link text**# Facial Emotion Detection**

Problem Definition

The context: Why is this problem important to solve?
The objectives: What is the intended goal?
The key questions: What are the key questions that need to be answered?
The problem formulation: What are we trying to solve using data science?

About the dataset

The data set consists of 3 folders, i.e., 'test', 'train', and 'validation'. Each of these folders has four subfolders:

'happy': Images of people who have happy facial expressions.
'sad': Images of people with sad or upset facial expressions.
'surprise': Images of people who have shocked or surprised facial expressions.
'neutral': Images of people showing no prominent emotion in their facial expression at all.

Important Notes

- This notebook can be considered a guide to refer to while solving the problem. The evaluation will be as per the Rubric shared for each Milestone. Unlike previous courses, it does not follow the pattern of the graded questions in different sections. This notebook would give you a direction on what steps need to be taken to get a feasible solution to the problem. Please note that this is just one way of doing this. **There can be other 'creative' ways to solve the problem, and we encourage you to feel free and explore them as an 'optional' exercise.**
- In the notebook, there are markdown cells called Observations and Insights. It is a good practice to provide observations and extract insights from the outputs.
- The naming convention for different variables can vary. **Please consider the code provided in this notebook as a sample code.**
- All the outputs in the notebook are just for reference and can be different if you follow a different approach.
- There are sections called **Think About It** in the notebook that will help you get a better understanding of the reasoning behind a particular technique/step. Interested learners can take alternative approaches if they want to explore different techniques.

Mounting the Drive

NOTE: Please use Google Colab from your browser for this notebook. **Google.colab is NOT a library that can be downloaded locally on your device.**

In [1]:

```
# Mounting the drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Importing the Libraries

In [ ]:

```
import zipfile
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import os

# Importing Deep Learning Libraries

from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Input, Dropout, GlobalAveragePooling2D, Flatt
en, Conv2D, BatchNormalization, Activation, MaxPooling2D, LeakyReLU
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
```

Let us load the data

Note:

- **You must download the dataset from the link provided on Olympus and upload the same on your Google drive before executing the code in the next cell.**
- **In case of any error, please make sure that the path of the file is correct as the path may be different for you.**

In [ ]:

```
# Storing the path of the data file from the Google drive
path = '/content/drive/MyDrive/Facial_emotion_images.zip'

# The data is provided as a zip file so we need to extract the files from the zip file
with zipfile.ZipFile(path, 'r') as zip_ref:
    zip_ref.extractall()
```

In [ ]:

```
picture_size = 48
folder_path = "Facial_emotion_images/"
```

Visualizing our Classes

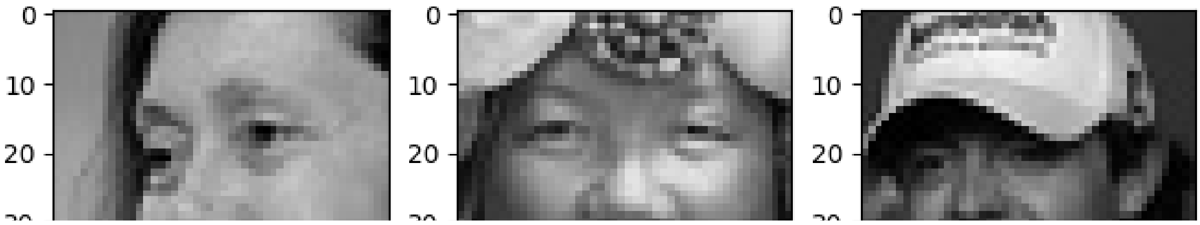
Let's look at our classes.

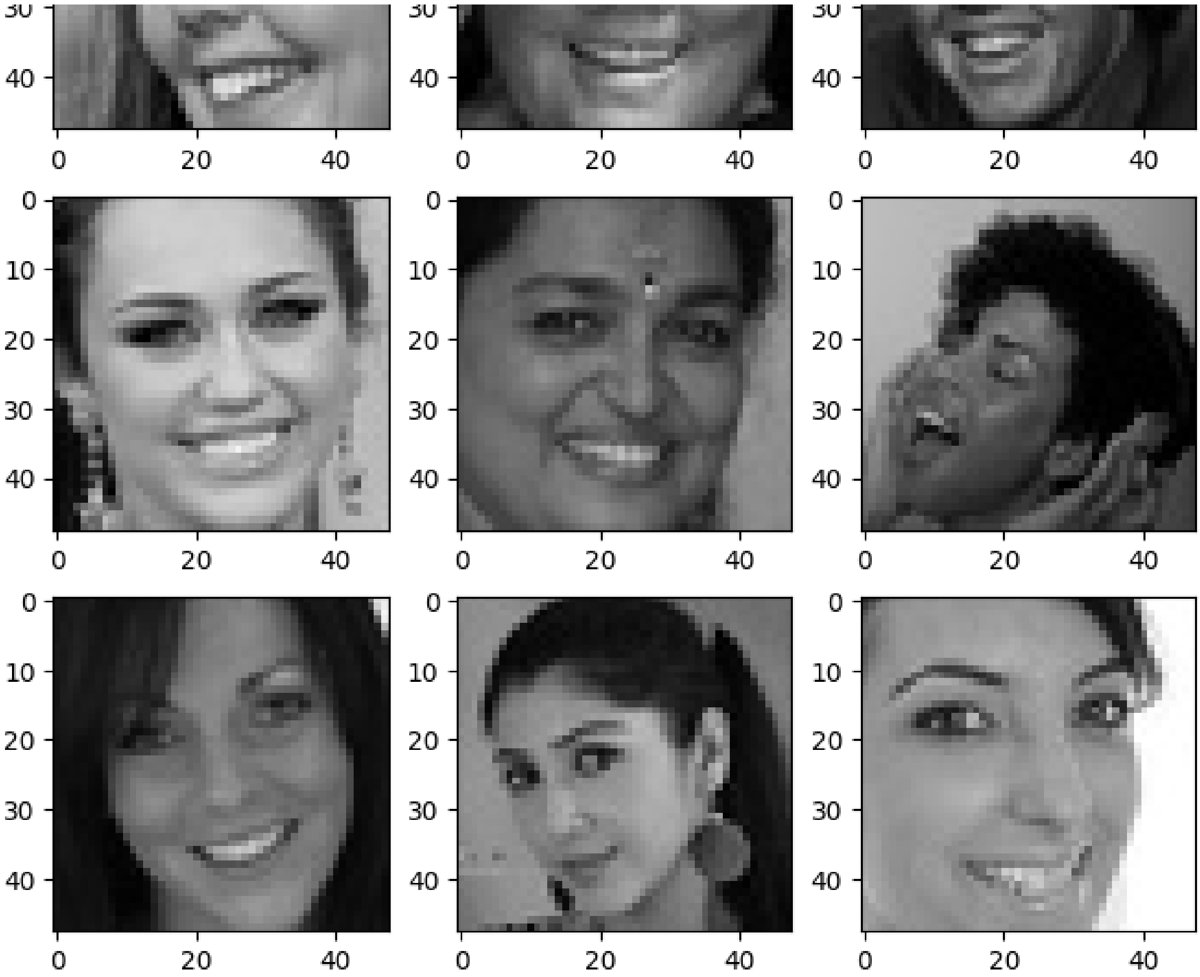
Write down your observation for each class. What do you think can be a unique feature of each emotion, that separates it from the remaining classes?

Happy

In [ ]:

```
expression = 'happy'

plt.figure(figsize= (8,8))
for i in range(1, 10, 1):
    plt.subplot(3, 3, i)

    img = load_img(folder_path + "train/" + expression + "/" +
                  os.listdir(folder_path + "train/" + expression)[i], target_size = (pic
ture_size, picture_size))
    plt.imshow(img)

plt.show()
```





Observations and Insights:

From top to bottom, we're seeing lots of smiles, a universal sign of happiness. The top gang's all grins, with the first two maybe sharing a joke, and the third rocking a hat with a chill vibe. Smiling eyes seal the deal, painting a picture of pure joy.

Diving into the middle, it's a mix of moods but still on the sunny side. One's playing it cool with a soft smile—content, maybe. The next one's warmth could light up a room, hinting at kindness or sheer happiness. And there's laughter that can't be contained, a dead giveaway of someone having a blast.

Down at the bottom, smiles all around but with a twist. One's all friendly-like, easy and happy. Next up, a composed smile whispers of satisfaction, maybe pride. And closing out, a wide grin that screams excitement or maybe just winning at life.

So, what's the scoop on spotting these vibes? Happiness hits with those full-face smiles, contentment's more about the chill smile, amusement has you laughing out loud, and satisfaction? That's when you're smiling like you've just nailed it.

Sad

In [ ]:

```
expression = 'sad'

plt.figure(figsize=(8, 8))
for i in range(1, 10):  # Adjust the range based on the number of images you want to dis
play
    plt.subplot(3, 3, i)

    img = load_img(folder_path + "train/" + expression + "/" +
                   os.listdir(folder_path + "train/" + expression)[i], target_size=(pict
ure_size, picture_size))
    plt.imshow(img)

plt.show()
```

Observations and Insights: Top to bottom, we've got a gallery of expressions. Starting off with a bit of side-eye smirk, hinting at someone not quite buying what they're hearing, skepticism in the air. Then, there's a baby, eyes wide with that "what's happening" look, could be surprise or just pure curiosity. Another's just chilling with a soft smile, radiating those good, content vibes.

Mid-way, we catch a kiddo looking down, face all serious—could be sadness or just lost in thought. Another's got that straight-faced, "thinking hard" look, eyes locked in. And then, a light smile breaks through, someone's definitely feeling the joy or just being super friendly.

Bottom line's more intense: one's all business, maybe even a bit miffed. A baby's frowning straight at us, could be a bit miffed or just confused. And finishing off with a stare that's all about focus, maybe a hint of worry or just frustration.

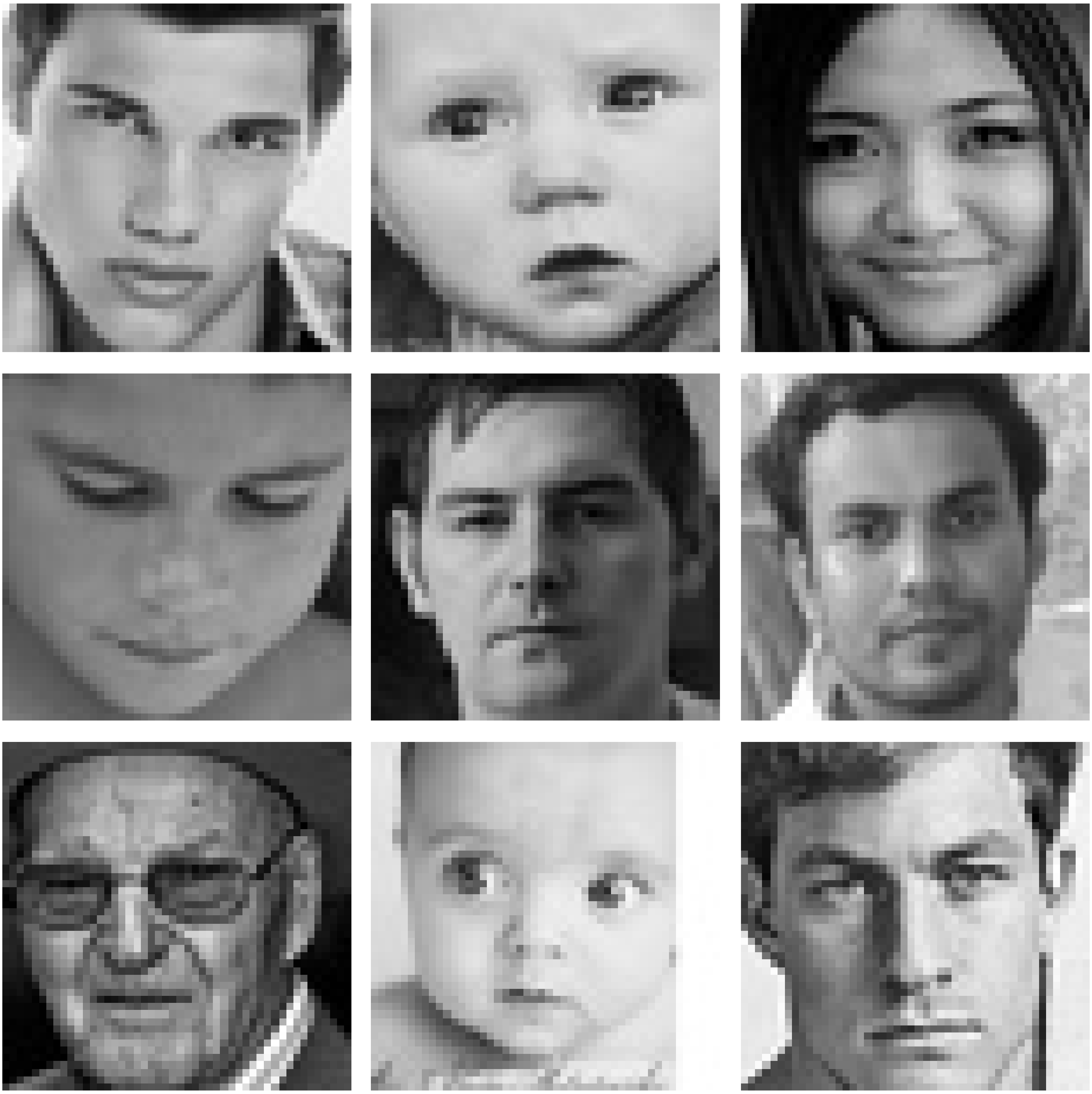
Breaking it down, each emotion's got its tell: skepticism's all about that quirky smirk, surprise pops with those big eyes, contentment eases in with gentle smiles. Sadness pulls the gaze down, seriousness locks it straight, and happiness spreads it wide. Anger tightens the lips, discontent furrows the brow, and concentration sharpens the look. Each face, a story of its own.

Neutral

In [ ]:

```
# Write your code to visualize images from the class 'neutral'.
expression = 'neutral'

plt.figure(figsize=(8, 8))
```

```
for i in range(1, 10):
    plt.subplot(3, 3, i)
    img_path = os.path.join(folder_path, "train", expression, os.listdir(os.path.join(fo
lder_path, "train", expression))[i])
    img = load_img(img_path, target_size=(picture_size, picture_size))   # Load the image
    plt.imshow(img)
    plt.axis('off')
plt.tight_layout()
plt.show()
```



Observations and Insights:

Across the board, we've got a mix of vibes. Top row's mostly lighting up with smiles and that sparkle in the eyes, spelling out happiness or sheer joy, except for one keeping it cool with a chill, neutral look.

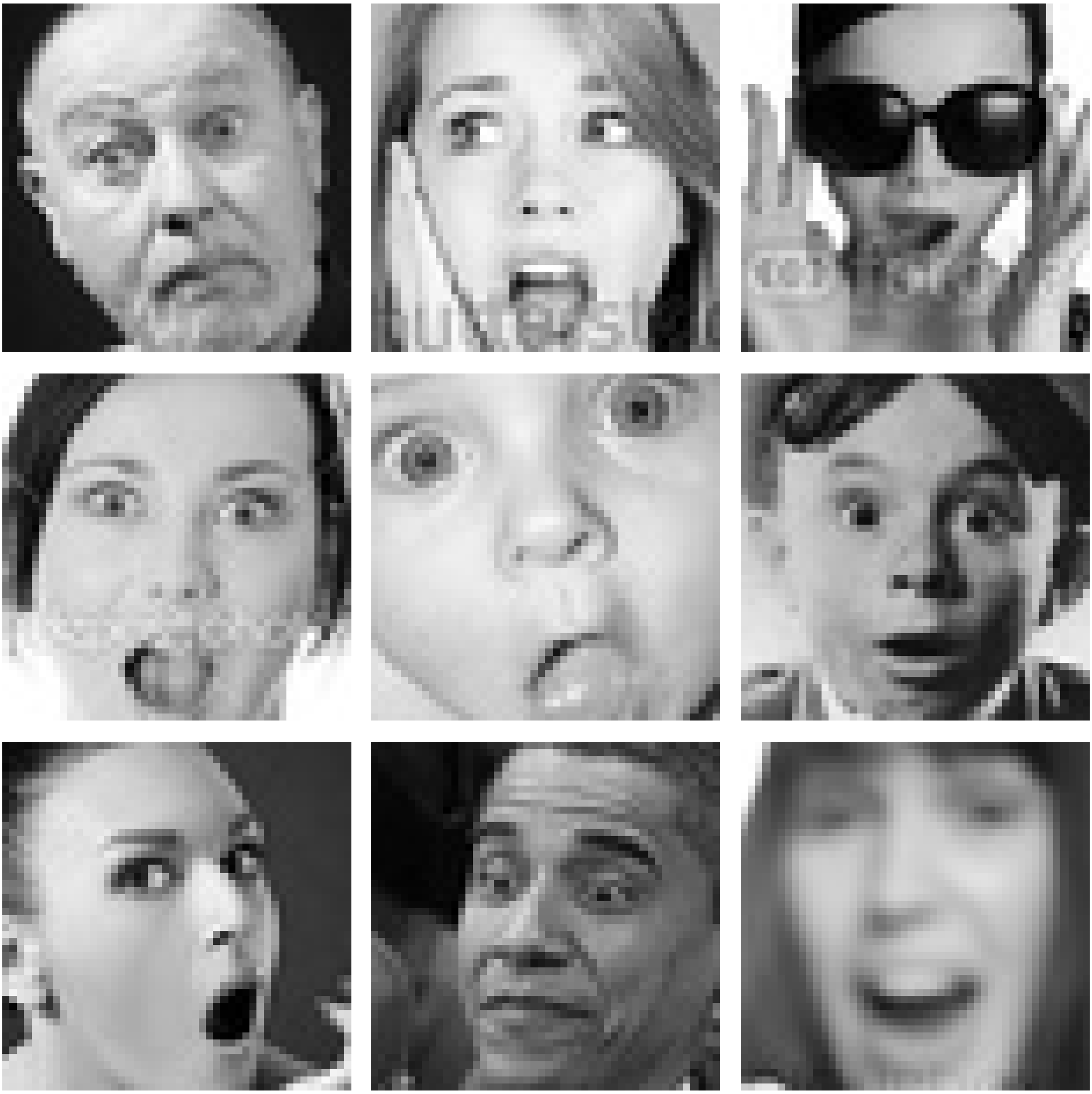
Diving into the middle, we're met with a bit of everything: one's rocking a serious vibe, maybe even a touch annoyed, while another's all smiles, oozing happiness or a warm, friendly feel. There's also someone hanging back with a laid-back, neutral face, just riding that calm wave.

Bottom row throws in a curveball, with one person looking down, maybe a tad sad or lost in thought. Another's not having any of it, lips pressed in a line that could mean disapproval or just sheer grit. And rounding it off, there's someone with a soft smile, hinting at being content or just neutrally happy.

Surprised

In [ ]:

```
# Write your code to visualize images from the class 'surprise'.

expression = 'surprise'  # Set the class to 'surprise'

plt.figure(figsize=(8, 8))
for i in range(1, 10):
    plt.subplot(3, 3, i)
    img_path = folder_path + "train/" + expression + "/" + os.listdir(folder_path + "train/" + expression)[i]
    img = load_img(img_path, target_size=(picture_size, picture_size))  # Load the image
    plt.imshow(img)
    plt.axis('off')
plt.tight_layout()
plt.show()
```



Observations and Insights:

Our lineup shows classic shock vibes across the board—raised brows, wide eyes, and open mouths all around, signaling surprise clear as day. From the playful to the downright astonished, each expression packs a punch of

signaling surprise clear as day. From the playful to the downright astonished, each expression packs a punch of surprise. A kid with eyes popping and another with a hand clamped over the mouth, all scream "I can't believe it!" Even with shades on or in a blurry shot, the surprise factor is unmistakable. Key giveaway? Those eyebrows shooting up, eyes going wide to catch every bit of the action, and mouths dropping open, maybe even to catch a breath. And let's not overlook those spontaneous hand moves, either covering a gasp or framing a shocked face.

Checking Distribution of Classes

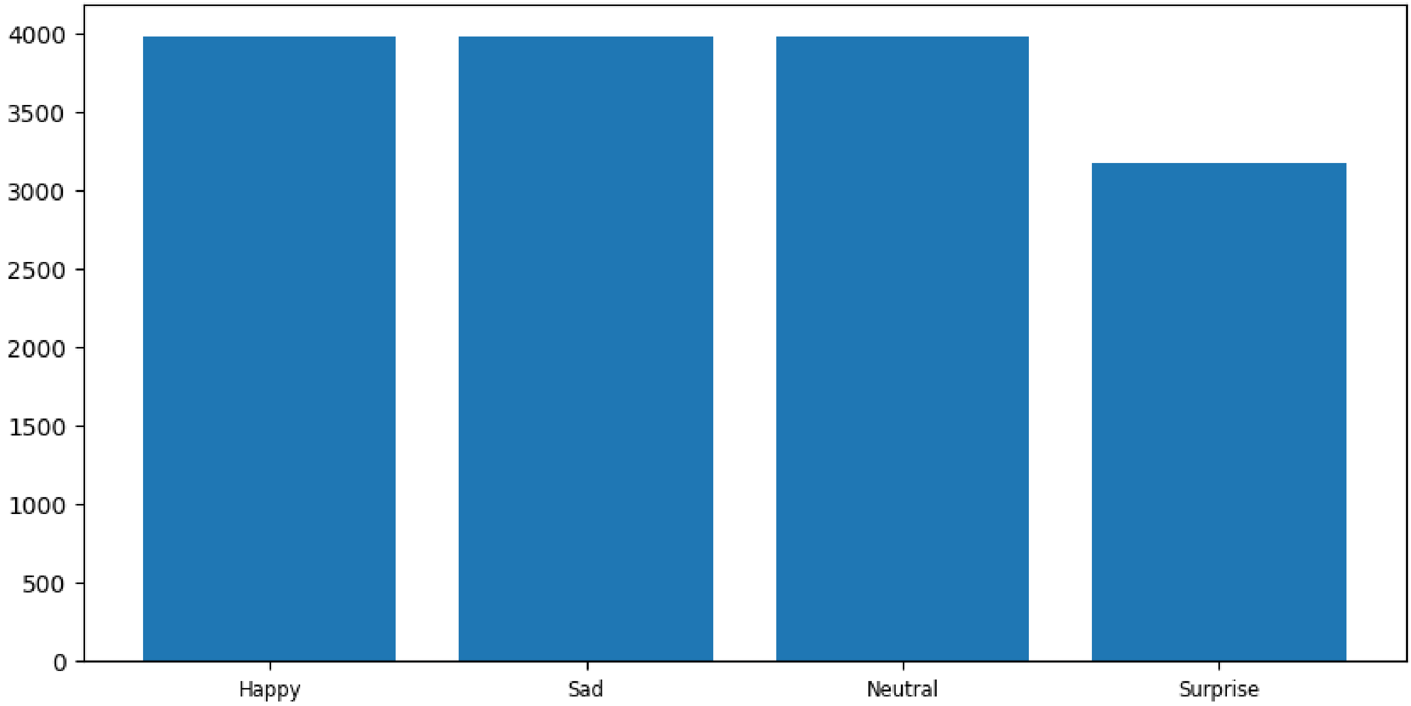
In [ ]:

```
# Getting count of images in each folder within our training path
num_happy = len(os.listdir(folder_path + "train/happy"))
print("Number of images in the class 'happy':    ", num_happy)
num_sad = len(os.listdir(folder_path + "train/sad"))
print("Number of images in the class 'sad':      ", num_sad)
num_neutral = len(os.listdir(folder_path + "train/neutral"))
print("Number of images in the class 'neutral': ", num_neutral)
num_surprise = num_surprise = len(os.listdir(folder_path + "train/surprise"))
print("Number of images in the class 'surprise':", num_surprise)
```

