

## Model Optimization and Tuning Phase Template

|               |   |
|---------------|---|
| Date          | 26 November 2024  |
| Team ID       | SWTID1727420425   |
| Project Title | Analysis of Amazon Cell Phone Reviews Using NLP Technique |
| Maximum Marks | 10 Marks  |

### Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

### Hyperparameter Tuning Documentation (8 Marks):

| Model     | Tuned Hyperparameters   |
|-----------|---|
| ANN Model | <pre> # Hyperparameter tuning for ANN tuner_ann = kt.RandomSearch(     build_ann_model,     objective='val_accuracy',     max_trials=5,     executions_per_trial=1,     directory='ann_tuning',     project_name='ann_model' )  /usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument 'input_length' is deprecated. Just remove it. warnings.warn(  [ ] # Splitting training data into train and validation sets for ANN tuning X_train_split, X_val, y_train_split, y_val = train_test_split(X_train_pad, y_train, test_size=0.2, random_state=42)  [ ] # Run the tuner tuner_ann.search(X_train_split, y_train_split, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)  Trial 5 Complete [00h 02m 49s] val_accuracy: 1.0  Best val_accuracy So Far: 1.0 Total elapsed time: 00h 14m 01s </pre> |
| CNN Model | <pre> # Tuning and training CNN model tuner_cnn = kt.RandomSearch(     build_cnn_model,     objective='val_accuracy',     max_trials=5,     executions_per_trial=1,     directory='cnn_tuning',     project_name='cnn_model' )  /usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument 'input_length' is deprecated. Just remove it. warnings.warn( </pre>  |

|              |  |
|--------------|--|
|              | <pre> tuner_cnn.search(X_train_split, y_train_split, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict) cnn_best_model = tuner_cnn.get_best_models(num_models=1)[0] cnn_best_model.fit(X_train_pad, y_train, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)  Trial 5 Complete [00h 05m 41s] val_accuracy: 1.0  Best val accuracy So Far: 1.0 Total elapsed time: 00h 19m 08s Epoch 1/5 /usr/local/lib/python3.10/dist-packages/keras/src/saving/saving_lib.py:713: UserWarning: Skipping variable loading for optimizer 'adam' saveable.load_own_variables(weights_store.get(inner_path)) 677/677 — 72s 104ms/step - accuracy: 1.0000 - loss: 3.2439e-06 - val_accuracy: 1.0000 - val_loss: 1.5969e-09 Epoch 2/5 677/677 — 71s 106ms/step - accuracy: 1.0000 - loss: 2.5446e-08 - val_accuracy: 1.0000 - val_loss: 4.8470e-10 Epoch 3/5 677/677 — 70s 103ms/step - accuracy: 1.0000 - loss: 9.0678e-09 - val_accuracy: 1.0000 - val_loss: 2.3031e-10 Epoch 4/5 677/677 — 82s 102ms/step - accuracy: 1.0000 - loss: 1.8188e-08 - val_accuracy: 1.0000 - val_loss: 6.8441e-11 Epoch 5/5 677/677 — 82s 103ms/step - accuracy: 1.0000 - loss: 2.7175e-09 - val_accuracy: 1.0000 - val_loss: 5.5722e-11 &lt;keras.src.callbacks.history.History at 0x7ed86b96f7c0&gt; </pre>  |
| LSTM Model   | <pre> [ ] # Hyperparameter tuning with Keras Tuner tuner = kt.RandomSearch(     build_lstm_model,     objective='val_accuracy',     max_trials=5,     executions_per_trial=1,     directory='lstm_tuning',     project_name='lstm_model' )  /usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument 'input_length' is deprecated. Just remove it. warnings.warn(  [ ] # Split training data into train and validation sets X_train_split, X_val, y_train_split, y_val = train_test_split(X_train_pad, y_train, test_size=0.2, random_state=42)  [ ] # Run the tuner tuner.search(X_train_split, y_train_split, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)  Trial 5 Complete [00h 20m 26s] val_accuracy: 1.0  Best val accuracy So Far: 1.0 Total elapsed time: 00h 43m 24s  [ ] # Get the best model best_hps = tuner.get_best_hyperparameters(num_trials=1)[0] lstm_best_model = tuner.get_best_models(num_models=1)[0]  /usr/local/lib/python3.10/dist-packages/keras/src/saving/saving_lib.py:713: UserWarning: Skipping variable loading for optimizer 'adam', because it has 2 va saveable.load_own_variables(weights_store.get(inner_path))  [ ] # Train the best model lstm_best_model.fit(X_train_pad, y_train, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict) </pre> |
| BiLSTM Model | <pre> [ ] # Tuning and training BiLSTM model tuner_bilstm = kt.RandomSearch(     build_bilstm_model,     objective='val_accuracy',     max_trials=5,     executions_per_trial=1,     directory='bilstm_tuning',     project_name='bilstm_model' )  /usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument 'input_length' is deprecated. Just remove it. warnings.warn( </pre>  |

```

tuner.bilstm.search(X_train_split, y_train_split, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)
bilstm_best_model = tuner.bilstm.get_best_models(num_models=1)[0]
bilstm_best_model.fit(X_train_pad, y_train, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)

