```
pip install pandas numpy scikit-learn tensorflow
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (2.0.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages (2.18.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.6
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/py
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.0.1)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (4.13.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.71.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.8.0)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.13.0)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0->tensorflow) (0
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.15.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.10
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard<2.19,
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorfl
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences


from google.colab import files
uploaded = files.upload()
```

```
Choose Files No file chosen
```

```
import os
from google.colab import drive
drive.mount('/content/drive')
train_directory = '/content/drive/MyDrive/EmotionsDataset_Splitted/data/test'
files = os.listdir(train_directory)
print(files)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
['sad', 'nothing', 'angry', 'happy']
```

```
import chardet

def read_file_safely(file_path):
    with open(file_path, 'rb') as f:
        raw_data = f.read()
        result = chardet.detect(raw_data)
        encoding = result['encoding']
```

```
    with open(file_path, 'r', encoding=encoding, errors='ignore') as f:
        return f.read().strip()

def load_data_from_folder(folder_path):
    texts = []
    labels = []

    for label in os.listdir(folder_path):
        label_folder = os.path.join(folder_path, label)
        if os.path.isdir(label_folder):
            for file in os.listdir(label_folder):
                file_path = os.path.join(label_folder, file)
                try:
                    text = read_file_safely(file_path)
                    texts.append(text)
                    labels.append(label)
                except Exception as e:
                    print(f"Skipping file {file_path} due to error: {e}")

    return texts, labels

test_folder = '/content/drive/MyDrive/EmotionsDataset_Splitted/data/test'
texts, labels = load_data_from_folder(test_folder)
```

```
import pandas as pd
test_df = pd.DataFrame({
    'text': texts,
    'emotion': labels
})
test_df.head()
```

	text	emotion
0	□JFIF□□□□C□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□...	sad
1	□JFIF□□□□C□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□...	sad
2	□JFIF□□□□C□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□...	sad
3	□JFIF□□□□C□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□...	sad
4	□JFIF□□□□C□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□...	sad

Next steps: Generate code with test_df | View recommended plots | New interactive sheet

```
print(test_df.shape)
print(test_df['emotion'].value_counts())
```

```
(788, 2)
emotion
nothing    219
angry      213
happy      207
sad        149
Name: count, dtype: int64
```

```
CONFIG = {
    "BATCH_SIZE": 32,
    "IM_SIZE": 128,
    "LEARNING_RATE": 1e-3,
    "N_EPOCHS": 5,
    "DROPOUT_RATE": 0.3,
    "REGULARIZATION_RATE": 0.001,
    "NUM_CLASSES": 3,
    "CLASS_NAMES": ["angry", "happy","nothing", "sad"]
}
```

```
import tensorflow as tf
train_df = tf.keras.utils.image_dataset_from_directory(
    train_directory,
    labels='inferred',
    label_mode='int',
    class_names=CONFIG["CLASS_NAMES"],
    color_mode='rgb',
    batch_size=CONFIG["BATCH_SIZE"],
    image_size=(CONFIG["IM_SIZE"], CONFIG["IM_SIZE"]),
```

```
    shuffle=True,
    seed=42
)
```

```
Found 788 files belonging to 4 classes.
```

```
from tensorflow.keras.layers import Resizing, Rescaling
resize_rescale_layers = tf.keras.Sequential([
    Resizing(CONFIG['IM_SIZE'], CONFIG['IM_SIZE']),
    Rescaling(1./255)
])
train_df = train_df.map(lambda x, y: (resize_rescale_layers(x), y), num_parallel_calls=tf.data.AUTOTUNE)
train_df = train_df.prefetch(buffer_size=tf.data.AUTOTUNE)
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 12))
for images, labels in train_df.take(1):
    for i in range(16):
        ax = plt.subplot(4, 4, i + 1)
        plt.imshow(images[i].numpy())
        plt.title(CONFIG["CLASS_NAMES"][labels[i].numpy()])
        plt.axis("off")
plt.show()
```


```
plt. figure(figsize = (12,12))
for images, labels in train_df. take (1) :
```

```
for i in range (16):
    ax = plt.subplot(4,4, i+1)
    plt. imshow(images [i].numpy())
    plt.title(CONFIG["CLASS_NAMES"][labels[i]])

    plt.axis ("off")
```


```
for i in train_df. take (1) :
    print(i)
```

```
[[4.70649540e-01, 4.66789246e-01, 4.28676486e-01],
 [2.62132376e-01, 2.96139717e-01, 2.59252459e-01],
 [1.04718141e-01, 1.40870109e-01, 1.30575985e-01],
 ...,
 [6.73897088e-01, 5.38725495e-01, 4.30759817e-01],
 [6.62316203e-01, 5.21139741e-01, 4.19178933e-01],
 [6.45098090e-01, 5.03921628e-01, 4.01960820e-01]],

...,

[[1.67218149e-01, 4.43995118e-01, 5.29840708e-01],
 [1.60477951e-01, 4.19424057e-01, 5.82107902e-01],
 [3.39460820e-02, 1.87806383e-01, 3.14460814e-01],
 ...,
 [2.38357857e-02, 3.19791675e-01, 3.50306392e-01],
 [9.03186351e-02, 3.93995136e-01, 4.32659328e-01],
 [5.58210835e-02, 3.52879912e-01, 4.01899546e-01]],

[[6.98529482e-02, 2.88296580e-01, 3.69852960e-01],
 [2.66544130e-02, 1.95526972e-01, 3.47058833e-01],
 [1.22303925e-01, 3.27205896e-01, 4.30698544e-01],
 ...,
 [5.04289232e-02, 3.71691197e-01, 3.95772070e-01],
 [4.18504924e-02, 3.07781875e-01, 3.37009817e-01],
 [2.57965699e-02, 3.13419133e-01, 3.38848054e-01]],

[[1.40931383e-01, 3.35784346e-01, 4.13235307e-01],
 [7.29779452e-02, 2.35477954e-01, 3.25612754e-01],
 [1.84926480e-01, 4.21629936e-01, 4.80882376e-01],
 ...,
 [6.55024573e-02, 3.43382359e-01, 3.95588249e-01],
 [6.25000000e-02, 3.57536793e-01, 3.96446109e-01],
 [7.24877492e-02, 4.18504924e-01, 4.40686315e-01]]]],
      dtype=float32)>, <tf.Tensor: shape=(32,), dtype=int32, numpy=
array([2, 0, 0, 2, 2, 1, 2, 0, 3, 0, 3, 1, 2, 1, 0, 2, 3, 0, 2, 0, 3, 1,
       3, 2, 3, 1, 1, 2, 3, 0, 2, 2], dtype=int32)>)
```

