Natural Language Processing

Assignment-07

Name: Adithi Shinde
No:2203A54032

1.Use a simple dataset for English-to-French translation. You can either use a small dataset like this or download a more extensive dataset such as the **Tab-delimited Bilingual Sentence Pairs** dataset from Tatoeba or **Parallel Corpus** from the European Parliament.

Example data (small English to French pairs)

data = [("hello", "bonjour"), ("how are you", "comment ça va"), ("I am fine", "je vais bien"), ("what is your name", "comment tu t'appelles"), ("my name is", "je m'appelle"), ("thank you", "merci"), ("goodbye", "au revoir")] [CO4]

(a)Data Preprocessing

```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Sample dataset
data = [
    ("hello", "bonjour"),
    ("how are you", "comment ça va"),
    ("I am fine", "je vais bien"),
    ("what is your name", "comment tu t'appelles"),
    ("my name is", "je m'appelle"),
    ("thank you", "merci"),
    ("goodbye", "au revoir")
]

# Splitting English and French phrases
english_texts, french_texts = zip(*data)

# Tokenizing and padding
english_tokenizer = Tokenizer()
french_tokenizer = Tokenizer()

english_tokenizer.fit_on_texts(english_texts)
french_tokenizer.fit_on_texts(french_texts)

# Convert text to sequences
english_sequences = english_tokenizer.texts_to_sequences(english_texts)
french_sequences = french_tokenizer.texts_to_sequences(french_texts)
```

```
# Padding sequences
english_sequences = pad_sequences(english_sequences, padding='post')
french_sequences = pad_sequences(french_sequences, padding='post')
```

(b) Build Seq2Seq Model

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, LSTM, Dense, Embedding

# Model parameters
embedding_dim = 64
lstm_units = 128

# Encoder
encoder_inputs = Input(shape=(None,))
encoder_embedding = Embedding(input_dim=len(english_tokenizer.word_index) + 1, output_dim=embedding_dim)(encoder_inputs)
encoder_lstm, state_h, state_c = LSTM(lstm_units, return_state=True)(encoder_embedding)
encoder_states = [state_h, state_c]

# Decoder
decoder_inputs = Input(shape=(None,))
decoder_embedding = Embedding(input_dim=len(french_tokenizer.word_index) + 1, output_dim=embedding_dim)(decoder_inputs)
decoder_lstm = LSTM(lstm_units, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_embedding, initial_state=encoder_states)
decoder_dense = Dense(len(french_tokenizer.word_index) + 1, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)

# Define model
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

(c) Preparing the Data for Training

```
[ ] import numpy as np

    # Prepare decoder input and target data
    decoder_input_data = french_sequences[:, :-1]
    decoder_target_data = french_sequences[:, 1:]
    decoder_target_data = np.expand_dims(decoder_target_data, -1)
```

(d) Train the model on the dataset

```
# Training
epochs = 100
batch_size = 64
model.fit([english_sequences, decoder_input_data], decoder_target_data,
          batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

```
Epoch 1/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 5s 5s/step - accuracy: 0.2000 - loss: 2.6379 - val_accuracy: 0.7500 - val_loss: 2.6233
Epoch 2/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 218ms/step - accuracy: 0.4000 - loss: 2.6245 - val_accuracy: 0.7500 - val_loss: 2.6078
Epoch 3/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 0.4000 - loss: 2.6109 - val_accuracy: 0.7500 - val_loss: 2.5914
Epoch 4/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - accuracy: 0.3000 - loss: 2.5967 - val_accuracy: 0.7500 - val_loss: 2.5737
Epoch 5/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 0.3000 - loss: 2.5817 - val_accuracy: 0.7500 - val_loss: 2.5540
Epoch 6/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.3000 - loss: 2.5655 - val_accuracy: 0.7500 - val_loss: 2.5319
Epoch 7/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 0.3000 - loss: 2.5478 - val_accuracy: 0.7500 - val_loss: 2.5067
Epoch 8/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.3000 - loss: 2.5281 - val_accuracy: 0.7500 - val_loss: 2.4778
Epoch 9/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.3000 - loss: 2.5062 - val_accuracy: 0.7500 - val_loss: 2.4445
Epoch 10/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - accuracy: 0.3000 - loss: 2.4815 - val_accuracy: 0.7500 - val_loss: 2.4059
Epoch 11/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.3000 - loss: 2.4536 - val_accuracy: 0.7500 - val_loss: 2.3611
Epoch 12/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 0.3000 - loss: 2.4217 - val_accuracy: 0.7500 - val_loss: 2.3092
Epoch 13/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - accuracy: 0.3000 - loss: 2.3854 - val_accuracy: 0.7500 - val_loss: 2.2489
Epoch 14/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step - accuracy: 0.3000 - loss: 2.3438 - val_accuracy: 0.7500 - val_loss: 2.1792
Epoch 15/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 85ms/step - accuracy: 0.3000 - loss: 2.2963 - val_accuracy: 0.7500 - val_loss: 2.0989
Epoch 16/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 124ms/step - accuracy: 0.3000 - loss: 2.2422 - val_accuracy: 0.7500 - val_loss: 2.0076
```

