lab-mse-part-b

December 4, 2023

data augmentation

[1]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
```

```
WARNING:tensorflow:From C:\Users\HP\anaconda3\envs\AML-manyahegde\lib\site-
packages\keras\src\losses.py:2976: The name
tf.losses.sparse_softmax_cross_entropy is deprecated. Please use
tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

[2]:
```
(x_train,y_train),(x_test,y_test)=keras.datasets.mnist.load_data()
```

[3]:
```
x_train=x_train.astype('float32')/255.0
x_test=x_test.astype('float32')/255.0

x_train=np.expand_dims(x_train,-1)
x_test=np.expand_dims(x_test,-1)

x_train,x_val=x_train[:5000],x_train[5000:]
y_train,y_val=y_train[:5000],y_train[5000:]
```

[4]:
```
model=keras.Sequential([
    keras.layers.
 ↪Conv2D(32,(3,3),activation='relu',padding='same',input_shape=(28,28,1)),
    keras.layers.MaxPooling2D((2,2)),
    keras.layers.Flatten(),
    keras.layers.Dense(10,activation='softmax')
])
```

```
WARNING:tensorflow:From C:\Users\HP\anaconda3\envs\AML-manyahegde\lib\site-
packages\keras\src\backend.py:873: The name tf.get_default_graph is deprecated.
Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\HP\anaconda3\envs\AML-manyahegde\lib\site-
packages\keras\src\layers\pooling\max_pooling2d.py:161: The name tf.nn.max_pool
```

```
is deprecated. Please use tf.nn.max_pool2d instead.
```

[5]:
```
datagen=ImageDataGenerator(
    rotation_range=10,
    zoom_range=0.1,
    horizontal_flip=True,
    vertical_flip=True,
    shear_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1
)
```

[6]:
```
#without data augmentation
model.
 ↪compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
history1=model.
 ↪fit(x_train,y_train,batch_size=32,epochs=5,validation_data=(x_val,y_val),verbose=0)
test_loss,test_acc=model.evaluate(x_test,y_test)
print(test_loss,test_acc)
```

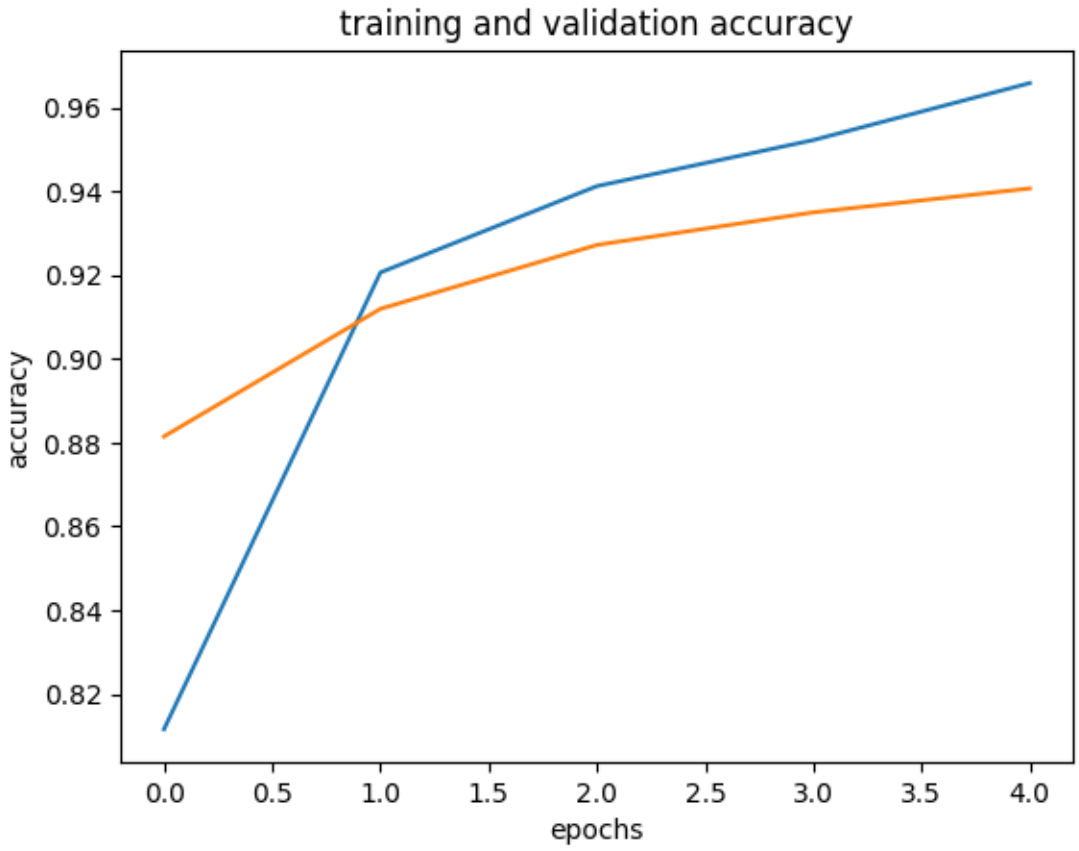
```
WARNING:tensorflow:From C:\Users\HP\anaconda3\envs\AML-manyahegde\lib\site-
packages\keras\src\optimizers\__init__.py:309: The name tf.train.Optimizer is
deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From C:\Users\HP\anaconda3\envs\AML-manyahegde\lib\site-
packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue
is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\HP\anaconda3\envs\AML-manyahegde\lib\site-
packages\keras\src\engine\base_layer_utils.py:384: The name
tf.executing_eagerly_outside_functions is deprecated. Please use
tf.compat.v1.executing_eagerly_outside_functions instead.

313/313 [==============================] - 1s 4ms/step - loss: 0.1842 -
accuracy: 0.9456
0.18418249487876892 0.9455999732017517
```

