In [ ]:
```python
# from tensorflow.keras.preprocessing.image import ImageDataGenerator


import tensorflow
```

In [ ]:
```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```python
train_dir = "/Users/ahmedibrahim/Desktop/Mids/Spring23/machine_learning/ML_Project2/Team


test_dir = "/Users/ahmedibrahim/Desktop/Mids/Spring23/machine_learning/ML_Project2/Team-
```

In [ ]:
```python
train_datagen = ImageDataGenerator(
    width_shift_range = 0.1,        # Randomly shift the width of images by up to 10%
    height_shift_range = 0.1,       # Randomly shift the height of images by up to 10%
    horizontal_flip = True,         # Flip images horizontally at random
    rescale = 1./255,               # Rescale pixel values to be between 0 and 1
    validation_split = 0.2          # Set aside 20% of the data for validation
)

validation_datagen = ImageDataGenerator(
    rescale = 1./255,               # Rescale pixel values to be between 0 and 1
    validation_split = 0.2          # Set aside 20% of the data for validation
)
```

In [ ]:
```python
train_generator = train_datagen.flow_from_directory(
    directory = train_dir,           # Directory containing the training data
    target_size = (48, 48),          # Resizes all images to 48x48 pixels
    batch_size = 64,                 # Number of images per batch
    color_mode = "grayscale",        # Converts the images to grayscale
    class_mode = "categorical",      # Classifies the images into 7 categories
    subset = "training"              # Uses the training subset of the data
)

validation_generator = validation_datagen.flow_from_directory(
    directory = test_dir,            # Directory containing the validation data
    target_size = (48, 48),          # Resizes all images to 48x48 pixels
    batch_size = 64,                 # Number of images per batch
    color_mode = "grayscale",        # Converts the images to grayscale
    class_mode = "categorical",      # Classifies the images into 7 categories
    subset = "validation"            # Uses the validation subset of the data
)
```

```
Found 22968 images belonging to 7 classes.
Found 1432 images belonging to 7 classes.
```

- CNN Model

In [ ]:
```python
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization
import tensorflow as tf

# Define the model architecture
model = Sequential()
```

```python
# Add a convolutional layer with 32 filters, 3x3 kernel size, and relu activation functi
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
# Add a batch normalization layer
model.add(BatchNormalization())
# Add a second convolutional layer with 64 filters, 3x3 kernel size, and relu activation
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
# Add a second batch normalization layer
model.add(BatchNormalization())
# Add a max pooling layer with 2x2 pool size
model.add(MaxPooling2D(pool_size=(2, 2)))
# Add a dropout layer with 0.25 dropout rate
model.add(Dropout(0.25))

# Add a third convolutional layer with 128 filters, 3x3 kernel size, and relu activation
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
# Add a third batch normalization layer
model.add(BatchNormalization())
# Add a fourth convolutional layer with 128 filters, 3x3 kernel size, and relu activatio
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
# Add a fourth batch normalization layer
model.add(BatchNormalization())
# Add a max pooling layer with 2x2 pool size
model.add(MaxPooling2D(pool_size=(2, 2)))
# Add a dropout layer with 0.25 dropout rate
model.add(Dropout(0.25))

# Add a fifth convolutional layer with 256 filters, 3x3 kernel size, and relu activation
model.add(Conv2D(256, kernel_size=(3, 3), activation='relu'))
# Add a fifth batch normalization layer
model.add(BatchNormalization())
# Add a sixth convolutional layer with 256 filters, 3x3 kernel size, and relu activation
model.add(Conv2D(256, kernel_size=(3, 3), activation='relu'))
# Add a sixth batch normalization layer
model.add(BatchNormalization())
# Add a max pooling layer with 2x2 pool size
model.add(MaxPooling2D(pool_size=(2, 2)))
# Add a dropout layer with 0.25 dropout rate
model.add(Dropout(0.25))

# Flatten the output of the convolutional layers
model.add(Flatten())
# Add a dense layer with 256 neurons and relu activation function
model.add(Dense(256, activation='relu'))
# Add a seventh batch normalization layer
model.add(BatchNormalization())
# Add a dropout layer with 0.5 dropout rate
model.add(Dropout(0.5))
# Add a dense layer with 7 neurons (one for each class) and softmax activation function
model.add(Dense(7, activation='softmax'))

# Compile the model with categorical cross-entropy loss, adam optimizer, and accuracy me
model.compile(loss="categorical_crossentropy", optimizer= tf.keras.optimizers.Adam(lr=0.
```

