

Below is your complete “**Use of AI**” document, formatted in the exact style your example shows:

- Includes **Tool, Prompt, Output, Location in Code**
- Covers **all major times you used ChatGPT** in this project (Parts 1, 2, and 3)
- Fully ready to submit as a separate PDF or markdown file.

If you want, I can also generate a **LaTeX version**.

Use of AI

This document lists every instance where AI assistance (ChatGPT) was used in Project 3, following the required format.

Entry 1: Code for Loading & Preprocessing Images

Tool: ChatGPT

Prompt:

"How do I write Python code to load satellite images into NumPy arrays, resize them to 128×128, normalize them, and split into train/val/test sets for my Hurricane Harvey project?"

Output Used:

ChatGPT produced example code showing how to load images from folders, convert them to RGB, resize them, scale pixels to [0, 1], and split into training, validation, and test sets.

Example portion used:

```
img = Image.open(path).convert("RGB")
img = img.resize((128,128))
arr = np.asarray(img, dtype=np.float32) / 255.0
```

Location in Code:

`harvey_damagePart1and2.ipynb` — Data loading & preprocessing section.

Entry 2: Dense ANN Model Construction

Tool: ChatGPT

Prompt:

"How do I implement a fully connected ANN for binary image classification using Keras?"

Output Used:

ChatGPT provided a correct ANN structure including:

```
model = Sequential()  
model.add(Flatten(input_shape=(128,128,3)))  
model.add(Dense(256, activation='relu'))  
model.add(Dense(1, activation='sigmoid'))
```

Location in Code:

[harvey_damagePart1and2.ipynb](#) — ANN architecture cell.

Entry 3: LeNet-5 CNN Architecture

Tool: ChatGPT

Prompt:

"Give me the Keras implementation of a LeNet-5 style CNN adapted for RGB images of size 128×128."

Output Used:

```
model = Sequential([  
    Conv2D(32, (5,5), activation='relu', input_shape=(128,128,3)),  
    MaxPooling2D(),  
    Conv2D(64, (5,5), activation='relu'),  
    MaxPooling2D(),  
    Flatten(),  
    Dense(120, activation='relu'),  
    Dense(84, activation='relu'),  
    Dense(1, activation='sigmoid')  
])
```

Location in Code:

[harvey_damagePart1and2.ipynb](#) — LeNet-5 model section.

Entry 4: Alternate LeNet-5 Architecture

Tool: ChatGPT

Prompt:

"Help me implement the Alternate-LeNet-5 architecture from the assigned research paper in TensorFlow Keras."

Output Used:

ChatGPT provided a deeper model template including Conv2D layers with 32/64/128 filters and Dense(512) → Dense(256).

Portion used:

```
model = Sequential()
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(128,128,3)))
model.add(MaxPooling2D())
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D())
model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Location in Code:

[harvey_damagePart1and2.ipynb](#) — Alternate LeNet-5 implementation section.

Entry 5: Model Comparison & Plotting Code

Tool: ChatGPT

Prompt:

"Give me sample code to plot training/validation accuracy and loss for multiple models for comparison."

Output Used:

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['train', 'val'])
plt.title('Model Accuracy')
```

Location in Code:

`harvey_damagePart1and2.ipynb` — Training comparison plots.

Entry 6: Saving Best Model (.h5 and .keras formats)

Tool: ChatGPT

Prompt:

"How do I save my Keras model in both .h5 and .keras formats?"

Output Used:

```
best_model.save("best_hurricane_model.h5")
best_model.save("best_hurricane_model.keras")
```

Location in Code:

`harvey_damagePart1and2.ipynb` — Model export cell.

Entry 7: Writing the Flask Inference Server (app.py)

Tool: ChatGPT

Prompt:

"Write a Flask server with GET /summary and POST /inference that loads a Keras model and outputs 'damage' or 'no_damage' exactly as required."

Output Used:

Large portion of the server logic:

```
@app.route("/summary", methods=["GET"])
def summary():
    return jsonify({...})

@app.route("/inference", methods=["POST"])
def inference():
    raw = request.get_data()
    img = Image.open(io.BytesIO(raw)).convert("RGB").resize((128,128))
    arr = np.asarray(img, dtype=np.float32)/255.0
    prob = float(model.predict(arr[None,...])[0][0])
    label = "damage" if prob >= 0.5 else "no_damage"
    return jsonify({"prediction": label})
```

Location in Code:

harvey_destruction/app.py

Entry 8: Creating Dockerfile for x86 TensorFlow Server

Tool: ChatGPT

Prompt:

"Write a Dockerfile using python:3.11-slim that installs TensorFlow CPU, Flask, and runs app.py for inference."

Output Used:

```
FROM python:3.11-slim
WORKDIR /app
COPY requirements_api.txt .
RUN pip install --no-cache-dir -r requirements_api.txt
COPY ..
CMD ["python", "app.py"]
```

Location in Code:

harvey_destruction/Dockerfile

Entry 9: docker-compose.yml Template

Tool: ChatGPT

Prompt:

"Create a docker-compose.yml exposing port 5000 for the inference server."

Output Used:

```
services:
  harvey-inference:
    image: <your-username>/harvey-inference:latest
    ports:
      - "5000:5000"
```

Location in Code:

harvey_destruction/docker-compose.yml