CNN MODEL

```
import os
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense, Dropout
CONFIG = {
    'IM_SIZE': 128,
    'DROPOUT_RATE': 0.5,
    'NUM_CLASSES': 10,
    'LEARNING_RATE': 0.001,
    'N_EPOCHS': 20

}
def create_cnn_model():
    model = Sequential([
        Input(shape=(CONFIG['IM_SIZE'], CONFIG['IM_SIZE'], 3)),
        Conv2D(32, (3, 3), activation='relu', padding='same'),
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu', padding='same'),
        MaxPooling2D((2, 2)),
        Conv2D(128, (3, 3), activation='relu', padding='same'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(CONFIG['DROPOUT_RATE']),
        Dense(CONFIG['NUM_CLASSES'], activation='softmax')
    ])
    return model
cnn_model = create_cnn_model()
cnn_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=CONFIG['LEARNING_RATE']),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
cnn_model.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 128, 128, 32)	896
max_pooling2d_10 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_11 (Conv2D)	(None, 64, 64, 64)	18,496
max_pooling2d_11 (MaxPooling2D)	(None, 32, 32, 64)	0
conv2d_12 (Conv2D)	(None, 32, 32, 128)	73,856
max_pooling2d_12 (MaxPooling2D)	(None, 16, 16, 128)	0
flatten_2 (Flatten)	(None, 32768)	0
dense_4 (Dense)	(None, 128)	4,194,432
dropout_2 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 10)	1,290

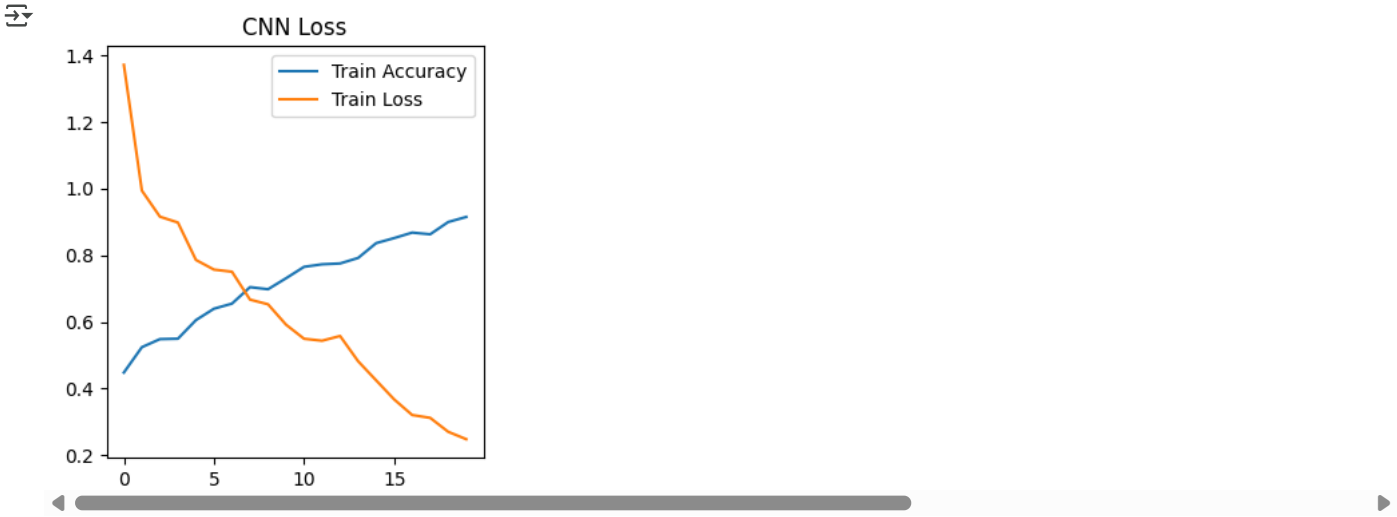
Total params: 4,288,970 (16.36 MB)

