```
import tensorflow as tf
from tensorflow.keras import layers,models
import numpy as np
from sklearn.preprocessing import StandardScaler
import pandas as pd

data=pd.read_csv('/content/creditcard.csv')

print(data)

scaler=StandardScaler()
data_scaled=scaler.fit_transform(data.drop('Class',axis=1))

input_dim=data_scaled.shape[1]
input_layer=layers.Input(shape=(input_dim,))
encoded=layers.Dense(64,activation='relu')(input_layer)
encoded=layers.Dense(32,activation='relu')(encoded)

decoded=layers.Dense(64,activation='relu')(encoded)
decoded=layers.Dense(input_dim,activation='sigmoid')(decoded)

autocoder=models.Model(input_layer,decoded)
autocoder.compile(optimizer='adam',loss='mean_squared_error')

autocoder.fit(data_scaled,data_scaled,validation_split=0.2,epochs=50,batch_size=256)

reconstructed=autocoder.predict(data_scaled)

mse=np.mean(np.square(data_scaled - reconstructed),axis=1)

threshold=np.percentile(mse,95)

anamolies=mse > threshold
```

```
            Time         V1         V2        V3  ...       V27       V28  Amount  Class
0            0.0  -1.359807  -0.072781  2.536347  ...  0.133558 -0.021053  149.62      0
1            0.0   1.191857   0.266151  0.166480  ... -0.008983  0.014724    2.69      0
2            1.0  -1.358354  -1.340163  1.773209  ... -0.055353 -0.059752  378.66      0
3            1.0  -0.966272  -0.185226  1.792993  ...  0.062723  0.061458  123.50      0
4            2.0  -1.158233   0.877737  1.548718  ...  0.219422  0.215153   69.99      0
...          ...        ...        ...       ...  ...       ...       ...     ...    ...
284802  172786.0 -11.881118  10.071785 -9.834783  ...  0.943651  0.823731    0.77      0
284803  172787.0  -0.732789  -0.055080  2.035030  ...  0.068472 -0.053527   24.79      0
284804  172788.0   1.919565  -0.301254 -3.249640  ...  0.004455 -0.026561   67.88      0
284805  172788.0  -0.240440   0.530483  0.702510  ...  0.108821  0.104533   10.00      0
284806  172792.0  -0.533413  -0.189733  0.703337  ... -0.002415  0.013649  217.00      0

[284807 rows x 31 columns]
Epoch 1/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - loss: 0.8881 - val_loss: 0.6803
Epoch 2/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.7191 - val_loss: 0.6588
Epoch 3/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.7003 - val_loss: 0.6498
Epoch 4/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - loss: 0.6943 - val_loss: 0.6470
Epoch 5/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - loss: 0.6998 - val_loss: 0.6460
Epoch 6/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.6906 - val_loss: 0.6454
Epoch 7/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.7171 - val_loss: 0.6453
Epoch 8/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.6868 - val_loss: 0.6451
Epoch 9/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - loss: 0.6960 - val_loss: 0.6455
Epoch 10/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 0.6932 - val_loss: 0.6450
Epoch 11/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.6870 - val_loss: 0.6450
Epoch 12/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.6922 - val_loss: 0.6453
Epoch 13/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.6898 - val_loss: 0.6450
Epoch 14/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - loss: 0.6984 - val_loss: 0.6383
Epoch 15/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 0.7034 - val_loss: 0.6336
```

```
Epoch 16/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.6892 - val_loss: 0.6327
Epoch 17/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.7013 - val_loss: 0.6324
Epoch 18/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.6673 - val_loss: 0.6326
Epoch 19/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 0.6740 - val_loss: 0.6326
Epoch 20/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - loss: 0.6950 - val_loss: 0.6325
Epoch 21/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.6896 - val_loss: 0.6326
Epoch 22/50
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import skipgrams
import numpy as np
from tensorflow.keras import layers

# a. Data preparation
text = "Your textual document here"
tokenizer = Tokenizer()
tokenizer.fit_on_texts([text])

word_index = tokenizer.word_index
sequences = tokenizer.texts_to_sequences([text])

