In [2]:
```
import os
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical

data_dir = "cropped_images"

classes = {
    'n02094433': 0,
    'n02099429': 1,
    'n02107142': 2,
    'n02111500': 3
}

IMG_HEIGHT = 64
IMG_WIDTH = 64

images = []
labels = []

for file_name in os.listdir(data_dir):
    file_path = os.path.join(data_dir, file_name)
    for class_id, label in classes.items():
        if class_id in file_name:
            img = cv2.imread(file_path)
            if img is not None:
                img = cv2.resize(img, (IMG_WIDTH, IMG_HEIGHT))
                images.append(img)
                labels.append(label)
            break

images = np.array(images, dtype='float32')
labels = np.array(labels, dtype='int')

images = images / 255.0

X_train, X_val, y_train, y_val = train_test_split(images, labels, test_size=0.2, st

y_train = to_categorical(y_train, num_classes=len(classes))
y_val = to_categorical(y_val, num_classes=len(classes))
```

In [3]:
```
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt

model = models.Sequential([

    layers.Conv2D(8, (3, 3), activation='relu', input_shape=(X_train.shape[1], X_tr
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(4, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(8, activation='relu'),
    layers.Dense(4, activation='softmax')
```

```
])

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

history = model.fit(X_train, y_train,
                    epochs=20,
                    batch_size=32,
                    validation_split=0.2,
                    verbose=1)

plt.figure(figsize=(10, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.grid()
plt.show()
```

```
c:\Users\Lenovo\anaconda3\Lib\site-packages\keras\src\layers\convolutional\base_con
v.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 lay
er in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
Epoch 1/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - accuracy: 0.2770 - loss: 1.3844 - val_accu
