imdb-bt-ex12-2

November 25, 2024

Name: Võ Hồng Quân Student ID: 22134012

[1]:
```python
import time
import plotly.graph_objects as go
from keras.datasets import imdb
from keras.preprocessing.sequence import pad_sequences
from keras.optimizers import Adam
from keras import Input
from keras.models import Sequential
from keras.layers import Embedding, SimpleRNN, LSTM, GRU, Dense, Bidirectional
```

[2]:
```python
# Chỉ sử dụng 10.000 từ phổ biến nhất
vocab_size = 10000

# Tải dữ liệu IMDB
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=vocab_size)

# Khởi tạo max_length
max_length = 500

# Embedding
embedding_dim = 100

# Padding các câu về cùng độ dài
x_train = pad_sequences(x_train, maxlen=max_length)
x_test = pad_sequences(x_test, maxlen=max_length)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/imdb.npz
17464789/17464789          0s
0us/step
```

[3]:
```python
def plot_history(history):
    # Biểu đồ độ chính xác
    fig_accuracy = go.Figure()
    fig_accuracy.add_trace(go.Scatter(y=history.history['accuracy'],␣
 ↪mode='lines', name='Train Accuracy'))
```

```
    fig_accuracy.add_trace(go.Scatter(y=history.history['val_accuracy'],␣
↪mode='lines', name='Validation Accuracy'))
    fig_accuracy.update_layout(
        title='Model Accuracy',
        xaxis_title='Epoch',
        yaxis_title='Accuracy',
        legend=dict(x=0, y=1)
    )
    fig_accuracy.show()

    # Biểu đồ loss
    fig_loss = go.Figure()
    fig_loss.add_trace(go.Scatter(y=history.history['loss'], mode='lines',␣
↪name='Train Loss'))
    fig_loss.add_trace(go.Scatter(y=history.history['val_loss'], mode='lines',␣
↪name='Validation Loss'))
    fig_loss.update_layout(
        title='Model Loss',
        xaxis_title='Epoch',
        yaxis_title='Loss',
        legend=dict(x=0, y=1)
    )
    fig_loss.show()
```

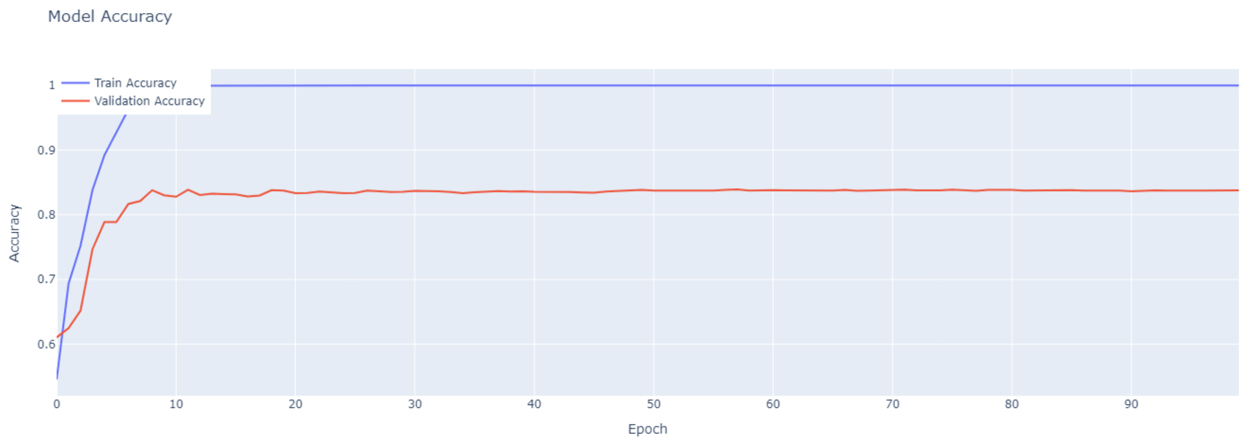
[4]:
```
import plotly.graph_objects as go

histories_list = []
def plot_histories(histories_list):
    # Biểu đồ độ chính xác
    fig_accuracy = go.Figure()
    for i, history in enumerate(histories_list):
        fig_accuracy.add_trace(go.Scatter(
            y=history.history['accuracy'],
            mode='lines',
            name=f'Train Accuracy {i+1}'
        ))
        fig_accuracy.add_trace(go.Scatter(
            y=history.history['val_accuracy'],
            mode='lines',
            name=f'Validation Accuracy {i+1}'
        ))
    fig_accuracy.update_layout(
        title='Model Accuracy',
        xaxis_title='Epoch',
        yaxis_title='Accuracy',
        legend=dict(x=0, y=1)
    )
    fig_accuracy.show()

    # Biểu đồ loss
    fig_loss = go.Figure()
    for i, history in enumerate(histories_list):
        fig_loss.add_trace(go.Scatter(
            y=history.history['loss'],
            mode='lines',
            name=f'Train Loss {i+1}'
        ))
        fig_loss.add_trace(go.Scatter(
            y=history.history['val_loss'],
            mode='lines',
            name=f'Validation Loss {i+1}'
        ))
    fig_loss.update_layout(
        title='Model Loss',
        xaxis_title='Epoch',
        yaxis_title='Loss',
        legend=dict(x=0, y=1)
    )
    fig_loss.show()
```

