6131881421 YOSCHANIN PULSIRIVONG

Clothing Classification and Recommendation

2143488.i Big Data and Artificial Intelligence (ICE)

Goals

What I aim to achieve

- Train with live data or data from Kaggle
- Get live photos from webcam device
- Successful clothing classification
- Successful clothing segmentation
- Computer vision (Image-to-Text)
- Filter our profanity
- Recommendation

Useful materials...

Research and Guides

Fashion MNIST in Python With TensorFlow Keras | Machine Learning Tutorial

11K views • 3 years ago

Qubits With Chris

Hi all, In this video I'll kick off talking about **Machine Learning** in depth. If you see a...

Import Tensorflow | Import the Data | Training... 12 moments

Deep Learning Tutorial 2 - Fashion MNIST Data Classification

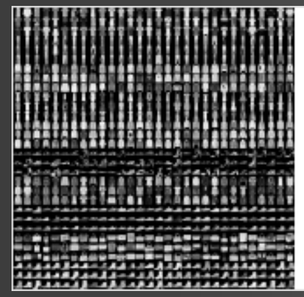
4.1K views • 1 year ago

KGP Talkie

Hi, In this tutorial you will learn, how to classify **fashion mnist** dataset using deep learning techniques with TensorFlow framework.

4K

https://www.youtube.com/watch?v=KDUnfk14aGk
https://www.youtube.com/watch?v=wSml2fEFuzs&t=944s



Fashion MNIST

An MNIST-like dataset of 70,000 28x28 labeled fashion images

kaggle.com

:www.kaggle.com/datasets/zalando-research/fashionmnist?select=fashion-mnist_test.csv

Text detection with Python and Opencv | OCR using EasyOCR | Computer vision tutorial

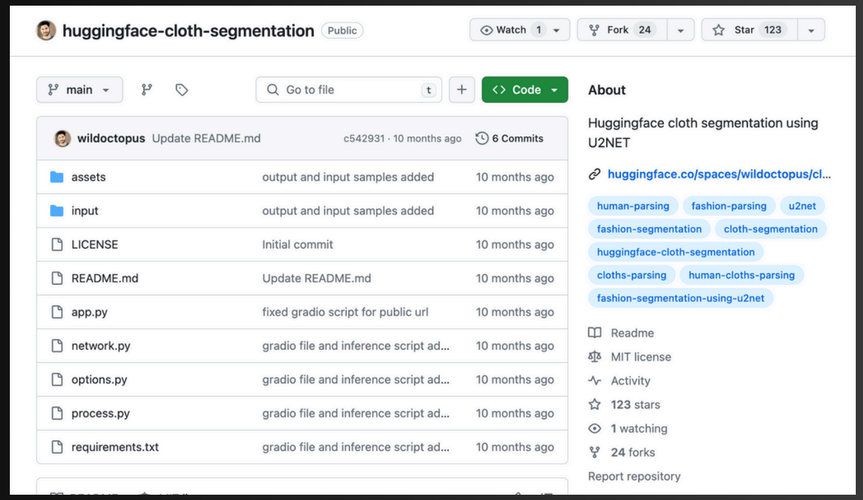
30K views • 1 year ago

Computer vision engineer

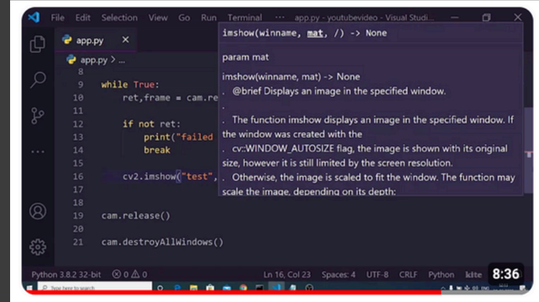
In this video I show you how to make an optical character recognition (OCR) using ...

