy applies.

Done

Then, you will see such python source code. Copy and paste it somewhere. We need it very soon.

6.2. RETRAIN OUR MODEL ON GOOGLE COLAB

Google offers a good server for free, so we will use Google Colab for retraining.

1. Go to the link below.

[YOLOv5-Custom-Training.ipynb - Colaboratory](#)

The screenshot shows a Google Colab notebook titled "YOLOv5-Custom-Training.ipynb". The interface includes a menu bar with File, Edit, View, Insert, Runtime, Tools, Help, and a toolbar with Share, Connect, Editing, and other options. The main content area displays a section titled "Custom Training with YOLOv5" which describes the steps: Gather a dataset of images and label our dataset, Export our dataset to YOLOv5, Train YOLOv5 to recognize the objects in our dataset, Evaluate our YOLOv5 model's performance, and Run test inference to view our model at work. Below this is a flowchart illustrating the process: Upload (purple), Annotate (purple), Train (blue), and Deploy (blue). A section titled "Step 1: Install Requirements" is expanded, showing a code cell with a play button highlighted by a red box. The code cell contains the following Python code:

```
# Clone YOLOv5 and
# !git clone https://github.com/ultralytics/yolov5 # clone repo
# !pip install -r requirements.txt # install dependencies
import torch
import os
from IPython.display import Image, clear_output # to display images
print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if torch.cuda.is_available() else 'CPU'})")
```

You need to sign in.

2. Run step 1 by clicking the play button.

The screenshot shows the execution of the first code cell in the "Step 1: Install Requirements" section. The play button in the cell header is highlighted with a red box. The output of the cell shows the command being run and its successful completion:

```
# Clone YOLOv5 and
# !git clone https://github.com/ultralytics/yolov5 # clone repo
# !pip install -r requirements.txt # install dependencies
import torch
import os
from IPython.display import Image, clear_output # to display images
print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if torch.cuda.is_available() else 'CPU'})")
```

3. Scroll down and run those two codes as well.

The screenshot shows three code cells in the notebook. Cell [2] imports Roboflow and initializes it with the model format "yolov5" and notebook "ultralytics". Cell [3] sets up the environment by defining the dataset directory. Both cells have green checkmarks indicating they have been successfully run.

```
[2]: from roboflow import Roboflow
rf = Roboflow(model_format="yolov5", notebook="ultralytics")
# upload and label your dataset, and get an API KEY here: https://app.roboflow.com/?model=yolov5&ref=ultralytics
[3]: # set up environment
os.environ["DATASET_DIRECTORY"] = "/content/datasets"
```

4. The next code will be something like below. This is where you need to paste the python code which you got from Roboflow.

```
[4] #after following the link above, receive python code with these fields filled in
# from roboflow import Roboflow
# rf = Roboflow(api_key="YOUR API KEY HERE")
# project = rf.workspace().project("YOUR PROJECT")
# dataset = project.version("YOUR VERSION").download("yolov5")
```

Your Download Code



Jupyter Terminal Raw URL

Paste this snippet into [a notebook from our model library](#) to download and unzip [your dataset](#):

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="████████████████")
project = rf.workspace().project("fire-hydrants")
dataset = project.version(2).download("yolov5")
```

⚠ Warning: Do not share this snippet beyond your team, it contains a private key that is tied to your Roboflow account. Acceptable use policy applies.

Done

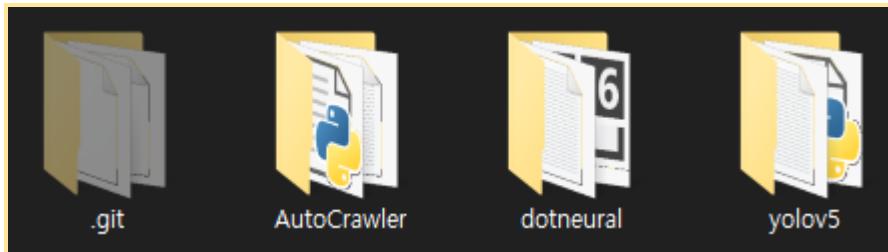
You need to copy and paste this code.

```
 !pip install roboflow

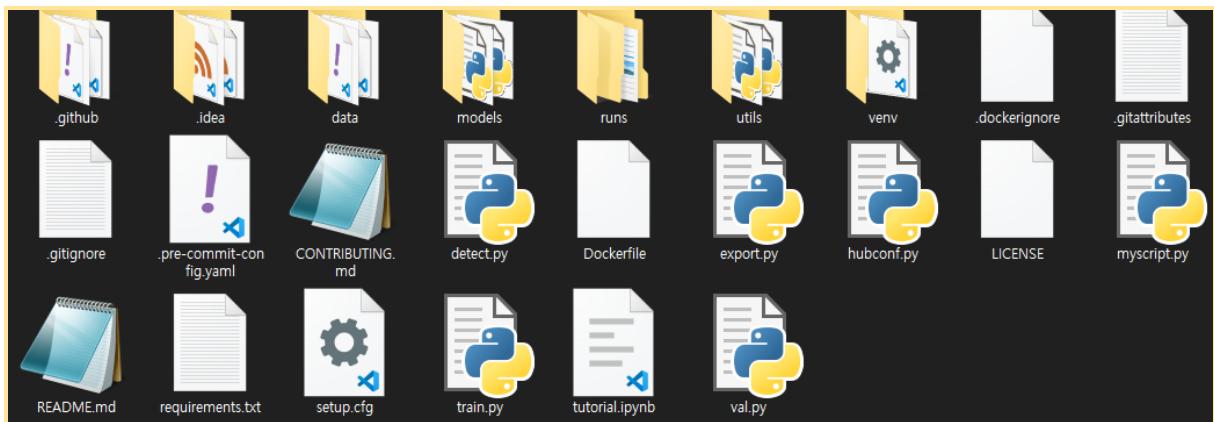
from roboflow import Roboflow
rf = Roboflow(api_key='████████████████')
project = rf.workspace().project("fire-hydrants")
dataset = project.version(2).download("yolov5")
```

Then run this code.

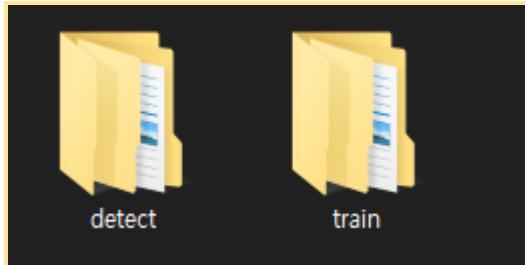
5. Open a folder on your local machine to get our current model data.



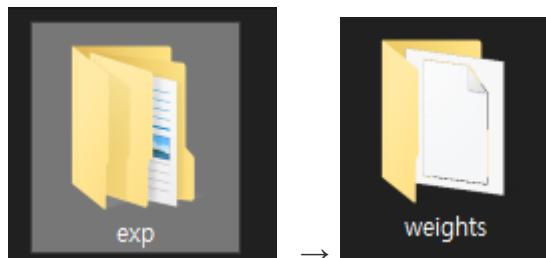
Go to the “yolov5”



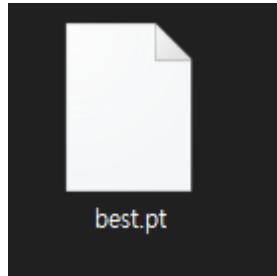
Go to the “runs”



Go to the “train”



Go to the “exp” -> “weights”



You will see the “best.pt” file. This is everything about our model, simply like a memory. We need to update this memory by retraining.

Let's go back to Google Colab.

```
CO YOLOv5-Custom-Training.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes

+ Code + Text Copy to Drive
project = rf.workspace().project("fire-hydrants")
dataset = project.version(2).download("yolov5")

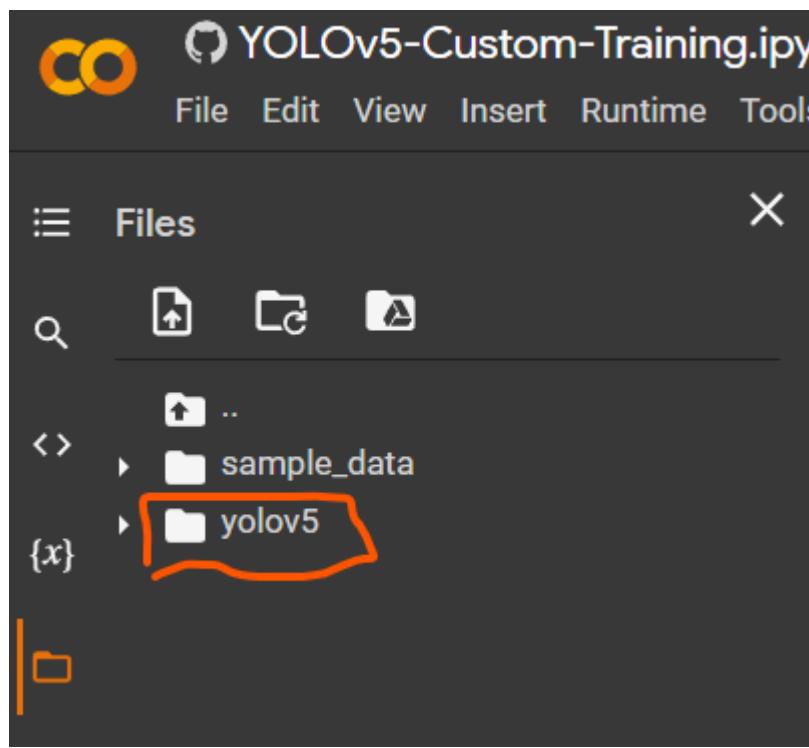
▼ Step 3: Train Our Custom YOLOv5 model

{x}
Here, we are able to pass a number of arguments:
📁 ↗

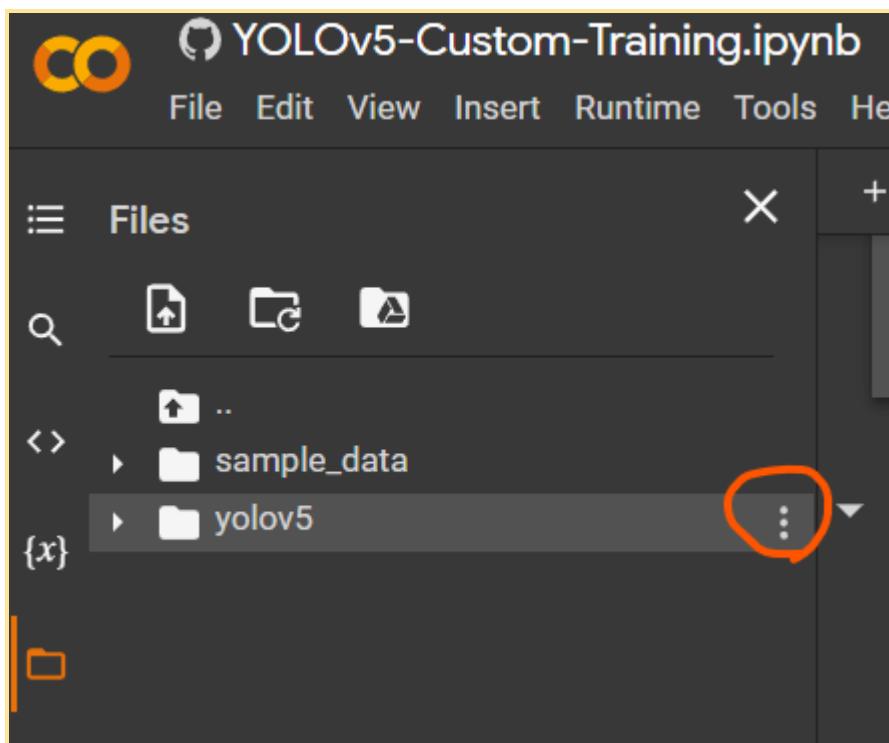

- img: define input image size
- batch: determine batch size
- epochs: define the number of training epochs. (Note: often, 3000+)
- data: Our dataset location is saved in the dataset.location
- weights: specify a path to weights to start transfer learning from. I
- cache: cache images for faster training

```

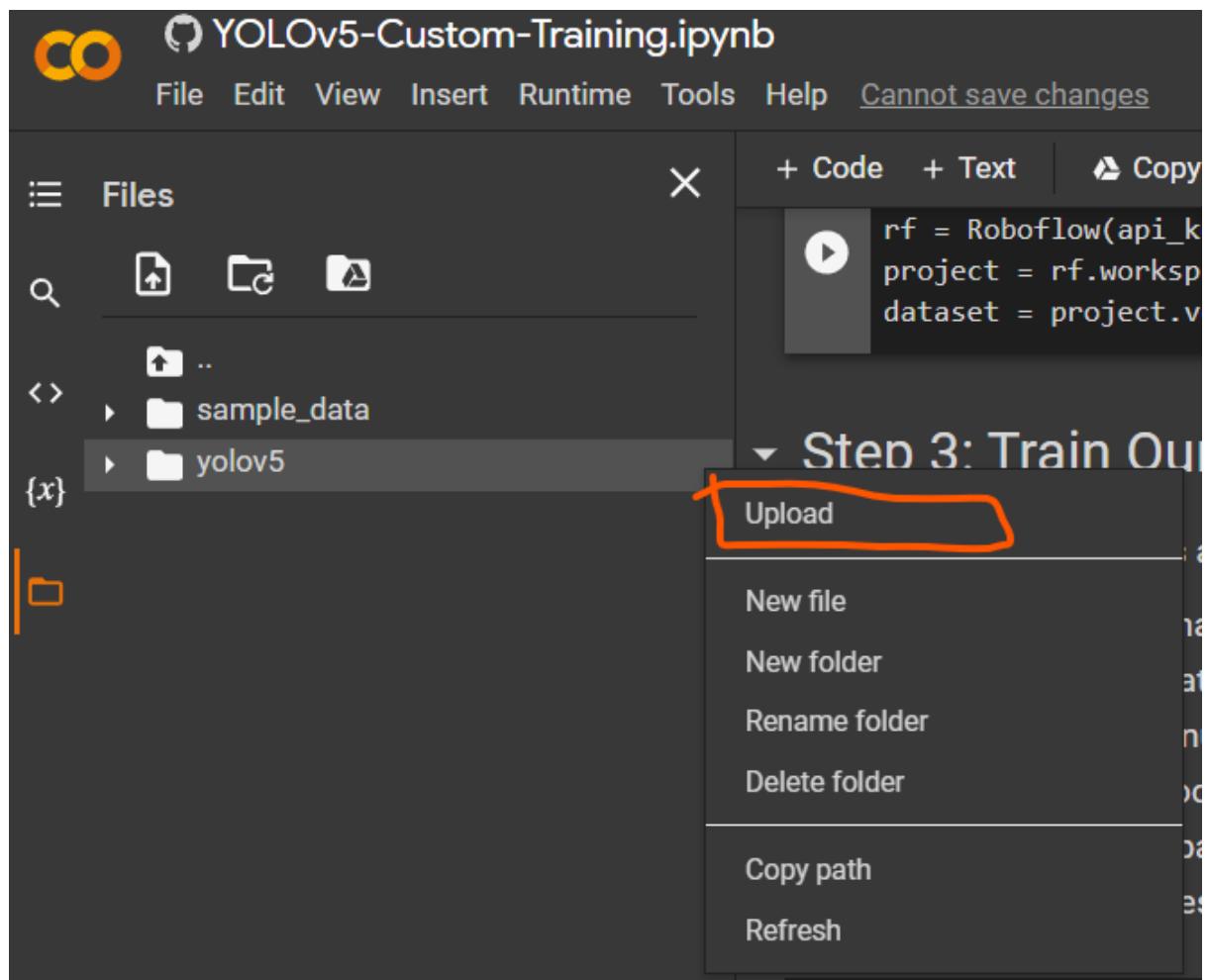
Click the folder icon



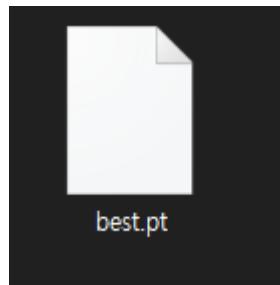
you will see yolov5 folder



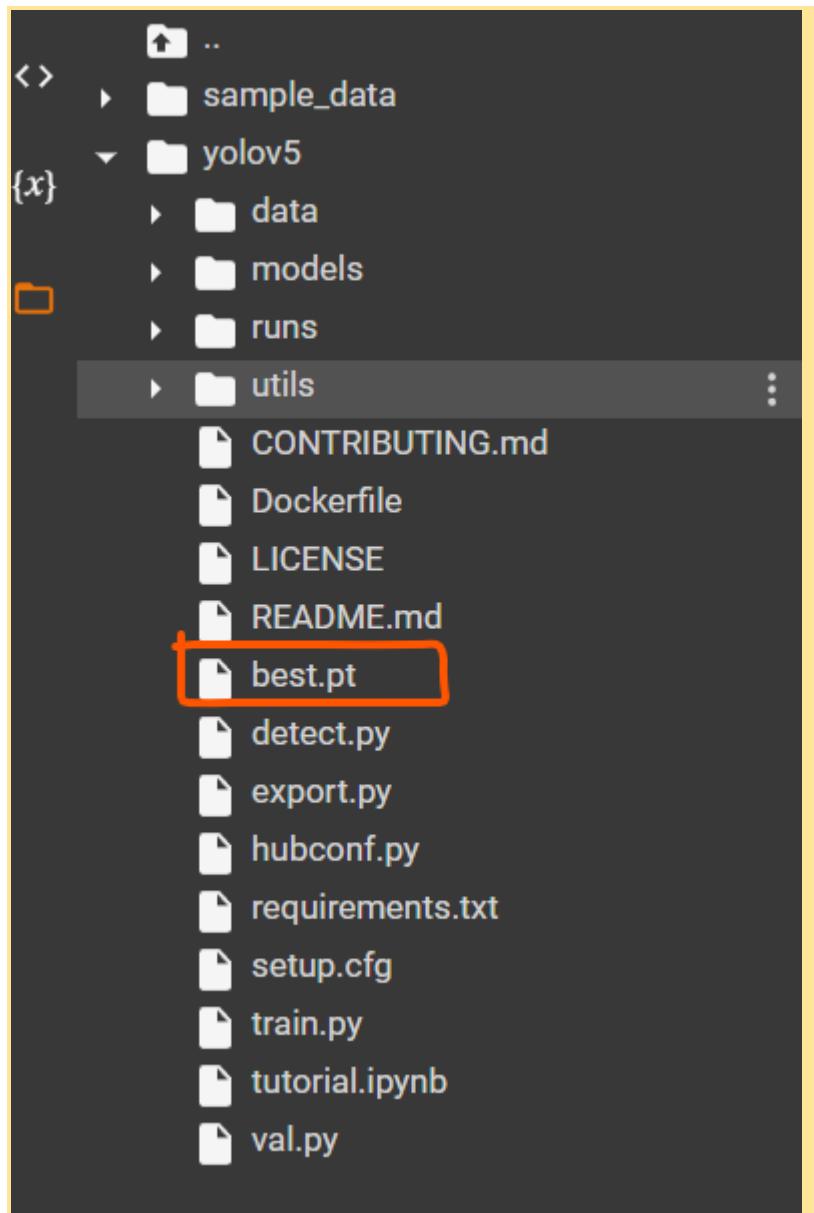
Click the dropdown menu button, orange circle.



Click the Upload.



Upload this "best.pt" file which we found before.



Once you upload it, our

"best.pt" will be inside "yolov5" folder on google colab.

Step 3: Train Our Custom YOLOv5 model

Here, we are able to pass a number of arguments:

- **img**: define input image size
- **batch**: determine batch size
- **epochs**: define the number of training epochs. (Note: often, 3000+ are common here!)
- **data**: Our dataset location is saved in the `dataset.location`
- **weights**: specify a path to weights to start transfer learning from. Here we choose the generic COCO pretrained checkpoint.
- **cache**: cache images for faster training