```
Number of images in the class 'happy':    3976
Number of images in the class 'sad':      3982
Number of images in the class 'neutral':  3978
Number of images in the class 'surprise': 3173
```

In [ ]:

```
# Code to plot histogram
plt.figure(figsize = (10, 5))

data = {'Happy': num_happy, 'Sad': num_sad, 'Neutral': num_neutral, 'Surprise' : num_sur
prise}

df = pd.Series(data)

plt.bar(range(len(df)), df.values, align = 'center')

plt.xticks(range(len(df)), df.index.values, size = 'small')

plt.show()
```



Observations and Insights:

Our dataset's looking pretty even with 'happy', 'sad', and 'neutral' classes each rocking around 4,000 images.

That's great news for avoiding bias and aiming for a model that doesn't play favorites. But here's the twist: our 'surprise' category's lagging with only 3173 pics, missing out on about 800 friends to be on par with the rest.

Here's the drill to level the playing field:

Spot Check: Keep a keen eye on how our model's doing with 'surprise'. We want it nailing this emotion just as well as the others. Pump Up the Volume: Crank up those 'surprise' image numbers with some clever data augmentation tricks. Mixing It Up: Think about reshuffling the deck. Maybe give 'surprise' more reps by oversampling or dial back on the other classes. Weightlifting: In training, tipping the scales with class weights can make each 'surprise' sighting hit harder and count for more. Deep Dive: Worth a peek to see if 'surprise' is just trickier or more varied, which might explain the learning curve. Gathering More: If we can, let's scout for more 'surprise' shots to beef up its lineup. While we're not in a dire imbalance crisis, balancing our dataset's key for a fair fight across all emotions during training and validation.

Think About It:

- Are the classes equally distributed? If not, do you think the imbalance is too high? Will it be a problem as we progress?
- Are there any Exploratory Data Analysis tasks that we can do here? Would they provide any meaningful insights?

Creating our Data Loaders

In this section, we are creating data loaders that we will use as inputs to our Neural Network. A sample of the required code has been given with respect to the training data. Please create the data loaders for validation and test set accordingly.

You have two options for the color_mode. You can set it to color_mode = 'rgb' or color_mode = 'grayscale'. You will need to try out both and see for yourself which one gives better performance.

In [ ]:

```
batch_size  = 32
img_size = 48

datagen_train = ImageDataGenerator(horizontal_flip = True,
                                    brightness_range=(0.,2.),
                                    rescale=1./255,
                                    shear_range=0.3)

train_set = datagen_train.flow_from_directory(folder_path + "train",
                                              target_size = (img_size, img_size),
                                              color_mode = 'grayscale' ,
                                              batch_size = batch_size,
                                              class_mode = 'categorical',
                                              shuffle = True)


datagen_validation = ImageDataGenerator(rescale=1./255)

validation_set = datagen_validation.flow_from_directory(
    folder_path + "validation",
    target_size=(img_size, img_size),
    color_mode='grayscale',
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True)

datagen_test = ImageDataGenerator(rescale=1./255)

test_set = datagen_test.flow_from_directory(
    folder_path + "test",
    target_size=(img_size, img_size),
    color_mode='grayscale',
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False)
```

```
Found 15109 images belonging to 4 classes.
Found 4977 images belonging to 4 classes.
Found 128 images belonging to 4 classes.
```

Model Building

Think About It:

- Are Convolutional Neural Networks the right approach? Should we have gone with Artificial Neural Networks instead?
- What are the advantages of CNNs over ANNs and are they applicable here?

Creating the Base Neural Network

Our Base Neural network will be a fairly simple model architecture.

- We want our Base Neural Network architecture to have 3 convolutional blocks.
- Each convolutional block must contain one Conv2D layer followed by a maxpooling layer and one Dropout layer. We can play around with the dropout ratio.
- Add first Conv2D layer with **64 filters** and a **kernel size of 2**. Use the 'same' padding and provide the **input_shape = (48, 48, 3) if you are using 'rgb' color mode in your dataloader or else input shape = (48, 48, 1) if you're using 'grayscale' colormode. Use 'relu' activation.**
- Add MaxPooling2D layer with **pool size = 2.**
- Add a Dropout layer with a dropout ratio of 0.2.
- Add a second Conv2D layer with **32 filters** and a **kernel size of 2**. Use the **'same' padding** and **'relu' activation.**
- Follow this up with a similar Maxpooling2D layer like above and a Dropout layer with 0.2 Dropout ratio to complete your second Convolutional Block.
- Add a third Conv2D layer with **32 filters** and a **kernel size of 2**. Use the **'same' padding** and **'relu' activation.** Once again, follow it up with a Maxpooling2D layer and a Dropout layer to complete your third Convolutional block.
- After adding your convolutional blocks, add your Flatten layer.
- Add your first Dense layer with **512 neurons. Use 'relu' activation function.**
- Add a Dropout layer with dropout ratio of 0.4.
- Add your final Dense Layer with 4 neurons and **'softmax' activation function**
- Print your model summary

In [ ]:

```
# Initializing a Sequential Model
model1 = Sequential()

input_shape = (48, 48, 1)

# Add the first Convolutional block
model1.add(Conv2D(64, (2, 2), padding='same', activation='relu', input_shape=input_shape
))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.2))

# Add the second Convolutional block
model1.add(Conv2D(32, (2, 2), padding='same', activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.2))

# Add the third Convolutional block
model1.add(Conv2D(32, (2, 2), padding='same', activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.2))
```

```
# Add the Flatten layer
model1.add(Flatten())

# Add the first Dense layer
model1.add(Dense(512, activation='relu'))
model1.add(Dropout(0.4))

# Add the Final layer
model1.add(Dense(4, activation='softmax'))

model1.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_3 (Conv2D)           (None, 48, 48, 64)        320       
                                                                 
 max_pooling2d_3 (MaxPoolin  (None, 24, 24, 64)        0         
 g2D)                                                            
                                                                 
 dropout_4 (Dropout)         (None, 24, 24, 64)        0         
                                                                 
 conv2d_4 (Conv2D)           (None, 24, 24, 32)        8224      
                                                                 
 max_pooling2d_4 (MaxPoolin  (None, 12, 12, 32)        0         
 g2D)                                                            
                                                                 
 dropout_5 (Dropout)         (None, 12, 12, 32)        0         
                                                                 
 conv2d_5 (Conv2D)           (None, 12, 12, 32)        4128      
                                                                 
 max_pooling2d_5 (MaxPoolin  (None, 6, 6, 32)          0         
 g2D)                                                            
                                                                 
 dropout_6 (Dropout)         (None, 6, 6, 32)          0         
                                                                 
 flatten_1 (Flatten)         (None, 1152)              0         
                                                                 
 dense_2 (Dense)             (None, 512)               590336    
                                                                 
 dropout_7 (Dropout)         (None, 512)               0         
                                                                 
 dense_3 (Dense)             (None, 4)                 2052      
                                                                 
=================================================================
Total params: 605060 (2.31 MB)
Trainable params: 605060 (2.31 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Compiling and Training the Model

In [ ]:

```
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

checkpoint = ModelCheckpoint("./model1.h5", monitor='val_acc', verbose=1, save_best_only
=True, mode='max')

early_stopping = EarlyStopping(monitor = 'val_loss',
                          min_delta = 0,
                          patience = 3,
                          verbose = 1,
                          restore_best_weights = True
                          )

reduce_learningrate = ReduceLROnPlateau(monitor = 'val_loss',
                              factor = 0.2,
                              patience = 3,
                              verbose = 1,
                              min_delta = 0.0001)

callbacks_list = [early_stopping, checkpoint, reduce_learningrate]

epochs = 20
```

In [ ]:

```
# Write your code to compile your model1. Use categorical crossentropy as your loss funct
ion, Adam Optimizer with 0.001 learning rate, and set your metrics to 'accuracy'.
model1.compile(
    loss='categorical_crossentropy',
    optimizer=Adam(learning_rate=0.001),
    metrics=['accuracy']
)
```

In [ ]:

```
# Write your code to fit your model1. Use train_set as your training data and validation_
set as your validation data. Train your model for 20 epochs.

history = model1.fit(
    train_set,
    steps_per_epoch=train_set.samples // batch_size,
    epochs=epochs,
    validation_data=validation_set,
    validation_steps=validation_set.samples // batch_size,
    verbose=1
)
```

```
Epoch 1/20
472/472 [==============================] - 84s 173ms/step - loss: 1.3597 - accuracy: 0.30
19 - val_loss: 1.2109 - val_accuracy: 0.4863
Epoch 2/20
472/472 [==============================] - 76s 161ms/step - loss: 1.2291 - accuracy: 0.44
56 - val_loss: 1.1080 - val_accuracy: 0.5337
Epoch 3/20
472/472 [==============================] - 76s 161ms/step - loss: 1.1508 - accuracy: 0.49
54 - val_loss: 1.0396 - val_accuracy: 0.5593
Epoch 4/20
472/472 [==============================] - 76s 161ms/step - loss: 1.1069 - accuracy: 0.51
24 - val_loss: 0.9912 - val_accuracy: 0.5873
Epoch 5/20
472/472 [==============================] - 76s 160ms/step - loss: 1.0581 - accuracy: 0.53
88 - val_loss: 0.9681 - val_accuracy: 0.5917
Epoch 6/20
472/472 [==============================] - 76s 160ms/step - loss: 1.0327 - accuracy: 0.54
81 - val_loss: 0.9209 - val_accuracy: 0.6192
Epoch 7/20
472/472 [==============================] - 76s 160ms/step - loss: 1.0091 - accuracy: 0.56
22 - val_loss: 0.9129 - val_accuracy: 0.6282
Epoch 8/20
472/472 [==============================] - 75s 158ms/step - loss: 0.9902 - accuracy: 0.57
88 - val_loss: 0.8525 - val_accuracy: 0.6450
Epoch 9/20
472/472 [==============================] - 76s 161ms/step - loss: 0.9628 - accuracy: 0.58
53 - val_loss: 0.8672 - val_accuracy: 0.6379
Epoch 10/20
472/472 [==============================] - 77s 162ms/step - loss: 0.9591 - accuracy: 0.59
10 - val_loss: 0.8863 - val_accuracy: 0.6369
Epoch 11/20
472/472 [==============================] - 76s 161ms/step - loss: 0.9427 - accuracy: 0.59
89 - val_loss: 0.8401 - val_accuracy: 0.6573
Epoch 12/20
472/472 [==============================] - 76s 160ms/step - loss: 0.9329 - accuracy: 0.59
91 - val_loss: 0.8226 - val_accuracy: 0.6613
Epoch 13/20
472/472 [==============================] - 76s 160ms/step - loss: 0.9277 - accuracy: 0.60
34 - val_loss: 0.8260 - val_accuracy: 0.6633
Epoch 14/20
```

```
472/472 [==============================] - 77s 164ms/step - loss: 0.9120 - accuracy: 0.61
57 - val_loss: 0.8013 - val_accuracy: 0.6746
Epoch 15/20
472/472 [==============================] - 76s 160ms/step - loss: 0.9083 - accuracy: 0.61
36 - val_loss: 0.7944 - val_accuracy: 0.6726
Epoch 16/20
472/472 [==============================] - 81s 171ms/step - loss: 0.8987 - accuracy: 0.61
72 - val_loss: 0.7908 - val_accuracy: 0.6831
Epoch 17/20
472/472 [==============================] - 76s 162ms/step - loss: 0.8888 - accuracy: 0.62
45 - val_loss: 0.7872 - val_accuracy: 0.6780
Epoch 18/20
472/472 [==============================] - 76s 160ms/step - loss: 0.8832 - accuracy: 0.63
13 - val_loss: 0.7827 - val_accuracy: 0.6831
Epoch 19/20
472/472 [==============================] - 80s 169ms/step - loss: 0.8696 - accuracy: 0.63
21 - val_loss: 0.7797 - val_accuracy: 0.6812
Epoch 20/20
472/472 [==============================] - 76s 161ms/step - loss: 0.8652 - accuracy: 0.63
31 - val_loss: 0.7947 - val_accuracy: 0.6716
```

Evaluating the Model on the Test Set

In [ ]:

```python
# Write your code to evaluate your model on test data.

# Evaluate the model on the test data
test_loss, test_accuracy = model1.evaluate(test_set, steps=test_set.samples // batch_size
)

# Print the results
print("Test loss:", test_loss)
print("Test accuracy:", test_accuracy)
```

```
4/4 [==============================] - 0s 56ms/step - loss: 0.8009 - accuracy: 0.6641
Test loss: 0.8009257316589355
Test accuracy: 0.6640625
```

Observations and Insights:

The model's doing alright, picking up some patterns from the training data, but it's still tripping up on a good chunk of the test set. This might mean our setup's too simple to catch all the data's subtle vibes or it's just memorizing training data and flopping when faced with new stuff. And with the loss not looking too hot, we've got our work cut out to dial that down, directly boosting accuracy.

Here's the scoop on boosting our game:

Data's Deep Cuts: We're dealing with emotions here, and they're tricky. A beefier model might just catch those fine lines we're missing. Keeping It Real: If we're acing the training but bombing the test, smells like overfitting. Time to mix in some dropout, get creative with data augmentation, and maybe tighten up with regularization. Shake Up the Data: Data augmentation's our secret sauce for teaching the model about variety. Different twists might just make it click. Tweak Mode: Nudging around the learning rate, batch size, or how deep our network goes could be game-changers. It's all about finding that sweet spot. Borrowing Brains: Leaning on a pre-trained model might just give us the edge we need, tapping into a treasure trove of learned features. Test the Waters: A test set that really mirrors the wild world out there ensures we're not just training a one-trick pony. Learn on the Fly: For data that keeps changing, setting up a way for our model to learn as it goes can keep it sharp. Feedback is Gold: When users are in the loop, their two cents can help us fine-tune and keep the model on its toes.

Creating the second Convolutional Neural Network

In the second Neural network, we will add a few more Convolutional blocks. We will also use Batch Normalization layers.

- **This time, each Convolutional block will have 1 Conv2D layer, followed by a BatchNormalization, LeakuRelU,** and a MaxPooling2D layer. We are not adding any Dropout layer this time.

- Add first Conv2D layer with **256 filters** and a **kernel size of 2**. Use the 'same' padding and provide the **input_shape = (48, 48, 3) if you are using 'rgb' color mode in your dataloader or else input shape = (48, 48, 1) if you're using 'grayscale' colormode. Use 'relu' activation.**
- Add your BatchNormalization layer followed by a LeakyRelU layer with Leaky ReLU parameter of **0.1**
- Add MaxPooling2D layer with **pool size = 2.**
- Add a second Conv2D layer with **128 filters** and a **kernel size of 2**. Use the **'same' padding** and **'relu' activation.**
- Follow this up with a similar BatchNormalization, LeakyRelU, and Maxpooling2D layer like above to complete your second Convolutional Block.
- Add a third Conv2D layer with **64 filters** and a **kernel size of 2**. Use the **'same' padding** and **'relu' activation.** Once again, follow it up with a BatchNormalization, LeakyRelU, and Maxpooling2D layer to complete your third Convolutional block.
- Add a fourth block, with the Conv2D layer having **32 filters.**
- After adding your convolutional blocks, add your Flatten layer.
- Add your first Dense layer with **512 neurons**. Use **'relu' activation function.**
- Add the second Dense Layer with **128 neurons** and use **'relu' activation** function.
- Add your final Dense Layer with 4 neurons and **'softmax' activation function**
- Print your model summary

In [ ]:

```python
# Creating sequential model
model2 = Sequential()

input_shape = (48, 48, 1)

# Add the first Convolutional block
model2.add(Conv2D(256, (2, 2), padding='same', activation='relu', input_shape=input_shap
e))
model2.add(BatchNormalization())
model2.add(LeakyReLU(alpha=0.1))
model2.add(MaxPooling2D(pool_size=(2, 2)))

# Add the second Convolutional block
model2.add(Conv2D(128, (2, 2), padding='same', activation='relu'))
model2.add(BatchNormalization())
model2.add(LeakyReLU(alpha=0.1))
model2.add(MaxPooling2D(pool_size=(2, 2)))

# Add the third Convolutional block
model2.add(Conv2D(64, (2, 2), padding='same', activation='relu'))
model2.add(BatchNormalization())
model2.add(LeakyReLU(alpha=0.1))
model2.add(MaxPooling2D(pool_size=(2, 2)))

# Add the fourth Convolutional block
model2.add(Conv2D(32, (2, 2), padding='same', activation='relu'))
model2.add(BatchNormalization())
model2.add(LeakyReLU(alpha=0.1))
model2.add(MaxPooling2D(pool_size=(2, 2)))

# Add the Flatten layer
model2.add(Flatten())

# Adding the Dense layers
model2.add(Dense(512, activation='relu'))
model2.add(Dense(128, activation='relu'))
model2.add(Dense(4, activation='softmax'))

model2.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 256)       1280
 batch_normalization (Batch  (None, 48, 48, 256)       1024
 Normalization)

 leaky_re_lu (LeakyReLU)     (None, 48, 48, 256)       0

 max_pooling2d (MaxPooling2  (None, 24, 24, 256)       0
 D)

 conv2d_1 (Conv2D)           (None, 24, 24, 128)       131200

 batch_normalization_1 (Bat  (None, 24, 24, 128)       512
 chNormalization)

 leaky_re_lu_1 (LeakyReLU)   (None, 24, 24, 128)       0

 max_pooling2d_1 (MaxPoolin  (None, 12, 12, 128)       0
 g2D)

 conv2d_2 (Conv2D)           (None, 12, 12, 64)        32832

 batch_normalization_2 (Bat  (None, 12, 12, 64)        256
 chNormalization)

 leaky_re_lu_2 (LeakyReLU)   (None, 12, 12, 64)        0

 max_pooling2d_2 (MaxPoolin  (None, 6, 6, 64)          0
 g2D)

 conv2d_3 (Conv2D)           (None, 6, 6, 32)          8224

 batch_normalization_3 (Bat  (None, 6, 6, 32)          128
 chNormalization)

 leaky_re_lu_3 (LeakyReLU)   (None, 6, 6, 32)          0

 max_pooling2d_3 (MaxPoolin  (None, 3, 3, 32)          0
 g2D)

 flatten (Flatten)           (None, 288)               0

 dense (Dense)               (None, 512)               147968

 dense_1 (Dense)             (None, 128)               65664

 dense_2 (Dense)             (None, 4)                 516

=================================================================
Total params: 389604 (1.49 MB)
Trainable params: 388644 (1.48 MB)
Non-trainable params: 960 (3.75 KB)
_________________________________________________________________
```

Compiling and Training the Model

Hint: Take reference from the code we used in the previous model for Compiling and Training the Model.

In [ ]:

```
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

checkpoint = ModelCheckpoint("./model2.h5", monitor='val_loss', verbose = 1, save_best_o
nly = True, mode = 'max')

early_stopping = EarlyStopping(monitor='val_loss', min_delta=0, patience=5, verbose=1, m
ode='min', restore_best_weights=True)
reduce_learningrate = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, verb
ose=1, mode='min', min_delta=0.0001, min_lr=0)

callbacks_list = [early_stopping, checkpoint, reduce_learningrate]
```

```
epochs = 20
```

In [ ]:

```
# Write your code to compile your model2. Use categorical crossentropy as the loss functi
on, Adam Optimizer with 0.001 learning rate, and set metrics as 'accuracy'.

model2.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

In [ ]:

```
# Write your code to fit your model2. Use train_set as the training data and validation_s
et as the validation data. Train your model for 20 epochs.

history = model2.fit(
    train_set,
    validation_data=validation_set,
    epochs=20,
    steps_per_epoch=train_set.samples // batch_size,
    validation_steps=validation_set.samples // batch_size,
    verbose=1
)
```

```
Epoch 1/20
472/472 [==============================] - 624s 1s/step - loss: 1.2777 - accuracy: 0.3960
- val_loss: 1.3052 - val_accuracy: 0.3387
Epoch 2/20
472/472 [==============================] - 569s 1s/step - loss: 1.1067 - accuracy: 0.4961
- val_loss: 1.1634 - val_accuracy: 0.4879
Epoch 3/20
472/472 [==============================] - 558s 1s/step - loss: 0.9725 - accuracy: 0.5772
- val_loss: 0.8865 - val_accuracy: 0.6343
Epoch 4/20
472/472 [==============================] - 554s 1s/step - loss: 0.9016 - accuracy: 0.6071
- val_loss: 1.0073 - val_accuracy: 0.5546
Epoch 5/20
472/472 [==============================] - 593s 1s/step - loss: 0.8548 - accuracy: 0.6324
- val_loss: 0.8039 - val_accuracy: 0.6700
Epoch 6/20
472/472 [==============================] - 564s 1s/step - loss: 0.8145 - accuracy: 0.6547
- val_loss: 0.8110 - val_accuracy: 0.6740
Epoch 7/20
472/472 [==============================] - 559s 1s/step - loss: 0.7851 - accuracy: 0.6678
- val_loss: 0.7600 - val_accuracy: 0.6960
Epoch 8/20
472/472 [==============================] - 562s 1s/step - loss: 0.7610 - accuracy: 0.6770
- val_loss: 0.7482 - val_accuracy: 0.6990
Epoch 9/20
472/472 [==============================] - 549s 1s/step - loss: 0.7377 - accuracy: 0.6893
- val_loss: 0.7861 - val_accuracy: 0.6772
Epoch 10/20
472/472 [==============================] - 553s 1s/step - loss: 0.7176 - accuracy: 0.6981
- val_loss: 0.7579 - val_accuracy: 0.6954
Epoch 11/20
472/472 [==============================] - 584s 1s/step - loss: 0.6939 - accuracy: 0.7080
- val_loss: 0.7330 - val_accuracy: 0.7121
Epoch 12/20
472/472 [==============================] - 581s 1s/step - loss: 0.6829 - accuracy: 0.7171
- val_loss: 0.7524 - val_accuracy: 0.7044
Epoch 13/20
472/472 [==============================] - 637s 1s/step - loss: 0.6637 - accuracy: 0.7241
- val_loss: 0.7088 - val_accuracy: 0.7181
Epoch 14/20
472/472 [==============================] - 568s 1s/step - loss: 0.6540 - accuracy: 0.7267
- val_loss: 0.7256 - val_accuracy: 0.7109
Epoch 15/20
472/472 [==============================] - 547s 1s/step - loss: 0.6403 - accuracy: 0.7340
- val_loss: 0.7665 - val_accuracy: 0.7032
```

```
Epoch 16/20
472/472 [==============================] - 537s 1s/step - loss: 0.6176 - accuracy: 0.7439
- val_loss: 0.7480 - val_accuracy: 0.7099
Epoch 17/20
472/472 [==============================] - 577s 1s/step - loss: 0.6007 - accuracy: 0.7504
- val_loss: 0.8234 - val_accuracy: 0.6897
Epoch 18/20
472/472 [==============================] - 619s 1s/step - loss: 0.5939 - accuracy: 0.7584
- val_loss: 0.8005 - val_accuracy: 0.6917
Epoch 19/20
472/472 [==============================] - 620s 1s/step - loss: 0.5754 - accuracy: 0.7594
- val_loss: 0.7739 - val_accuracy: 0.6978
Epoch 20/20
472/472 [==============================] - 570s 1s/step - loss: 0.5669 - accuracy: 0.7660
- val_loss: 0.7420 - val_accuracy: 0.7220
```

Evaluating the Model on the Test Set

In [ ]:

```
# Write your code to evaluate model's test performance
test_loss, test_accuracy = model2.evaluate(test_set, steps=test_set.samples // batch_size
)

# Print out the results
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
4/4 [==============================] - 2s 291ms/step - loss: 1.3853 - accuracy: 0.2500
Test Loss: 1.3853180408477783
Test Accuracy: 0.25
```

Observations and Insights: With a loss high and accuracy at a mere 25%, our model's basically shooting in the dark, mirroring what you'd expect from random guesses in a 4-way split. This screams potential overfitting if it was a champ on training data but flopped here, or simply, it hasn't quite cracked the code on learning from our training set. Could be due to its setup being too basic, not enough training time, or maybe our data prep missed the mark.

Quick Fixes to Try:

Amp Up Data Prep: Let's double-check our augmentation and make sure our preprocessing aligns with what the model needs. Model Makeover: Might be time to beef up the architecture or borrow brains from a pre-trained model. Hyperparameter Tweaking: Playing around with learning rates, batch sizes, and optimizers could uncover a winning combo. Bring in Backup: Integrating dropout or other smart regularization could prevent the model from getting too clingy with the training data. Balance the Scales: A skewed dataset could throw things off. Balancing it might just do the trick. Training Time: More epochs could help unless we're overfitting, then it's time for early stopping. Smart Learning Rate Adjustments: A scheduler to tweak the learning rate as we go might offer a smoother learning curve. Cross-Validation: To really trust our model's chops, cross-validation can give us a clearer picture of its true performance.

Think About It:

- **Did the models have a satisfactory performance? If not, then what are the possible reasons?**
- **Which Color mode showed better overall performance? What are the possible reasons? Do you think having 'rgb' color mode is needed because the images are already black and white?**

Transfer Learning Architectures

In this section, we will create several Transfer Learning architectures. For the pre-trained models, we will select three popular architectures namely, VGG16, ResNet v2, and Efficient Net. The difference between these architectures and the previous architectures is that these will require 3 input channels while the earlier ones worked on 'grayscale' images. Therefore, we need to create new DataLoaders.

Creating our Data Loaders for Transfer Learning Architectures

In this section, we are creating data loaders that we will use as inputs to our Neural Network. We will have to go with color_mode = 'rgb' as this is the required format for the transfer learning architectures.

In [ ]:

```
batch_size  = 32
img_size = 48

datagen_train = ImageDataGenerator(horizontal_flip = True,
                                    brightness_range = (0., 2.),
                                    rescale = 1./255,
                                    shear_range = 0.3)

train_set = datagen_train.flow_from_directory(folder_path + "train",
                                              target_size = (img_size, img_size),
                                              color_mode = 'rgb',
                                              batch_size = batch_size,
                                              class_mode = 'categorical',
                                              classes = ['happy', 'sad', 'neutral', 'su
rprise'],
                                              shuffle = True)

datagen_validation = ImageDataGenerator(rescale=1./255)

validation_set = datagen_validation.flow_from_directory(
    folder_path + "validation",
    target_size=(img_size, img_size),
    color_mode='rgb',
    batch_size=batch_size,
    class_mode='categorical',
    classes=['happy', 'sad', 'neutral', 'surprise'],
    shuffle=True)

datagen_test = ImageDataGenerator(rescale=1./255)
test_set = datagen_test.flow_from_directory(
    folder_path + "test",
    target_size=(img_size, img_size),
    color_mode='rgb',
    batch_size=batch_size,
    class_mode='categorical',
    classes=['happy', 'sad', 'neutral', 'surprise'],
    shuffle=False)
```

```
Found 15109 images belonging to 4 classes.
Found 4977 images belonging to 4 classes.
Found 128 images belonging to 4 classes.
```

VGG16 Model

Importing the VGG16 Architecture

In [ ]:

```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras import Model

vgg = VGG16(include_top = False, weights = 'imagenet', input_shape = (48, 48, 3))
vgg.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/
vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 0s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
```

```
=================================================================
 input_1 (InputLayer)        [(None, 48, 48, 3)]       0

 block1_conv1 (Conv2D)       (None, 48, 48, 64)        1792

 block1_conv2 (Conv2D)       (None, 48, 48, 64)        36928

 block1_pool (MaxPooling2D)  (None, 24, 24, 64)        0

 block2_conv1 (Conv2D)       (None, 24, 24, 128)       73856

 block2_conv2 (Conv2D)       (None, 24, 24, 128)       147584

 block2_pool (MaxPooling2D)  (None, 12, 12, 128)       0

 block3_conv1 (Conv2D)       (None, 12, 12, 256)       295168

 block3_conv2 (Conv2D)       (None, 12, 12, 256)       590080

 block3_conv3 (Conv2D)       (None, 12, 12, 256)       590080

 block3_pool (MaxPooling2D)  (None, 6, 6, 256)         0

 block4_conv1 (Conv2D)       (None, 6, 6, 512)         1180160

 block4_conv2 (Conv2D)       (None, 6, 6, 512)         2359808

 block4_conv3 (Conv2D)       (None, 6, 6, 512)         2359808

 block4_pool (MaxPooling2D)  (None, 3, 3, 512)         0

 block5_conv1 (Conv2D)       (None, 3, 3, 512)         2359808

 block5_conv2 (Conv2D)       (None, 3, 3, 512)         2359808

 block5_conv3 (Conv2D)       (None, 3, 3, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 1, 1, 512)         0

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 14714688 (56.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Model Building

- **In this model, we will import till the 'block5_pool' layer of the VGG16 model. You can scroll down in the model summary and look for 'block5_pool'. You can choose any other layer as well.**
- **Then we will add a Flatten layer, which receives the output of the 'block5_pool' layer as its input.**
- **We will add a few Dense layers and use 'relu' activation function on them.**
- **You may use Dropout and BatchNormalization layers as well.**
- **Then we will add our last dense layer, which must have 4 neurons and a 'softmax' activation function.**

In [ ]:

```
vgg = VGG16(weights='imagenet', include_top=False, input_shape=(48, 48, 3))
transfer_layer = vgg.get_layer('block5_pool')
vgg.trainable = False

# Flattenning the output from the 3rd block of the VGG16 model
x = Flatten()(transfer_layer.output)

# Adding a Dense layer with 256 neurons
x = Dense(256, activation = 'relu')(x)

# Add a Dense Layer with 128 neurons
x = Dense(128, activation='relu')(x)
```

```python
# Add a DropOut layer with Drop out ratio of 0.3
x = Dropout(0.3)(x)

# Add a Dense Layer with 64 neurons
x = Dense(64, activation='relu')(x)

# Add a Batch Normalization layer
x = BatchNormalization()(x)

# Adding the final dense layer with 4 neurons and use 'softmax' activation
pred = Dense(4, activation='softmax')(x)

vggmodel = Model(vgg.input, pred) # Initializing the model
vggmodel.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_2 (InputLayer)        [(None, 48, 48, 3)]       0         
                                                                 
 block1_conv1 (Conv2D)       (None, 48, 48, 64)        1792      
                                                                 
 block1_conv2 (Conv2D)       (None, 48, 48, 64)        36928     
                                                                 
 block1_pool (MaxPooling2D)  (None, 24, 24, 64)        0         
                                                                 
 block2_conv1 (Conv2D)       (None, 24, 24, 128)       73856     
                                                                 
 block2_conv2 (Conv2D)       (None, 24, 24, 128)       147584    
                                                                 
 block2_pool (MaxPooling2D)  (None, 12, 12, 128)       0         
                                                                 
 block3_conv1 (Conv2D)       (None, 12, 12, 256)       295168    
                                                                 
 block3_conv2 (Conv2D)       (None, 12, 12, 256)       590080    
                                                                 
 block3_conv3 (Conv2D)       (None, 12, 12, 256)       590080    
                                                                 
 block3_pool (MaxPooling2D)  (None, 6, 6, 256)         0         
                                                                 
 block4_conv1 (Conv2D)       (None, 6, 6, 512)         1180160   
                                                                 
 block4_conv2 (Conv2D)       (None, 6, 6, 512)         2359808   
                                                                 
 block4_conv3 (Conv2D)       (None, 6, 6, 512)         2359808   
                                                                 
 block4_pool (MaxPooling2D)  (None, 3, 3, 512)         0         
                                                                 
 block5_conv1 (Conv2D)       (None, 3, 3, 512)         2359808   
                                                                 
 block5_conv2 (Conv2D)       (None, 3, 3, 512)         2359808   
                                                                 
 block5_conv3 (Conv2D)       (None, 3, 3, 512)         2359808   
                                                                 
 block5_pool (MaxPooling2D)  (None, 1, 1, 512)         0         
                                                                 
 flatten_1 (Flatten)         (None, 512)               0         
                                                                 
 dense_3 (Dense)             (None, 256)               131328    
                                                                 
 dense_4 (Dense)             (None, 128)               32896     
                                                                 
 dropout (Dropout)           (None, 128)               0         
                                                                 
 dense_5 (Dense)             (None, 64)                8256      
                                                                 
 batch_normalization_4 (Bat  (None, 64)                256       
 chNormalization)                                                
                                                                 
 dense_6 (Dense)             (None, 4)                 260       
```

```
=================================================================
Total params: 14887684 (56.79 MB)
Trainable params: 172868 (675.27 KB)
Non-trainable params: 14714816 (56.13 MB)
_________________________________________________________________
```

Compiling and Training the VGG16 Model

In [ ]:

```
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

checkpoint = ModelCheckpoint("./vggmodel.h5", monitor = 'val_loss', verbose = 1, save_be
st_only = True, mode = 'max')

early_stopping = EarlyStopping(monitor = 'val_loss',
                          min_delta = 0,
                          patience = 3,
                          verbose = 1,
                          restore_best_weights = True
                          )

reduce_learningrate = ReduceLROnPlateau(monitor = 'val_loss',
                              factor = 0.2,
                              patience = 3,
                              verbose = 1,
                              min_delta = 0.0001)

callbacks_list = [early_stopping, checkpoint, reduce_learningrate]

epochs = 20
```

In [ ]:

```
# Write your code to compile the vggmodel. Use categorical crossentropy as the loss funct
ion, Adam Optimizer with 0.001 learning rate, and set metrics to 'accuracy'.
vggmodel.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

In [ ]:

```
history = vggmodel.fit(
    train_set,
    validation_data=validation_set,
    epochs=20,
    callbacks=callbacks_list,
    verbose=1
)
```

```
Epoch 1/20
473/473 [==============================] - ETA: 0s - loss: 1.0974 - accuracy: 0.5174
Epoch 1: val_loss did not improve from 1.13104
473/473 [==============================] - 589s 1s/step - loss: 1.0974 - accuracy: 0.5174
- val_loss: 1.0567 - val_accuracy: 0.5489 - lr: 2.0000e-04
Epoch 2/20
473/473 [==============================] - ETA: 0s - loss: 1.0973 - accuracy: 0.5194
Epoch 2: val_loss did not improve from 1.13104
473/473 [==============================] - 575s 1s/step - loss: 1.0973 - accuracy: 0.5194
- val_loss: 1.0419 - val_accuracy: 0.5558 - lr: 2.0000e-04
Epoch 3/20
473/473 [==============================] - ETA: 0s - loss: 1.0863 - accuracy: 0.5288
Epoch 3: val_loss did not improve from 1.13104
473/473 [==============================] - 571s 1s/step - loss: 1.0863 - accuracy: 0.5288
- val_loss: 1.0513 - val_accuracy: 0.5431 - lr: 2.0000e-04
Epoch 4/20
473/473 [==============================] - ETA: 0s - loss: 1.0901 - accuracy: 0.5184
Epoch 4: val_loss did not improve from 1.13104
473/473 [==============================] - 563s 1s/step - loss: 1.0901 - accuracy: 0.5184
```

```
473/473 [==============================] - 563s 1s/step - loss: 1.0901 - accuracy: 0.5184
- val_loss: 1.0530 - val_accuracy: 0.5477 - lr: 2.0000e-04
Epoch 5/20
473/473 [==============================] - ETA: 0s - loss: 1.0745 - accuracy: 0.5303Resto
ring model weights from the end of the best epoch: 2.

Epoch 5: val_loss did not improve from 1.13104

Epoch 5: ReduceLROnPlateau reducing learning rate to 4.0000001899898055e-05.
473/473 [==============================] - 570s 1s/step - loss: 1.0745 - accuracy: 0.5303
- val_loss: 1.0498 - val_accuracy: 0.5499 - lr: 2.0000e-04
Epoch 5: early stopping
```

Evaluating the VGG16 model

In [ ]:

```python
# Write your code to evaluate model performance on the test set
test_loss, test_accuracy = vggmodel.evaluate(test_set, steps=test_set.samples // batch_si
ze)
# Print out the test loss and accuracy
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
4/4 [==============================] - 6s 1s/step - loss: 1.0681 - accuracy: 0.5391
Test Loss: 1.0681047439575195
Test Accuracy: 0.5390625
```

Think About It:

- **What do you infer from the general trend in the training performance?**
- **Is the training accuracy consistently improving?**
- **Is the validation accuracy also improving similarly?**

Observations and Insights:

Seeing the training accuracy tick upwards hints our model's on the right track, soaking up the training data vibes. Yet, without a blow-by-blow of each training round, it's tough to say if we're on a steady climb or hitting snags along the way. If it's not a smooth ride up, we might need to tweak the learning rate, beef up the model, or give it more time to learn.

On the flip side, if validation accuracy's keeping pace, that's a solid sign our model's not just cramming but actually getting smarter. But if it's lagging or hits a wall, we could be staring down the barrel of overfitting or maxing out what our model can learn with what it's got.

Landing a test accuracy around 53.91% tells us there's more work to do. It's better than wild guesses, but we're not exactly acing the test. To level up, consider mixing up the learning rate, adding some dropout, cranking up data quality and variety, or diving into the hyperparameter tweak tank.

And hey, let's not put all our eggs in one basket—exploring different pre-trained models or tweaking our transfer learning strategy could give us the edge we need. Plus, eyeballing other metrics like precision and recall can shine a light on exactly where our model's making the grade or needs a tutor.

```
# This is formatted as code
```

Note: You can even go back and build your own architecture on top of the VGG16 Transfer layer and see if you can improve the performance

ResNet V2 Model

In [ ]:

```python
import tensorflow as tf
import tensorflow.keras.applications as ap
from tensorflow.keras import Model

Resnet = ap.ResNet101(include_top = False, weights = "imagenet", input_shape=(48,48,3))
Resnet.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet
/resnet101_weights_tf_dim_ordering_tf_kernels_notop.h5
171446536/171446536 [==============================] - 2s 0us/step
Model: "resnet101"
__________________________________________________________________________________________
_________
 Layer (type)                Output Shape                 Param #    Connected to

==========================================================================================
=========
 input_3 (InputLayer)        [(None, 48, 48, 3)]          0          []



 conv1_pad (ZeroPadding2D)   (None, 54, 54, 3)            0          ['input_3[0][0]']



 conv1_conv (Conv2D)         (None, 24, 24, 64)           9472       ['conv1_pad[0][0]']



 conv1_bn (BatchNormalizati  (None, 24, 24, 64)           256        ['conv1_conv[0][0]']

 on)



 conv1_relu (Activation)     (None, 24, 24, 64)           0          ['conv1_bn[0][0]']



 pool1_pad (ZeroPadding2D)   (None, 26, 26, 64)           0          ['conv1_relu[0][0]']



 pool1_pool (MaxPooling2D)   (None, 12, 12, 64)           0          ['pool1_pad[0][0]']



 conv2_block1_1_conv (Conv2  (None, 12, 12, 64)           4160       ['pool1_pool[0][0]']

 D)



 conv2_block1_1_bn (BatchNo  (None, 12, 12, 64)           256        ['conv2_block1_1_conv
[0][0]']
 rmalization)



 conv2_block1_1_relu (Activ  (None, 12, 12, 64)           0          ['conv2_block1_1_bn[
0][0]']
 ation)



 conv2_block1_2_conv (Conv2  (None, 12, 12, 64)           36928      ['conv2_block1_1_relu
[0][0]']
 D)



 conv2_block1_2_bn (BatchNo  (None, 12, 12, 64)           256        ['conv2_block1_2_conv
[0][0]']
```

```
 rmalization)



 conv2_block1_2_relu (Activ  (None, 12, 12, 64)          0          ['conv2_block1_2_bn[
0][0]']
 ation)



 conv2_block1_0_conv (Conv2  (None, 12, 12, 256)         16640      ['pool1_pool[0][0]']

 D)



 conv2_block1_3_conv (Conv2  (None, 12, 12, 256)         16640      ['conv2_block1_2_relu
[0][0]']
 D)



 conv2_block1_0_bn (BatchNo  (None, 12, 12, 256)         1024       ['conv2_block1_0_conv
[0][0]']
 rmalization)



 conv2_block1_3_bn (BatchNo  (None, 12, 12, 256)         1024       ['conv2_block1_3_conv
[0][0]']
 rmalization)



 conv2_block1_add (Add)      (None, 12, 12, 256)         0          ['conv2_block1_0_bn[
0][0]',

                                                                     'conv2_block1_3_bn
[0][0]']

 conv2_block1_out (Activati  (None, 12, 12, 256)         0          ['conv2_block1_add[0
][0]']
 on)



 conv2_block2_1_conv (Conv2  (None, 12, 12, 64)          16448      ['conv2_block1_out[0]
[0]']
 D)



 conv2_block2_1_bn (BatchNo  (None, 12, 12, 64)          256        ['conv2_block2_1_conv
[0][0]']
 rmalization)



 conv2_block2_1_relu (Activ  (None, 12, 12, 64)          0          ['conv2_block2_1_bn[
0][0]']
 ation)



 conv2_block2_2_conv (Conv2  (None, 12, 12, 64)          36928      ['conv2_block2_1_relu
[0][0]']
 D)



 conv2_block2_2_bn (BatchNo  (None, 12, 12, 64)          256        ['conv2_block2_2_conv
[0][0]']
```

```
 rmalization)


 conv2_block2_2_relu (Activ  (None, 12, 12, 64)           0         ['conv2_block2_2_bn[
0][0]']
 ation)


 conv2_block2_3_conv (Conv2  (None, 12, 12, 256)          16640     ['conv2_block2_2_relu
[0][0]']
 D)


 conv2_block2_3_bn (BatchNo  (None, 12, 12, 256)          1024      ['conv2_block2_3_conv
[0][0]']
 rmalization)


 conv2_block2_add (Add)      (None, 12, 12, 256)          0         ['conv2_block1_out[0
][0]',

                                                                     'conv2_block2_3_bn
[0][0]']

 conv2_block2_out (Activati  (None, 12, 12, 256)          0         ['conv2_block2_add[0
][0]']
 on)


 conv2_block3_1_conv (Conv2  (None, 12, 12, 64)           16448     ['conv2_block2_out[0]
[0]']
 D)


 conv2_block3_1_bn (BatchNo  (None, 12, 12, 64)           256       ['conv2_block3_1_conv
[0][0]']
 rmalization)


 conv2_block3_1_relu (Activ  (None, 12, 12, 64)           0         ['conv2_block3_1_bn[
0][0]']
 ation)


 conv2_block3_2_conv (Conv2  (None, 12, 12, 64)           36928     ['conv2_block3_1_relu
[0][0]']
 D)


 conv2_block3_2_bn (BatchNo  (None, 12, 12, 64)           256       ['conv2_block3_2_conv
[0][0]']
 rmalization)


 conv2_block3_2_relu (Activ  (None, 12, 12, 64)           0         ['conv2_block3_2_bn[
0][0]']
 ation)


 conv2_block3_3_conv (Conv2  (None, 12, 12, 256)          16640     ['conv2_block3_2_relu
[0][0]']
```

```
 D)


 conv2_block3_3_bn (BatchNo  (None, 12, 12, 256)          1024      ['conv2_block3_3_conv
[0][0]']
 rmalization)


 conv2_block3_add (Add)      (None, 12, 12, 256)          0         ['conv2_block2_out[0
][0]',
                                                                     'conv2_block3_3_bn
[0][0]']

 conv2_block3_out (Activati  (None, 12, 12, 256)          0         ['conv2_block3_add[0
][0]']
 on)


 conv3_block1_1_conv (Conv2  (None, 6, 6, 128)            32896     ['conv2_block3_out[0]
[0]']
 D)


 conv3_block1_1_bn (BatchNo  (None, 6, 6, 128)            512       ['conv3_block1_1_conv
[0][0]']
 rmalization)


 conv3_block1_1_relu (Activ  (None, 6, 6, 128)            0         ['conv3_block1_1_bn[
0][0]']
 ation)


 conv3_block1_2_conv (Conv2  (None, 6, 6, 128)            147584    ['conv3_block1_1_relu
[0][0]']
 D)


 conv3_block1_2_bn (BatchNo  (None, 6, 6, 128)            512       ['conv3_block1_2_conv
[0][0]']
 rmalization)


 conv3_block1_2_relu (Activ  (None, 6, 6, 128)            0         ['conv3_block1_2_bn[
0][0]']
 ation)


 conv3_block1_0_conv (Conv2  (None, 6, 6, 512)            131584    ['conv2_block3_out[0]
[0]']
 D)


 conv3_block1_3_conv (Conv2  (None, 6, 6, 512)            66048     ['conv3_block1_2_relu
[0][0]']
 D)


 conv3_block1_0_bn (BatchNo  (None, 6, 6, 512)            2048      ['conv3_block1_0_conv
[0][0]']
```

```
 rmalization)


 conv3_block1_3_bn (BatchNo  (None, 6, 6, 512)            2048      ['conv3_block1_3_conv
[0][0]']
 rmalization)


 conv3_block1_add (Add)      (None, 6, 6, 512)            0         ['conv3_block1_0_bn[
0][0]',
                                                                     'conv3_block1_3_bn
[0][0]']

 conv3_block1_out (Activati  (None, 6, 6, 512)            0         ['conv3_block1_add[0
][0]']
 on)


 conv3_block2_1_conv (Conv2  (None, 6, 6, 128)            65664     ['conv3_block1_out[0]
[0]']
 D)


 conv3_block2_1_bn (BatchNo  (None, 6, 6, 128)            512       ['conv3_block2_1_conv
[0][0]']
 rmalization)


 conv3_block2_1_relu (Activ  (None, 6, 6, 128)            0         ['conv3_block2_1_bn[
0][0]']
 ation)


 conv3_block2_2_conv (Conv2  (None, 6, 6, 128)            147584    ['conv3_block2_1_relu
[0][0]']
 D)


 conv3_block2_2_bn (BatchNo  (None, 6, 6, 128)            512       ['conv3_block2_2_conv
[0][0]']
 rmalization)


 conv3_block2_2_relu (Activ  (None, 6, 6, 128)            0         ['conv3_block2_2_bn[
0][0]']
 ation)


 conv3_block2_3_conv (Conv2  (None, 6, 6, 512)            66048     ['conv3_block2_2_relu
[0][0]']
 D)


 conv3_block2_3_bn (BatchNo  (None, 6, 6, 512)            2048      ['conv3_block2_3_conv
[0][0]']
 rmalization)


 conv3_block2_add (Add)      (None, 6, 6, 512)            0         ['conv3_block1_out[0
][0]',
                                                                  'conv3_block2_3_bn
[0][0]']

 conv3_block2_out (Activati  (None, 6, 6, 512)            0          ['conv3_block2_add[0
][0]']
 on)



 conv3_block3_1_conv (Conv2  (None, 6, 6, 128)            65664      ['conv3_block2_out[0]
[0]']
 D)



 conv3_block3_1_bn (BatchNo  (None, 6, 6, 128)            512        ['conv3_block3_1_conv
[0][0]']
 rmalization)



 conv3_block3_1_relu (Activ  (None, 6, 6, 128)            0          ['conv3_block3_1_bn[
0][0]']
 ation)



 conv3_block3_2_conv (Conv2  (None, 6, 6, 128)            147584     ['conv3_block3_1_relu
[0][0]']
 D)



 conv3_block3_2_bn (BatchNo  (None, 6, 6, 128)            512        ['conv3_block3_2_conv
[0][0]']
 rmalization)



 conv3_block3_2_relu (Activ  (None, 6, 6, 128)            0          ['conv3_block3_2_bn[
0][0]']
 ation)



 conv3_block3_3_conv (Conv2  (None, 6, 6, 512)            66048      ['conv3_block3_2_relu
[0][0]']
 D)



 conv3_block3_3_bn (BatchNo  (None, 6, 6, 512)            2048       ['conv3_block3_3_conv
[0][0]']
 rmalization)



 conv3_block3_add (Add)      (None, 6, 6, 512)            0          ['conv3_block2_out[0
][0]',