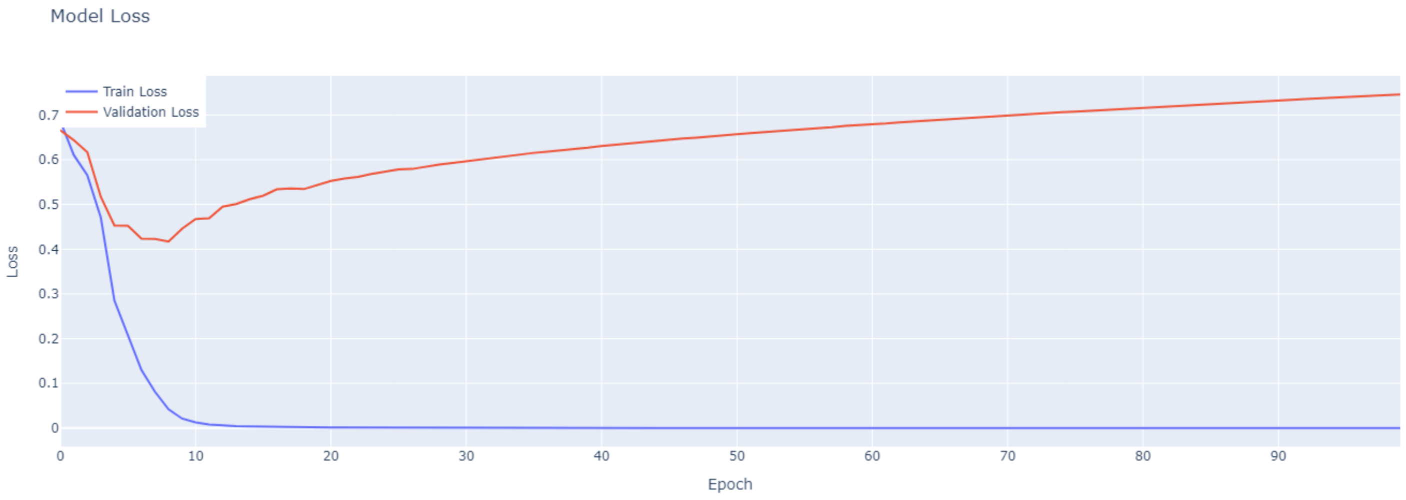
[5]:
```
# 1. RNN
model_rnn = Sequential([
    Input(shape=(x_train.shape[1],)),
    Embedding(input_dim=vocab_size, output_dim=embedding_dim),
    SimpleRNN(units=128, activation='tanh'),
    Dense(1, activation='sigmoid')
])
# Cấu hình chung
model_rnn.compile(optimizer=Adam(learning_rate=0.001),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

# Huấn luyện
start = time.time()
history_model_rnn = model_rnn.fit(x_train, y_train, epochs=100,␣
 ↪batch_size=2048, validation_data=(x_test, y_test), verbose="False")
RNN_time = time.time() - start
plot_history(history_model_rnn)
histories_list.append(history_model_rnn)
```

```
Epoch 1/100

WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
I0000 00:00:1732544492.289634      64 service.cc:145] XLA service 0x5b8a53bef4c0
```

```
initialized for platform CUDA (this does not guarantee that XLA will be used).
Devices:
I0000 00:00:1732544492.289723      64 service.cc:153]   StreamExecutor device
(0): Tesla P100-PCIE-16GB, Compute Capability 6.0
I0000 00:00:1732544493.324437      64 device_compiler.h:188] Compiled cluster
using XLA!  This line is logged at most once for the lifetime of the process.
```

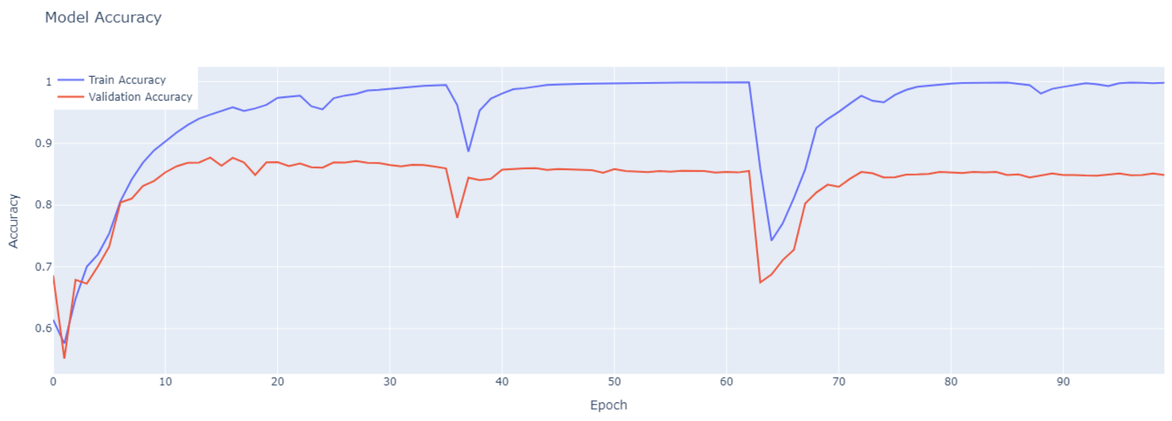
```
Epoch 2/100
Epoch 3/100
Epoch 4/100
Epoch 5/100
Epoch 6/100
Epoch 7/100
Epoch 8/100
Epoch 9/100
Epoch 10/100
Epoch 11/100
Epoch 12/100
Epoch 13/100
Epoch 14/100
Epoch 15/100
Epoch 16/100
Epoch 17/100
Epoch 18/100
Epoch 19/100
Epoch 20/100
Epoch 21/100
Epoch 22/100
Epoch 23/100
Epoch 24/100
Epoch 25/100
Epoch 26/100
Epoch 27/100
Epoch 28/100
Epoch 29/100
Epoch 30/100
Epoch 31/100
Epoch 32/100
Epoch 33/100
Epoch 34/100
Epoch 35/100
Epoch 36/100
Epoch 37/100
Epoch 38/100
Epoch 39/100
Epoch 40/100
Epoch 41/100
Epoch 42/100
Epoch 43/100
Epoch 44/100
Epoch 45/100
Epoch 46/100
Epoch 47/100
Epoch 48/100
Epoch 49/100
Epoch 50/100
Epoch 51/100
Epoch 52/100
Epoch 53/100
Epoch 54/100
Epoch 55/100
Epoch 56/100
Epoch 57/100
Epoch 58/100
Epoch 59/100
Epoch 60/100
Epoch 61/100
Epoch 62/100
Epoch 63/100
Epoch 64/100
Epoch 65/100
Epoch 66/100
Epoch 67/100
Epoch 68/100
Epoch 69/100
Epoch 70/100
Epoch 71/100
Epoch 72/100
Epoch 73/100
Epoch 74/100
Epoch 75/100
Epoch 76/100
Epoch 77/100
Epoch 78/100
Epoch 79/100
Epoch 80/100
Epoch 81/100
Epoch 82/100
Epoch 83/100
Epoch 84/100
Epoch 85/100
Epoch 86/100
Epoch 87/100
Epoch 88/100
Epoch 89/100
Epoch 90/100
```





