West Visayas State University
(Formerly Iloilo Normal School)
COLLEGE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY
Luna St., La Paz, Iloilo City 5000
Iloilo, Philippines
* Trunkline: (063) (033) 320-0870 to 78 loc. 1403 * Telefax No.: (033) 320-0879
* Website: www.wvsu.edu.ph * Email Address: cict@wvsu.edu.ph

# A Deep Learning–Based Fake Review Detector Using Bidirectional GRU

**Submitted by:**

Alvarez, Jan Daniel

Baladjay, Aser Jr.

Barlas, Ramuel Carl Q.

**BSCS 3A-AI**

**Submitted to:**

John Cristopher Mateo

**December 2025**

## Problem

Online shopping platforms such as **Shopee, Lazada, and eBay** contain large numbers of **fake, bot-generated, or AI-generated reviews**.
These reviews artificially inflate product ratings and mislead buyers by:

- creating a false sense of product quality,
- manipulating consumer perception,
- overshadowing genuine customer experiences, and
- increasing the risk of purchasing low-quality or fraudulent products.

Because review authenticity influences purchase decisions, this problem can cost consumers **time, trust, and money**. Detecting and filtering out fake content is essential to restoring confidence in online shopping systems.

## Solution

The proposed solution is an **AI-powered Fake Review Detection System** that identifies whether a product review is:

- **Original / Human-written**, or
- **AI-generated / bot-generated**

The system uses **deep learning**, specifically a **Bidirectional Recurrent Neural Network (Bi-RNN)**, enhanced with **GRU layers**, to analyze linguistic patterns commonly found in AI-generated text.

The model is deployed using **Streamlit**, allowing users to:

- upload datasets
- train or load the AI model
- test individual reviews
- view explainability results through enhanced **LIME visualizations**

## Structure of Neural Network

The neural network architecture consists of:

1. **Embedding Layer (Word Embeddings)**
   Converts words into dense numerical vectors that capture semantic meaning.

2. **Bidirectional GRU Layer**
   Processes sequences in **both forward and backward directions**, helping the model detect contextual cues typical of AI-generated reviews.

3. **Dropout Layers**
   Prevent overfitting by randomly disabling neurons during training.

4. **Dense Layers**
   Additional layers to refine learned features.

5. **Output Layer (Sigmoid Activation)**
   Produces a probability between **0 and 1**, where:

   - 0 → Original (human)
   - 1 → AI-generated

## Tools

### Development Environment

- **Visual Studio Code (VSCode)**
  Used for writing, testing, and managing the project's Python-based source code.

### Programming Language

- **Python**

| Library | Purpose |
|---|---|
| **Keras** | Neural network layers, training, and model structure |

| | |
|---|---|
| **NumPy** | Numerical and tensor operations |
| **Pandas** | Dataset management and preprocessing |
| **Scikit-learn** | Splitting data, evaluation metrics, preprocessing |
| **Streamlit** | Web app/UI for model usage |
| **TensorFlow** | Backend engine powering Keras, training, optimization |

## Source of the Dataset

- https://github.com/congruiyS2023/GeneratedReviewsDetection/blob/main
- https://www.kaggle.com/datasets/ankushnarwade/ai-vs-human-product-reviews-for-sentiment-analysis
- https://www.kaggle.com/datasets/mexwell/fake-reviews-dataset

## Neural Network used

The core model is a **Bidirectional Recurrent Neural Network (Bi-RNN)** using **GRU (Gated Recurrent Unit)** cells.

### Why Bi-RNN?

- It reads text **left-to-right** and **right-to-left**, improving understanding of context.
- AI reviews often contain structured or repetitive wording patterns that BiRNNs can detect.
- GRU cells handle long sequences efficiently (better than vanilla RNNs).

### Why GRU?

- Faster than LSTM
- Less memory usage

- Performs strongly on text classification tasks

## Identify the optimizers

The model uses:

### Adam (Adaptive Moment Estimation) Optimizer

**Why Adam?**

- Combines the benefits of Momentum + RMSProp
- Adapts the learning rate during training
- Works exceptionally well on noisy textual data
- Converges quickly

The improved version, **AdamW** (Adam with weight decay), was also used for better generalization and reduced overfitting in the second and third Hyperparameter tuning.

First Hyper-parameter tuning:

**Training Configuration:**
**Embedding**
Vocabulary size (MAX_WORDS)
Embedding dimension (EMBED_DIM)
Fixed sequence length (MAX_LEN)

**Recurrent Layer**
Bidirectional GRU
Units: 256
Dropout (recurrent): 0.3
return_sequences = False

**Dense Layers**
Dense(256, ReLU)

Output: Dense(1, Sigmoid)

**Regularization**
Dropout: 0.5 (applied twice)

Training Results:

| Epoch | Training Accuracy | Training Loss |
|---|---|---|
| 1 | ~80.3% | 0.408 |
| 2 | ~89.6% | 0.238 |
| 3 | ~92.8% | 0.164 |
| 4 | ~93.6% | 0.142 |
| 5 | ~95.1% | 0.110 |
| 6 | ~95.6% | 0.093 |

Validation Results:

| Epoch | Validation Accuracy | Validation Loss |
|---|---|---|
| 1 | ~88.1% | 0.269 |
| 2 | ~88.1% | 0.255 |
| 3 | ~88.8% | 0.253 |
| 4 | ~89.0% | 0.288 |
| 5 | ~88.7% | 0.314 |
| 6 | ~89.9% | 0.328 |

The first hyper-parameter tuning established a baseline Bidirectional GRU-based RNN using fixed embeddings, 256 GRU units, dropout regularization, AdamW optimization, and early stopping. This configuration achieved stable convergence and consistent validation performance but showed early performance saturation, indicating limited sensitivity to harder or minority-class samples.