In [ ]:

```python
from tensorflow.keras.callbacks import ModelCheckpoint

# Define the callback
checkpoint_callback = ModelCheckpoint(
    filepath='model_weights.h5',
```

```
    monitor='val_accuracy',
    save_best_only=True,
    save_weights_only=True,
    mode='max',
    verbose=1
)

# Train the model with the callback
history = model.fit(
    train_generator,
    steps_per_epoch=len(train_generator),
    epochs=50,
    validation_data=validation_generator,
    validation_steps=len(validation_generator),
    callbacks=[checkpoint_callback]
)
```

```
Epoch 1/50
359/359 [==============================] - 95s 262ms/step - loss: 2.7694 - accuracy: 0.1
752 - val_loss: 1.9975 - val_accuracy: 0.2137

Epoch 00001: val_accuracy improved from -inf to 0.21369, saving model to model_weights.h
5
Epoch 2/50
359/359 [==============================] - 93s 258ms/step - loss: 2.3867 - accuracy: 0.2
144 - val_loss: 1.7622 - val_accuracy: 0.3198

Epoch 00002: val_accuracy improved from 0.21369 to 0.31983, saving model to model_weight
s.h5
Epoch 3/50
359/359 [==============================] - 89s 249ms/step - loss: 2.2011 - accuracy: 0.2
485 - val_loss: 1.6647 - val_accuracy: 0.3492

Epoch 00003: val_accuracy improved from 0.31983 to 0.34916, saving model to model_weight
s.h5
Epoch 4/50
359/359 [==============================] - 91s 254ms/step - loss: 2.0667 - accuracy: 0.2
827 - val_loss: 1.6458 - val_accuracy: 0.3764

Epoch 00004: val_accuracy improved from 0.34916 to 0.37640, saving model to model_weight
s.h5
Epoch 5/50
359/359 [==============================] - 90s 249ms/step - loss: 1.9615 - accuracy: 0.3
082 - val_loss: 1.6313 - val_accuracy: 0.3771

Epoch 00005: val_accuracy improved from 0.37640 to 0.37709, saving model to model_weight
s.h5
Epoch 6/50
359/359 [==============================] - 1678s 5s/step - loss: 1.8904 - accuracy: 0.32
61 - val_loss: 1.5858 - val_accuracy: 0.3911

Epoch 00006: val_accuracy improved from 0.37709 to 0.39106, saving model to model_weight
s.h5
Epoch 7/50
359/359 [==============================] - 90s 250ms/step - loss: 1.8191 - accuracy: 0.3
454 - val_loss: 1.5468 - val_accuracy: 0.4148

Epoch 00007: val_accuracy improved from 0.39106 to 0.41480, saving model to model_weight
s.h5
Epoch 8/50
359/359 [==============================] - 162s 451ms/step - loss: 1.7506 - accuracy: 0.
3601 - val_loss: 1.4238 - val_accuracy: 0.4658

Epoch 00008: val_accuracy improved from 0.41480 to 0.46578, saving model to model_weight
s.h5
Epoch 9/50
359/359 [==============================] - 92s 257ms/step - loss: 1.6813 - accuracy: 0.3
806 - val_loss: 1.4007 - val_accuracy: 0.4811

Epoch 00009: val_accuracy improved from 0.46578 to 0.48115, saving model to model_weight
s.h5
Epoch 10/50
359/359 [==============================] - 90s 249ms/step - loss: 1.6287 - accuracy: 0.4
000 - val_loss: 1.3988 - val_accuracy: 0.4818

Epoch 00010: val_accuracy improved from 0.48115 to 0.48184, saving model to model_weight
```

```
s.h5
Epoch 11/50
