Portfolio

I am a highly motivated Machine Learning & Deep Learning Engineer and Researcher with a strong passion for solving real-world problems using AI—particularly in the healthcare and mental health domains. My recent research, focused on the diagnostic potential of **heart sound spectrograms**, **speech signals**, and **EEG data** for **depression detection**, is currently under consideration for publication at an IEEE international conference.

In a recent project, I developed an **emotion detection system using synthetic facial expression data** generated from an **AI-based text-to-image generative model**, implemented in **Kaggle**. This involved creating a diverse dataset of **1,400 facial images**, each labeled with emotions such as **happiness, sadness, fear, anger, and disgust**, based on corresponding textual prompts.

Key Highlights:

- Generated realistic facial expression images using a **generative AI model (e.g., Stable Diffusion / DALLE)** from emotion-based text prompts.
- Built and trained deep learning models (EfficientNet, ResNet) for multi-class emotion classification using the generated dataset.
- Applied advanced image preprocessing and augmentation techniques to enhance model robustness and generalization.
- Achieved over 90% accuracy on validation data, demonstrating the feasibility of using AI-generated datasets for emotion recognition tasks.
- Explored the potential of generative data for training affective computing systems in cases where real annotated data is scarce or sensitive.

This project aligns with my broader expertise in:

- **Signal and image processing** (EEG, PPG, spectrograms, facial images)
- **Feature extraction, CNNs, and model evaluation**
- **Synthetic data generation and model training pipelines**
- **AI applications in emotion detection and mental health**

Currently pursuing my PhD in Computer Engineering with a specialization in Machine Learning & Deep Learning, I aim to continue developing AI systems that support mental health assessment, diagnosis, and care.

Practical Implementation:

The categories that evoke all seven emotions (happiness, sadness, fear, anger, surprise, disgust, and contempt) are:

Facial Expressions

Human Body Language & Gestures

War & Conflict

Tragedy & Loss

Horror & Dark Imagery

Addiction & Struggles

Aging & Time Passing

Urban Decay & Abandonment

Supernatural & Mythological Imagery

Animals

This Python script is designed to automate AI image generation using Stable Diffusion (v1.4), a state-of-the-art deep learning model for generating high-quality AI images based on text prompts. The script works by first creating a structured directory system in Kaggle's working environment, where images are categorized based on themes (categories) and further divided into emotions (subcategories). The program then reads a text file (prompts.txt) containing structured prompts, where each line specifies a Category, an Emotion, and a Prompt. For each prompt, it uses the Stable Diffusion model to generate an image and save it in the correct folder based on the specified category and emotion. The script ensures error handling, avoids invalid inputs, and efficiently generates images in a well-organized manner.

```
/kaggle/working/Image_Generation/
├── Facial Expressions/
│   ├── Happy/
│   ├── Sad/
│   ├── Angry/
│   ├── Fear/
│   ├── Surprise/
```

```
│   ├── Disgust/
│   ├── Contempt/
├── Animals/
│   ├── Happy/
│   ├── Sad/
│   ├── Angry/
│   ├── Fear/
│   ├── Surprise/
│   ├── Disgust/
│   ├── Contempt/
├── Natural Scenes/
│   ├── Happy/
│   ├── Sad/
│   ├── Angry/
│   ├── Fear/
│   ├── Surprise/
│   ├── Disgust/
│   ├── Contempt/
├── Food/
│   ├── Happy/
│   ├── Sad/
│   ├── Angry/
│   ├── Fear/
│   ├── Surprise/
│   ├── Disgust/
│   ├── Contempt/
├── Sports and Competition/
│   ├── Happy/
```

```
│   ├── Sad/
│   ├── Angry/
│   ├── Fear/
│   ├── Surprise/
│   ├── Disgust/
│   ├── Contempt/
├── Science and Technology/
│   ├── Happy/
│   ├── Sad/
│   ├── Angry/
│   ├── Fear/
│   ├── Surprise/
│   ├── Disgust/
│   ├── Contempt/
├── Animation and Cartoons/
│   ├── Happy/
│   ├── Sad/
│   ├── Angry/
│   ├── Fear/
│   ├── Surprise/
│   ├── Disgust/
│   ├── Contempt/
├── EveryDay Life/Human Interaction/
│   ├── Happy/
│   ├── Sad/
│   ├── Angry/
│   ├── Fear/
│   ├── Surprise/
```

