Applying Object-detection from the Images Generated by Stable-Diffusion

Soyeon Kim, Yejin Kim, Yoojin Oh
Ewha Womans University, Dept of Artificial Intelligence

Outline

- Introduction
- Background (Stable diffusion, YOLO)
- Data Collection and Preprocessing
- Modeling
- Performance results
- Discussion & Conclusion

Introduction

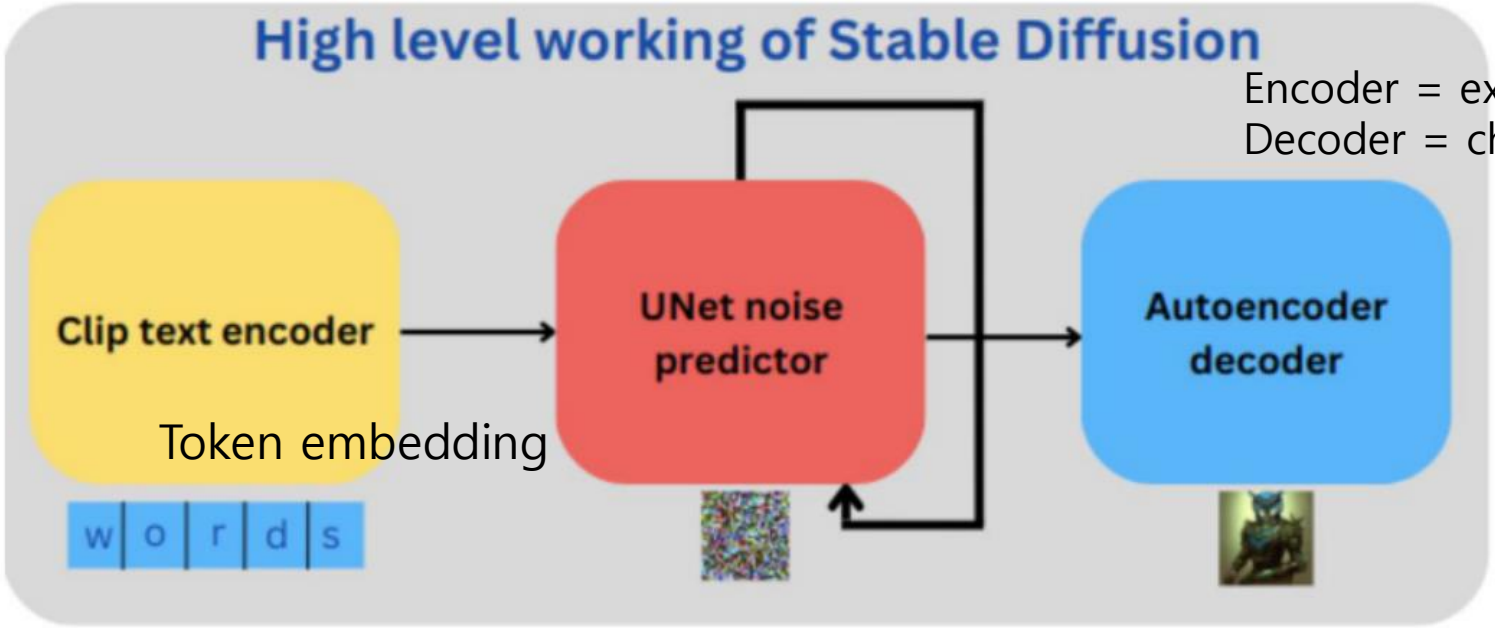
- These days, a substantial volume of data with high quality is required for improving the performance of image-related models. Through Stable-Diffusion models, we can generate images from text which are valuable.

- In this research, we'll apply generated images to YOLOv5 model for Object-detection in order to check how useful these generated images are.