**Second Hyper-parameter Tuning**

**Embedding**
Vocabulary size (MAX_WORDS)
Embedding dimension (EMBED_DIM)

Fixed sequence length (MAX_LEN = 350)

**Recurrent Layer**
Bidirectional GRU
Units: 512
Dropout (recurrent): 0.3
return_sequences = False

**Dense Layers**
Dense(256, ReLU)
Output: Dense(1, Sigmoid)
Lower sigmoid decision threshold applied during inference

**Regularization**
Dropout: 0.5 (applied twice)
Class Weights Enabled
- Class 0 (Human): 1.0
- Class 1 (AI-generated): 1.5

**Optimizer and Loss Function**
Optimizer: AdamW
- Learning rate: 3e-4
- Weight decay: 1e-4

Loss Function: Focal Loss
- Gamma = 2
- Alpha = 0.25

**Training Strategy:**
Batch size: 128
Epochs: 25 (maximum)
Validation split: 0.2
Early Stopping
Monitor: val_loss
Patience: 3
Restore best weights: True
Training Results:

| Epoch | Training Accuracy | Training Loss |
|---|---|---|
| 1 | 79.91% | 0.4027 |
| 2 | 94.33% | 0.2197 |

| | | |
|---|---|---|
| 3 | 92.49% | 0.1699 |
| 4 | 93.92% | 0.1337 |
| 5 | 94.76% | 0.1141 |

Validation Results:

| Epoch | Validation Accuracy | Validation Loss |
|---|---|---|
| 1 | 86.57% | 0.3098 |
| 2 | 88.32% | 0.2454 |
| 3 | 88.21% | 0.2579 |
| 4 | 88.43% | 0.2667 |
| 5 | 87.55% | 0.3173 |

The second tuning increased model capacity by expanding the GRU to 512 units and extending the input sequence length, while introducing focal loss and class weighting. Although overall validation accuracy did not significantly improve, the model demonstrated better robustness and improved sensitivity to AI-generated reviews.

**Final Hyperparameter Training**

**Training Configuration:**

**Model Architecture**
Embedding layer with fixed vocabulary size (MAX_WORDS)
Embedding dimension (EMBED_DIM)
Fixed sequence length (MAX_LEN)
Bidirectional GRU-based RNN (architecture unchanged from previous tuning)
Dense hidden layer: 128 units, ReLU activation
Output layer: 1 unit, Sigmoid activation

**Recurrent Layer**
Bidirectional GRU
Units: 128
Recurrent dropout: 0.3
return_sequences = False

**Regularization**

Dropout: 0.5 (applied after GRU and Dense layer)
L1 and L2 regularization applied to dense layer weights (final refinement)
Optimizer and Loss Function
Optimizer: AdamW
Learning rate: 3e-4
Weight decay: 1e-4
Loss function: Binary Cross-Entropy

**Training Strategy:**
Early Stopping:
Monitor: val_loss
Patience: 3
Restore best weights: True
Class weights:
Human reviews (Class 0): 1.0
AI-generated reviews (Class 1): 1.5

**Training Parameters:**
Batch size: 128
Maximum epochs: 25
Validation split: 0.2
Verbose training output enabled

Training Results:

| Epoch | Training Accuracy | Training Loss |
|---|---|---|
| 1 | ~61.6% | 0.145 |
| 2 | ~86.4% | 0.057 |
| 3 | ~89.2% | 0.037 |
| 4 | ~89.9% | 0.031 |
| 5 | ~90.8% | 0.028 |
| 6 | ~91.9% | 0.026 |
| 7 | ~92.7% | 0.023 |
| 8 | ~93.1% | 0.023 |
| 9 | ~93.7% | 0.021 |
| 10 | ~93.7% | 0.020 |

| | | |
|---|---|---|
| 11 | ~94.0% | 0.020 |

Validation Results:

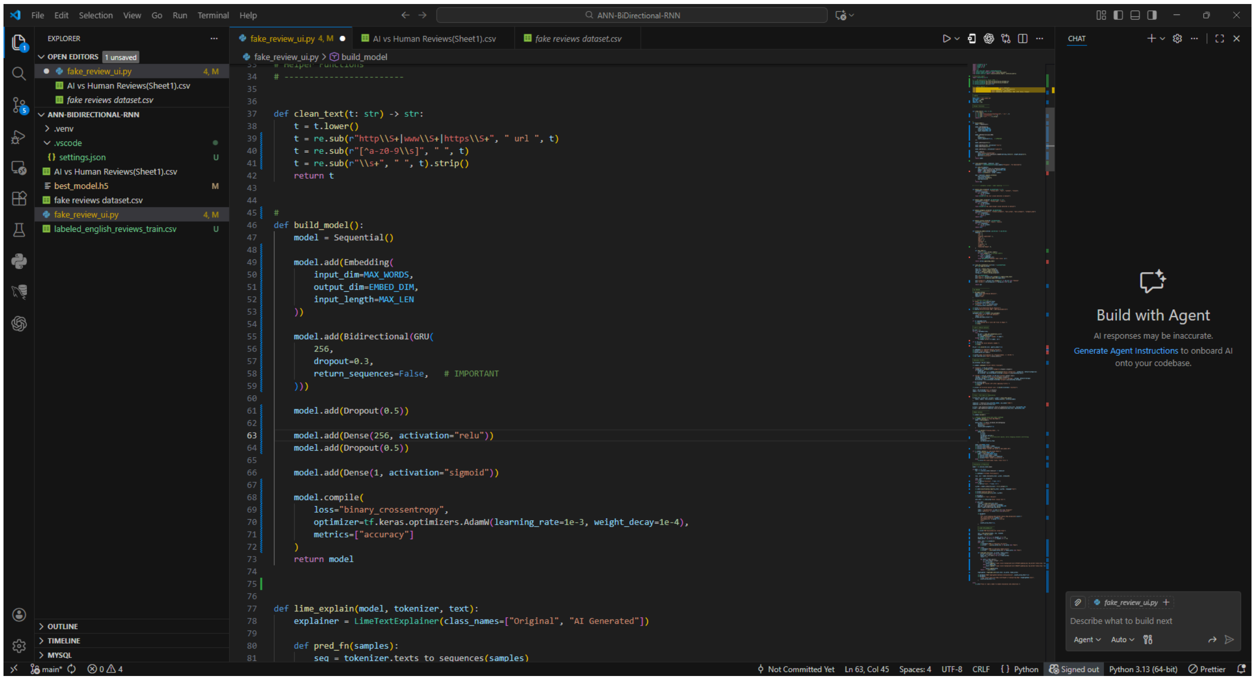
| Epoch | Training Accuracy | Training Loss |
|---|---|---|
| 1 | ~83.8% | 0.073 |
| 2 | ~87.9% | 0.037 |
| 3 | ~88.6% | 0.030 |
| 4 | ~86.8% | 0.030 |
| 5 | ~89.1% | 0.026 |
| 6 | ~89.1% | 0.025 |
| 7 | ~87.3% | 0.027 |
| 8 | ~89.1% | 0.025 |
| 9 | ~89.1% | 0.025 |
| 10 | ~89.2% | 0.025 |
| 11 | ~88.7% | 0.026 |

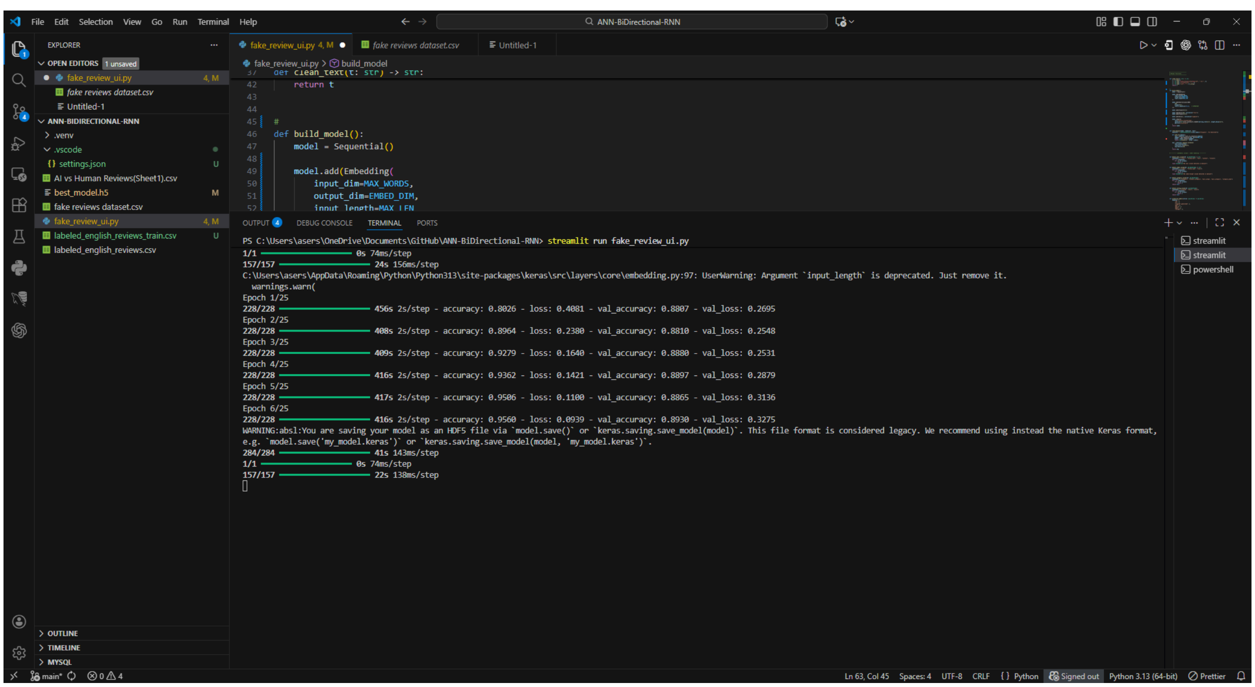
The final tuning focused on refining training stability without altering the model architecture by emphasizing early stopping, class weighting, and additional regularization. This approach reduced overfitting and produced a more reliable and generalizable model suitable for final project deployment.

