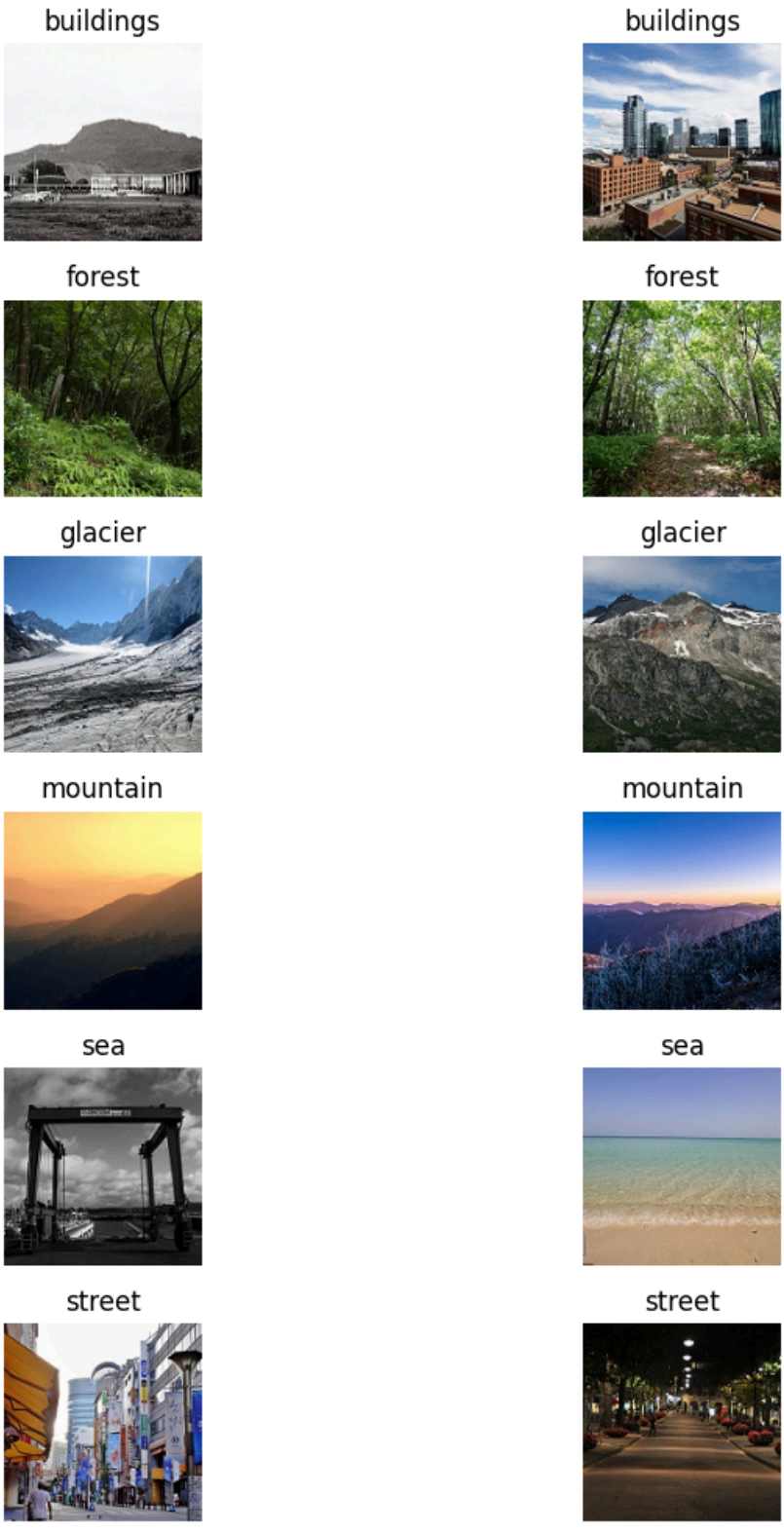
LAB5

Name: Kushal Sourav B

Regno: 2347125

Displaying the Images

In [2]:
```
import matplotlib.pyplot as plt
import os
from PIL import Image

def display_samples(dataset_dir, categories, num_samples=2):
    fig, axes = plt.subplots(len(categories), num_samples, figsize=(10, 10))
    for i, category in enumerate(categories):
        folder = os.path.join(dataset_dir, category)
        for j in range(num_samples):
            img_path = os.path.join(folder, os.listdir(folder)[j])
            img = Image.open(img_path)
            axes[i, j].imshow(img)
            axes[i, j].set_title(category)
            axes[i, j].axis('off')
    plt.tight_layout()
    plt.show()


categories = ['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street']
display_samples('dataset/seg_train/seg_train/', categories)
```