```
from tensorflow.keras.losses import SparseCategoricalCrossentropy
metrics = [
    tf.keras.metrics.SparseCategoricalAccuracy(name='accuracy'),
    tf.keras.metrics.SparseTopKCategoricalAccuracy(k=2, name='top_k_accuracy')
]
loss_function = SparseCategoricalCrossentropy()
cnn_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=CONFIG['LEARNING_RATE']),
    loss=loss_function,
    metrics=metrics
)
history_cnn = cnn_model.fit(
    train_df,
    epochs=CONFIG['N_EPOCHS'],
    verbose=1
)
```

```
Epoch 1/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 35s 1s/step - accuracy: 0.3968 - loss: 1.5981 - top_k_accuracy: 0.6352
Epoch 2/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 38s 1s/step - accuracy: 0.5320 - loss: 1.0160 - top_k_accuracy: 0.7999
Epoch 3/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 40s 1s/step - accuracy: 0.5743 - loss: 0.8940 - top_k_accuracy: 0.7796
Epoch 4/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 30s 1s/step - accuracy: 0.5424 - loss: 0.8895 - top_k_accuracy: 0.7993
Epoch 5/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 40s 1s/step - accuracy: 0.6129 - loss: 0.7696 - top_k_accuracy: 0.8425
Epoch 6/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 28s 1s/step - accuracy: 0.6512 - loss: 0.7511 - top_k_accuracy: 0.8580
Epoch 7/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 41s 1s/step - accuracy: 0.6633 - loss: 0.7107 - top_k_accuracy: 0.8791
Epoch 8/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 28s 1s/step - accuracy: 0.7131 - loss: 0.6619 - top_k_accuracy: 0.8981
Epoch 9/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 28s 1s/step - accuracy: 0.6947 - loss: 0.6415 - top_k_accuracy: 0.8858
Epoch 10/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 41s 1s/step - accuracy: 0.7456 - loss: 0.5920 - top_k_accuracy: 0.8972
Epoch 11/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 28s 1s/step - accuracy: 0.7544 - loss: 0.5614 - top_k_accuracy: 0.9377
Epoch 12/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 41s 1s/step - accuracy: 0.7985 - loss: 0.5052 - top_k_accuracy: 0.9367
Epoch 13/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 41s 1s/step - accuracy: 0.7958 - loss: 0.5455 - top_k_accuracy: 0.9312
Epoch 14/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 29s 1s/step - accuracy: 0.7979 - loss: 0.4614 - top_k_accuracy: 0.9476
Epoch 15/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 41s 1s/step - accuracy: 0.8473 - loss: 0.4237 - top_k_accuracy: 0.9541
Epoch 16/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 41s 1s/step - accuracy: 0.8564 - loss: 0.3843 - top_k_accuracy: 0.9430
Epoch 17/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 29s 1s/step - accuracy: 0.8751 - loss: 0.3255 - top_k_accuracy: 0.9730
Epoch 18/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 28s 1s/step - accuracy: 0.8525 - loss: 0.3466 - top_k_accuracy: 0.9677
Epoch 19/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 29s 1s/step - accuracy: 0.8870 - loss: 0.2928 - top_k_accuracy: 0.9792
Epoch 20/20
25/25 ━━━━━━━━━━━━━━━━━━━━ 41s 1s/step - accuracy: 0.9108 - loss: 0.2620 - top_k_accuracy: 0.9820
```

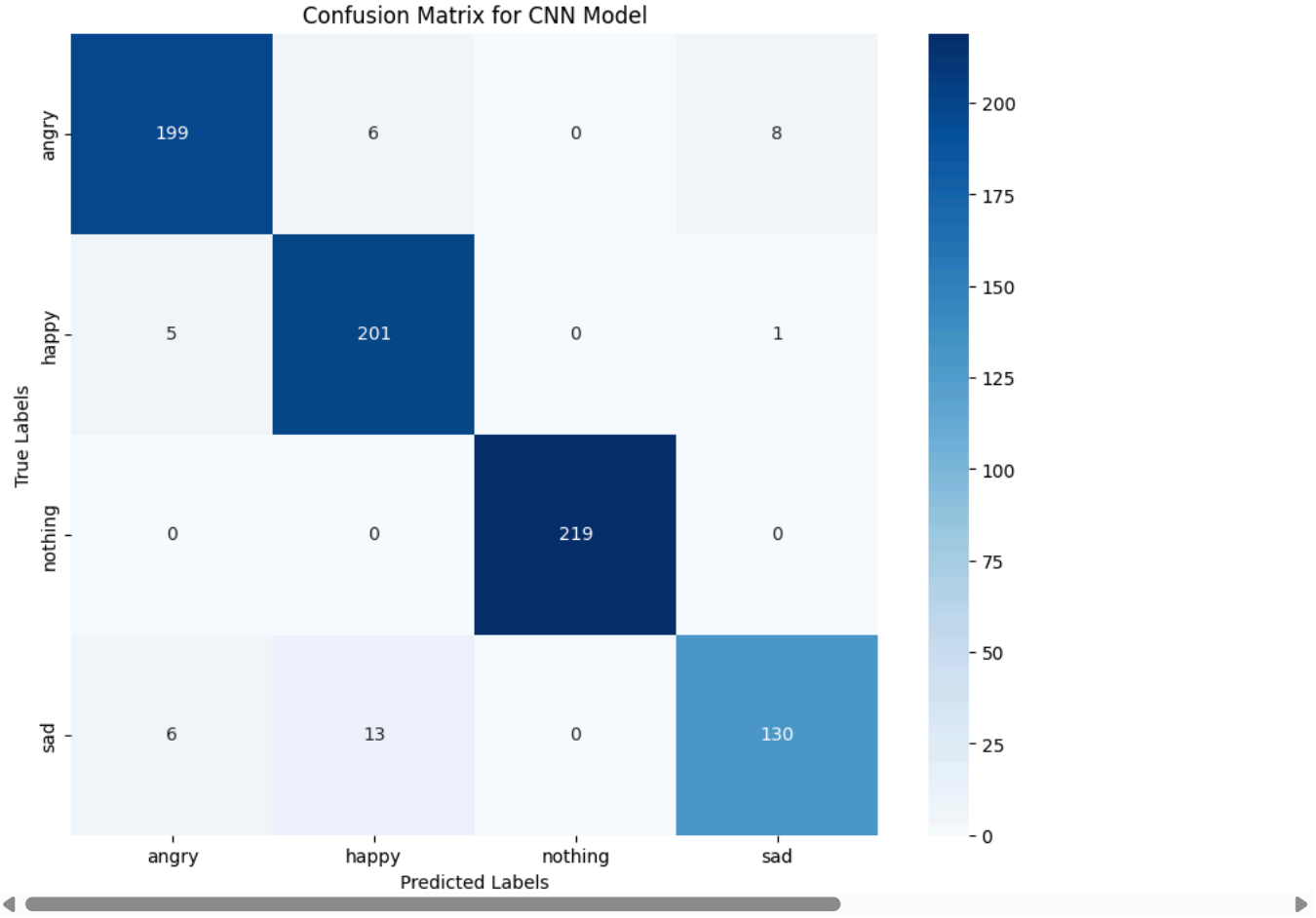
```
plt.figure(figsize=(8, 4))
plt.subplot(1, 2, 1)
```