Background : Stable-Diffusion

◆The characteristics of **Stable Diffusion**

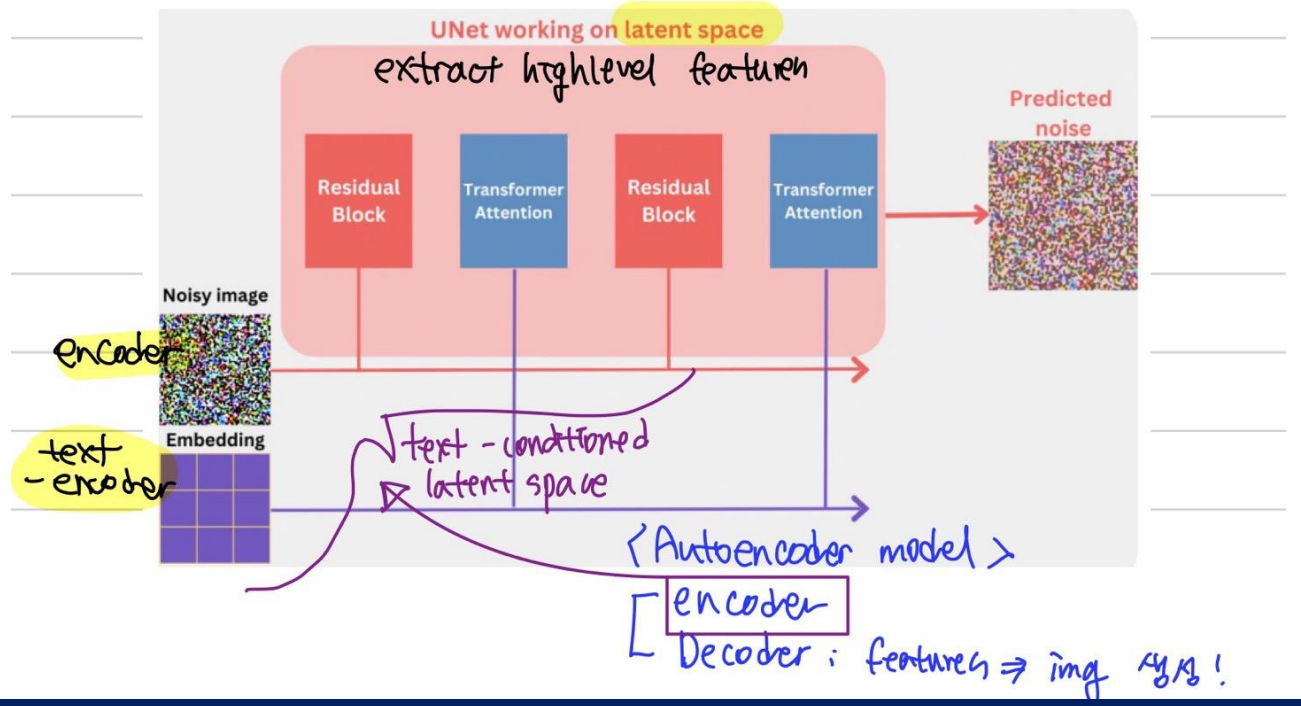
1. Text-to-image (txt2img) (also img2img possible)
2. It is open-source and publicly available, allowing anyone, including the general public, to use it.
3. With the ControlNet plugin, it is possible to control sd models. (adding condition)



Background : Stable-Diffusion

◆UNet Noise Predictor

- Goal : Predict noise which is added to image
- Only considers “high level feature” of “latent space”.
- No need of encoder for inference (only denoising)