                                                                  'conv3_block3_3_bn
[0][0]']

 conv3_block3_out (Activati  (None, 6, 6, 512)            0          ['conv3_block3_add[0
][0]']
 on)



 conv3_block4_1_conv (Conv2  (None, 6, 6, 128)            65664      ['conv3_block3_out[0]
[0]']
```

```
 D)



 conv3_block4_1_bn (BatchNo  (None, 6, 6, 128)            512       ['conv3_block4_1_conv
[0][0]']
 rmalization)



 conv3_block4_1_relu (Activ  (None, 6, 6, 128)            0         ['conv3_block4_1_bn[
0][0]']
 ation)



 conv3_block4_2_conv (Conv2  (None, 6, 6, 128)            147584    ['conv3_block4_1_relu
[0][0]']
 D)



 conv3_block4_2_bn (BatchNo  (None, 6, 6, 128)            512       ['conv3_block4_2_conv
[0][0]']
 rmalization)



 conv3_block4_2_relu (Activ  (None, 6, 6, 128)            0         ['conv3_block4_2_bn[
0][0]']
 ation)



 conv3_block4_3_conv (Conv2  (None, 6, 6, 512)            66048     ['conv3_block4_2_relu
[0][0]']
 D)



 conv3_block4_3_bn (BatchNo  (None, 6, 6, 512)            2048      ['conv3_block4_3_conv
[0][0]']
 rmalization)



 conv3_block4_add (Add)      (None, 6, 6, 512)            0         ['conv3_block3_out[0
][0]',

                                                                     'conv3_block4_3_bn
[0][0]']


 conv3_block4_out (Activati  (None, 6, 6, 512)            0         ['conv3_block4_add[0
][0]']
 on)



 conv4_block1_1_conv (Conv2  (None, 3, 3, 256)            131328    ['conv3_block4_out[0]
[0]']
 D)



 conv4_block1_1_bn (BatchNo  (None, 3, 3, 256)            1024      ['conv4_block1_1_conv
[0][0]']
 rmalization)



 conv4_block1_1_relu (Activ  (None, 3, 3, 256)            0         ['conv4_block1_1_bn[
0][0]']
```

```
 ation)


 conv4_block1_2_conv (Conv2  (None, 3, 3, 256)            590080    ['conv4_block1_1_relu
[0][0]']
 D)


 conv4_block1_2_bn (BatchNo  (None, 3, 3, 256)            1024      ['conv4_block1_2_conv
[0][0]']
 rmalization)


 conv4_block1_2_relu (Activ  (None, 3, 3, 256)            0         ['conv4_block1_2_bn[
0][0]']
 ation)


 conv4_block1_0_conv (Conv2  (None, 3, 3, 1024)           525312    ['conv3_block4_out[0]
[0]']
 D)


 conv4_block1_3_conv (Conv2  (None, 3, 3, 1024)           263168    ['conv4_block1_2_relu
[0][0]']
 D)


 conv4_block1_0_bn (BatchNo  (None, 3, 3, 1024)           4096      ['conv4_block1_0_conv
[0][0]']
 rmalization)


 conv4_block1_3_bn (BatchNo  (None, 3, 3, 1024)           4096      ['conv4_block1_3_conv
[0][0]']
 rmalization)


 conv4_block1_add (Add)      (None, 3, 3, 1024)           0         ['conv4_block1_0_bn[
0][0]',
                                                                     'conv4_block1_3_bn
[0][0]']

 conv4_block1_out (Activati  (None, 3, 3, 1024)           0         ['conv4_block1_add[0
][0]']
 on)


 conv4_block2_1_conv (Conv2  (None, 3, 3, 256)            262400    ['conv4_block1_out[0]
[0]']
 D)


 conv4_block2_1_bn (BatchNo  (None, 3, 3, 256)            1024      ['conv4_block2_1_conv
[0][0]']
 rmalization)


 conv4_block2_1_relu (Activ  (None, 3, 3, 256)            0         ['conv4_block2_1_bn[
0][0]']
```

```
 ation)


 conv4_block2_2_conv (Conv2  (None, 3, 3, 256)            590080    ['conv4_block2_1_relu
[0][0]']
 D)


 conv4_block2_2_bn (BatchNo  (None, 3, 3, 256)            1024      ['conv4_block2_2_conv
[0][0]']
 rmalization)


 conv4_block2_2_relu (Activ  (None, 3, 3, 256)            0         ['conv4_block2_2_bn[
0][0]']
 ation)


 conv4_block2_3_conv (Conv2  (None, 3, 3, 1024)           263168    ['conv4_block2_2_relu
[0][0]']
 D)


 conv4_block2_3_bn (BatchNo  (None, 3, 3, 1024)           4096      ['conv4_block2_3_conv
[0][0]']
 rmalization)


 conv4_block2_add (Add)      (None, 3, 3, 1024)           0         ['conv4_block1_out[0
][0]',

                                                                     'conv4_block2_3_bn
[0][0]']

 conv4_block2_out (Activati  (None, 3, 3, 1024)           0         ['conv4_block2_add[0
][0]']
 on)


 conv4_block3_1_conv (Conv2  (None, 3, 3, 256)            262400    ['conv4_block2_out[0]
[0]']
 D)


 conv4_block3_1_bn (BatchNo  (None, 3, 3, 256)            1024      ['conv4_block3_1_conv
[0][0]']
 rmalization)


 conv4_block3_1_relu (Activ  (None, 3, 3, 256)            0         ['conv4_block3_1_bn[
0][0]']
 ation)


 conv4_block3_2_conv (Conv2  (None, 3, 3, 256)            590080    ['conv4_block3_1_relu
[0][0]']
 D)


 conv4_block3_2_bn (BatchNo  (None, 3, 3, 256)            1024      ['conv4_block3_2_conv
[0][0]']
```

```
 rmalization)


 conv4_block3_2_relu (Activ  (None, 3, 3, 256)            0          ['conv4_block3_2_bn[
0][0]']
 ation)


 conv4_block3_3_conv (Conv2  (None, 3, 3, 1024)           263168     ['conv4_block3_2_relu
[0][0]']
 D)


 conv4_block3_3_bn (BatchNo  (None, 3, 3, 1024)           4096       ['conv4_block3_3_conv
[0][0]']
 rmalization)


 conv4_block3_add (Add)      (None, 3, 3, 1024)           0          ['conv4_block2_out[0
][0]',
                                                                       'conv4_block3_3_bn
[0][0]']

 conv4_block3_out (Activati  (None, 3, 3, 1024)           0          ['conv4_block3_add[0
][0]']
 on)


 conv4_block4_1_conv (Conv2  (None, 3, 3, 256)            262400     ['conv4_block3_out[0]
[0]']
 D)


 conv4_block4_1_bn (BatchNo  (None, 3, 3, 256)            1024       ['conv4_block4_1_conv
[0][0]']
 rmalization)


 conv4_block4_1_relu (Activ  (None, 3, 3, 256)            0          ['conv4_block4_1_bn[
0][0]']
 ation)


 conv4_block4_2_conv (Conv2  (None, 3, 3, 256)            590080     ['conv4_block4_1_relu
[0][0]']
 D)


 conv4_block4_2_bn (BatchNo  (None, 3, 3, 256)            1024       ['conv4_block4_2_conv
[0][0]']
 rmalization)


 conv4_block4_2_relu (Activ  (None, 3, 3, 256)            0          ['conv4_block4_2_bn[
0][0]']
 ation)


 conv4_block4_3_conv (Conv2  (None, 3, 3, 1024)           263168     ['conv4_block4_2_relu
[0][0]']
```

```
 D)


 conv4_block4_3_bn (BatchNo  (None, 3, 3, 1024)           4096      ['conv4_block4_3_conv
[0][0]']
 rmalization)


 conv4_block4_add (Add)      (None, 3, 3, 1024)           0         ['conv4_block3_out[0
][0]',
                                                                      'conv4_block4_3_bn
[0][0]']

 conv4_block4_out (Activati  (None, 3, 3, 1024)           0         ['conv4_block4_add[0
][0]']
 on)


 conv4_block5_1_conv (Conv2  (None, 3, 3, 256)            262400    ['conv4_block4_out[0]
[0]']
 D)


 conv4_block5_1_bn (BatchNo  (None, 3, 3, 256)            1024      ['conv4_block5_1_conv
[0][0]']
 rmalization)


 conv4_block5_1_relu (Activ  (None, 3, 3, 256)            0         ['conv4_block5_1_bn[
0][0]']
 ation)


 conv4_block5_2_conv (Conv2  (None, 3, 3, 256)            590080    ['conv4_block5_1_relu
[0][0]']
 D)


 conv4_block5_2_bn (BatchNo  (None, 3, 3, 256)            1024      ['conv4_block5_2_conv
[0][0]']
 rmalization)


 conv4_block5_2_relu (Activ  (None, 3, 3, 256)            0         ['conv4_block5_2_bn[
0][0]']
 ation)


 conv4_block5_3_conv (Conv2  (None, 3, 3, 1024)           263168    ['conv4_block5_2_relu
[0][0]']
 D)


 conv4_block5_3_bn (BatchNo  (None, 3, 3, 1024)           4096      ['conv4_block5_3_conv
[0][0]']
 rmalization)


 conv4_block5_add (Add)      (None, 3, 3, 1024)           0         ['conv4_block4_out[0
][0]',
                                                                         'conv4_block5_3_bn
[0][0]']


 conv4_block5_out (Activati  (None, 3, 3, 1024)           0          ['conv4_block5_add[0
][0]']
 on)


 conv4_block6_1_conv (Conv2  (None, 3, 3, 256)            262400     ['conv4_block5_out[0]
[0]']
 D)


 conv4_block6_1_bn (BatchNo  (None, 3, 3, 256)            1024       ['conv4_block6_1_conv
[0][0]']
 rmalization)


 conv4_block6_1_relu (Activ  (None, 3, 3, 256)            0          ['conv4_block6_1_bn[
0][0]']
 ation)


 conv4_block6_2_conv (Conv2  (None, 3, 3, 256)            590080     ['conv4_block6_1_relu
[0][0]']
 D)


 conv4_block6_2_bn (BatchNo  (None, 3, 3, 256)            1024       ['conv4_block6_2_conv
[0][0]']
 rmalization)


 conv4_block6_2_relu (Activ  (None, 3, 3, 256)            0          ['conv4_block6_2_bn[
0][0]']
 ation)


 conv4_block6_3_conv (Conv2  (None, 3, 3, 1024)           263168     ['conv4_block6_2_relu
[0][0]']
 D)


 conv4_block6_3_bn (BatchNo  (None, 3, 3, 1024)           4096       ['conv4_block6_3_conv
[0][0]']
 rmalization)


 conv4_block6_add (Add)      (None, 3, 3, 1024)           0          ['conv4_block5_out[0
][0]',
                                                                      'conv4_block6_3_bn
[0][0]']


 conv4_block6_out (Activati  (None, 3, 3, 1024)           0          ['conv4_block6_add[0
][0]']
 on)


 conv4_block7_1_conv (Conv2  (None, 3, 3, 256)            262400     ['conv4_block6_out[0]
[0]']
```

```
 D)


 conv4_block7_1_bn (BatchNo  (None, 3, 3, 256)            1024        ['conv4_block7_1_conv
[0][0]']
 rmalization)


 conv4_block7_1_relu (Activ  (None, 3, 3, 256)            0           ['conv4_block7_1_bn[
0][0]']
 ation)


 conv4_block7_2_conv (Conv2  (None, 3, 3, 256)            590080      ['conv4_block7_1_relu
[0][0]']
 D)


 conv4_block7_2_bn (BatchNo  (None, 3, 3, 256)            1024        ['conv4_block7_2_conv
[0][0]']
 rmalization)


 conv4_block7_2_relu (Activ  (None, 3, 3, 256)            0           ['conv4_block7_2_bn[
0][0]']
 ation)


 conv4_block7_3_conv (Conv2  (None, 3, 3, 1024)           263168      ['conv4_block7_2_relu
[0][0]']
 D)


 conv4_block7_3_bn (BatchNo  (None, 3, 3, 1024)           4096        ['conv4_block7_3_conv
[0][0]']
 rmalization)


 conv4_block7_add (Add)      (None, 3, 3, 1024)           0           ['conv4_block6_out[0
][0]',

                                                                       'conv4_block7_3_bn
[0][0]']


 conv4_block7_out (Activati  (None, 3, 3, 1024)           0           ['conv4_block7_add[0
][0]']
 on)


 conv4_block8_1_conv (Conv2  (None, 3, 3, 256)            262400      ['conv4_block7_out[0]
[0]']
 D)


 conv4_block8_1_bn (BatchNo  (None, 3, 3, 256)            1024        ['conv4_block8_1_conv
[0][0]']
 rmalization)


 conv4_block8_1_relu (Activ  (None, 3, 3, 256)            0           ['conv4_block8_1_bn[
0][0]']
```

```
 ation)


 conv4_block8_2_conv (Conv2  (None, 3, 3, 256)           590080    ['conv4_block8_1_relu
[0][0]']
 D)


 conv4_block8_2_bn (BatchNo  (None, 3, 3, 256)           1024      ['conv4_block8_2_conv
[0][0]']
 rmalization)


 conv4_block8_2_relu (Activ  (None, 3, 3, 256)           0         ['conv4_block8_2_bn[
0][0]']
 ation)


 conv4_block8_3_conv (Conv2  (None, 3, 3, 1024)          263168    ['conv4_block8_2_relu
[0][0]']
 D)


 conv4_block8_3_bn (BatchNo  (None, 3, 3, 1024)          4096      ['conv4_block8_3_conv
[0][0]']
 rmalization)


 conv4_block8_add (Add)      (None, 3, 3, 1024)          0         ['conv4_block7_out[0
][0]',

                                                                    'conv4_block8_3_bn
[0][0]']

 conv4_block8_out (Activati  (None, 3, 3, 1024)          0         ['conv4_block8_add[0
][0]']
 on)


 conv4_block9_1_conv (Conv2  (None, 3, 3, 256)           262400    ['conv4_block8_out[0]
[0]']
 D)


 conv4_block9_1_bn (BatchNo  (None, 3, 3, 256)           1024      ['conv4_block9_1_conv
[0][0]']
 rmalization)


 conv4_block9_1_relu (Activ  (None, 3, 3, 256)           0         ['conv4_block9_1_bn[
0][0]']
 ation)


 conv4_block9_2_conv (Conv2  (None, 3, 3, 256)           590080    ['conv4_block9_1_relu
[0][0]']
 D)


 conv4_block9_2_bn (BatchNo  (None, 3, 3, 256)           1024      ['conv4_block9_2_conv
[0][0]']
```

```
 rmalization)


 conv4_block9_2_relu (Activ  (None, 3, 3, 256)           0         ['conv4_block9_2_bn[
0][0]']
 ation)


 conv4_block9_3_conv (Conv2  (None, 3, 3, 1024)          263168    ['conv4_block9_2_relu
[0][0]']
 D)


 conv4_block9_3_bn (BatchNo  (None, 3, 3, 1024)          4096      ['conv4_block9_3_conv
[0][0]']
 rmalization)


 conv4_block9_add (Add)      (None, 3, 3, 1024)          0         ['conv4_block8_out[0
][0]',
                                                                    'conv4_block9_3_bn
[0][0]']

 conv4_block9_out (Activati  (None, 3, 3, 1024)          0         ['conv4_block9_add[0
][0]']
 on)


 conv4_block10_1_conv (Conv  (None, 3, 3, 256)           262400    ['conv4_block9_out[0]
[0]']
 2D)


 conv4_block10_1_bn (BatchN  (None, 3, 3, 256)           1024      ['conv4_block10_1_con
v[0][0]']
 ormalization)


 conv4_block10_1_relu (Acti  (None, 3, 3, 256)           0         ['conv4_block10_1_bn
[0][0]']
 vation)


 conv4_block10_2_conv (Conv  (None, 3, 3, 256)           590080    ['conv4_block10_1_rel
u[0][0]']
 2D)


 conv4_block10_2_bn (BatchN  (None, 3, 3, 256)           1024      ['conv4_block10_2_con
v[0][0]']
 ormalization)


 conv4_block10_2_relu (Acti  (None, 3, 3, 256)           0         ['conv4_block10_2_bn
[0][0]']
 vation)


 conv4_block10_3_conv (Conv  (None, 3, 3, 1024)          263168    ['conv4_block10_2_rel
u[0][0]']
```

```
 2D)


 conv4_block10_3_bn (BatchN  (None, 3, 3, 1024)           4096        ['conv4_block10_3_con
v[0][0]']
 ormalization)


 conv4_block10_add (Add)     (None, 3, 3, 1024)           0           ['conv4_block9_out[0
][0]',
                                                                       'conv4_block10_3_b
n[0][0]']

 conv4_block10_out (Activat  (None, 3, 3, 1024)           0           ['conv4_block10_add[0
][0]']
 ion)


 conv4_block11_1_conv (Conv  (None, 3, 3, 256)            262400      ['conv4_block10_out[0
][0]']
 2D)


 conv4_block11_1_bn (BatchN  (None, 3, 3, 256)            1024        ['conv4_block11_1_con
v[0][0]']
 ormalization)


 conv4_block11_1_relu (Acti  (None, 3, 3, 256)            0           ['conv4_block11_1_bn
[0][0]']
 vation)


 conv4_block11_2_conv (Conv  (None, 3, 3, 256)            590080      ['conv4_block11_1_rel
u[0][0]']
 2D)


 conv4_block11_2_bn (BatchN  (None, 3, 3, 256)            1024        ['conv4_block11_2_con
v[0][0]']
 ormalization)


 conv4_block11_2_relu (Acti  (None, 3, 3, 256)            0           ['conv4_block11_2_bn
[0][0]']
 vation)


 conv4_block11_3_conv (Conv  (None, 3, 3, 1024)           263168      ['conv4_block11_2_rel
u[0][0]']
 2D)


 conv4_block11_3_bn (BatchN  (None, 3, 3, 1024)           4096        ['conv4_block11_3_con
v[0][0]']
 ormalization)


 conv4_block11_add (Add)     (None, 3, 3, 1024)           0           ['conv4_block10_out[
0][0]',
                                                                          'conv4_block11_3_b
n[0][0]']

 conv4_block11_out (Activat  (None, 3, 3, 1024)          0          ['conv4_block11_add[0
][0]']
 ion)


 conv4_block12_1_conv (Conv  (None, 3, 3, 256)           262400     ['conv4_block11_out[0
][0]']
 2D)


 conv4_block12_1_bn (BatchN  (None, 3, 3, 256)           1024       ['conv4_block12_1_con
v[0][0]']
 ormalization)


 conv4_block12_1_relu (Acti  (None, 3, 3, 256)           0          ['conv4_block12_1_bn
[0][0]']
 vation)


 conv4_block12_2_conv (Conv  (None, 3, 3, 256)           590080     ['conv4_block12_1_rel
u[0][0]']
 2D)


 conv4_block12_2_bn (BatchN  (None, 3, 3, 256)           1024       ['conv4_block12_2_con
v[0][0]']
 ormalization)


 conv4_block12_2_relu (Acti  (None, 3, 3, 256)           0          ['conv4_block12_2_bn
[0][0]']
 vation)


 conv4_block12_3_conv (Conv  (None, 3, 3, 1024)          263168     ['conv4_block12_2_rel
u[0][0]']
 2D)


 conv4_block12_3_bn (BatchN  (None, 3, 3, 1024)          4096       ['conv4_block12_3_con
v[0][0]']
 ormalization)


 conv4_block12_add (Add)     (None, 3, 3, 1024)          0          ['conv4_block11_out[
0][0]',

                                                                     'conv4_block12_3_b
n[0][0]']

 conv4_block12_out (Activat  (None, 3, 3, 1024)          0          ['conv4_block12_add[0
][0]']
 ion)


 conv4_block13_1_conv (Conv  (None, 3, 3, 256)           262400     ['conv4_block12_out[0
][0]']
```

```
 2D)


 conv4_block13_1_bn (BatchN  (None, 3, 3, 256)            1024      ['conv4_block13_1_con
v[0][0]']
 ormalization)


 conv4_block13_1_relu (Acti  (None, 3, 3, 256)            0         ['conv4_block13_1_bn
[0][0]']
 vation)


 conv4_block13_2_conv (Conv  (None, 3, 3, 256)            590080    ['conv4_block13_1_rel
u[0][0]']
 2D)


 conv4_block13_2_bn (BatchN  (None, 3, 3, 256)            1024      ['conv4_block13_2_con
v[0][0]']
 ormalization)


 conv4_block13_2_relu (Acti  (None, 3, 3, 256)            0         ['conv4_block13_2_bn
[0][0]']
 vation)


 conv4_block13_3_conv (Conv  (None, 3, 3, 1024)           263168    ['conv4_block13_2_rel
u[0][0]']
 2D)


 conv4_block13_3_bn (BatchN  (None, 3, 3, 1024)           4096      ['conv4_block13_3_con
v[0][0]']
 ormalization)


 conv4_block13_add (Add)     (None, 3, 3, 1024)           0         ['conv4_block12_out[
0][0]',
                                                                     'conv4_block13_3_b
n[0][0]']

 conv4_block13_out (Activat  (None, 3, 3, 1024)           0         ['conv4_block13_add[0
][0]']
 ion)


 conv4_block14_1_conv (Conv  (None, 3, 3, 256)            262400    ['conv4_block13_out[0
][0]']
 2D)


 conv4_block14_1_bn (BatchN  (None, 3, 3, 256)            1024      ['conv4_block14_1_con
v[0][0]']
 ormalization)


 conv4_block14_1_relu (Acti  (None, 3, 3, 256)            0         ['conv4_block14_1_bn
[0][0]']
```

```
 vation)


 conv4_block14_2_conv (Conv  (None, 3, 3, 256)            590080    ['conv4_block14_1_rel
u[0][0]']
 2D)


 conv4_block14_2_bn (BatchN  (None, 3, 3, 256)            1024      ['conv4_block14_2_con
v[0][0]']
 ormalization)


 conv4_block14_2_relu (Acti  (None, 3, 3, 256)            0         ['conv4_block14_2_bn
[0][0]']
 vation)


 conv4_block14_3_conv (Conv  (None, 3, 3, 1024)           263168    ['conv4_block14_2_rel
u[0][0]']
 2D)


 conv4_block14_3_bn (BatchN  (None, 3, 3, 1024)           4096      ['conv4_block14_3_con
v[0][0]']
 ormalization)


 conv4_block14_add (Add)     (None, 3, 3, 1024)           0         ['conv4_block13_out[
0][0]',