```
plt.plot(history_cnn.history['accuracy'], label='Train Accuracy')
plt.plot(history_cnn.history['loss'], label='Train Loss')
plt.legend()
plt.title('CNN Accuracy')
plt.title('CNN Loss')

plt.show()
```




```
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import seaborn as sns
import numpy as np
test_dataset = tf.keras.utils.image_dataset_from_directory(
    '/content/drive/MyDrive/EmotionsDataset_Splitted/data/test',
    image_size=(CONFIG['IM_SIZE'], CONFIG['IM_SIZE']),
    batch_size=32
)
CONFIG['CLASS_NAMES'] = test_dataset.class_names
CNN_INPUT_SIZE = cnn_model.input_shape[1:3]
true_labels = []
predicted_labels = []
for images, labels in test_dataset:
    images = images / 255.0
    preds = cnn_model(images, training=False)
    pred_label_idx = tf.argmax(preds, axis=1).numpy()
    true_labels.extend(labels.numpy())
    predicted_labels.extend(pred_label_idx)
true_labels = np.array(true_labels)
predicted_labels = np.array(predicted_labels)
cm = confusion_matrix(true_labels, predicted_labels)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=CONFIG['CLASS_NAMES'], yticklabels=CONFIG['CLASS_NAMES'])
plt.title("Confusion Matrix for CNN Model")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

```
Found 788 files belonging to 4 classes.
```

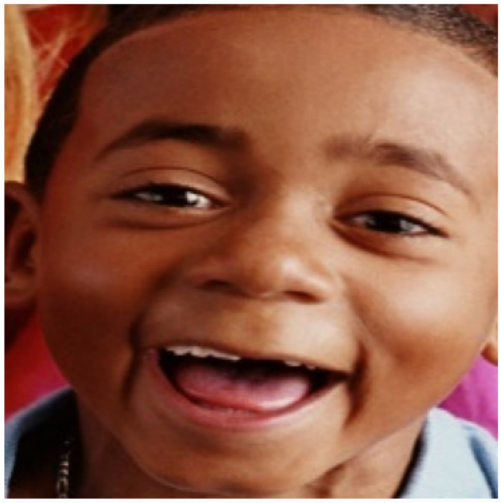



```
cnn_eval_results = cnn_model.evaluate(train_df, verbose=1)
print(f"\nCNN Model Evaluation Results:\nLoss: {cnn_eval_results[0]:.4f}, Accuracy: {cnn_eval_results[1]:.4f}, Top-2 Accuracy: {cnn_eva
```

```
25/25 ━━━━━━━━━━━━━━━━━━━━ 10s 397ms/step - accuracy: 0.9374 - loss: 0.1537 - top_k_accuracy: 0.9939

CNN Model Evaluation Results:
Loss: 0.1359, Accuracy: 0.9505, Top-2 Accuracy: 0.9949
```

```
import cv2
import matplotlib.pyplot as plt

img_path = '/content/drive/MyDrive/EmotionsDataset_Splitted/data/test/sad/sad_140.jpg'
test_image = cv2.imread(img_path)

if test_image is None:
    print("Image not found or failed to load:", img_path)
else:
    test_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB)
    plt.imshow(test_image)
    plt.axis('off')
    plt.show()
```

```
import cv2
import matplotlib.pyplot as plt

img_path = '/content/drive/MyDrive/EmotionsDataset_Splitted/data/test/happy/happy_140.jpg'
test_image = cv2.imread(img_path)

if test_image is None:
    print("Image not found or failed to load:", img_path)
else:
    test_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB)
    plt.imshow(test_image)
    plt.axis('off')
    plt.show()
```