Intro | Project overview | Read an image | Create ... 9 chapters

https://www.youtube.com/watch?v=n-8oCPjpEvM&t=325s



https://github.com/wildoctopus/huggingface-cloth-segmentation



Python OpenCV Tutorial To Capture Images From Webcam Full Project For Beginners

64K views • 3 years ago

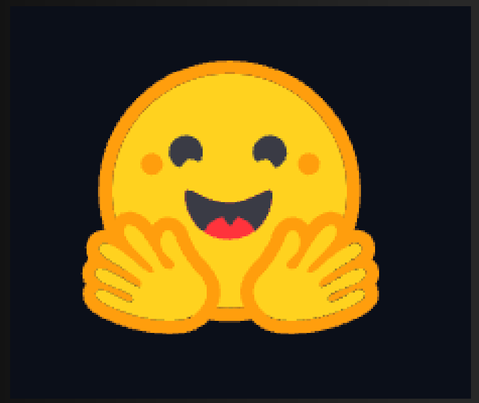
Coding Shiksha

python #opencv #pythonproject Welcome Folks My name is Gautam and Welcome to Coding Shiksha a Place for All ...

https://www.youtube.com/watch?v=IhRfqiC29Ds&t=424s

ChatGPT

Install & Import Dependencies



```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from keras.utils import to_categorical
from skimage import io
import csv
import os
from PIL import Image
import pandas as pd
import cv2
from IPython.display import Image
import pytesseract as pyt
import easyocr
from better_profanity import profanity
```

Taking the Photo


```python
folder_path = "/Users/chae/Desktop/BigDataProj/input"
if not os.path.exists(folder_path):
    os.makedirs(folder_path)

cam = cv2.VideoCapture(0)
cv2.namedWindow("Clothing Recommendation Machine")
img_counter = 0

while True:
    ret, frame = cam.read()
    if not ret:
        print("failed to grab frame")
        break
    cv2.imshow("test", frame)
    k = cv2.waitKey(1)
    if k%256 == 27:
        print("Escape hit, closing...")
        break
    elif k%256 == 32:
        img_name = os.path.join(folder_path, "opencv_frame_{}.png".format(img_counter))
        cv2.imwrite(img_name, frame)
        print("Screenshot Taken")
        img_counter += 1

cam.release()
cv2.destroyAllWindows()
```

‘Spacebar’ to take image
‘esc’ to close

Handling Data

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()
train_indices = np.where(y_train != 8)
test_indices = np.where(y_test != 8)

x_train_filtered, y_train_filtered = x_train[train_indices], y_train[train_indices]
x_test_filtered, y_test_filtered = x_test[test_indices], y_test[test_indices]

x_train = x_train / 255
x_test = x_test / 255
```

```
model = tf.keras.models.Sequential([ tf.keras.layers.Flatten(input_shape=(28, 28, 1)), tf.keras.layers.Dense(128, activation='relu'), tf.keras.layers.Dense(10, activation='softmax') ])
```

- First model...

Handling Data

- Second model...

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()
train_indices = np.where(y_train != 8)
test_indices = np.where(y_test != 8)

x_train_filtered, y_train_filtered = x_train[train_indices], y_train[train_indices]
x_test_filtered, y_test_filtered = x_test[test_indices], y_test[test_indices]

x_train = x_train / 255
x_test = x_test / 255
```

```
model = tf.keras.models.Sequential([ tf.keras.layers.Flatten(input_shape=(28, 28, 1)), tf.keras.layers.Dense(128, activation='relu'), tf.keras.layers.Dense(10, activation='softmax') ])
```

```
import tensorflow as tf

model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])
```

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/keras/src/layers/convolutional/base_conv.py:99: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(
```

Compiling and Training Model

```
y_train_one_hot_filtered = to_categorical(y_train_filtered)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train_filtered, y_train_one_hot_filtered, epochs=10)
```

```
Epoch 3/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1s 698us/step - accuracy: 0.8414 - loss: 0.4599
Epoch 4/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1s 738us/step - accuracy: 0.8387 - loss: 0.4672
Epoch 5/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1s 818us/step - accuracy: 0.8430 - loss: 0.4561
Epoch 6/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1s 750us/step - accuracy: 0.8433 - loss: 0.4532
Epoch 7/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1s 742us/step - accuracy: 0.8465 - loss: 0.4482
Epoch 8/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1s 759us/step - accuracy: 0.8437 - loss: 0.4536
Epoch 9/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1s 823us/step - accuracy: 0.8447 - loss: 0.4619
Epoch 10/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1s 723us/step - accuracy: 0.8459 - loss: 0.4575