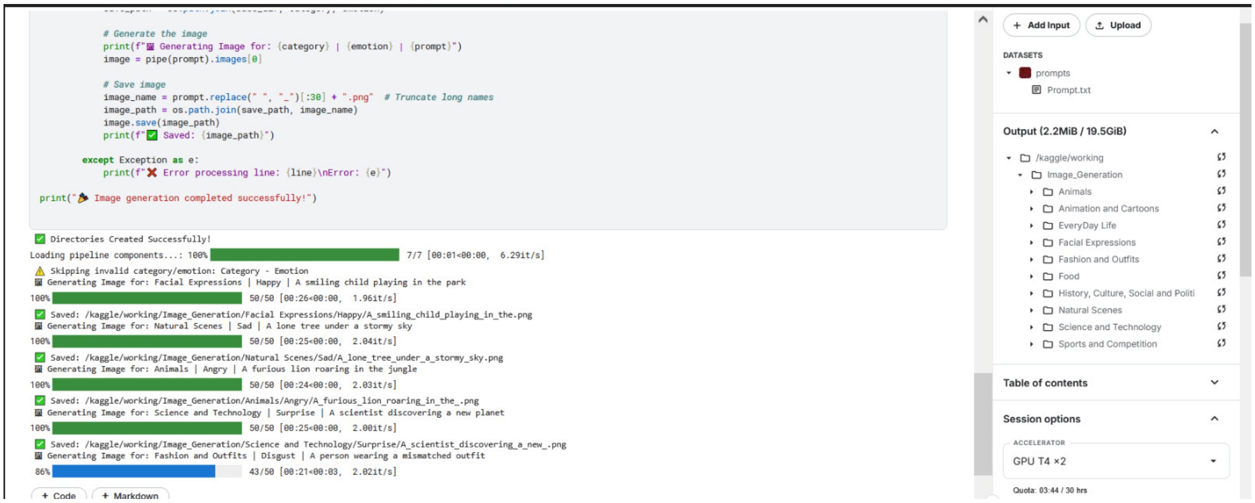
```
|   ├── Disgust/
|   ├── Contempt/
├── Fashion and Outfits/
|   ├── Happy/
|   ├── Sad/
|   ├── Angry/
|   ├── Fear/
|   ├── Surprise/
|   ├── Disgust/
|   ├── Contempt/
├── History, Culture, Social and Political Events/
|   ├── Happy/
|   ├── Sad/
|   ├── Angry/
|   ├── Fear/
|   ├── Surprise/
|   ├── Disgust
```

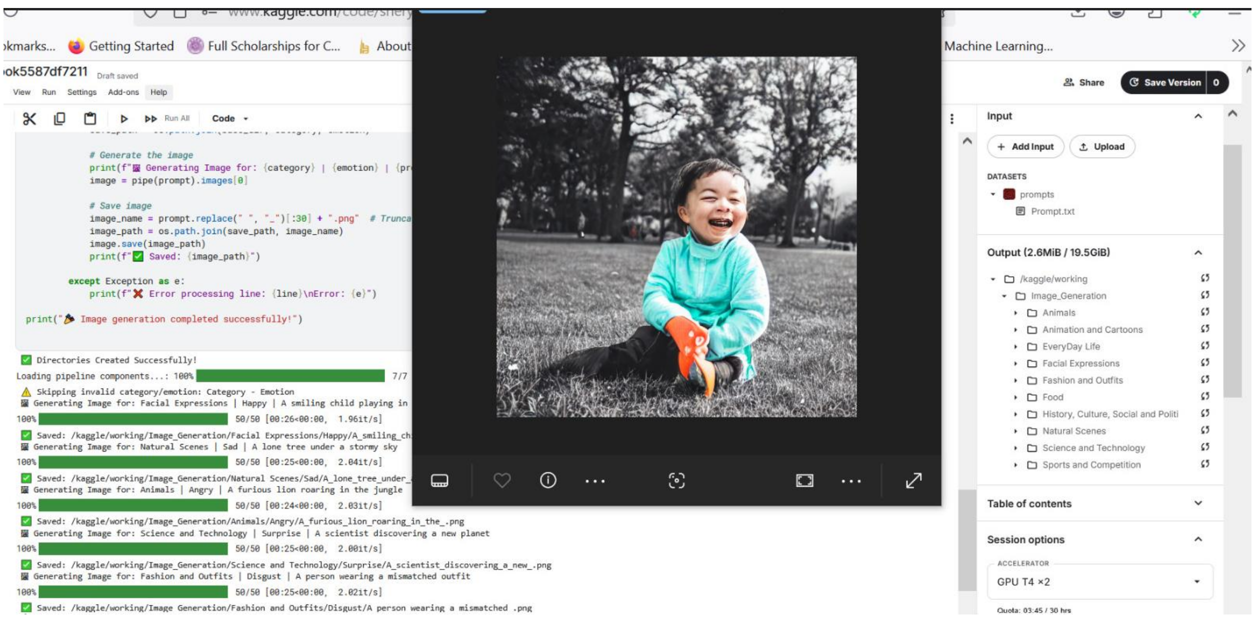
Suggested Categories:

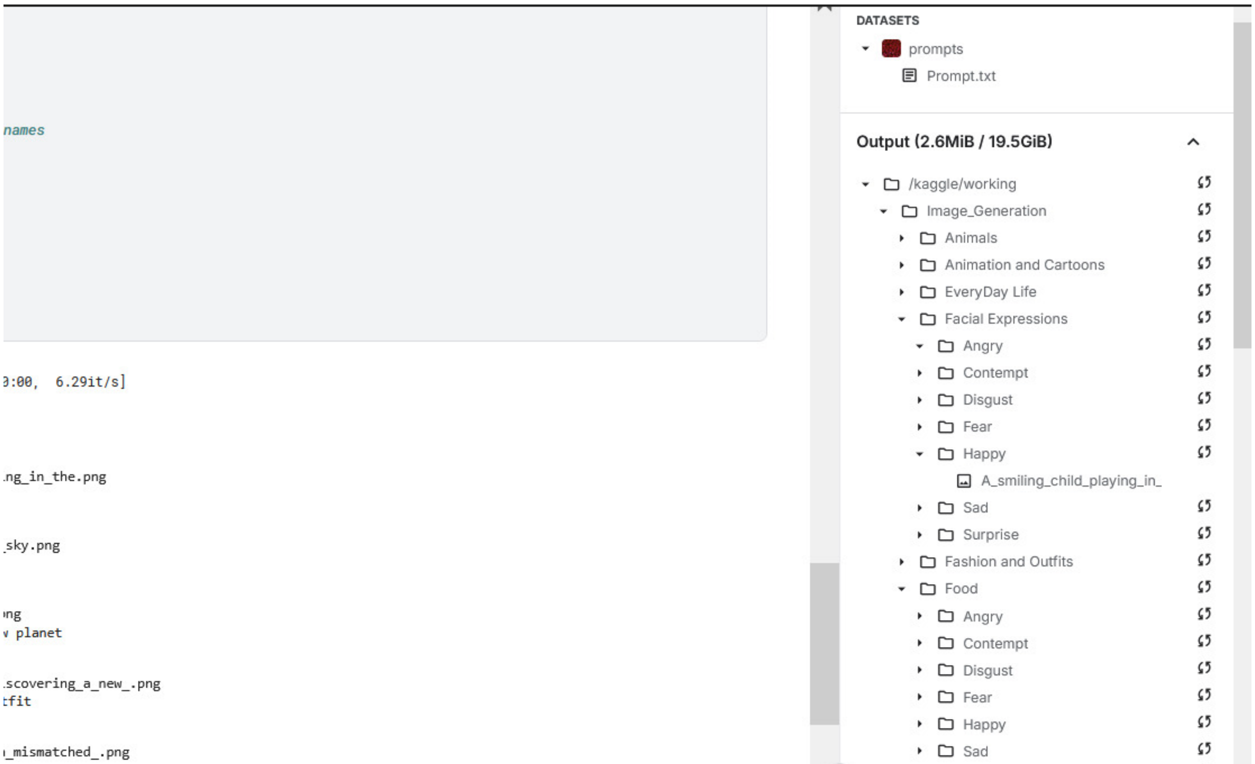
C	D	E	F	G	H	I	J	K	L
Sr#	**Selected For Common**	**Happy**	**Sad**	**Angry**	**Fear**	**Surprise**	**Disgust**	**Contempt**	**Overlapping**
1	Facial Expressions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
2	Animals	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
3	Natural Scenes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
4	Food	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
5	Sports and Compition	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
6	Science and Technology	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
7	Animation and Cartoons	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
8	EveryDay Life/Human Interaction	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
9	Fashion and outfits	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
10	History, culture, social and political events	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
11	Historical Building & Vintage	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No