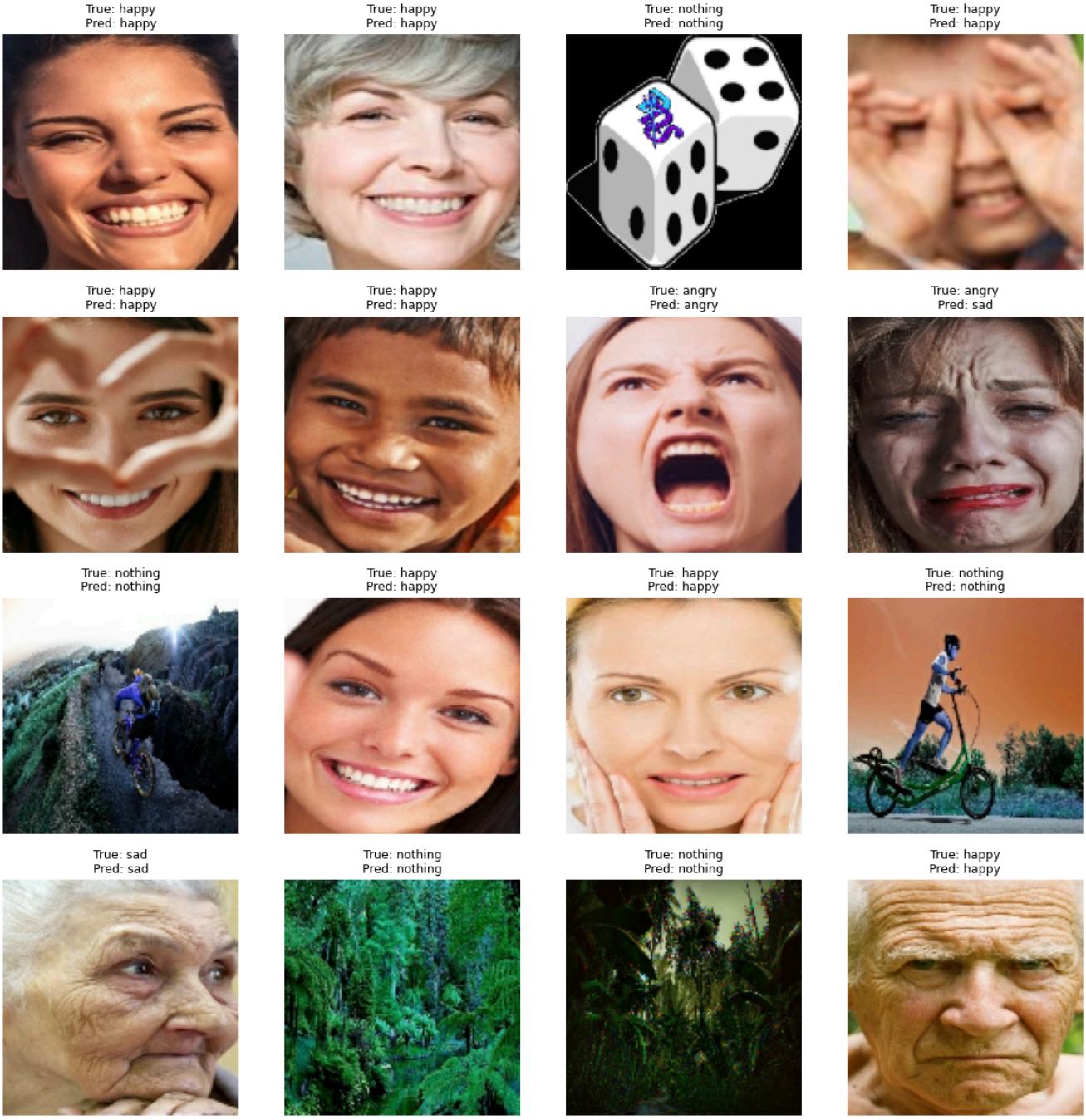
```
import cv2
import matplotlib.pyplot as plt

img_path = '/content/drive/MyDrive/EmotionsDataset_Splitted/data/test/angry/angry_140.jpg'
test_image = cv2.imread(img_path)

if test_image is None:
    print("Image not found or failed to load:", img_path)
else:
    test_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB)
    plt.imshow(test_image)
    plt.axis('off')
    plt.show()
```

```
import cv2
import matplotlib.pyplot as plt

img_path = '/content/drive/MyDrive/EmotionsDataset_Splitted/data/test/nothing/nothing_140.jpg'
test_image = cv2.imread(img_path)

if test_image is None:
    print("Image not found or failed to load:", img_path)
else:
    test_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB)
    plt.imshow(test_image)
    plt.axis('off')
    plt.show()
```

```
CONFIG = {
    'IM_SIZE': 128,
    'DROPOUT_RATE': 0.5,
    'NUM_CLASSES': 3,
    'LEARNING_RATE': 0.001,
    'N_EPOCHS': 10,
    'CLASS_NAMES': ['sad', 'happy', 'angry','nothing']
}



import tensorflow as tf
import matplotlib.pyplot as plt
train_dataset = tf.keras.utils.image_dataset_from_directory(
    '/content/drive/MyDrive/EmotionsDataset_Splitted/data/test',
    image_size=(CONFIG['IM_SIZE'], CONFIG['IM_SIZE']),
    batch_size=32
)

CONFIG['CLASS_NAMES'] = train_dataset.class_names
CNN_INPUT_SIZE = cnn_model.input_shape[1:3]
print("CNN Model expects input shape:", CNN_INPUT_SIZE)
```

```
plt.figure(figsize=(12, 12))
for images, labels in train_dataset.take(1):
    images = images / 255.0
    for i in range(16):
        ax = plt.subplot(4, 4, i + 1)

        img = tf.image.resize(images[i], CNN_INPUT_SIZE)
        img_expanded = tf.expand_dims(img, axis=0)
        preds = cnn_model(img_expanded, training=False)
        pred_label_idx = tf.argmax(preds, axis=1).numpy()[0]
        pred_label = CONFIG["CLASS_NAMES"][pred_label_idx]

        true_label_idx = labels[i].numpy()
        true_label = CONFIG["CLASS_NAMES"][true_label_idx]

        plt.imshow(img)
        plt.title(f"True: {true_label}\nPred: {pred_label}", fontsize=9)
        plt.axis("off")

plt.tight_layout()
plt.show()
```

```
Found 788 files belonging to 4 classes.
CNN Model expects input shape: (128, 128)
```