359/359 [==============================] - 89s 249ms/step - loss: 1.5731 - accuracy: 0.4
128 - val_loss: 1.3241 - val_accuracy: 0.5007

Epoch 00011: val_accuracy improved from 0.48184 to 0.50070, saving model to model_weight
s.h5
Epoch 12/50
359/359 [==============================] - 91s 254ms/step - loss: 1.5213 - accuracy: 0.4
333 - val_loss: 1.3156 - val_accuracy: 0.5084

Epoch 00012: val_accuracy improved from 0.50070 to 0.50838, saving model to model_weight
s.h5
Epoch 13/50
359/359 [==============================] - 89s 248ms/step - loss: 1.4785 - accuracy: 0.4
473 - val_loss: 1.2769 - val_accuracy: 0.5091

Epoch 00013: val_accuracy improved from 0.50838 to 0.50908, saving model to model_weight
s.h5
Epoch 14/50
359/359 [==============================] - 90s 250ms/step - loss: 1.4403 - accuracy: 0.4
558 - val_loss: 1.2343 - val_accuracy: 0.5307

Epoch 00014: val_accuracy improved from 0.50908 to 0.53073, saving model to model_weight
s.h5
Epoch 15/50
359/359 [==============================] - 90s 252ms/step - loss: 1.3980 - accuracy: 0.4
663 - val_loss: 1.2366 - val_accuracy: 0.5314

Epoch 00015: val_accuracy improved from 0.53073 to 0.53142, saving model to model_weight
s.h5
Epoch 16/50
359/359 [==============================] - 89s 249ms/step - loss: 1.3721 - accuracy: 0.4
825 - val_loss: 1.2175 - val_accuracy: 0.5384

Epoch 00016: val_accuracy improved from 0.53142 to 0.53841, saving model to model_weight
s.h5
Epoch 17/50
359/359 [==============================] - 178s 497ms/step - loss: 1.3365 - accuracy: 0.
4902 - val_loss: 1.2064 - val_accuracy: 0.5489

Epoch 00017: val_accuracy improved from 0.53841 to 0.54888, saving model to model_weight
s.h5
Epoch 18/50
359/359 [==============================] - 96s 267ms/step - loss: 1.3237 - accuracy: 0.4
999 - val_loss: 1.2082 - val_accuracy: 0.5447

Epoch 00018: val_accuracy did not improve from 0.54888
Epoch 19/50
359/359 [==============================] - 93s 259ms/step - loss: 1.2972 - accuracy: 0.5
098 - val_loss: 1.1696 - val_accuracy: 0.5670

Epoch 00019: val_accuracy improved from 0.54888 to 0.56704, saving model to model_weight
s.h5
Epoch 20/50
359/359 [==============================] - 95s 263ms/step - loss: 1.2630 - accuracy: 0.5
221 - val_loss: 1.1577 - val_accuracy: 0.5733

Epoch 00020: val_accuracy improved from 0.56704 to 0.57332, saving model to model_weight
```

```
s.h5
Epoch 21/50
359/359 [==============================] - 94s 262ms/step - loss: 1.2521 - accuracy: 0.5
261 - val_loss: 1.1225 - val_accuracy: 0.5719

Epoch 00021: val_accuracy did not improve from 0.57332
Epoch 22/50
359/359 [==============================] - 95s 265ms/step - loss: 1.2312 - accuracy: 0.5
353 - val_loss: 1.0924 - val_accuracy: 0.5803

Epoch 00022: val_accuracy improved from 0.57332 to 0.58031, saving model to model_weight
s.h5
Epoch 23/50
359/359 [==============================] - 90s 249ms/step - loss: 1.2243 - accuracy: 0.5
428 - val_loss: 1.1084 - val_accuracy: 0.5691