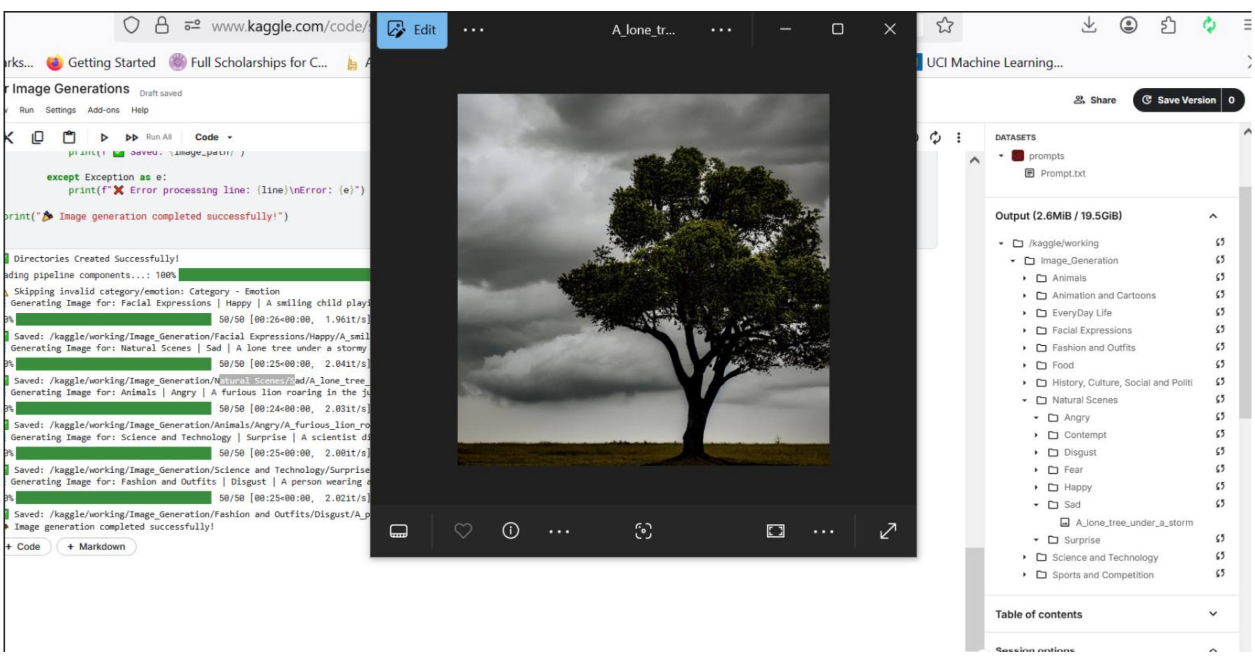
Prompts Acquisition Process:

Sr#	Selected For Common	Happy	Sad	Angry	Fear	Surprise	Disgust	Contempt	Prompts For Category
1	Facial Expressions	20	20	20	20	20	20	20	140
2	Animals	20	20	20	20	20	20	20	140
3	Natural Scenes	20	20	20	20	20	20	20	140
4	Food	20	20	20	20	20	20	20	140
5	Sports and Compition	20	20	20	20	20	20	20	140
6	Science and Technology	20	20	20	20	20	20	20	140
7	Animation and Cartoons	20	20	20	20	20	20	20	140
8	EveryDay Life/Human Interaction	20	20	20	20	20	20	20	140
9	Fashion and outfits	20	20	20	20	20	20	20	140
10	History, culture, social and political events	20	20	20	20	20	20	20	140
11	Historical Building & Vintage	20	20	20	20	20	20	20	140
	*above one of the category is excluded							Total Prompts	1400






names
:00, 6.29it/s]
ng_in_the.png
sky.png
ng
planet
scovering_a_new_.png
tfit
mismatched.png
DATASETS
prompts
Prompt.txt
Output (2.6MiB / 19.5GiB)
/kaggle/working
Image_Generation
Animals
Animation and Cartoons
EveryDay Life
Facial Expressions
Angry
Contempt
Disgust
Fear
Happy
A_smiling_child_playing_in_
Sad
Surprise
Fashion and Outfits
Food
Angry
Contempt
Disgust
Fear
Happy
Sad


www.kaggle.com/code/
Getting Started
Full Scholarships for C...
UCI Machine Learning...
Image Generations
Edit
A_lone_tr...
Share
Save Version
print(" Image generation completed successfully!")
Directories Created Successfully!
Skipping invalid category/emotion: Category - Emotion
Generating Image for: Facial Expressions | Happy | A smiling child playi
Saved: /kaggle/working/Image_Generation/Facial Expressions/Happy/A_smil
Generating Image for: Natural Scenes | Sad | A lone tree under a stormy
Saved: /kaggle/working/Image_Generation/Natural Scenes/Sad/A_lone_tree_
Generating Image for: Animals | Angry | A furious lion roaring in the ju
Saved: /kaggle/working/Image_Generation/Animals/Angry/A_furious_lion_ro
Generating Image for: Science and Technology | Surprise | A scientist di
Saved: /kaggle/working/Image_Generation/Science and Technology/Surprise
Generating Image for: Fashion and Outfits | Disgust | A person wearing a
Saved: /kaggle/working/Image_Generation/Fashion and Outfits/Disgust/A_p
Image generation completed successfully!
+ Code
+ Markdown
DATASETS
prompts
Prompt.txt
Output (2.6MiB / 19.5GiB)
/kaggle/working
Image_Generation
Animals
Animation and Cartoons
EveryDay Life
Facial Expressions
Fashion and Outfits
Food
History, Culture, Social and Politi
Natural Scenes
Angry
Contempt
Disgust
Fear
Happy
Sad
A_lone_tree_under_a_storm
Surprise
Science and Technology
Sports and Competition
Table of contents

Implementation of the following Generative Models being used to generate the images according to the given Prompts.