[7]:
```
#with data augmentation
model.
 ↪compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
history2=model.fit(datagen.
 ↪flow(x_train,y_train,batch_size=32),epochs=5,validation_data=(x_val,y_val),verbose=0)
test_loss,test_acc=model.evaluate(x_test,y_test)
print(test_loss,test_acc)
```

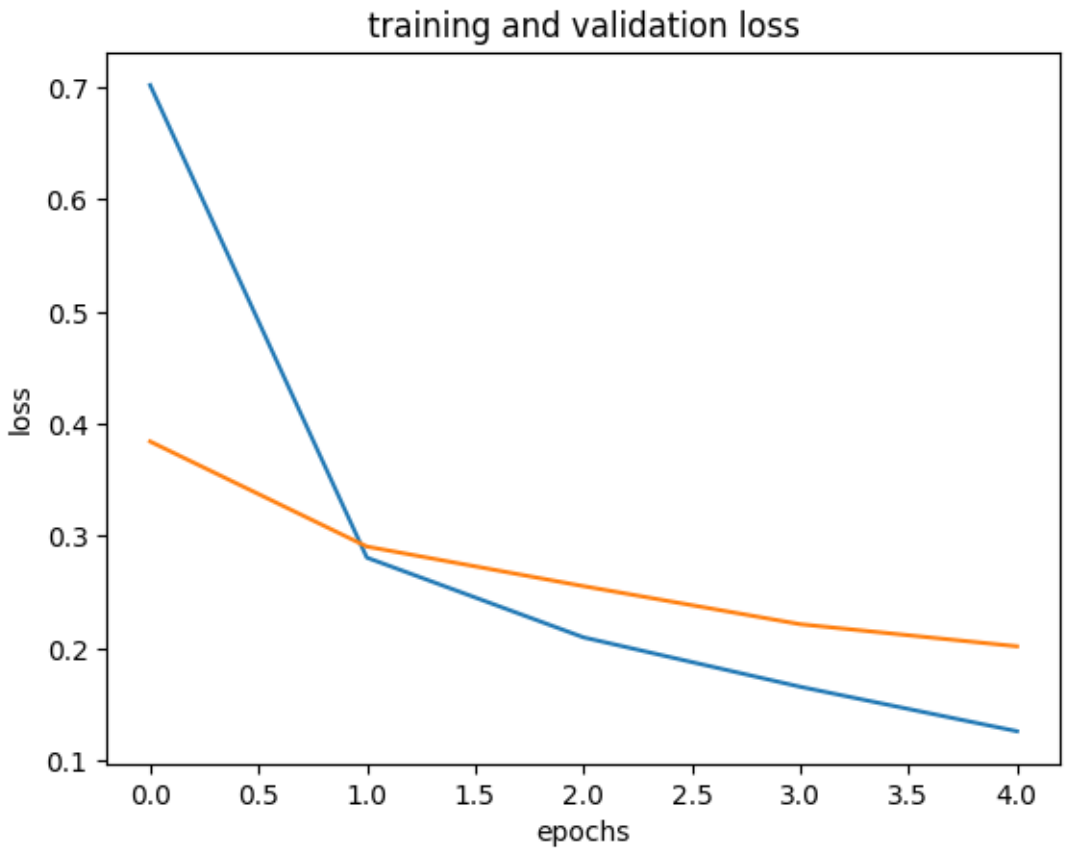
```
313/313 [==============================] - 1s 4ms/step - loss: 0.6357 -
accuracy: 0.7960
```

```
0.6357226967811584 0.7960000038146973
```