```
Epoch 17/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.3000 - loss: 2.1811 - val_accuracy: 0.7500 - val_loss: 1.9055
Epoch 18/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 0.3000 - loss: 2.1134 - val_accuracy: 0.7500 - val_loss: 1.7951
Epoch 19/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.3000 - loss: 2.0406 - val_accuracy: 0.7500 - val_loss: 1.6823
Epoch 20/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.3000 - loss: 1.9664 - val_accuracy: 0.7500 - val_loss: 1.5789
Epoch 21/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 0.3000 - loss: 1.8964 - val_accuracy: 0.7500 - val_loss: 1.5020
Epoch 22/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.3000 - loss: 1.8377 - val_accuracy: 0.7500 - val_loss: 1.4685
Epoch 23/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.3000 - loss: 1.7949 - val_accuracy: 0.7500 - val_loss: 1.4848
Epoch 24/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.3000 - loss: 1.7673 - val_accuracy: 0.7500 - val_loss: 1.5414
Epoch 25/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 0.3000 - loss: 1.7472 - val_accuracy: 0.7500 - val_loss: 1.6190
Epoch 26/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.3000 - loss: 1.7245 - val_accuracy: 0.7500 - val_loss: 1.7000
Epoch 27/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 0.3000 - loss: 1.6933 - val_accuracy: 0.7500 - val_loss: 1.7745
Epoch 28/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step - accuracy: 0.3000 - loss: 1.6533 - val_accuracy: 0.7500 - val_loss: 1.8394
Epoch 29/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 145ms/step - accuracy: 0.3000 - loss: 1.6085 - val_accuracy: 0.7500 - val_loss: 1.8961
Epoch 30/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 122ms/step - accuracy: 0.3000 - loss: 1.5632 - val_accuracy: 0.7500 - val_loss: 1.9473
Epoch 31/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 0.3000 - loss: 1.5204 - val_accuracy: 0.7500 - val_loss: 1.9964
Epoch 32/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 0.3000 - loss: 1.4814 - val_accuracy: 0.7500 - val_loss: 2.0463
Epoch 33/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - accuracy: 0.5000 - loss: 1.4456 - val_accuracy: 0.7500 - val_loss: 2.0994
```