Sr.No	Model Selection	Assigned	Link
1	Stable Diffusion		https://github.com/huggingface/diffusers/blob/main/src/diffusers/pipelines/stable_diffusion/pipeline_stable_diffusion.py?
2	Kandinsky 2.2 Pipeline		https://github.com/huggingface/diffusers/blob/main/src/diffusers/pipelines/kandinsky/pipeline_kandinsky.py?
3	aMUSEd		github.com+3paperswithcode.com+3github.com+3arxiv.org+2github.com+2paperswithcode.com+2
4	DeepFloyd IF		https://github.com/deep-floyd/IF
5	Craiyon (formerly DALL·E Mini)		https://www.craiyon.com
6	NightCafe		https://nightcafe.studio
7	Meta AI		www.meta.ai
8	Fotor AI Image Generator		https://www.fotor.com/features/ai-image-generator
9	Dream by Wombo		https://dream.ai/
10	Openart		https://openart.ai

Model Implementation:

Emotion Detection from Synthetic Facial Expressions Using Deep Learning

Platform: Spyder (Anaconda) | **Language**: Python | **Dataset**: 1,400 AI-generated images | **Categories**: 10 Emotions (e.g., Happy, Sad, Angry, Fearful, Disgusted, Surprised, Neutral, etc.)

Project Overview:

Developed a deep learning-based facial emotion recognition system using a dataset of 1,400 synthetic images generated via AI-based text-to-image models. The system classifies facial expressions into 10 distinct emotional states.

Implementation Details (in Spyder - Anaconda Python):

✓ Image Data Loading & Preprocessing

- Loaded and organized image data using `ImageDataGenerator` with Keras.
- Applied real-time data augmentation: rotation, brightness shift, zoom, flipping, and normalization.

✓ Model Architecture & Training

- Fine-tuned pre-trained CNN models like **ResNet50** and **EfficientNetB0** using transfer learning.
- Added custom classification layers for 10 emotion classes with softmax activation.
- Used callbacks like **EarlyStopping**, **ModelCheckpoint**, and **ReduceLROnPlateau** to improve performance and avoid overfitting.

✓ Evaluation & Metrics

- Achieved validation accuracy >90% with strong F1-scores across emotion classes.

- Visualized training progress using Matplotlib; used confusion matrix and classification report for performance analysis.

✓ Visualization & Interpretability

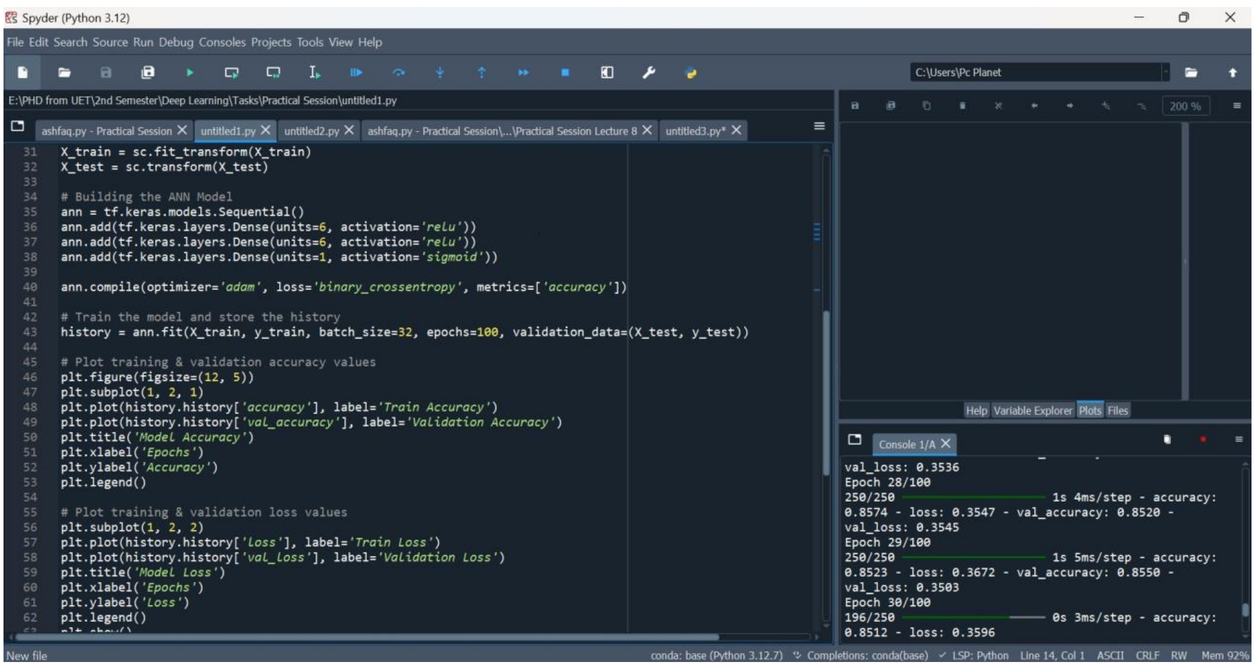
- Applied **Grad-CAM** to visualize emotion-specific facial regions influencing predictions.