<keras.src.callbacks.history.History at 0x34f56c200>
```

```
y_train_one_hot_filtered = to_categorical(y_train_filtered)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train_filtered, y_train_one_hot_filtered, epochs=10)
```

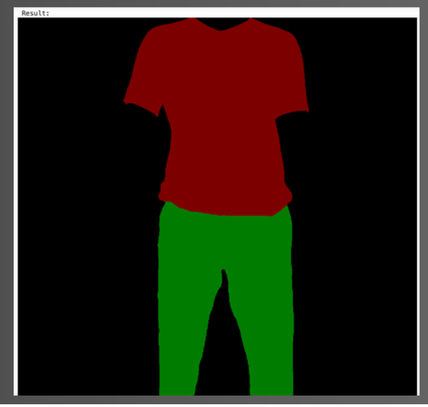
```
Epoch 1/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 13s 7ms/step - accuracy: 0.7544 - loss: 0.8650
Epoch 2/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 12s 7ms/step - accuracy: 0.8694 - loss: 0.3539
Epoch 3/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 12s 7ms/step - accuracy: 0.8843 - loss: 0.3060
Epoch 4/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 12s 7ms/step - accuracy: 0.8955 - loss: 0.2824
Epoch 5/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 12s 7ms/step - accuracy: 0.9051 - loss: 0.2574
Epoch 6/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 13s 8ms/step - accuracy: 0.9060 - loss: 0.2485
Epoch 7/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 13s 8ms/step - accuracy: 0.9143 - loss: 0.2282
Epoch 8/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 13s 8ms/step - accuracy: 0.9168 - loss: 0.2157
Epoch 9/10
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 13s 7ms/step - accuracy: 0.9232 - loss: 0.2014
```

Clothing Segmentation

```
!python3 process.py --image '/Users/chae/Desktop/BigDataProj/input/Screenshot 2024-05-05 at 17.59.2
```

```
Model already exists.
----checkpoints loaded from path: model/cloth_segm.pth----
[W NNPACK.cpp:53] Could not initialize NNPACK! Reason: Unsupported hardware.
/usr/local/lib/python3.7/site-packages/torch/nn/functional.py:3734: UserWarning: nn.functional.ups
ample is deprecated. Use nn.functional.interpolate instead.
  warnings.warn("nn.functional.upsample is deprecated. Use nn.functional.interpolate instead.")
```

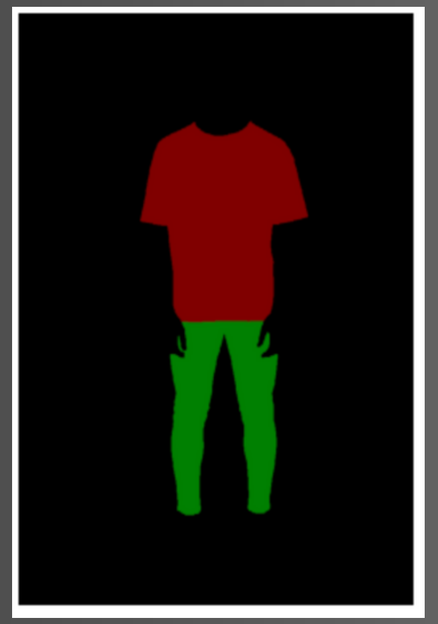
```python
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Load the image
img = mpimg.imread('/Users/chae/Desktop/BigDataProj/output/cloth_segm/final_seg.png')