Epoch 00023: val_accuracy did not improve from 0.58031
Epoch 24/50
359/359 [==============================] - 98s 273ms/step - loss: 1.1988 - accuracy: 0.5
522 - val_loss: 1.1025 - val_accuracy: 0.5761

Epoch 00024: val_accuracy did not improve from 0.58031
Epoch 25/50
359/359 [==============================] - 95s 264ms/step - loss: 1.1903 - accuracy: 0.5
544 - val_loss: 1.0872 - val_accuracy: 0.5810

Epoch 00025: val_accuracy improved from 0.58031 to 0.58101, saving model to model_weight
s.h5
Epoch 26/50
359/359 [==============================] - 95s 264ms/step - loss: 1.1751 - accuracy: 0.5
585 - val_loss: 1.0948 - val_accuracy: 0.5915

Epoch 00026: val_accuracy improved from 0.58101 to 0.59148, saving model to model_weight
s.h5
Epoch 27/50
359/359 [==============================] - 98s 272ms/step - loss: 1.1612 - accuracy: 0.5
600 - val_loss: 1.0846 - val_accuracy: 0.5929

Epoch 00027: val_accuracy improved from 0.59148 to 0.59288, saving model to model_weight
s.h5
Epoch 28/50
359/359 [==============================] - 98s 274ms/step - loss: 1.1535 - accuracy: 0.5
683 - val_loss: 1.0596 - val_accuracy: 0.5957

Epoch 00028: val_accuracy improved from 0.59288 to 0.59567, saving model to model_weight
s.h5
Epoch 29/50
359/359 [==============================] - 98s 274ms/step - loss: 1.1381 - accuracy: 0.5
717 - val_loss: 1.0546 - val_accuracy: 0.5999

Epoch 00029: val_accuracy improved from 0.59567 to 0.59986, saving model to model_weight
s.h5
Epoch 30/50
359/359 [==============================] - 98s 273ms/step - loss: 1.1257 - accuracy: 0.5
769 - val_loss: 1.0357 - val_accuracy: 0.6131

Epoch 00030: val_accuracy improved from 0.59986 to 0.61313, saving model to model_weight
s.h5
Epoch 31/50
```

```
359/359 [==============================] - 96s 268ms/step - loss: 1.1138 - accuracy: 0.5
839 - val_loss: 1.0227 - val_accuracy: 0.6117

Epoch 00031: val_accuracy did not improve from 0.61313
Epoch 32/50
359/359 [==============================] - 94s 263ms/step - loss: 1.1116 - accuracy: 0.5
770 - val_loss: 1.0454 - val_accuracy: 0.5929

Epoch 00032: val_accuracy did not improve from 0.61313
Epoch 33/50
359/359 [==============================] - 100s 278ms/step - loss: 1.0961 - accuracy: 0.
5893 - val_loss: 1.0290 - val_accuracy: 0.6027

Epoch 00033: val_accuracy did not improve from 0.61313
Epoch 34/50
359/359 [==============================] - 105s 291ms/step - loss: 1.0896 - accuracy: 0.
5906 - val_loss: 1.0092 - val_accuracy: 0.6250

Epoch 00034: val_accuracy improved from 0.61313 to 0.62500, saving model to model_weight
s.h5
Epoch 35/50
359/359 [==============================] - 104s 290ms/step - loss: 1.0731 - accuracy: 0.
5967 - val_loss: 1.0312 - val_accuracy: 0.6117

Epoch 00035: val_accuracy did not improve from 0.62500
Epoch 36/50
359/359 [==============================] - 103s 288ms/step - loss: 1.0714 - accuracy: 0.
5987 - val_loss: 1.0002 - val_accuracy: 0.6271

Epoch 00036: val_accuracy improved from 0.62500 to 0.62709, saving model to model_weight