```

Trial 5 Complete [00h 20m 43s]  
val\_accuracy: 1.0

Best val\_accuracy So Far: 1.0  
Total elapsed time: 01h 08m 49s  
Epoch 1/5  
/usr/local/lib/python3.10/dist-packages/keras/src/saving/saving\_lib.py:713: UserWarning: Skipping variable loading for optimizer 'adam', because it has 2 variables whereas the  
saveable.load\_own\_variables(weights\_store.get(inner\_path))  
Epoch 2/5  
677/677 — 86s 122ms/step - accuracy: 1.0000 - loss: 8.2559e-07 - val\_accuracy: 1.0000 - val\_loss: 5.5065e-09  
Epoch 3/5  
677/677 — 73s 108ms/step - accuracy: 1.0000 - loss: 1.0173e-08 - val\_accuracy: 1.0000 - val\_loss: 2.2621e-09  
Epoch 4/5  
677/677 — 72s 107ms/step - accuracy: 1.0000 - loss: 4.8849e-09 - val\_accuracy: 1.0000 - val\_loss: 1.3754e-09  
Epoch 5/5  
677/677 — 80s 104ms/step - accuracy: 1.0000 - loss: 3.2930e-09 - val\_accuracy: 1.0000 - val\_loss: 1.0158e-09  
677/677 — 86s 110ms/step - accuracy: 1.0000 - loss: 2.5530e-09 - val\_accuracy: 1.0000 - val\_loss: 8.0055e-10  
<keras.src.callbacks.history.History at 0x7eds7819e0b0>

```

bilstm_best_model.summary()

```

Model: "sequential"

| Layer (type)                  | Output Shape     | Param #   |
|-------------------------------|------------------|-----------|
| embedding (Embedding)         | (None, 100, 128) | 2,560,000 |
| bidirectional (Bidirectional) | (None, 64)       | 41,216    |
| dropout (Dropout)             | (None, 64)       | 0         |
| dense (Dense)                 | (None, 128)      | 8,320     |
| dense_1 (Dense)               | (None, 1)        | 129       |

Total params: 7,828,997 (29.87 MB)  
Trainable params: 2,609,665 (9.96 MB)  
Non-trainable params: 0 (0.00 B)  
Optimizer params: 5,219,332 (19.91 MB)

```

[ ] # Evaluate BiLSTM Model
test_loss, test_accuracy = bilstm_best_model.evaluate(X_test_pad, y_test)
print(f"BiLSTM Model Test Accuracy: {test_accuracy:.4f}")

```

170/170 — 3s 20ms/step - accuracy: 1.0000 - loss: 7.9719e-10  
BiLSTM Model Test Accuracy: 1.0000

### Final Model Selection Justification (2 Marks):

| Final Model | Reasoning   |
|-------------|---|
| ANN Model   | The ANN Model is more Accuracy for another three Model. |

```
# Define the ANN model
def build_ann_model(hp):
    model = Sequential()
    model.add(Embedding(input_dim=20000, output_dim=hp.Int('embedding_dim', 64, 256, step=64), input_length=max_sequence_length))
    model.add(Flatten())
    model.add(Dense(units=hp.Int('dense_units1', 32, 256, step=32), activation='relu'))
    model.add(Dropout(hp.Float('dropout_rate1', 0.2, 0.5, step=0.1)))
    model.add(Dense(units=hp.Int('dense_units2', 32, 128, step=32), activation='relu'))
    model.add(Dropout(hp.Float('dropout_rate2', 0.2, 0.5, step=0.1)))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer=Adam(learning_rate=hp.Float('learning_rate', 1e-4, 1e-2, sampling='log')),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

    return model

# Hyperparameter tuning for ANN
tuner_ann = kt.RandomSearch(
    build_ann_model,
    objective='val_accuracy',
    max_trials=5,
    executions_per_trial=1,
    directory='ann_tuning',
    project_name='ann_model'
)
```

```
[ ] # Splitting training data into train and validation sets for ANN tuning
X_train_split, X_val, y_train_split, y_val = train_test_split(X_train_pad, y_train, test_size=0.2, random_state=42)

[ ] # Run the tuner
tuner_ann.search(X_train_split, y_train_split, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)

Trial 5 Complete [00h 02m 49s]
val_accuracy: 1.0

Best val accuracy so far: 1.0
Total elapsed time: 00h 14m 01s

# Get the best model
ann_best_model = tuner_ann.get_best_models(num_models=1)[0]

/usr/local/lib/python3.10/dist-packages/keras/src/saving/saving_lib.py:713: UserWarning: Skipping variable loading for optimizer 'adam', because it has 2 variables whereas the saved saveable.load_own_variables(weights_store.get(inner_path))

[ ] # Train the best ANN model
ann_best_model.fit(X_train_pad, y_train, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)
```

```
[ ] # Evaluate the ANN model
test_loss, test_accuracy = ann_best_model.evaluate(X_test_pad, y_test)
print(f"ANN Model Test Accuracy: {test_accuracy:.4f}")

170/170 ————— 0s 2ms/step - accuracy: 1.0000 - loss: 3.7855e-24
ANN Model Test Accuracy: 1.0000
```

```
[ ] # Generate predictions and classification report
y_pred_probs_ann = ann_best_model.predict(X_test_pad)
y_pred_ann = (y_pred_probs_ann > 0.5).astype(int)
print("\nANN Classification Report:")
print(classification_report(y_test, y_pred_ann))
```

```
170/170 ————— 1s 3ms/step

ANN Classification Report:
              precision    recall  f1-score   support

         1              1.00      1.00      1.00     5416

   accuracy              1.00              1.00      1.00     5416
  macro avg              1.00      1.00      1.00     5416
 weighted avg              1.00      1.00      1.00     5416
```