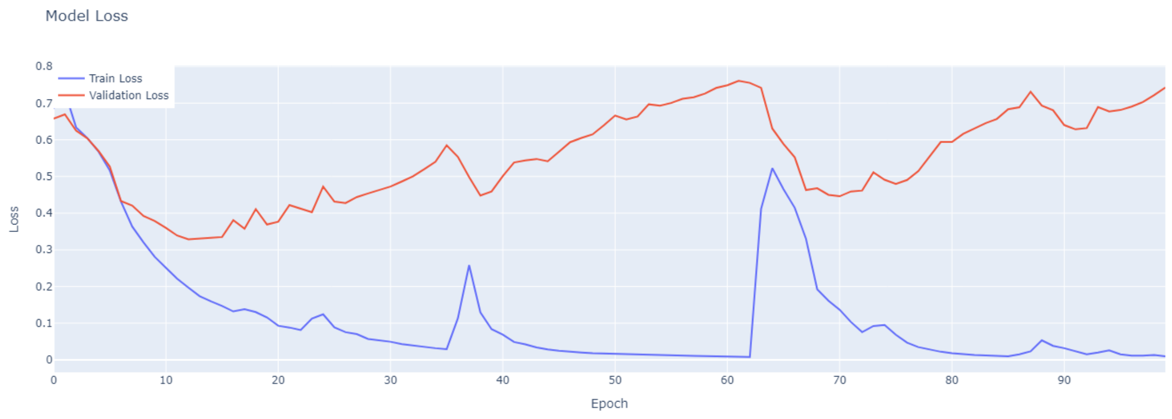
[6]:
```
# 2. LSTM
model_lstm = Sequential([
    Input(shape=(x_train.shape[1],)),
    Embedding(input_dim=vocab_size, output_dim=embedding_dim),
    LSTM(units=128, activation='tanh'),
    Dense(1, activation='sigmoid')
])
# Cấu hình chung
model_lstm.compile(optimizer=Adam(learning_rate=0.001),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

# Huấn luyện
start = time.time()
history_model_lstm = model_lstm.fit(x_train, y_train, epochs=100,␣
 ↪batch_size=2048, validation_split=0.2, verbose="False")
LSTM_time = time.time() - start
plot_history(history_model_lstm)
histories_list.append(history_model_lstm)
```

```
Epoch 1/100
Epoch 2/100
Epoch 3/100
Epoch 4/100
Epoch 5/100
Epoch 6/100
Epoch 7/100
Epoch 8/100
Epoch 9/100
Epoch 10/100
Epoch 11/100
Epoch 12/100
Epoch 13/100
Epoch 14/100
Epoch 15/100
Epoch 16/100
Epoch 17/100
Epoch 18/100
Epoch 19/100
Epoch 20/100
Epoch 21/100
Epoch 22/100
Epoch 23/100
Epoch 24/100
Epoch 25/100
Epoch 26/100
Epoch 27/100
Epoch 28/100
Epoch 29/100
Epoch 30/100
Epoch 31/100
Epoch 32/100
Epoch 33/100
Epoch 34/100
Epoch 35/100
Epoch 36/100
Epoch 37/100
Epoch 38/100
Epoch 39/100
Epoch 40/100
Epoch 41/100
Epoch 42/100
Epoch 43/100
Epoch 44/100
Epoch 45/100
Epoch 46/100
Epoch 47/100
Epoch 48/100
Epoch 49/100
Epoch 50/100
Epoch 51/100
Epoch 52/100
Epoch 53/100
Epoch 54/100
Epoch 55/100
Epoch 56/100
Epoch 57/100
Epoch 58/100
Epoch 59/100
Epoch 60/100
Epoch 61/100
Epoch 62/100
Epoch 63/100
Epoch 64/100
Epoch 65/100
Epoch 66/100
Epoch 67/100
Epoch 68/100
Epoch 69/100
Epoch 70/100
Epoch 71/100
Epoch 72/100
Epoch 73/100
Epoch 74/100
Epoch 75/100
Epoch 76/100
Epoch 77/100
Epoch 78/100
Epoch 79/100
Epoch 80/100
Epoch 81/100
Epoch 82/100
Epoch 83/100
Epoch 84/100
Epoch 85/100
Epoch 86/100
Epoch 87/100
Epoch 88/100
Epoch 89/100
Epoch 90/100
Epoch 91/100
Epoch 92/100
Epoch 93/100
Epoch 94/100
Epoch 95/100
Epoch 96/100
```



```
[ ]:
```