s.h5
Epoch 37/50
359/359 [==============================] - 102s 284ms/step - loss: 1.0607 - accuracy: 0.
6002 - val_loss: 1.0162 - val_accuracy: 0.6222

Epoch 00037: val_accuracy did not improve from 0.62709
Epoch 38/50
359/359 [==============================] - 93s 260ms/step - loss: 1.0582 - accuracy: 0.6
034 - val_loss: 0.9728 - val_accuracy: 0.6397

Epoch 00038: val_accuracy improved from 0.62709 to 0.63966, saving model to model_weight
s.h5
Epoch 39/50
359/359 [==============================] - 95s 266ms/step - loss: 1.0435 - accuracy: 0.6
081 - val_loss: 1.0038 - val_accuracy: 0.6180

Epoch 00039: val_accuracy did not improve from 0.63966
Epoch 40/50
359/359 [==============================] - 98s 273ms/step - loss: 1.0433 - accuracy: 0.6
119 - val_loss: 0.9848 - val_accuracy: 0.6397

Epoch 00040: val_accuracy did not improve from 0.63966
Epoch 41/50
359/359 [==============================] - 97s 271ms/step - loss: 1.0334 - accuracy: 0.6
107 - val_loss: 0.9740 - val_accuracy: 0.6334

Epoch 00041: val_accuracy did not improve from 0.63966
Epoch 42/50
359/359 [==============================] - 99s 276ms/step - loss: 1.0289 - accuracy: 0.6
```

```
143 - val_loss: 0.9968 - val_accuracy: 0.6320

Epoch 00042: val_accuracy did not improve from 0.63966
Epoch 43/50
359/359 [==============================] - 105s 293ms/step - loss: 1.0137 - accuracy: 0.
6180 - val_loss: 1.0003 - val_accuracy: 0.6243

Epoch 00043: val_accuracy did not improve from 0.63966
Epoch 44/50
359/359 [==============================] - 106s 295ms/step - loss: 1.0148 - accuracy: 0.
6208 - val_loss: 0.9996 - val_accuracy: 0.6285

Epoch 00044: val_accuracy did not improve from 0.63966
Epoch 45/50
359/359 [==============================] - 103s 286ms/step - loss: 0.9994 - accuracy: 0.
6226 - val_loss: 0.9740 - val_accuracy: 0.6390

Epoch 00045: val_accuracy did not improve from 0.63966
Epoch 46/50
359/359 [==============================] - 90s 251ms/step - loss: 0.9986 - accuracy: 0.6
252 - val_loss: 0.9688 - val_accuracy: 0.6425

Epoch 00046: val_accuracy improved from 0.63966 to 0.64246, saving model to model_weight
s.h5
Epoch 47/50
359/359 [==============================] - 101s 281ms/step - loss: 0.9966 - accuracy: 0.
6253 - val_loss: 0.9570 - val_accuracy: 0.6473

Epoch 00047: val_accuracy improved from 0.64246 to 0.64735, saving model to model_weight
s.h5
Epoch 48/50
359/359 [==============================] - 90s 252ms/step - loss: 0.9852 - accuracy: 0.6
294 - val_loss: 0.9759 - val_accuracy: 0.6341

Epoch 00048: val_accuracy did not improve from 0.64735
Epoch 49/50
359/359 [==============================] - 161s 448ms/step - loss: 0.9752 - accuracy: 0.
6358 - val_loss: 0.9463 - val_accuracy: 0.6446

Epoch 00049: val_accuracy did not improve from 0.64735
Epoch 50/50
359/359 [==============================] - 90s 250ms/step - loss: 0.9751 - accuracy: 0.6
351 - val_loss: 0.9499 - val_accuracy: 0.6432

Epoch 00050: val_accuracy did not improve from 0.64735
```