# b. Generate training data (CBOW)
pairs, labels = skipgrams(sequences[0], vocabulary_size=len(word_index)+1, window_size=2)

# Convert pairs and labels to NumPy arrays
pairs = np.array(pairs)
labels = np.array(labels)

# c. Define the model
vocab_size = len(word_index) + 1  # Vocabulary size
embedding_dim = 10

# The input is a word, so we use input_length=1
model = tf.keras.Sequential([
    layers.Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=1),
    layers.Flatten(),  # Flatten the embedding output
    layers.Dense(10, activation='relu'),
    layers.Dense(vocab_size, activation='softmax')
])

# Use sparse_categorical_crossentropy to allow integer labels
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy')

# Train the model
model.fit(pairs, labels, epochs=5)

# d. Output predictions
predictions = model.predict(pairs)
```

```
Epoch 1/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - loss: 1.6121
Epoch 2/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 1.6076
Epoch 3/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - loss: 1.6031
Epoch 4/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 1.5985
Epoch 5/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - loss: 1.5942
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step
```

```
from inspect import FullArgSpec
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras import layers,models

base_model=VGG16(weights='imagenet',include_top=False,input_shape=(224,224,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```python
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# a. Load in a pre-trained CNN model
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# b. Freeze parameters (weights) in model’s lower convolutional layers
base_model.trainable = False

# c. Add custom classifier
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(1024, activation='relu'),
    layers.Dense(10, activation='softmax')  # Adjust the number of classes
])

# Set up the data generator for training data
train_data_dir = 'path/to/train_data_directory'  # Update with the path to your training data
train_datagen = ImageDataGenerator(rescale=1.0/255)

# Load training data
train_data = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

# d. Train classifier layers on training data
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(train_data, epochs=5)

# e. Fine-tune hyperparameters and unfreeze more layers if needed
base_model.trainable = True
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(train_data, epochs=5)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-35-106de7867d4e> in <cell line: 25>()
     23 
     24 # Load training data
---> 25 train_data = train_datagen.flow_from_directory(
     26     train_data_dir,
     27     target_size=(224, 224),

  1 frames

/usr/local/lib/python3.10/dist-packages/keras/src/legacy/preprocessing/image.py in __init__(self, directory, image_data_generator, target_size, color_mode, classes, class_mode, batch_size, shuffle, seed, data_format, save_to_dir, save_prefix, save_format, follow_links, subset, interpolation, keep_aspect_ratio, dtype)
    451         if not classes:
    452             classes = []
--> 453             for subdir in sorted(os.listdir(directory)):
    454                 if os.path.isdir(os.path.join(directory, subdir)):
    455                     classes.append(subdir)

FileNotFoundError: [Errno 2] No such file or directory: 'path/to/train_data_directory'
```

```python
import tensorflow as tf
from tensorflow.keras import layers,models
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt

(x_train,y_train),(x_test,y_test)=mnist.load_data()
x_train=x_train/255
x_test=x_test/255

model=models.Sequential([
   layers.Flatten(input_shape=(28,28)),
   layers.Dense(128,activation='relu'),
   layers.Dropout(0.2),
```

```
    layers.Dense(10,activation='softmax')
])

model.compile(optimizer='sgd',loss='sparse_categorical_crossentropy',metrics=['accuracy'])

history=model.fit(x_train,y_train,epochs=11,validation_data=(x_test,y_test))

test_loss,test_acc=model.evaluate(x_test,y_test)
```

```
Epoch 1/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 6ms/step - accuracy: 0.6816 - loss: 1.1174 - val_accuracy: 0.9022 - val_loss: 0.3659