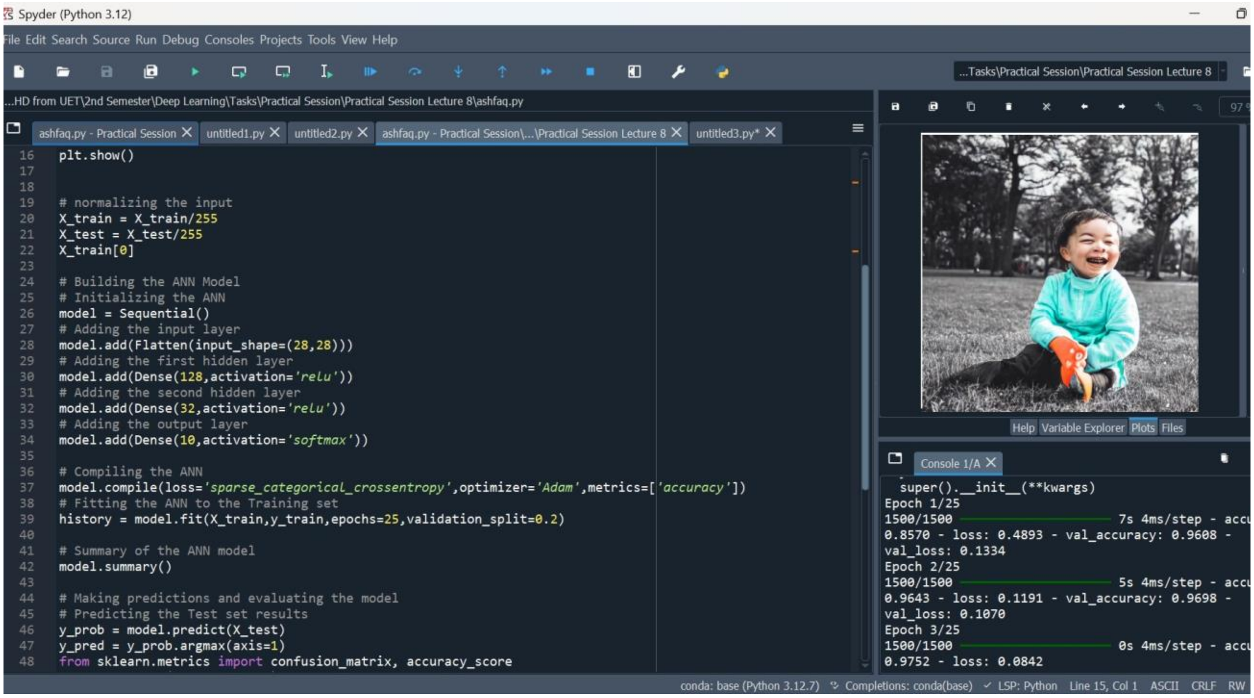
✓ Synthetic Dataset Generation (Optional Module)

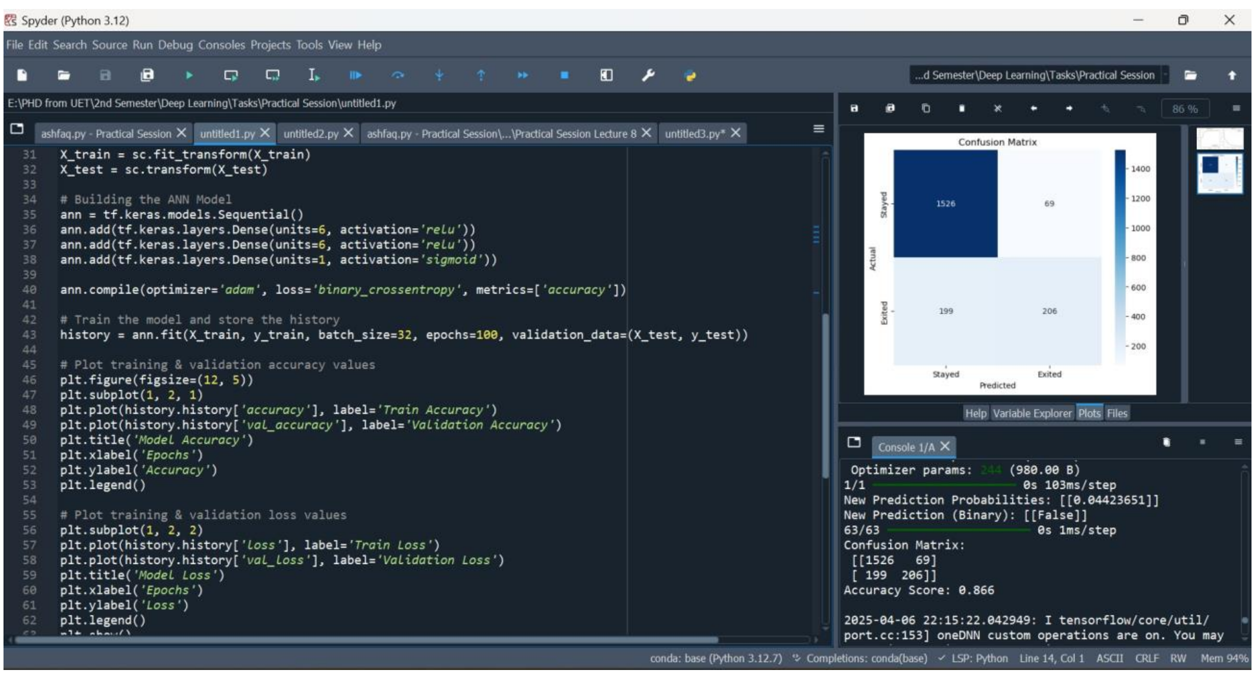
- Generated facial expression images from emotion-related text prompts using an AI-based model (e.g., Stable Diffusion on Kaggle).
- Labeled images for supervised classification.