                                                                     'conv4_block14_3_b
n[0][0]']

 conv4_block14_out (Activat  (None, 3, 3, 1024)           0         ['conv4_block14_add[0
][0]']
 ion)


 conv4_block15_1_conv (Conv  (None, 3, 3, 256)            262400    ['conv4_block14_out[0
][0]']
 2D)


 conv4_block15_1_bn (BatchN  (None, 3, 3, 256)            1024      ['conv4_block15_1_con
v[0][0]']
 ormalization)


 conv4_block15_1_relu (Acti  (None, 3, 3, 256)            0         ['conv4_block15_1_bn
[0][0]']
 vation)


 conv4_block15_2_conv (Conv  (None, 3, 3, 256)            590080    ['conv4_block15_1_rel
u[0][0]']
 2D)


 conv4_block15_2_bn (BatchN  (None, 3, 3, 256)            1024      ['conv4_block15_2_con
v[0][0]']
```

```
 ormalization)


 conv4_block15_2_relu (Acti  (None, 3, 3, 256)            0          ['conv4_block15_2_bn
[0][0]']
 vation)


 conv4_block15_3_conv (Conv  (None, 3, 3, 1024)           263168     ['conv4_block15_2_rel
u[0][0]']
 2D)


 conv4_block15_3_bn (BatchN  (None, 3, 3, 1024)           4096       ['conv4_block15_3_con
v[0][0]']
 ormalization)


 conv4_block15_add (Add)     (None, 3, 3, 1024)           0          ['conv4_block14_out[
0][0]',

                                                                      'conv4_block15_3_b
n[0][0]']

 conv4_block15_out (Activat  (None, 3, 3, 1024)           0          ['conv4_block15_add[0
][0]']
 ion)


 conv4_block16_1_conv (Conv  (None, 3, 3, 256)            262400     ['conv4_block15_out[0
][0]']
 2D)


 conv4_block16_1_bn (BatchN  (None, 3, 3, 256)            1024       ['conv4_block16_1_con
v[0][0]']
 ormalization)


 conv4_block16_1_relu (Acti  (None, 3, 3, 256)            0          ['conv4_block16_1_bn
[0][0]']
 vation)


 conv4_block16_2_conv (Conv  (None, 3, 3, 256)            590080     ['conv4_block16_1_rel
u[0][0]']
 2D)


 conv4_block16_2_bn (BatchN  (None, 3, 3, 256)            1024       ['conv4_block16_2_con
v[0][0]']
 ormalization)


 conv4_block16_2_relu (Acti  (None, 3, 3, 256)            0          ['conv4_block16_2_bn
[0][0]']
 vation)


 conv4_block16_3_conv (Conv  (None, 3, 3, 1024)           263168     ['conv4_block16_2_rel
u[0][0]']
```

```
 2D)


 conv4_block16_3_bn (BatchN  (None, 3, 3, 1024)          4096      ['conv4_block16_3_con
v[0][0]']
 ormalization)


 conv4_block16_add (Add)     (None, 3, 3, 1024)          0         ['conv4_block15_out[
0][0]',

                                                                    'conv4_block16_3_b
n[0][0]']

 conv4_block16_out (Activat  (None, 3, 3, 1024)          0         ['conv4_block16_add[0
][0]']
 ion)


 conv4_block17_1_conv (Conv  (None, 3, 3, 256)           262400    ['conv4_block16_out[0
][0]']
 2D)


 conv4_block17_1_bn (BatchN  (None, 3, 3, 256)           1024      ['conv4_block17_1_con
v[0][0]']
 ormalization)


 conv4_block17_1_relu (Acti  (None, 3, 3, 256)           0         ['conv4_block17_1_bn
[0][0]']
 vation)


 conv4_block17_2_conv (Conv  (None, 3, 3, 256)           590080    ['conv4_block17_1_rel
u[0][0]']
 2D)


 conv4_block17_2_bn (BatchN  (None, 3, 3, 256)           1024      ['conv4_block17_2_con
v[0][0]']
 ormalization)


 conv4_block17_2_relu (Acti  (None, 3, 3, 256)           0         ['conv4_block17_2_bn
[0][0]']
 vation)


 conv4_block17_3_conv (Conv  (None, 3, 3, 1024)          263168    ['conv4_block17_2_rel
u[0][0]']
 2D)


 conv4_block17_3_bn (BatchN  (None, 3, 3, 1024)          4096      ['conv4_block17_3_con
v[0][0]']
 ormalization)


 conv4_block17_add (Add)     (None, 3, 3, 1024)          0         ['conv4_block16_out[
0][0]',
                                                                      'conv4_block17_3_b
n[0][0]']

 conv4_block17_out (Activat  (None, 3, 3, 1024)          0          ['conv4_block17_add[0
][0]']
 ion)


 conv4_block18_1_conv (Conv  (None, 3, 3, 256)           262400     ['conv4_block17_out[0
][0]']
 2D)


 conv4_block18_1_bn (BatchN  (None, 3, 3, 256)           1024       ['conv4_block18_1_con
v[0][0]']
 ormalization)


 conv4_block18_1_relu (Acti  (None, 3, 3, 256)           0          ['conv4_block18_1_bn
[0][0]']
 vation)


 conv4_block18_2_conv (Conv  (None, 3, 3, 256)           590080     ['conv4_block18_1_rel
u[0][0]']
 2D)


 conv4_block18_2_bn (BatchN  (None, 3, 3, 256)           1024       ['conv4_block18_2_con
v[0][0]']
 ormalization)


 conv4_block18_2_relu (Acti  (None, 3, 3, 256)           0          ['conv4_block18_2_bn
[0][0]']
 vation)


 conv4_block18_3_conv (Conv  (None, 3, 3, 1024)          263168     ['conv4_block18_2_rel
u[0][0]']
 2D)


 conv4_block18_3_bn (BatchN  (None, 3, 3, 1024)          4096       ['conv4_block18_3_con
v[0][0]']
 ormalization)


 conv4_block18_add (Add)     (None, 3, 3, 1024)          0          ['conv4_block17_out[
0][0]',

                                                                      'conv4_block18_3_b
n[0][0]']

 conv4_block18_out (Activat  (None, 3, 3, 1024)          0          ['conv4_block18_add[0
][0]']
 ion)


 conv4_block19_1_conv (Conv  (None, 3, 3, 256)           262400     ['conv4_block18_out[0
][0]']
```

```
 2D)


 conv4_block19_1_bn (BatchN  (None, 3, 3, 256)           1024      ['conv4_block19_1_con
v[0][0]']
 ormalization)


 conv4_block19_1_relu (Acti  (None, 3, 3, 256)           0         ['conv4_block19_1_bn
[0][0]']
 vation)


 conv4_block19_2_conv (Conv  (None, 3, 3, 256)           590080    ['conv4_block19_1_rel
u[0][0]']
 2D)


 conv4_block19_2_bn (BatchN  (None, 3, 3, 256)           1024      ['conv4_block19_2_con
v[0][0]']
 ormalization)


 conv4_block19_2_relu (Acti  (None, 3, 3, 256)           0         ['conv4_block19_2_bn
[0][0]']
 vation)


 conv4_block19_3_conv (Conv  (None, 3, 3, 1024)          263168    ['conv4_block19_2_rel
u[0][0]']
 2D)


 conv4_block19_3_bn (BatchN  (None, 3, 3, 1024)          4096      ['conv4_block19_3_con
v[0][0]']
 ormalization)


 conv4_block19_add (Add)     (None, 3, 3, 1024)          0         ['conv4_block18_out[
0][0]',
                                                                    'conv4_block19_3_b
n[0][0]']

 conv4_block19_out (Activat  (None, 3, 3, 1024)          0         ['conv4_block19_add[0
][0]']
 ion)


 conv4_block20_1_conv (Conv  (None, 3, 3, 256)           262400    ['conv4_block19_out[0
][0]']
 2D)


 conv4_block20_1_bn (BatchN  (None, 3, 3, 256)           1024      ['conv4_block20_1_con
v[0][0]']
 ormalization)


 conv4_block20_1_relu (Acti  (None, 3, 3, 256)           0         ['conv4_block20_1_bn
[0][0]']
```

```
 vation)


 conv4_block20_2_conv (Conv  (None, 3, 3, 256)            590080    ['conv4_block20_1_rel
u[0][0]']
 2D)


 conv4_block20_2_bn (BatchN  (None, 3, 3, 256)            1024      ['conv4_block20_2_con
v[0][0]']
 ormalization)


 conv4_block20_2_relu (Acti  (None, 3, 3, 256)            0         ['conv4_block20_2_bn
[0][0]']
 vation)


 conv4_block20_3_conv (Conv  (None, 3, 3, 1024)           263168    ['conv4_block20_2_rel
u[0][0]']
 2D)


 conv4_block20_3_bn (BatchN  (None, 3, 3, 1024)           4096      ['conv4_block20_3_con
v[0][0]']
 ormalization)


 conv4_block20_add (Add)     (None, 3, 3, 1024)           0         ['conv4_block19_out[
0][0]',

                                                                     'conv4_block20_3_b
n[0][0]']

 conv4_block20_out (Activat  (None, 3, 3, 1024)           0         ['conv4_block20_add[0
][0]']
 ion)


 conv4_block21_1_conv (Conv  (None, 3, 3, 256)            262400    ['conv4_block20_out[0
][0]']
 2D)


 conv4_block21_1_bn (BatchN  (None, 3, 3, 256)            1024      ['conv4_block21_1_con
v[0][0]']
 ormalization)


 conv4_block21_1_relu (Acti  (None, 3, 3, 256)            0         ['conv4_block21_1_bn
[0][0]']
 vation)


 conv4_block21_2_conv (Conv  (None, 3, 3, 256)            590080    ['conv4_block21_1_rel
u[0][0]']
 2D)


 conv4_block21_2_bn (BatchN  (None, 3, 3, 256)            1024      ['conv4_block21_2_con
v[0][0]']
```

```
 ormalization)


 conv4_block21_2_relu (Acti  (None, 3, 3, 256)            0          ['conv4_block21_2_bn
[0][0]']
 vation)


 conv4_block21_3_conv (Conv  (None, 3, 3, 1024)           263168     ['conv4_block21_2_rel
u[0][0]']
 2D)


 conv4_block21_3_bn (BatchN  (None, 3, 3, 1024)           4096       ['conv4_block21_3_con
v[0][0]']
 ormalization)


 conv4_block21_add (Add)     (None, 3, 3, 1024)           0          ['conv4_block20_out[
0][0]',
                                                                      'conv4_block21_3_b
n[0][0]']

 conv4_block21_out (Activat  (None, 3, 3, 1024)           0          ['conv4_block21_add[0
][0]']
 ion)


 conv4_block22_1_conv (Conv  (None, 3, 3, 256)            262400     ['conv4_block21_out[0
][0]']
 2D)


 conv4_block22_1_bn (BatchN  (None, 3, 3, 256)            1024       ['conv4_block22_1_con
v[0][0]']
 ormalization)


 conv4_block22_1_relu (Acti  (None, 3, 3, 256)            0          ['conv4_block22_1_bn
[0][0]']
 vation)


 conv4_block22_2_conv (Conv  (None, 3, 3, 256)            590080     ['conv4_block22_1_rel
u[0][0]']
 2D)


 conv4_block22_2_bn (BatchN  (None, 3, 3, 256)            1024       ['conv4_block22_2_con
v[0][0]']
 ormalization)


 conv4_block22_2_relu (Acti  (None, 3, 3, 256)            0          ['conv4_block22_2_bn
[0][0]']
 vation)


 conv4_block22_3_conv (Conv  (None, 3, 3, 1024)           263168     ['conv4_block22_2_rel
u[0][0]']
```

```
 2D)


 conv4_block22_3_bn (BatchN  (None, 3, 3, 1024)           4096      ['conv4_block22_3_con
v[0][0]']
 ormalization)


 conv4_block22_add (Add)     (None, 3, 3, 1024)           0         ['conv4_block21_out[
0][0]',
                                                                     'conv4_block22_3_b
n[0][0]']

 conv4_block22_out (Activat  (None, 3, 3, 1024)           0         ['conv4_block22_add[0
][0]']
 ion)


 conv4_block23_1_conv (Conv  (None, 3, 3, 256)            262400    ['conv4_block22_out[0
][0]']
 2D)


 conv4_block23_1_bn (BatchN  (None, 3, 3, 256)            1024      ['conv4_block23_1_con
v[0][0]']
 ormalization)


 conv4_block23_1_relu (Acti  (None, 3, 3, 256)            0         ['conv4_block23_1_bn
[0][0]']
 vation)


 conv4_block23_2_conv (Conv  (None, 3, 3, 256)            590080    ['conv4_block23_1_rel
u[0][0]']
 2D)


 conv4_block23_2_bn (BatchN  (None, 3, 3, 256)            1024      ['conv4_block23_2_con
v[0][0]']
 ormalization)


 conv4_block23_2_relu (Acti  (None, 3, 3, 256)            0         ['conv4_block23_2_bn
[0][0]']
 vation)


 conv4_block23_3_conv (Conv  (None, 3, 3, 1024)           263168    ['conv4_block23_2_rel
u[0][0]']
 2D)


 conv4_block23_3_bn (BatchN  (None, 3, 3, 1024)           4096      ['conv4_block23_3_con
v[0][0]']
 ormalization)


 conv4_block23_add (Add)     (None, 3, 3, 1024)           0         ['conv4_block22_out[
0][0]',
                                                                          'conv4_block23_3_b
n[0][0]']


 conv4_block23_out (Activat  (None, 3, 3, 1024)           0           ['conv4_block23_add[0
][0]']
 ion)



 conv5_block1_1_conv (Conv2  (None, 2, 2, 512)            524800      ['conv4_block23_out[0
][0]']
 D)



 conv5_block1_1_bn (BatchNo  (None, 2, 2, 512)            2048        ['conv5_block1_1_conv
[0][0]']
 rmalization)



 conv5_block1_1_relu (Activ  (None, 2, 2, 512)            0           ['conv5_block1_1_bn[
0][0]']
 ation)



 conv5_block1_2_conv (Conv2  (None, 2, 2, 512)            2359808     ['conv5_block1_1_relu
[0][0]']
 D)



 conv5_block1_2_bn (BatchNo  (None, 2, 2, 512)            2048        ['conv5_block1_2_conv
[0][0]']
 rmalization)



 conv5_block1_2_relu (Activ  (None, 2, 2, 512)            0           ['conv5_block1_2_bn[
0][0]']
 ation)



 conv5_block1_0_conv (Conv2  (None, 2, 2, 2048)           2099200     ['conv4_block23_out[0
][0]']
 D)



 conv5_block1_3_conv (Conv2  (None, 2, 2, 2048)           1050624     ['conv5_block1_2_relu
[0][0]']
 D)



 conv5_block1_0_bn (BatchNo  (None, 2, 2, 2048)           8192        ['conv5_block1_0_conv
[0][0]']
 rmalization)



 conv5_block1_3_bn (BatchNo  (None, 2, 2, 2048)           8192        ['conv5_block1_3_conv
[0][0]']
 rmalization)



 conv5_block1_add (Add)      (None, 2, 2, 2048)           0           ['conv5_block1_0_bn[
0][0]',
                                                                    'conv5_block1_3_bn
[0][0]']


 conv5_block1_out (Activati  (None, 2, 2, 2048)          0          ['conv5_block1_add[0
][0]']
 on)



 conv5_block2_1_conv (Conv2  (None, 2, 2, 512)           1049088    ['conv5_block1_out[0]
[0]']
 D)



 conv5_block2_1_bn (BatchNo  (None, 2, 2, 512)           2048       ['conv5_block2_1_conv
[0][0]']
 rmalization)



 conv5_block2_1_relu (Activ  (None, 2, 2, 512)           0          ['conv5_block2_1_bn[
0][0]']
 ation)



 conv5_block2_2_conv (Conv2  (None, 2, 2, 512)           2359808    ['conv5_block2_1_relu
[0][0]']
 D)



 conv5_block2_2_bn (BatchNo  (None, 2, 2, 512)           2048       ['conv5_block2_2_conv
[0][0]']
 rmalization)



 conv5_block2_2_relu (Activ  (None, 2, 2, 512)           0          ['conv5_block2_2_bn[
0][0]']
 ation)



 conv5_block2_3_conv (Conv2  (None, 2, 2, 2048)          1050624    ['conv5_block2_2_relu
[0][0]']
 D)



 conv5_block2_3_bn (BatchNo  (None, 2, 2, 2048)          8192       ['conv5_block2_3_conv
[0][0]']
 rmalization)



 conv5_block2_add (Add)      (None, 2, 2, 2048)          0          ['conv5_block1_out[0
][0]',

                                                                     'conv5_block2_3_bn
[0][0]']


 conv5_block2_out (Activati  (None, 2, 2, 2048)          0          ['conv5_block2_add[0
][0]']
 on)



 conv5_block3_1_conv (Conv2  (None, 2, 2, 512)           1049088    ['conv5_block2_out[0]
[0]']
```

```
 D)


 conv5_block3_1_bn (BatchNo  (None, 2, 2, 512)            2048      ['conv5_block3_1_conv
[0][0]']
 rmalization)


 conv5_block3_1_relu (Activ  (None, 2, 2, 512)            0         ['conv5_block3_1_bn[
0][0]']
 ation)


 conv5_block3_2_conv (Conv2  (None, 2, 2, 512)            2359808   ['conv5_block3_1_relu
[0][0]']
 D)


 conv5_block3_2_bn (BatchNo  (None, 2, 2, 512)            2048      ['conv5_block3_2_conv
[0][0]']
 rmalization)


 conv5_block3_2_relu (Activ  (None, 2, 2, 512)            0         ['conv5_block3_2_bn[
0][0]']
 ation)


 conv5_block3_3_conv (Conv2  (None, 2, 2, 2048)           1050624   ['conv5_block3_2_relu
[0][0]']
 D)


 conv5_block3_3_bn (BatchNo  (None, 2, 2, 2048)           8192      ['conv5_block3_3_conv
[0][0]']
 rmalization)


 conv5_block3_add (Add)      (None, 2, 2, 2048)           0         ['conv5_block2_out[0
][0]',
                                                                     'conv5_block3_3_bn

[0][0]']


 conv5_block3_out (Activati  (None, 2, 2, 2048)           0         ['conv5_block3_add[0
][0]']
 on)


==========================================================================================
=========
Total params: 42658176 (162.73 MB)
Trainable params: 42552832 (162.33 MB)
Non-trainable params: 105344 (411.50 KB)
__________________________________________________________________________________________
_________
```

Model Building

- **In this model, we will import till the 'conv5_block3_add' layer of the ResNet model. You can scroll down in the model summary and look for 'conv5_block3_add'. You can choose any other layer as well.**
- **Then we will add a Flatten layer, which receives the output of the 'conv5_block3_add' layer as its input.**

- Then we will add a Flatten layer, which receives the output of the 'conv5_block3_add' layer as its input.
- We will add a few Dense layers and use 'relu' activation function on them.
- You may use Dropout and BatchNormalization layers as well.
- Then we will add our last dense layer, which must have 4 neurons and a 'softmax' activation function.

In [ ]:

```
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout, BatchNormaliz
ation
from tensorflow.keras.regularizers import l2

base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
base_model.trainable = False

x = GlobalAveragePooling2D()(base_model.output)
x = Dense(1024, activation='relu', kernel_regularizer=l2(0.01))(x)
x = Dropout(0.5)(x)
x = BatchNormalization()(x)
x = Dense(512, activation='relu', kernel_regularizer=l2(0.01))(x)
x = Dropout(0.5)(x)
predictions = Dense(4, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
```

Compiling and Training the Model

In [ ]:

```
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

checkpoint = ModelCheckpoint("./Resnetmodel.h5", monitor = 'val_acc', verbose = 1, save_
best_only = True, mode = 'max')

early_stopping = EarlyStopping(
    monitor='val_loss',
    min_delta=0.001,
    patience=5,
    verbose=1,
    mode='min',
    restore_best_weights=True
)


reduce_learningrate = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.2,
    patience=3,
    verbose=1,
    mode='min',
    min_delta=0.0001,
    cooldown=0,
    min_lr=0
)
callbacks_list = [early_stopping, checkpoint, reduce_learningrate]
epochs = 10
```

In [ ]:

```
# Write your code to compile your resnetmodel. Use categorical crossentropy as your loss
function, Adam Optimizer with 0.001 learning rate, and set your metrics to 'accuracy'.
resnetmodel.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

In [ ]:

```
history = resnetmodel.fit(
    train_set,
    epochs=20,
    validation_data=validation_set,
    callbacks=callbacks_list,
    verbose=1
)
```

```
Epoch 1/20
473/473 [==============================] - ETA: 0s - loss: 1.3967 - accuracy: 0.2563
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 262s 545ms/step - loss: 1.3967 - accuracy: 0.2
563 - val_loss: 1.3712 - val_accuracy: 0.2289 - lr: 0.0010
Epoch 2/20
473/473 [==============================] - ETA: 0s - loss: 1.3910 - accuracy: 0.2671
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 254s 536ms/step - loss: 1.3910 - accuracy: 0.2
671 - val_loss: 1.3479 - val_accuracy: 0.3667 - lr: 0.0010
Epoch 3/20
473/473 [==============================] - ETA: 0s - loss: 1.3903 - accuracy: 0.2640
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 220s 466ms/step - loss: 1.3903 - accuracy: 0.2
640 - val_loss: 1.3632 - val_accuracy: 0.2301 - lr: 0.0010
Epoch 4/20
473/473 [==============================] - ETA: 0s - loss: 1.3865 - accuracy: 0.2717
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 224s 473ms/step - loss: 1.3865 - accuracy: 0.2
717 - val_loss: 1.4000 - val_accuracy: 0.2289 - lr: 0.0010
Epoch 5/20
473/473 [==============================] - ETA: 0s - loss: 1.3803 - accuracy: 0.2802
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
Epoch 5: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
473/473 [==============================] - 255s 539ms/step - loss: 1.3803 - accuracy: 0.2
802 - val_loss: 1.3943 - val_accuracy: 0.2311 - lr: 0.0010
Epoch 6/20
473/473 [==============================] - ETA: 0s - loss: 1.3509 - accuracy: 0.3075
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 255s 540ms/step - loss: 1.3509 - accuracy: 0.3
075 - val_loss: 1.3629 - val_accuracy: 0.2371 - lr: 2.0000e-04
Epoch 7/20
473/473 [==============================] - ETA: 0s - loss: 1.3419 - accuracy: 0.3205
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 222s 469ms/step - loss: 1.3419 - accuracy: 0.3
205 - val_loss: 1.3428 - val_accuracy: 0.2467 - lr: 2.0000e-04
Epoch 8/20
473/473 [==============================] - ETA: 0s - loss: 1.3380 - accuracy: 0.3148
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 254s 537ms/step - loss: 1.3380 - accuracy: 0.3
148 - val_loss: 1.3406 - val_accuracy: 0.2489 - lr: 2.0000e-04
Epoch 9/20
473/473 [==============================] - ETA: 0s - loss: 1.3367 - accuracy: 0.3213
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 255s 538ms/step - loss: 1.3367 - accuracy: 0.3
213 - val_loss: 1.3414 - val_accuracy: 0.2508 - lr: 2.0000e-04
Epoch 10/20
473/473 [==============================] - ETA: 0s - loss: 1.3309 - accuracy: 0.3234
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 220s 466ms/step - loss: 1.3309 - accuracy: 0.3
234 - val_loss: 1.3129 - val_accuracy: 0.3209 - lr: 2.0000e-04
Epoch 11/20
473/473 [==============================] - ETA: 0s - loss: 1.3304 - accuracy: 0.3240
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 252s 533ms/step - loss: 1.3304 - accuracy: 0.3
240 - val_loss: 1.3565 - val_accuracy: 0.2427 - lr: 2.0000e-04
Epoch 12/20
473/473 [==============================] - ETA: 0s - loss: 1.3283 - accuracy: 0.3235
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 253s 536ms/step - loss: 1.3283 - accuracy: 0.3
235 - val_loss: 1.3148 - val_accuracy: 0.2837 - lr: 2.0000e-04
Epoch 13/20
473/473 [==============================] - ETA: 0s - loss: 1.3219 - accuracy: 0.3290
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
Epoch 13: ReduceLROnPlateau reducing learning rate to 4.0000001899898055e-05.
473/473 [==============================] - 220s 465ms/step - loss: 1.3219 - accuracy: 0.3
290 - val_loss: 1.3209 - val_accuracy: 0.2944 - lr: 2.0000e-04
Epoch 14/20
473/473 [==============================] - ETA: 0s - loss: 1.3255 - accuracy: 0.3287
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 253s 535ms/step - loss: 1.3255 - accuracy: 0.3
287 - val_loss: 1.3118 - val_accuracy: 0.3124 - lr: 4.0000e-05
Epoch 15/20
473/473 [==============================] - ETA: 0s - loss: 1.3201 - accuracy: 0.3348
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 219s 464ms/step - loss: 1.3201 - accuracy: 0.3
348 - val_loss: 1.3071 - val_accuracy: 0.3251 - lr: 4.0000e-05
Epoch 16/20
473/473 [==============================] - ETA: 0s - loss: 1.3178 - accuracy: 0.3352
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 221s 467ms/step - loss: 1.3178 - accuracy: 0.3
352 - val_loss: 1.3080 - val_accuracy: 0.3022 - lr: 4.0000e-05
Epoch 17/20
473/473 [==============================] - ETA: 0s - loss: 1.3127 - accuracy: 0.3377
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 219s 463ms/step - loss: 1.3127 - accuracy: 0.3
377 - val_loss: 1.3005 - val_accuracy: 0.3528 - lr: 4.0000e-05
Epoch 18/20
473/473 [==============================] - ETA: 0s - loss: 1.3189 - accuracy: 0.3401
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 254s 537ms/step - loss: 1.3189 - accuracy: 0.3
401 - val_loss: 1.3022 - val_accuracy: 0.3582 - lr: 4.0000e-05
Epoch 19/20
473/473 [==============================] - ETA: 0s - loss: 1.3171 - accuracy: 0.3438
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 254s 538ms/step - loss: 1.3171 - accuracy: 0.3
438 - val_loss: 1.3003 - val_accuracy: 0.3653 - lr: 4.0000e-05
Epoch 20/20
473/473 [==============================] - ETA: 0s - loss: 1.3132 - accuracy: 0.3486
```

```
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