```
Epoch 34/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step - accuracy: 0.7000 - loss: 1.4114 - val_accuracy: 0.7500 - val_loss: 2.1573
Epoch 35/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.7000 - loss: 1.3768 - val_accuracy: 0.7500 - val_loss: 2.2212
Epoch 36/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 0.7000 - loss: 1.3397 - val_accuracy: 0.7500 - val_loss: 2.2920
Epoch 37/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - accuracy: 0.8000 - loss: 1.2986 - val_accuracy: 0.7500 - val_loss: 2.3702
Epoch 38/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 68ms/step - accuracy: 0.8000 - loss: 1.2523 - val_accuracy: 0.7500 - val_loss: 2.4562
Epoch 39/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step - accuracy: 0.7000 - loss: 1.2004 - val_accuracy: 0.7500 - val_loss: 2.5504
Epoch 40/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.7000 - loss: 1.1430 - val_accuracy: 0.7500 - val_loss: 2.6527
Epoch 41/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step - accuracy: 0.7000 - loss: 1.0809 - val_accuracy: 0.7500 - val_loss: 2.7628
Epoch 42/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.7000 - loss: 1.0154 - val_accuracy: 0.7500 - val_loss: 2.8802
Epoch 43/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - accuracy: 0.7000 - loss: 0.9481 - val_accuracy: 0.7500 - val_loss: 3.0033
Epoch 44/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - accuracy: 0.7000 - loss: 0.8811 - val_accuracy: 0.7500 - val_loss: 3.1300
Epoch 45/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.8000 - loss: 0.8160 - val_accuracy: 0.7500 - val_loss: 3.2575
Epoch 46/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.8000 - loss: 0.7540 - val_accuracy: 0.7500 - val_loss: 3.3824
Epoch 47/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 0.8000 - loss: 0.6959 - val_accuracy: 0.7500 - val_loss: 3.5018
Epoch 48/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.8000 - loss: 0.6416 - val_accuracy: 0.7500 - val_loss: 3.6137
Epoch 49/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 0.8000 - loss: 0.5913 - val_accuracy: 0.7500 - val_loss: 3.7173
Epoch 50/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.8000 - loss: 0.5446 - val_accuracy: 0.7500 - val_loss: 3.8136
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step - accuracy: 1.0000 - loss: 0.0230 - val_accuracy: 0.7500 - val_loss: 4.9054
Epoch 85/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step - accuracy: 1.0000 - loss: 0.0216 - val_accuracy: 0.7500 - val_loss: 4.9131
Epoch 86/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - accuracy: 1.0000 - loss: 0.0203 - val_accuracy: 0.7500 - val_loss: 4.9203
Epoch 87/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 150ms/step - accuracy: 1.0000 - loss: 0.0192 - val_accuracy: 0.7500 - val_loss: 4.9271
Epoch 88/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step - accuracy: 1.0000 - loss: 0.0181 - val_accuracy: 0.7500 - val_loss: 4.9335
Epoch 89/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step - accuracy: 1.0000 - loss: 0.0172 - val_accuracy: 0.7500 - val_loss: 4.9395
Epoch 90/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 1.0000 - loss: 0.0163 - val_accuracy: 0.7500 - val_loss: 4.9451
Epoch 91/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step - accuracy: 1.0000 - loss: 0.0155 - val_accuracy: 0.7500 - val_loss: 4.9504
Epoch 92/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - accuracy: 1.0000 - loss: 0.0148 - val_accuracy: 0.7500 - val_loss: 4.9554
Epoch 93/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 1.0000 - loss: 0.0142 - val_accuracy: 0.7500 - val_loss: 4.9602
Epoch 94/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 1.0000 - loss: 0.0136 - val_accuracy: 0.7500 - val_loss: 4.9648
Epoch 95/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 1.0000 - loss: 0.0130 - val_accuracy: 0.7500 - val_loss: 4.9692
Epoch 96/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 1.0000 - loss: 0.0125 - val_accuracy: 0.7500 - val_loss: 4.9733
Epoch 97/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step - accuracy: 1.0000 - loss: 0.0121 - val_accuracy: 0.7500 - val_loss: 4.9772
Epoch 98/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 1.0000 - loss: 0.0116 - val_accuracy: 0.7500 - val_loss: 4.9809
Epoch 99/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step - accuracy: 1.0000 - loss: 0.0112 - val_accuracy: 0.7500 - val_loss: 4.9843
Epoch 100/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step - accuracy: 1.0000 - loss: 0.0108 - val_accuracy: 0.7500 - val_loss: 4.9874
<keras.src.callbacks.history.History at 0x79345bcb8d90>
```

(e) Inference Setup for Translation

```
# Encoder model for inference
encoder_model = Model(encoder_inputs, encoder_states)

# Decoder model for inference
decoder_state_input_h = Input(shape=(lstm_units,))
decoder_state_input_c = Input(shape=(lstm_units,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
decoder_outputs, state_h, state_c = decoder_lstm(decoder_embedding, initial_state=decoder_states_inputs)
decoder_states = [state_h, state_c]
decoder_outputs = decoder_dense(decoder_outputs)
decoder_model = Model([decoder_inputs] + decoder_states_inputs, [decoder_outputs] + decoder_states)
```

(f) Translate New Sentences