West Visayas State University
(Formerly Iloilo Normal School)
COLLEGE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY
Luna St., La Paz, Iloilo City 5000
Iloilo, Philippines
* Trunkline: (063) (033) 320-0870 to 78 loc. 1403 * Telefax No.: (033) 320-0879
* Website: www.wvsu.edu.ph * Email Address: cict@wvsu.edu.ph

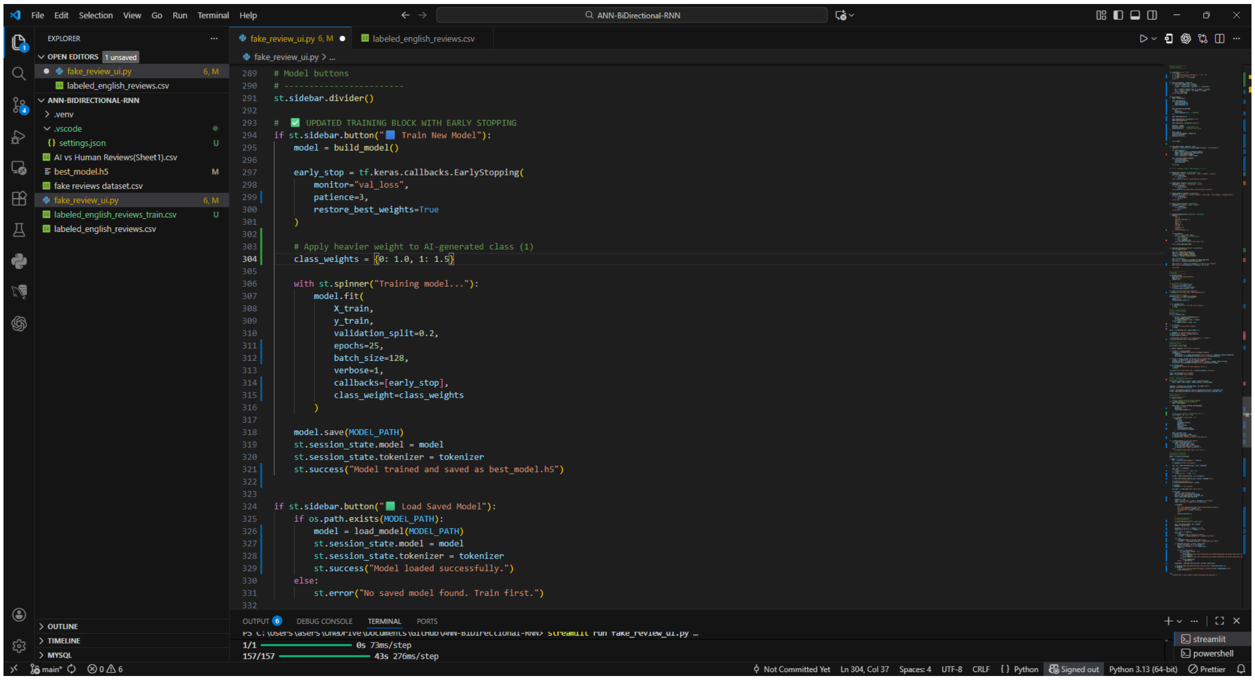
Screenshots:

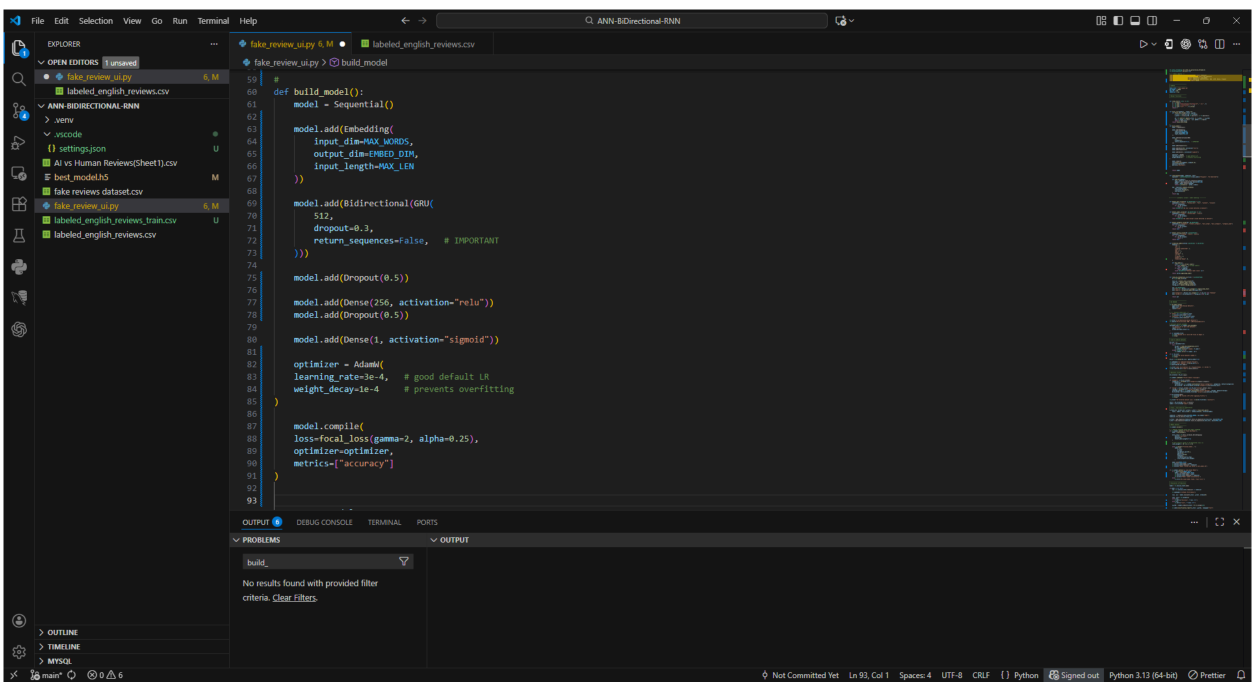
Hyperparameter 1

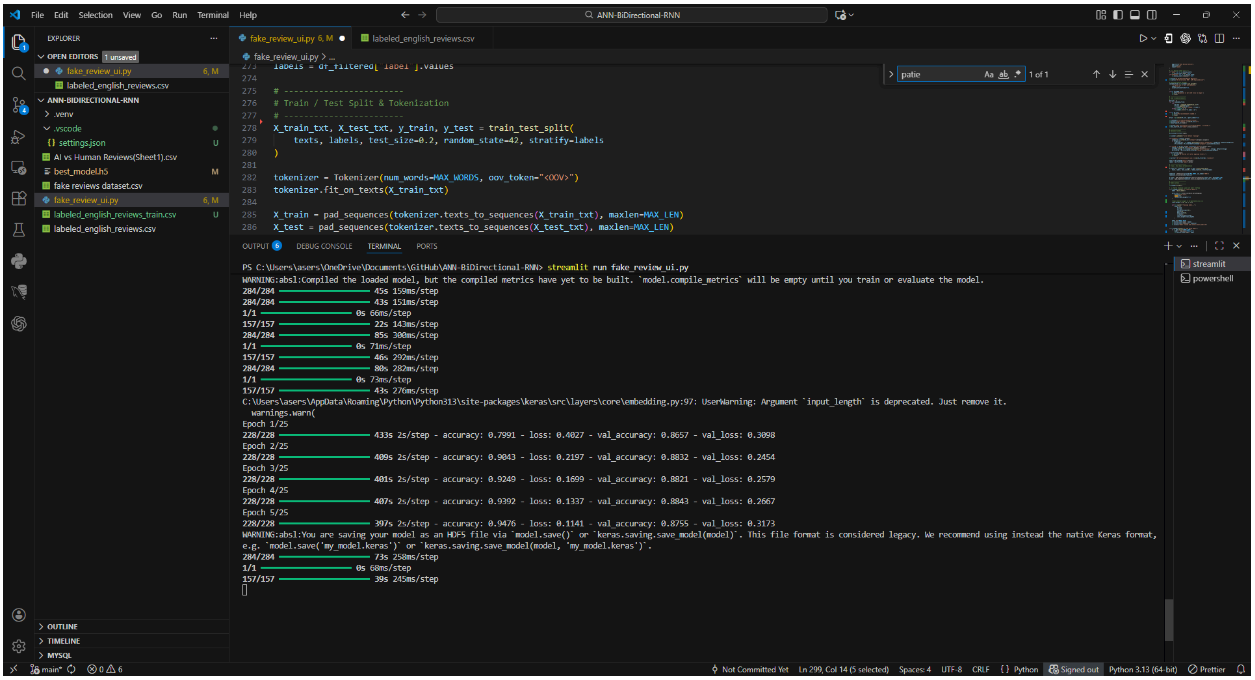


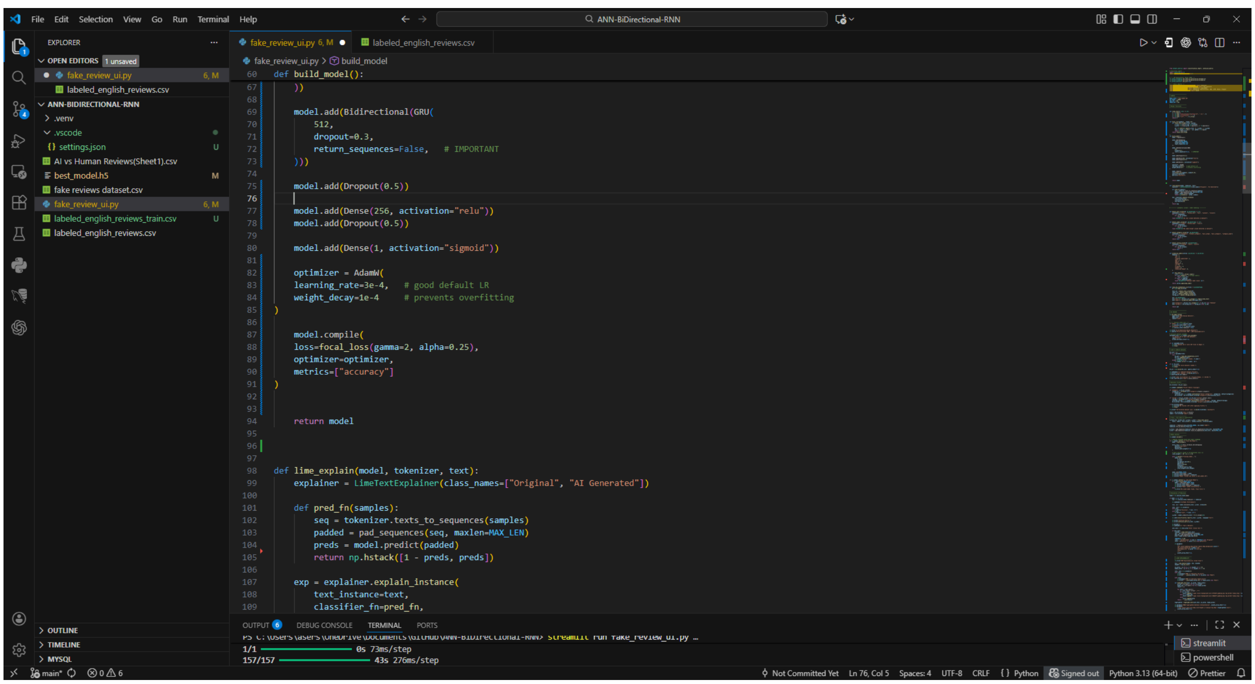



Hyperparameter 2:




ANN-BiDirectional-RNN
fake_review_ui.py 6, M
labeled_english_reviews.csv
labels = df_filtered["label"].values
# -----------------------
# Train / Test Split & Tokenization
# -----------------------
X_train_txt, X_test_txt, y_train, y_test = train_test_split(
texts, labels, test_size=0.2, random_state=42, stratify=labels
)
tokenizer = Tokenizer(num_words=MAX_WORDS, oov_token="<OOV>")
tokenizer.fit_on_texts(X_train_txt)
X_train = pad_sequences(tokenizer.texts_to_sequences(X_train_txt), maxlen=MAX_LEN)
X_test = pad_sequences(tokenizer.texts_to_sequences(X_test_txt), maxlen=MAX_LEN)
OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\asers\OneDrive\Documents\GitHub\ANN-BiDirectional-RNN> streamlit run fake_review_ui.py
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
284/284 45s 159ms/step
284/284 43s 151ms/step
1/1 0s 66ms/step
157/157 22s 143ms/step
284/284 85s 300ms/step
1/1 0s 71ms/step
157/157 46s 292ms/step
284/284 80s 282ms/step
1/1 0s 73ms/step
157/157 43s 276ms/step
C:\Users\asers\AppData\Roaming\Python\Python313\site-packages\keras\src\layers\core\embedding.py:97: UserWarning: Argument `input_length` is deprecated. Just remove it.
warnings.warn(
Epoch 1/25
228/228 433s 2s/step - accuracy: 0.7991 - loss: 0.4027 - val_accuracy: 0.8657 - val_loss: 0.3098
Epoch 2/25
228/228 409s 2s/step - accuracy: 0.9043 - loss: 0.2197 - val_accuracy: 0.8832 - val_loss: 0.2454
Epoch 3/25
228/228 401s 2s/step - accuracy: 0.9249 - loss: 0.1699 - val_accuracy: 0.8821 - val_loss: 0.2579