Epoch 2/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 15s 3ms/step - accuracy: 0.8813 - loss: 0.4129 - val_accuracy: 0.9174 - val_loss: 0.2937
Epoch 3/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.9049 - loss: 0.3280 - val_accuracy: 0.9285 - val_loss: 0.2556
Epoch 4/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9153 - loss: 0.2939 - val_accuracy: 0.9356 - val_loss: 0.2303
Epoch 5/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.9255 - loss: 0.2626 - val_accuracy: 0.9420 - val_loss: 0.2086
Epoch 6/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9317 - loss: 0.2412 - val_accuracy: 0.9453 - val_loss: 0.1935
Epoch 7/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - accuracy: 0.9351 - loss: 0.2279 - val_accuracy: 0.9483 - val_loss: 0.1826
Epoch 8/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9411 - loss: 0.2069 - val_accuracy: 0.9512 - val_loss: 0.1702
Epoch 9/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9452 - loss: 0.1958 - val_accuracy: 0.9539 - val_loss: 0.1617
Epoch 10/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9460 - loss: 0.1901 - val_accuracy: 0.9561 - val_loss: 0.1528
Epoch 11/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9486 - loss: 0.1779 - val_accuracy: 0.9574 - val_loss: 0.1463
Epoch 1/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7010 - loss: 1.0778 - val_accuracy: 0.9051 - val_loss: 0.3629
Epoch 2/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8868 - loss: 0.4006 - val_accuracy: 0.9215 - val_loss: 0.2895
Epoch 3/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.9067 - loss: 0.3280 - val_accuracy: 0.9299 - val_loss: 0.2527
Epoch 4/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9176 - loss: 0.2918 - val_accuracy: 0.9350 - val_loss: 0.2267
Epoch 5/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - accuracy: 0.9256 - loss: 0.2624 - val_accuracy: 0.9417 - val_loss: 0.2075
Epoch 6/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - accuracy: 0.9332 - loss: 0.2388 - val_accuracy: 0.9446 - val_loss: 0.1917
Epoch 7/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.9363 - loss: 0.2235 - val_accuracy: 0.9483 - val_loss: 0.1796
Epoch 8/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9411 - loss: 0.2056 - val_accuracy: 0.9509 - val_loss: 0.1687
Epoch 9/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 13s 5ms/step - accuracy: 0.9445 - loss: 0.1983 - val_accuracy: 0.9532 - val_loss: 0.1603
Epoch 10/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9458 - loss: 0.1895 - val_accuracy: 0.9555 - val_loss: 0.1527
Epoch 11/11
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9497 - loss: 0.1778 - val_accuracy: 0.9565 - val_loss: 0.1476
```

```
import tensorflow as tf
from tensorflow.keras import layers,models
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist

(x_train,y_train),(x_test,y_test)=mnist.load_data()
x_train=x_train[...,None]/255
x_test=x_test[...,None]/255
9
model=models.Sequential([
    layers.Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64,activation='relu'),
    layers.Dense(10,activation='softmax')
])

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

history=model.fit(x_train,y_train,epochs=5,validation_data=(x_test,y_test))

test_acc,test_loss=model.evaluate(x_test,y_test)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 69s 36ms/step - accuracy: 0.9054 - loss: 0.3052 - val_accuracy: 0.9821 - val_loss: 0.0538
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 63s 33ms/step - accuracy: 0.9852 - loss: 0.0481 - val_accuracy: 0.9870 - val_loss: 0.0352
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 78s 32ms/step - accuracy: 0.9900 - loss: 0.0314 - val_accuracy: 0.9914 - val_loss: 0.0272
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 59s 32ms/step - accuracy: 0.9937 - loss: 0.0206 - val_accuracy: 0.9918 - val_loss: 0.0243
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 59s 31ms/step - accuracy: 0.9947 - loss: 0.0177 - val_accuracy: 0.9907 - val_loss: 0.0287
```

```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# a. Import the necessary packages
from tensorflow.keras.datasets import mnist

# b. Load the training and testing data
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