```
473/473 [==============================] - 223s 471ms/step - loss: 1.3132 - accuracy: 0.3
486 - val_loss: 1.3028 - val_accuracy: 0.3325 - lr: 4.0000e-05
```

Evaluating the ResNet Model

In [ ]:

```
# Write your code to evaluate model performance on the test set

test_loss, test_accuracy = resnetmodel.evaluate(test_set)

# Print the test loss and accuracy
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
4/4 [==============================] - 1s 294ms/step - loss: 1.2990 - accuracy: 0.4062
Test Loss: 1.2989614009857178
Test Accuracy: 0.40625
```

Observations and Insights:

The model's jump in accuracy to 40.62% marks a solid improvement, showing the tweaks we've made are steering us in the right direction. Yet, we're not quite there for a victory lap—there's plenty of space to push those numbers higher. Here's the lowdown:

Sizing Up the Model: Gotta check if our model's too big for its boots. If it's too complex, might be time to trim it down or crank up the regularization to avoid overfitting. Diving Deeper with Fine-Tuning: If those ResNet50 layers are still on ice, thawing some for fine-tuning could give us that extra edge, making the model feel more at home with our data. Amping Up Data Augmentation: Time to get more creative with our data augmentation strategies. A little more variety might just be the spice our model needs. Tweaking the Learning Rate: Playing around with the learning rate and introducing a schedule for it could smooth out the learning process, leading to better results. Hyperparameter Hustle: There's more to tune than just the learning rate. Batch size, the optimizer, you name it—let's not leave stones unturned. Next-Level Regularization: Beyond the usual suspects like dropout, exploring new regularization techniques could give us a leg up. Beyond Accuracy: If our dataset's skewed or if some mistakes cost more than others, let's not forget to check out other metrics like precision and recall. Spotting Errors: A closer look at where our model's tripping can clue us in on what needs fixing, be it more data for certain classes or tweaking the model. Teamwork with Ensembles: Why rely on one model when a team can bring home the win? Ensemble methods could be our ticket to top-tier accuracy. Data is King: At the end of the day, the more quality data we can feed our model, the better it's gonna perform. In short, we've made some headway, but the road to model perfection is paved with more tuning, testing, and data gathering.

EfficientNet Model

In [ ]:

```
import tensorflow as tf
import tensorflow.keras.applications as ap
from tensorflow.keras import Model
EfficientNet = ap.EfficientNetV2B2(include_top=False,weights="imagenet", input_shape= (4
8, 48, 3))

EfficientNet.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/effici
entnet_v2/efficientnetv2-b2_notop.h5
35839040/35839040 [==============================] - 0s 0us/step
Model: "efficientnetv2-b2"
_________________________________________________________________________________________
_________
 Layer (type)                Output Shape                 Param #     Connected to

=========================================================================================
=========
 input_16 (InputLayer)       [(None, 48, 48, 3)]          0           []


 rescaling (Rescaling)       (None, 48, 48, 3)            0           ['input_16[0][0]']
 normalization (Normalizati  (None, 48, 48, 3)          0          ['rescaling[0][0]']

 on)


 stem_conv (Conv2D)          (None, 24, 24, 32)         864        ['normalization[0][0]
']

 stem_bn (BatchNormalizatio  (None, 24, 24, 32)         128        ['stem_conv[0][0]']

 n)


 stem_activation (Activatio  (None, 24, 24, 32)         0          ['stem_bn[0][0]']

 n)


 block1a_project_conv (Conv  (None, 24, 24, 16)         4608       ['stem_activation[0][
0]']
 2D)


 block1a_project_bn (BatchN  (None, 24, 24, 16)         64         ['block1a_project_con
v[0][0]']
 ormalization)


 block1a_project_activation  (None, 24, 24, 16)         0          ['block1a_project_bn
[0][0]']
  (Activation)


 block1b_project_conv (Conv  (None, 24, 24, 16)         2304       ['block1a_project_act
ivation[0
 2D)                                                               ][0]']


 block1b_project_bn (BatchN  (None, 24, 24, 16)         64         ['block1b_project_con
v[0][0]']
 ormalization)


 block1b_project_activation  (None, 24, 24, 16)         0          ['block1b_project_bn
[0][0]']
  (Activation)


 block1b_drop (Dropout)      (None, 24, 24, 16)         0          ['block1b_project_ac
tivation[0
                                                                   ][0]']


 block1b_add (Add)           (None, 24, 24, 16)         0          ['block1b_drop[0][0]
',
                                                                    'block1a_project_a
ctivation[0
                                                                   ][0]']
 block2a_expand_conv (Conv2  (None, 12, 12, 64)           9216      ['block1b_add[0][0]']

 D)



 block2a_expand_bn (BatchNo  (None, 12, 12, 64)           256       ['block2a_expand_conv
[0][0]']
 rmalization)



 block2a_expand_activation   (None, 12, 12, 64)           0         ['block2a_expand_bn[
0][0]']
 (Activation)



 block2a_project_conv (Conv  (None, 12, 12, 32)           2048      ['block2a_expand_acti
vation[0]
 2D)                                                                [0]']



 block2a_project_bn (BatchN  (None, 12, 12, 32)           128       ['block2a_project_con
v[0][0]']
 ormalization)



 block2b_expand_conv (Conv2  (None, 12, 12, 128)          36864     ['block2a_project_bn[
0][0]']
 D)



 block2b_expand_bn (BatchNo  (None, 12, 12, 128)          512       ['block2b_expand_conv
[0][0]']
 rmalization)



 block2b_expand_activation   (None, 12, 12, 128)          0         ['block2b_expand_bn[
0][0]']
 (Activation)



 block2b_project_conv (Conv  (None, 12, 12, 32)           4096      ['block2b_expand_acti
vation[0]
 2D)                                                                [0]']



 block2b_project_bn (BatchN  (None, 12, 12, 32)           128       ['block2b_project_con
v[0][0]']
 ormalization)



 block2b_drop (Dropout)      (None, 12, 12, 32)           0         ['block2b_project_bn
[0][0]']

 block2b_add (Add)           (None, 12, 12, 32)           0         ['block2b_drop[0][0]
',
                                                                     'block2a_project_b

n[0][0]']


 block2c_expand_conv (Conv2  (None, 12, 12, 128)          36864     ['block2b_add[0][0]']
 block2c_expand_conv (Conv2  (None, 12, 12, 128)          36864     ['block2b_add[0][0]']

 D)


 block2c_expand_bn (BatchNo  (None, 12, 12, 128)          512       ['block2c_expand_conv
[0][0]']
 rmalization)


 block2c_expand_activation   (None, 12, 12, 128)          0         ['block2c_expand_bn[
0][0]']
 (Activation)


 block2c_project_conv (Conv  (None, 12, 12, 32)           4096      ['block2c_expand_acti
vation[0]
 2D)                                                                [0]']


 block2c_project_bn (BatchN  (None, 12, 12, 32)           128       ['block2c_project_con
v[0][0]']
 ormalization)


 block2c_drop (Dropout)      (None, 12, 12, 32)           0         ['block2c_project_bn
[0][0]']

 block2c_add (Add)           (None, 12, 12, 32)           0         ['block2c_drop[0][0]
',
                                                                     'block2b_add[0][0]
']

 block3a_expand_conv (Conv2  (None, 6, 6, 128)            36864     ['block2c_add[0][0]']

 D)


 block3a_expand_bn (BatchNo  (None, 6, 6, 128)            512       ['block3a_expand_conv
[0][0]']
 rmalization)


 block3a_expand_activation   (None, 6, 6, 128)            0         ['block3a_expand_bn[
0][0]']
 (Activation)


 block3a_project_conv (Conv  (None, 6, 6, 56)             7168      ['block3a_expand_acti
vation[0]
 2D)                                                                [0]']


 block3a_project_bn (BatchN  (None, 6, 6, 56)             224       ['block3a_project_con
v[0][0]']
 ormalization)


 block3b_expand_conv (Conv2  (None, 6, 6, 224)            112896    ['block3a_project_bn[
0][0]']
 D)
 D)


 block3b_expand_bn (BatchNo  (None, 6, 6, 224)            896       ['block3b_expand_conv
[0][0]']
 rmalization)


 block3b_expand_activation   (None, 6, 6, 224)            0         ['block3b_expand_bn[
0][0]']
 (Activation)


 block3b_project_conv (Conv  (None, 6, 6, 56)             12544     ['block3b_expand_acti
vation[0]
 2D)                                                                [0]']


 block3b_project_bn (BatchN  (None, 6, 6, 56)             224       ['block3b_project_con
v[0][0]']
 ormalization)


 block3b_drop (Dropout)      (None, 6, 6, 56)             0         ['block3b_project_bn
[0][0]']

 block3b_add (Add)           (None, 6, 6, 56)             0         ['block3b_drop[0][0]
',
                                                                     'block3a_project_b
n[0][0]']

 block3c_expand_conv (Conv2  (None, 6, 6, 224)            112896    ['block3b_add[0][0]']

 D)


 block3c_expand_bn (BatchNo  (None, 6, 6, 224)            896       ['block3c_expand_conv
[0][0]']
 rmalization)


 block3c_expand_activation   (None, 6, 6, 224)            0         ['block3c_expand_bn[
0][0]']
 (Activation)


 block3c_project_conv (Conv  (None, 6, 6, 56)             12544     ['block3c_expand_acti
vation[0]
 2D)                                                                [0]']


 block3c_project_bn (BatchN  (None, 6, 6, 56)             224       ['block3c_project_con
v[0][0]']
 ormalization)


 block3c_drop (Dropout)      (None, 6, 6, 56)             0         ['block3c_project_bn
[0][0]']

 block3c_add (Add)           (None, 6, 6, 56)             0         ['block3c_drop[0][0]
 block3c_add (Add)              (None, 6, 6, 56)             0           ['block3c_drop[0][0]
',
                                                                          'block3b_add[0][0]
']

 block4a_expand_conv (Conv2D    (None, 6, 6, 224)            12544       ['block3c_add[0][0]']
 D)

 block4a_expand_bn (BatchNo     (None, 6, 6, 224)            896         ['block4a_expand_conv
[0][0]']
 rmalization)

 block4a_expand_activation      (None, 6, 6, 224)            0           ['block4a_expand_bn[
0][0]']
  (Activation)

 block4a_dwconv2 (Depthwise     (None, 3, 3, 224)            2016        ['block4a_expand_acti
vation[0]
 Conv2D)                                                                 [0]']

 block4a_bn (BatchNormaliza     (None, 3, 3, 224)            896         ['block4a_dwconv2[0][
0]']
 tion)

 block4a_activation (Activa     (None, 3, 3, 224)            0           ['block4a_bn[0][0]']
 tion)

 block4a_se_squeeze (Global     (None, 224)                  0           ['block4a_activation
[0][0]']
 AveragePooling2D)

 block4a_se_reshape (Reshap     (None, 1, 1, 224)            0           ['block4a_se_squeeze[
0][0]']
 e)

 block4a_se_reduce (Conv2D)     (None, 1, 1, 14)             3150        ['block4a_se_reshape[
0][0]']

 block4a_se_expand (Conv2D)     (None, 1, 1, 224)            3360        ['block4a_se_reduce[0
][0]']

 block4a_se_excite (Multipl     (None, 3, 3, 224)            0           ['block4a_activation[
0][0]',
 y)                                                                       'block4a_se_expand
[0][0]']

 block4a_project_conv (Conv     (None, 3, 3, 104)            23296       ['block4a_se_excite[0
][0]']
 2D)
 block4a_project_bn (BatchN  (None, 3, 3, 104)            416        ['block4a_project_con
v[0][0]']
 ormalization)


 block4b_expand_conv (Conv2  (None, 3, 3, 416)            43264      ['block4a_project_bn[
0][0]']
 D)


 block4b_expand_bn (BatchNo  (None, 3, 3, 416)            1664       ['block4b_expand_conv
[0][0]']
 rmalization)


 block4b_expand_activation   (None, 3, 3, 416)            0          ['block4b_expand_bn[
0][0]']
 (Activation)


 block4b_dwconv2 (Depthwise  (None, 3, 3, 416)            3744       ['block4b_expand_acti
vation[0]
 Conv2D)                                                              [0]']


 block4b_bn (BatchNormaliza  (None, 3, 3, 416)            1664       ['block4b_dwconv2[0][
0]']
 tion)


 block4b_activation (Activa  (None, 3, 3, 416)            0          ['block4b_bn[0][0]']

 tion)


 block4b_se_squeeze (Global  (None, 416)                  0          ['block4b_activation
[0][0]']
 AveragePooling2D)


 block4b_se_reshape (Reshap  (None, 1, 1, 416)            0          ['block4b_se_squeeze[
0][0]']
 e)


 block4b_se_reduce (Conv2D)  (None, 1, 1, 26)             10842      ['block4b_se_reshape[
0][0]']

 block4b_se_expand (Conv2D)  (None, 1, 1, 416)            11232      ['block4b_se_reduce[0
][0]']

 block4b_se_excite (Multipl  (None, 3, 3, 416)            0          ['block4b_activation[
0][0]',
 y)                                                                    'block4b_se_expand
[0][0]']

 block4b_project_conv (Conv  (None, 3, 3, 104)            43264      ['block4b_se_excite[0
][0]']
 2D)
 2D)


 block4b_project_bn (BatchN  (None, 3, 3, 104)            416       ['block4b_project_con
v[0][0]']
 ormalization)



 block4b_drop (Dropout)      (None, 3, 3, 104)            0         ['block4b_project_bn
[0][0]']

 block4b_add (Add)           (None, 3, 3, 104)            0         ['block4b_drop[0][0]
',
                                                                     'block4a_project_b
n[0][0]']

 block4c_expand_conv (Conv2  (None, 3, 3, 416)            43264     ['block4b_add[0][0]']

 D)



 block4c_expand_bn (BatchNo  (None, 3, 3, 416)            1664      ['block4c_expand_conv
[0][0]']
 rmalization)



 block4c_expand_activation   (None, 3, 3, 416)            0         ['block4c_expand_bn[
0][0]']
 (Activation)



 block4c_dwconv2 (Depthwise  (None, 3, 3, 416)            3744      ['block4c_expand_acti
vation[0]
 Conv2D)                                                            [0]']



 block4c_bn (BatchNormaliza  (None, 3, 3, 416)            1664      ['block4c_dwconv2[0][
0]']
 tion)



 block4c_activation (Activa  (None, 3, 3, 416)            0         ['block4c_bn[0][0]']

 tion)



 block4c_se_squeeze (Global  (None, 416)                  0         ['block4c_activation
[0][0]']
 AveragePooling2D)



 block4c_se_reshape (Reshap  (None, 1, 1, 416)            0         ['block4c_se_squeeze[
0][0]']
 e)



 block4c_se_reduce (Conv2D)  (None, 1, 1, 26)             10842     ['block4c_se_reshape[
0][0]']

 block4c_se_expand (Conv2D)  (None, 1, 1, 416)            11232     ['block4c_se_reduce[0
 block4c_se_expand (Conv2D)  (None, 1, 1, 416)            11232      ['block4c_se_reduce[0
][0]']

 block4c_se_excite (Multipl  (None, 3, 3, 416)            0          ['block4c_activation[
0][0]',
 y)                                                                   'block4c_se_expand
[0][0]']

 block4c_project_conv (Conv  (None, 3, 3, 104)            43264      ['block4c_se_excite[0
][0]']
 2D)


 block4c_project_bn (BatchN  (None, 3, 3, 104)            416        ['block4c_project_con
v[0][0]']
 ormalization)


 block4c_drop (Dropout)      (None, 3, 3, 104)            0          ['block4c_project_bn
[0][0]']

 block4c_add (Add)           (None, 3, 3, 104)            0          ['block4c_drop[0][0]
',
                                                                      'block4b_add[0][0]
']

 block4d_expand_conv (Conv2  (None, 3, 3, 416)            43264      ['block4c_add[0][0]']

 D)


 block4d_expand_bn (BatchNo  (None, 3, 3, 416)            1664       ['block4d_expand_conv
[0][0]']
 rmalization)


 block4d_expand_activation   (None, 3, 3, 416)            0          ['block4d_expand_bn[
0][0]']
 (Activation)


 block4d_dwconv2 (Depthwise  (None, 3, 3, 416)            3744       ['block4d_expand_acti
vation[0]
 Conv2D)                                                             [0]']


 block4d_bn (BatchNormaliza  (None, 3, 3, 416)            1664       ['block4d_dwconv2[0][
0]']
 tion)


 block4d_activation (Activa  (None, 3, 3, 416)            0          ['block4d_bn[0][0]']

 tion)


 block4d_se_squeeze (Global  (None, 416)                  0          ['block4d_activation
[0][0]']
 AveragePooling2D)
 block4d_se_reshape (Reshap  (None, 1, 1, 416)            0         ['block4d_se_squeeze[
0][0]']
 e)



 block4d_se_reduce (Conv2D)  (None, 1, 1, 26)             10842     ['block4d_se_reshape[
0][0]']

 block4d_se_expand (Conv2D)  (None, 1, 1, 416)            11232     ['block4d_se_reduce[0
][0]']

 block4d_se_excite (Multipl  (None, 3, 3, 416)            0         ['block4d_activation[
0][0]',
 y)                                                                  'block4d_se_expand
[0][0]']

 block4d_project_conv (Conv  (None, 3, 3, 104)            43264     ['block4d_se_excite[0
][0]']
 2D)



 block4d_project_bn (BatchN  (None, 3, 3, 104)            416       ['block4d_project_con
v[0][0]']
 ormalization)



 block4d_drop (Dropout)      (None, 3, 3, 104)            0         ['block4d_project_bn
[0][0]']

 block4d_add (Add)           (None, 3, 3, 104)            0         ['block4d_drop[0][0]
',
                                                                     'block4c_add[0][0]
']

 block5a_expand_conv (Conv2  (None, 3, 3, 624)            64896     ['block4d_add[0][0]']

 D)



 block5a_expand_bn (BatchNo  (None, 3, 3, 624)            2496      ['block5a_expand_conv
[0][0]']
 rmalization)



 block5a_expand_activation   (None, 3, 3, 624)            0         ['block5a_expand_bn[
0][0]']
 (Activation)



 block5a_dwconv2 (Depthwise  (None, 3, 3, 624)            5616      ['block5a_expand_acti
vation[0]
 Conv2D)                                                            [0]']



 block5a_bn (BatchNormaliza  (None, 3, 3, 624)            2496      ['block5a_dwconv2[0][
0]']
 tion)
 block5a_activation (Activa  (None, 3, 3, 624)            0          ['block5a_bn[0][0]']

 tion)


 block5a_se_squeeze (Global  (None, 624)                  0          ['block5a_activation
[0][0]']
 AveragePooling2D)


 block5a_se_reshape (Reshap  (None, 1, 1, 624)            0          ['block5a_se_squeeze[
0][0]']
 e)


 block5a_se_reduce (Conv2D)  (None, 1, 1, 26)             16250      ['block5a_se_reshape[
0][0]']

 block5a_se_expand (Conv2D)  (None, 1, 1, 624)            16848      ['block5a_se_reduce[0
][0]']

 block5a_se_excite (Multipl  (None, 3, 3, 624)            0          ['block5a_activation[
0][0]',
 y)                                                                   'block5a_se_expand
[0][0]']

 block5a_project_conv (Conv  (None, 3, 3, 120)            74880      ['block5a_se_excite[0
][0]']
 2D)


 block5a_project_bn (BatchN  (None, 3, 3, 120)            480        ['block5a_project_con
v[0][0]']
 ormalization)


 block5b_expand_conv (Conv2  (None, 3, 3, 720)            86400      ['block5a_project_bn[
0][0]']
 D)


 block5b_expand_bn (BatchNo  (None, 3, 3, 720)            2880       ['block5b_expand_conv
[0][0]']
 rmalization)


 block5b_expand_activation   (None, 3, 3, 720)            0          ['block5b_expand_bn[
0][0]']
 (Activation)


 block5b_dwconv2 (Depthwise  (None, 3, 3, 720)            6480       ['block5b_expand_acti
vation[0]
 Conv2D)                                                             [0]']


 block5b_bn (BatchNormaliza  (None, 3, 3, 720)            2880       ['block5b_dwconv2[0][
0]']
 tion)
 tion)


 block5b_activation (Activa  (None, 3, 3, 720)            0           ['block5b_bn[0][0]']

 tion)


 block5b_se_squeeze (Global  (None, 720)                  0           ['block5b_activation
[0][0]']
 AveragePooling2D)


 block5b_se_reshape (Reshap  (None, 1, 1, 720)            0           ['block5b_se_squeeze[
0][0]']
 e)


 block5b_se_reduce (Conv2D)  (None, 1, 1, 30)             21630       ['block5b_se_reshape[
0][0]']

 block5b_se_expand (Conv2D)  (None, 1, 1, 720)            22320       ['block5b_se_reduce[0
][0]']

 block5b_se_excite (Multipl  (None, 3, 3, 720)            0           ['block5b_activation[
0][0]',
 y)                                                                    'block5b_se_expand
[0][0]']

 block5b_project_conv (Conv  (None, 3, 3, 120)            86400       ['block5b_se_excite[0
][0]']
 2D)


 block5b_project_bn (BatchN  (None, 3, 3, 120)            480         ['block5b_project_con
v[0][0]']
 ormalization)


 block5b_drop (Dropout)      (None, 3, 3, 120)            0           ['block5b_project_bn
[0][0]']

 block5b_add (Add)           (None, 3, 3, 120)            0           ['block5b_drop[0][0]
',
                                                                       'block5a_project_b
n[0][0]']

 block5c_expand_conv (Conv2  (None, 3, 3, 720)            86400       ['block5b_add[0][0]']