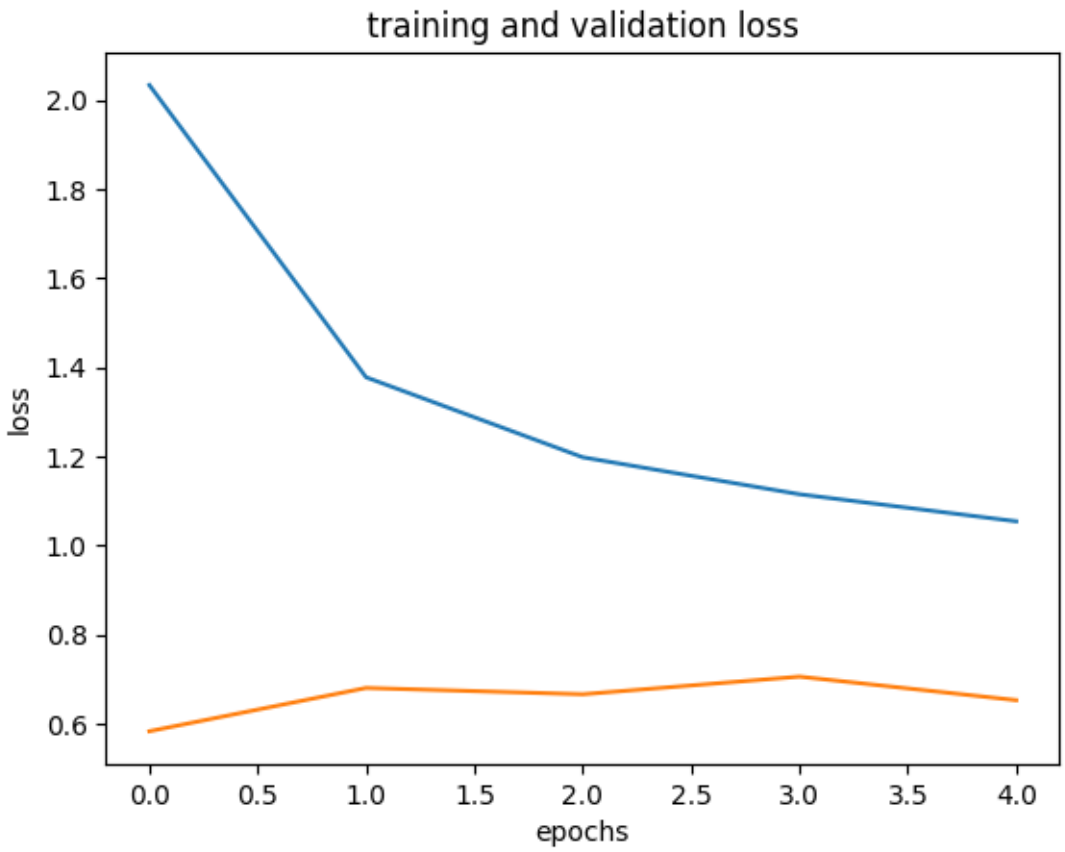
[8]:
```
import matplotlib.pyplot as plt

plt.plot(history1.history['accuracy'],label='training accuracy')
plt.plot(history1.history['val_accuracy'],label='validation accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.title('training and validation accuracy')
plt.show()
```



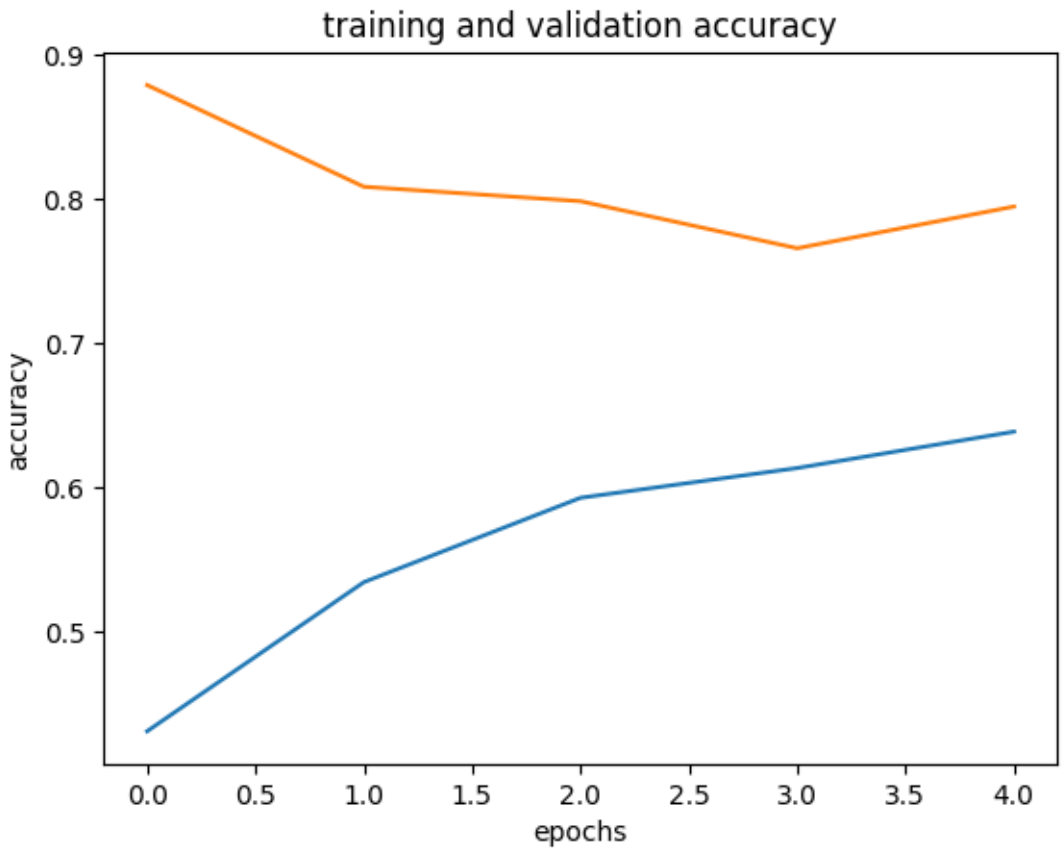
[9]:
```
plt.plot(history1.history['loss'],label='training loss')
plt.plot(history1.history['val_loss'],label='validation loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.title('training and validation loss')
plt.show()
```