# c. Define the network architecture using Keras
model = models.Sequential([
    layers.Flatten(input_shape=(28, 28)),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(10, activation='softmax')
])

# d. Compile the model
model.compile(optimizer='sgd',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# e. Train the model
history = model.fit(x_train, y_train, epochs=11, validation_data=(x_test, y_test))

# f. Evaluate the network
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_acc}')

# Plot the training loss and accuracy
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['loss'], label='Loss')
plt.title('Training Loss and Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Value')
plt.legend()
plt.show()
```

```
import tensorflow as tf
from tensorflow.keras import layers,models
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np

data=pd.read_csv('/content/creditcard.csv')

scaler=StandardScaler()
data_scaled=scaler.fit_transform(data.drop('Class',axis=1))

input_dim=data_scaled.shape[1]
input_layer=layers.Input(shape=(input_dim,))

encoded=layers.Dense(64,activation='relu')(input_layer)
encoded=layers.Dense(32,activation='relu')(encoded)

decoded=layers.Dense(64,activation='relu')(encoded)
decoded=layers.Dense(input_dim,activation='relu')(decoded)
```

```
autocoder=models.Model(input_layer,decoded)
autocoder.compile(optimizer='adam',loss='mean_squared_error')

autocoder.fit(data_scaled,data_scaled,validation_split=0.2,batch_size=256,epochs=50)

reconstructed=autocoder.predict(data_scaled)

mse=np.mean(np.square(reconstructed-data_scaled),axis=1)
threshold=np.percentile(mse,95)
anamolies=mse>threshold
```

```
Epoch 1/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - loss: 0.7637 - val_loss: 0.5362
Epoch 2/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.5586 - val_loss: 0.5004
Epoch 3/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.5544 - val_loss: 0.4930
Epoch 4/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.5457 - val_loss: 0.4907
Epoch 5/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - loss: 0.5410 - val_loss: 0.5073
Epoch 6/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.5443 - val_loss: 0.5042
Epoch 7/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.5497 - val_loss: 0.5058
Epoch 8/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.5513 - val_loss: 0.5035
Epoch 9/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 0.5501 - val_loss: 0.5054
Epoch 10/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - loss: 0.5452 - val_loss: 0.5035
Epoch 11/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 0.5516 - val_loss: 0.5041
Epoch 12/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 0.5451 - val_loss: 0.5034
Epoch 13/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.5480 - val_loss: 0.4903
Epoch 14/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.5456 - val_loss: 0.5044
Epoch 15/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - loss: 0.5414 - val_loss: 0.5038
Epoch 16/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - loss: 0.5342 - val_loss: 0.5034
Epoch 17/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 0.5520 - val_loss: 0.5034
Epoch 18/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - loss: 0.5479 - val_loss: 0.5039
Epoch 19/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - loss: 0.5501 - val_loss: 0.5038
Epoch 20/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - loss: 0.5480 - val_loss: 0.5034
Epoch 21/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.5454 - val_loss: 0.4906
Epoch 22/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 0.5435 - val_loss: 0.4882
Epoch 23/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - loss: 0.5480 - val_loss: 0.4883
Epoch 24/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.5485 - val_loss: 0.4882
Epoch 25/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.5394 - val_loss: 0.4884
Epoch 26/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.5367 - val_loss: 0.4886
Epoch 27/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 0.5461 - val_loss: 0.4890
Epoch 28/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - loss: 0.5430 - val_loss: 0.4883
Epoch 29/50
891/891 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - loss: 0.5455 - val_loss: 0.4882
```