```
def translate_sentence(input_text):
    # Preprocess the input text
    input_seq = english_tokenizer.texts_to_sequences([input_text])
    input_seq = pad_sequences(input_seq, maxlen=english_sequences.shape[1], padding='post')

    # Encode the input
    states_value = encoder_model.predict(input_seq)

    # Generate an empty target sequence
    target_seq = np.zeros((1, 1))
    target_seq[0, 0] = french_tokenizer.word_index['starttoken']  # assuming we have a start token

    # Translation loop
    stop_condition = False
    translated_sentence = ''
    while not stop_condition:
        output_tokens, h, c = decoder_model.predict([target_seq] + states_value)

        # Sample the most probable word
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        sampled_word = french_tokenizer.index_word[sampled_token_index]
        translated_sentence += ' ' + sampled_word

        if sampled_word == 'endtoken' or len(translated_sentence.split()) > 50:
            stop_condition = True

        # Update the target sequence
        target_seq = np.zeros((1, 1))
        target_seq[0, 0] = sampled_token_index
```

```
        # Update states
        states_value = [h, c]

    return translated_sentence
```

(g) Experimenting and Improving the Model by large dataset and hyper tune parameter.

1. Modify Training Code to Include History Logging

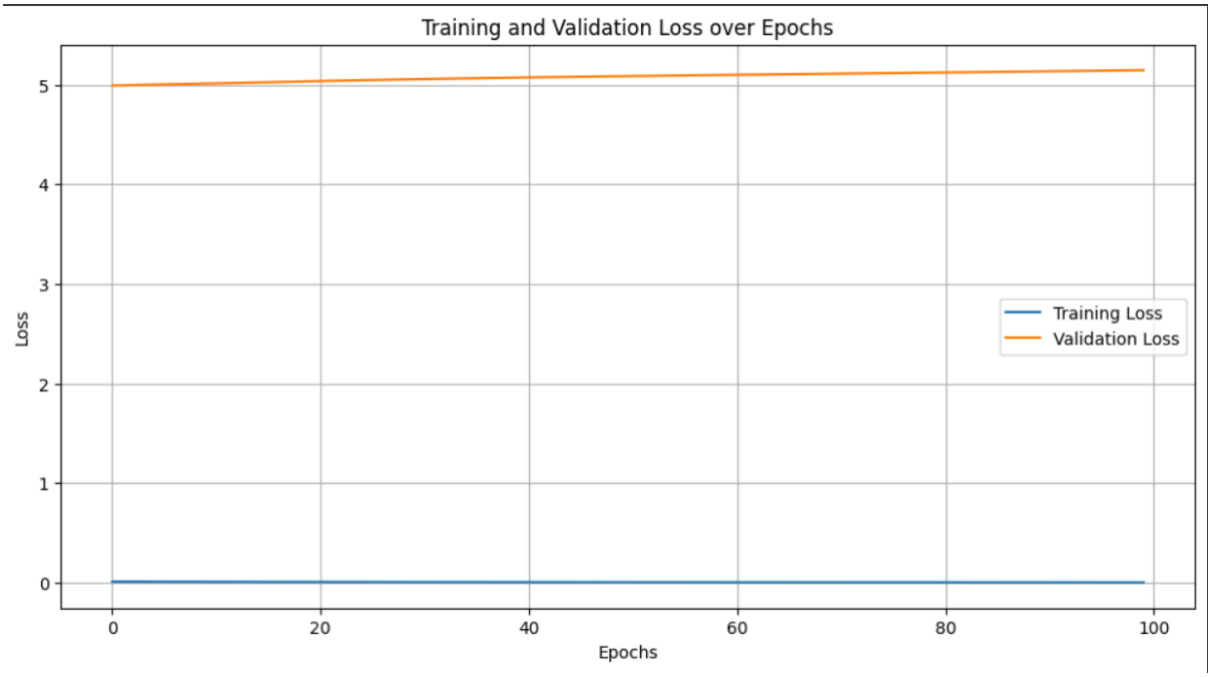
```python
import matplotlib.pyplot as plt

