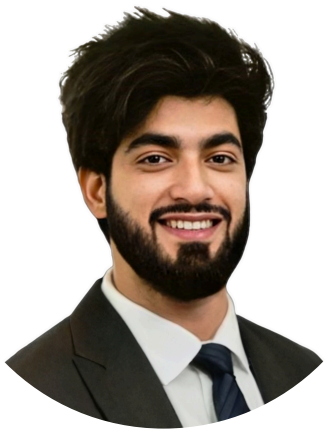

Tajamul Khan

Computer Vision Cheat Sheet

@Tajamulkhann

Computer Vision Basics

Image Processing – Manipulating and analyzing images
Object Detection – Identifying objects in images
Image Classification – Assigning labels to images
Segmentation – Dividing images into meaningful regions
Face Recognition – Identifying faces in images

@Tajamulkhann

Image Preprocessing

- Convert images to grayscale
- Resize images
- Normalize pixel values
- Apply filters (Gaussian Blur, Edge Detection)

```
import cv2
import numpy as np
image = cv2.imread("image.jpg")
# Load image
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Convert to grayscale
resized = cv2.resize(gray, (128, 128))
# Resize image
normalized = resized / 255.0
# Normalize pixel values
```

@Tajamulkhann

Edge Detection & Image Filtering

- **Sobel & Canny Edge Detection**
- **Blurring** (Gaussian, Median, Bilateral)

```
edges = cv2.Canny(gray, 100, 200)
# Canny Edge Detection
blurred = cv2.GaussianBlur(gray,
(5,5), 0)  # Gaussian Blur
```

@Tajamulkhann

Image Augmentation

- **Rotation, Flipping, Zooming, Shearing**
- Used to increase dataset diversity in deep learning

```
from
tensorflow.keras.preprocessing.image
import ImageDataGenerator

datagen =
ImageDataGenerator(rotation_range=30,
horizontal_flip=True)
augmented_image =
datagen.random_transform(image)
```

@Tajamulkhann

Image Classification (CNNs)

Classifies images into categories

```
import tensorflow as tf

model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3,3),
activation='relu', input_shape=(128,128,3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=
['accuracy'])
```

@Tajamulkhann

Object Detection

- **SSD, Faster R-CNN**
- **YOLO (You Only Look Once)** – Real-time object detection

```
import cv2

net =
cv2.dnn.readNet("yolov3.weights",
"yolov3.cfg")
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1]
for i in
net.getUnconnectedOutLayers()]
```

Face Detection & Recognition

- **Haar Cascades** – Pre-trained classifiers for detecting faces

```
face_cascade =
cv2.CascadeClassifier(cv2.data.haarcascade
s + "haarcascade_frontalface_default.xml")
faces =
face_cascade.detectMultiScale(gray, 1.1,
4)

for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x + w, y
+ h), (255, 0, 0), 2)
```

@Tajamulkhann

Image Segmentation

Divides an image into meaningful parts

```
import cv2


ret, thresh = cv2.threshold(gray, 127,
255, cv2.THRESH_BINARY)
contours, hierarchy =
cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(image, contours, -1,
(0,255,0), 3)
```

@Tajamulkhann

Optical Character Recognition (OCR)

Extracts text from images

```
import pytesseract

text =
pytesseract.image_to_string
(gray)
print(text)
```

@Tajamulkhann

Generative Models

- **(GANs, Autoencoders)**
- **GANs (Generative Adversarial Networks)** – Generate new images)

```
from tensorflow.keras.layers import Dense,
LeakyReLU
from tensorflow.keras.models import
Sequential

generator = Sequential([
    Dense(256, input_dim=100),
    LeakyReLU(alpha=0.2),
    Dense(512, activation='relu'),
    Dense(1024, activation='relu'),
    Dense(784, activation='tanh')
])
```

@Tajamulkhann

Found Helpful ?

Repost

Follow for more!