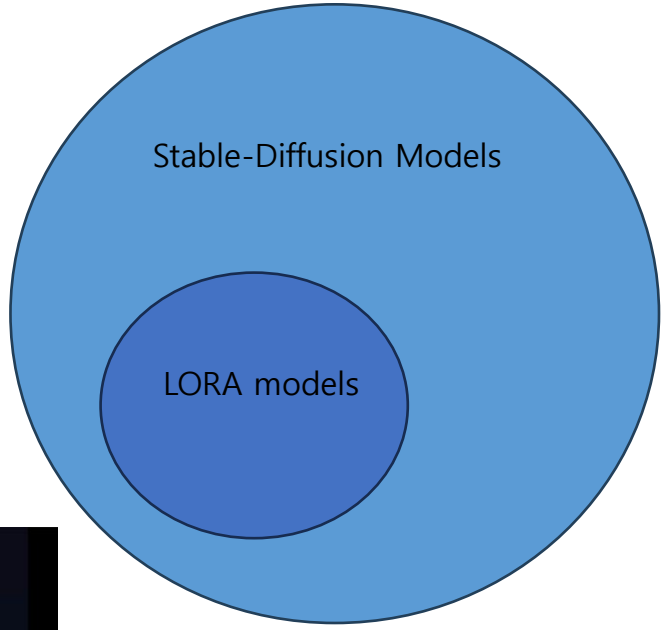
Background : Stable-Diffusion

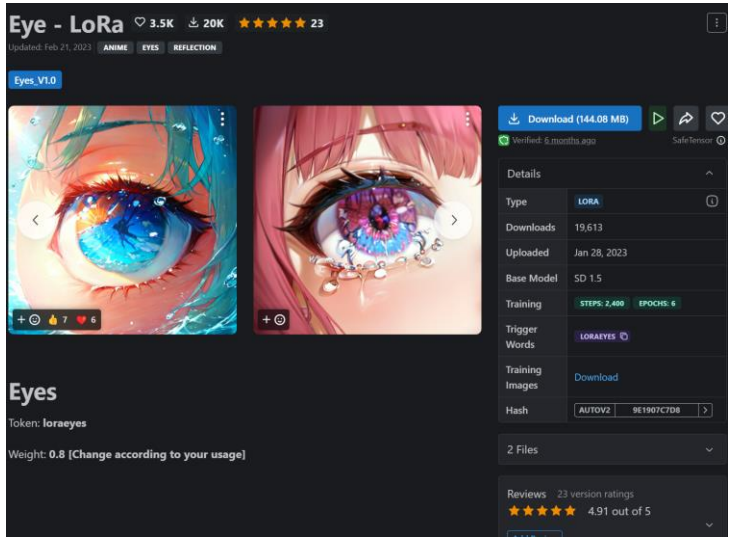
◆**How to Train the model?**

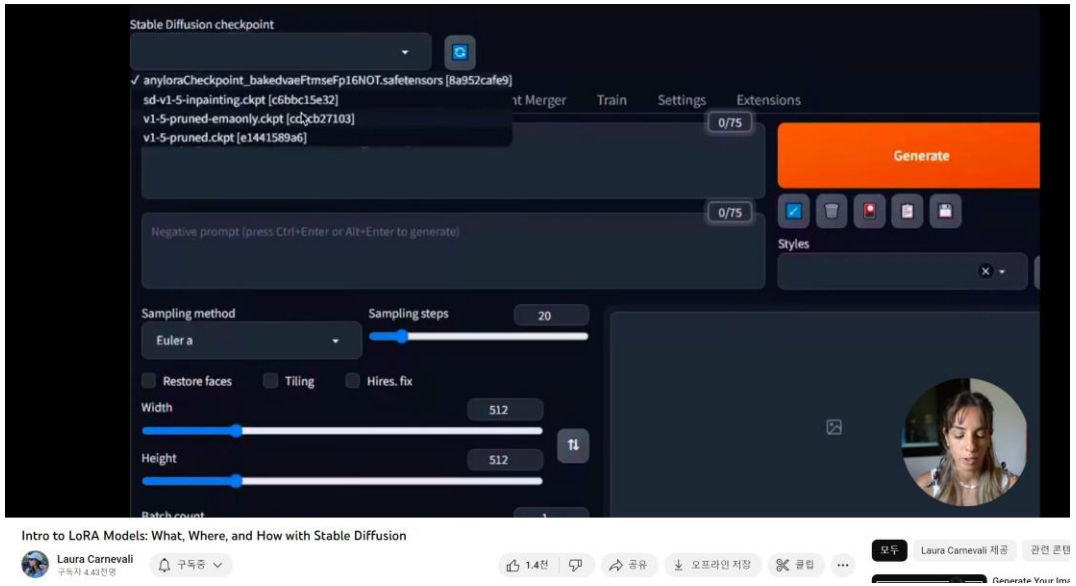
1) Dreambooth 2) Embedding

◆**LORA**

: layer that can be added to base model ,small changes or fine-tuning



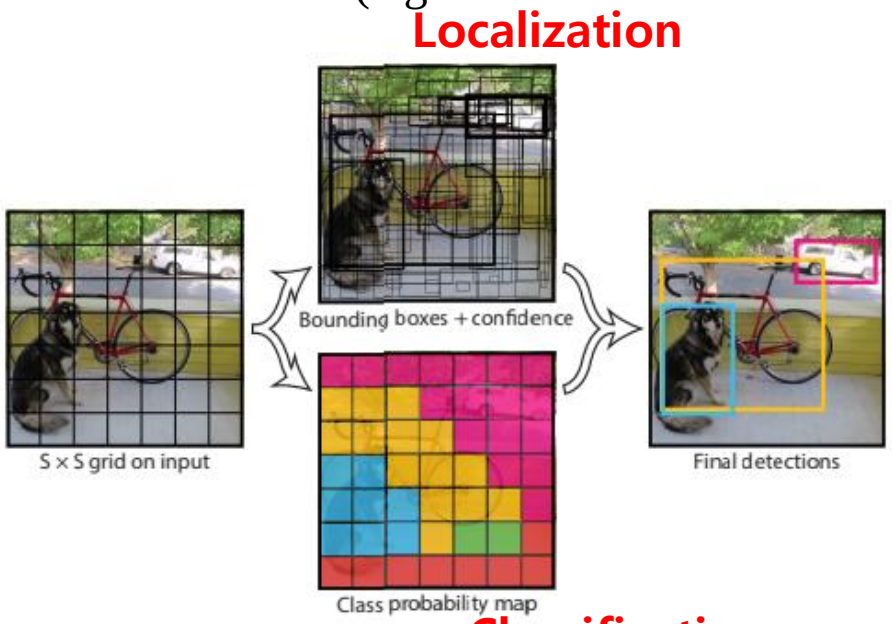




Background : YOLO [Short Paper Review]

◆ YOLO (You Only Look Once: Unified, Real-Time Object Detection)

- Unified Architecture : (Classification + Localization in one step, One-stage detector)
- Improved velocity compared to DPM, RCNN models
- Available at various domains (e.g. artwork and natural images)