[10]:
```
plt.plot(history2.history['loss'],label='training loss')
plt.plot(history2.history['val_loss'],label='validation loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.title('training and validation loss')
plt.show()
```



[11]:
```
plt.plot(history2.history['accuracy'],label='training accuracy')
plt.plot(history2.history['val_accuracy'],label='validation accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.title('training and validation accuracy')
plt.show()
```



Transfer learning model using VGG16, VGG19, RESNET50

[1]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.applications import VGG16,VGG19,ResNet50
from tensorflow.keras.utils import to_categorical
```

[2]:
```
(x_train,y_train),(x_test,y_test)=keras.datasets.cifar10.load_data()
```

[3]:
```
x_train=x_train.astype('float32')/255.0
x_test=x_test.astype('float32')/255.0

y_train=to_categorical(y_train,10)
y_test=to_categorical(y_test,10)
```

[4]:
```
vgg16=VGG16(weights='imagenet',include_top=False,input_shape=(32,32,3))
vgg19=VGG19(weights='imagenet',include_top=False,input_shape=(32,32,3))
```

```
resnet=ResNet50(weights='imagenet',include_top=False,input_shape=(32,32,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [==============================] - 11s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 14s 0us/step
```

[11]:
```
vgg16_output=layers.GlobalAveragePooling2D()(vgg16.output)
vgg16_output=layers.Dense(10,activation='softmax')(vgg16_output)

vgg19_output=layers.GlobalAveragePooling2D()(vgg19.output)
vgg19_output=layers.Dense(10,activation='softmax')(vgg19_output)

resnet_output=layers.GlobalAveragePooling2D()(resnet.output)
resnet_output=layers.Dense(10,activation='softmax')(resnet_output)
```

[13]:
```
vgg16_model=keras.Model(inputs=vgg16.input, outputs=vgg16_output)
vgg19_model=keras.Model(inputs=vgg19.input, outputs=vgg19_output)
resnet_model=keras.Model(inputs=resnet.input, outputs=resnet_output)
```

[14]:
```
vgg16_model.
 ↪compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
vgg19_model.
 ↪compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
resnet_model.
 ↪compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

[15]:
```
vgg16_loss,vgg16_accuracy=vgg16_model.evaluate(x_test,y_test)
vgg19_loss,vgg19_accuracy=vgg19_model.evaluate(x_test,y_test)
resnet_loss,resnet_accuracy=resnet_model.evaluate(x_test,y_test)
```

```
313/313 [==============================] - 29s 90ms/step - loss: 2.5483 -
accuracy: 0.1126
313/313 [==============================] - 43s 137ms/step - loss: 2.7462 -
accuracy: 0.1248
313/313 [==============================] - 24s 70ms/step - loss: 3.4471 -
accuracy: 0.1007
```

[16]:
```
print("VGG16 Test Accuracy:",vgg16_accuracy)
print("VGG19 Test Accuracy:",vgg19_accuracy)
print("ResNet50 Test Accuracy:",resnet_accuracy)
```

```
VGG16 Test Accuracy: 0.11259999871253967
VGG19 Test Accuracy: 0.12479999661445618
ResNet50 Test Accuracy: 0.1006999984383583
```

Sentimental analysis using LSTM

[20]:
```
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from keras.layers import Embedding, LSTM, Dense
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

[21]:
```
df=pd.read_csv('Sentimental Analysis.csv')
df.head()
```

[21]:

	text	label
0	I grew up (b. 1965) watching and loving the Th…	0
1	When I put this movie in my DVD player, and sa…	0
2	Why do people who do not know what a particula…	0
3	Even though I have great interest in Biblical …	0
4	Im a die hard Dads Army fan and nothing will e…	1

[22]:
```
text=df.iloc[:,1]
labels=df.iloc[:,-1]
```

[31]:
```
text = [str(item) for item in text]

tokenizer = Tokenizer()
tokenizer.fit_on_texts(text)
sequences = tokenizer.texts_to_sequences(text)
data = pad_sequences(sequences)
```

[36]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(data,labels,test_size=0.
 ↪2,random_state=32)
```

[38]:
```
model = keras.Sequential([
    Embedding(len(tokenizer.word_index)+1,32,input_length=data.shape[1],),
    LSTM(64),
    Dense(1,activation="sigmoid")
])
```

[39]:
```
y_train = np.asarray(y_train,dtype=float)
y_test = np.asarray(y_test,dtype=float)
```

[40]:
```
model.compile(optimizer="adam",loss="binary_crossentropy",metrics= ["accuracy"])
```

[41]:
```
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=1)
```

```
1000/1000 [==============================] - 8s 5ms/step - loss: 0.0506 -
accuracy: 1.0000 - val_loss: 3.1816e-04 - val_accuracy: 1.0000
```