BERT MODEL

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization, LeakyReLU, MaxPooling2D, Flatten, Dense, Dropout

CONFIG = {
    'IM_SIZE': 128,
    'DROPOUT_RATE': 0.5,
    'NUM_CLASSES': 10,
    'LEARNING_RATE': 0.001,
    'N_EPOCHS': 20
}
```

```
def darknet_block(filters, kernel_size, strides=1, padding='same'):
    return [
        Conv2D(filters, kernel_size, strides=strides, padding=padding, use_bias=False),
        BatchNormalization(),
        LeakyReLU(alpha=0.1)
    ]

def create_darknet_model():
    model = Sequential()
    model.add(Input(shape=(CONFIG['IM_SIZE'], CONFIG['IM_SIZE'], 3)))

    # Darknet-like Conv blocks
    for block in [
        (32, 3),
        (64, 3),
        (128, 3),
        (256, 3),
        (512, 3)
    ]:
        filters, k = block
        for layer in darknet_block(filters, k):
            model.add(layer)
        model.add(MaxPooling2D(pool_size=(2, 2)))

    # Classification head
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(CONFIG['DROPOUT_RATE']))
    model.add(Dense(CONFIG['NUM_CLASSES'], activation='softmax'))

    return model

darknet_model = create_darknet_model()
darknet_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=CONFIG['LEARNING_RATE']),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

darknet_model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/activations/leaky_relu.py:41: UserWarning: Argument `alpha` is deprecated.
  warnings.warn(
```

Model: "sequential_4"

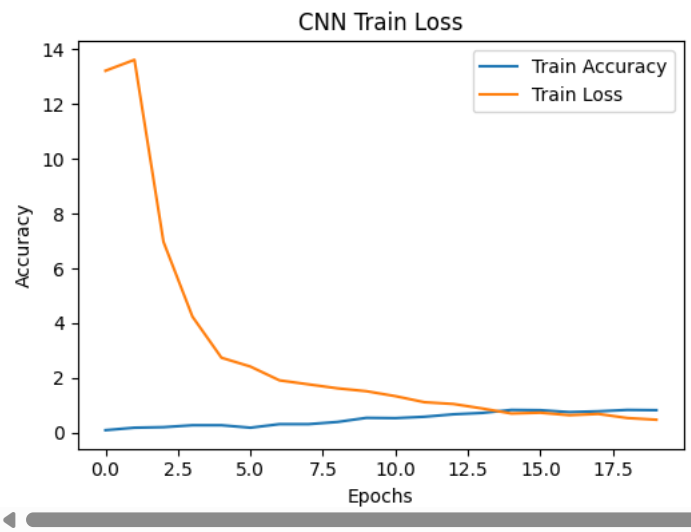
Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 128, 128, 32)	864
batch_normalization_10 (BatchNormalization)	(None, 128, 128, 32)	128
leaky_re_lu_10 (LeakyReLU)	(None, 128, 128, 32)	0
max_pooling2d_13 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_14 (Conv2D)	(None, 64, 64, 64)	18,432
batch_normalization_11 (BatchNormalization)	(None, 64, 64, 64)	256
leaky_re_lu_11 (LeakyReLU)	(None, 64, 64, 64)	0
max_pooling2d_14 (MaxPooling2D)	(None, 32, 32, 64)	0
conv2d_15 (Conv2D)	(None, 32, 32, 128)	73,728
batch_normalization_12 (BatchNormalization)	(None, 32, 32, 128)	512
leaky_re_lu_12 (LeakyReLU)	(None, 32, 32, 128)	0
max_pooling2d_15 (MaxPooling2D)	(None, 16, 16, 128)	0
conv2d_16 (Conv2D)	(None, 16, 16, 256)	294,912
batch_normalization_13 (BatchNormalization)	(None, 16, 16, 256)	1,024
leaky_re_lu_13 (LeakyReLU)	(None, 16, 16, 256)	0
max_pooling2d_16 (MaxPooling2D)	(None, 8, 8, 256)	0
conv2d_17 (Conv2D)	(None, 8, 8, 512)	1,179,648
batch_normalization_14 (BatchNormalization)	(None, 8, 8, 512)	2,048
leaky_re_lu_14 (LeakyReLU)	(None, 8, 8, 512)	0
max_pooling2d_17 (MaxPooling2D)	(None, 4, 4, 512)	0
flatten_3 (Flatten)	(None, 8192)	0
dense_6 (Dense)	(None, 256)	2,097,408
dropout_3 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 10)	2,570