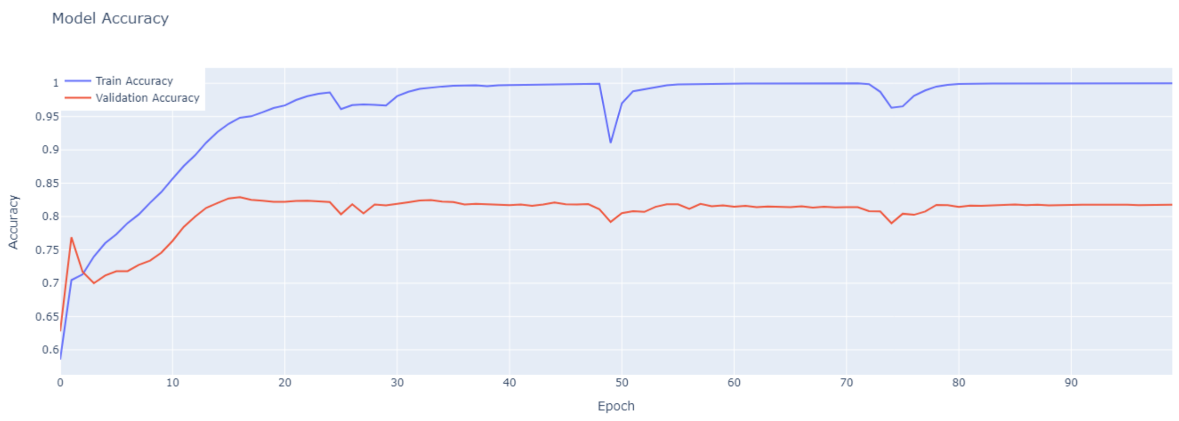
```
[ ]:
```



```
[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:
```

```
[7]: # 3. GRU
model_gru = Sequential([
    Input(shape=(x_train.shape[1],)),
    Embedding(input_dim=vocab_size, output_dim=embedding_dim),
    GRU(units=128, activation='tanh'),
    Dense(1, activation='sigmoid')
])
# Cấu hình chung
model_gru.compile(optimizer=Adam(learning_rate=0.001),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

# Huấn luyện
start = time.time()
history_model_gru = model_gru.fit(x_train, y_train, epochs=100,␣
 ↪batch_size=2048, validation_split=0.2, verbose="False")
GRU_time = time.time() - start
plot_history(history_model_gru)
histories_list.append(history_model_gru)
```

```
Epoch 1/100
Epoch 2/100
Epoch 3/100
Epoch 4/100
Epoch 5/100
Epoch 6/100
Epoch 7/100
Epoch 8/100
Epoch 9/100
Epoch 10/100
Epoch 11/100
Epoch 12/100
Epoch 13/100
Epoch 14/100
Epoch 15/100
Epoch 16/100
Epoch 17/100
Epoch 18/100
Epoch 19/100
Epoch 20/100
Epoch 21/100
Epoch 22/100
Epoch 23/100
Epoch 24/100
Epoch 25/100
Epoch 26/100
Epoch 27/100
Epoch 28/100
Epoch 29/100
Epoch 30/100
Epoch 31/100
Epoch 32/100
Epoch 33/100
Epoch 34/100
Epoch 35/100
Epoch 36/100
Epoch 37/100
Epoch 38/100
Epoch 39/100
Epoch 40/100
Epoch 41/100
Epoch 42/100
Epoch 43/100
Epoch 44/100
Epoch 45/100
Epoch 46/100
Epoch 47/100
Epoch 48/100
Epoch 49/100
Epoch 50/100
Epoch 51/100
Epoch 52/100
Epoch 53/100
Epoch 54/100
Epoch 55/100
Epoch 56/100
Epoch 57/100
Epoch 58/100
Epoch 59/100
Epoch 60/100
Epoch 61/100
Epoch 62/100
Epoch 63/100
Epoch 64/100
Epoch 65/100
Epoch 66/100
Epoch 67/100
Epoch 68/100
Epoch 69/100
Epoch 70/100
Epoch 71/100
Epoch 72/100
Epoch 73/100
Epoch 74/100
Epoch 75/100
Epoch 76/100
Epoch 77/100
Epoch 78/100
Epoch 79/100
Epoch 80/100
Epoch 81/100
Epoch 82/100
Epoch 83/100
Epoch 84/100
Epoch 85/100
Epoch 86/100
Epoch 87/100
Epoch 88/100
```



[ ]:

[ ]:

[ ]:

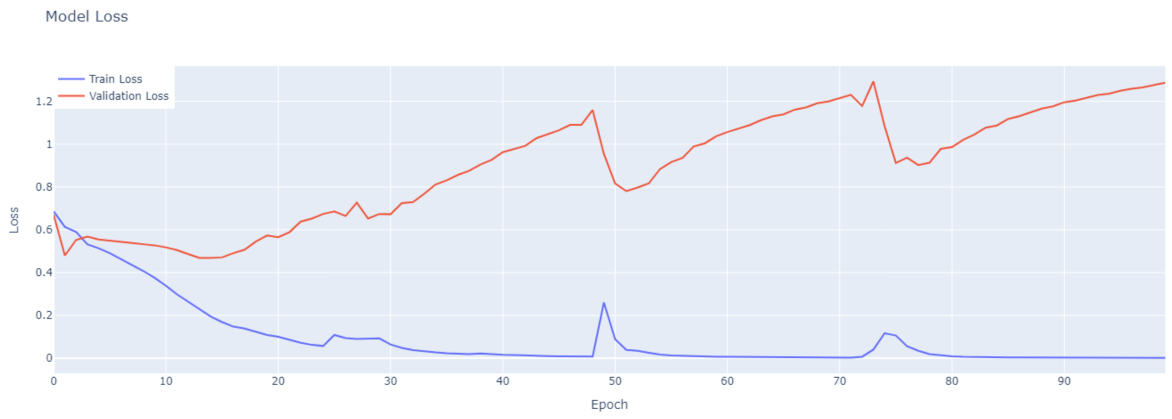
[ ]:

[ ]:

[ ]:

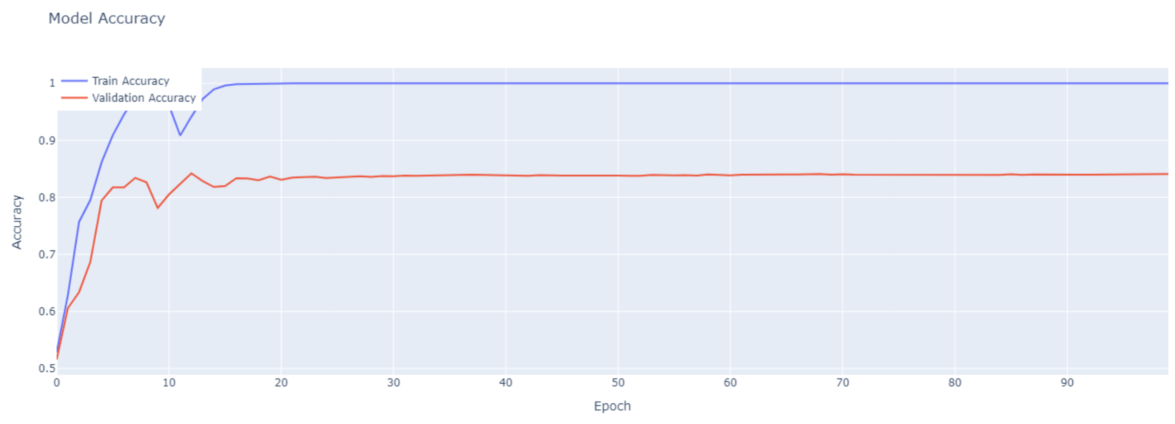
[ ]:

[ ]:



[8]:
```
# 4. Bi-RNN
model_birnn = Sequential([
    Embedding(input_dim=vocab_size, output_dim=embedding_dim,␣
  ↪input_length=max_length),
    Bidirectional(SimpleRNN(units=128, activation='tanh')),
    Dense(1, activation='sigmoid')
])
# Cấu hình chung
model_birnn.compile(optimizer=Adam(learning_rate=0.001),
                    loss='binary_crossentropy',
                    metrics=['accuracy'])

# Huấn luyện
start = time.time()
history_model_birnn = model_birnn.fit(x_train, y_train, epochs=100,␣
  ↪batch_size=2048, validation_split=0.2, verbose="False")
Bi_RNN_time = time.time() - start
plot_history(history_model_birnn)
histories_list.append(history_model_birnn)
```

```
Epoch 1/100

/opt/conda/lib/python3.10/site-packages/keras/src/layers/core/embedding.py:90:
UserWarning:

Argument `input_length` is deprecated. Just remove it.


Epoch 2/100
Epoch 3/100
Epoch 4/100
Epoch 5/100
Epoch 6/100
Epoch 7/100
Epoch 8/100
Epoch 9/100
Epoch 10/100
Epoch 11/100
Epoch 12/100
Epoch 13/100
Epoch 14/100
Epoch 15/100
Epoch 16/100
Epoch 17/100
Epoch 18/100
Epoch 19/100
Epoch 20/100
Epoch 21/100
Epoch 22/100
Epoch 23/100
Epoch 24/100
Epoch 25/100
Epoch 26/100
Epoch 27/100
Epoch 28/100
Epoch 29/100
Epoch 30/100
Epoch 31/100
Epoch 32/100
Epoch 33/100
Epoch 34/100
Epoch 35/100
Epoch 36/100
Epoch 37/100
Epoch 38/100
Epoch 39/100
Epoch 40/100
Epoch 41/100
Epoch 42/100
Epoch 43/100
Epoch 44/100
Epoch 45/100
Epoch 46/100
Epoch 47/100
Epoch 48/100
Epoch 49/100
Epoch 50/100
Epoch 51/100
Epoch 52/100
Epoch 53/100
Epoch 54/100
Epoch 55/100
Epoch 56/100
Epoch 57/100
Epoch 58/100
Epoch 59/100
Epoch 60/100
Epoch 61/100
Epoch 62/100
Epoch 63/100
Epoch 64/100
Epoch 65/100
Epoch 66/100
Epoch 67/100
Epoch 68/100
Epoch 69/100
Epoch 70/100
Epoch 71/100
Epoch 72/100
Epoch 73/100
Epoch 74/100
Epoch 75/100
Epoch 76/100
Epoch 77/100
Epoch 78/100
Epoch 79/100
Epoch 80/100
Epoch 81/100
Epoch 82/100
Epoch 83/100
Epoch 84/100
Epoch 85/100
Epoch 86/100
Epoch 87/100
```



-

```
[ ]:
```

```
[ ]:
```