[41]:
```
<keras.callbacks.History at 0x1fc1d375be0>
```

[42]:
```
new_text = "I had a terrible experience with the product and service"
new_sequence = tokenizer.texts_to_sequences([new_text])
new_data = pad_sequences(new_sequence,maxlen=data.shape[1])
predicted_sentiment = model.predict(new_data)
```

```
1/1 [==============================] - 1s 759ms/step
```

[43]:
```
predicted_sentiment
```

[43]:
```
array([[0.6304072]], dtype=float32)
```

[44]:
```
scores = model.evaluate(x_test,y_test)
print("accuracy:",scores[1])
```

```
250/250 [==============================] - 1s 2ms/step - loss: 3.1816e-04 -
accuracy: 1.0000
accuracy: 1.0
```

Prediction of next word using RNN

[35]:
```
import tensorflow as tf
import numpy as np
```

[36]:
```
text='This is my repo for NNDL lab programs. Here I will be uploading my lab␣
↪programs. You can refer it from here.'
```

[38]:
```
# Preprocess the text and create a vocabulary
tokenizer = tf.keras.layers.TextVectorization()
tokenizer.adapt(text.split())
```

[39]:
```
# Convert text to sequences of token indices
text_sequences = tokenizer(text)
text_sequences
```

[39]:
```
<tf.Tensor: shape=(22,), dtype=int64, numpy=
array([ 9, 14,  3, 10, 17, 12,  4,  2,  5, 15,  7, 19,  8,  3,  4,  2,  6,
       18, 11, 13, 16,  5], dtype=int64)>
```

[40]:
```
# Create training data (X) and target data (y)
X = text_sequences[:-1]
y = text_sequences[1:]
```

[42]:
```
X
```

[42]:
```
<tf.Tensor: shape=(21,), dtype=int64, numpy=
array([ 9, 14,  3, 10, 17, 12,  4,  2,  5, 15,  7, 19,  8,  3,  4,  2,  6,
```

```
       18, 11, 13, 16], dtype=int64)>
```

[43]:
```
y
```

[43]:
```
<tf.Tensor: shape=(21,), dtype=int64, numpy=
array([14,  3, 10, 17, 12,  4,  2,  5, 15,  7, 19,  8,  3,  4,  2,  6, 18,
       11, 13, 16,  5], dtype=int64)>
```

[44]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=len(tokenizer.get_vocabulary()),␣
 ↪output_dim=64, input_length=1),
    tf.keras.layers.SimpleRNN(128, return_sequences=True),
    tf.keras.layers.Dense(len(tokenizer.get_vocabulary()), activation='softmax')
])
```

[45]:
```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam')
```

[46]:
```
model.fit(X, y, epochs=50)
```

```
Epoch 1/50
1/1 [==============================] - 2s 2s/step - loss: 2.9924
Epoch 2/50
1/1 [==============================] - 0s 16ms/step - loss: 2.9772
Epoch 3/50
1/1 [==============================] - 0s 0s/step - loss: 2.9621
Epoch 4/50
1/1 [==============================] - 0s 16ms/step - loss: 2.9469
Epoch 5/50
1/1 [==============================] - 0s 16ms/step - loss: 2.9317
Epoch 6/50
1/1 [==============================] - 0s 7ms/step - loss: 2.9164
Epoch 7/50
1/1 [==============================] - 0s 9ms/step - loss: 2.9009
Epoch 8/50
1/1 [==============================] - 0s 17ms/step - loss: 2.8853
Epoch 9/50
1/1 [==============================] - 0s 9ms/step - loss: 2.8694
Epoch 10/50
1/1 [==============================] - 0s 8ms/step - loss: 2.8533
Epoch 11/50
1/1 [==============================] - 0s 14ms/step - loss: 2.8368
Epoch 12/50
1/1 [==============================] - 0s 10ms/step - loss: 2.8200
Epoch 13/50
1/1 [==============================] - 0s 8ms/step - loss: 2.8028
Epoch 14/50
1/1 [==============================] - 0s 10ms/step - loss: 2.7851
```

```
Epoch 15/50
1/1 [==============================] - 0s 7ms/step - loss: 2.7670
Epoch 16/50
1/1 [==============================] - 0s 11ms/step - loss: 2.7483
Epoch 17/50
1/1 [==============================] - 0s 8ms/step - loss: 2.7291
Epoch 18/50
1/1 [==============================] - 0s 8ms/step - loss: 2.7093
Epoch 19/50
1/1 [==============================] - 0s 7ms/step - loss: 2.6888
Epoch 20/50
1/1 [==============================] - 0s 8ms/step - loss: 2.6677
Epoch 21/50
1/1 [==============================] - 0s 8ms/step - loss: 2.6459
Epoch 22/50
1/1 [==============================] - 0s 6ms/step - loss: 2.6234
Epoch 23/50
1/1 [==============================] - 0s 8ms/step - loss: 2.6001
Epoch 24/50
1/1 [==============================] - 0s 8ms/step - loss: 2.5760
Epoch 25/50
1/1 [==============================] - 0s 11ms/step - loss: 2.5511
Epoch 26/50
1/1 [==============================] - 0s 5ms/step - loss: 2.5254
Epoch 27/50
1/1 [==============================] - 0s 11ms/step - loss: 2.4988
Epoch 28/50
1/1 [==============================] - 0s 8ms/step - loss: 2.4713
Epoch 29/50
1/1 [==============================] - 0s 3ms/step - loss: 2.4429
Epoch 30/50
1/1 [==============================] - 0s 14ms/step - loss: 2.4136
Epoch 31/50
1/1 [==============================] - 0s 5ms/step - loss: 2.3833
Epoch 32/50
1/1 [==============================] - 0s 4ms/step - loss: 2.3521
Epoch 33/50
1/1 [==============================] - 0s 8ms/step - loss: 2.3199
Epoch 34/50
1/1 [==============================] - 0s 7ms/step - loss: 2.2867
Epoch 35/50
1/1 [==============================] - 0s 10ms/step - loss: 2.2525
Epoch 36/50
1/1 [==============================] - 0s 0s/step - loss: 2.2174
Epoch 37/50
1/1 [==============================] - 0s 0s/step - loss: 2.1812
Epoch 38/50
1/1 [==============================] - 0s 16ms/step - loss: 2.1441
```