*** Object-Detection**

: Classification + Localization

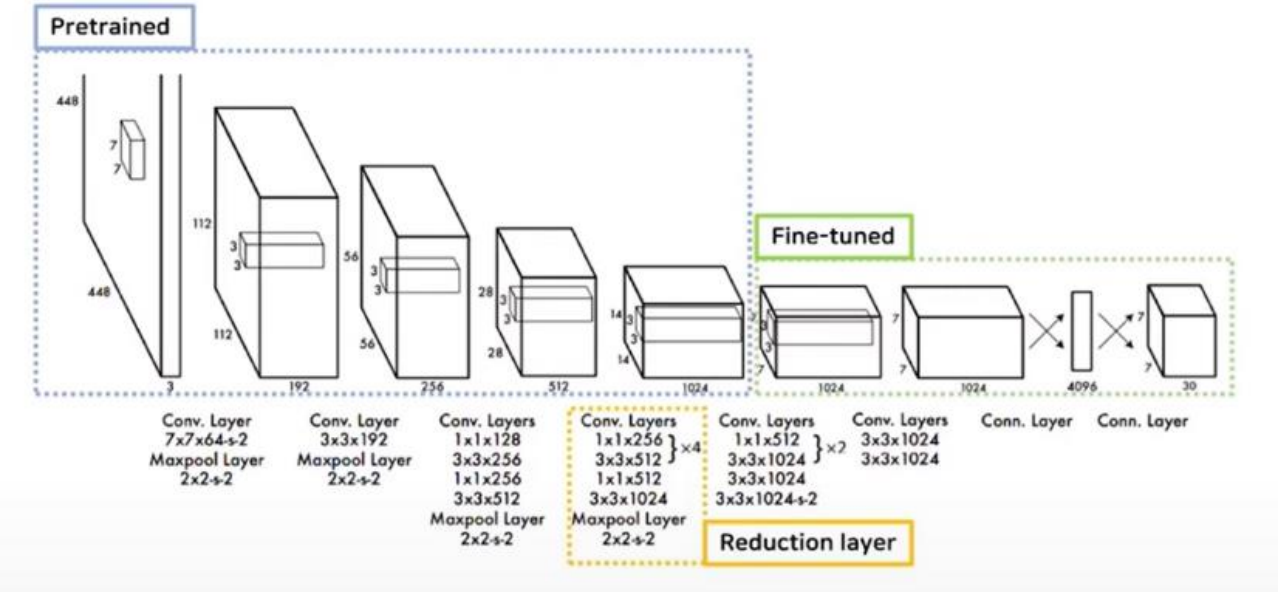
(detect object) (generate bounding box)

https://arxiv.org/abs/1506.026

Background : YOLO [Short Paper Review]

◆ Network Design

- 20 conv layer(pretrained) + (4 conv layer + 2 fc layer)(fine-tuned)
- Preceding layers : 1x1 convolutional layers -> reduce the feature space**(+)**



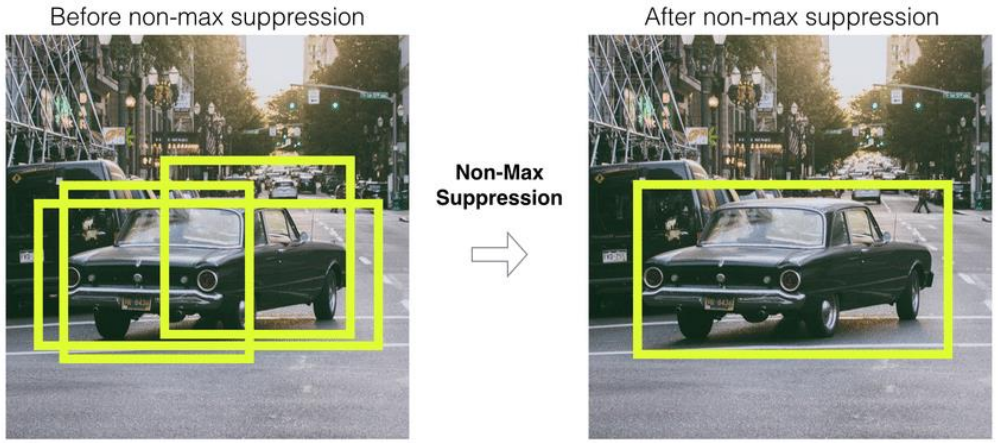
Background : YOLO [Short Paper Review]

◆ Inference

- **Non-Maximum Suppression (NMS)**

: check scores of bounding boxes for each class

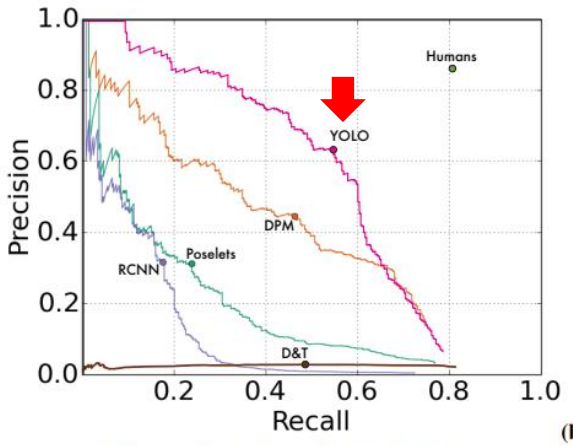
And leave the box of the highest score.



◆ Conclusion

- Trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly.

*YOLOv5 came after YOLO, …
It is Easier to configure and implement env when we use YOLOv5

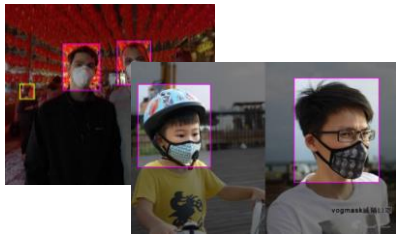


(a) Picasso Dataset precision-recall curves.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

Project Pipeline Process Steps



Mask Wearing Dataset

input

Custom Trained YOLO v5

Object-Detection

test

Generated Images Through txt prompt in Stable-Diffusion model + LORA

Results : Well Detected or not?