# Display the image
plt.imshow(img)
plt.axis('off')
plt.show()
```

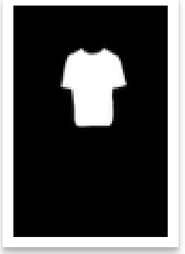


1.png

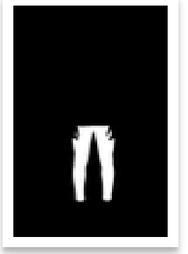
2.png

Clothing Segmentation



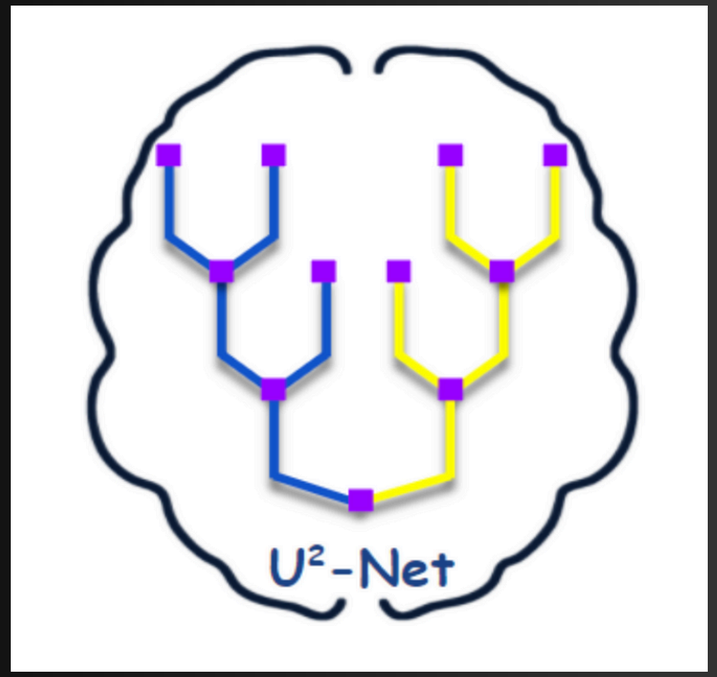


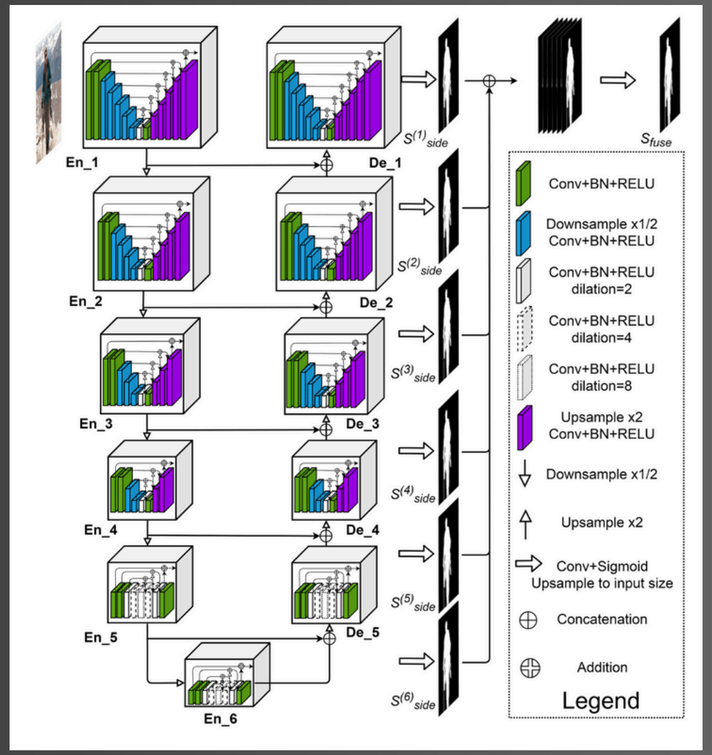
1.png



2.png

Clothing Segmentation





U2-Net: U Square Net - Nested U-Structure for Salient Object Detection
https://github.com/xuebinqin/U-2-Net

Image Processing

- Rescaling
- Convert to brightness value
- Save

```python
def resize_image(image_path, target_size):
    image = Image.open(image_path)
    image = image.convert("L")
    threshold = 200
    # Reversing the brightness scale
    image = image.point(lambda p: 255 - p if p < threshold else 0)
    image = image.resize(target_size)
    return image

def save_image(image, output_path):
    image.save(output_path)

def resize_images_in_directory(input_dir, output_dir, target_size):
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    for filename in os.listdir(input_dir):
        if filename.endswith(('.png', '.jpg', '.jpeg')):
            input_image_path = os.path.join(input_dir, filename)
            output_image_path = os.path.join(output_dir, filename)
            resized_image = resize_image(input_image_path, target_size)
            save_image(resized_image, output_image_path)
            print(f"Resized and saved {input_image_path} to {output_image_path}")

def main():
    input_dir = "output/alpha"
    output_dir = "rescaled_images"
    target_size = (28, 28)
    resize_images_in_directory(input_dir, output_dir, target_size)
    print("All images resized and saved successfully!")

if __name__ == "__main__":
    main()
```

```
Resized and saved output/alpha/2.png to rescaled_images/2.png
Resized and saved output/alpha/1.png to rescaled_images/1.png
All images resized and saved successfully!
```

Prediction

- Flatten to 1D array
- Save to CSV
- Load pixel data
- Convert to correct format
- Make prediction
- Repeat for each row in CSV file