```
Epoch 39/50
1/1 [==============================] - 0s 16ms/step - loss: 2.1060
Epoch 40/50
1/1 [==============================] - 0s 0s/step - loss: 2.0670
Epoch 41/50
1/1 [==============================] - 0s 19ms/step - loss: 2.0270
Epoch 42/50
1/1 [==============================] - 0s 14ms/step - loss: 1.9861
Epoch 43/50
1/1 [==============================] - 0s 14ms/step - loss: 1.9443
Epoch 44/50
1/1 [==============================] - 0s 0s/step - loss: 1.9016
Epoch 45/50
1/1 [==============================] - 0s 16ms/step - loss: 1.8582
Epoch 46/50
1/1 [==============================] - 0s 17ms/step - loss: 1.8140
Epoch 47/50
1/1 [==============================] - 0s 14ms/step - loss: 1.7690
Epoch 48/50
1/1 [==============================] - 0s 13ms/step - loss: 1.7235
Epoch 49/50
1/1 [==============================] - 0s 8ms/step - loss: 1.6774
Epoch 50/50
1/1 [==============================] - 0s 10ms/step - loss: 1.6307
```

[46]:
```
<keras.callbacks.History at 0x199d07bb190>
```

[47]:
```
def generate_next_word(seed_text):
    seed_sequence = tokenizer(seed_text)
    predicted_probabilities = model.predict(seed_sequence)
    predicted_index = np.argmax(predicted_probabilities)
    predicted_word = tokenizer.get_vocabulary()[predicted_index]
    return predicted_word
```

[48]:
```
# Test the predictive text system
input_text = "from"
predicted_word = generate_next_word(input_text)
print(f"Input: '{input_text}', Predicted: '{predicted_word}'")
```

```
1/1 [==============================] - 0s 275ms/step
Input: 'from', Predicted: 'here'
```

Prediction of next word using LSTM

[49]:
```
import tensorflow as tf
import numpy as np
```

[50]:
```
text='This is my repo for NNDL lab programs. Here I will be uploading my lab␣
↪programs. You can refer it from here.'
```

[51]:
```
# Preprocess the text and create a vocabulary
tokenizer = tf.keras.layers.TextVectorization()
tokenizer.adapt(text.split())
```

[52]:
```
# Convert text to sequences of token indices
text_sequences = tokenizer(text)
```

[53]:
```
# Create training data (X) and target data (y)
X = text_sequences[:-1]
y = text_sequences[1:]
```

[54]:
```
# Build an LSTM model using Keras
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=len(tokenizer.get_vocabulary()),␣
↪output_dim=64, input_length=1),
    tf.keras.layers.LSTM(128, return_sequences=True),
    tf.keras.layers.Dense(len(tokenizer.get_vocabulary()), activation='softmax')
])
```

[55]:
```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam')
```

[56]:
```
# Train the model
model.fit(X, y, epochs=50)
```

```
Epoch 1/50
1/1 [==============================] - 3s 3s/step - loss: 2.9969
Epoch 2/50
1/1 [==============================] - 0s 10ms/step - loss: 2.9937
Epoch 3/50
1/1 [==============================] - 0s 8ms/step - loss: 2.9906
Epoch 4/50
1/1 [==============================] - 0s 5ms/step - loss: 2.9874
Epoch 5/50
1/1 [==============================] - 0s 13ms/step - loss: 2.9843
Epoch 6/50
1/1 [==============================] - 0s 15ms/step - loss: 2.9810
Epoch 7/50
1/1 [==============================] - 0s 17ms/step - loss: 2.9778
Epoch 8/50
1/1 [==============================] - 0s 12ms/step - loss: 2.9745
Epoch 9/50
1/1 [==============================] - 0s 4ms/step - loss: 2.9711
Epoch 10/50
1/1 [==============================] - 0s 19ms/step - loss: 2.9676
```