Data Collection & Preprocessing

◆Data Collection

Collected Mask Wearing dataset from Roboflow.

size : 416 x 416 px

class : 2 (Mask, No-Mask)

Export Size : 105 images

```
[1]  !curl -L "https://public.roboflow.com/ds/kwNaEyhiG3?key=bXsno2FuR8" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
     extracting: train/labels/1042977068_jpg.rf.84c3af9736d1fe2db012d0e91aee5b23.txt
```

◆ Data Preprocessing

Split Train : Test = 80 : 20 (true amount = 84, 21)

random_state = 42

```
%cat /content/dataset/data.yaml

train: ../train/images
val: ../valid/images

nc: 2
names: ['mask', 'no-mask']
```

nc (클래스의 갯수) = 2

```
%cd /
from glob import glob

img_list = glob('/content/dataset/train/images/*.jpg')

print(len(img_list)) #이미지 갯수 출력 ( =10개 )
```

```
/
105
```

https://public.roboflow.com/object-detection/mask-wearing

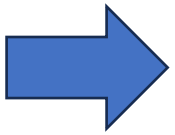
Data Collection & Preprocessing

◆Data Preprocessing

Edited yaml file to move directory & add txt file. (train.txt, val.txt)

```
%cat /content/dataset/data.yaml
```

```
train: ../train/images
val: ../valid/images

nc: 2
names: ['mask', 'no-mask']
```

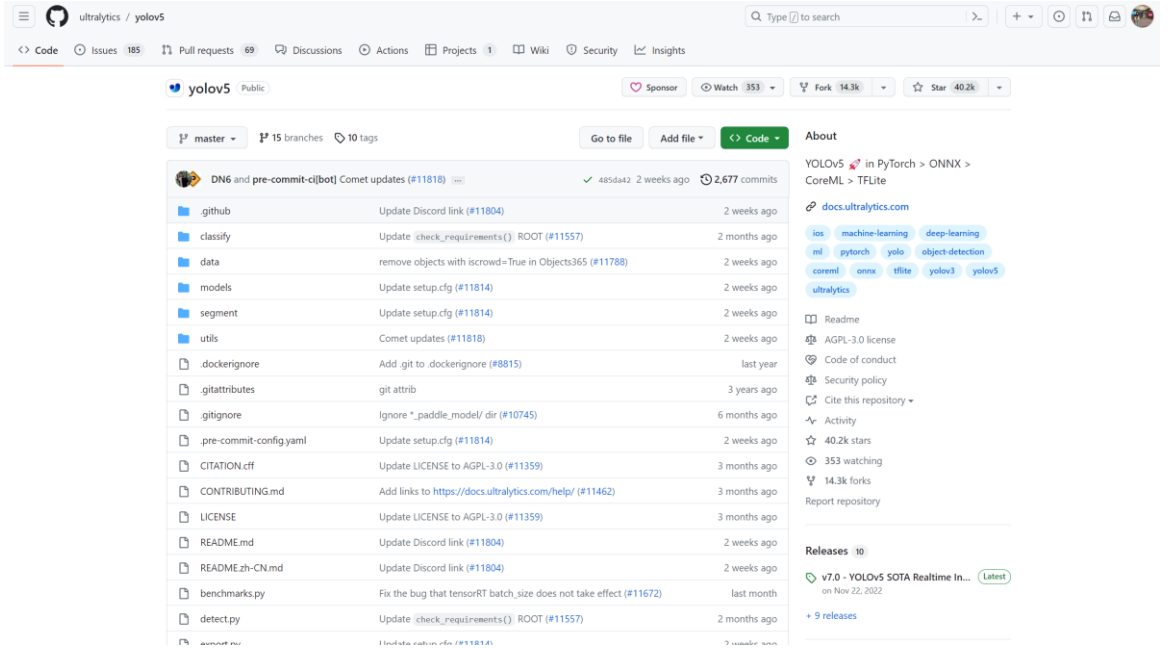


```
with open('/content/dataset/data.yaml', 'r') as f:
  data = yaml.safe_load(f)

print(data)

data['train'] = '/content/dataset/train.txt'
data['val'] = '/content/dataset/val.txt'

with open('/content/dataset/data.yaml', 'w') as f:
  yaml.dump(data, f)

print(data)
```

```
{'train': '../train/images', 'val': '../valid/images', 'nc': 2, 'names': ['mask', 'no-mask']}
{'train': '/content/dataset/train.txt', 'val': '/content/dataset/val.txt', 'nc': 2, 'names': ['mask', 'no-mask']}
```

Load & Train Model with Custom Dataset

> Loaded our model(YOLOv5) through cloning YOLOv5 repo.