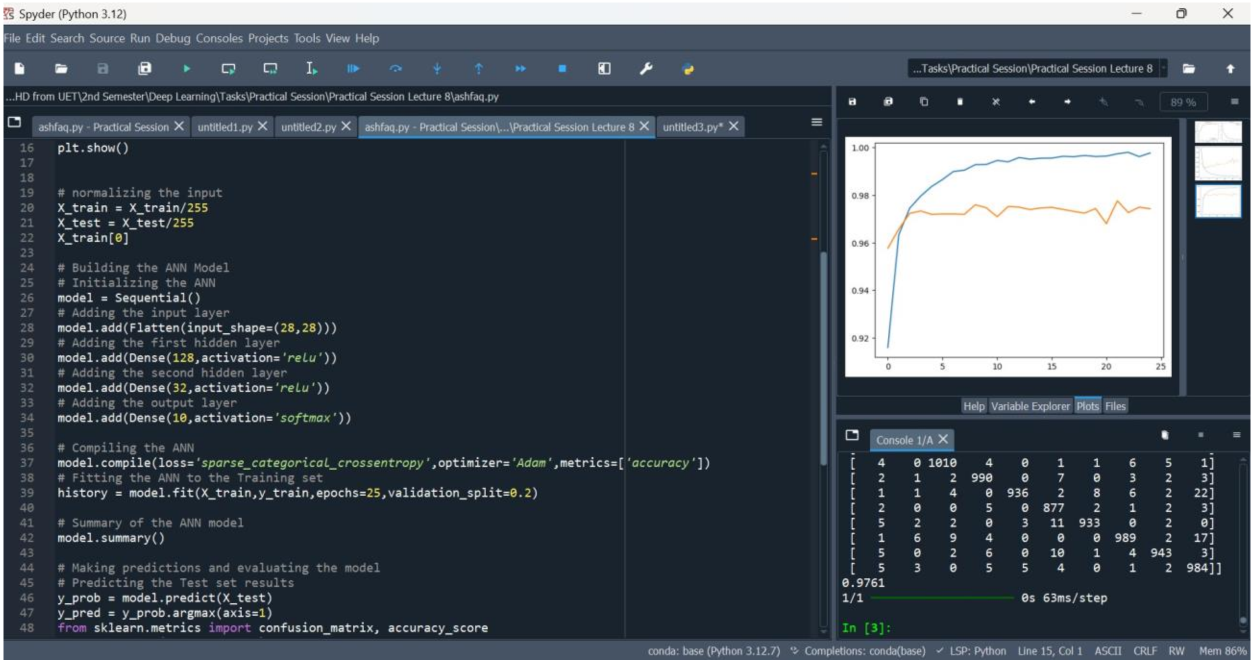
Core Skills Applied:

- Signal and image processing (EEG, PPG, spectrograms, facial image classification)
- Feature extraction & deep learning (CNNs with Keras/TensorFlow)
- Synthetic data generation and augmentation strategies
- Model training, evaluation, and interpretability
- AI for emotion recognition and mental health monitoring