Epoch 4/25
228/228 407s 2s/step - accuracy: 0.9392 - loss: 0.1337 - val_accuracy: 0.8843 - val_loss: 0.2667
Epoch 5/25
228/228 397s 2s/step - accuracy: 0.9476 - loss: 0.1141 - val_accuracy: 0.8755 - val_loss: 0.3173
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
284/284 73s 258ms/step
1/1 0s 68ms/step
157/157 39s 245ms/step
Not Committed Yet Ln 299, Col 14 (5 selected) Spaces: 4 UTF-8 CRLF Python Signed out Python 3.13 (64-bit) Prettier


ANN-BiDirectional-RNN
fake_review_ui.py 6, M
labeled_english_reviews.csv
fake_review_ui.py > build_model
def build_model():
))
model.add(Bidirectional(GRU(
512,
dropout=0.3,
return_sequences=False, # IMPORTANT
)))
model.add(Dropout(0.5))
model.add(Dense(256, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(1, activation="sigmoid"))
optimizer = AdamW(
learning_rate=3e-4, # good default LR
weight_decay=1e-4 # prevents overfitting
)
model.compile(
loss=focal_loss(gamma=2, alpha=0.25),
optimizer=optimizer,
metrics=["accuracy"]
)
return model
def lime_explain(model, tokenizer, text):
explainer = LimeTextExplainer(class_names=["Original", "AI Generated"])
def pred_fn(samples):
seq = tokenizer.texts_to_sequences(samples)
padded = pad_sequences(seq, maxlen=MAX_LEN)
preds = model.predict(padded)
return np.hstack([1 - preds, preds])
exp = explainer.explain_instance(
text_instance=text,
classifier_fn=pred_fn,
OUTPUT DEBUG CONSOLE TERMINAL PORTS
1/1 0s 73ms/step
157/157 43s 276ms/step
Not Committed Yet Ln 76, Col 5 Spaces: 4 UTF-8 CRLF Python Signed out Python 3.13 (64-bit) Prettier

# West Visayas State University

(Formerly Iloilo Normal School)