racy: 0.4655 - val_loss: 1.3455
Epoch 2/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4642 - loss: 1.3244 - val_accu
racy: 0.4138 - val_loss: 1.2675
Epoch 3/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.4355 - loss: 1.2223 - val_accu
racy: 0.4828 - val_loss: 1.2248
Epoch 4/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.5030 - loss: 1.1529 - val_accu
racy: 0.5172 - val_loss: 1.1482
Epoch 5/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.5615 - loss: 1.0513 - val_accu
racy: 0.4569 - val_loss: 1.1501
Epoch 6/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.5872 - loss: 0.9907 - val_accu
racy: 0.5517 - val_loss: 1.0617
Epoch 7/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.6481 - loss: 0.8718 - val_accu
racy: 0.5172 - val_loss: 1.0677
Epoch 8/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.6465 - loss: 0.8273 - val_accu
racy: 0.5517 - val_loss: 1.0133
Epoch 9/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.6317 - loss: 0.7743 - val_accu
racy: 0.6466 - val_loss: 0.9391
Epoch 10/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - accuracy: 0.7571 - loss: 0.5885 - val_accu
racy: 0.6379 - val_loss: 0.9461
Epoch 11/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.7393 - loss: 0.6275 - val_accu
racy: 0.6466 - val_loss: 0.9499
Epoch 12/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7307 - loss: 0.6377 - val_accu
racy: 0.6034 - val_loss: 1.0331
Epoch 13/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7206 - loss: 0.6884 - val_accu
racy: 0.5776 - val_loss: 1.1026
Epoch 14/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7839 - loss: 0.5980 - val_accu
racy: 0.6724 - val_loss: 0.8915
Epoch 15/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8085 - loss: 0.5192 - val_accu
racy: 0.6466 - val_loss: 0.9508
Epoch 16/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8213 - loss: 0.4598 - val_accu
racy: 0.6638 - val_loss: 0.9101
Epoch 17/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8518 - loss: 0.4053 - val_accu