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
tf.__version__

img_generator = tf.keras.preprocessing.image.ImageDataGenerator(#rotation_range=90,
                                                                brightness_range=(0.5,1),
                                                                #shear_range=0.2,
                                                                #zoom_range=0.2,
                                                                channel_shift_range=0.2,
```

```
                                             horizontal_flip=True,
                                             vertical_flip=True,
                                             rescale=1./255,
                                             validation_split=0.3)

root_dir = '101_ObjectCategories'
img_generator_flow_train = img_generator.flow_from_directory(
    directory=root_dir,
    target_size=(224, 224),
    batch_size=32,
    shuffle=True,
    subset="training")

img_generator_flow_valid = img_generator.flow_from_directory(
    directory=root_dir,
    target_size=(224, 224),
    batch_size=32,
    shuffle=True,
    subset="validation")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-17-7ac3efe33d85> in <cell line: 16>()
     14                                              validation_split=0.3)
     15 root_dir = '101_ObjectCategories'
---> 16 img_generator_flow_train = img_generator.flow_from_directory(
     17     directory=root_dir,
     18     target_size=(224, 224),

                                 1 frames
/usr/local/lib/python3.10/dist-packages/keras/src/legacy/preprocessing/image.py in __init__(self, directory, image_data_generator,
target_size, color_mode, classes, class_mode, batch_size, shuffle, seed, data_format, save_to_dir, save_prefix, save_format,
follow_links, subset, interpolation, keep_aspect_ratio, dtype)
    451         if not classes:
    452             classes = []
--> 453             for subdir in sorted(os.listdir(directory)):
    454                 if os.path.isdir(os.path.join(directory, subdir)):
    455                     classes.append(subdir)

FileNotFoundError: [Errno 2] No such file or directory: '101_ObjectCategories'
```

```
from os import waitpid
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import skipgrams
from tensorflow.keras import layers,models
import numpy as np

text="Your Document is here"
tokenizer=Tokenizer()
tokenizer.fit_on_texts([text])
word_index=tokenizer.word_index
sequences=tokenizer.texts_to_sequences([text])

pairs,labels=skipgrams(sequences[0],vocabulary_size=len(word_index)+1,window_size=2)

pairs=np.array(pairs)
labels=np.array(labels)

vocab_size=len(word_index)+1
embedding_dim=10

model=tf.keras.Sequential([
    layers.Embedding(input_dim=vocab_size,output_dim=embedding_dim),
    layers.Flatten(),
    layers.Dense(10,activation='relu'),
    layers.Dense(vocab_size,activation='softmax')
])

model.compile(optimizer='adam',loss='sparse_categorical_crossentropy')

predictions=model.fit(pairs,labels,epochs=5);

model.predict(pairs)
```

```
Epoch 1/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step - loss: 1.6290
Epoch 2/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - loss: 1.6239
```

```
Epoch 3/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 1.6194
Epoch 4/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - loss: 1.6152
Epoch 5/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - loss: 1.6116
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 107ms/step
array([[0.20166315, 0.20049031, 0.19866715, 0.1988709 , 0.20030849],
       [0.20166315, 0.20049031, 0.19866715, 0.1988709 , 0.20030849],
       [0.20124595, 0.2001096 , 0.20187652, 0.19888763, 0.1978803 ],
       [0.20039305, 0.2004738 , 0.2010251 , 0.1990366 , 0.1990715 ],
       [0.19969876, 0.19775316, 0.20183253, 0.20143819, 0.19927734],
       [0.19967465, 0.20048477, 0.20137277, 0.19826165, 0.20020615],
       [0.2041203 , 0.19610062, 0.20568493, 0.19811188, 0.19598228],
       [0.20313333, 0.19869377, 0.2005919 , 0.20059021, 0.19699074],
       [0.19842471, 0.20238212, 0.19592878, 0.20232327, 0.20094106],
       [0.20165712, 0.20060578, 0.20025012, 0.20033959, 0.1971473 ],
       [0.20354581, 0.19725166, 0.20294914, 0.20200284, 0.19425048],
       [0.19969876, 0.19775316, 0.20183253, 0.20143819, 0.19927734],
       [0.2041203 , 0.19610062, 0.20568493, 0.19811188, 0.19598228],
       [0.19989578, 0.20129879, 0.19853218, 0.19922256, 0.20105058],
       [0.20544447, 0.19847797, 0.19982524, 0.19965297, 0.1965993 ],
       [0.1989479 , 0.1969421 , 0.20604545, 0.1982765 , 0.19978786],
       [0.20165712, 0.20060578, 0.20025012, 0.20033959, 0.1971473 ],
       [0.20313333, 0.19869377, 0.2005919 , 0.20059021, 0.19699074],
       [0.19820169, 0.20342936, 0.19372055, 0.2037455 , 0.20090288],
       [0.1998958 , 0.2012988 , 0.1985322 , 0.19922258, 0.2010506 ]],
      dtype=float32)
```