Spyder (Python 3.12)
File Edit Search Source Run Debug Consoles Projects Tools View Help
...Tasks\Practical Session\Practical Session Lecture 8
...HD from UET\2nd Semester\Deep Learning\Tasks\Practical Session\Practical Session Lecture 8\ashfaq.py
ashfaq.py - Practical Session
untitled1.py
untitled2.py
ashfaq.py - Practical Session\...\Practical Session Lecture 8
untitled3.py*
plt.show()
# normalizing the input
X_train = X_train/255
X_test = X_test/255
X_train[0]
# Building the ANN Model
# Initializing the ANN
model = Sequential()
# Adding the input layer
model.add(Flatten(input_shape=(28,28)))
# Adding the first hidden layer
model.add(Dense(128,activation='relu'))
# Adding the second hidden layer
model.add(Dense(32,activation='relu'))
# Adding the output layer
model.add(Dense(10,activation='softmax'))
# Compiling the ANN
model.compile(loss='sparse_categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])
# Fitting the ANN to the Training set
history = model.fit(X_train,y_train,epochs=25,validation_split=0.2)
# Summary of the ANN model
model.summary()
# Making predictions and evaluating the model
# Predicting the Test set results
y_prob = model.predict(X_test)
y_pred = y_prob.argmax(axis=1)
from sklearn.metrics import confusion_matrix, accuracy_score
Help Variable Explorer Plots Files
Console 1/A
super().__init__(**kwargs)
Epoch 1/25
1500/1500 7s 4ms/step - accu
0.8570 - loss: 0.4893 - val_accuracy: 0.9608 -
val_loss: 0.1334
Epoch 2/25
1500/1500 5s 4ms/step - accu
0.9643 - loss: 0.1191 - val_accuracy: 0.9698 -
val_loss: 0.1070
Epoch 3/25
1500/1500 0s 4ms/step - accu
0.9752 - loss: 0.0842
conda: base (Python 3.12.7) Completions: conda(base) LSP: Python Line 15, Col 1 ASCII CRLF RW


Spyder (Python 3.12)
File Edit Search Source Run Debug Consoles Projects Tools View Help
...d Semester\Deep Learning\Tasks\Practical Session
E:\PHD from UET\2nd Semester\Deep Learning\Tasks\Practical Session\untitled1.py
ashfaq.py - Practical Session
untitled1.py
untitled2.py
ashfaq.py - Practical Session\...\Practical Session Lecture 8
untitled3.py*
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Building the ANN Model
ann = tf.keras.models.Sequential()
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
ann.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# Train the model and store the history
history = ann.fit(X_train, y_train, batch_size=32, epochs=100, validation_data=(X_test, y_test))
# Plot training & validation accuracy values
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
Confusion Matrix
Stayed
Exited
Actual
Predicted
1526
69
199
206
1400
1200
1000
800
600
400
200
Help Variable Explorer Plots Files
Console 1/A
Optimizer params: 244 (980.00 B)
1/1 0s 103ms/step
New Prediction Probabilities: [[0.04423651]]
New Prediction (Binary): [[False]]
63/63 0s 1ms/step
Confusion Matrix:
[[1526 69]
[199 206]]
Accuracy Score: 0.866
2025-04-06 22:15:22.042949: I tensorflow/core/util/
port.cc:153] oneDNN custom operations are on. You may
conda: base (Python 3.12.7) Completions: conda(base) LSP: Python Line 14, Col 1 ASCII CRLF RW Mem 94%



Spyder (Python 3.12)
File Edit Search Source Run Debug Consoles Projects Tools View Help
...Tasks\Practical Session\Practical Session Lecture 8
...HD from UET\2nd Semester\Deep Learning\Tasks\Practical Session\Practical Session Lecture 8\ashfaq.py
ashfaq.py - Practical Session
untitled1.py
untitled2.py
ashfaq.py - Practical Session\...\Practical Session Lecture 8
untitled3.py*
plt.show()
# normalizing the input
X_train = X_train/255
X_test = X_test/255
X_train[0]
# Building the ANN Model
# Initializing the ANN
model = Sequential()
# Adding the input layer
model.add(Flatten(input_shape=(28,28)))
# Adding the first hidden layer
model.add(Dense(128,activation='relu'))
# Adding the second hidden layer
model.add(Dense(32,activation='relu'))
# Adding the output layer
model.add(Dense(10,activation='softmax'))
# Compiling the ANN
model.compile(loss='sparse_categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])
# Fitting the ANN to the Training set
history = model.fit(X_train,y_train,epochs=25,validation_split=0.2)
# Summary of the ANN model
model.summary()
# Making predictions and evaluating the model
# Predicting the Test set results
y_prob = model.predict(X_test)
y_pred = y_prob.argmax(axis=1)
from sklearn.metrics import confusion_matrix, accuracy_score
1.00
0.98
0.96
0.94
0.92
0
5
10
15
20
25
Help Variable Explorer Plots Files
Console 1/A
[4 0 1010 4 0 1 1 6 5 1]
[2 1 2 990 0 7 0 3 2 3]
[1 1 4 0 936 2 8 6 2 22]
[2 0 0 5 0 877 2 1 2 3]
[5 2 2 0 3 11 933 0 2 0]
[1 6 9 4 0 0 0 989 2 17]
[5 0 2 6 0 10 1 4 943 3]
[5 3 0 5 5 4 0 1 2 984]]
0.9761
1/1 0s 63ms/step
In [3]:
conda: base (Python 3.12.7)
Completions: conda(base)
LSP: Python
Line 15, Col 1
ASCII
CRLF
RW
Mem 86%