> Train Model

- Train the model using preprocessed dataset.
- Image size = 416 x 416
- Batch = 16
- Epochs = 50

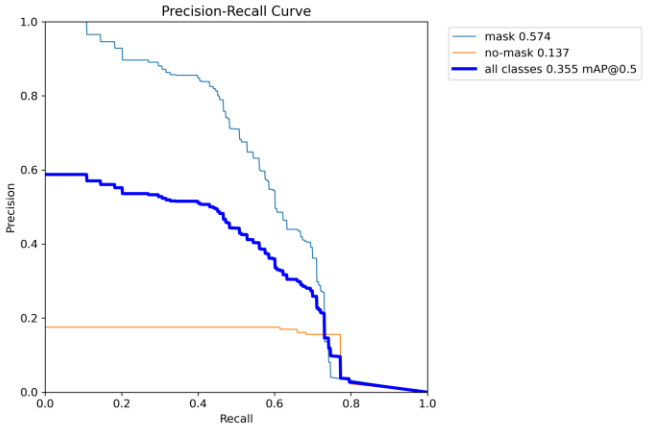
50 epochs completed in 0.037 hours.

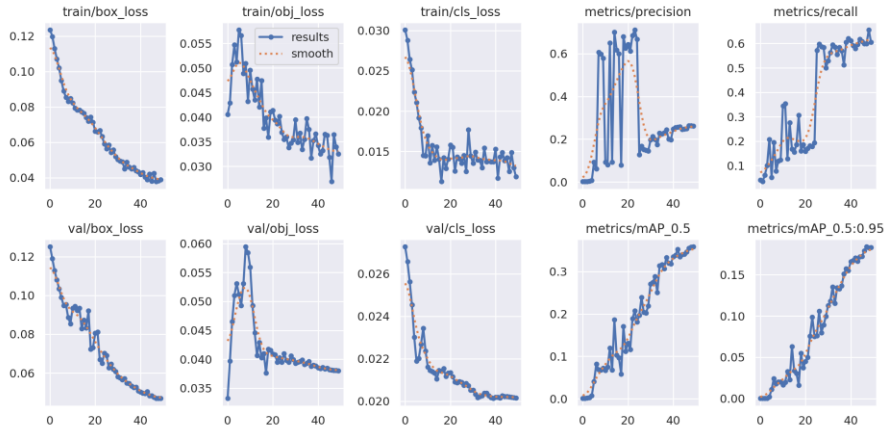
```
Epoch   GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
24/49     1.54G    0.05909     0.0369    0.01411         19        416: 100% 6/6 [00:00<00:00,  6.93it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 1/1 [00:00<00:00,  2.74it/s]
             all         21        237      0.668      0.194      0.181     0.0752

Epoch   GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
25/49     1.54G    0.05636    0.03545    0.01353         22        416: 100% 6/6 [00:00<00:00,  6.17it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 1/1 [00:00<00:00,  2.12it/s]
             all         21        237      0.128      0.572      0.197      0.076

Epoch   GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
26/49     1.54G    0.05846      0.036    0.01452         39        416: 100% 6/6 [00:00<00:00,  6.33it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 1/1 [00:00<00:00,  1.55it/s]
             all         21        237      0.168      0.598      0.239      0.106

Epoch   GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
27/49     1.54G    0.05426    0.03379    0.01254         19        416: 100% 6/6 [00:01<00:00,  4.56it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 1/1 [00:00<00:00,  2.45it/s]
             all         21        237      0.152      0.587      0.204     0.0799

Epoch   GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
28/49     1.54G    0.05578    0.03472    0.01769         23        416: 100% 6/6 [00:00<00:00,  7.07it/s]
           Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 1/1 [00:00<00:00,  2.16it/s]
             all         21        237      0.148      0.584      0.202     0.0891
```