 D)


 block5c_expand_bn (BatchNo  (None, 3, 3, 720)            2880        ['block5c_expand_conv
[0][0]']
 rmalization)


 block5c_expand_activation   (None, 3, 3, 720)            0           ['block5c_expand_bn[
0][0]']
 (Activation)
 (Activation)


 block5c_dwconv2 (Depthwise  (None, 3, 3, 720)            6480      ['block5c_expand_acti
vation[0]
 Conv2D)                                                            [0]']



 block5c_bn (BatchNormaliza  (None, 3, 3, 720)            2880      ['block5c_dwconv2[0][
0]']
 tion)



 block5c_activation (Activa  (None, 3, 3, 720)            0         ['block5c_bn[0][0]']

 tion)



 block5c_se_squeeze (Global  (None, 720)                  0         ['block5c_activation
[0][0]']
 AveragePooling2D)



 block5c_se_reshape (Reshap  (None, 1, 1, 720)            0         ['block5c_se_squeeze[
0][0]']
 e)



 block5c_se_reduce (Conv2D)  (None, 1, 1, 30)             21630     ['block5c_se_reshape[
0][0]']

 block5c_se_expand (Conv2D)  (None, 1, 1, 720)            22320     ['block5c_se_reduce[0
][0]']

 block5c_se_excite (Multipl  (None, 3, 3, 720)            0         ['block5c_activation[
0][0]',
 y)                                                                  'block5c_se_expand
[0][0]']

 block5c_project_conv (Conv  (None, 3, 3, 120)            86400     ['block5c_se_excite[0
][0]']
 2D)



 block5c_project_bn (BatchN  (None, 3, 3, 120)            480       ['block5c_project_con
v[0][0]']
 ormalization)



 block5c_drop (Dropout)      (None, 3, 3, 120)            0         ['block5c_project_bn
[0][0]']

 block5c_add (Add)           (None, 3, 3, 120)            0         ['block5c_drop[0][0]
',
                                                                     'block5b_add[0][0]
']

 block5d_expand_conv (Conv2  (None, 3, 3, 720)            86400     ['block5c_add[0][0]']

 D)
 D)

 block5d_expand_bn (BatchNo  (None, 3, 3, 720)         2880       ['block5d_expand_conv
[0][0]']
 rmalization)

 block5d_expand_activation   (None, 3, 3, 720)         0          ['block5d_expand_bn[
0][0]']
 (Activation)

 block5d_dwconv2 (Depthwise  (None, 3, 3, 720)         6480       ['block5d_expand_acti
vation[0]
 Conv2D)                                                          [0]']

 block5d_bn (BatchNormaliza  (None, 3, 3, 720)         2880       ['block5d_dwconv2[0][
0]']
 tion)

 block5d_activation (Activa  (None, 3, 3, 720)         0          ['block5d_bn[0][0]']

 tion)

 block5d_se_squeeze (Global  (None, 720)               0          ['block5d_activation
[0][0]']
 AveragePooling2D)

 block5d_se_reshape (Reshap  (None, 1, 1, 720)         0          ['block5d_se_squeeze[
0][0]']
 e)

 block5d_se_reduce (Conv2D)  (None, 1, 1, 30)          21630      ['block5d_se_reshape[
0][0]']

 block5d_se_expand (Conv2D)  (None, 1, 1, 720)         22320      ['block5d_se_reduce[0
][0]']

 block5d_se_excite (Multipl  (None, 3, 3, 720)         0          ['block5d_activation[
0][0]',
 y)                                                                'block5d_se_expand
[0][0]']

 block5d_project_conv (Conv  (None, 3, 3, 120)         86400      ['block5d_se_excite[0
][0]']
 2D)

 block5d_project_bn (BatchN  (None, 3, 3, 120)         480        ['block5d_project_con
v[0][0]']
 ormalization)

 block5d_drop (Dropout)      (None, 3, 3, 120)         0          ['block5d_project_bn
 block5d_drop (Dropout)      (None, 3, 3, 120)            0         ['block5d_project_bn
[0][0]']

 block5d_add (Add)           (None, 3, 3, 120)            0         ['block5d_drop[0][0]
',
                                                                     'block5c_add[0][0]
']

 block5e_expand_conv (Conv2  (None, 3, 3, 720)            86400     ['block5d_add[0][0]']

 D)


 block5e_expand_bn (BatchNo  (None, 3, 3, 720)            2880      ['block5e_expand_conv
[0][0]']
 rmalization)


 block5e_expand_activation   (None, 3, 3, 720)            0         ['block5e_expand_bn[
0][0]']
 (Activation)


 block5e_dwconv2 (Depthwise  (None, 3, 3, 720)            6480      ['block5e_expand_acti
vation[0]
 Conv2D)                                                            [0]']


 block5e_bn (BatchNormaliza  (None, 3, 3, 720)            2880      ['block5e_dwconv2[0][
0]']
 tion)


 block5e_activation (Activa  (None, 3, 3, 720)            0         ['block5e_bn[0][0]']

 tion)


 block5e_se_squeeze (Global  (None, 720)                  0         ['block5e_activation
[0][0]']
 AveragePooling2D)


 block5e_se_reshape (Reshap  (None, 1, 1, 720)            0         ['block5e_se_squeeze[
0][0]']
 e)


 block5e_se_reduce (Conv2D)  (None, 1, 1, 30)             21630     ['block5e_se_reshape[
0][0]']

 block5e_se_expand (Conv2D)  (None, 1, 1, 720)            22320     ['block5e_se_reduce[0
][0]']

 block5e_se_excite (Multipl  (None, 3, 3, 720)            0         ['block5e_activation[
0][0]',
 y)                                                                  'block5e_se_expand
[0][0]']

 block5e_project_conv (Conv  (None, 3, 3, 120)            86400     ['block5e_se_excite[0
 block5e_project_conv (Conv  (None, 3, 3, 120)            86400     ['block5e_se_excite[0
][0]']
 2D)


 block5e_project_bn (BatchN  (None, 3, 3, 120)            480       ['block5e_project_con
v[0][0]']
 ormalization)


 block5e_drop (Dropout)      (None, 3, 3, 120)            0         ['block5e_project_bn
[0][0]']

 block5e_add (Add)           (None, 3, 3, 120)            0         ['block5e_drop[0][0]
',
                                                                     'block5d_add[0][0]
']

 block5f_expand_conv (Conv2  (None, 3, 3, 720)            86400     ['block5e_add[0][0]']

 D)


 block5f_expand_bn (BatchNo  (None, 3, 3, 720)            2880      ['block5f_expand_conv
[0][0]']
 rmalization)


 block5f_expand_activation   (None, 3, 3, 720)            0         ['block5f_expand_bn[
0][0]']
  (Activation)


 block5f_dwconv2 (Depthwise  (None, 3, 3, 720)            6480      ['block5f_expand_acti
vation[0]
 Conv2D)                                                            [0]']


 block5f_bn (BatchNormaliza  (None, 3, 3, 720)            2880      ['block5f_dwconv2[0][
0]']
 tion)


 block5f_activation (Activa  (None, 3, 3, 720)            0         ['block5f_bn[0][0]']

 tion)


 block5f_se_squeeze (Global  (None, 720)                  0         ['block5f_activation
[0][0]']
 AveragePooling2D)


 block5f_se_reshape (Reshap  (None, 1, 1, 720)            0         ['block5f_se_squeeze[
0][0]']
 e)


 block5f_se_reduce (Conv2D)  (None, 1, 1, 30)             21630     ['block5f_se_reshape[
0][0]']
```

```
 block5f_se_expand (Conv2D)  (None, 1, 1, 720)            22320     ['block5f_se_reduce[0
][0]']

 block5f_se_excite (Multipl  (None, 3, 3, 720)            0         ['block5f_activation[
0][0]',
 y)                                                                  'block5f_se_expand
[0][0]']

 block5f_project_conv (Conv  (None, 3, 3, 120)            86400     ['block5f_se_excite[0
][0]']
 2D)


 block5f_project_bn (BatchN  (None, 3, 3, 120)            480       ['block5f_project_con
v[0][0]']
 ormalization)


 block5f_drop (Dropout)      (None, 3, 3, 120)            0         ['block5f_project_bn
[0][0]']

 block5f_add (Add)           (None, 3, 3, 120)            0         ['block5f_drop[0][0]
',
                                                                     'block5e_add[0][0]
']

 block6a_expand_conv (Conv2  (None, 3, 3, 720)            86400     ['block5f_add[0][0]']

 D)


 block6a_expand_bn (BatchNo  (None, 3, 3, 720)            2880      ['block6a_expand_conv
[0][0]']
 rmalization)


 block6a_expand_activation   (None, 3, 3, 720)            0         ['block6a_expand_bn[
0][0]']
 (Activation)


 block6a_dwconv2 (Depthwise  (None, 2, 2, 720)            6480      ['block6a_expand_acti
vation[0]
 Conv2D)                                                            [0]']


 block6a_bn (BatchNormaliza  (None, 2, 2, 720)            2880      ['block6a_dwconv2[0][
0]']
 tion)


 block6a_activation (Activa  (None, 2, 2, 720)            0         ['block6a_bn[0][0]']

 tion)


 block6a_se_squeeze (Global  (None, 720)                  0         ['block6a_activation
[0][0]']
 AveragePooling2D)
 AveragePooling2D)


 block6a_se_reshape (Reshap  (None, 1, 1, 720)           0          ['block6a_se_squeeze[
0][0]']
 e)


 block6a_se_reduce (Conv2D)  (None, 1, 1, 30)            21630      ['block6a_se_reshape[
0][0]']

 block6a_se_expand (Conv2D)  (None, 1, 1, 720)           22320      ['block6a_se_reduce[0
][0]']

 block6a_se_excite (Multipl  (None, 2, 2, 720)           0          ['block6a_activation[
0][0]',
 y)                                                                   'block6a_se_expand
[0][0]']

 block6a_project_conv (Conv  (None, 2, 2, 208)           149760     ['block6a_se_excite[0
][0]']
 2D)


 block6a_project_bn (BatchN  (None, 2, 2, 208)           832        ['block6a_project_con
v[0][0]']
 ormalization)


 block6b_expand_conv (Conv2  (None, 2, 2, 1248)          259584     ['block6a_project_bn[
0][0]']
 D)


 block6b_expand_bn (BatchNo  (None, 2, 2, 1248)          4992       ['block6b_expand_conv
[0][0]']
 rmalization)


 block6b_expand_activation   (None, 2, 2, 1248)          0          ['block6b_expand_bn[
0][0]']
 (Activation)


 block6b_dwconv2 (Depthwise  (None, 2, 2, 1248)          11232      ['block6b_expand_acti
vation[0]
 Conv2D)                                                            [0]']


 block6b_bn (BatchNormaliza  (None, 2, 2, 1248)          4992       ['block6b_dwconv2[0][
0]']
 tion)


 block6b_activation (Activa  (None, 2, 2, 1248)          0          ['block6b_bn[0][0]']

 tion)


 block6b_se_squeeze (Global  (None, 1248)                0          ['block6b_activation
 block6b_se_squeeze (Global  (None, 1248)                 0         ['block6b_activation
[0][0]']
 AveragePooling2D)


 block6b_se_reshape (Reshap  (None, 1, 1, 1248)           0         ['block6b_se_squeeze[
0][0]']
 e)


 block6b_se_reduce (Conv2D)  (None, 1, 1, 52)             64948     ['block6b_se_reshape[
0][0]']

 block6b_se_expand (Conv2D)  (None, 1, 1, 1248)           66144     ['block6b_se_reduce[0
][0]']

 block6b_se_excite (Multipl  (None, 2, 2, 1248)           0         ['block6b_activation[
0][0]',
 y)                                                                  'block6b_se_expand
[0][0]']

 block6b_project_conv (Conv  (None, 2, 2, 208)            259584    ['block6b_se_excite[0
][0]']
 2D)


 block6b_project_bn (BatchN  (None, 2, 2, 208)            832       ['block6b_project_con
v[0][0]']
 ormalization)


 block6b_drop (Dropout)      (None, 2, 2, 208)            0         ['block6b_project_bn
[0][0]']

 block6b_add (Add)           (None, 2, 2, 208)            0         ['block6b_drop[0][0]
',
                                                                     'block6a_project_b
n[0][0]']

 block6c_expand_conv (Conv2  (None, 2, 2, 1248)           259584    ['block6b_add[0][0]']

 D)


 block6c_expand_bn (BatchNo  (None, 2, 2, 1248)           4992      ['block6c_expand_conv
[0][0]']
 rmalization)


 block6c_expand_activation   (None, 2, 2, 1248)           0         ['block6c_expand_bn[
0][0]']
 (Activation)


 block6c_dwconv2 (Depthwise  (None, 2, 2, 1248)           11232     ['block6c_expand_acti
vation[0]
 Conv2D)                                                            [0]']


 block6c_bn (BatchNormaliza  (None, 2, 2, 1248)           4992      ['block6c_dwconv2[0][
 block6c_bn (BatchNormaliza  (None, 2, 2, 1248)           4992      ['block6c_dwconv2[0][
0]']
 tion)



 block6c_activation (Activa  (None, 2, 2, 1248)           0         ['block6c_bn[0][0]']

 tion)



 block6c_se_squeeze (Global  (None, 1248)                 0         ['block6c_activation
[0][0]']
 AveragePooling2D)



 block6c_se_reshape (Reshap  (None, 1, 1, 1248)           0         ['block6c_se_squeeze[
0][0]']
 e)



 block6c_se_reduce (Conv2D)  (None, 1, 1, 52)             64948     ['block6c_se_reshape[
0][0]']


 block6c_se_expand (Conv2D)  (None, 1, 1, 1248)           66144     ['block6c_se_reduce[0
][0]']


 block6c_se_excite (Multipl  (None, 2, 2, 1248)           0         ['block6c_activation[
0][0]',
 y)                                                                  'block6c_se_expand
[0][0]']


 block6c_project_conv (Conv  (None, 2, 2, 208)            259584    ['block6c_se_excite[0
][0]']
 2D)



 block6c_project_bn (BatchN  (None, 2, 2, 208)            832       ['block6c_project_con
v[0][0]']
 ormalization)



 block6c_drop (Dropout)      (None, 2, 2, 208)            0         ['block6c_project_bn
[0][0]']


 block6c_add (Add)           (None, 2, 2, 208)            0         ['block6c_drop[0][0]
',
                                                                     'block6b_add[0][0]
']


 block6d_expand_conv (Conv2  (None, 2, 2, 1248)           259584    ['block6c_add[0][0]']

 D)



 block6d_expand_bn (BatchNo  (None, 2, 2, 1248)           4992      ['block6d_expand_conv
[0][0]']
 rmalization)



 block6d_expand_activation   (None, 2, 2, 1248)           0         ['block6d_expand_bn[
 block6d_expand_activation    (None, 2, 2, 1248)          0         ['block6d_expand_bn[
0][0]']
 (Activation)



 block6d_dwconv2 (Depthwise  (None, 2, 2, 1248)           11232     ['block6d_expand_acti
vation[0]
 Conv2D)                                                            [0]']



 block6d_bn (BatchNormaliza  (None, 2, 2, 1248)           4992      ['block6d_dwconv2[0][
0]']
 tion)



 block6d_activation (Activa  (None, 2, 2, 1248)           0         ['block6d_bn[0][0]']

 tion)



 block6d_se_squeeze (Global  (None, 1248)                 0         ['block6d_activation
[0][0]']
 AveragePooling2D)



 block6d_se_reshape (Reshap  (None, 1, 1, 1248)           0         ['block6d_se_squeeze[
0][0]']
 e)



 block6d_se_reduce (Conv2D)  (None, 1, 1, 52)             64948     ['block6d_se_reshape[
0][0]']

 block6d_se_expand (Conv2D)  (None, 1, 1, 1248)           66144     ['block6d_se_reduce[0
][0]']

 block6d_se_excite (Multipl  (None, 2, 2, 1248)           0         ['block6d_activation[
0][0]',
 y)                                                                  'block6d_se_expand
[0][0]']

 block6d_project_conv (Conv  (None, 2, 2, 208)            259584    ['block6d_se_excite[0
][0]']
 2D)



 block6d_project_bn (BatchN  (None, 2, 2, 208)            832       ['block6d_project_con
v[0][0]']
 ormalization)



 block6d_drop (Dropout)      (None, 2, 2, 208)            0         ['block6d_project_bn
[0][0]']

 block6d_add (Add)           (None, 2, 2, 208)            0         ['block6d_drop[0][0]
',
                                                                     'block6c_add[0][0]
']

 block6e_expand_conv (Conv2  (None, 2, 2, 1248)           259584    ['block6d_add[0][0]']
 block6e_expand_conv (Conv2  (None, 2, 2, 1248)           259584    ['block6d_add[0][0]']

 D)


 block6e_expand_bn (BatchNo  (None, 2, 2, 1248)           4992      ['block6e_expand_conv
[0][0]']
 rmalization)


 block6e_expand_activation   (None, 2, 2, 1248)           0         ['block6e_expand_bn[
0][0]']
 (Activation)


 block6e_dwconv2 (Depthwise  (None, 2, 2, 1248)           11232     ['block6e_expand_acti
vation[0]
 Conv2D)                                                            [0]']


 block6e_bn (BatchNormaliza  (None, 2, 2, 1248)           4992      ['block6e_dwconv2[0][
0]']
 tion)


 block6e_activation (Activa  (None, 2, 2, 1248)           0         ['block6e_bn[0][0]']

 tion)


 block6e_se_squeeze (Global  (None, 1248)                 0         ['block6e_activation
[0][0]']
 AveragePooling2D)


 block6e_se_reshape (Reshap  (None, 1, 1, 1248)           0         ['block6e_se_squeeze[
0][0]']
 e)


 block6e_se_reduce (Conv2D)  (None, 1, 1, 52)             64948     ['block6e_se_reshape[
0][0]']

 block6e_se_expand (Conv2D)  (None, 1, 1, 1248)           66144     ['block6e_se_reduce[0
][0]']

 block6e_se_excite (Multipl  (None, 2, 2, 1248)           0         ['block6e_activation[
0][0]',
 y)                                                                  'block6e_se_expand
[0][0]']

 block6e_project_conv (Conv  (None, 2, 2, 208)            259584    ['block6e_se_excite[0
][0]']
 2D)


 block6e_project_bn (BatchN  (None, 2, 2, 208)            832       ['block6e_project_con
v[0][0]']
 ormalization)
 block6e_drop (Dropout)      (None, 2, 2, 208)            0          ['block6e_project_bn
[0][0]']

 block6e_add (Add)           (None, 2, 2, 208)            0          ['block6e_drop[0][0]
',
                                                                      'block6d_add[0][0]
']

 block6f_expand_conv (Conv2  (None, 2, 2, 1248)           259584     ['block6e_add[0][0]']

 D)


 block6f_expand_bn (BatchNo  (None, 2, 2, 1248)           4992       ['block6f_expand_conv
[0][0]']
 rmalization)


 block6f_expand_activation   (None, 2, 2, 1248)           0          ['block6f_expand_bn[
0][0]']
 (Activation)


 block6f_dwconv2 (Depthwise  (None, 2, 2, 1248)           11232      ['block6f_expand_acti
vation[0]
 Conv2D)                                                             [0]']


 block6f_bn (BatchNormaliza  (None, 2, 2, 1248)           4992       ['block6f_dwconv2[0][
0]']
 tion)


 block6f_activation (Activa  (None, 2, 2, 1248)           0          ['block6f_bn[0][0]']

 tion)


 block6f_se_squeeze (Global  (None, 1248)                 0          ['block6f_activation
[0][0]']
 AveragePooling2D)


 block6f_se_reshape (Reshap  (None, 1, 1, 1248)           0          ['block6f_se_squeeze[
0][0]']
 e)


 block6f_se_reduce (Conv2D)  (None, 1, 1, 52)             64948      ['block6f_se_reshape[
0][0]']

 block6f_se_expand (Conv2D)  (None, 1, 1, 1248)           66144      ['block6f_se_reduce[0
][0]']

 block6f_se_excite (Multipl  (None, 2, 2, 1248)           0          ['block6f_activation[
0][0]',
 y)                                                                   'block6f_se_expand
[0][0]']
```

```
 block6f_project_conv (Conv  (None, 2, 2, 208)            259584    ['block6f_se_excite[0
][0]']
 2D)


 block6f_project_bn (BatchN  (None, 2, 2, 208)            832       ['block6f_project_con
v[0][0]']
 ormalization)


 block6f_drop (Dropout)      (None, 2, 2, 208)            0         ['block6f_project_bn
[0][0]']

 block6f_add (Add)           (None, 2, 2, 208)            0         ['block6f_drop[0][0]
',
                                                                      'block6e_add[0][0]
']

 block6g_expand_conv (Conv2  (None, 2, 2, 1248)           259584    ['block6f_add[0][0]']

 D)


 block6g_expand_bn (BatchNo  (None, 2, 2, 1248)           4992      ['block6g_expand_conv
[0][0]']
 rmalization)


 block6g_expand_activation   (None, 2, 2, 1248)           0         ['block6g_expand_bn[
0][0]']
 (Activation)


 block6g_dwconv2 (Depthwise  (None, 2, 2, 1248)           11232     ['block6g_expand_acti
vation[0]
 Conv2D)                                                            [0]']


 block6g_bn (BatchNormaliza  (None, 2, 2, 1248)           4992      ['block6g_dwconv2[0][
0]']
 tion)


 block6g_activation (Activa  (None, 2, 2, 1248)           0         ['block6g_bn[0][0]']

 tion)


 block6g_se_squeeze (Global  (None, 1248)                 0         ['block6g_activation
[0][0]']
 AveragePooling2D)


 block6g_se_reshape (Reshap  (None, 1, 1, 1248)           0         ['block6g_se_squeeze[
0][0]']
 e)


 block6g_se_reduce (Conv2D)  (None, 1, 1, 52)             64948     ['block6g_se_reshape[
 block6g_se_reduce (Conv2D)  (None, 1, 1, 52)             64948     ['block6g_se_reshape[
0][0]']


 block6g_se_expand (Conv2D)  (None, 1, 1, 1248)           66144     ['block6g_se_reduce[0
]][0]']


 block6g_se_excite (Multipl  (None, 2, 2, 1248)           0         ['block6g_activation[
0][0]',
 y)                                                                  'block6g_se_expand
[0][0]']


 block6g_project_conv (Conv  (None, 2, 2, 208)            259584    ['block6g_se_excite[0
]][0]']
 2D)



 block6g_project_bn (BatchN  (None, 2, 2, 208)            832       ['block6g_project_con
v[0][0]']
 ormalization)



 block6g_drop (Dropout)      (None, 2, 2, 208)            0         ['block6g_project_bn
[0][0]']


 block6g_add (Add)           (None, 2, 2, 208)            0         ['block6g_drop[0][0]
',
                                                                     'block6f_add[0][0]
']


 block6h_expand_conv (Conv2  (None, 2, 2, 1248)           259584    ['block6g_add[0][0]']