```
import numpy as np

# Example: 100 RGB images of size 128x128 and their labels
X_train = np.random.rand(100, CONFIG['IM_SIZE'], CONFIG['IM_SIZE'], 3).astype(np.float32)
y_train = np.random.randint(0, CONFIG['NUM_CLASSES'], size=(100,))

train_data = tf.data.Dataset.from_tensor_slices((X_train, y_train)).batch(32)



darknet_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=CONFIG['LEARNING_RATE']),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(),
    metrics=[
        tf.keras.metrics.SparseCategoricalAccuracy(name='accuracy'),
        tf.keras.metrics.SparseTopKCategoricalAccuracy(k=2, name='top_k_accuracy')
    ]
)

history_darknet = darknet_model.fit(
    train_data,
    epochs=CONFIG['N_EPOCHS'],
    verbose=1
)
```

```
Epoch 1/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 12s 1s/step - accuracy: 0.0798 - loss: 11.2750 - top_k_accuracy: 0.1614
```

```
Epoch 2/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 8s 2s/step - accuracy: 0.1793 - loss: 14.7179 - top_k_accuracy: 0.2752
Epoch 3/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 6s 1s/step - accuracy: 0.1873 - loss: 7.5774 - top_k_accuracy: 0.3289
Epoch 4/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 8s 2s/step - accuracy: 0.2736 - loss: 4.3722 - top_k_accuracy: 0.3902
Epoch 5/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 6s 1s/step - accuracy: 0.2247 - loss: 3.0244 - top_k_accuracy: 0.3600
Epoch 6/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 8s 2s/step - accuracy: 0.1480 - loss: 2.6373 - top_k_accuracy: 0.2370
Epoch 7/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 7s 2s/step - accuracy: 0.2834 - loss: 1.9917 - top_k_accuracy: 0.4802
Epoch 8/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 8s 2s/step - accuracy: 0.2782 - loss: 1.8180 - top_k_accuracy: 0.5099
Epoch 9/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 7s 2s/step - accuracy: 0.3550 - loss: 1.6828 - top_k_accuracy: 0.5579
Epoch 10/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 8s 2s/step - accuracy: 0.5045 - loss: 1.5674 - top_k_accuracy: 0.6553
Epoch 11/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 6s 1s/step - accuracy: 0.5141 - loss: 1.3654 - top_k_accuracy: 0.6862
Epoch 12/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.5778 - loss: 1.1295 - top_k_accuracy: 0.7877
Epoch 13/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 8s 2s/step - accuracy: 0.6420 - loss: 1.0896 - top_k_accuracy: 0.8205
Epoch 14/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 6s 1s/step - accuracy: 0.7099 - loss: 0.9110 - top_k_accuracy: 0.8832
Epoch 15/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 8s 2s/step - accuracy: 0.8278 - loss: 0.6879 - top_k_accuracy: 0.9355
Epoch 16/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 7s 2s/step - accuracy: 0.8176 - loss: 0.7575 - top_k_accuracy: 0.9138
Epoch 17/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 8s 2s/step - accuracy: 0.7385 - loss: 0.6539 - top_k_accuracy: 0.8956
Epoch 18/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 9s 2s/step - accuracy: 0.7620 - loss: 0.7121 - top_k_accuracy: 0.8306
Epoch 19/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 8s 1s/step - accuracy: 0.8560 - loss: 0.4980 - top_k_accuracy: 0.9449
Epoch 20/20
4/4 ━━━━━━━━━━━━━━━━━━━━ 7s 2s/step - accuracy: 0.8363 - loss: 0.4434 - top_k_accuracy: 0.9291
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
plt.plot(history_darknet.history['accuracy'], label='Train Accuracy')
plt.plot(history_darknet.history['loss'], label='Train Loss')
plt.title('CNN Train Accuracy')
plt.title('CNN Train Loss')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```




```
darknet_eval_results = darknet_model.evaluate(train_data, verbose=1)
print(f"\nDarknet Model Evaluation Results:\nLoss: {darknet_eval_results[0]:.4f}, Accuracy: {darknet_eval_results[1]:.4f}, Top-2 Accurac
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 3s 531ms/step - accuracy: 0.0928 - loss: 2.3114 - top_k_accuracy: 0.3202

Darknet Model Evaluation Results:
Loss: 2.2877, Accuracy: 0.1200, Top-2 Accuracy: 0.3500
```

