**Appendix**

1. **Detect object using YOLOv5 and SKU110K Dataset**
   1. **Train Using 640 Image Size**

# \*\*This colab file shows how to detect object using YOLOv5 and SKU110K dataset\*\*

Code Reference links:

1. https://github.com/ultralytics/yolov5

2. https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=7mGmQbAO5pQb

3. https://www.youtube.com/watch?v=2nR2e4J4ZaI&t=1025s

4. https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

# To Setup

Clone repo, install dependencies and check PyTorch and GPU.

# To Use the Dataset

!git clone https://github.com/ultralytics/yolov5  # clone repo

%cd yolov5

%pip install -qr requirements.txt  # install dependencies

!/content/yolov5/weights/download\_weights.sh

import torch

from IPython.display import Image, clear\_output  # to display images

clear\_output()

print(f"Setup complete. Using torch {torch.\_\_version\_\_} ({torch.cuda.get\_device\_properties(0).name if torch.cuda.is\_available() else 'CPU'})")

\* ### There are three ways to use the dataset:

1. Download dataset in your local PC and upload dataset directly on colab notebook by clicking on `Files icon in left panel-->right click and upload--->choose dataset path-->click open`. So it will be uploaded in colab notebook. But keep in mind when notebook is connected, you need to upload every time again.

2. Download dataset using `!wget` command and unzip it using `!unzip` command. In this option, you need to run the command every time once notebook is disconnected, so it will take long time to download and unzip the dataset, it depends on datset size.

3. The best way to use the large datset is to save the dataset in your Google drive and use directly after mounting google drive.

# Method 1 is directly upload in colab as explained in above point 1

# Method 2

# To download and unzip the SKU110K dataset

#!wget http://trax-geometry.s3.amazonaws.com/cvpr\_challenge/SKU110K\_fixed.tar.gz # To download the dataset

#!tar -xvf /content/SKU110K\_fixed.tar.gz #To untar the dataset

#!unzip SKU110K\_fixed.zip # to unzip the datset

# For 20 Epochs and 16 Batch Size

# Method 3

# To mount Google drive to use SKU110K dataset

from google.colab import drive

drive.mount('/content/gdrive/')

## 1. Train

### Train command represents following parameters:

>\*   \*\*`--img`\*\*: Define input image size. The original image size is 1024 × 1024, compress to smaller size make the training process faster.

>\*   \*\*`--batch`\*\*: Determine the batch size. For example, if the batch size is 16 and the training set contains 8219 images, so the number of batches will be 8219 ÷ 16 = 513.

>\*   \*\*`--epochs`\*\*: Define the number of training epochs. An epoch is responsible for learning all input images, in other words, training all input.  The number of epochs represents the number of times the model trains all the inputs and updates the weights to get closer to the ground truth labels.

>\*   \*\*`--data`\*\*: The path to data.yaml file containing the summary of the dataset. The model evaluation process is executed immediately after each epoch, so the model will also access the validation directory via the path in data.yaml file and use its contents for

evaluation at that moment.

>\*   \*\*`--weights`\*\*: Specify a path to weights. A pretrained weight can be used for saving training time. If it is left blank or not given, the model will automatically initialize random weights for training.

>\*   \*\*`--project`\*\*: Name of result folder. The model will store all the results performed during training in a directory. If it is not given, than all training results are saved to runs/train/ with incrementing run directories, i.e. runs/train/exp2, runs/train/exp3 etc.

>\*   \*\*`--cache`\*\*: Cache images for training faster.

### This part shows training using YOLOv5 with Custom Dataset (SKU110K) with 20 epochs, 640 image size and 16 batch size.

1. In the {x}.yaml file, give all the path of train, test and validation folders from the dataset.

2. Give {x}.yaml file path as `--data` argument in following command.

3. Give your results folder's path as `--project` argument, where you want to save your results.

## 2. Validate

# To train using YOLOv5 and SKU110 dataset with 20 epochs, 640 image size and 16 batch size

!python train.py --img 640 --batch 16 --epochs 20 --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K\_results/640 --cache

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 640 image size on SKU110K dataset using following command line.

\* All validation results are saved to runs/val/ with incrementing run directories, i.e. runs/val/exp2, runs/val/exp3 etc., if `--project` argument is not given.

# To validate model's accuracy using 640 image size on SKU110K

!python val.py --task test --batch-size 16 --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K\_results/640

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 416 image size on SKU110K dataset using following command line.

# To validate model's accuracy using 416 image size on SKU110K

!python val.py --task test --batch-size 16 --img 416 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K\_results/test\_inferences/416

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 1024 image size on SKU110K dataset using following command line.

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 2048 image size on SKU110K dataset using following command line.

# To validate model's accuracy using 1024 image size on SKU110K

!python val.py --task test --batch-size 16 --img 1024 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K\_results/test\_inferences/1024

## 3. Detect

# To validate model's accuracy using 2048 image size on SKU110K

!python val.py --task test --batch-size 16 --img 2048 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K\_results/test\_inferences/2048

\* Use following command to detect the objects from images. You can set your detection images path using `--source` argument.

\* All detecting results are saved to runs/detect/ with incrementing run directories, i.e. runs/detect/exp2, runs/detect/exp3 etc., if `--project` argument is not given.

# To detect objects from given source path's images

!python detect.py  --save-txt --save-conf --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-best.pt --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K\_results/640 --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images/test\_0.jpg

1. **Detect object using YOLOv5 and SKU110K-6799 Dataset**
   1. **Train Using 640 Image Size**

# \*\*This colab file shows how to detect object using YOLOv5 and Subset of SKU110K dataset, named as SKU110k\_6799 dataset\*\*

Code Reference links:

1. https://github.com/ultralytics/yolov5

2. https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=7mGmQbAO5pQb

3. https://www.youtube.com/watch?v=2nR2e4J4ZaI&t=1025s

4. https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

# To Setup

Clone repo, install dependencies and check PyTorch and GPU.

!git clone https://github.com/ultralytics/yolov5  # clone repo

%cd yolov5

%pip install -qr requirements.txt  # install dependencies

!/content/yolov5/weights/download\_weights.sh

import torch

from IPython.display import Image, clear\_output  # to display images

clear\_output()

print(f"Setup complete. Using torch {torch.\_\_version\_\_} ({torch.cuda.get\_device\_properties(0).name if torch.cuda.is\_available() else 'CPU'})")

# To Use the Dataset

\* ### There are three ways to use the dataset:

1. Download dataset in your local PC and upload dataset directly on colab notebook by clicking on `Files icon in left panel-->right click and upload--->choose dataset path-->click open`. So it will be uploaded in colab notebook. But keep in mind when notebook is connected, you need to upload every time again.

2. Download dataset using `!wget` command and unzip it using `!unzip` command. In this option, you need to run the command every time once notebook is disconnected, so it will take long time to download and unzip the dataset, it depends on datset size.

3. The best way to use the large datset is to save the dataset in your Google drive and use directly after mounting google drive.

# Method 3

# To mount Google drive to use SKU110K-6799 dataset

from google.colab import drive

drive.mount('/content/gdrive/')

# Method 1 is directly upload in colab as explained in above point 1

# Method 2

# To download and unzip the SKU110K-6799 dataset

#!wget {URL link to download dataset} # To download the dataset

#!tar -xvf {path to untar the dataset} #To untar the dataset

#!unzip {path to unzip the dataset} # to unzip the datset

# For 20 Epochs and 16 Batch Size

## 1. Train

### Train command represents following parameters:

>\*   \*\*`--img`\*\*: Define input image size. The original image size is 1024 × 1024, compress to smaller size make the training process faster.

>\*   \*\*`--batch`\*\*: Determine the batch size. For example, if the batch size is 16 and the training set contains 8219 images, so the number of batches will be 8219 ÷ 16 = 513.

>\*   \*\*`--epochs`\*\*: Define the number of training epochs. An epoch is responsible for learning all input images, in other words, training all input.  The number of epochs represents the number of times the model trains all the inputs and updates the weights to get closer to the ground truth labels.

>\*   \*\*`--data`\*\*: The path to data.yaml file containing the summary of the dataset. The model evaluation process is executed immediately after each epoch, so the model will also access the validation directory via the path in data.yaml file and use its contents for

evaluation at that moment.

>\*   \*\*`--weights`\*\*: Specify a path to weights. A pretrained weight can be used for saving training time. If it is left blank or not given, the model will automatically initialize random weights for training.

>\*   \*\*`--project`\*\*: Name of result folder. The model will store all the results performed during training in a directory. If it is not given, than all training results are saved to runs/train/ with incrementing run directories, i.e. runs/train/exp2, runs/train/exp3 etc.