```
[5] !python train.py --img 416 --batch 16 --epochs 150 --data {dataset.location}/data.yaml --weights yolov5s.pt --cache  
=416, rect=False, resume=False, nosave=False, noval=False, noautoanchor=False, evolve=None, bucket=, cache=ram, image_weights=False, device=, multi_scale
```

Now, we get back to Step 3, on Colab codes. There is one line code, which is by default

```
!python train.py --img 416 --batch 16 --epochs 150  
--data {dataset.location}/data.yaml --weights  
yolov5s.pt --cache
```

There are a couple of things you might need to know.

- “epochs” parameter is the number of passes of the entire training dataset. In this case epoch is 150, which means our model will see the “fire hydrant” dataset 150 times during training. You can reduce this number depending on your situation. The higher the epoch is, the more precise our model can detect that object on street images. You should test with epoch 1 or 2 because it will be done very quickly if the number is low.
- “weights” parameter is a starting point where our model starts to retrain. If we did not set the “weights” parameter, our model will forget previous data. So, we set this parameter with our previous data, which is the “best.pt” we looked for before. Briefly, if you set this parameter with a previous data, it will add new data onto the previous one. If you do not, it will forget all previous data and get trained with only new object data.

We need to change the --weights parameter with our “best.pt” file.

```
!python train.py --img 416 --batch 16 --epochs 150  
--data {dataset.location}/data.yaml --weights best.pt  
--cache
```



```
!python train.py --img 416 --batch 16 --epochs 150 --data {dataset.location}/data.yaml --weights best.pt --cache
```

Then run this code.

```

train: New cache created: /content/datasets/fire-hydrants-2/train/labels.cache
train: Caching images (0.0GB ram): 100% 42/42 [00:00<00:00, 359.18it/s]
... val: Scanning '/content/datasets/fire-hydrants-2/valid/labels' images and labels... 5 found, 0 missing, 1 empty, 0 corrupted: 100% 5/5 [6
val: New cache created: /content/datasets/fire-hydrants-2/valid/labels.cache
val: Caching images (0.0GB ram): 100% 5/5 [00:00<00:00, 114.90it/s]
Plotting labels to runs/train/exp2/labels.jpg...

AutoAnchor: 4.00 anchors/target, 1.000 Best Possible Recall (BPR). Current anchors are a good fit to dataset ✅
Image sizes 416 train, 416 val
Using 2 dataloader workers
Logging results to runs/train/exp2
Starting training for 150 epochs...

Epoch 0/149 gpu_mem      box    obj    cls   labels img_size
1.41G  0.109  0.02194  0.02609      23      416: 100% 3/3 [00:04<00:00, 1.36s/it]
Class   Images     Labels       P       R   mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00, 4.07it/s]
      all      5          4  0.00178      0.25  0.00097  9.7e-05

Epoch 1/149 gpu_mem      box    obj    cls   labels img_size
1.41G  0.1085  0.02311  0.02597      19      416: 100% 3/3 [00:01<00:00, 1.85it/s]
Class   Images     Labels       P       R   mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00, 8.34it/s]
      all      5          4  0.0017      0.25  0.000915  9.15e-05

Epoch 2/149 gpu_mem      box    obj    cls   labels img_size
1.41G  0.111  0.02069  0.02524      19      416: 100% 3/3 [00:01<00:00, 1.86it/s]
Class   Images     Labels       P       R   mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00, 8.82it/s]
      all      5          4  0.00282      0.5   0.00208  0.000208

Epoch 3/149 gpu_mem      box    obj    cls   labels img_size
1.41G  0.1108  0.01926  0.02539      18      416: 100% 3/3 [00:01<00:00, 1.84it/s]
Class   Images     Labels       P       R   mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00, 8.88it/s]
      all      5          4  0.00282      0.5   0.00209  0.000209

Epoch 4/149 gpu_mem      box    obj    cls   labels img_size
1.41G  0.1038  0.02041  0.02427      22      416: 100% 3/3 [00:01<00:00, 1.84it/s]
Class   Images     Labels       P       R   mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00, 9.04it/s]
      all      5          4  0.00177      0.25  0.000927  9.02e-05

```

Training started like this.

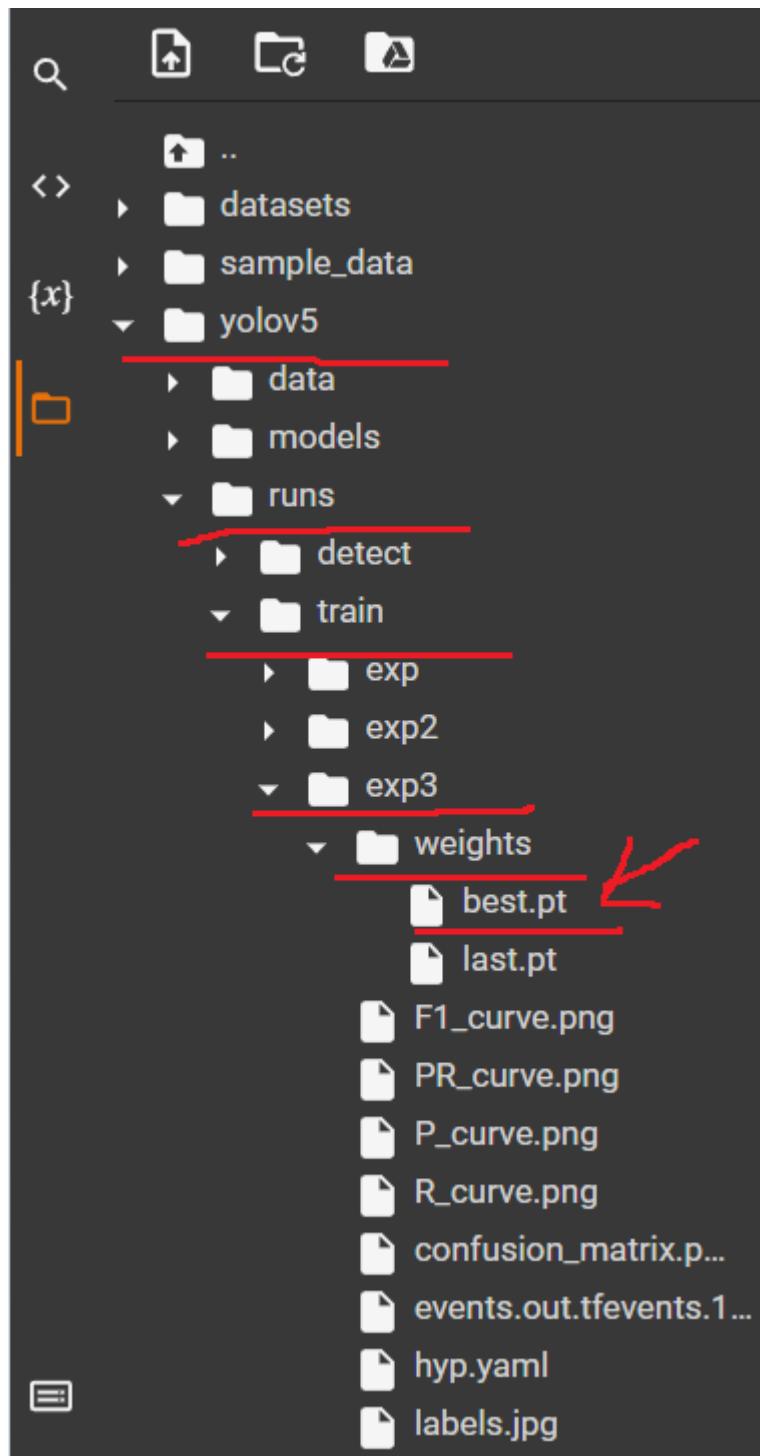
```

1 epochs completed in 0.001 hours.
Optimizer stripped from runs/train/exp3/weights/last.pt, 14.3MB
Optimizer stripped from runs/train/exp3/weights/best.pt, 14.3MB

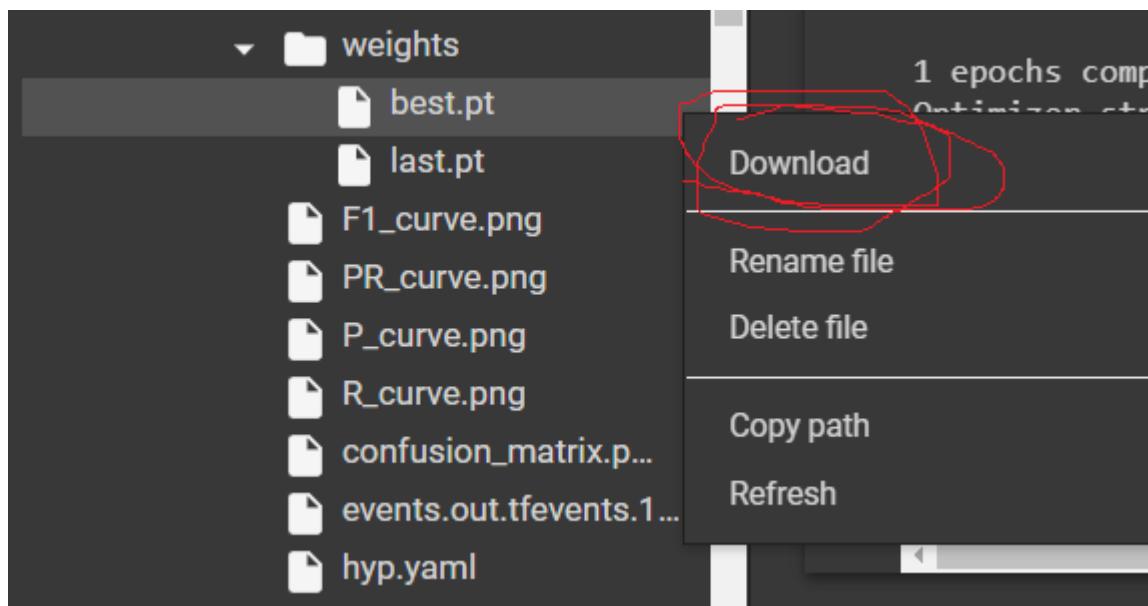
Validating runs/train/exp3/weights/best.pt...
Fusing layers...
Model Summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
      Class   Images     Labels       P       R   mAP@.5 mAP@.5:.95: 100% 1/1 [00:00<00:00, 4.03it/s]
          all      5          4  0.00178      0.25  0.000972  9.72e-05
          fire hydrant 5          4  0.00178      0.25  0.000972  9.72e-05
Results saved to runs/train/exp3

```

After training is done, you will see “Results saved to runs/train/exp3 (this will be different on yours)”.

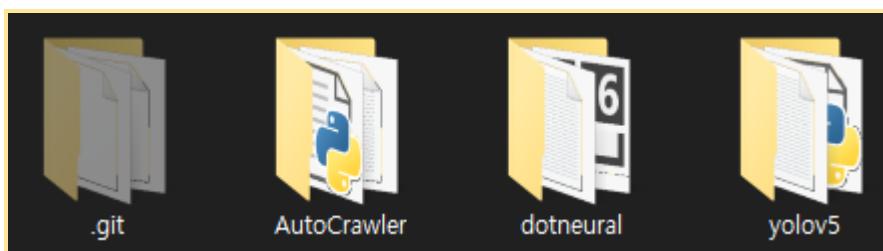


Go to that location, and you will see a “weight” folder and inside there will be an updated new best.pt

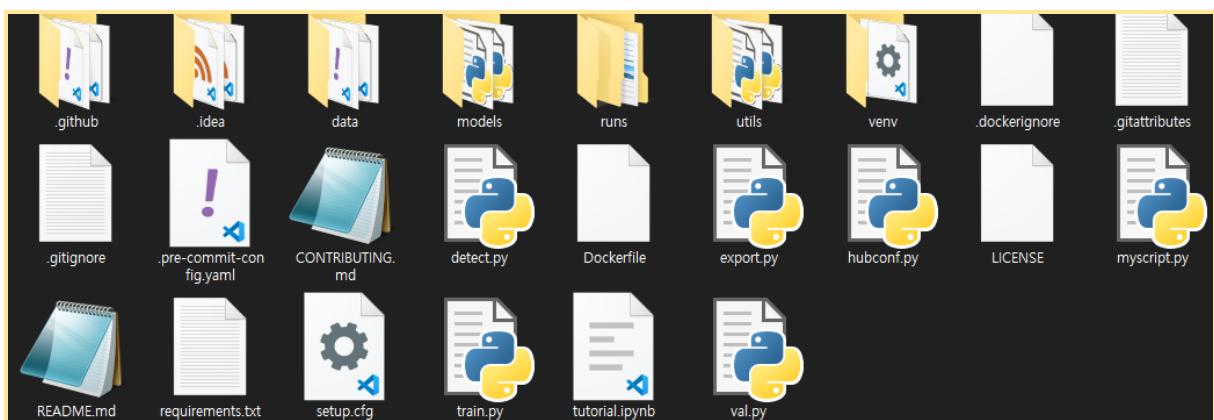


Download the “best.pt”.

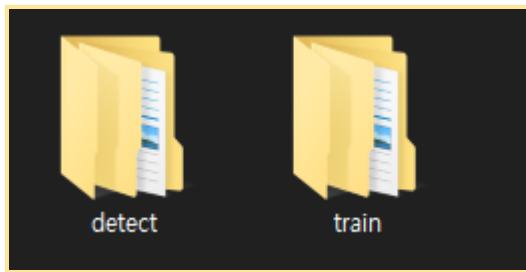
Then, replace a new “best.pt” with your current best.pt on your local machine.



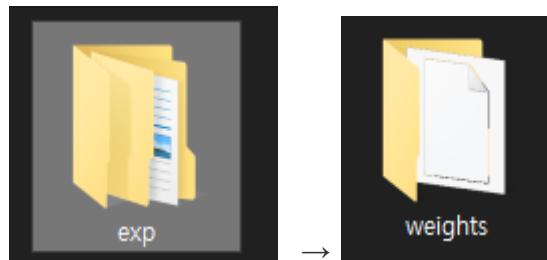
Go to the “yolov5”



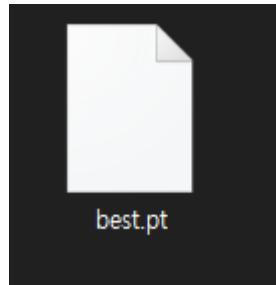
Go to the “runs”



Go to the “train”



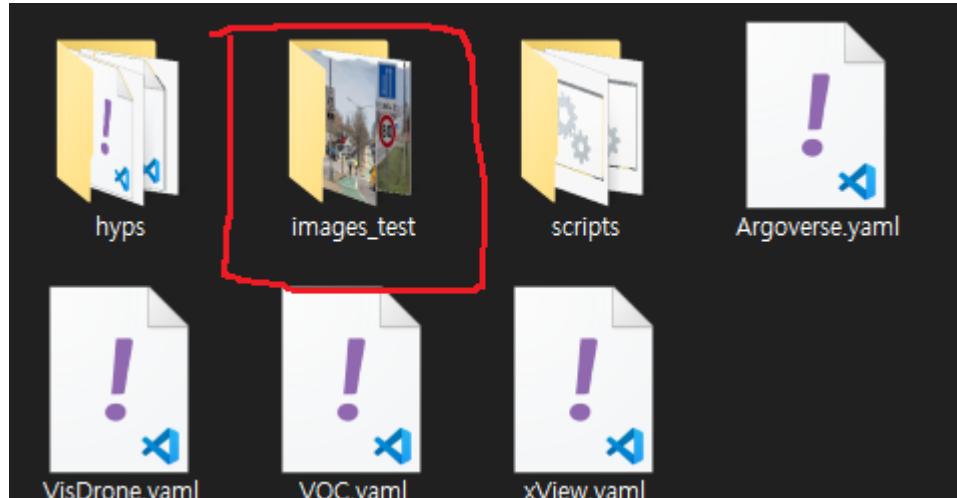
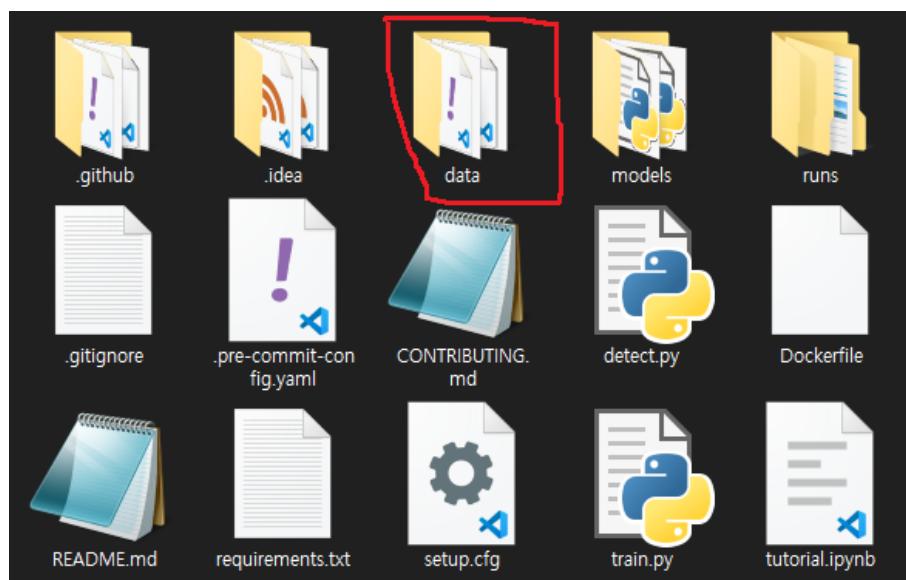
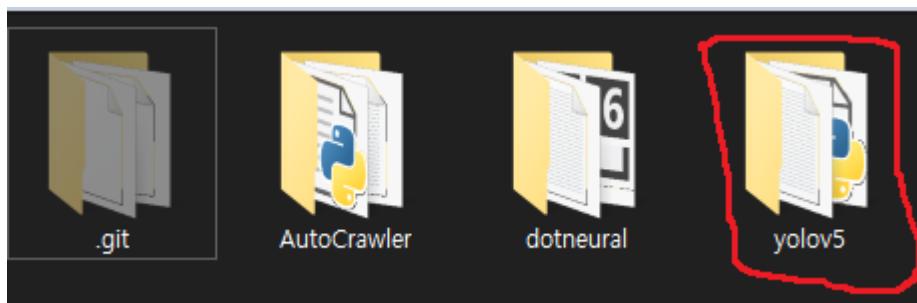
Go to the “exp” -> “weights”



This “best.pt” is now an old one. It must be replaced with the new one, which you download from Google Colab right before step.

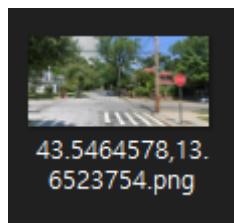
Now, “weights” are updated, which means our model has more knowledge than before.

```
# Street images should be inside \DOTNN\yolov5\data\images_test\
```



\DOTNN\yolov5\data\images_test\

```
# A name of a street image should be formatted like [ longitude, latitude ].png  
EX] -24.43256,62.235235.png
```



Now, we are ready to detect new objects on street images.

Let's click the "detect" button to find objects you wanted to look for in the beginning.

The screenshot shows a software interface titled "RETRAIN". On the left, there are three icons: a house, a magnifying glass over a document, and a gear. The main area has two sections: "Step 1" and "Step 2". In "Step 1", there is a text input field with placeholder text "Type a keyword of an image you like to download" and a dropdown menu set to "50". Below these are two buttons: "DOWNLOAD" and "DETECT". In "Step 2", there is a text input field with placeholder text "Follow our user documentation" and a "DETECT" button. A red rectangular box highlights the "DETECT" button in "Step 2".

you will see a prompt come up and as soon as our model finishes it will disappear.

Let's search "fire hydrant" again, if our model detected it, this time we will see its information.

The screenshot shows a software interface titled "Search". On the left, there are three icons: a house, a magnifying glass over a document, and a gear. The main area has a search bar with the text "fire hydrant" and a "SEARCH" button. Below the search bar is a table with the following data:

No	Object	Latitude	Longitude	Amount
1	fire hydrant	13.432453	73.542356	1

7. HOW TO CHANGE DATABASE SERVER

The image contains two side-by-side screenshots. The left screenshot shows the MongoDB account login page with fields for 'Email Address' and 'Next' button. The right screenshot shows a dark-themed page titled 'Manage Atlas clusters with Kubernetes' featuring a ship's wheel icon and a document icon.

MongoDB.

Log in to your account

Log in with Google

Email Address

Next

Don't have an account? [Sign Up](#)

[Privacy - Terms](#)

Manage Atlas clusters with Kubernetes

The MongoDB Atlas Operator for Kubernetes lets you control your Atlas clusters natively from Kubernetes.

[View Quickstart →](#)

1. Go to the MongoDB website and log in.

<https://account.mongodb.com/account/login?signedOut=true>

2. Create a free-tier server. [Getting Your Free MongoDB Atlas Cluster](#)

<https://www.youtube.com/watch?v=rPqRyYJmx2g>

3. Click “Connect”.

The screenshot shows the 'Database Deployments' page. On the left, there's a sidebar with sections for Deployment, Data Services (Triggers, Data API with a 'PREVIEW' button), and Security (Database Access, Network Access, Advanced). The main area displays a deployment named 'qple-core' with a green status indicator. A search bar at the top says 'Find a database deployment...'. Below the deployment name are two buttons: 'Connect' (which is highlighted with a red box) and 'View Monitoring'. Underneath, there are two sets of metrics: 'R 0' and 'W 0' (Last 6 hours: 0.03/s) and 'Connections 5.0' (Last 6 hours: 14.0). At the bottom of the main area are three tabs: VERSION, REGION, and CLUSTER TIER.

4. Click “Connect you application”

Connect to qple-core

✓ **Setup connection security** > Choose a connection method > Connect

Choose a connection method [View documentation](#)

Get your pre-formatted connection string by selecting your tool below.



Connect with the MongoDB Shell

Interact with your cluster using MongoDB's interactive Javascript interface



Connect your application

Connect your application to your cluster using MongoDB's native drivers



Connect using MongoDB Compass

Explore, modify, and visualize your data with MongoDB's GUI



[Go Back](#)

[Close](#)

5. Change “Driver” to “Python”, and “Version” to “3.6 or Later”.

Connect to qple-core

✓ Setup connection security ✓ Choose a connection method Connect

1 Select your driver and version

DRIVER	VERSION
Python	3.6 or later

2 Add your connection string into your application code

Include full driver code example