racy: 0.6897 - val_loss: 0.9527
Epoch 18/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8204 - loss: 0.4610 - val_accu
racy: 0.6724 - val_loss: 0.9237
Epoch 19/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8385 - loss: 0.4286 - val_accu
```

```
racy: 0.6983 - val_loss: 0.8930
Epoch 20/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.8714 - loss: 0.3654 - val_accu
racy: 0.6983 - val_loss: 0.9810
```

Banner ID: 916478184 (b) Train the CNN using 2 other number of filters: 8 and 16 for the 2nd convolution layer (i) with all other parameters unchanged

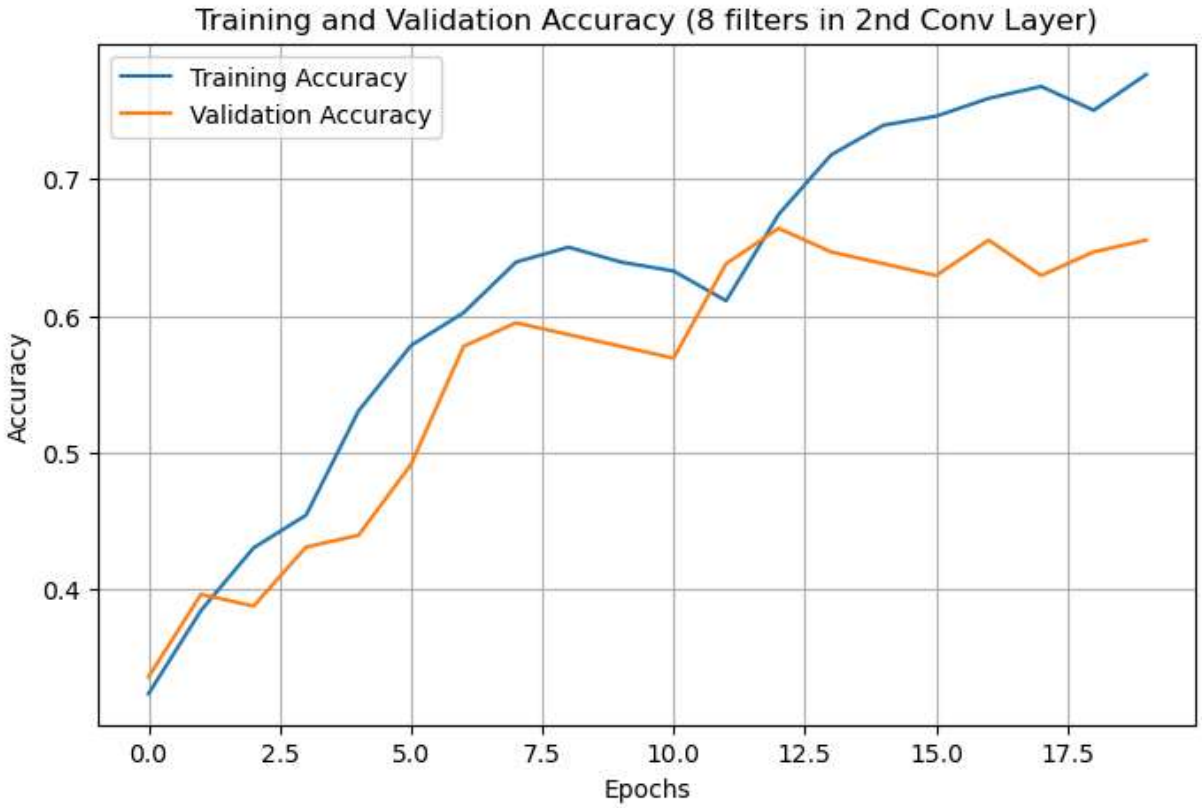
In [ ]:
```python
def build_cnn(num_filters_second_layer):
    model = models.Sequential([
        layers.Conv2D(8, (3, 3), activation='relu', input_shape=(X_train.shape[1],
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(num_filters_second_layer, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Flatten(),
        layers.Dense(8, activation='relu'),
        layers.Dense(4, activation='softmax')
    ])
    return model

histories = {}

for num_filters in [8, 16]:
    print(f"Training model with {num_filters} filters in the second convolutional l
    model = build_cnn(num_filters)
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    history = model.fit(X_train, y_train,
                        epochs=20,
                        batch_size=32,
```

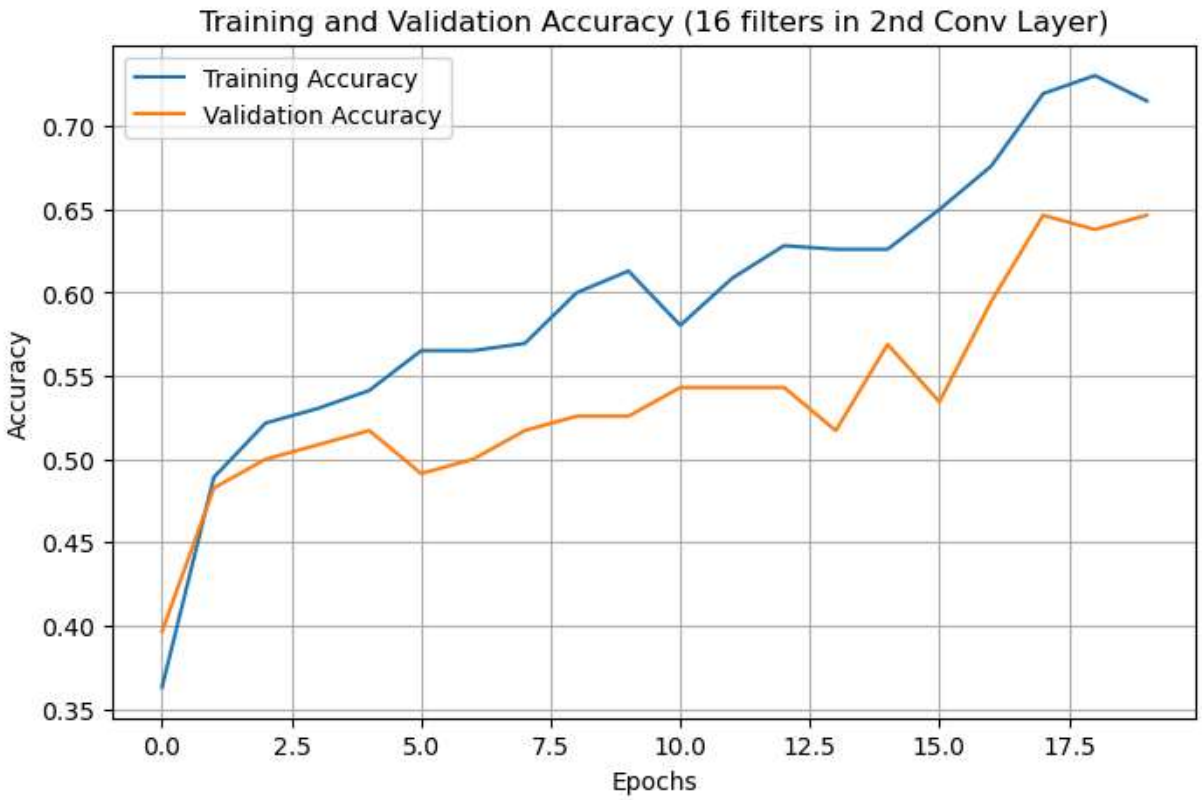
```
                        validation_split=0.2,
                        verbose=1)
    histories[num_filters] = history

for num_filters, history in histories.items():
    plt.figure(figsize=(8, 5))
    plt.plot(history.history['accuracy'], label='Training Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.title(f'Training and Validation Accuracy ({num_filters} filters in 2nd Conv
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.grid()
    plt.show()
```

```
Training model with 8 filters in the second convolutional layer...