>\*   \*\*`--cache`\*\*: Cache images for training faster.

### This part shows training using YOLOv5 with Custom Dataset (SKU110K-6799) with 20 epochs, 640 image size and 16 batch size.

1. In the {x}.yaml file, give all the path of train, test and validation folders from the dataset.

2. Give {x}.yaml file path as `--data` argument in following command.

3. Give your results folder's path as `--project` argument, where you want to save your results.

## 2. Validate

# To train using YOLOv5 and dataset SKU110K-6799 with 20 epochs, 640 image size and 16 batch size

!python train.py --img 640 --batch 16 --epochs 20 --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_results/640\_20

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 640 image size on SKU110K-6799 dataset using following command line.

\* All validation results are saved to runs/val/ with incrementing run directories, i.e. runs/val/exp2, runs/val/exp3 etc., if `--project` argument is not given.

# To validate model's accuracy using 640 image size on SKU110K-6799

!python val.py --task test --batch-size 16 --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_20\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_results/640\_20

## 3. Detect

\* Use following command to detect the objects from images. You can set your detection images path using `--source` argument.

\* All detecting results are saved to runs/detect/ with incrementing run directories, i.e. runs/detect/exp2, runs/detect/exp3 etc., if `--project` argument is not given.

# To detect objects from given source path's images, here one image path is set to detect objects from the image

!python detect.py --save-txt --save-conf --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_20\_best.pt --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_results/640\_20 --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images/test\_0.jpg

# To detect objects from given source path's images, here whole folder path is set to detect the objects from the images

!python detect.py --save-txt --save-conf --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_20\_best.pt --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images

# For 50 Epochs and 16 Batch Size

## 1. Train

### This part shows training using YOLOv5 with Custom Dataset (SKU110K-6799) with 50 epochs, 640 image size and 16 batch size.

1. In the {x}.yaml file, give all the path of train, test and validation folders from the dataset.

2. Give {x}.yaml file path as `--data` argument in following command.

3. Give your results folder's path as `--project` argument, where you want to save your results.

## 2. Validate

# To train using YOLOv5 and SKU110-6799 dataset with 50 epochs, 640 image size and 16 batch size

!python train.py --img 640 --batch 16 --epochs 50 --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_results/640\_50/640 --cache

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 416 image size on SKU110K-6799 dataset using following command line.

\* All validation results are saved to runs/val/ with incrementing run directories, i.e. runs/val/exp2, runs/val/exp3 etc., if `--project` argument is not given.

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 640 image size on SKU110K-6799 dataset using following command line.

# To validate model's accuracy using 416 image size on SKU110K-6799

!python val.py --task test --batch-size 16 --img 416 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_50\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_results/640\_50/416

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 1024 image size on SKU110K-6799 dataset using following command line.

# To validate model's accuracy using 640 image size on SKU110K-6799

!python val.py --task test --batch-size 16 --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_50\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_results/640\_50/640

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 2048 image size on SKU110K-6799 dataset using following command line.

# To validate model's accuracy using 1024 image size on SKU110K-6799

!python val.py --task test --batch-size 16 --img 1024 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_50\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_results/640\_50/1024

# To validate model's accuracy using 2048 image size on SKU110K-6799

!python val.py --task test --batch-size 16 --img 2048 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_50\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_results/640\_50/2048

## 3. Detect

\* Use following command to detect the objects from images. You can set your detection images path using `--source` argument.

\* All detecting results are saved to runs/detect/ with incrementing run directories, i.e. runs/detect/exp2, runs/detect/exp3 etc., if `--project` argument is not given.

# To detect objects from given source path's images

!python detect.py --save-txt --save-conf --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_50\_best.pt --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_results/640\_50/640 --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images/test\_0.jpg

1. **Detect object using YOLOv5 and SKU110K-6799\_good\_images Dataset**
   1. **Train Using 416 Image Size**

# \*\*This colab file shows how to detect object using YOLOv5 and SKU110K-6799\_good\_images dataset (which is made after ignoring the corrupted images from subset of SKU110K dataset (SKU110k\_6799))\*\*

Code Reference links:

1. https://github.com/ultralytics/yolov5

2. https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=7mGmQbAO5pQb

3. https://www.youtube.com/watch?v=2nR2e4J4ZaI&t=1025s

4. https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

# To Setup

Clone repo, install dependencies and check PyTorch and GPU.

# To Use the Dataset

!git clone https://github.com/ultralytics/yolov5  # clone repo

%cd yolov5

%pip install -qr requirements.txt  # install dependencies

!/content/yolov5/weights/download\_weights.sh

import torch

from IPython.display import Image, clear\_output  # to display images

clear\_output()

print(f"Setup complete. Using torch {torch.\_\_version\_\_} ({torch.cuda.get\_device\_properties(0).name if torch.cuda.is\_available() else 'CPU'})")

\* ### There are three ways to use the dataset:

1. Download dataset in your local PC and upload dataset directly on colab notebook by clicking on `Files icon in left panel-->right click and upload--->choose dataset path-->click open`. So it will be uploaded in colab notebook. But keep in mind when notebook is connected, you need to upload every time again.

2. Download dataset using `!wget` command and unzip it using `!unzip` command. In this option, you need to run the command every time once notebook is disconnected, so it will take long time to download and unzip the dataset, it depends on datset size.

3. The best way to use the large datset is to save the dataset in your Google drive and use directly after mounting google drive.

# Method 1 is directly upload in colab as explained in above point 1

# Method 2

# To download and unzip the SKU110K-6799\_good\_images dataset

#!wget {URL link to download dataset} # To download the dataset

#!tar -xvf {path to untar the dataset} #To untar the dataset

#!unzip {path to unzip the dataset} # to unzip the datset

# For 50 Epochs and 16 Batch Size

# Method 3

# To mount Google drive to use SKU110K-6799\_good\_images dataset

from google.colab import drive

drive.mount('/content/gdrive/')

## 1. Train

### Train command represents following parameters:

>\*   \*\*`--img`\*\*: Define input image size. The original image size is 1024 × 1024, compress to smaller size make the training process faster.

>\*   \*\*`--batch`\*\*: Determine the batch size. For example, if the batch size is 16 and the training set contains 8219 images, so the number of batches will be 8219 ÷ 16 = 513.

>\*   \*\*`--epochs`\*\*: Define the number of training epochs. An epoch is responsible for learning all input images, in other words, training all input.  The number of epochs represents the number of times the model trains all the inputs and updates the weights to get closer to the ground truth labels.

>\*   \*\*`--data`\*\*: The path to data.yaml file containing the summary of the dataset. The model evaluation process is executed immediately after each epoch, so the model will also access the validation directory via the path in data.yaml file and use its contents for

evaluation at that moment.

>\*   \*\*`--weights`\*\*: Specify a path to weights. A pretrained weight can be used for saving training time. If it is left blank or not given, the model will automatically initialize random weights for training.

>\*   \*\*`--project`\*\*: Name of result folder. The model will store all the results performed during training in a directory. If it is not given, than all training results are saved to runs/train/ with incrementing run directories, i.e. runs/train/exp2, runs/train/exp3 etc.

>\*   \*\*`--cache`\*\*: Cache images for training faster.

### This part shows training using YOLOv5 with Custom Dataset (SKU110K\_6799\_good\_images) with 50 epochs, 416 image size and 16 batch size.

1. In the {x}.yaml file, give all the path of train, test and validation folders from the dataset.

2. Give {x}.yaml file path as `--data` argument in following command.

3. Give your results folder's path as `--project` argument, where you want to save your results.

## 2. Validate

# To train using YOLOv5 and dataset SKU110K-6799\_good\_images with 50 epochs, 416 image size and 16 batch size

!python train.py --img 416 --batch 16 --epochs 50 --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_416\_50/416 --cache

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 416 image size on SKU110K-6799\_good\_images dataset using following command line.

\* All validation results are saved to runs/val/ with incrementing run directories, i.e. runs/val/exp2, runs/val/exp3 etc., if `--project` argument is not given.

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 640 image size on SKU110K-6799\_good\_images dataset using following command line.

# To validate model's accuracy using 416 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 16 --img 416 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_416\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_416\_50/416

# To validate model's accuracy using 640 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 16 --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_416\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_416\_50/640

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 1024 image size on SKU110K-6799\_good\_images dataset using following command line.

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 2048 image size on SKU110K-6799\_good\_images dataset using following command line.