Evaluation

In [ ]:

```
# save model

model.save('/Users/ahmedibrahim/Desktop/Mids/Spring23/machine_learning/ML_Project2/Team-

model.save_weights('/Users/ahmedibrahim/Desktop/Mids/Spring23/machine_learning/ML_Projec

# save the model to disk
```

In [ ]:

```
loaded_model = tf.keras.models.load_model('model.h5')
```

```
loaded_model.load_weights('model_weights.h5')

# Evaluate the model on the test data using `evaluate`

print("Evaluate on test data")

results = loaded_model.evaluate(validation_generator, batch_size=128)

print("test loss, test acc:", results)
```

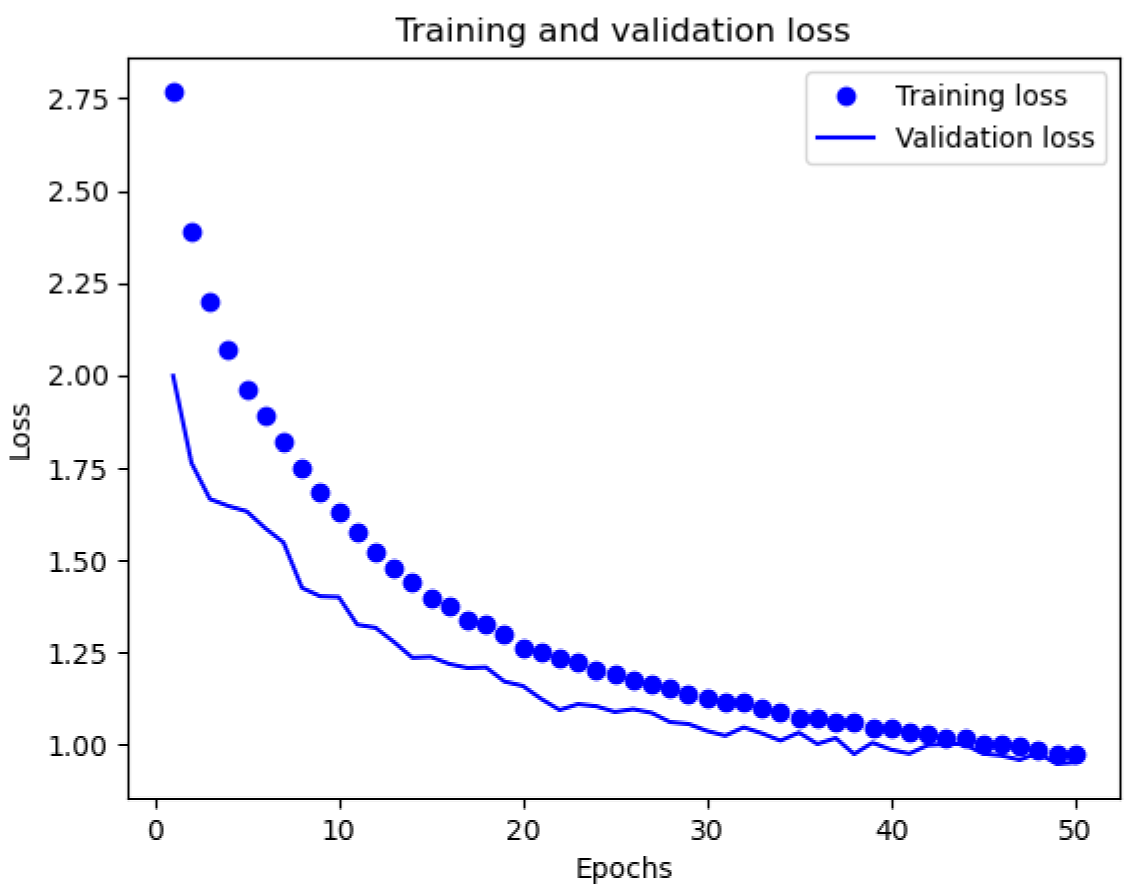
```
Evaluate on test data
23/23 [==============================] - 2s 62ms/step - loss: 0.9499 - accuracy: 0.6432
test loss, test acc: [0.9498971104621887, 0.6431564092636108]
```