**COLLEGE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY**

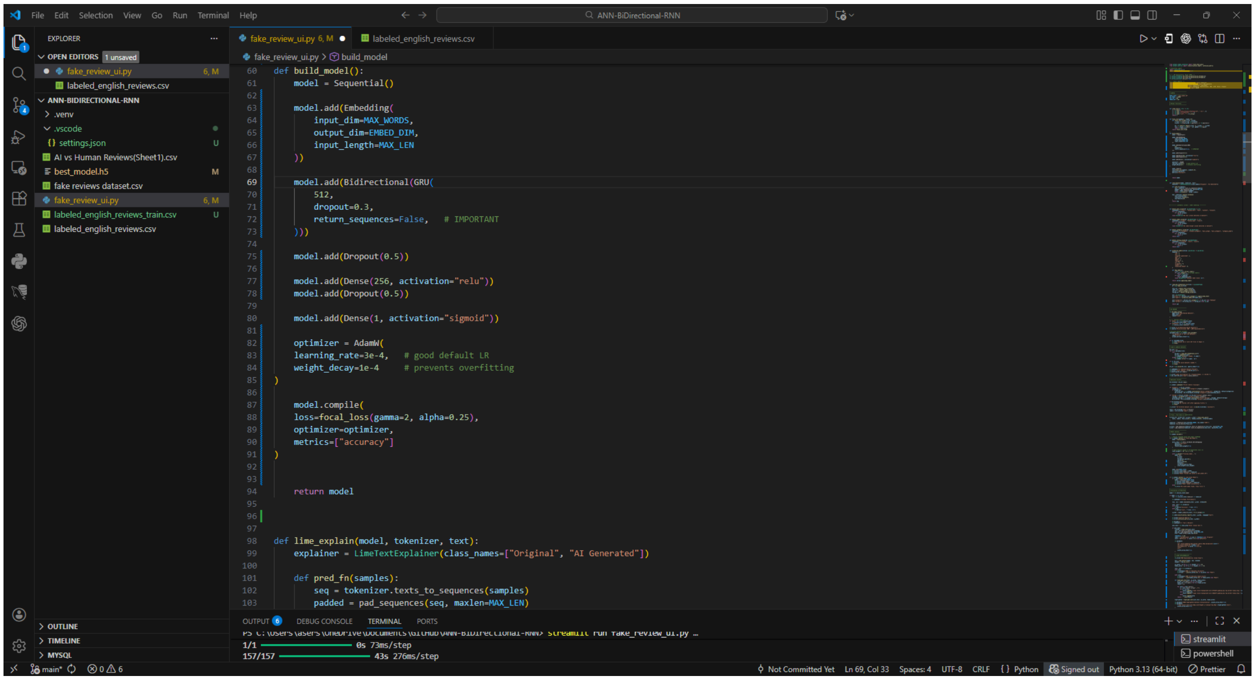
Luna St., La Paz, Iloilo City 5000

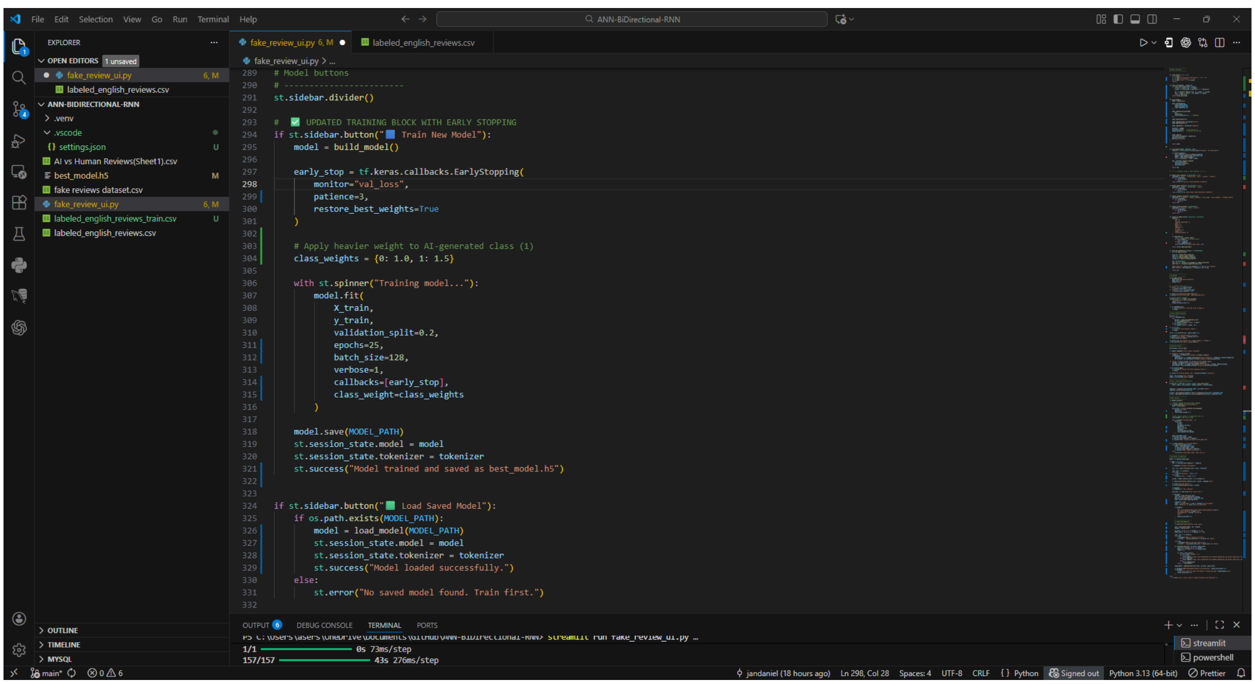
Iloilo, Philippines

* Trunkline: (063) (033) 320-0870 to 78 loc. 1403 * Telefax No.: (033) 320-0879

* Website: www.wvsu.edu.ph * Email Address: cict@wvsu.edu.ph


BAGONG PILIPINAS



Hyperparameter 3:

```python
def build_model():
    model = Sequential()
    model.add(Embedding(input_dim=MAX_WORDS, output_dim=EMBED_DIM, input_length=MAX_LEN))

    model.add(Bidirectional(GRU(
        128,
        dropout=0.3,
        recurrent_dropout=0.3,
        kernel_regularizer=regularizers.L1L2(l1=1e-5, l2=1e-4),
        recurrent_regularizer=regularizers.L1L2(l1=1e-5, l2=1e-4),
        bias_regularizer=regularizers.L2(1e-4)
    )))

    model.add(Dropout(0.6))

    model.add(Dense(
        64, activation="relu",
        kernel_regularizer=regularizers.L2(1e-4)
    ))
    model.add(Dropout(0.6))

    model.add(Dense(
        1, activation="sigmoid",
        kernel_regularizer=regularizers.L2(1e-4)
    ))

    optimizer = AdamW(
        learning_rate=3e-4,
        weight_decay=1e-4
    )

    model.compile(
        loss=focal_loss(gamma=2, alpha=0.25),
        optimizer=optimizer,
        metrics=["accuracy"]
    )

    return model
```

```python
# ✅ UPDATED TRAINING BLOCK WITH EARLY STOPPING
if st.sidebar.button("🟦 Train New Model"):
    model = build_model()

    early_stop = tf.keras.callbacks.EarlyStopping(
        monitor="val_loss",
        patience=3,
        restore_best_weights=True
    )

    # Apply heavier weight to AI-generated class (1)
    class_weights = {0: 1.0, 1: 1.5}

    with st.spinner("Training model..."):
        model.fit(
            X_train,
            y_train,
            validation_split=0.2,
            epochs=25,
            batch_size=128,
            verbose=1,
            callbacks=[early_stop],
            class_weight=class_weights