[ ]:

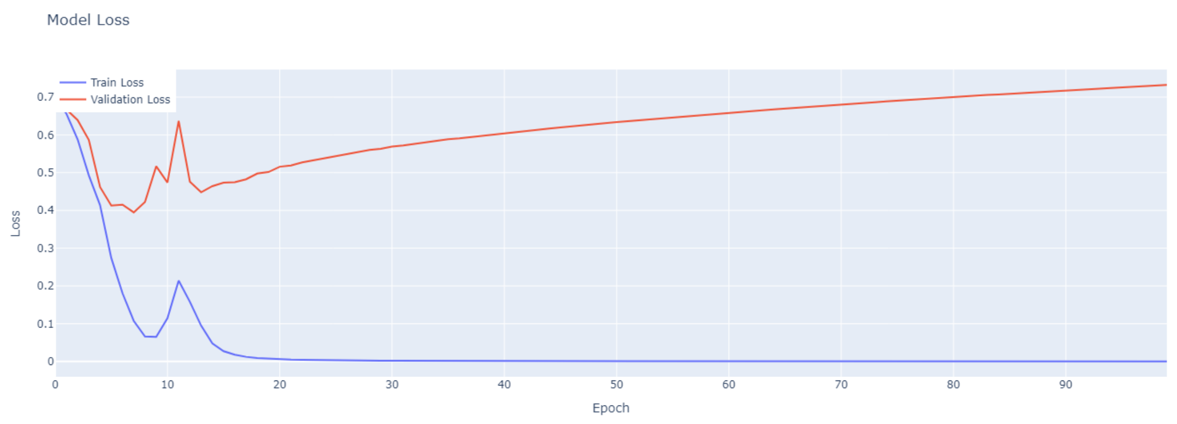
[ ]:

[ ]:

[ ]:

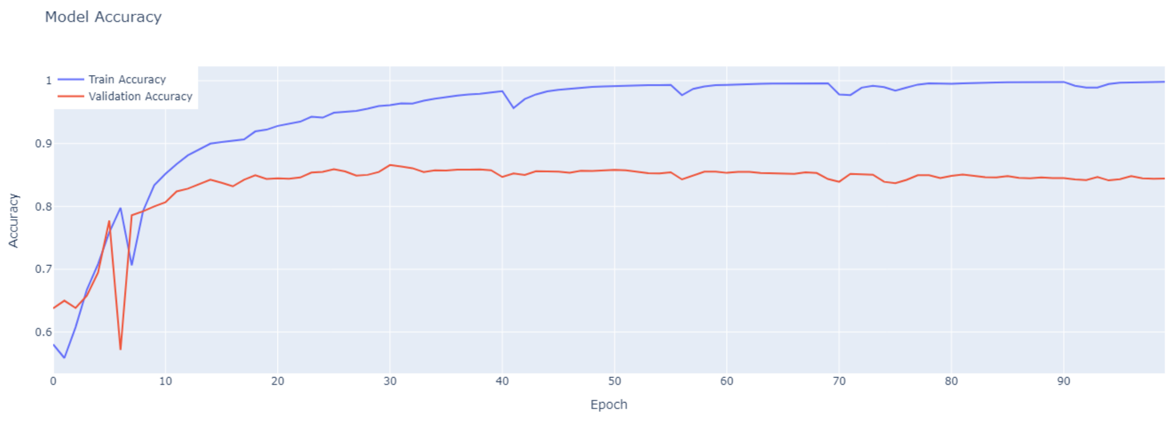
[ ]:

[ ]:



[9]:
```
# 5. Bi-LSTM
model_bilstm = Sequential([
    Embedding(input_dim=vocab_size, output_dim=embedding_dim,␣
 ↪input_length=max_length),
    Bidirectional(LSTM(units=128, activation='tanh')),
    Dense(1, activation='sigmoid')
])
# Cấu hình chung
model_bilstm.compile(optimizer=Adam(learning_rate=0.001),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

# Huấn luyện
start = time.time()
history_model_bilstm = model_bilstm.fit(x_train, y_train, epochs=100,␣
 ↪batch_size=2048, validation_split=0.2, verbose="False")
Bi_LSTM_time = time.time() - start
plot_history(history_model_bilstm)
histories_list.append(history_model_bilstm)
```

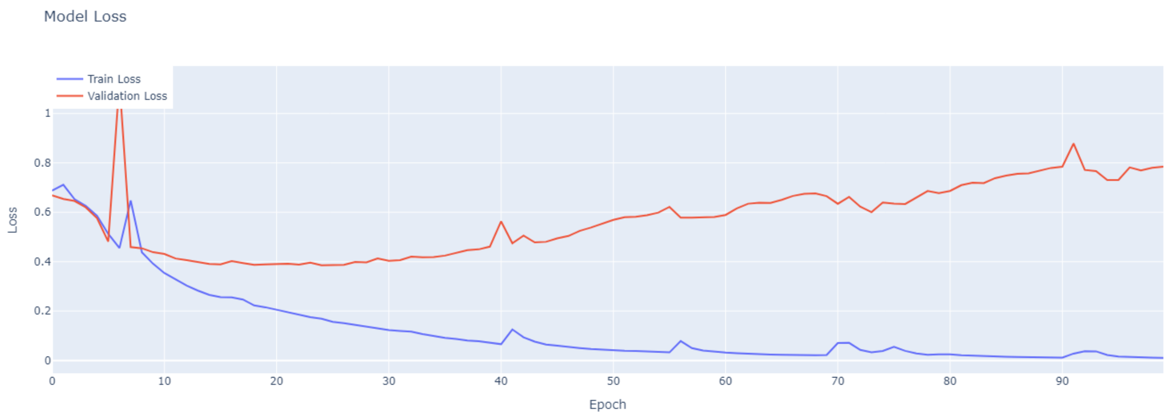
```
Epoch 1/100
Epoch 2/100
Epoch 3/100
Epoch 4/100
Epoch 5/100
Epoch 6/100
Epoch 7/100
Epoch 8/100
Epoch 9/100
Epoch 10/100
Epoch 11/100
Epoch 12/100
Epoch 13/100
Epoch 14/100
Epoch 15/100
Epoch 16/100
Epoch 17/100
Epoch 18/100
Epoch 19/100
Epoch 20/100
Epoch 21/100
Epoch 22/100
Epoch 23/100
Epoch 24/100
Epoch 25/100
Epoch 26/100
Epoch 27/100
Epoch 28/100
Epoch 29/100
Epoch 30/100
Epoch 31/100
Epoch 32/100
Epoch 33/100
Epoch 34/100
Epoch 35/100
Epoch 36/100
Epoch 37/100
Epoch 38/100
Epoch 39/100
Epoch 40/100
Epoch 41/100
Epoch 42/100
Epoch 43/100
Epoch 44/100
Epoch 45/100
Epoch 46/100
Epoch 47/100
Epoch 48/100
Epoch 49/100
Epoch 50/100
Epoch 51/100
Epoch 52/100
Epoch 53/100
Epoch 54/100
Epoch 55/100
Epoch 56/100
Epoch 57/100
Epoch 58/100
Epoch 59/100
Epoch 60/100
Epoch 61/100
Epoch 62/100
Epoch 63/100
Epoch 64/100
Epoch 65/100
Epoch 66/100
Epoch 67/100
Epoch 68/100
Epoch 69/100
Epoch 70/100
Epoch 71/100
Epoch 72/100
Epoch 73/100
Epoch 74/100
Epoch 75/100
Epoch 76/100
Epoch 77/100
Epoch 78/100
Epoch 79/100
Epoch 80/100
Epoch 81/100
Epoch 82/100
Epoch 83/100
Epoch 84/100
Epoch 85/100
Epoch 86/100
Epoch 87/100
Epoch 88/100
Epoch 89/100
Epoch 90/100
Epoch 91/100
Epoch 92/100
Epoch 93/100
Epoch 94/100
```



```
[ ]:
```

```
[ ]:
```