- Training and validation loss curves

In [ ]:

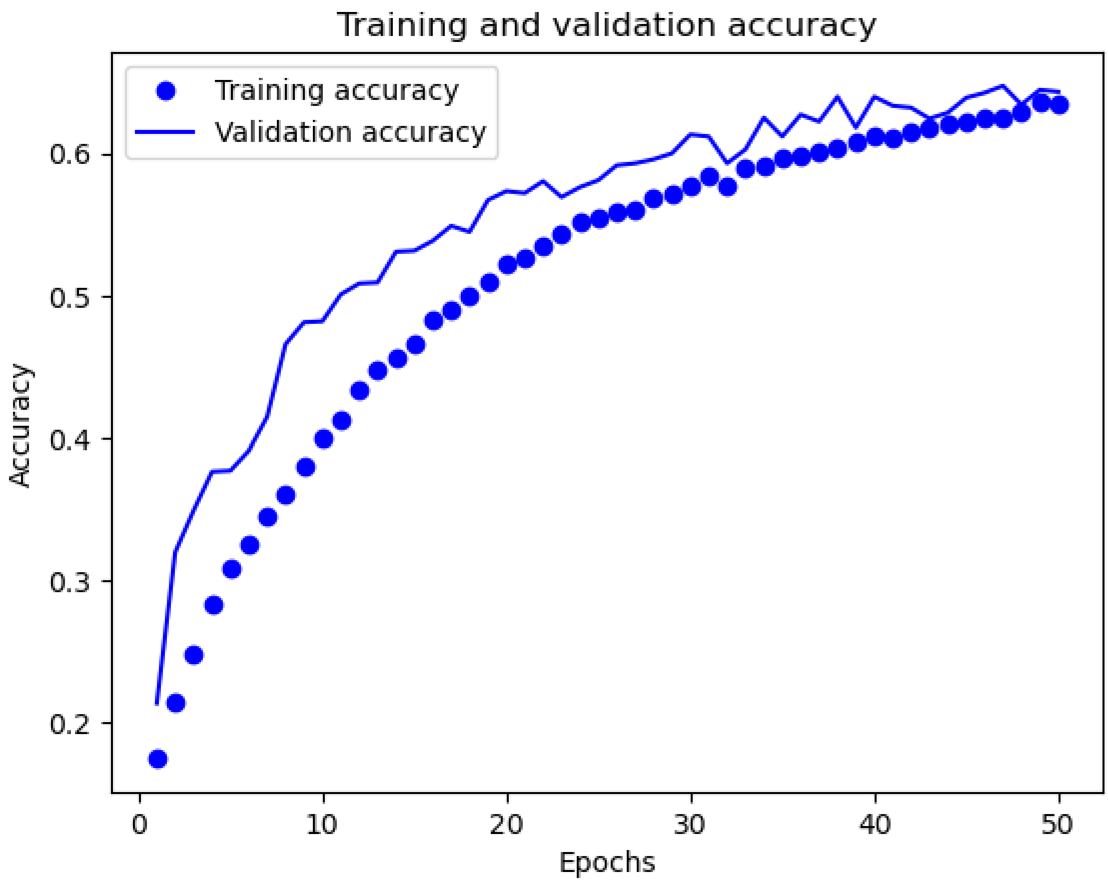
```
# Plot the train and validation loss

import matplotlib.pyplot as plt
train_loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(train_loss) + 1)
plt.plot(epochs, train_loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [ ]:

```
# Plot the train and validation accuracy
train_acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
plt.plot(epochs, train_acc, 'bo', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