        )

    model.save(MODEL_PATH)
    st.session_state.model = model
    st.session_state.tokenizer = tokenizer
    st.success("Model trained and saved as best_model.h5")
```

```
C:\Users\USER\Downloads\ANN-BiDirectional-RNN-main\ANN-BiDirectional-RNN-main\flow\Lib\site-packages\keras\src\layers\core\embedding.py:97: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
2025-12-13 17:43:10.550520: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 1/25
203/203 ━━━━━━━━━━━━━━━━━━━━ 604s 3s/step - accuracy: 0.6159 - loss: 0.1454 - val_accuracy: 0.8375 - val_loss: 0.0727
Epoch 2/25
203/203 ━━━━━━━━━━━━━━━━━━━━ 662s 3s/step - accuracy: 0.8644 - loss: 0.0571 - val_accuracy: 0.8793 - val_loss: 0.0369
Epoch 3/25
203/203 ━━━━━━━━━━━━━━━━━━━━ 936s 5s/step - accuracy: 0.8920 - loss: 0.0368 - val_accuracy: 0.8858 - val_loss: 0.0295
Epoch 4/25
203/203 ━━━━━━━━━━━━━━━━━━━━ 869s 4s/step - accuracy: 0.8989 - loss: 0.0314 - val_accuracy: 0.8684 - val_loss: 0.0299
Epoch 5/25
203/203 ━━━━━━━━━━━━━━━━━━━━ 901s 4s/step - accuracy: 0.9075 - loss: 0.0278 - val_accuracy: 0.8913 - val_loss: 0.0255
Epoch 6/25
203/203 ━━━━━━━━━━━━━━━━━━━━ 907s 4s/step - accuracy: 0.9185 - loss: 0.0255 - val_accuracy: 0.8912 - val_loss: 0.0251
Epoch 7/25
203/203 ━━━━━━━━━━━━━━━━━━━━ 1015s 5s/step - accuracy: 0.9266 - loss: 0.0234 - val_accuracy: 0.8732 - val_loss: 0.0270
Epoch 8/25
203/203 ━━━━━━━━━━━━━━━━━━━━ 972s 5s/step - accuracy: 0.9306 - loss: 0.0226 - val_accuracy: 0.8913 - val_loss: 0.0246
Epoch 9/25
203/203 ━━━━━━━━━━━━━━━━━━━━ 961s 4s/step - accuracy: 0.9369 - loss: 0.0208 - val_accuracy: 0.8913 - val_loss: 0.0253
Epoch 10/25
203/203 ━━━━━━━━━━━━━━━━━━━━ 929s 5s/step - accuracy: 0.9368 - loss: 0.0203 - val_accuracy: 0.8923 - val_loss: 0.0250
Epoch 11/25
203/203 ━━━━━━━━━━━━━━━━━━━━ 954s 5s/step - accuracy: 0.9403 - loss: 0.0197 - val_accuracy: 0.8870 - val_loss: 0.0255
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
253/253 ━━━━━━━━━━━━━━━━━━━━ 32s 123ms/step
```