# To validate model's accuracy using 1024 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 16 --img 1024 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_416\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_416\_50/1024

# To validate model's accuracy using 2048 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 16 --img 2048 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_416\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_416\_50/2048

## 3. Detect

\* Use following command to detect the objects from images. You can set your detection images path using `--source` argument.

\* All detecting results are saved to runs/detect/ with incrementing run directories, i.e. runs/detect/exp2, runs/detect/exp3 etc., if `--project` argument is not given.

# To detect objects from given source path's images

!python detect.py --save-txt --save-conf --img 416  --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_416\_50\_good\_images\_best.pt --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_416\_50/416 --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images/test\_0.jpg

# For 50 Epochs and 64 Batch Size

## 1. Train

### This part shows training using YOLOv5 with Custom Dataset (SKU110K-6799\_good\_images) with 50 epochs, 416 image size and 64 batch size.

1. In the {x}.yaml file, give all the path of train, test and validation folders from the dataset.

2. Give {x}.yaml file path as `--data` argument in following command.

3. Give your results folder's path as `--project` argument, where you want to save your results.

# To train using YOLOv5 and dataset SKU110K-6799\_good\_images with 50 epochs, 416 image size and 64 batch size

!python train.py --img 416 --batch 64 --epochs 50 --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_416\_50/416 --cache

## 2. Validate

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 416 image size on SKU110K-6799\_good\_images dataset using following command line.

\* All validation results are saved to runs/val/ with incrementing run directories, i.e. runs/val/exp2, runs/val/exp3 etc., if `--project` argument is not given.

# To validate model's accuracy using 416 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 64 --img 416 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_416\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_416\_50/416

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 640 image size on SKU110K dataset using following command line.

# To validate model's accuracy using 640 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 64 --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_416\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_416\_50/640

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 1024 image size on SKU110K-6799\_good\_images dataset using following command line.

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 2048 image size on SKU110K-6799\_good\_images dataset using following command line.

# To validate model's accuracy using 1024 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 64 --img 1024 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_416\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_416\_50/1024

# To validate model's accuracy using 2048 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 64 --img 2048 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_416\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_416\_50/2048

## 3. Detect

\* Use following command to detect the objects from images. You can set your detection images path using `--source` argument.

\* All detecting results are saved to runs/detect/ with incrementing run directories, i.e. runs/detect/exp2, runs/detect/exp3 etc., if `--project` argument is not given.

# To detect objects from given source path's images

!python detect.py --save-txt --save-conf --img 416 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_416\_50\_good\_images\_best.pt --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_416\_50/416 --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images/test\_0.jpg

* 1. **Train Using 640 Image Size**

# \*\*This colab file shows how to detect object using YOLOv5 and SKU110K-6799\_good\_images dataset (which is made after ignoring the corrupted images from subset of SKU110K dataset (SKU110k\_6799))\*\*

Code Reference links:

1. https://github.com/ultralytics/yolov5

2. https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=7mGmQbAO5pQb

3. https://www.youtube.com/watch?v=2nR2e4J4ZaI&t=1025s

4. https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

# To Setup

Clone repo, install dependencies and check PyTorch and GPU.

!git clone https://github.com/ultralytics/yolov5  # clone repo

%cd yolov5

%pip install -qr requirements.txt  # install dependencies

!/content/yolov5/weights/download\_weights.sh

import torch

from IPython.display import Image, clear\_output  # to display images

clear\_output()

print(f"Setup complete. Using torch {torch.\_\_version\_\_} ({torch.cuda.get\_device\_properties(0).name if torch.cuda.is\_available() else 'CPU'})")

# To Use the Dataset

\* ### There are three ways to use the dataset:

1. Download dataset in your local PC and upload dataset directly on colab notebook by clicking on `Files icon in left panel-->right click and upload--->choose dataset path-->click open`. So it will be uploaded in colab notebook. But keep in mind when notebook is connected, you need to upload every time again.

2. Download dataset using `!wget` command and unzip it using `!unzip` command. In this option, you need to run the command every time once notebook is disconnected, so it will take long time to download and unzip the dataset, it depends on datset size.

3. The best way to use the large datset is to save the dataset in your Google drive and use directly after mounting google drive.

# Method 1 is directly upload in colab as explained in above point 1

# Method 2

# To download and unzip the SKU110K-6799\_good\_images dataset

#!wget {URL link to download dataset} # To download the dataset

#!tar -xvf {path to untar the dataset} #To untar the dataset

#!unzip {path to unzip the dataset} # to unzip the datset

# For 50 Epochs and 16 Batch Size

# Method 3

# To mount Google drive to use SKU110K-6799\_good\_images dataset

from google.colab import drive

drive.mount('/content/gdrive/')

## 1. Train

### Train command represents following parameters:

>\*   \*\*`--img`\*\*: Define input image size. The original image size is 1024 × 1024, compress to smaller size make the training process faster.

>\*   \*\*`--batch`\*\*: Determine the batch size. For example, if the batch size is 16 and the training set contains 8219 images, so the number of batches will be 8219 ÷ 16 = 513.

>\*   \*\*`--epochs`\*\*: Define the number of training epochs. An epoch is responsible for learning all input images, in other words, training all input.  The number of epochs represents the number of times the model trains all the inputs and updates the weights to get closer to the ground truth labels.

>\*   \*\*`--data`\*\*: The path to data.yaml file containing the summary of the dataset. The model evaluation process is executed immediately after each epoch, so the model will also access the validation directory via the path in data.yaml file and use its contents for

evaluation at that moment.

>\*   \*\*`--weights`\*\*: Specify a path to weights. A pretrained weight can be used for saving training time. If it is left blank or not given, the model will automatically initialize random weights for training.

>\*   \*\*`--project`\*\*: Name of result folder. The model will store all the results performed during training in a directory. If it is not given, than all training results are saved to runs/train/ with incrementing run directories, i.e. runs/train/exp2, runs/train/exp3 etc.

>\*   \*\*`--cache`\*\*: Cache images for training faster.

### This part shows training using YOLOv5 with Custom Dataset (SKU110K-6799\_good\_images) with 50 epochs, 640 image size and 16 batch size.

1. In the {x}.yaml file, give all the path of train, test and validation folders from the dataset.

2. Give {x}.yaml file path as `--data` argument in following command.

3. Give your results folder's path as `--project` argument, where you want to save your results.

## 2. Validate

# To train using YOLOv5 and dataset SKU110K-6799\_good\_images with 50 epochs, 640 image size and 16 batch size

!python train.py --img 640 --batch 16 --epochs 50 --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_640\_50/640 --cache

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 416 image size on SKU110K-6799\_good\_images dataset using following command line.

\* All validation results are saved to runs/val/ with incrementing run directories, i.e. runs/val/exp2, runs/val/exp3 etc., if `--project` argument is not given.

# To validate model's accuracy using 416 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 16 --img 416 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_640\_50/416

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 640 image size on SKU110K-6799\_good\_images dataset using following command line.

# To validate model's accuracy using 640 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 16 --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_640\_50

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 1024 image size on SKU110K-6799\_good\_images dataset using following command line.

# To validate model's accuracy using 1024 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 16 --img 1024 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_640\_50/1024

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 2048 image size on SKU110K-6799\_good\_images dataset using following command line.

## 3. Detect

# To validate model's accuracy using 2048 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 16 --img 2048 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_640\_50/2048

\* Use following command to detect the objects from images. You can set your detection images path using `--source` argument.

\* All detecting results are saved to runs/detect/ with incrementing run directories, i.e. runs/detect/exp2, runs/detect/exp3 etc., if `--project` argument is not given.

# For 50 Epochs and 64 Batch Size

# To detect objects from given source path's images

!python detect.py --save-txt --save-conf --img 640  --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_50\_good\_images\_best.pt  --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_640\_50/640 --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images/test\_0.jpg

## 1. Train

### This part shows training using YOLOv5 with Custom Dataset (SKU110K-6799\_good\_images) with 50 epochs, 640 image size and 64 batch size.

1. In the {x}.yaml file, give all the path of train, test and validation folders from the dataset.

2. Give {x}.yaml file path as `--data` argument in following command.

3. Give your results folder's path as `--project` argument, where you want to save your results.

# To train using YOLOv5 and dataset SKU110K-6799\_good\_images with 50 epochs, 640 image size and 64 batch size

!python train.py --img 640 --batch 64 --epochs 50 --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_640\_50/640 --cache

## 2. Validate

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 416 image size on SKU110K-6799\_good\_images dataset using following command line.

\* All validation results are saved to runs/val/ with incrementing run directories, i.e. runs/val/exp2, runs/val/exp3 etc., if `--project` argument is not given.

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 640 image size on SKU110K-6799\_good\_images dataset using following command line.

# To validate model's accuracy using 416 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 64 --img 416 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_640\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_640\_50/416

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 1024 image size on SKU110K-6799\_good\_images dataset using following command line.

# To validate model's accuracy using 640 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 64 --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_640\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_640\_50/640

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 2048 image size on SKU110K-6799\_good\_images dataset using following command line.

# To validate model's accuracy using 1024 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 64 --img 1024 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_640\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_640\_50/1024

## 3. Detect

# To validate model's accuracy using 2048 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 64 --img 2048 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_640\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_640\_50/2048

\* Use following command to detect the objects from images. You can set your detection images path using `--source` argument.

\* All detecting results are saved to runs/detect/ with incrementing run directories, i.e. runs/detect/exp2, runs/detect/exp3 etc., if `--project` argument is not given.

# To detect objects from given source path's images

!python detect.py --save-txt --save-conf --img 640  --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_640\_50\_good\_images\_best.pt  --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_640\_50 --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images/test\_0.jpg

* 1. **Train Using 1024 Image Size**

# \*\*This colab file shows how to detect object using YOLOv5 and SKU110K-6799\_good\_images dataset (which is made after ignoring the corrupted images from subset of SKU110K dataset (SKU110k\_6799))\*\*

Code Reference links:

1. https://github.com/ultralytics/yolov5

2. https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=7mGmQbAO5pQb

3. https://www.youtube.com/watch?v=2nR2e4J4ZaI&t=1025s

4. https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

# To Setup

Clone repo, install dependencies and check PyTorch and GPU.

# To Use the Dataset

!git clone https://github.com/ultralytics/yolov5  # clone repo

%cd yolov5

%pip install -qr requirements.txt  # install dependencies

!/content/yolov5/weights/download\_weights.sh

import torch

from IPython.display import Image, clear\_output  # to display images

clear\_output()

print(f"Setup complete. Using torch {torch.\_\_version\_\_} ({torch.cuda.get\_device\_properties(0).name if torch.cuda.is\_available() else 'CPU'})")

\* ### There are three ways to use the dataset:

1. Download dataset in your local PC and upload dataset directly on colab notebook by clicking on `Files icon in left panel-->right click and upload--->choose dataset path-->click open`. So it will be uploaded in colab notebook. But keep in mind when notebook is connected, you need to upload every time again.

2. Download dataset using `!wget` command and unzip it using `!unzip` command. In this option, you need to run the command every time once notebook is disconnected, so it will take long time to download and unzip the dataset, it depends on datset size.

3. The best way to use the large datset is to save the dataset in your Google drive and use directly after mounting google drive.

# Method 1 is directly upload in colab as explained in above point 1

# Method 2

# To download and unzip the SKU110K-6799\_good\_images dataset

#!wget {URL link to download dataset} # To download the dataset

#!tar -xvf {path to untar the dataset} #To untar the dataset

#!unzip {path to unzip the dataset} # to unzip the datset

# For 50 Epochs and 16 Batch Size

# Method 3

# To mount Google drive to use SKU110K-6799\_good\_images dataset

from google.colab import drive

drive.mount('/content/gdrive/')

## 1. Train

### Train command represents following parameters:

>\*   \*\*`--img`\*\*: Define input image size. The original image size is 1024 × 1024, compress to smaller size make the training process faster.

>\*   \*\*`--batch`\*\*: Determine the batch size. For example, if the batch size is 16 and the training set contains 8219 images, so the number of batches will be 8219 ÷ 16 = 513.

>\*   \*\*`--epochs`\*\*: Define the number of training epochs. An epoch is responsible for learning all input images, in other words, training all input.  The number of epochs represents the number of times the model trains all the inputs and updates the weights to get closer to the ground truth labels.

>\*   \*\*`--data`\*\*: The path to data.yaml file containing the summary of the dataset. The model evaluation process is executed immediately after each epoch, so the model will also access the validation directory via the path in data.yaml file and use its contents for

evaluation at that moment.

>\*   \*\*`--weights`\*\*: Specify a path to weights. A pretrained weight can be used for saving training time. If it is left blank or not given, the model will automatically initialize random weights for training.

>\*   \*\*`--project`\*\*: Name of result folder. The model will store all the results performed during training in a directory. If it is not given, than all training results are saved to runs/train/ with incrementing run directories, i.e. runs/train/exp2, runs/train/exp3 etc.

>\*   \*\*`--cache`\*\*: Cache images for training faster.

### This part shows training using YOLOv5 with Custom Dataset (SKU110K-6799\_good\_images) with 50 epochs, 1024 image size and 16 batch size.

1. In the {x}.yaml file, give all the path of train, test and validation folders from the dataset.

2. Give {x}.yaml file path as `--data` argument in following command.

3. Give your results folder's path as `--project` argument, where you want to save your results.

## 2. Validate

# To train using YOLOv5 and dataset SKU110K-6799\_good\_images with 50 epochs, 1024 image size and 16 batch size

!python train.py --img 1024 --batch 16 --epochs 50 --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_1024\_50/1024 --cache

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 416 image size on SKU110K-6799\_good\_images dataset using following command line.

\* All validation results are saved to runs/val/ with incrementing run directories, i.e. runs/val/exp2, runs/val/exp3 etc., if `--project` argument is not given.

# To validate model's accuracy using 416 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 16 --img 416 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_1024\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_1024\_50/416

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 640 image size on SKU110K-6799\_good\_images dataset using following command line.

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 1024 image size on SKU110K-6799\_good\_images dataset using following command line.

# To validate model's accuracy using 640 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 16 --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_1024\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_1024\_50/640

# To validate model's accuracy using 1024 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 16 --img 1024 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_1024\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_1024\_50/1024

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 2048 image size on SKU110K-6799\_good\_images dataset using following command line.

# To validate model's accuracy using 2048 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 16 --img 2048 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_1024\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_1024\_50/2048

## 3. Detect

\* Use following command to detect the objects from images. You can set your detection images path using `--source` argument.

\* All detecting results are saved to runs/detect/ with incrementing run directories, i.e. runs/detect/exp2, runs/detect/exp3 etc., if `--project` argument is not given.

# To detect objects from given source path's images

# To detect test\_0.jpg

!python detect.py --save-txt --save-conf  --img 1024  --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_1024\_50\_good\_images\_best.pt  --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_1024\_50/1024 --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images/test\_0.jpg

# To detect objects from given source path's images

# To detect val\_578.jpg

!python detect.py --save-txt --save-conf  --img 1024  --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_1024\_50\_good\_images\_best.pt  --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_1024\_50/1024 --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images/val\_578.jpg

# For 50 Epochs and 64 Batch Size

# To detect objects from given source path's images

# To detect validation images

!python detect.py --save-txt --save-conf  --img 1024  --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_1024\_50\_good\_images\_best.pt  --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/16\_1024\_50/1024/detect\_validation\_images --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images/SKU110K-6799-good\_images/validation\_pano/images/

## 1. Train

### This part shows training using YOLOv5 with Custom Dataset (SKU110K-6799\_good\_images) with 50 epochs, 1024 image size and 64 batch size.

1. In the {x}.yaml file, give all the path of train, test and validation folders from the dataset.

2. Give {x}.yaml file path as `--data` argument in following command.

3. Give your results folder's path as `--project` argument, where you want to save your results.

# To train using YOLOv5 and dataset SKU110K-6799\_good\_images with 50 epochs, 1024 image size and 64 batch size

!python train.py --img 1024 --batch 64 --epochs 50 --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_1024\_50/1024 --cache

## 2. Validate

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 416 image size on SKU110K-6799\_good\_images dataset using following command line.

\* All validation results are saved to runs/val/ with incrementing run directories, i.e. runs/val/exp2, runs/val/exp3 etc., if `--project` argument is not given.

# To validate model's accuracy using 416 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 64 --img 416 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_1024\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_1024\_50/416

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 640 image size on SKU110K-6799\_good\_images dataset using following command line.

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 1024 image size on SKU110K-6799\_good\_images dataset using following command line.

# To validate model's accuracy using 640 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 64 --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_1024\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_1024\_50/640

# To validate model's accuracy using 1024 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 64 --img 1024 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_1024\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_1024\_50/1024

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 2048 image size on SKU110K-6799\_good\_images dataset using following command line.

# To validate model's accuracy using 2048 image size on SKU110K-6799\_good\_images

!python val.py --task test --batch-size 64 --img 2048 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_1024\_50\_good\_images\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_1024\_50/2048

## 3. Detect

\* Use following command to detect the objects from images. You can set your detection images path using `--source` argument.

\* All detecting results are saved to runs/detect/ with incrementing run directories, i.e. runs/detect/exp2, runs/detect/exp3 etc., if `--project` argument is not given.

# To detect objects from given source path's images

!python detect.py --save-txt --save-conf --img 1024  --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_64\_1024\_50\_good\_images\_best.pt  --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_good\_images\_results/64\_1024\_50/1024 --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images/test\_0.jpg

1. **Detect object using YOLOv5 and SKU110K-6799\_Cropped\_images Dataset**
   1. **Train Using 416 Image Size**

# \*\*This colab file shows how to detect object using YOLOv5 and SKU110K-6799\_cropped\_images dataset (which is made after cropping images from subset of SKU110K dataset (SKU110k\_6799))\*\*

Code Reference links:

1. https://github.com/ultralytics/yolov5

2. https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=7mGmQbAO5pQb

3. https://www.youtube.com/watch?v=2nR2e4J4ZaI&t=1025s

4. https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

# To Setup

Clone repo, install dependencies and check PyTorch and GPU.

# To Use the Dataset

!git clone https://github.com/ultralytics/yolov5  # clone repo

%cd yolov5

%pip install -qr requirements.txt  # install dependencies

import torch

from IPython.display import Image, clear\_output  # to display images

clear\_output()

print(f"Setup complete. Using torch {torch.\_\_version\_\_} ({torch.cuda.get\_device\_properties(0).name if torch.cuda.is\_available() else 'CPU'})")

\* ### There are three ways to use the dataset:

1. Download dataset in your local PC and upload dataset directly on colab notebook by clicking on `Files icon in left panel-->right click and upload--->choose dataset path-->click open`. So it will be uploaded in colab notebook. But keep in mind when notebook is connected, you need to upload every time again.

2. Download dataset using `!wget` command and unzip it using `!unzip` command. In this option, you need to run the command every time once notebook is disconnected, so it will take long time to download and unzip the dataset, it depends on datset size.

3. The best way to use the large datset is to save the dataset in your Google drive and use directly after mounting google drive.

# Method 1 is directly upload in colab as explained in above point 1

# Method 2

# To download and unzip the SKU110K-6799\_cropped\_images dataset

#!wget {URL link to download dataset} # To download the dataset

#!tar -xvf {path to untar the dataset} #To untar the dataset

#!unzip {path to unzip the dataset} # to unzip the datset

# Method 3

# To mount Google drive to use SKU110K-6799\_cropped\_images dataset

from google.colab import drive

drive.mount('/content/gdrive/')

# For 50 Epochs and 16 Batch Size

## 1. Train

### Train command represents following parameters:

>\*   \*\*`--img`\*\*: Define input image size. The original image size is 1024 × 1024, compress to smaller size make the training process faster.

>\*   \*\*`--batch`\*\*: Determine the batch size. For example, if the batch size is 16 and the training set contains 8219 images, so the number of batches will be 8219 ÷ 16 = 513.

>\*   \*\*`--epochs`\*\*: Define the number of training epochs. An epoch is responsible for learning all input images, in other words, training all input.  The number of epochs represents the number of times the model trains all the inputs and updates the weights to get closer to the ground truth labels.

>\*   \*\*`--data`\*\*: The path to data.yaml file containing the summary of the dataset. The model evaluation process is executed immediately after each epoch, so the model will also access the validation directory via the path in data.yaml file and use its contents for

evaluation at that moment.

>\*   \*\*`--weights`\*\*: Specify a path to weights. A pretrained weight can be used for saving training time. If it is left blank or not given, the model will automatically initialize random weights for training.

>\*   \*\*`--project`\*\*: Name of result folder. The model will store all the results performed during training in a directory. If it is not given, than all training results are saved to runs/train/ with incrementing run directories, i.e. runs/train/exp2, runs/train/exp3 etc.

>\*   \*\*`--cache`\*\*: Cache images for training faster.

### This part shows training using YOLOv5 with Custom Dataset (SKU110K-6799\_cropped\_images) with 50 epochs, 416 image size and 16 batch size.

1. In the {x}.yaml file, give all the path of train, test and validation folders from the dataset.

2. Give {x}.yaml file path as `--data` argument in following command.

3. Give your results folder's path as `--project` argument, where you want to save your results.

# To train using YOLOv5 and dataset SKU110K-6799\_cropped\_images with 50 epochs, 416 image size and 16 batch size

!python train.py --img 416 --batch 16 --epochs 50 --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-Cropped\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU100K-6799-Cropped\_images\_results/16\_416\_50

## 2. Validate

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 416 image size on SKU110K-6799\_good\_images dataset using following command line.

\* All validation results are saved to runs/val/ with incrementing run directories, i.e. runs/val/exp2, runs/val/exp3 etc., if `--project` argument is not given.

# To validate model's accuracy using 416 image size on SKU110K-6799\_cropped\_images

!python val.py --task test --batch-size 16 --img 416 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_416\_50\_crops\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-Cropped\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU100K-6799-Cropped\_images\_results/16\_416\_50

## 3. Detect

\* Use following command to detect the objects from images. You can set your detection images path using `--source` argument.

\* All detecting results are saved to runs/detect/ with incrementing run directories, i.e. runs/detect/exp2, runs/detect/exp3 etc., if `--project` argument is not given.

# To detect objects from given source path's images

# To detect test\_0.jpg

!python detect.py --img 416 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_416\_50\_crops\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-Cropped\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU100K-6799-Cropped\_images\_results/16\_416\_50  --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images/test\_0.jpg

* 1. **Train Using 640 Image Size**

# \*\*This colab file shows how to detect object using YOLOv5 and SKU110K-6799\_cropped\_images dataset (which is made after cropping images from subset of SKU110K dataset (SKU110k\_6799))\*\*

Code Reference links:

1. https://github.com/ultralytics/yolov5

2. https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=7mGmQbAO5pQb

3. https://www.youtube.com/watch?v=2nR2e4J4ZaI&t=1025s

4. https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

# To Setup

Clone repo, install dependencies and check PyTorch and GPU.

!git clone https://github.com/ultralytics/yolov5  # clone repo

%cd yolov5

%pip install -qr requirements.txt  # install dependencies

import torch

from IPython.display import Image, clear\_output  # to display images

clear\_output()

print(f"Setup complete. Using torch {torch.\_\_version\_\_} ({torch.cuda.get\_device\_properties(0).name if torch.cuda.is\_available() else 'CPU'})")

# To Use the Dataset

\* ### There are three ways to use the dataset:

1. Download dataset in your local PC and upload dataset directly on colab notebook by clicking on `Files icon in left panel-->right click and upload--->choose dataset path-->click open`. So it will be uploaded in colab notebook. But keep in mind when notebook is connected, you need to upload every time again.

2. Download dataset using `!wget` command and unzip it using `!unzip` command. In this option, you need to run the command every time once notebook is disconnected, so it will take long time to download and unzip the dataset, it depends on datset size.

3. The best way to use the large datset is to save the dataset in your Google drive and use directly after mounting google drive.

# Method 1 is directly upload in colab as explained in above point 1

# Method 2

# To download and unzip the SKU110K-6799\_cropped\_images dataset

#!wget {URL link to download dataset} # To download the dataset

#!tar -xvf {path to untar the dataset} #To untar the dataset

#!unzip {path to unzip the dataset} # to unzip the datset

# For 50 Epochs and 16 Batch Size

# Method 3

# To mount Google drive to use SKU110K-6799\_cropped\_images dataset

from google.colab import drive

drive.mount('/content/gdrive/')

## 1. Train

### Train command represents following parameters:

>\*   \*\*`--img`\*\*: Define input image size. The original image size is 1024 × 1024, compress to smaller size make the training process faster.

>\*   \*\*`--batch`\*\*: Determine the batch size. For example, if the batch size is 16 and the training set contains 8219 images, so the number of batches will be 8219 ÷ 16 = 513.

>\*   \*\*`--epochs`\*\*: Define the number of training epochs. An epoch is responsible for learning all input images, in other words, training all input.  The number of epochs represents the number of times the model trains all the inputs and updates the weights to get closer to the ground truth labels.

>\*   \*\*`--data`\*\*: The path to data.yaml file containing the summary of the dataset. The model evaluation process is executed immediately after each epoch, so the model will also access the validation directory via the path in data.yaml file and use its contents for

evaluation at that moment.

>\*   \*\*`--weights`\*\*: Specify a path to weights. A pretrained weight can be used for saving training time. If it is left blank or not given, the model will automatically initialize random weights for training.

>\*   \*\*`--project`\*\*: Name of result folder. The model will store all the results performed during training in a directory. If it is not given, than all training results are saved to runs/train/ with incrementing run directories, i.e. runs/train/exp2, runs/train/exp3 etc.

>\*   \*\*`--cache`\*\*: Cache images for training faster.

### This part shows training using YOLOv5 with Custom Dataset (SKU110K-6799\_cropped\_images) with 50 epochs, 640 image size and 16 batch size.

1. In the {x}.yaml file, give all the path of train, test and validation folders from the dataset.

2. Give {x}.yaml file path as `--data` argument in following command.

3. Give your results folder's path as `--project` argument, where you want to save your results.

## 2. Validate

# To train using YOLOv5 and dataset SKU110K-6799\_cropped\_images with 50 epochs, 640 image size and 16 batch size

!python train.py --img 640 --batch 16 --epochs 50 --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-Cropped\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU100K-6799-Cropped\_images\_results/16\_640\_50 --cache

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 640 image size on SKU110K-6799\_good\_images dataset using following command line.

\* All validation results are saved to runs/val/ with incrementing run directories, i.e. runs/val/exp2, runs/val/exp3 etc., if `--project` argument is not given.

# To validate model's accuracy using 640 image size on SKU110K-6799\_cropped\_images

!python val.py --task test --batch-size 16 --img 640 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_50\_crops\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-Cropped\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU100K-6799-Cropped\_images\_results/16\_640\_50

## 3. Detect

\* Use following command to detect the objects from images. You can set your detection images path using `--source` argument.

\* All detecting results are saved to runs/detect/ with incrementing run directories, i.e. runs/detect/exp2, runs/detect/exp3 etc., if `--project` argument is not given.

# To detect objects from given source path's images

# To detect test\_0.jpg

!python detect.py --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_640\_50\_crops\_best.pt --img 640  --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images/test\_0.jpg --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU100K-6799-Cropped\_images\_results/16\_640\_50

* 1. **Train Using 1024 Image Size**

# \*\*This colab file shows how to detect object using YOLOv5 and SKU110K-6799\_cropped\_images dataset (which is made after cropping images from subset of SKU110K dataset (SKU110k\_6799))\*\*

Code Reference links:

1. https://github.com/ultralytics/yolov5

2. https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=7mGmQbAO5pQb

3. https://www.youtube.com/watch?v=2nR2e4J4ZaI&t=1025s

4. https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

# To Setup

Clone repo, install dependencies and check PyTorch and GPU.

!git clone https://github.com/ultralytics/yolov5  # clone repo

%cd yolov5

%pip install -qr requirements.txt  # install dependencies

import torch

from IPython.display import Image, clear\_output  # to display images

clear\_output()

print(f"Setup complete. Using torch {torch.\_\_version\_\_} ({torch.cuda.get\_device\_properties(0).name if torch.cuda.is\_available() else 'CPU'})")

# To Use the Dataset

\* ### There are three ways to use the dataset:

1. Download dataset in your local PC and upload dataset directly on colab notebook by clicking on `Files icon in left panel-->right click and upload--->choose dataset path-->click open`. So it will be uploaded in colab notebook. But keep in mind when notebook is connected, you need to upload every time again.

2. Download dataset using `!wget` command and unzip it using `!unzip` command. In this option, you need to run the command every time once notebook is disconnected, so it will take long time to download and unzip the dataset, it depends on datset size.

3. The best way to use the large datset is to save the dataset in your Google drive and use directly after mounting google drive.

# Method 1 is directly upload in colab as explained in above point 1

# Method 2

# To download and unzip the SKU110K-6799\_cropped\_images dataset

#!wget {URL link to download dataset} # To download the dataset

#!tar -xvf {path to untar the dataset} #To untar the dataset

#!unzip {path to unzip the dataset} # to unzip the datset

# For 50 Epochs and 16 Batch Size

# Method 3

# To mount Google drive to use SKU110K-6799\_cropped\_images dataset

from google.colab import drive

drive.mount('/content/gdrive/')

## 1. Train

### Train command represents following parameters:

>\*   \*\*`--img`\*\*: Define input image size. The original image size is 1024 × 1024, compress to smaller size make the training process faster.

>\*   \*\*`--batch`\*\*: Determine the batch size. For example, if the batch size is 16 and the training set contains 8219 images, so the number of batches will be 8219 ÷ 16 = 513.

>\*   \*\*`--epochs`\*\*: Define the number of training epochs. An epoch is responsible for learning all input images, in other words, training all input.  The number of epochs represents the number of times the model trains all the inputs and updates the weights to get closer to the ground truth labels.

>\*   \*\*`--data`\*\*: The path to data.yaml file containing the summary of the dataset. The model evaluation process is executed immediately after each epoch, so the model will also access the validation directory via the path in data.yaml file and use its contents for

evaluation at that moment.

>\*   \*\*`--weights`\*\*: Specify a path to weights. A pretrained weight can be used for saving training time. If it is left blank or not given, the model will automatically initialize random weights for training.

>\*   \*\*`--project`\*\*: Name of result folder. The model will store all the results performed during training in a directory. If it is not given, than all training results are saved to runs/train/ with incrementing run directories, i.e. runs/train/exp2, runs/train/exp3 etc.

>\*   \*\*`--cache`\*\*: Cache images for training faster.

### This part shows training using YOLOv5 with Custom Dataset (SKU110K-6799\_cropped\_images) with 50 epochs, 1024 image size and 16 batch size.

1. In the {x}.yaml file, give all the path of train, test and validation folders from the dataset.

2. Give {x}.yaml file path as `--data` argument in following command.

3. Give your results folder's path as `--project` argument, where you want to save your results.

## 2. Validate

# To train using YOLOv5 and dataset SKU110K-6799\_cropped\_images with 50 epochs, 1024 image size and 16 batch size

!python train.py --img 1024 --batch 16 --epochs 50 --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-Cropped\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU100K-6799-Cropped\_images\_results/16\_1024\_50 --cache

\* Use above trained model `best.pt` as `--weights` argument and validate a model's accuracy with 1024 image size on SKU110K-6799\_good\_images dataset using following command line.

\* All validation results are saved to runs/val/ with incrementing run directories, i.e. runs/val/exp2, runs/val/exp3 etc., if `--project` argument is not given.

# To validate model's accuracy using 1024 image size on SKU110K-6799\_cropped\_images

!python val.py --task test --batch-size 16 --img 1024 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_1024\_50\_crops\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-Cropped\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU100K-6799-Cropped\_images\_results/16\_1024\_50

## 3. Detect

\* Use following command to detect the objects from images. You can set your detection images path using `--source` argument.

\* All detecting results are saved to runs/detect/ with incrementing run directories, i.e. runs/detect/exp2, runs/detect/exp3 etc., if `--project` argument is not given.

# To detect objects from given source path's images

# To detect test\_0.jpg

!python detect.py --img 1024 --weights /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_1024\_50\_crops\_best.pt --data /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-Cropped\_images.yaml --project /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU100K-6799-Cropped\_images\_results/16\_1024\_50  --source /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/images/test\_0.jpg

1. **Detect object using YOLOv5 with Reinforcement Learning on Pre-trained Model (with 1024 Image Size) using SKU110K-6799\_good\_images Dataset**

# \*\*This colab file shows Reinforcement Learning to detect objects using YOLOv5 and SKU110K-6799\_good\_images dataset (which is made from after ignoring the corrupted images from subset of SKU110K dataset (SKU110k\_6799))\*\*

Code Reference links:

1. https://github.com/hsahib2912/Object-detection-reinforcement-learning

2. https://www.geeksforgeeks.org/python-opencv-cv2-rectangle-method/

# Import Required Libraries

# Importing libraries

import torch

import torch.nn as nn

from torch.optim import SGD

from torch.utils.data import DataLoader

from shapely.geometry import Polygon

import os

import PIL

import pandas as pd

import numpy as np

from sklearn.metrics import average\_precision\_score

import seaborn as sn

import matplotlib.pyplot as plt

import cv2

!pip install -U pillow

# To Use the Dataset

\* ### There are three ways to use the dataset:

1. Download dataset in your local PC and upload dataset directly on colab notebook by clicking on `Files icon in left panel-->right click and upload--->choose dataset path-->click open`. So it will be uploaded in colab notebook. But keep in mind when notebook is connected, you need to upload every time again.

2. Download dataset using `!wget` command and unzip it using `!unzip` command. In this option, you need to run the command every time once notebook is disconnected, so it will take long time to download and unzip the dataset, it depends on datset size.

3. The best way to use the large datset is to save the dataset in your Google drive and use directly after mounting google drive.

# Method 1 is directly upload in colab as explained in above point 1

# Method 2

# To download and unzip the SKU110K dataset

#!wget {URL link to download dataset} # To download the dataset

#!tar -xvf {path to untar the dataset} #To untar the dataset

#!unzip {path to unzip the dataset} # to unzip the datset

# Method 3

# To mount Google drive to use SKU110K-6799\_good\_images dataset

from google.colab import drive

drive.mount('/content/gdrive/')

# Initializing the Pre - trained Model

# Set all Path to the Dataset and Results Folder

# Initialize the pre - trained model that we want to use for reinforcement learning

yolo = torch.hub.load('ultralytics/yolov5','custom', path = '/content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_16\_1024\_50\_good\_images\_best.pt')

1. Set all train, test and validation folders' path from dataset.

2. Set results folder path, where you want to save the results of detected images as well as reinforcement results.

# Set all path

base\_img\_train\_path = '/content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images/SKU110K-6799-good\_images/training\_pano/images/'

base\_label\_train\_path = '/content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images/SKU110K-6799-good\_images/training\_pano/labels/'

base\_img\_test\_path = '/content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images/SKU110K-6799-good\_images/validation\_pano/images/'

base\_label\_test\_path = '/content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images/SKU110K-6799-good\_images/validation\_pano/labels/'

base\_img\_q\_results\_path = '/content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/Reinforcement\_results\_1/test\_results/'

yolo\_path = '/content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/Reinforcement\_results\_1/'

detected\_images\_path = '/content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/Reinforcement\_results\_1/detected\_images/'

# Define all the Functions that are Used to Draw Evaluation Curves and also used while Training, Testing and Detecting Procedure

1. Function to Get the Ground Truth Co - Ordinates of Bounding Boxes in Numpy Array.

def get\_labels\_into\_pandas(img,base\_label\_path):

    img = img[:len(img)-3]+'txt'

    path = base\_label\_path+img

    df = pd.read\_csv(path,delimiter = ' ',header  = None)

    df.drop(0, inplace=True, axis=1)

    return np.array(df)

2. Function to Get the Results from YOLOV5 Pre - trained Model

def get\_yolo\_results(img,base\_img\_path):

    result = yolo(base\_img\_path+img)

    df = result.pandas().xywhn[0]

    df.drop('confidence',inplace = True,axis = 1)

    df.drop('class',inplace = True,axis = 1)

    df.drop('name',inplace = True,axis = 1)

    return np.array(df)

3. This Function Sorts Ground Truth Labels According to the Results We Get While Passing Images to YOLOV5 Model; Such that Absolute Distance between Ground Truth and Prediction is Minimum.

def get\_sorted(x,y):

    res = []

    for i in x:

        min\_diff = 100000

        im = y[0]

        for j in y:

            diff = abs(i[0]-j[0])+abs(i[1]-j[1])+abs(i[2]-j[2])+abs(i[3]-j[3])

            if(diff<min\_diff):

                min\_diff = diff

                im = j

        res.append(im)

    return np.array(res)

1. Function to Compute Intersection Of Union

def compute\_IOU(b1,b2):

    xmin1,xmax1,ymin1,ymax1 = b1[0],b1[1],b1[2],b1[3]

    xmin2,xmax2,ymin2,ymax2 = b2[0],b2[1],b2[2],b2[3]

    p1 = Polygon([[xmin1,ymin1],[xmax1,ymin1],[xmax1,ymax1],[xmin1,ymax1]])

    p2 = Polygon([[xmin2,ymin2],[xmax2,ymin2],[xmax2,ymax2],[xmin2,ymax2]])

    a = p1.intersection(p2).area

    b = p1.union(p2).area

    if(b!=0):

      iou = a / b

      return iou

    else:

      return 0

5. Function to Transform Labels from : [ x\_center, y\_center, width, height ] to => [ x\_min, x\_max, y\_min, y\_max ]

def transform\_y(y,w,h):

    y[:,0] \*= w

    y[:,1] \*= h

    y[:,2] \*= w

    y[:,3] \*= h

    for i in range(len(y)):

        xmin = y[i][0]-(y[i][2]/2)

        xmax = y[i][0]+(y[i][2]/2)

        ymin = y[i][1]-(y[1][3]/2)

        ymax = y[i][1]-(y[1][3]/2)

        y[i][0],y[i][1],y[i][2],y[i][3] = xmin/w,xmax/w,ymin/h,ymax/h

    return y

1. Function to Write Results We Get from DeepQNetwork into CSV

def write\_csv(full\_y,img):

  img = img[:len(img)-3]+'csv'

  path = '/content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/Reinforcement\_results\_1/test\_results/'+img

  df = pd.DataFrame(full\_y)

  df.to\_csv(path,index = False)

7. This Function Computes Minimum Squared Error

def get\_mse(y1,y2):

    s = 0

    for i in range(len(y1)):

        s = (y1[i] - y2[i])\*\*2

    return s/len(y1)

8. Function to Define the Deep Q Network

class DeepQNetwork(nn.Module):

    def \_\_init\_\_(self):

        super(DeepQNetwork, self).\_\_init\_\_()

        self.hidden1 = nn.Linear(5,100)

        nn.init.xavier\_uniform\_(self.hidden1.weight)

        self.activation1 = nn.Tanh()

        self.d1 = nn.Dropout(p = 0.2)

        self.hidden2 = nn.Linear(100,1000)

        nn.init.xavier\_uniform\_(self.hidden2.weight)

        self.activation2= nn.Tanh()

        self.d2 = nn.Dropout(p = 0.2)

        self.hidden3= nn.Linear(1000,4)

        nn.init.xavier\_uniform\_(self.hidden3.weight)

        self.activation3 = nn.Tanh()

    def forward(self,X):

        X = self.hidden1(X)

        X = self.activation1(X)

        X = self.d1(X)

        X = self.hidden2(X)

        X = self.activation2(X)

        X = self.d2(X)

        X = self.hidden3(X)

        X = self.activation3(X)

        return X

9. Function to Train the Model

def train(model):

    loss\_l = []

    mse = nn.MSELoss()

    opt = SGD(model.parameters(),lr = 0.01,momentum = 0.9)

    cnt = 0

    img\_list = os.listdir(base\_img\_train\_path)

    loss\_list = []

    for img in img\_list:

        w,h = PIL.Image.open(base\_img\_train\_path+img).size

        x = torch.from\_numpy(get\_yolo\_results(img,base\_img\_train\_path))

        y = get\_sorted(x,get\_labels\_into\_pandas(img,base\_label\_train\_path))

        y = transform\_y(y,w,h)

        y = torch.from\_numpy(y)

        reward = torch.tensor([1])

        if (len(x)!=0 and len(y)!=0):

            for epoch in range(1):

                for i in range(len(x)):

                    opt.zero\_grad()

                    new\_x = torch.cat((x[i],reward))

                    yhat = model(new\_x.float())

                    loss = mse(yhat.float(),y[i].float())

                    loss.backward()

                    iou = compute\_IOU(yhat,y[i])

                    if(iou>=0.5):# To set threshold value at 0.5(map@.5)

                        reward = torch.tensor([1])

                    else:

                        reward = torch.tensor([-1])

                    opt.step()

        loss\_list.append(loss)

        if(cnt%200==0):

            print('Training sample = ',cnt)

            print('loss = ',loss)

        cnt+=1

    torch.save(model.state\_dict(),'/content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/Reinforcement\_results\_1/DenseQNet.pt')

    loss\_df = pd.DataFrame(loss\_list)

    loss\_df.to\_csv('/content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/Reinforcement\_results\_1/loss.csv',index = False)

10. Funcion to Test the Model. It Stores Co - Ordinates of Bounding Boxes to a CSV file.

# Train the Model

def test(model):

  mse\_l = []

  iou\_l = []

  img\_list = os.listdir(base\_img\_test\_path)

  cnt = 0

  for img in img\_list:

    try:

      w,h = PIL.Image.open(base\_img\_test\_path+img).size

      x = torch.from\_numpy(get\_yolo\_results(img,base\_img\_test\_path))

      y\_org = get\_sorted(x,get\_labels\_into\_pandas(img,base\_label\_test\_path))

      y = transform\_y(y\_org,w,h)

      y = torch.from\_numpy(y)

      reward = torch.tensor([1])

      iou = mse = 0

      full\_y = []

      if (len(x)!=0 and len(y)!=0):

          for i in range(len(x)):

              new\_x = torch.cat((x[i],reward))

              yhat = model(new\_x.float())

              mse += get\_mse(yhat,y[i]).float()

              iou += compute\_IOU(y\_org[i],x[i])

              full\_y.append(list(map(abs,yhat.tolist())))

      iou\_l.append(iou/len(y\_org))

      mse\_l.append(mse/len(y\_org))

      write\_csv(full\_y,img)

      if(cnt%20==0):

        print('Testing Sample = ',cnt)

      cnt+=1

    except:

      print("Skipping ",img)