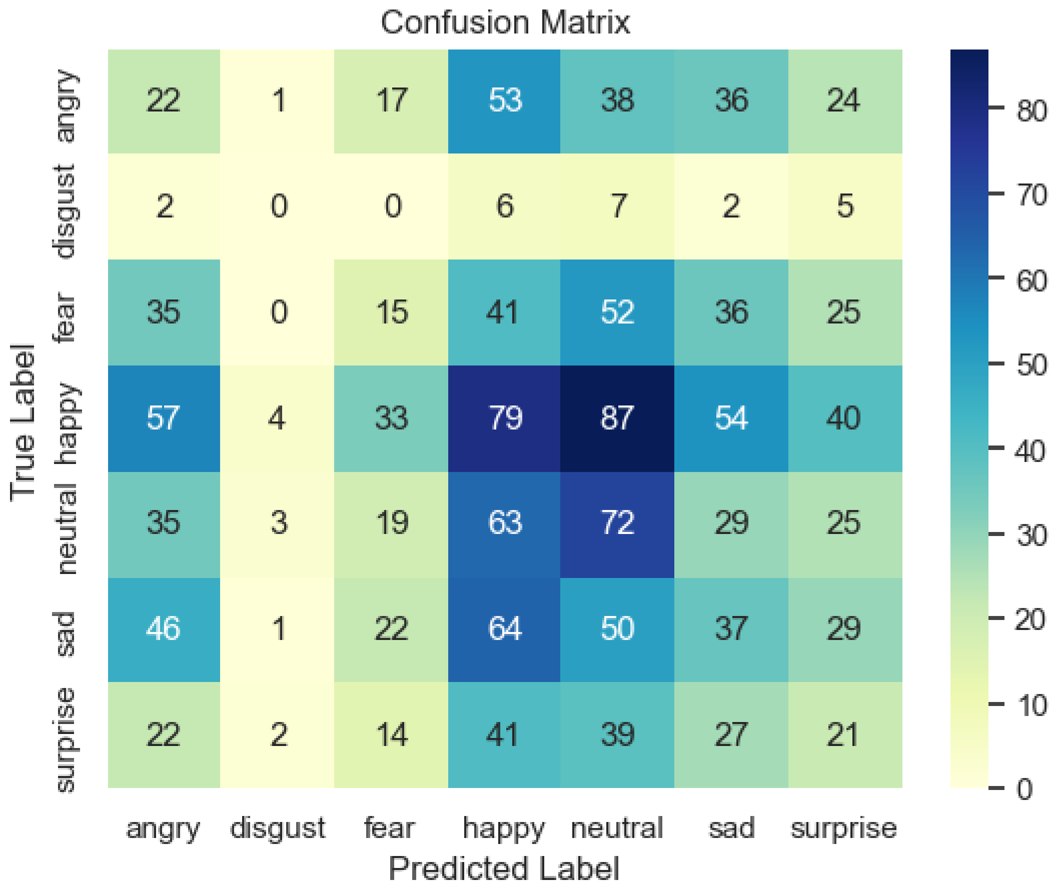


In [ ]:

```
import seaborn as sns
from sklearn.metrics import confusion_matrix
import numpy as np
import seaborn as sns

# Get the true labels and predicted labels for the validation set
validation_labels = validation_generator.classes
validation_pred_probs = model.predict(validation_generator)
validation_pred_labels = np.argmax(validation_pred_probs, axis=1)

# Compute the confusion matrix
confusion_mtx = confusion_matrix(validation_labels, validation_pred_labels)
class_names = list(train_generator.class_indices.keys())
sns.set()
sns.heatmap(confusion_mtx, annot=True, fmt='d', cmap='YlGnBu',
            xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```


Confusion Matrix
True Label
Predicted Label
angry
disgust
fear
happy
neutral
sad
surprise
22 1 17 53 38 36 24
2 0 0 6 7 2 5
35 0 15 41 52 36 25
57 4 33 79 87 54 40
35 3 19 63 72 29 25
46 1 22 64 50 37 29
22 2 14 41 39 27 21
0
10
20
30
40
50
60
70
80