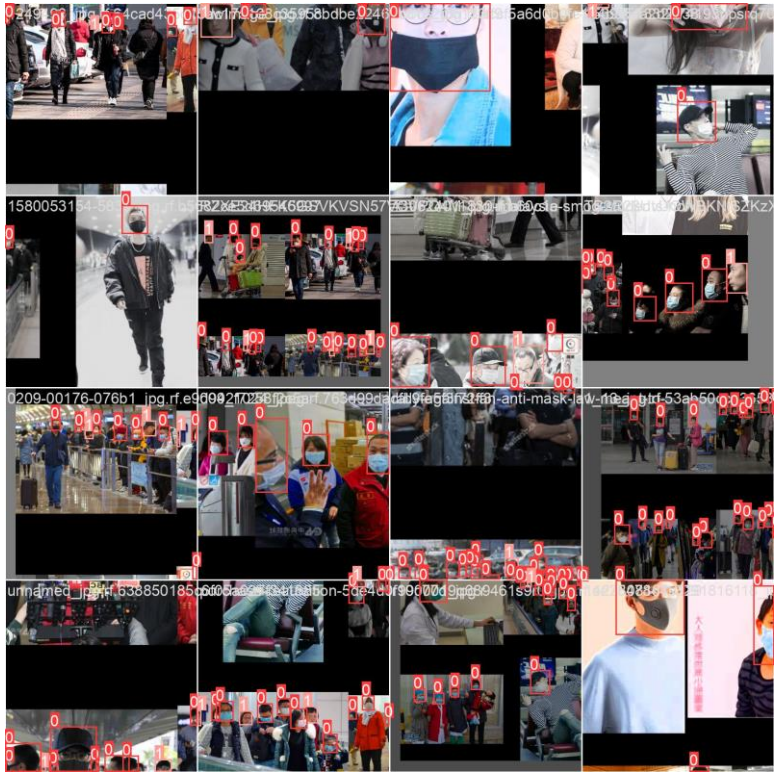
https://github.com/ultralytics/yolov5

Performance Results (1) : metrics

```
YOLOv5s summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 1/1 [00:00<00:00,  2.32it/s]
                   all         21        237      0.264        0.6      0.355      0.184
                  mask         21        193      0.363      0.699      0.574      0.303
               no-mask         21         44      0.165        0.5      0.137     0.0657
```



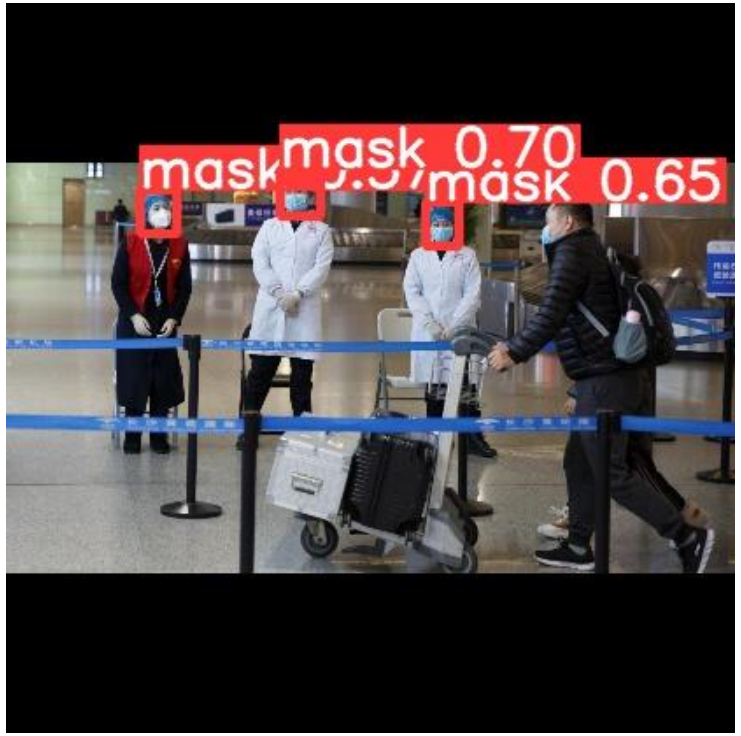




Train_Batch0.jpg

*mAP gives insight into the model's ability to accurately localize objects in the image.

Performance results (2)



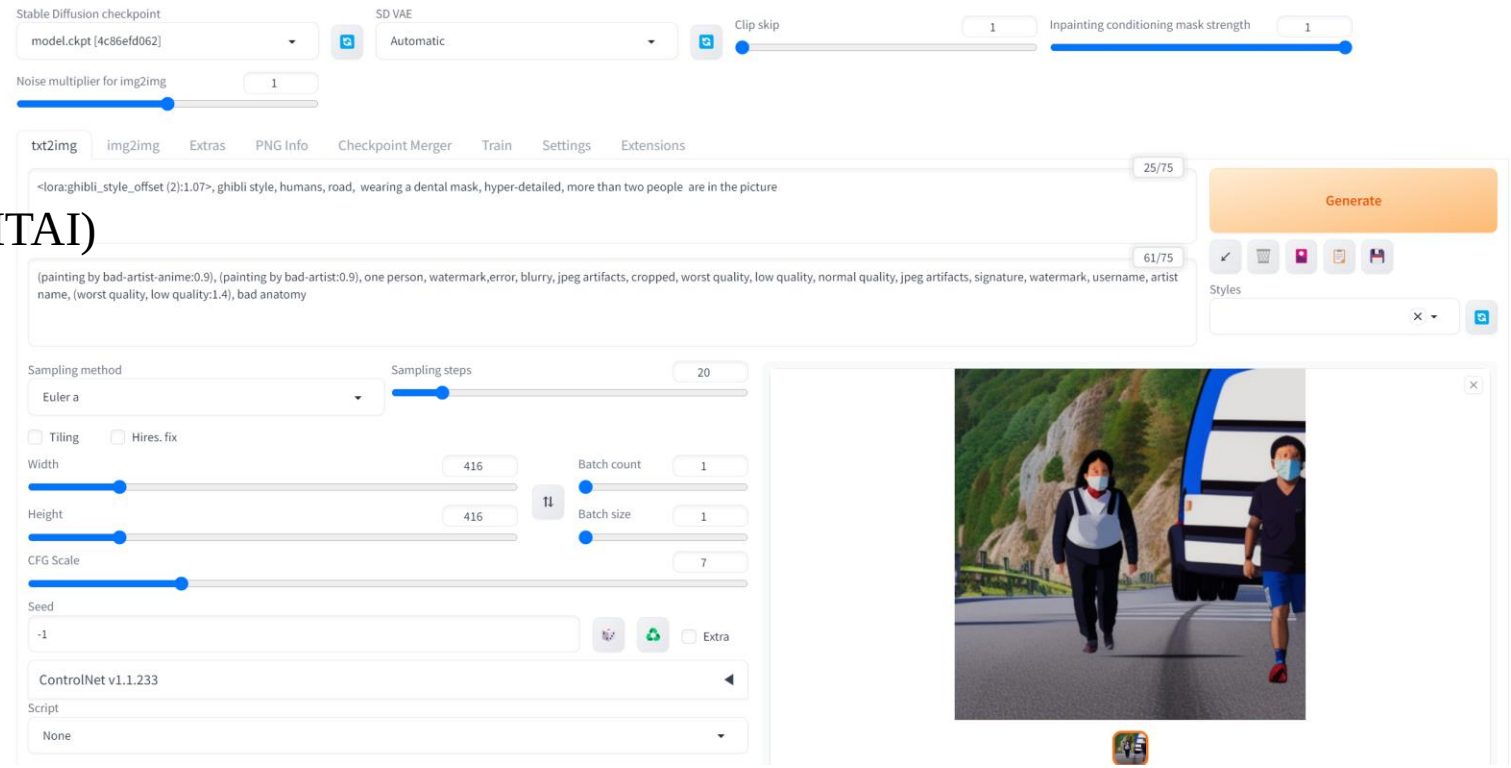
→ Seems that the model detected the objects quite successfully