# Model training with history logging for analysis
epochs = 100
batch_size = 64

history = model.fit(
    [english_sequences, decoder_input_data],
    decoder_target_data,
    batch_size=batch_size,
    epochs=epochs,
    validation_split=0.2
)
```

```
Epoch 1/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 118ms/step - accuracy: 1.0000 - loss: 0.0105 - val_accuracy: 0.7500 - val_loss: 4.9904
Epoch 2/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 1.0000 - loss: 0.0101 - val_accuracy: 0.7500 - val_loss: 4.9930
Epoch 3/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 1.0000 - loss: 0.0098 - val_accuracy: 0.7500 - val_loss: 4.9955
Epoch 4/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 1.0000 - loss: 0.0095 - val_accuracy: 0.7500 - val_loss: 4.9979
Epoch 5/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 1.0000 - loss: 0.0093 - val_accuracy: 0.7500 - val_loss: 5.0000
Epoch 6/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - accuracy: 1.0000 - loss: 0.0090 - val_accuracy: 0.7500 - val_loss: 5.0021
Epoch 7/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 1.0000 - loss: 0.0087 - val_accuracy: 0.7500 - val_loss: 5.0042
Epoch 8/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - accuracy: 1.0000 - loss: 0.0085 - val_accuracy: 0.7500 - val_loss: 5.0062
Epoch 9/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 124ms/step - accuracy: 1.0000 - loss: 0.0083 - val_accuracy: 0.7500 - val_loss: 5.0082
Epoch 10/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - accuracy: 1.0000 - loss: 0.0081 - val_accuracy: 0.7500 - val_loss: 5.0103
Epoch 11/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 1.0000 - loss: 0.0079 - val_accuracy: 0.7500 - val_loss: 5.0124
Epoch 12/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 1.0000 - loss: 0.0077 - val_accuracy: 0.7500 - val_loss: 5.0146
Epoch 13/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 1.0000 - loss: 0.0075 - val_accuracy: 0.7500 - val_loss: 5.0169
Epoch 14/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 1.0000 - loss: 0.0073 - val_accuracy: 0.7500 - val_loss: 5.0192
Epoch 15/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 1.0000 - loss: 0.0072 - val_accuracy: 0.7500 - val_loss: 5.0215
Epoch 16/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 1.0000 - loss: 0.0070 - val_accuracy: 0.7500 - val_loss: 5.0239
Epoch 17/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - accuracy: 1.0000 - loss: 0.0069 - val_accuracy: 0.7500 - val_loss: 5.0263
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step - accuracy: 1.0000 - loss: 0.0026 - val_accuracy: 0.7500 - val_loss: 5.1256
Epoch 85/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - accuracy: 1.0000 - loss: 0.0026 - val_accuracy: 0.7500 - val_loss: 5.1268
