In [1]:
```
import os
import cv2
import numpy as np

# Path of dataset
dataset_path = r"C:\Users\hemnb\Downloads\imgdata\train"

# Emotion categories
categories = ['angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral']
X = []
y = []

# Load images and labels
for label_index, category in enumerate(categories):
    folder = os.path.join(dataset_path, category)
    for image_name in os.listdir(folder):
        image_path = os.path.join(folder, image_name)
        img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
        if img is not None:
            img = cv2.resize(img, (48, 48))  # resize to fixed size
            X.append(img)
            y.append(label_index)

# Convert to numpy arrays
X = np.array(X).reshape(-1, 48, 48, 1) / 255.0  # Normalize pixel values
y = np.array(y)

print("Total samples loaded:", len(X))
print("Image shape:", X[0].shape)
print("Unique labels:", np.unique(y))
```

```
Total samples loaded: 28709
Image shape: (48, 48, 1)
Unique labels: [0 1 2 3 4 5 6]
```

In [2]:
```
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical

y_encoded = to_categorical(y, num_classes=7)

# train and test data 80:20
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

print("Training samples:", X_train.shape[0])
print("Testing samples:", X_test.shape[0])
```

```
Training samples: 22967
Testing samples: 5742
```

In [3]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, Input

# Build the CNN
model = Sequential([
    Input(shape=(48, 48, 1)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(7, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 46, 46, 32)	320
max_pooling2d (MaxPooling2D)	(None, 23, 23, 32)	0
conv2d_1 (Conv2D)	(None, 21, 21, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 10, 10, 64)	0
flatten (Flatten)	(None, 6400)	0
dense (Dense)	(None, 128)	819,328
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 7)	903

Total params: 839,047 (3.20 MB)

Trainable params: 839,047 (3.20 MB)

Non-trainable params: 0 (0.00 B)

In [4]:
```
# Train the model
history = model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test))
```

```
Epoch 1/10
718/718 ━━━━━━━━━━━━━━━━━━━━ 31s 39ms/step - accuracy: 0.2964 - loss: 1.7482 - val_accuracy: 0.4242 - val_loss:
1.4773
Epoch 2/10
718/718 ━━━━━━━━━━━━━━━━━━━━ 27s 38ms/step - accuracy: 0.4339 - loss: 1.4639 - val_accuracy: 0.4582 - val_loss:
1.3909
Epoch 3/10
718/718 ━━━━━━━━━━━━━━━━━━━━ 27s 38ms/step - accuracy: 0.4788 - loss: 1.3577 - val_accuracy: 0.4859 - val_loss:
1.3314
Epoch 4/10
718/718 ━━━━━━━━━━━━━━━━━━━━ 27s 38ms/step - accuracy: 0.5145 - loss: 1.2831 - val_accuracy: 0.5044 - val_loss:
1.2933
Epoch 5/10
718/718 ━━━━━━━━━━━━━━━━━━━━ 27s 38ms/step - accuracy: 0.5410 - loss: 1.2068 - val_accuracy: 0.5103 - val_loss:
1.2948
Epoch 6/10
718/718 ━━━━━━━━━━━━━━━━━━━━ 27s 37ms/step - accuracy: 0.5666 - loss: 1.1400 - val_accuracy: 0.5195 - val_loss:
1.2646
Epoch 7/10
718/718 ━━━━━━━━━━━━━━━━━━━━ 27s 37ms/step - accuracy: 0.5898 - loss: 1.0752 - val_accuracy: 0.5226 - val_loss:
1.2601
Epoch 8/10
718/718 ━━━━━━━━━━━━━━━━━━━━ 27s 37ms/step - accuracy: 0.6265 - loss: 0.9938 - val_accuracy: 0.5289 - val_loss:
1.2965
Epoch 9/10
718/718 ━━━━━━━━━━━━━━━━━━━━ 41s 38ms/step - accuracy: 0.6499 - loss: 0.9364 - val_accuracy: 0.5284 - val_loss:
1.3010
Epoch 10/10
718/718 ━━━━━━━━━━━━━━━━━━━━ 27s 37ms/step - accuracy: 0.6704 - loss: 0.8736 - val_accuracy: 0.5338 - val_loss:
1.3117
```

In [5]:
```
#save model
model.save("emotion_model.h5")
print("Model saved successfully!")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. T
his file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_m
odel.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
Model saved successfully!
```

In [ ]:
```

```