Model Architecture:

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

model = Sequential()

# 1st Convolutional Block
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

# 2nd Convolutional Block
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(BatchNormalization())

# 3rd Convolutional Block
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

# Fully Connected Layers
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(6, activation='softmax'))
```

```
c:\Users\USER\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model Training:

In [12]:
```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Image Data Augmentation
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)  # 80/20 split

train_generator = train_datagen.flow_from_directory(
    'dataset/seg_train/seg_train/',
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    subset='training')

validation_generator = train_datagen.flow_from_directory(
    'dataset/seg_train/seg_train/',
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    subset='validation')

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

# Train the model
history = model.fit(train_generator, epochs=20, validation_data=validation_generator)
```

```
Found 11230 images belonging to 6 classes.
Found 2804 images belonging to 6 classes.
Epoch 1/20
```

```
c:\Users\USER\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:122: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
```

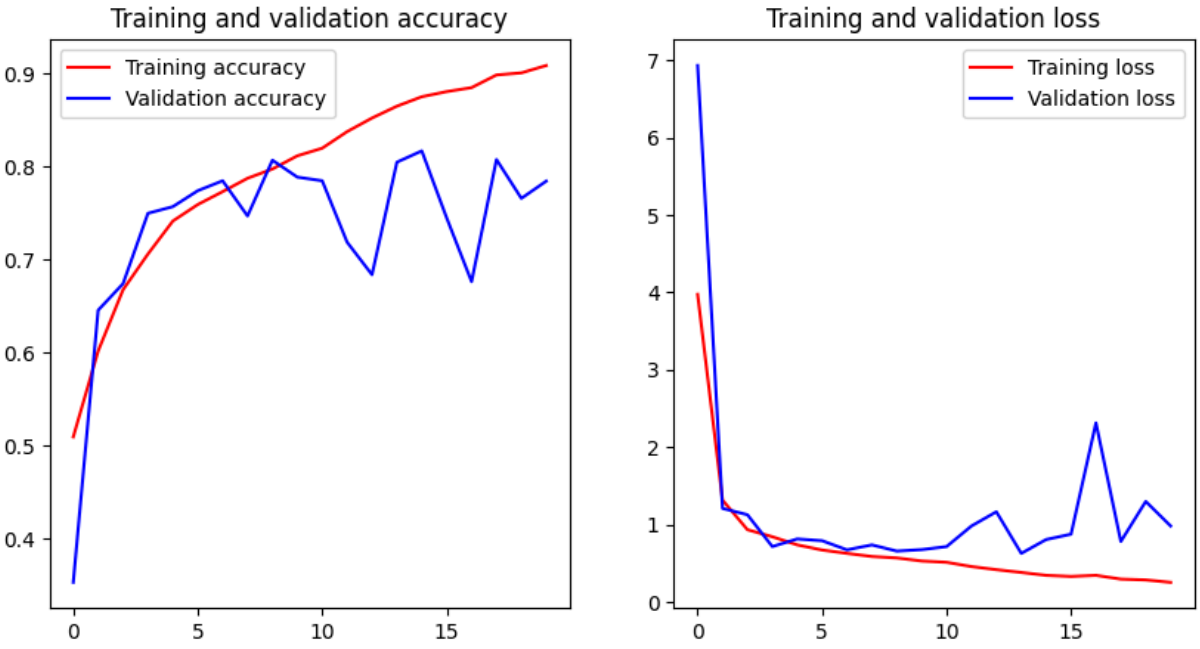
```
351/351 ━━━━━━━━━━━━━━━━━━━━ 315s 884ms/step - accuracy: 0.4831 - loss: 5.0357 - val_accuracy: 0.3531 - v
al_loss: 6.9323
Epoch 2/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 249s 709ms/step - accuracy: 0.5745 - loss: 1.4792 - val_accuracy: 0.6455 - v
al_loss: 1.2076
Epoch 3/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 234s 666ms/step - accuracy: 0.6516 - loss: 0.9739 - val_accuracy: 0.6740 - v
al_loss: 1.1260
Epoch 4/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 253s 721ms/step - accuracy: 0.7071 - loss: 0.8253 - val_accuracy: 0.7496 - v
al_loss: 0.7145
Epoch 5/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 245s 696ms/step - accuracy: 0.7432 - loss: 0.7367 - val_accuracy: 0.7568 - v
al_loss: 0.8130