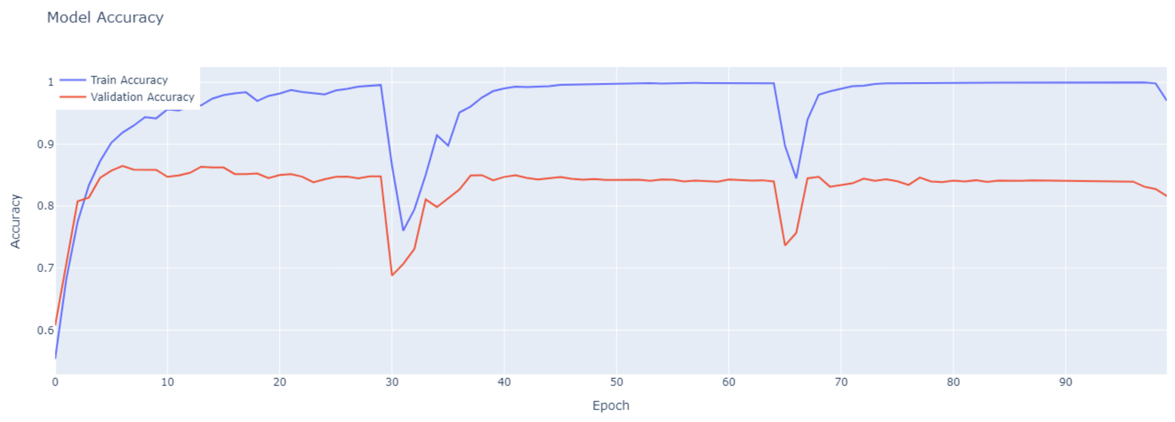
```
[ ]:
```



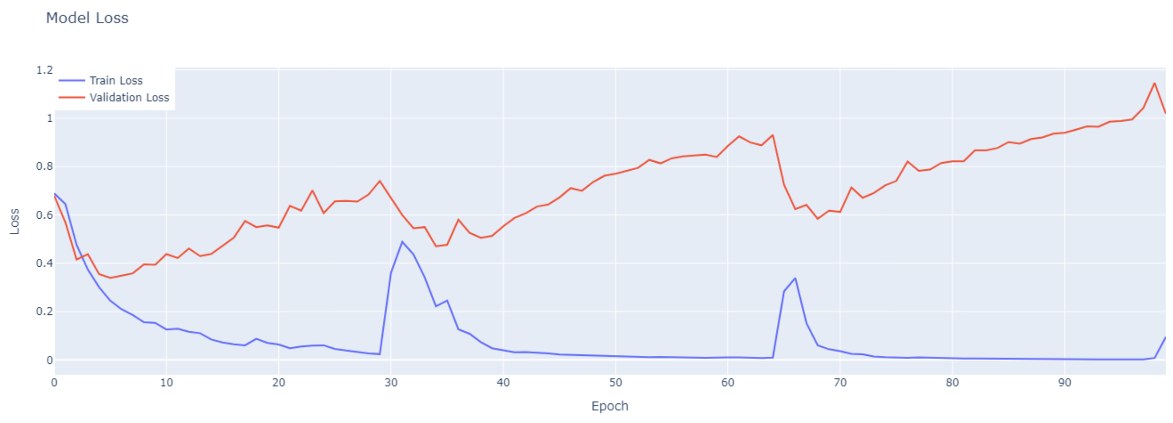
```
[ ]:

[ ]:

[ ]:

[ ]:

[ ]:
```

```
[10]: # 6. Bi-GRU
model_bigru = Sequential([
    Embedding(input_dim=vocab_size, output_dim=embedding_dim,␣
 ↪input_length=max_length),
    Bidirectional(GRU(units=128, activation='tanh')),
    Dense(1, activation='sigmoid')
])
# Cấu hình chung
model_bigru.compile(optimizer=Adam(learning_rate=0.001),
                    loss='binary_crossentropy',
                    metrics=['accuracy'])

# Huấn luyện
start = time.time()
history_model_bigru = model_bigru.fit(x_train, y_train, epochs=100,␣
 ↪batch_size=2048, validation_split=0.2, verbose="False")
Bi_GRU_time = time.time() - start
plot_history(history_model_bigru)
histories_list.append(history_model_bigru)
```

```
Epoch 1/100
Epoch 2/100
Epoch 3/100
Epoch 4/100
Epoch 5/100
Epoch 6/100
Epoch 7/100
Epoch 8/100
Epoch 9/100
Epoch 10/100
Epoch 11/100
Epoch 12/100
Epoch 13/100
Epoch 14/100
Epoch 15/100
Epoch 16/100
Epoch 17/100
Epoch 18/100
Epoch 19/100
Epoch 20/100
Epoch 21/100
Epoch 22/100
Epoch 23/100
Epoch 24/100
Epoch 25/100
Epoch 26/100
Epoch 27/100
Epoch 28/100
Epoch 29/100
Epoch 30/100
Epoch 31/100
Epoch 32/100
Epoch 33/100
Epoch 34/100
Epoch 35/100
Epoch 36/100
Epoch 37/100
Epoch 38/100
Epoch 39/100
Epoch 40/100
Epoch 41/100
Epoch 42/100
Epoch 43/100
Epoch 44/100
Epoch 45/100
Epoch 46/100
Epoch 47/100
Epoch 48/100
Epoch 49/100
Epoch 50/100
Epoch 51/100
Epoch 52/100
Epoch 53/100
Epoch 54/100
Epoch 55/100
Epoch 56/100
Epoch 57/100
Epoch 58/100
Epoch 59/100
Epoch 60/100
Epoch 61/100
Epoch 62/100
Epoch 63/100
Epoch 64/100
Epoch 65/100
Epoch 66/100
Epoch 67/100
Epoch 68/100
Epoch 69/100
Epoch 70/100
Epoch 71/100
Epoch 72/100
Epoch 73/100
Epoch 74/100
Epoch 75/100
Epoch 76/100
Epoch 77/100
Epoch 78/100
Epoch 79/100
Epoch 80/100
Epoch 81/100
Epoch 82/100
Epoch 83/100
Epoch 84/100
Epoch 85/100
Epoch 86/100
Epoch 87/100
Epoch 88/100
```



```
[ ]:
```



```
[ ]:

[ ]:

[ ]:

[ ]:

[ ]:
```

[ ]:

```

```

[ ]:

```

```

[11]:

```
data = {'RNN_time': RNN_time, 'LSTM_time': LSTM_time, 'GRU_time': GRU_time,␣
↪'Bi_RNN_time': Bi_RNN_time, 'Bi_LSTM_time': Bi_LSTM_time, 'Bi_GRU_time':␣
↪Bi_GRU_time}
sorted_by_key = dict(sorted(data.items(), key=lambda item: item[1]))  # Sắp xếp␣
↪key theo thứ tự tăng dần
print(sorted_by_key)
```

```
{'RNN_time': 270.2134635448456, 'GRU_time': 279.0114424228668, 'LSTM_time':
307.46430110931396, 'Bi_RNN_time': 323.57692217826843, 'Bi_GRU_time':
521.9530775547028, 'Bi_LSTM_time': 573.4361057281494}
```

[12]:

```
plot_histories(histories_list)
```

1 7. Conclusion

The training time comparison is as follows: Bi_LSTM_time > Bi_GRU_time > Bi_RNN_time > LSTM_time > GRU_time > RNN_time.

Most models achieve high accuracy during training, but their performance on the validation set is not as strong as on the training set. Among the models, Bi-LSTM and Bi-RNN show the lowest error rates on the validation set. But in opposite, GRU is highest error rates.

LSTM, Bi-LSTM is the highest accuracy model in vali. RNN and GRU is lower than the other models.