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img_path = '/content/drive/MyDrive/EmotionsDataset_Splitted/data/test/sad/sad_99.jpg'

test_image = cv2.imread(img_path)

if test_image is None:
    print("Image not found or failed to load:", img_path)
else:
    test_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB)
    test_image_resized = cv2.resize(test_image, (CONFIG['IM_SIZE'], CONFIG['IM_SIZE']))
    test_image_resized = test_image_resized.astype('float32') / 255.0
    test_image_resized = np.expand_dims(test_image_resized, axis=0)
    prediction = darknet_model.predict(test_image_resized)
    predicted_class = np.argmax(prediction, axis=1)
    plt.imshow(test_image)
    plt.axis('off')
    plt.show()
    print(f"Predicted class: {predicted_class[0]}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 206ms/step
```

```
Predicted class: 8
```

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img_path = '/content/drive/MyDrive/EmotionsDataset_Splitted/data/test/angry/angry_99.jpg'

test_image = cv2.imread(img_path)

if test_image is None:
    print("Image not found or failed to load:", img_path)
else:
    test_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB)
    test_image_resized = cv2.resize(test_image, (CONFIG['IM_SIZE'], CONFIG['IM_SIZE']))
    test_image_resized = test_image_resized.astype('float32') / 255.0
    test_image_resized = np.expand_dims(test_image_resized, axis=0)
    prediction = darknet_model.predict(test_image_resized)
    predicted_class = np.argmax(prediction, axis=1)
    plt.imshow(test_image)
    plt.axis('off')
    plt.show()
    print(f"Predicted class: {predicted_class[0]}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step
```

```
Predicted class: 8
```

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img_path = '/content/drive/MyDrive/EmotionsDataset_Splitted/data/test/happy/happy_99.jpg'

test_image = cv2.imread(img_path)

if test_image is None:
    print("Image not found or failed to load:", img_path)
else:
    test_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB)
    test_image_resized = cv2.resize(test_image, (CONFIG['IM_SIZE'], CONFIG['IM_SIZE']))
    test_image_resized = test_image_resized.astype('float32') / 255.0
    test_image_resized = np.expand_dims(test_image_resized, axis=0)
    prediction = darknet_model.predict(test_image_resized)
    predicted_class = np.argmax(prediction, axis=1)
    plt.imshow(test_image)
    plt.axis('off')
    plt.show()
    print(f"Predicted class: {predicted_class[0]}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step
```

```
Predicted class: 8
```

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img_path = '/content/drive/MyDrive/EmotionsDataset_Splitted/data/test/nothing/nothing_99.jpg'

test_image = cv2.imread(img_path)

if test_image is None:
    print("Image not found or failed to load:", img_path)
else:
```

```
    test_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB)
    test_image_resized = cv2.resize(test_image, (CONFIG['IM_SIZE'], CONFIG['IM_SIZE']))
    test_image_resized = test_image_resized.astype('float32') / 255.0
    test_image_resized = np.expand_dims(test_image_resized, axis=0)
    prediction = darknet_model.predict(test_image_resized)
    predicted_class = np.argmax(prediction, axis=1)
    plt.imshow(test_image)
    plt.axis('off')
    plt.show()
    print(f"Predicted class: {predicted_class[0]}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step
```

```
Predicted class: 8
```

```
CONFIG = {
    'IM_SIZE': 128,
    'DROPOUT_RATE': 0.5,
    'NUM_CLASSES': 3,
    'LEARNING_RATE': 0.001,
    'N_EPOCHS': 10,
    'CLASS_NAMES': ['sad', 'happy', 'angry','nothing']
}
```

```
import tensorflow as tf
import matplotlib.pyplot as plt

train_dataset = tf.keras.utils.image_dataset_from_directory(
    '/content/drive/MyDrive/EmotionsDataset_Splitted/data/test',
    image_size=(CONFIG['IM_SIZE'], CONFIG['IM_SIZE']),
    batch_size=32
)

CONFIG['CLASS_NAMES'] = train_dataset.class_names
CNN_INPUT_SIZE = darknet_model.input_shape[1:3]

print("Darknet Model expects input shape:", CNN_INPUT_SIZE)
plt.figure(figsize=(12, 12))
for images, labels in train_dataset.take(1):
    images = images / 255.0
    for i in range(16):
        ax = plt.subplot(4, 4, i + 1)

        img = tf.image.resize(images[i], CNN_INPUT_SIZE)

        img_expanded = tf.expand_dims(img, axis=0)
        preds = cnn_model(img_expanded, training=False)
        pred_label_idx = tf.argmax(preds, axis=1).numpy()[0]
        pred_label = CONFIG["CLASS_NAMES"][pred_label_idx]
        true_label_idx = labels[i].numpy()
        true_label = CONFIG["CLASS_NAMES"][true_label_idx]

        plt.imshow(img)
        plt.title(f"True: {true_label}\nPred: {pred_label}", fontsize=9)
        plt.axis("off")

plt.tight_layout()
plt.show()
```