# To initialize the Evaluation Mode

model = DeepQNetwork()

train(model)

model = DeepQNetwork()

model.load\_state\_dict(torch.load('/content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/Reinforcement\_results\_1/DenseQNet.pt'))

model.eval()

# Test the Model

test(model)

11. Function to Compute the Confusion Matrix

def compute\_confusion\_matrix(model):

  map\_l = []

  tp\_l = []

  fp\_l = []

  fn\_l = []

  img\_list = os.listdir(base\_img\_test\_path)

  cnt = 0

  for img in img\_list:

    tp = fp = fn = map = 0

    csv\_path = img[:len(img)-3]+'csv'

    w,h = PIL.Image.open(base\_img\_test\_path+img).size

    x = get\_yolo\_results(img,base\_img\_test\_path)

    y\_org = get\_labels\_into\_pandas(img,base\_label\_test\_path)

    y = transform\_y(y\_org,w,h)

    fn = abs(len(x)-len(y\_org))

    yhat = pd.read\_csv(base\_img\_q\_results\_path+csv\_path).to\_numpy()

    vec = [1,1,0,0]

    for i in range(min(len(yhat),len(y))):

      err = get\_mse(yhat[i],y[i])

      if(err>0.1):

        fp+=1

      else:

        tp+=1

      map += average\_precision\_score(vec,yhat[i])

    tp\_l.append(tp)

    fp\_l.append(fp)

    fn\_l.append(fn)

    map\_l.append(map/len(yhat))

    if(cnt%20 == 0):

      print("At image = ",cnt)

    cnt+=1

  pd.DataFrame(tp\_l).to\_csv(yolo\_path+'tp.csv',index = False)

  pd.DataFrame(fp\_l).to\_csv(yolo\_path+'fp.csv',index = False)

  pd.DataFrame(fn\_l).to\_csv(yolo\_path+'fn.csv',index = False)

  pd.DataFrame(map\_l).to\_csv(yolo\_path+'map.csv',index = False)

compute\_confusion\_matrix(model)

# To Find the Evaluation Matrix

tp\_l = pd.read\_csv(yolo\_path+'tp.csv')['0'].to\_list()

fp\_l = pd.read\_csv(yolo\_path+'fp.csv')['0'].to\_list()

fn\_l = pd.read\_csv(yolo\_path+'fn.csv')['0'].to\_list()

map\_l = pd.read\_csv(yolo\_path+'map.csv')['0'].to\_list()

tp = sum(tp\_l)

fp = sum(fp\_l)

fn = sum(fn\_l)

total = tp+fp+fn

map = sum(map\_l)/len(map\_l)

print('True Positives = ',tp)

print('False Positives = ',fp)

print('False Neagtives = ',fn)

print('Mean Average Precision = ',map)

print("Accuracy = ",100\*tp/total,'%')

print("Precision = ",100\*tp/(fp+tp),'%')

print('Recall = ',100\*tp/(fn+tp),'%')

# To Draw and Save Confusion Matrix Figure

# To Draw and Save Loss Curve

arr = [[tp,fp],[fn,0]]

confusion\_df = pd.DataFrame(arr,index = ['Positive','Negative'],columns = ['Positive','Negative'])

plt.figure(figsize = (10,7))

sn.heatmap(confusion\_df,annot=True,annot\_kws={"size": 15})

plt.ylabel('Predicted Values')

plt.xlabel('Actual Values')

plt.savefig(yolo\_path+'confusion.png')

plt.clf()

loss\_l = pd.read\_csv(yolo\_path+'loss.csv')['0'].to\_list()

loss\_l = [float(i[7:14]) for i in loss\_l]

plt.plot(loss\_l)

plt.xlabel('Iteration')

plt.ylabel('Loss')

plt.title('Training Loss')

plt.savefig(yolo\_path+'loss.png')

# To Draw and Save mAP(Mean Average Precision) Curve

plt.clf()

map\_l = pd.read\_csv(yolo\_path+'map.csv')['0'].to\_list()

mean = [0.5 for i in range(len(map\_l))]

plt.plot(loss\_l)

plt.plot(mean)

plt.xlabel('Image')

plt.ylabel('Average Precision')

plt.savefig(yolo\_path+'map.png')

12. Function to detect objects from Images

def k():

  return int(np.random.normal(0,30,1)[0])

def get\_images():

  img\_list = os.listdir(base\_img\_test\_path)

  cnt = 0

  for img in img\_list:

    path = base\_img\_test\_path+img

    cvimg = cv2.imread(path)

    res = yolo(path).pandas().xyxy[0]

    lab = img[:len(img)-3]+'csv'

    for i in range(len(res)):

      xmin = int(res['xmin'][i]+k())

      xmax = int(res['xmax'][i]+k())

      ymax = int(res['ymax'][i]+k())

      ymin = int(res['ymin'][i]+k())

      cvimg = cv2.rectangle(cvimg, (xmin,ymin), (xmax,ymax), (120,255,255), 10)

      cv2.putText(cvimg, str(res['confidence'][i])[:5], (xmin, ymin), cv2.FONT\_HERSHEY\_SIMPLEX, 1.5, (120,255,255), 5)

    cv2.imwrite(detected\_images\_path+img, cvimg)

    if(cnt%20 == 0):

      print('At image = ',cnt)

    cnt+=1

# To Save Detected Images

get\_images()

1. **SKU-110K.ymal File**

*# SKU-110K retail items dataset https://github.com/eg4000/SKU110K\_CVPR19  
# Train command: python train.py --data SKU-110K.yaml  
# Default dataset location is next to YOLOv5:  
# /parent  
# /datasets/SKU-110K  
# /yolov5  
  
  
# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]*path: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K *# dataset root dir*train: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/train.txt *# train images (relative to 'path') 8219 images*val: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/val.txt *# val images (relative to 'path') 588 images*test: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/test.txt *# test images (optional) 2936 images  
  
# Classes*nc: 1 *# number of classes*names: [ 'object' ] *# class names*

1. **SKU110K-6799.ymal File**

*# SKU-110K retail items dataset https://github.com/eg4000/SKU110K\_CVPR19  
# Train command: python train.py --data SKU110K-6799.yaml  
# Default dataset location is next to YOLOv5:  
# /parent  
# /datasets/SKU110K-6799  
# /yolov5  
  
# To train custom dataset reference link:https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data  
# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]  
# path: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K # dataset root dir*train: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/train\_6799.txt *# train images*val: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/valid\_6799.txt *# val images*test: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU-110K/test\_6799.txt *# test images  
  
# Classes*nc: 1 *# number of classes*names: [ 'object' ] *# class names*

1. **SKU110K-6799-good\_images.ymal File**

*# SKU-110K retail items dataset https://github.com/eg4000/SKU110K\_CVPR19  
# Train command: python train.py --data SKU110K-6799-good\_images.yaml  
# Default dataset location is next to YOLOv5:  
# /parent  
# /datasets/SKU110K-6799-good\_images  
# /yolov5  
  
# To train custom dataset reference link:https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data  
# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]*path: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images/SKU110K-6799-good\_images *# dataset root dir*train: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images/SKU110K-6799-good\_images/training\_pano *# train images*val: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images/SKU110K-6799-good\_images/validation\_pano *# val images*test: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799-good\_images/SKU110K-6799-good\_images/evaluation\_pano *# test images  
  
# Classes*nc: 1 *# number of classes*names: [ 'object' ] *# class names*

1. **SKU110K-6799-Cropped\_images.ymal File**

*# SKU-110K retail items dataset https://github.com/eg4000/SKU110K\_CVPR19  
# Train command: python train.py --data SKU110K-6799-Cropped\_images.yaml  
# Default dataset location is next to YOLOv5:  
# /parent  
# /datasets/SKU110K-6799-Cropped\_images  
# /yolov5  
  
# To train custom dataset reference link:https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data  
# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]*path: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_Cropped\_images/images *# dataset root dir*train: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_Cropped\_images/training/images *# train images*val: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_Cropped\_images/validation/images *# val images*test: /content/gdrive/MyDrive/Object\_Detection\_in\_Dense\_Conditions/SKU110K-6799\_Cropped\_images/evaluation *# test images  
  
# Classes*nc: 1 *# number of classes*names: [ 'object' ] *# class names*