```
Epoch 11/50
1/1 [==============================] - 0s 12ms/step - loss: 2.9641
Epoch 12/50
1/1 [==============================] - 0s 11ms/step - loss: 2.9604
Epoch 13/50
1/1 [==============================] - 0s 9ms/step - loss: 2.9566
Epoch 14/50
1/1 [==============================] - 0s 7ms/step - loss: 2.9527
Epoch 15/50
1/1 [==============================] - 0s 9ms/step - loss: 2.9486
Epoch 16/50
1/1 [==============================] - 0s 9ms/step - loss: 2.9444
Epoch 17/50
1/1 [==============================] - 0s 8ms/step - loss: 2.9400
Epoch 18/50
1/1 [==============================] - 0s 17ms/step - loss: 2.9354
Epoch 19/50
1/1 [==============================] - 0s 0s/step - loss: 2.9306
Epoch 20/50
1/1 [==============================] - 0s 2ms/step - loss: 2.9257
Epoch 21/50
1/1 [==============================] - 0s 17ms/step - loss: 2.9204
Epoch 22/50
1/1 [==============================] - 0s 13ms/step - loss: 2.9150
Epoch 23/50
1/1 [==============================] - 0s 0s/step - loss: 2.9093
Epoch 24/50
1/1 [==============================] - 0s 4ms/step - loss: 2.9034
Epoch 25/50
1/1 [==============================] - 0s 17ms/step - loss: 2.8971
Epoch 26/50
1/1 [==============================] - 0s 11ms/step - loss: 2.8906
Epoch 27/50
1/1 [==============================] - 0s 8ms/step - loss: 2.8838
Epoch 28/50
1/1 [==============================] - 0s 14ms/step - loss: 2.8766
Epoch 29/50
1/1 [==============================] - 0s 7ms/step - loss: 2.8691
Epoch 30/50
1/1 [==============================] - 0s 8ms/step - loss: 2.8613
Epoch 31/50
1/1 [==============================] - 0s 8ms/step - loss: 2.8531
Epoch 32/50
1/1 [==============================] - 0s 8ms/step - loss: 2.8445
Epoch 33/50
1/1 [==============================] - 0s 3ms/step - loss: 2.8355
Epoch 34/50
1/1 [==============================] - 0s 15ms/step - loss: 2.8261
```

```
Epoch 35/50
1/1 [==============================] - 0s 14ms/step - loss: 2.8162
Epoch 36/50
1/1 [==============================] - 0s 0s/step - loss: 2.8058
Epoch 37/50
1/1 [==============================] - 0s 15ms/step - loss: 2.7950
Epoch 38/50
1/1 [==============================] - 0s 17ms/step - loss: 2.7837
Epoch 39/50
1/1 [==============================] - 0s 12ms/step - loss: 2.7719
Epoch 40/50
1/1 [==============================] - 0s 6ms/step - loss: 2.7595
Epoch 41/50
1/1 [==============================] - 0s 17ms/step - loss: 2.7465
Epoch 42/50
1/1 [==============================] - 0s 9ms/step - loss: 2.7330
Epoch 43/50
1/1 [==============================] - 0s 3ms/step - loss: 2.7188
Epoch 44/50
1/1 [==============================] - 0s 7ms/step - loss: 2.7041
Epoch 45/50
1/1 [==============================] - 0s 9ms/step - loss: 2.6886
Epoch 46/50
1/1 [==============================] - 0s 0s/step - loss: 2.6725
Epoch 47/50
1/1 [==============================] - 0s 17ms/step - loss: 2.6557
Epoch 48/50
1/1 [==============================] - 0s 17ms/step - loss: 2.6382
Epoch 49/50
1/1 [==============================] - 0s 0s/step - loss: 2.6199
Epoch 50/50
1/1 [==============================] - 0s 14ms/step - loss: 2.6009
```

[56]:
```
<keras.callbacks.History at 0x199d12bf670>
```

[57]:
```python
# Function to generate the next word
def generate_next_word(seed_text):
    seed_sequence = tokenizer(seed_text)
    predicted_probabilities = model.predict(seed_sequence)
    predicted_index = np.argmax(predicted_probabilities)
    predicted_word = tokenizer.get_vocabulary()[predicted_index]
    return predicted_word
```

[59]:
```python
# Test the predictive text system
input_text = "can"
predicted_word = generate_next_word(input_text)
print(f"Input: '{input_text}', Predicted: '{predicted_word}'")
```

```
1/1 [==============================] - 0s 32ms/step
Input: 'can', Predicted: 'refer'
```

Explainable AI: LIME model prediction

[15]:
```
import lime
from lime.lime_tabular import LimeTabularExplainer
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
```

[16]:
```
iris=load_iris()
x=iris.data
y=iris.target
```

[17]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

[18]:
```
iris.feature_names
```

[18]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

[19]:
```
model=keras.Sequential([
    keras.layers.Dense(8,input_dim=4,activation='relu'),
    keras.layers.Dense(3,activation='softmax')
])
```

[20]:
```
model.
 ↪compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

[21]:
```
model.fit(x_train,y_train,epochs=50,batch_size=16,verbose=0)
```

[21]:
```
<keras.callbacks.History at 0x199c18f99d0>
```

[22]:
```
explainer=LimeTabularExplainer(x_train, mode='classification')
```

[23]:
```
explanation=explainer.explain_instance(x_test[0],model.predict,num_features=4)
explanation.show_in_notebook()
```

```
157/157 [==============================] - 0s 2ms/step
```

```
<IPython.core.display.HTML object>
```