```
mongodb+srv://<username>:<password>@qple-core.jxnch.mongodb.net/myFirstDatabase?  
retryWrites=true&w=majority
```

Replace <password> with the password for the <username> user. Replace myFirstDatabase with the name of the database that connections will use by default. Ensure any option params are URL encoded.

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back Close

6. Change <username> and <password> to your username and password

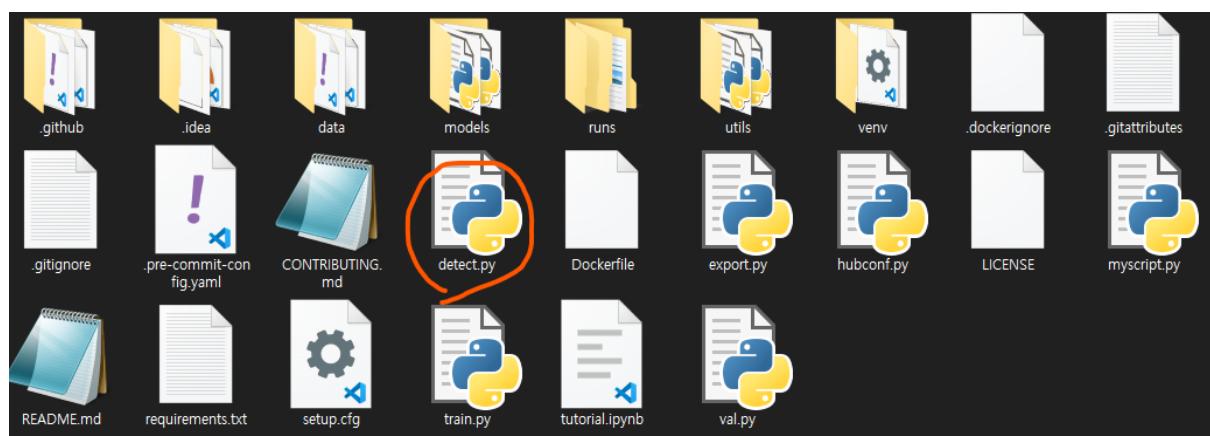
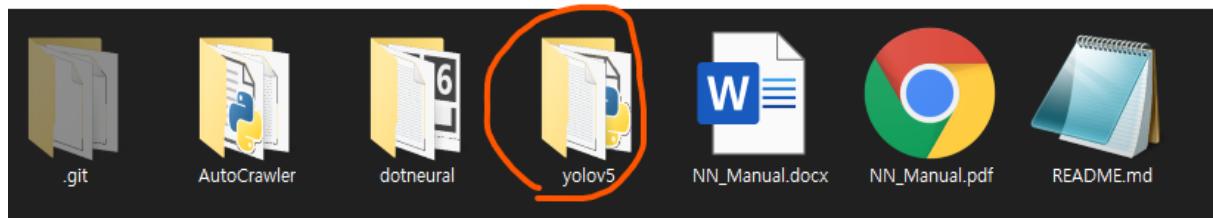
```
mongodb+srv://<username>:<password>@qple-core.jxnch.mongodb.net/myFirstDatabase?  
retryWrites=true&w=majority
```

Let's say your username is **jack77** and password is **abcd77**.

your connection string should be below.

```
mongodb+srv://jack77:abcd77@YOUR-SERVER.jxnch.mongodb.net/myFirstDatabase?  
retryWrites=true&w=majority
```

7. Go to the yolov5 directory and open detect.py file.



8. Go down to line 43.

```
# MongoDB Connection
from pymongo import MongoClient
from collections import defaultdict
client=
MongoClient('mongodb+srv://developer1:hyunsoo97@qple-co
re.jxnch.mongodb.net/myFirstDatabase?retryWrites=true&w
=majority')
db = client['Lemmefind']
obj = db['Objects']
# END
```

9. Change line 43 with connection string in step 7.6

```
# MongoDB Connection
from pymongo import MongoClient
from collections import defaultdict
client=
MongoClient('mongodb+srv://jack77:abcd77@YOUR-SERVER.jxnch.m
ongodb.net/myFirstDatabase?retryWrites=true&w=majority')
db = client['Lemmefind']
obj = db['Objects']
# END
```

8. CONCLUSION

Thank you for following this long document with me.