```python
class_names = {
    0: "T-shirt/top",
    1: "Trouser",
    2: "Pullover",
    3: "Dress",
    4: "Coat",
    5: "Sandal",
    6: "Shirt",
    7: "Sneaker",
    8: "Bag",
    9: "Ankle boot"
}

input_directory = "rescaled_images"
csv_path = "output_image_scikit.csv"
image_files = [f for f in os.listdir(input_directory) if f.endswith(('.png', '.jpg', '.jpeg'))]
pixels = []

for idx, image in enumerate(image_files, start=1):
    image_path = os.path.join(input_directory, image)
    img_gray = io.imread(image_path, as_gray=True)
    pixel_array = img_gray.flatten()
    # pixel_array = 255 - pixel_array
    pixels.append(np.concatenate(([image], pixel_array)))
    print(f"Converted image {idx}: {image}")

np.savetxt(csv_path, pixels, delimiter=',', fmt='%s')
print(f"CSV file saved at: {csv_path}")

test = pd.read_csv('output_image_scikit.csv', header=None, usecols=range(1, 785))
test = test.values
test = test.reshape(-1, 28, 28, 1)

predictions = []

for idx, row in enumerate(test):
    file_name = pixels[idx][0]
    prediction = model.predict(np.array([row]))
    predicted_class = np.argmax(prediction)
    predicted_class_name = class_names[predicted_class]
    print(f"File: {file_name}, Predicted class: {predicted_class_name}")
    predictions.append(predicted_class_name)
```

```
Converted image 1: 2.png
Converted image 2: 1.png
CSV file saved at: output_image_scikit.csv
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
File: 2.png, Predicted class: Shirt
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
File: 1.png, Predicted class: Shirt
```

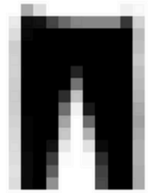
Prediction

- Make prediction and check if correct
  - **Relatively low accuracy**
- Print image
- Check if suitable

```
test = pd.read_csv('output_image_scikit.csv', header=None, usecols=range(1, 785))
test = test.values
test = test.reshape(-1, 28, 28, 1)
clothing_detected = False

for idx, row in enumerate(test):
    file_name = pixels[idx][0]
    prediction = model.predict(np.array([row]))
    predicted_class = np.argmax(prediction)
    predicted_class_name = class_names[predicted_class]

    # Check if either trousers or dress is detected
    if 'trousers' in predicted_class_name.lower() or 'dress' in predicted_class_name.lower():
        clothing_detected = True
    if 'T-shirt/top' in predicted_class_name.lower():
        clothing_detected = False

    image_data = row.reshape(28, 28)
    # Print the file name and predicted class
    print(f"File: {file_name}, Predicted class: {predicted_class_name}")

    plt.imshow(image_data, cmap='gray')
    plt.title(f'File: {file_name}, Predicted class: {predicted_class_name}')
    plt.axis('off')
    plt.show()
```

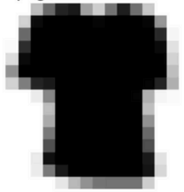
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
File: 2.png, Predicted class: Shirt
```

File: 2.png, Predicted class: Shirt



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step
File: 1.png, Predicted class: Shirt
```

File: 1.png, Predicted class: Shirt



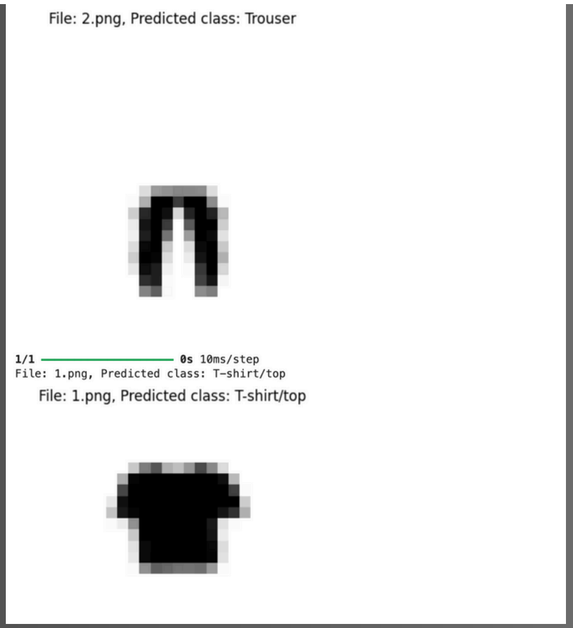
Prediction