Test on Generated images

Model : 1.5
LORA : Studio Ghibli Style LoRA (CIVITAI)

Sampler = Euler a
Width, height = 416, 416
Sampling steps = 20
Batch count, size = 1
Seed = -1

We've Done "txt2img" task



https://civitai.com/models/6526?modelVersionId=7657

Test on Generated images

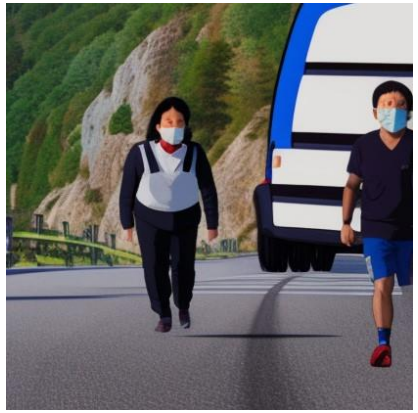
Model : 1.5
LORA : Studio Ghibli Style LoRA (CIVITAI)

<lora:ghibli_style_offset (2):1.07>, ghibli style,
majority of people, wearing a mouth mask, hyper-detailed

Negative prompt: (painting by bad-artist-anime:0.9), (painting by bad-artist:0.9), watermark,error, blurry, jpeg artifacts, cropped, worst quality, low quality, normal quality, jpeg artifacts, signature, watermark, username, artist name, (worst quality, low quality:1.4), bad anatomy

With LORA



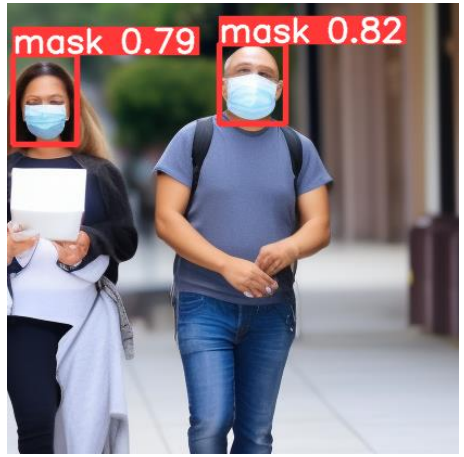


Without LORA


Performance results (3) : Apply on Generated images

Test on other images

```
g_img_list = glob('/content/g_imgs/*.png')

from IPython.display import Image
import os

val_img_path = g_img_list[0]
!python detect.py --weights /content/yolov5/runs/train/gun_yolov5s_results/weights/best.pt --img 416 --conf 0.5 --source "{val_img_path}"
```



Not Detected

Well-Detected

→ Seems that the model also detected the generated images quite successfully

Discussion & Conclusion

- What we've done: develop YOLOv5 model, object-detection, observe generated images made with LORA.
- Consequently, as mAP(Mean Average Precision) was almost 0.6 and higher compared to no-mask/all task, we can notice that the model is successfully detecting the mask-wearing cases.
- But, the quality of the generated images could be more improved if we give more detailed prompt.

- Outlook: Use of mask detection - public places, healthcare facilities, and crowded areas.
- Thus the content of this project might help to reduce the spread of respiratory illnesses, including COVID-19 and other airborne infections.

```
YOLOv5s summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 1/1 [00:00<00:00,  2.32it/s]
                   all         21        237      0.264        0.6      0.355      0.184
                  mask         21        193      0.363      0.699      0.574      0.303
               no-mask         21         44      0.165        0.5      0.137     0.0657
```

Thank you

Soyeon Kim, Yejin Kim, Yoojin Oh
Ewha Womans University, Dept of Artificial Intelligence