Epoch 86/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 127ms/step - accuracy: 1.0000 - loss: 0.0026 - val_accuracy: 0.7500 - val_loss: 5.1280
Epoch 87/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 1.0000 - loss: 0.0025 - val_accuracy: 0.7500 - val_loss: 5.1292
Epoch 88/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 1.0000 - loss: 0.0025 - val_accuracy: 0.7500 - val_loss: 5.1304
Epoch 89/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - accuracy: 1.0000 - loss: 0.0025 - val_accuracy: 0.7500 - val_loss: 5.1316
Epoch 90/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 1.0000 - loss: 0.0025 - val_accuracy: 0.7500 - val_loss: 5.1328
Epoch 91/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - accuracy: 1.0000 - loss: 0.0024 - val_accuracy: 0.7500 - val_loss: 5.1340
Epoch 92/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - accuracy: 1.0000 - loss: 0.0024 - val_accuracy: 0.7500 - val_loss: 5.1352
Epoch 93/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 1.0000 - loss: 0.0024 - val_accuracy: 0.7500 - val_loss: 5.1363
Epoch 94/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - accuracy: 1.0000 - loss: 0.0024 - val_accuracy: 0.7500 - val_loss: 5.1375
Epoch 95/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step - accuracy: 1.0000 - loss: 0.0023 - val_accuracy: 0.7500 - val_loss: 5.1387
Epoch 96/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - accuracy: 1.0000 - loss: 0.0023 - val_accuracy: 0.7500 - val_loss: 5.1399
Epoch 97/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - accuracy: 1.0000 - loss: 0.0023 - val_accuracy: 0.7500 - val_loss: 5.1411
Epoch 98/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 1.0000 - loss: 0.0023 - val_accuracy: 0.7500 - val_loss: 5.1422
Epoch 99/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 66ms/step - accuracy: 1.0000 - loss: 0.0022 - val_accuracy: 0.7500 - val_loss: 5.1434
Epoch 100/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 139ms/step - accuracy: 1.0000 - loss: 0.0022 - val_accuracy: 0.7500 - val_loss: 5.1446
```

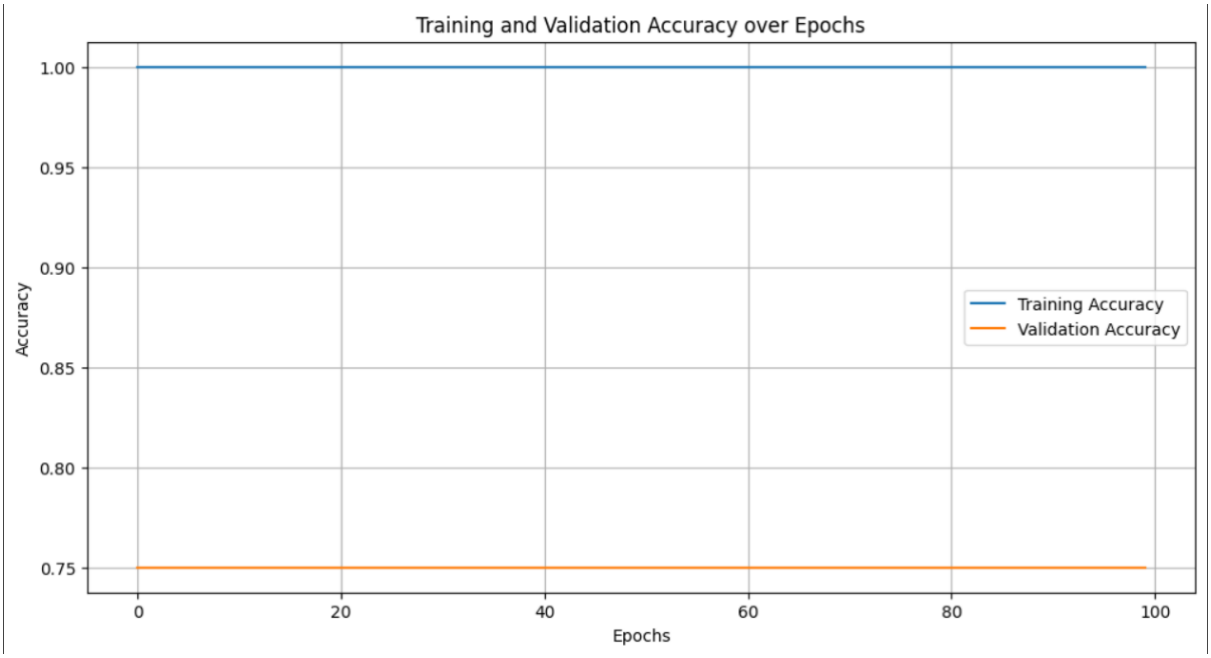
2.Plot Training and Validation Loss

```python
# Plotting training and validation loss
plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss over Epochs')
plt.legend()
plt.grid()
plt.show()
```



3. Plot Training and Validation Accuracy

```
# Plotting training and validation accuracy
plt.figure(figsize=(12, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy over Epochs')
plt.legend()
plt.grid()
plt.show()
```


Training and Validation Accuracy over Epochs
1.00
0.95
0.90
0.85
0.80
0.75
Accuracy
0
20
40
60
80
100
Epochs
Training Accuracy
Validation Accuracy