```python
class_names = {
    0: "T-shirt/top",
    1: "Trouser",
    2: "Pullover",
    3: "Dress",
    4: "Coat",
    5: "Sandal",
    6: "Shirt",
    7: "Sneaker",
    8: "Bag",
    9: "Ankle boot"
}

input_directory = "rescaled_images"
csv_path = "output_image_scikit.csv"
image_files = [f for f in os.listdir(input_directory) if f.endswith(('.png', '.jpg', '.jpeg'))]
pixels = []

for idx, image in enumerate(image_files, start=1):
    image_path = os.path.join(input_directory, image)
    img_gray = io.imread(image_path, as_gray=True)
    pixel_array = img_gray.flatten()
    # pixel_array = 255 - pixel_array
    pixels.append(np.concatenate(([image], pixel_array)))
    print(f"Converted image {idx}: {image}")

np.savetxt(csv_path, pixels, delimiter=',', fmt='%s')
print(f"CSV file saved at: {csv_path}")

test = pd.read_csv('output_image_scikit.csv', header=None, usecols=range(1, 785))
test = test.values
test = test.reshape(-1, 28, 28, 1)

predictions = []

for idx, row in enumerate(test):
    file_name = pixels[idx][0]
    prediction = model.predict(np.array([row]))
    predicted_class = np.argmax(prediction)
    predicted_class_name = class_names[predicted_class]
    print(f"File: {file_name}, Predicted class: {predicted_class_name}")
    predictions.append(predicted_class_name)
```

```
Converted image 1: 2.png
Converted image 2: 1.png
CSV file saved at: output_image_scikit.csv
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
File: 2.png, Predicted class: Trouser
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
File: 1.png, Predicted class: T-shirt/top
```

```python
test = pd.read_csv('output_image_scikit.csv', header=None, usecols=range(1, 785))
test = test.values
test = test.reshape(-1, 28, 28, 1)
clothing_detected = False

for idx, row in enumerate(test):
    file_name = pixels[idx][0]
    prediction = model.predict(np.array([row]))
    predicted_class = np.argmax(prediction)
    predicted_class_name = class_names[predicted_class]

    # Check if either trousers or dress is detected
    if 'trousers' in predicted_class_name.lower() or 'dress' in predicted_class_name.lower():
        clothing_detected = True
    if 'T-shirt/top' in predicted_class_name.lower():
        clothing_detected = False

    image_data = row.reshape(28, 28)
    # Print the file name and predicted class
    print(f"File: {file_name}, Predicted class: {predicted_class_name}")

    plt.imshow(image_data, cmap='gray')
    plt.title(f'File: {file_name}, Predicted class: {predicted_class_name}')
    plt.axis('off')
    plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
File: 2.png, Predicted class: Trouser
```



Much more accurate!

Computer Vision

- **Text to Image**
- Draw bounding boxes

```
image_path = '/Users/chae/Desktop/BigDataProj/input/Screenshot 2024-05-05 at 17.59.24.png'
img = cv2.imread(image_path)

reader = easyocr.Reader(['en'], gpu=False)

text = reader.readtext(img)
store = []

for t in text:
    print(t)
    store.append(t)
    bbox, text, score = t
    pt1 = (int(bbox[0][0]), int(bbox[0][1]))
    pt2 = (int(bbox[2][0]), int(bbox[2][1]))
    cv2.rectangle(img, pt1, pt2, (0, 255, 0), 5)

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

```
Using CPU. Note: This module is much faster with a GPU.
([[470, 146], [673, 146], [673, 209], [470, 209]], 'Standard', 0.4882621699467346)
([[467, 194], [672, 194], [672, 251], [467, 251]], 'Chartered', 0.9371480032896289)
```

Final Result

```
store_words = [entry[1] for entry in store]
combined_sentence = ' '.join(store_words)
words = combined_sentence.split()

if len(words) < 5 and clothing_detected:
    if profanity.contains_profanity(combined_sentence):
        print('Inappropriate Clothing')
    else:
        print('Suitable Clothing')
else:
    print('Inappropriate Clothing')
```

```
Inappropriate Clothing
```

THANK YOU FOR LISTENING