Epoch 1/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - accuracy: 0.3073 - loss: 1.3880 - val_accu
racy: 0.3362 - val_loss: 1.3844
Epoch 2/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.3699 - loss: 1.3462 - val_accu
racy: 0.3966 - val_loss: 1.3526
Epoch 3/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.4182 - loss: 1.2901 - val_accu
racy: 0.3879 - val_loss: 1.2895
Epoch 4/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - accuracy: 0.4150 - loss: 1.2473 - val_accu
racy: 0.4310 - val_loss: 1.2276
Epoch 5/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.5164 - loss: 1.1496 - val_accu
racy: 0.4397 - val_loss: 1.2900
Epoch 6/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.5792 - loss: 1.1042 - val_accu
racy: 0.4914 - val_loss: 1.1738
Epoch 7/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.5596 - loss: 1.0464 - val_accu
racy: 0.5776 - val_loss: 1.0932
Epoch 8/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.6254 - loss: 0.9745 - val_accu
racy: 0.5948 - val_loss: 1.0457
Epoch 9/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.6675 - loss: 0.8744 - val_accu
racy: 0.5862 - val_loss: 1.0462
Epoch 10/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.6538 - loss: 0.8892 - val_accu
racy: 0.5776 - val_loss: 1.0490
Epoch 11/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.6213 - loss: 0.8808 - val_accu
racy: 0.5690 - val_loss: 1.0461
Epoch 12/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.6039 - loss: 0.8934 - val_accu
racy: 0.6379 - val_loss: 0.9817
Epoch 13/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.7099 - loss: 0.8072 - val_accu
racy: 0.6638 - val_loss: 0.9635
Epoch 14/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7178 - loss: 0.7658 - val_accu
racy: 0.6466 - val_loss: 0.9525
Epoch 15/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7428 - loss: 0.6955 - val_accu
racy: 0.6379 - val_loss: 0.9495
Epoch 16/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7565 - loss: 0.6396 - val_accu
racy: 0.6293 - val_loss: 0.9712
Epoch 17/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7531 - loss: 0.6326 - val_accu
racy: 0.6552 - val_loss: 0.9408
Epoch 18/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7837 - loss: 0.6016 - val_accu
racy: 0.6293 - val_loss: 0.9411
Epoch 19/20
```

```
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7121 - loss: 0.6504 - val_accu
racy: 0.6466 - val_loss: 0.9268
Epoch 20/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7820 - loss: 0.5474 - val_accu
racy: 0.6552 - val_loss: 0.9323
Training model with 16 filters in the second convolutional layer...
Epoch 1/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 2s 32ms/step - accuracy: 0.2911 - loss: 1.3680 - val_accu
racy: 0.3966 - val_loss: 1.2214
Epoch 2/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.4545 - loss: 1.1805 - val_accu
racy: 0.4828 - val_loss: 1.0841
Epoch 3/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.5019 - loss: 1.0358 - val_accu
racy: 0.5000 - val_loss: 1.0477
Epoch 4/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.5176 - loss: 0.9784 - val_accu
racy: 0.5086 - val_loss: 1.0525
Epoch 5/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.5355 - loss: 0.9458 - val_accu
racy: 0.5172 - val_loss: 1.0413
Epoch 6/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.5589 - loss: 0.8978 - val_accu
racy: 0.4914 - val_loss: 1.0761
Epoch 7/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.5389 - loss: 0.8963 - val_accu
racy: 0.5000 - val_loss: 1.0042
Epoch 8/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.5630 - loss: 0.9079 - val_accu
racy: 0.5172 - val_loss: 1.0366
Epoch 9/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.5696 - loss: 0.9164 - val_accu
racy: 0.5259 - val_loss: 1.0390
Epoch 10/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.6209 - loss: 0.7789 - val_accu
racy: 0.5259 - val_loss: 0.9896
Epoch 11/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.5889 - loss: 0.7829 - val_accu
racy: 0.5431 - val_loss: 0.9846
Epoch 12/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.6274 - loss: 0.7330 - val_accu
racy: 0.5431 - val_loss: 0.9750
Epoch 13/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.6343 - loss: 0.7915 - val_accu
racy: 0.5431 - val_loss: 1.0518
Epoch 14/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.6256 - loss: 0.7710 - val_accu
racy: 0.5172 - val_loss: 1.0721
Epoch 15/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.5983 - loss: 0.7461 - val_accu
racy: 0.5690 - val_loss: 1.0081
Epoch 16/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.6610 - loss: 0.6675 - val_accu
racy: 0.5345 - val_loss: 0.9752
Epoch 17/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.6769 - loss: 0.6624 - val_accu
```

```
racy: 0.5948 - val_loss: 0.9757
Epoch 18/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.7075 - loss: 0.6949 - val_accu
racy: 0.6466 - val_loss: 0.9613
Epoch 19/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7376 - loss: 0.6655 - val_accu
racy: 0.6379 - val_loss: 0.9738
Epoch 20/20
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7206 - loss: 0.6167 - val_accu
racy: 0.6466 - val_loss: 0.9577
```





The first model with 4 filters in the 2nd Conv Layer shows signs of underfitting, as it has fewer filters and, consequently, less capacity to learn complex patterns. The second model with 8 filters in the 2nd Conv strikes a balance between underfitting and overfitting. It's likely a better fit for the dataset as the gap between training and validation accuracy is small. It is just right for this dataset. Third model shows signs of overfitting, as it may capture noise and irrelevant details in the training data, leading to more gap between training and validation performance.