Epoch 6/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 284s 810ms/step - accuracy: 0.7556 - loss: 0.6726 - val_accuracy: 0.7739 - v
al_loss: 0.7900
Epoch 7/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 263s 748ms/step - accuracy: 0.7719 - loss: 0.6267 - val_accuracy: 0.7846 - v
al_loss: 0.6722
Epoch 8/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 244s 694ms/step - accuracy: 0.7963 - loss: 0.5608 - val_accuracy: 0.7468 - v
al_loss: 0.7361
Epoch 9/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 222s 631ms/step - accuracy: 0.7963 - loss: 0.5657 - val_accuracy: 0.8067 - v
al_loss: 0.6560
Epoch 10/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 256s 730ms/step - accuracy: 0.8186 - loss: 0.5093 - val_accuracy: 0.7885 - v
al_loss: 0.6749
Epoch 11/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 269s 765ms/step - accuracy: 0.8267 - loss: 0.4883 - val_accuracy: 0.7846 - v
al_loss: 0.7140
Epoch 12/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 245s 696ms/step - accuracy: 0.8368 - loss: 0.4558 - val_accuracy: 0.7186 - v
al_loss: 0.9805
Epoch 13/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 232s 661ms/step - accuracy: 0.8580 - loss: 0.4046 - val_accuracy: 0.6837 - v
al_loss: 1.1641
Epoch 14/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 229s 651ms/step - accuracy: 0.8643 - loss: 0.3843 - val_accuracy: 0.8046 - v
al_loss: 0.6261
Epoch 15/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 229s 651ms/step - accuracy: 0.8858 - loss: 0.3146 - val_accuracy: 0.8167 - v
al_loss: 0.8053
Epoch 16/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 252s 718ms/step - accuracy: 0.8823 - loss: 0.3204 - val_accuracy: 0.7447 - v
al_loss: 0.8746
Epoch 17/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 256s 730ms/step - accuracy: 0.8819 - loss: 0.3324 - val_accuracy: 0.6762 - v
al_loss: 2.3130
Epoch 18/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 254s 723ms/step - accuracy: 0.8986 - loss: 0.2931 - val_accuracy: 0.8074 - v
al_loss: 0.7799
Epoch 19/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 251s 715ms/step - accuracy: 0.9047 - loss: 0.2619 - val_accuracy: 0.7657 - v
al_loss: 1.2976
Epoch 20/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 246s 699ms/step - accuracy: 0.9135 - loss: 0.2453 - val_accuracy: 0.7842 - v
al_loss: 0.9805
```

Evaluation:

In [13]:

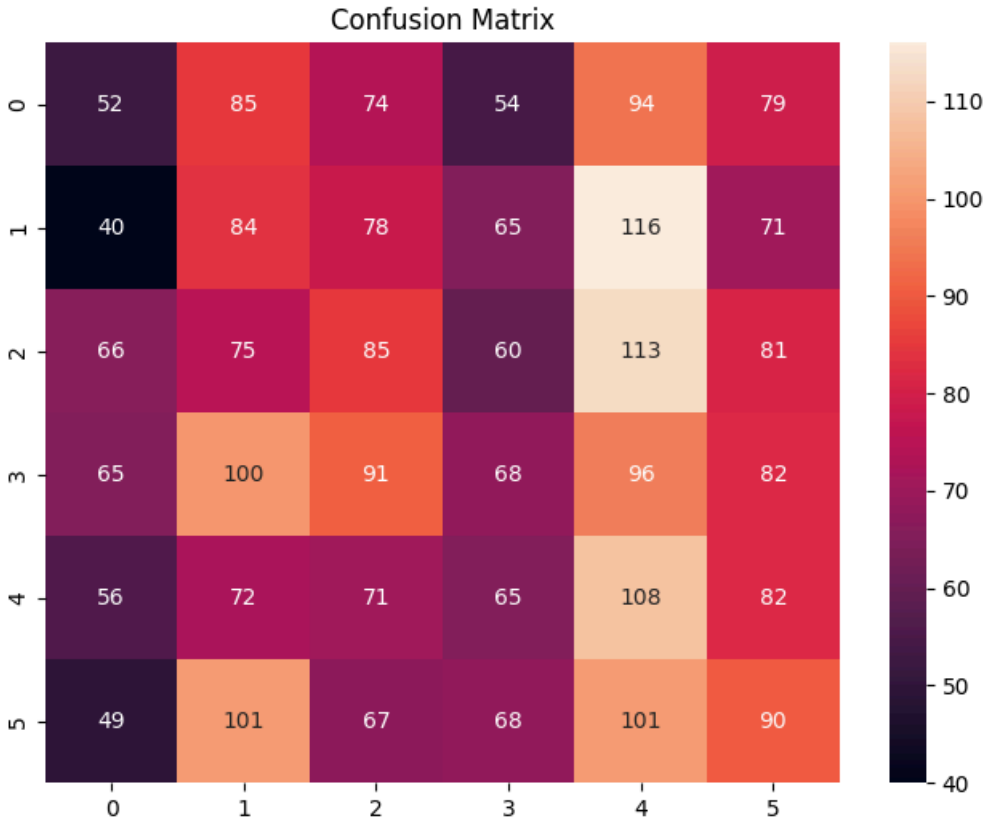
```python
import matplotlib.pyplot as plt

def plot_history(history):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    epochs = range(len(acc))

    plt.figure(figsize=(10,5))
```

```
    plt.subplot(1,2,1)
    plt.plot(epochs, acc, 'r', label='Training accuracy')
    plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
    plt.title('Training and validation accuracy')
    plt.legend()


    plt.subplot(1,2,2)
    plt.plot(epochs, loss, 'r', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend()

    plt.show()

plot_history(history)
```



In [15]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import numpy as np


Y_pred = model.predict(validation_generator)
y_pred = np.argmax(Y_pred, axis=1)


cm = confusion_matrix(validation_generator.classes, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Confusion Matrix')
plt.show()
```

```
88/88 ━━━━━━━━━━━━━━━━━━━━ 13s 147ms/step
```



Optimization

In [4]:
```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2)

train_generator = train_datagen.flow_from_directory(
    'dataset/seg_train/seg_train/',
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    subset='training')

validation_generator = train_datagen.flow_from_directory(
    'dataset/seg_train/seg_train/',
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    subset='validation')

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

# Train the model
history = model.fit(train_generator, epochs=20, validation_data=validation_generator)
```

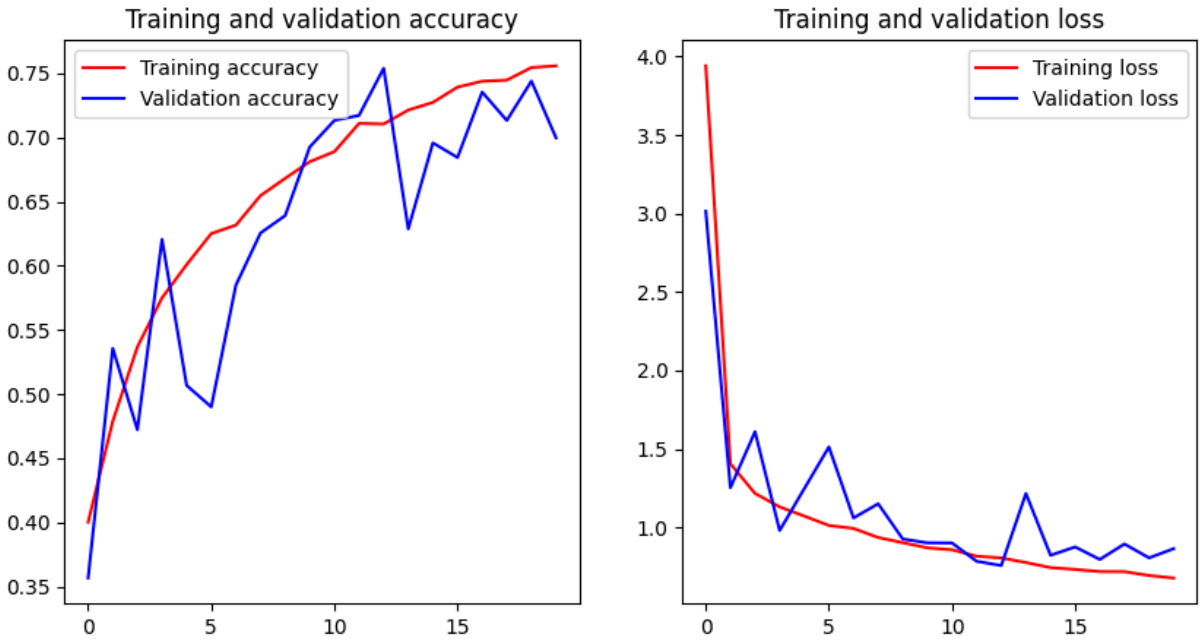
```
Found 11230 images belonging to 6 classes.
Found 2804 images belonging to 6 classes.
Epoch 1/20
```

```
c:\Users\USER\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\trainers\data_adapters
\py_dataset_adapter.py:122: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)`
in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pas
s these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
351/351 ━━━━━━━━━━━━━━━━━━━━ 315s 885ms/step - accuracy: 0.3856 - loss: 5.5625 - val_accuracy: 0.3566 - v
al_loss: 3.0143
Epoch 2/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 246s 699ms/step - accuracy: 0.4612 - loss: 1.4952 - val_accuracy: 0.5357 - v
al_loss: 1.2523
Epoch 3/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 254s 720ms/step - accuracy: 0.5257 - loss: 1.2428 - val_accuracy: 0.4722 - v
al_loss: 1.6088
Epoch 4/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 270s 768ms/step - accuracy: 0.5670 - loss: 1.1505 - val_accuracy: 0.6205 - v
al_loss: 0.9796
Epoch 5/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 254s 722ms/step - accuracy: 0.6081 - loss: 1.0669 - val_accuracy: 0.5068 - v
al_loss: 1.2464
Epoch 6/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 254s 721ms/step - accuracy: 0.6156 - loss: 1.0243 - val_accuracy: 0.4900 - v
al_loss: 1.5121
Epoch 7/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 269s 763ms/step - accuracy: 0.6279 - loss: 1.0115 - val_accuracy: 0.5849 - v
al_loss: 1.0594
Epoch 8/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 263s 746ms/step - accuracy: 0.6491 - loss: 0.9474 - val_accuracy: 0.6255 - v
al_loss: 1.1502
Epoch 9/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 262s 744ms/step - accuracy: 0.6644 - loss: 0.9183 - val_accuracy: 0.6391 - v
al_loss: 0.9254
Epoch 10/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 249s 707ms/step - accuracy: 0.6788 - loss: 0.8754 - val_accuracy: 0.6926 - v
al_loss: 0.9011
Epoch 11/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 248s 705ms/step - accuracy: 0.6898 - loss: 0.8524 - val_accuracy: 0.7133 - v
al_loss: 0.9001
Epoch 12/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 244s 692ms/step - accuracy: 0.7086 - loss: 0.8287 - val_accuracy: 0.7172 - v
al_loss: 0.7833
Epoch 13/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 244s 692ms/step - accuracy: 0.7179 - loss: 0.8030 - val_accuracy: 0.7539 - v
al_loss: 0.7564
Epoch 14/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 244s 693ms/step - accuracy: 0.7176 - loss: 0.7811 - val_accuracy: 0.6287 - v
al_loss: 1.2154
Epoch 15/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 248s 705ms/step - accuracy: 0.7328 - loss: 0.7229 - val_accuracy: 0.6958 - v
al_loss: 0.8219
Epoch 16/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 245s 695ms/step - accuracy: 0.7360 - loss: 0.7356 - val_accuracy: 0.6844 - v
al_loss: 0.8748
Epoch 17/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 244s 694ms/step - accuracy: 0.7521 - loss: 0.7032 - val_accuracy: 0.7354 - v
al_loss: 0.7952
Epoch 18/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 245s 695ms/step - accuracy: 0.7387 - loss: 0.7260 - val_accuracy: 0.7133 - v
al_loss: 0.8937
Epoch 19/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 244s 693ms/step - accuracy: 0.7589 - loss: 0.6831 - val_accuracy: 0.7439 - v
al_loss: 0.8052
Epoch 20/20
351/351 ━━━━━━━━━━━━━━━━━━━━ 244s 694ms/step - accuracy: 0.7592 - loss: 0.6826 - val_accuracy: 0.6997 - v
al_loss: 0.8637
```

In [5]:

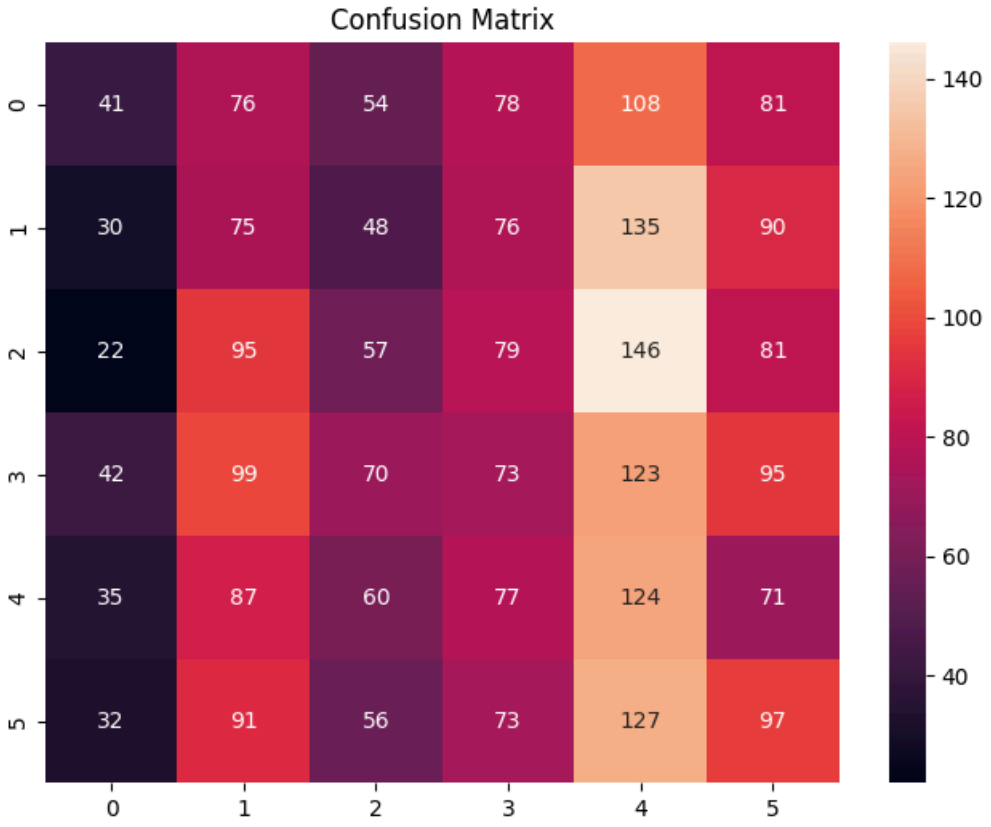
```python
import matplotlib.pyplot as plt

def plot_history(history):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    epochs = range(len(acc))
```

```
    plt.figure(figsize=(10,5))

    plt.subplot(1,2,1)
    plt.plot(epochs, acc, 'r', label='Training accuracy')
    plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
    plt.title('Training and validation accuracy')
    plt.legend()


    plt.subplot(1,2,2)
    plt.plot(epochs, loss, 'r', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend()

    plt.show()

plot_history(history)
```



In [6]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import numpy as np


Y_pred = model.predict(validation_generator)
y_pred = np.argmax(Y_pred, axis=1)


cm = confusion_matrix(validation_generator.classes, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Confusion Matrix')
plt.show()
```

```
88/88 ━━━━━━━━━━━━━━━━━━━━ 19s 216ms/step
```


Confusion Matrix
41 76 54 78 108 81
30 75 48 76 135 90
22 95 57 79 146 81
42 99 70 73 123 95
35 87 60 77 124 71
32 91 56 73 127 97
0 1 2 3 4 5
140 120 100 80 60 40