Explainable AI: SHAP model prediction

[26]:
```
import shap
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
```

[27]:
```
iris=load_iris()
x=iris.data
y=iris.target
```

[28]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

[29]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

[30]:
```
feature_names_encoded=le.fit_transform(iris.feature_names)
feature_names_decoded=le.inverse_transform(feature_names_encoded)
```

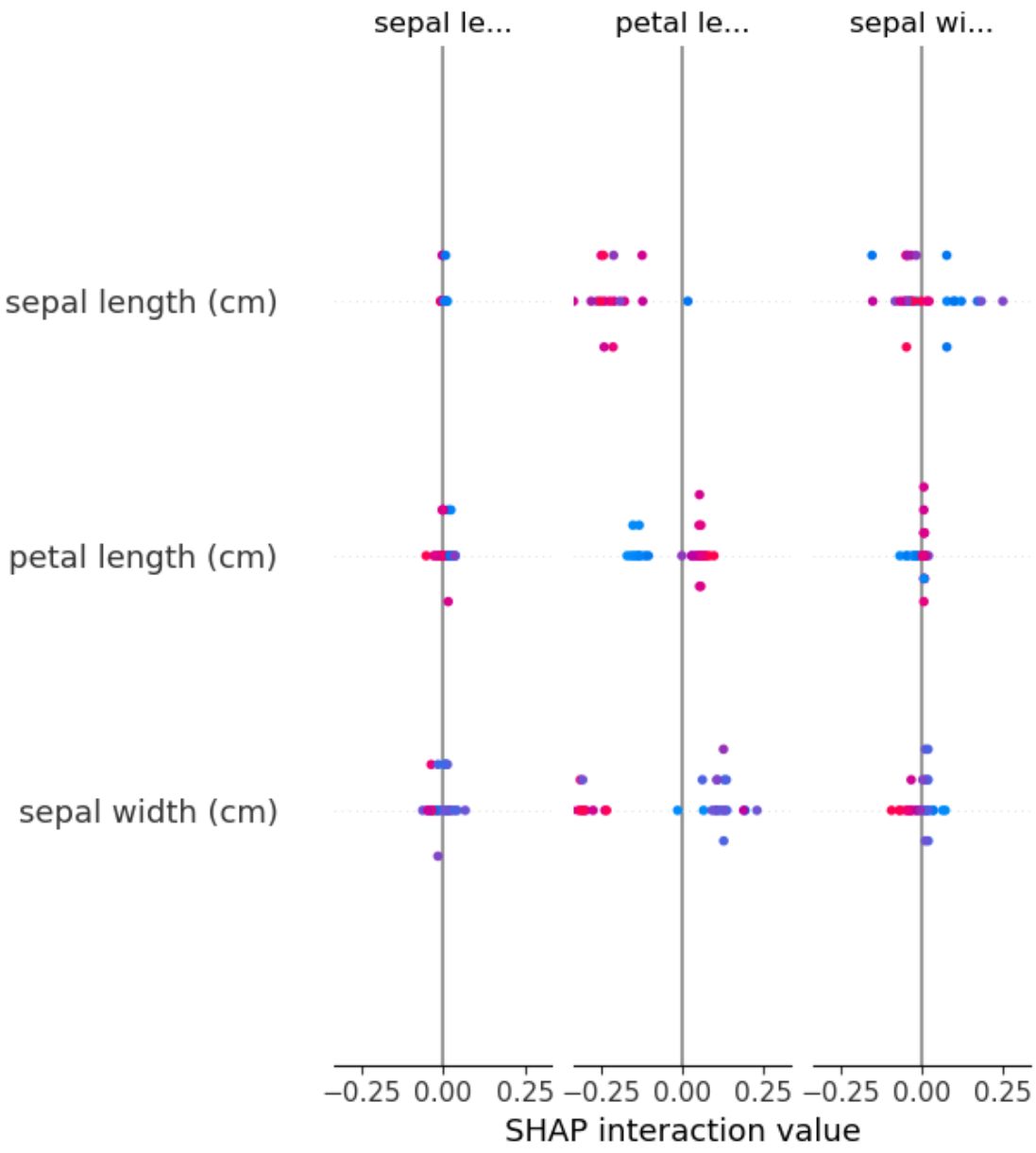
[31]:
```
model=keras.Sequential([
    keras.layers.Dense(8,input_dim=4,activation='relu'),
    keras.layers.Dense(3,activation='softmax')
])
```

[32]:
```
model.
 ↪compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

[33]:
```
model.fit(x_train,y_train,epochs=50,batch_size=16,verbose=0)
```

[33]:
```
<keras.callbacks.History at 0x199ce9a9280>
```

[34]:
```
shap_explainer=shap.Explainer(model,x_train)
shap_values=shap_explainer(x_test)
shap.summary_plot(shap_values,x_test,feature_names=feature_names_decoded)
```


sepal le...
petal le...
sepal wi...
sepal length (cm)
petal length (cm)
sepal width (cm)
−0.25 0.00 0.25
−0.25 0.00 0.25
−0.25 0.00 0.25
SHAP interaction value

[ ]: