

---



# Klasifikasi Ruangan di Gedung Pascasarjana PENS

Anggota Kelompok:

Silfiana Nur Hamida (1223800005)

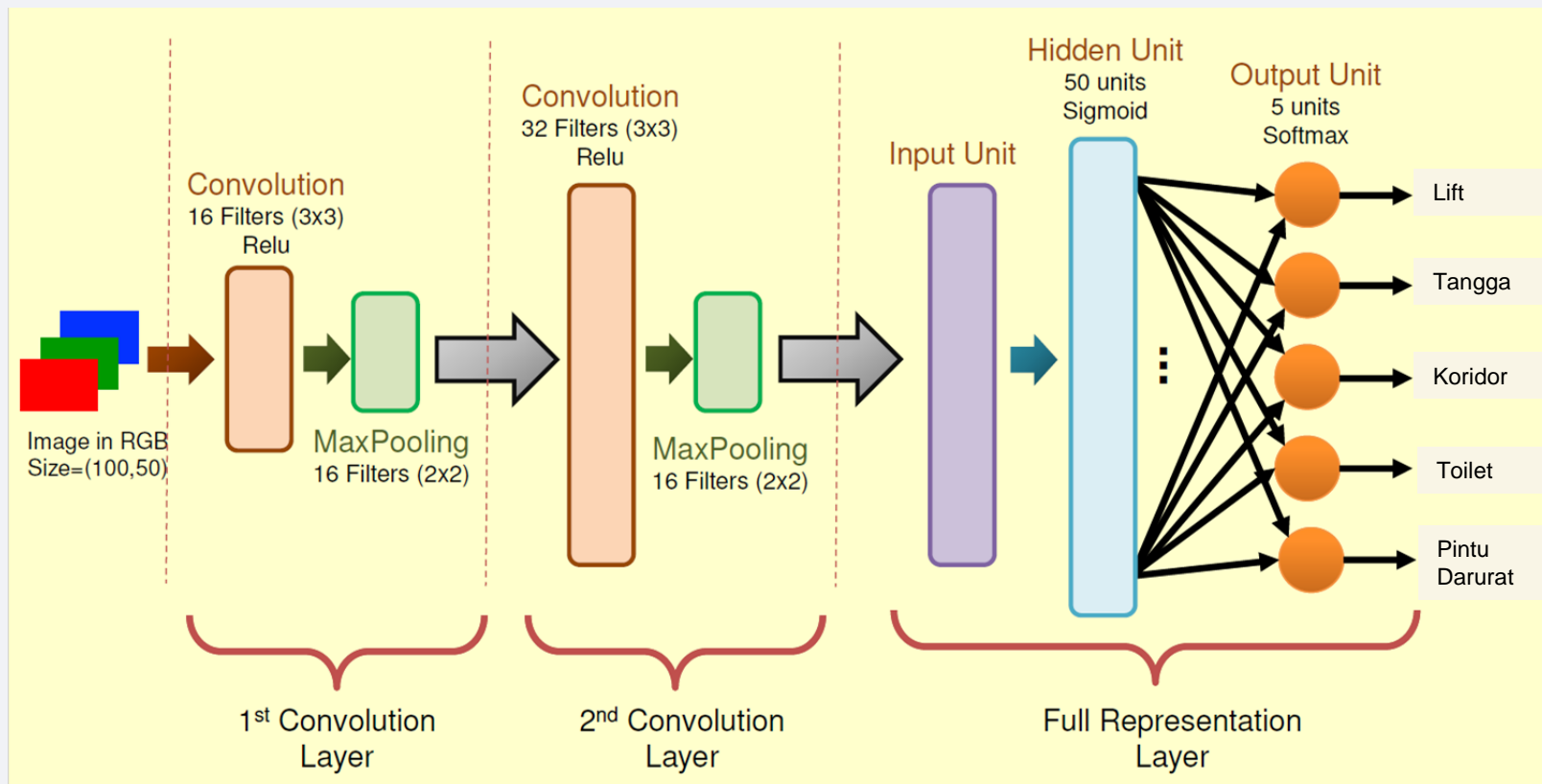
Ambarwati Rizkia Putri (1123800008)

Firnanda Pristiana Nurmaida (1122800004)

Mochamad Riswandha Lazuardi (1123800006)



# Pendahuluan



---

# Melakukan Klasifikasi Gambar

Gambar diklasifikasikan dalam 6 macam, yaitu:

- Lift
- Tangga
- Koridor
- Buntu
- Toilet
- Pintu Darurat

Semua gambar yang didapatkan kemudian dibagi menjadi tiga bagian, yaitu data training, validation, dan test.

---

# Code

## Membuka direktori

```
%cd /content/drive/MyDrive/CNN_Image4_5
```

## Membaca folder dataset

```
import os  
base_dir = "/content/drive/MyDrive/CNN_Image4_5/dataset"  
train_dir = os.path.join(base_dir, 'train')  
validation_dir = os.path.join(base_dir, 'validation')  
test_dir = os.path.join(base_dir, 'test')  
folders=os.listdir(train_dir)
```

---

## Melakukan preprocessing gambar dengan menggunakan ImageDataGenerator

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
train_dir,
target_size=(100, 50),
batch_size=1)
validation_generator = val_datagen.flow_from_directory(
validation_dir,
target_size=(100, 50),
batch_size=1)
```

## Melakukan Identifikasi class train dan class validation

```
class_names_train = train_generator.class_indices  
class_names_validation = validation_generator.class_indices
```

```
print("Nama Kelas Train:", class_names_train)  
print("Nama Kelas Validation:", class_names_validation)
```

## Menggunakan keras dan import layers dan model

```
from tensorflow.keras import layers  
from tensorflow.keras import Model  
img_input = layers.Input(shape=(100, 50, 3))  
x = layers.Conv2D(16, 3, activation='relu')(img_input)  
x = layers.MaxPooling2D(2)(x)  
x = layers.Conv2D(32, 3, activation='relu')(x)  
x = layers.MaxPooling2D(2)(x)  
x = layers.Flatten()(x)  
x = layers.Dense(50, activation='sigmoid')(x)  
output = layers.Dense(5, activation='softmax')(x)  
model = Model(img_input, output)  
model.compile(loss='mean_squared_error', optimizer='SGD', metrics=['acc'])
```

## Melakukan training data dengan 100 iterasi

```
history = model.fit_generator(  
train_generator,  
steps_per_epoch=70,  
epochs=100,  
validation_data=validation_generator,  
validation_steps=20,  
verbose=2)
```

## Menampilkan hasil akurasi dan loss dari data training dan validation

```
import matplotlib.pyplot as plt  
acc = history.history['acc']  
val_acc = history.history['val_acc']  
loss = history.history['loss']  
val_loss = history.history['val_loss']  
epochs = range(len(acc))  
plt.plot(epochs, acc, color='b', label='Train Accuracy')  
plt.plot(epochs, val_acc, color='r', label='Validation Accuracy')  
plt.title('Training and validation accuracy')  
plt.legend()  
plt.figure()  
plt.plot(epochs, loss, color='b', label='Train Loss')  
plt.plot(epochs, val_loss, color='r', label='Validation Loss')  
plt.title('Training and validation loss')  
plt.legend()
```

---

Menentukan nilai output (hasil prediksi). Penentuan kelas yaitu berdasarkan nilai prediksi tertinggi

```
from keras.preprocessing.image import img_to_array, load_img
import numpy as np
img = load_img(test_dir+'/20231109_183136.jpg', False, target_size=(100,50))
x = img_to_array(img)
x = np.expand_dims(x, axis=0)
preds = model.predict(x)
print("Nilai Output Units:\n", preds)
index_preds = np.argmax(preds)
print("\nPredicted :", index_preds)
```





1/1 [=====] - 0s 19ms/step

Nilai Output Units:

[[0.00580255 0.0069457 0.9762245 0.00246838 0.00855886]]

Predicted : 2

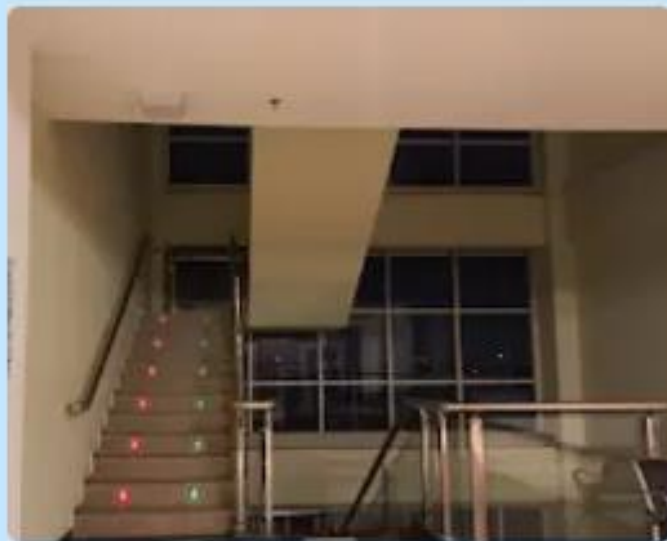
---

Menentukan nilai output (hasil prediksi). Penentuan kelas yaitu berdasarkan nilai prediksi tertinggi

```
from keras.preprocessing.image import img_to_array, load_img
import numpy as np
img = load_img(test_dir+'/20231109_183454.jpg', False, target_size=(100,50))
x = img_to_array(img)
x = np.expand_dims(x, axis=0)
preds = model.predict(x)
print("Nilai Output Units:\n", preds)
index_preds = np.argmax(preds)
print("\nPredicted :", index_preds)
```



20231109\_18345...



1/1 [=====] - 0s 36ms/step

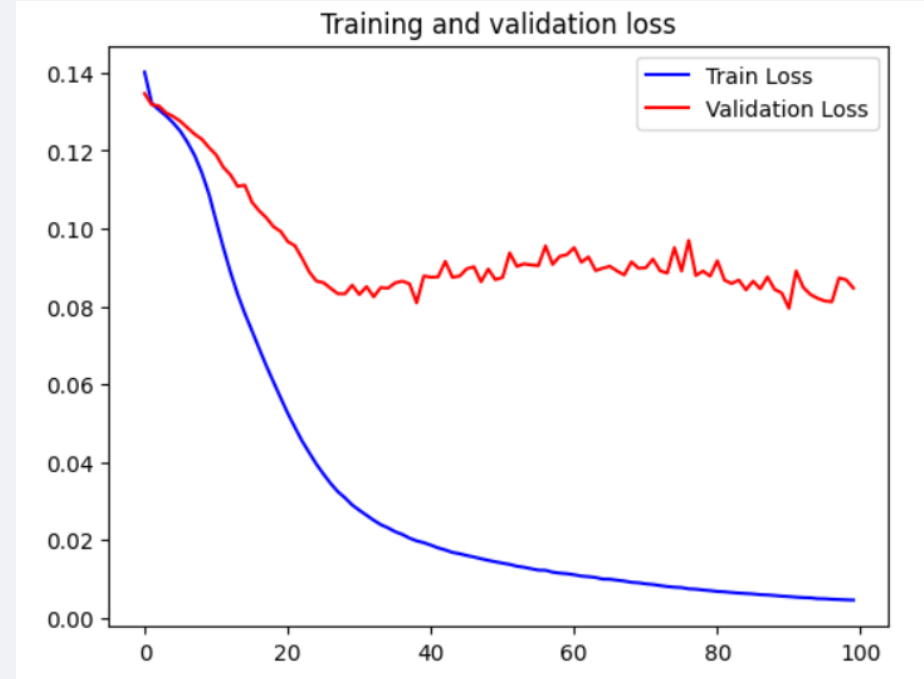
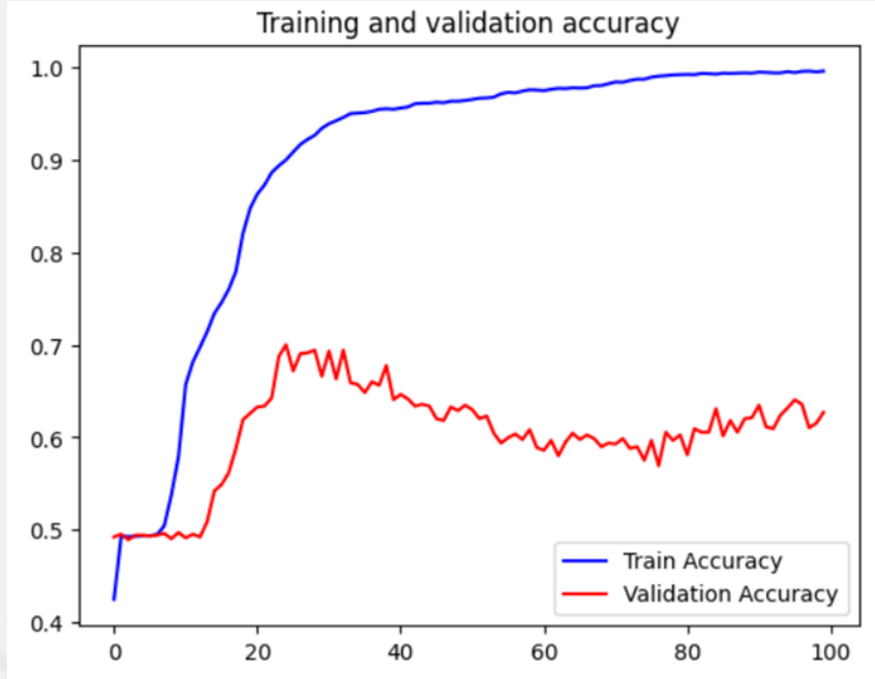
Nilai Output Units:

[[0.00499308 0.00457729 0.00705227 0.9821929 0.00118443]]

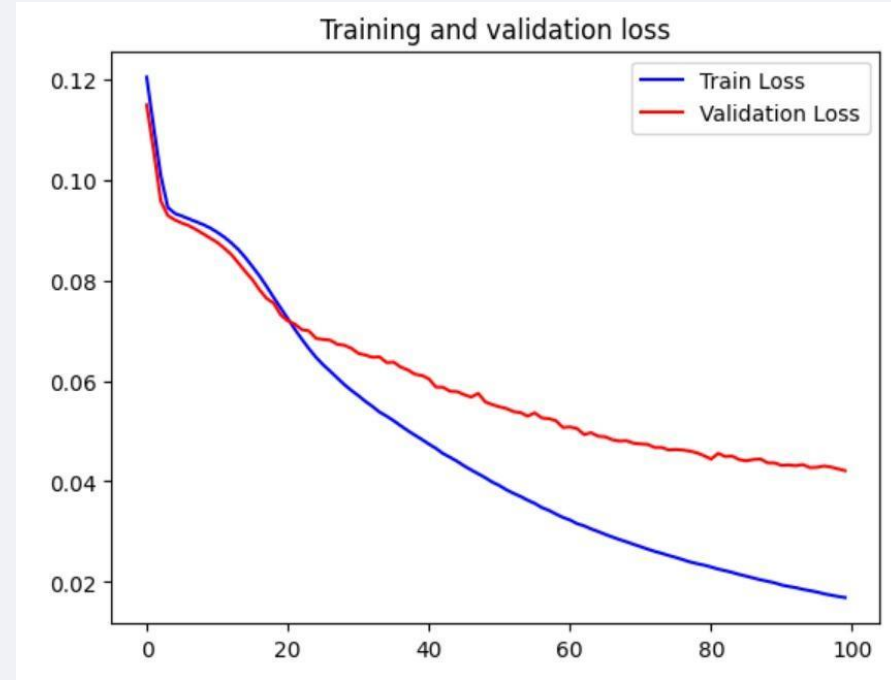
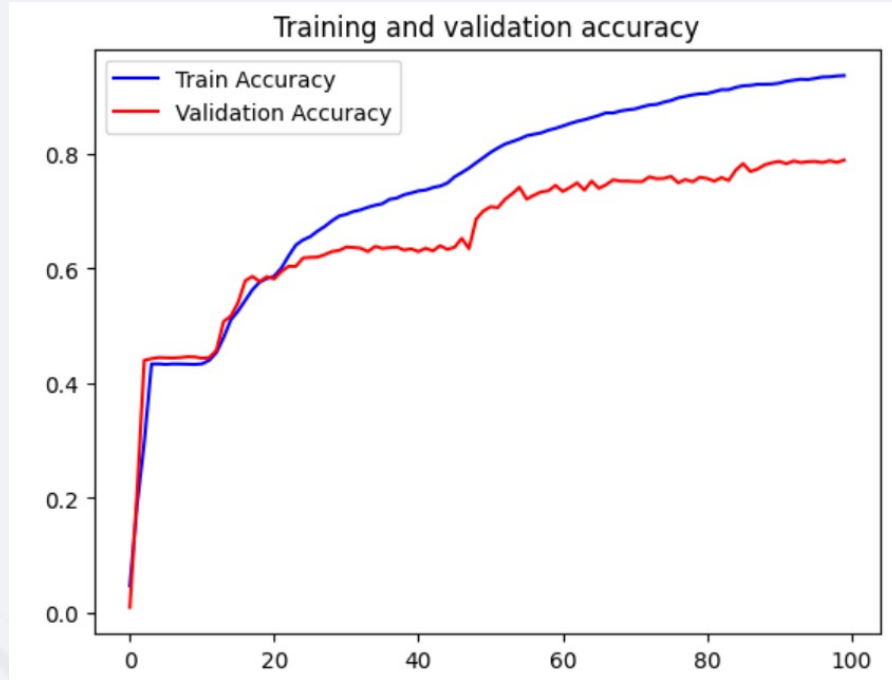
Predicted : 3

# Hasil (Perbandingan dataset sebelumnya dengan dataset saat ini)

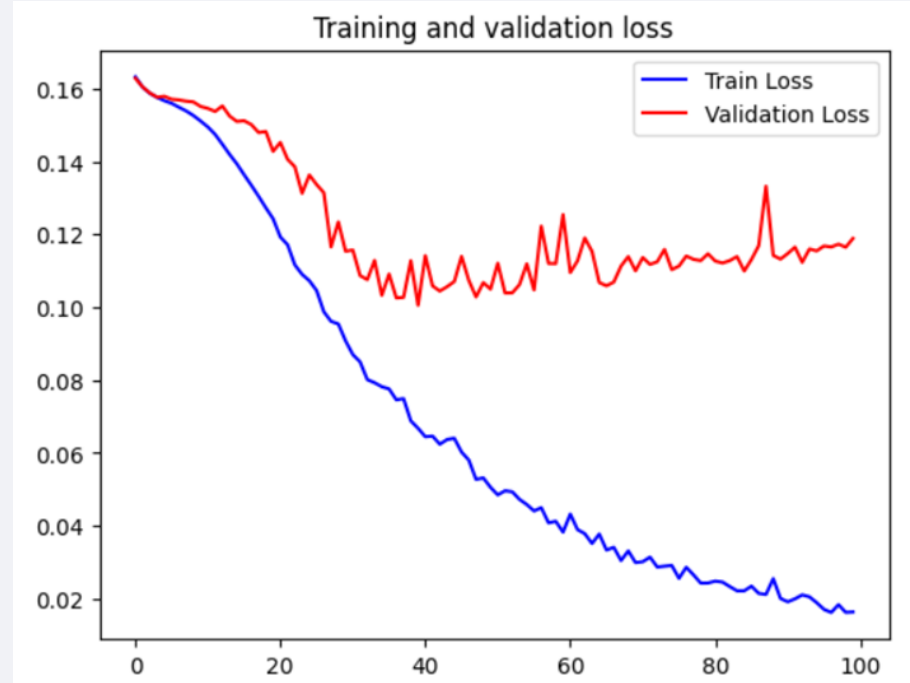
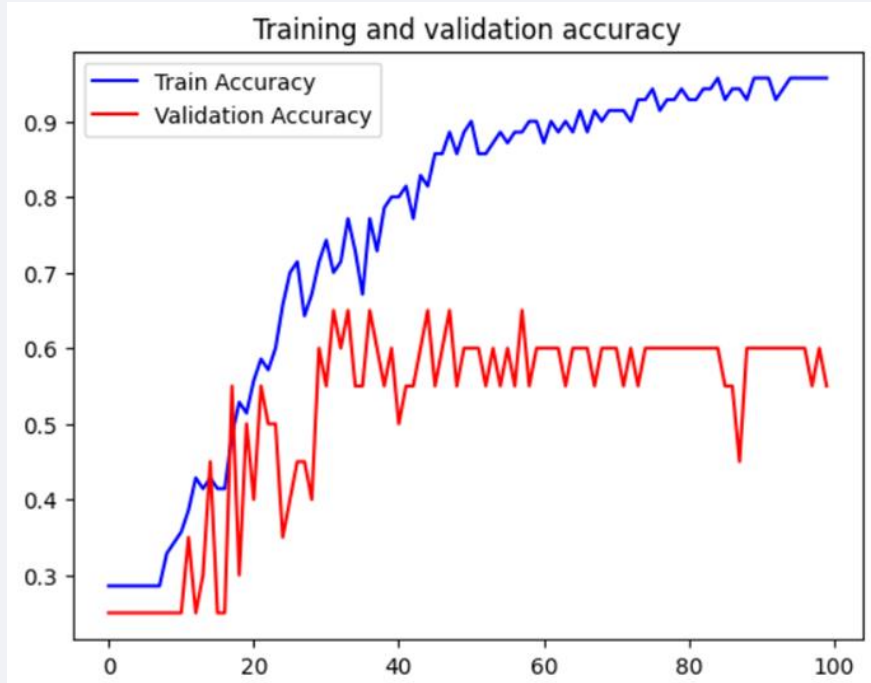
## Dataset Sebelumnya



# Dataset Kelas



# Dataset Kami

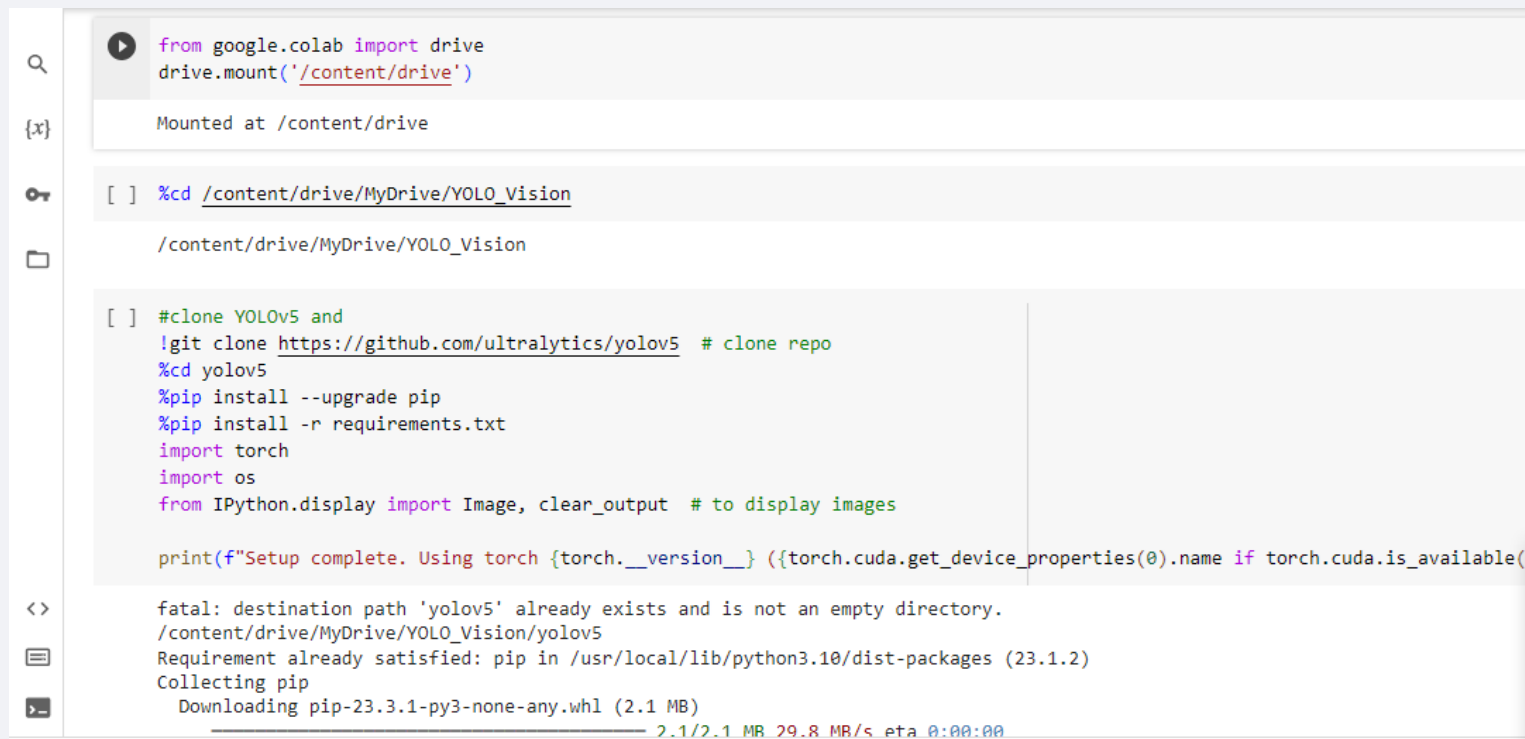


---

# Percobaan ke 2

# Menggunakan Yolo

# Processing Using Yolo



The image shows a Google Colab notebook interface. On the left is a sidebar with icons for search, file explorer, and other tools. The main area displays a code cell with the following content:

```
from google.colab import drive
drive.mount('/content/drive')
```

Below the code, the output shows the drive is mounted at `/content/drive`. The next code cell contains:

```
[ ] %cd /content/drive/MyDrive/YOLO_Vision
```

The output for this cell is `/content/drive/MyDrive/YOLO_Vision`. The third code cell contains a series of commands to clone the YOLOv5 repository and install dependencies:

```
[ ] #clone YOLOv5 and
!git clone https://github.com/ultralytics/yolov5 # clone repo
%cd yolov5
%pip install --upgrade pip
%pip install -r requirements.txt
import torch
import os
from IPython.display import Image, clear_output # to display images

print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if torch.cuda.is_available(
```

The output for this cell shows an error: `fatal: destination path 'yolov5' already exists and is not an empty directory.` This is followed by the output of the `%pip` commands, which shows that the requirements are already satisfied and pip is collecting packages. The final output shows the download progress for `pip-23.3.1-py3-none-any.whl` (2.1 MB).



# Processing Using Yolo

```
[ ] !unzip dataset.zip -d /content/drive/MyDrive/YOLO_Vision/yolov5
```

Archive: dataset.zip

```
  creating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/
  creating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/20231109_183136.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/20231109_183151.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/20231109_183454.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/20231109_183459.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/IMG_20231109_182120.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/IMG_20231109_182727.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/IMG_20231109_182840.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/IMG_20231109_182854.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/IMG_20231109_182927.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/IMG_20231109_182944.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/IMG_20231109_182952.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/IMG_20231109_183018.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/IMG_20231109_183104.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/test/IMG_20231109_183122.jpg
  creating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/train/
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/train/20231109_181321(1).jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/train/20231109_181321.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/train/20231109_181332.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/train/20231109_181350.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/train/20231109_181524.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/train/20231109_181526.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/train/20231109_181553.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/train/20231109_181557.jpg
 inflating: /content/drive/MyDrive/YOLO_Vision/yolov5/dataset/images/train/20231109_181621.jpg
```

# Processing Using Yolo

```
[ ] %cd /content/drive/MyDrive/YOLO_Vision/yolov5
```

```
/content/drive/MyDrive/YOLO_Vision/yolov5
```

Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
158/349	2.82G	0.02314	0.01079	0.002028	15	416: 100% 5/5 [01:15<00:00, 15.12s/it]
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% 1/1 [00:00<00:00, 1.53it/s]
	all	20	31	0.899	0.805	0.814 0.534
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
159/349	2.82G	0.02289	0.009525	0.003646	14	416: 100% 5/5 [01:31<00:00, 18.24s/it]
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% 1/1 [00:00<00:00, 1.71it/s]
	all	20	31	0.912	0.808	0.822 0.572
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
160/349	2.82G	0.02378	0.01015	0.002336	17	416: 100% 5/5 [01:16<00:00, 15.33s/it]
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% 1/1 [00:00<00:00, 2.69it/s]
	all	20	31	0.956	0.859	0.878 0.558
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
161/349	2.82G	0.02618	0.009896	0.001849	18	416: 100% 5/5 [01:27<00:00, 17.51s/it]
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% 1/1 [00:00<00:00, 1.66it/s]
	all	20	31	0.904	0.801	0.803 0.521
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
162/349	2.82G	0.02481	0.009632	0.00197	47	416: 60% 3/5 [00:41<00:27, 13.75s/it]

# Result using yolo

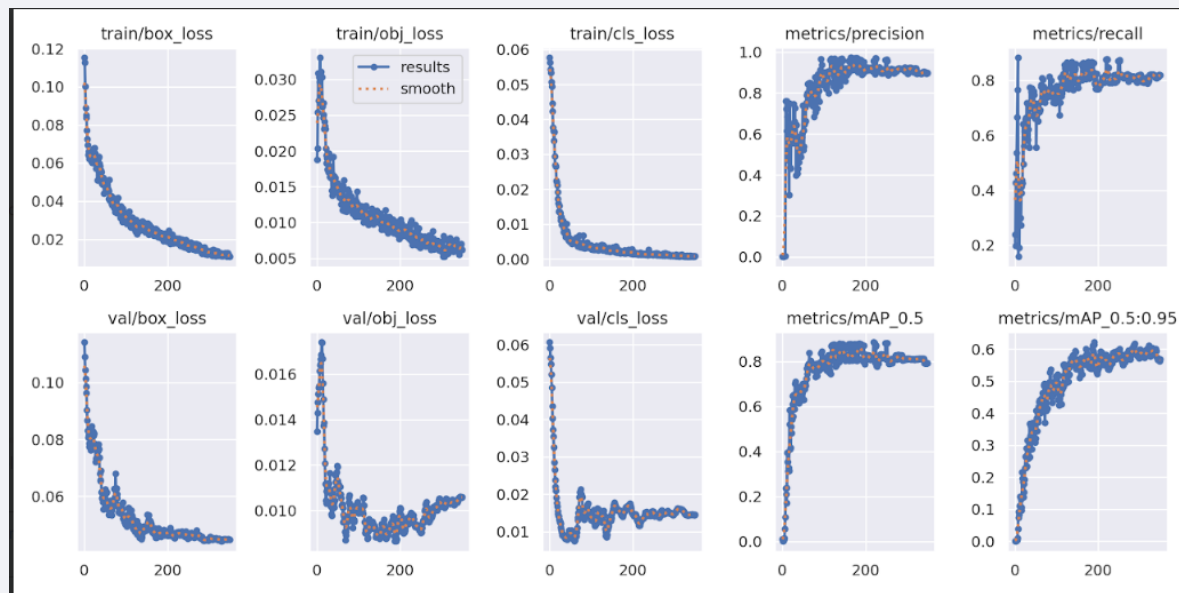
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
234/349	2.74G	0.01811	0.00789	0.001332	14	416: 100% 5/5 [01:13<00:00, 14.76s/it]
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% 1/1 [00:00<00:00, 2.29it/s]
	all	20	31	0.916	0.798	0.811 0.555
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
235/349	2.74G	0.01981	0.008608	0.001126	19	416: 100% 5/5 [01:25<00:00, 17.05s/it]
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% 1/1 [00:00<00:00, 1.63it/s]
	all	20	31	0.913	0.801	0.829 0.588
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
236/349	2.74G	0.01847	0.009666	0.001218	24	416: 100% 5/5 [01:08<00:00, 13.69s/it]
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% 1/1 [00:00<00:00, 1.42it/s]
	all	20	31	0.911	0.803	0.832 0.577
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
237/349	2.74G	0.01818	0.007652	0.001182	16	416: 100% 5/5 [01:27<00:00, 17.53s/it]
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% 1/1 [00:00<00:00, 2.60it/s]
	all	20	31	0.908	0.804	0.829 0.571
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
238/349	2.74G	0.01605	0.00783	0.002691	14	416: 100% 5/5 [01:14<00:00, 14.80s/it]
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% 1/1 [00:00<00:00, 1.66it/s]
	all	20	31	0.896	0.82	0.833 0.581
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size

```
Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 212 layers, 20873139 parameters, 0 gradients, 47.9 GFLOPs
      Class      Images  Instances      P          R      mAP50  mAP50-95: 100% 1/1 [00:00<00:00, 1.01it/s]
      all         20         31         0.9         0.811         0.86         0.621
      koridor      20          5         0.935         0.8         0.802         0.593
      lift         20          5         0.941         1         0.995         0.637
      pintu darurat 20          4          1         0.996         0.995         0.731
      tangga       20          4         0.92         1         0.995         0.759
      toilet       20          3         0.603         0.667         0.863         0.669
      apar         20         10          1         0.405         0.511         0.338
Results saved to runs/train/exp
```

---

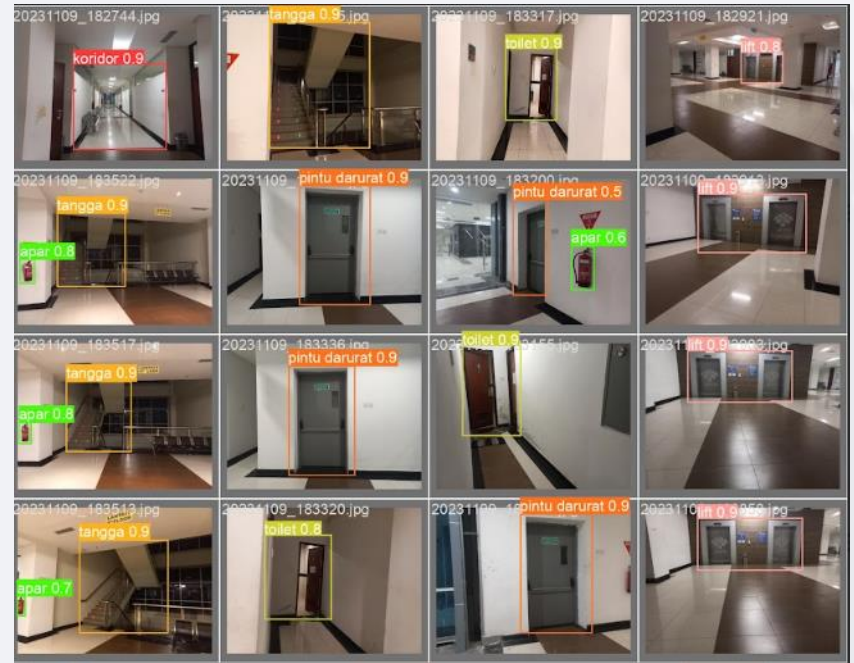
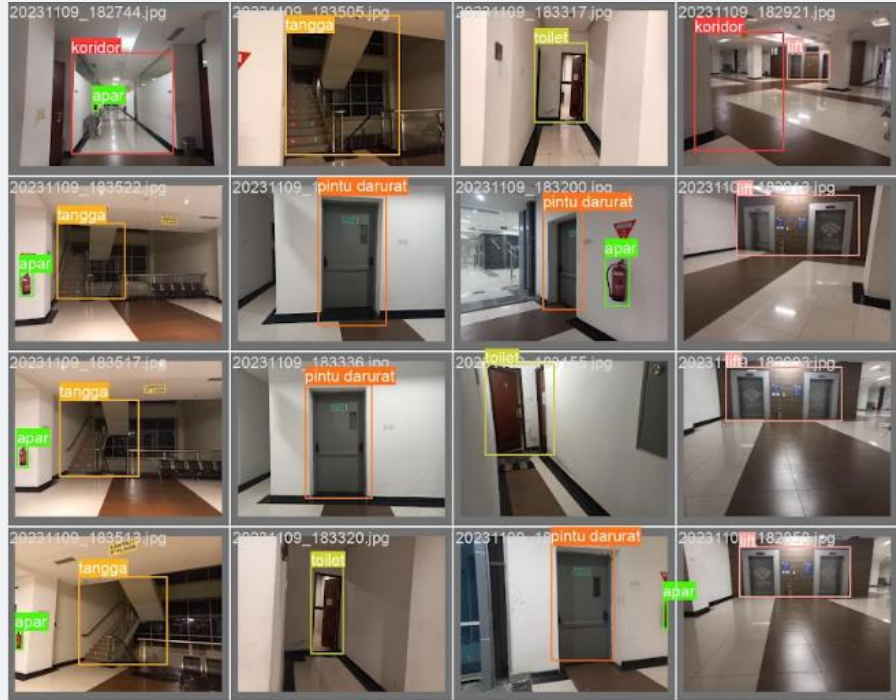
Result using yolo

Result with yolo





# Result using yolo



## Perbandingan dengan metode yang digunakan kelompok lain

Kelompok	Metode	Dataset yang digunakan	Loss	Hasil Akurasi
Kelompok 1	CNN custom 2 layer	Ada 8 class names ['apar': 0, 'kelas': 1, 'koridor': 2, 'lift': 3, 'orang': 4, 'pintu_darurat': 5, 'pintu_ruang': 6, 'tangga': 7]	0.0169	0.936
	Yolo	Ada 6 class names: ["koridor", "lift", "pintu darurat", "tangga", "toilet", "apar"]	0.0007749	0.83
Kelompok 2	CNN custom	Ada 5 class names: ["koridor", "lift", "pintu darurat", "pintu_ruangan", "tangga"]	0.02	0.98
Kelompok 3	VGG16	-	0.0645	0.982

---

## LinkCode

cnncustom:[https://colab.research.google.com/drive/17cvyHYcCU9Zd\\_8FAL6VYJLNl-Y6mBq2C?usp=drive\\_link](https://colab.research.google.com/drive/17cvyHYcCU9Zd_8FAL6VYJLNl-Y6mBq2C?usp=drive_link)

yolo : [https://colab.research.google.com/drive/1\\_9hkuF6cve2F\\_V\\_5QUblb-RFirSlYtTL#scrollTo=DZM4Mvp8j-gp](https://colab.research.google.com/drive/1_9hkuF6cve2F_V_5QUblb-RFirSlYtTL#scrollTo=DZM4Mvp8j-gp)



---

Terimakasih