 D)



 block6h_expand_bn (BatchNo  (None, 2, 2, 1248)           4992      ['block6h_expand_conv
[0][0]']
 rmalization)



 block6h_expand_activation   (None, 2, 2, 1248)           0         ['block6h_expand_bn[
0][0]']
 (Activation)



 block6h_dwconv2 (Depthwise  (None, 2, 2, 1248)           11232     ['block6h_expand_acti
vation[0]
 Conv2D)                                                            [0]']



 block6h_bn (BatchNormaliza  (None, 2, 2, 1248)           4992      ['block6h_dwconv2[0][
0]']
 tion)



 block6h_activation (Activa  (None, 2, 2, 1248)           0         ['block6h_bn[0][0]']

 tion)



 block6h_se_squeeze (Global  (None, 1248)                 0         ['block6h_activation
 block6h_se_squeeze (Global  (None, 1248)                 0          ['block6h_activation
[0][0]']
 AveragePooling2D)


 block6h_se_reshape (Reshap  (None, 1, 1, 1248)           0          ['block6h_se_squeeze[
0][0]']
 e)


 block6h_se_reduce (Conv2D)  (None, 1, 1, 52)             64948      ['block6h_se_reshape[
0][0]']

 block6h_se_expand (Conv2D)  (None, 1, 1, 1248)           66144      ['block6h_se_reduce[0
][0]']

 block6h_se_excite (Multipl  (None, 2, 2, 1248)           0          ['block6h_activation[
0][0]',
 y)                                                                   'block6h_se_expand
[0][0]']

 block6h_project_conv (Conv  (None, 2, 2, 208)            259584     ['block6h_se_excite[0
][0]']
 2D)


 block6h_project_bn (BatchN  (None, 2, 2, 208)            832        ['block6h_project_con
v[0][0]']
 ormalization)


 block6h_drop (Dropout)      (None, 2, 2, 208)            0          ['block6h_project_bn
[0][0]']

 block6h_add (Add)           (None, 2, 2, 208)            0          ['block6h_drop[0][0]
',
                                                                      'block6g_add[0][0]
']

 block6i_expand_conv (Conv2  (None, 2, 2, 1248)           259584     ['block6h_add[0][0]']

 D)


 block6i_expand_bn (BatchNo  (None, 2, 2, 1248)           4992       ['block6i_expand_conv
[0][0]']
 rmalization)


 block6i_expand_activation   (None, 2, 2, 1248)           0          ['block6i_expand_bn[
0][0]']
 (Activation)


 block6i_dwconv2 (Depthwise  (None, 2, 2, 1248)           11232      ['block6i_expand_acti
vation[0]
 Conv2D)                                                             [0]']


 block6i_bn (BatchNormaliza  (None, 2, 2, 1248)           4992       ['block6i_dwconv2[0][
 block6i_bn (BatchNormaliza  (None, 2, 2, 1248)            4992      ['block6i_dwconv2[0][
0]']
 tion)


 block6i_activation (Activa  (None, 2, 2, 1248)            0         ['block6i_bn[0][0]']

 tion)


 block6i_se_squeeze (Global  (None, 1248)                  0         ['block6i_activation
[0][0]']
 AveragePooling2D)


 block6i_se_reshape (Reshap  (None, 1, 1, 1248)            0         ['block6i_se_squeeze[
0][0]']
 e)


 block6i_se_reduce (Conv2D)  (None, 1, 1, 52)              64948     ['block6i_se_reshape[
0][0]']

 block6i_se_expand (Conv2D)  (None, 1, 1, 1248)            66144     ['block6i_se_reduce[0
][0]']

 block6i_se_excite (Multipl  (None, 2, 2, 1248)            0         ['block6i_activation[
0][0]',
 y)                                                                   'block6i_se_expand
[0][0]']

 block6i_project_conv (Conv  (None, 2, 2, 208)             259584    ['block6i_se_excite[0
][0]']
 2D)


 block6i_project_bn (BatchN  (None, 2, 2, 208)             832       ['block6i_project_con
v[0][0]']
 ormalization)


 block6i_drop (Dropout)      (None, 2, 2, 208)             0         ['block6i_project_bn
[0][0]']

 block6i_add (Add)           (None, 2, 2, 208)             0         ['block6i_drop[0][0]
',
                                                                      'block6h_add[0][0]
']

 block6j_expand_conv (Conv2  (None, 2, 2, 1248)            259584    ['block6i_add[0][0]']

 D)


 block6j_expand_bn (BatchNo  (None, 2, 2, 1248)            4992      ['block6j_expand_conv
[0][0]']
 rmalization)


 block6j_expand_activation   (None, 2, 2, 1248)            0         ['block6j_expand_bn[
 block6j_expand_activation   (None, 2, 2, 1248)           0          ['block6j_expand_bn[
0][0]']
 (Activation)



 block6j_dwconv2 (Depthwise  (None, 2, 2, 1248)           11232      ['block6j_expand_acti
vation[0]
 Conv2D)                                                            [0]']



 block6j_bn (BatchNormaliza  (None, 2, 2, 1248)           4992       ['block6j_dwconv2[0][
0]']
 tion)



 block6j_activation (Activa  (None, 2, 2, 1248)           0          ['block6j_bn[0][0]']

 tion)



 block6j_se_squeeze (Global  (None, 1248)                 0          ['block6j_activation
[0][0]']
 AveragePooling2D)



 block6j_se_reshape (Reshap  (None, 1, 1, 1248)           0          ['block6j_se_squeeze[
0][0]']
 e)



 block6j_se_reduce (Conv2D)  (None, 1, 1, 52)             64948      ['block6j_se_reshape[
0][0]']

 block6j_se_expand (Conv2D)  (None, 1, 1, 1248)           66144      ['block6j_se_reduce[0
][0]']

 block6j_se_excite (Multipl  (None, 2, 2, 1248)           0          ['block6j_activation[
0][0]',
 y)                                                                   'block6j_se_expand
[0][0]']

 block6j_project_conv (Conv  (None, 2, 2, 208)            259584     ['block6j_se_excite[0
][0]']
 2D)



 block6j_project_bn (BatchN  (None, 2, 2, 208)            832        ['block6j_project_con
v[0][0]']
 ormalization)



 block6j_drop (Dropout)      (None, 2, 2, 208)            0          ['block6j_project_bn
[0][0]']

 block6j_add (Add)           (None, 2, 2, 208)            0          ['block6j_drop[0][0]
',
                                                                      'block6i_add[0][0]
']

 top_conv (Conv2D)           (None, 2, 2, 1408)           292864     ['block6j_add[0][0]'
 top_conv (Conv2D)           (None, 2, 2, 1408)           292864    ['block6j_add[0][0]
]


 top_bn (BatchNormalization  (None, 2, 2, 1408)           5632      ['top_conv[0][0]']

 )



 top_activation (Activation  (None, 2, 2, 1408)           0         ['top_bn[0][0]']

 )



==================================================================================================
=========
Total params: 8769374 (33.45 MB)
Trainable params: 8687086 (33.14 MB)
Non-trainable params: 82288 (321.44 KB)
__________________________________________________________________________________________________
_________
```

Model Building

Build your own Architecture on top of the transfer layer. Be sure to have a Flatten layer after your transfer layer and also make sure you have 4 neurons and softmax activation function in your last dense layer

In [ ]:

```
from tensorflow.keras.applications import EfficientNetB0
EfficientNet = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(224, 2
24, 3))
transfer_layer_EfficientNet = EfficientNet.get_layer('block6a_expand_activation').output
EfficientNet.trainable = False

x = GlobalAveragePooling2D()(transfer_layer_EfficientNet)

# Add your Dense layers and/or BatchNormalization and Dropout layers
x = Dense(256, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)

x = Dense(128, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)

# Add your final Dense layer with 4 neurons and softmax activation function.
pred = Dense(4, activation='softmax')(x)

Efficientnetmodel = Model(inputs=EfficientNet.input, outputs=pred)
```

Compiling and Training the Model

In [ ]:

```
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

checkpoint = ModelCheckpoint(
    "./Efficientnetmodel.h5",
    monitor='val_accuracy',
    verbose=1,
    save_best_only=True,
    mode='max'
)

early_stopping = EarlyStopping(
    monitor='val_loss',
    min_delta=0.001,
    patience=5,
    verbose=1,
    mode='min',
    restore_best_weights=True
)

reduce_learningrate = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.2,
    patience=3,
    verbose=1,
    mode='min',
    min_delta=0.0001,
    cooldown=0,
    min_lr=0
)
callbacks_list = [early_stopping,checkpoint,reduce_learningrate]

epochs = 10
```

In [ ]:

```
# Write your code to compile your Efficientnetmodel. Use categorical crossentropy as your
loss function, Adam Optimizer with 0.001 learning rate, and set your metrics to 'accuracy
'.
Efficientnetmodel.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

In [ ]:

```
history = Efficientnetmodel.fit(
    train_set,
    epochs=20,
    validation_data=validation_set,
    callbacks=callbacks_list,
    verbose=1
)
```

```
Epoch 1/20
473/473 [==============================] - ETA: 0s - loss: 1.7580 - accuracy: 0.2562
Epoch 1: val_accuracy improved from -inf to 0.36669, saving model to ./Efficientnetmodel.
h5
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: Y
ou are saving your model as an HDF5 file via `model.save()`. This file format is consider
ed legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model
.keras')`.
  saving_api.save_model(
```

```
473/473 [==============================] - 90s 177ms/step - loss: 1.7580 - accuracy: 0.25
62 - val_loss: 1.3756 - val_accuracy: 0.3667 - lr: 0.0010
Epoch 2/20
473/473 [==============================] - ETA: 0s - loss: 1.4844 - accuracy: 0.2597
Epoch 2: val_accuracy did not improve from 0.36669
473/473 [==============================] - 82s 173ms/step - loss: 1.4844 - accuracy: 0.25
97 - val_loss: 1.3499 - val_accuracy: 0.3667 - lr: 0.0010
Epoch 3/20
473/473 [==============================] - ETA: 0s - loss: 1.4111 - accuracy: 0.2651
Epoch 3: val_accuracy did not improve from 0.36669
473/473 [==============================] - 81s 172ms/step - loss: 1.4111 - accuracy: 0.26
51 - val_loss: 1.3670 - val_accuracy: 0.2871 - lr: 0.0010
Epoch 4/20
473/473 [==============================] - ETA: 0s - loss: 1.3964 - accuracy: 0.2677
Epoch 4: val_accuracy did not improve from 0.36669
473/473 [==============================] - 81s 172ms/step - loss: 1.3964 - accuracy: 0.26
77 - val_loss: 1.3679 - val_accuracy: 0.3665 - lr: 0.0010
Epoch 5/20
473/473 [==============================] - ETA: 0s - loss: 1.3943 - accuracy: 0.2662
```

```
Epoch 5: val_accuracy did not improve from 0.36669

Epoch 5: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
473/473 [==============================] - 81s 172ms/step - loss: 1.3943 - accuracy: 0.26
62 - val_loss: 1.3700 - val_accuracy: 0.3667 - lr: 0.0010
Epoch 6/20
473/473 [==============================] - ETA: 0s - loss: 1.3879 - accuracy: 0.2637
Epoch 6: val_accuracy did not improve from 0.36669
473/473 [==============================] - 77s 162ms/step - loss: 1.3879 - accuracy: 0.26
37 - val_loss: 1.3651 - val_accuracy: 0.3667 - lr: 2.0000e-04
Epoch 7/20
473/473 [==============================] - ETA: 0s - loss: 1.3862 - accuracy: 0.2646Resto
ring model weights from the end of the best epoch: 2.

Epoch 7: val_accuracy did not improve from 0.36669
473/473 [==============================] - 74s 156ms/step - loss: 1.3862 - accuracy: 0.26
46 - val_loss: 1.3670 - val_accuracy: 0.3603 - lr: 2.0000e-04
Epoch 7: early stopping
```

Evaluating the EfficientnetNet Model

In [ ]:

```
# Write your code to evaluate the model performance on the test set
test_loss, test_accuracy = Efficientnetmodel.evaluate(test_set)

# Print the test loss and accuracy
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
4/4 [==============================] - 1s 130ms/step - loss: 1.4037 - accuracy: 0.2500
Test Loss: 1.4037209749221802
Test Accuracy: 0.25
```

Observations and Insights:

The test results, showing a 25% accuracy and a loss of 1.4037, highlight that our model's essentially guessing, hinting we've got some work to do. It seems like we're facing overfitting, as the model's ace on training data but flops on unseen data. Plus, it looks like our model architecture and data handling need a rethink.

Here’s the game plan to boost our model:

Dial Down Overfitting: Let's beef up dropout and regularization, maybe even tone down the model complexity to keep it from memorizing training data. Amp Up Data Augmentation: More variety in training data through enhanced augmentation could help the model learn more general patterns. Optimize Learning Rate: Adjusting the learning rate or using a scheduler might just hit the sweet spot for better learning without overshooting or dragging. Revamp the Architecture: Time to play around with the model structure—layer counts, types, and neuron numbers—to find what clicks for this task. Fine-Tune and Analyze Errors: If we've got layers frozen, thawing some for fine-tuning could be beneficial. Also, a deep dive into the errors to see if there's a pattern in the misfires. Balance the Classes: Our dataset might be skewing the model's learning; class weighting or resampling could level the playing field. Test Other Models: EfficientNet's cool, but maybe another pre-trained model like ResNet or InceptionV3 could be a better fit. Worth a shot. Broaden Evaluation Metrics: Beyond accuracy, checking precision, recall, and F1 scores can give us a fuller picture of where the model stands. In short, we're looking at a mix of fine-tuning, architecture tweaks, and smarter data handling to get our model back on track. Let's iterate and see how these changes play out!

Think About It:

- **What is your overall performance of these Transfer Learning Architectures? Can we draw a comparison of these models' performances. Are we satisfied with the accuracies that we have received?**
- **Do you think our issue lies with 'rgb' color_mode?**

Now that we have tried multiple pre-trained models, let's build a complex CNN architecture and see if we can get better performance.

Building a Complex Neural Network Architecture

In this section, we will build a more complex Convolutional Neural Network Model that has close to as many parameters as we had in our Transfer Learning Models. However, we will have only 1 input channel for our input images.

Creating our Data Loaders

In this section, we are creating data loaders which we will use as inputs to the more Complicated Convolutional Neural Network. We will go ahead with color_mode = 'grayscale'.

In [ ]:

```
batch_size  = 32
img_size = 48

datagen_train = ImageDataGenerator(horizontal_flip = True,
                                    brightness_range = (0., 2.),
                                    rescale = 1./255,
                                    shear_range = 0.3)

train_set = datagen_train.flow_from_directory(folder_path + "train",
                                              target_size = (img_size, img_size),
                                              color_mode = 'grayscale',
                                              batch_size = batch_size,
                                              class_mode = 'categorical',
                                              classes = ['happy', 'sad', 'neutral', 'su
rprise'],

                                              shuffle = True)


datagen_validation = ImageDataGenerator(rescale=1./255)

validation_set = datagen_validation.flow_from_directory(
    folder_path + "validation",
    target_size=(img_size, img_size),
    color_mode='grayscale',
    batch_size=batch_size,
    class_mode='categorical',
    classes=['happy', 'sad', 'neutral', 'surprise'],
    shuffle=True
)

datagen_test = ImageDataGenerator(rescale=1./255)

test_set = datagen_test.flow_from_directory(
    folder_path + "test",
    target_size=(img_size, img_size),
    color_mode='grayscale',
    batch_size=batch_size,
    class_mode='categorical',
    classes=['happy', 'sad', 'neutral', 'surprise'],
    shuffle=False
)
```

```
Found 15109 images belonging to 4 classes.
Found 4977 images belonging to 4 classes.
Found 128 images belonging to 4 classes.
```

Model Building

- **In this network, we plan to have 5 Convolutional Blocks**
- **Add first Conv2D layer with 64 filters and a kernel size of 2. Use the 'same' padding and provide the input shape = (48, 48, 1). Use 'relu' activation.**
- **Add your BatchNormalization layer followed by a LeakyRelU layer with Leaky ReLU parameter of 0.1**
- **Add MaxPooling2D layer with pool size = 2**

- Add MaxPooling2D layer with **pool size = 2**.
- Add a Dropout layer with a Dropout Ratio of **0.2**. This completes the first Convolutional block.
- Add a second Conv2D layer with **128 filters** and a **kernel size of 2**. Use the **'same' padding** and **'relu' activation.**
- Follow this up with a similar BatchNormalization, LeakyRelU, Maxpooling2D, and Dropout layer like above to complete your second Convolutional Block.
- Add a third Conv2D layer with **512 filters** and a **kernel size of 2**. Use the **'same' padding** and **'relu' activation.** Once again, follow it up with a BatchNormalization, LeakyRelU, Maxpooling2D, and Dropout layer to complete your third Convolutional block.
- Add a fourth block, with the Conv2D layer having **512 filters**.
- Add the fifth block, having **128 filters**.
- Then add your Flatten layer, followed by your Dense layers.
- Add your first Dense layer with **256 neurons** followed by a BatchNormalization layer, a **'relu'** Activation, and a Dropout layer. This forms your first Fully Connected block
- Add your second Dense layer with **512 neurons**, again followed by a BatchNormalization layer, **relu** activation, and a Dropout layer.
- Add your final Dense layer with 4 neurons.
- Compile your model with the optimizer of your choice.

In [ ]:

```
no_of_classes = 4

model3 = Sequential()

# Add 1st CNN Block
model3.add(Conv2D(64, kernel_size=2, padding='same', input_shape=(48, 48, 1), activation
='relu'))
model3.add(BatchNormalization())
model3.add(LeakyReLU(0.1))
model3.add(MaxPooling2D(pool_size=2))
model3.add(Dropout(0.2))

# Add 2nd CNN Block
model3.add(Conv2D(128, kernel_size=2, padding='same', activation='relu'))
model3.add(BatchNormalization())
model3.add(LeakyReLU(0.1))
model3.add(MaxPooling2D(pool_size=2))
model3.add(Dropout(0.2))

# Add 3rd CNN Block
model3.add(Conv2D(512, kernel_size=2, padding='same', activation='relu'))
model3.add(BatchNormalization())
model3.add(LeakyReLU(0.1))
model3.add(MaxPooling2D(pool_size=2))
model3.add(Dropout(0.3))

# Add 4th CNN Block
model3.add(Conv2D(512, kernel_size=2, padding='same', activation='relu'))
model3.add(BatchNormalization())
model3.add(LeakyReLU(0.1))
model3.add(MaxPooling2D(pool_size=2))
model3.add(Dropout(0.3))


# Add 5th CNN Block
model3.add(Conv2D(128, kernel_size=2, padding='same', activation='relu'))
model3.add(BatchNormalization())
model3.add(LeakyReLU(0.1))
model3.add(MaxPooling2D(pool_size=2))
model3.add(Dropout(0.4))

model3.add(Flatten())

# First fully connected layer
model3.add(Dense(256, activation='relu'))
model3.add(BatchNormalization())
model3.add(Dropout(0.5))
```

```
# Second fully connected layer
model3.add(Dense(512, activation='relu'))
model3.add(BatchNormalization())
model3.add(Dropout(0.5))

model3.add(Dense(no_of_classes, activation = 'softmax'))
```

Compiling and Training the Model

In [ ]:

```
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, CSVLogger

epochs = 35

steps_per_epoch = train_set.n//train_set.batch_size
validation_steps = validation_set.n//validation_set.batch_size

checkpoint = ModelCheckpoint("model3.h5", monitor = 'val_accuracy',
                            save_weights_only = True, model = 'max', verbose = 1)

reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.1, patience = 2, min_lr =
0.0001 , model = 'auto')

callbacks = [checkpoint, reduce_lr]
```

In [ ]:

```
# Write your code to compile your model3. Use categorical crossentropy as the loss functi
on, Adam Optimizer with 0.003 learning rate, and set metrics to 'accuracy'.
model3.compile(
    optimizer=Adam(learning_rate=0.003),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

In [ ]:

```
history = model3.fit(
    train_set,
    epochs=35,
    validation_data=validation_set,
    verbose=1
)
```

```
Epoch 1/35
473/473 [==============================] - 401s 838ms/step - loss: 1.6943 - accuracy: 0.2
636 - val_loss: 1.5614 - val_accuracy: 0.2485
Epoch 2/35
473/473 [==============================] - 388s 821ms/step - loss: 1.4958 - accuracy: 0.2
690 - val_loss: 1.4084 - val_accuracy: 0.2592
Epoch 3/35
473/473 [==============================] - 387s 818ms/step - loss: 1.4549 - accuracy: 0.2
737 - val_loss: 1.4141 - val_accuracy: 0.2680
Epoch 4/35
473/473 [==============================] - 393s 831ms/step - loss: 1.4405 - accuracy: 0.2
725 - val_loss: 1.4215 - val_accuracy: 0.2200
Epoch 5/35
473/473 [==============================] - 392s 829ms/step - loss: 1.4196 - accuracy: 0.2
751 - val_loss: 1.4030 - val_accuracy: 0.2727
Epoch 6/35
473/473 [==============================] - 376s 793ms/step - loss: 1.4096 - accuracy: 0.2
691 - val_loss: 1.3711 - val_accuracy: 0.2688
Epoch 7/35
473/473 [==============================] - 388s 820ms/step - loss: 1.3965 - accuracy: 0.2
702 - val_loss: 1.3789 - val_accuracy: 0.2837
Epoch 8/35
473/473 [==============================] - 387s 819ms/step - loss: 1.3867 - accuracy: 0.2
749 - val_loss: 1.3853 - val_accuracy: 0.3253
Epoch 9/35
```

```
473/473 [==============================] - 395s 835ms/step - loss: 1.3694 - accuracy: 0.2
969 - val_loss: 1.3162 - val_accuracy: 0.3048
Epoch 10/35
473/473 [==============================] - 389s 823ms/step - loss: 1.3356 - accuracy: 0.3
322 - val_loss: 1.2293 - val_accuracy: 0.4306
Epoch 11/35
473/473 [==============================] - 374s 791ms/step - loss: 1.2851 - accuracy: 0.3
654 - val_loss: 1.2371 - val_accuracy: 0.4235
Epoch 12/35
473/473 [==============================] - 389s 823ms/step - loss: 1.2480 - accuracy: 0.3
852 - val_loss: 1.8150 - val_accuracy: 0.2936
Epoch 13/35
473/473 [==============================] - 396s 836ms/step - loss: 1.2394 - accuracy: 0.3
965 - val_loss: 1.2021 - val_accuracy: 0.4893
Epoch 14/35
473/473 [==============================] - 393s 832ms/step - loss: 1.1968 - accuracy: 0.4
402 - val_loss: 1.0487 - val_accuracy: 0.5588
Epoch 15/35
473/473 [==============================] - 390s 825ms/step - loss: 1.1329 - accuracy: 0.4
876 - val_loss: 0.9332 - val_accuracy: 0.6229
Epoch 16/35
473/473 [==============================] - 389s 822ms/step - loss: 1.0649 - accuracy: 0.5
299 - val_loss: 0.9893 - val_accuracy: 0.5927
Epoch 17/35
473/473 [==============================] - 386s 817ms/step - loss: 1.0303 - accuracy: 0.5
524 - val_loss: 0.8797 - val_accuracy: 0.6283
Epoch 18/35
473/473 [==============================] - 369s 780ms/step - loss: 1.0112 - accuracy: 0.5
624 - val_loss: 0.8384 - val_accuracy: 0.6528
Epoch 19/35
473/473 [==============================] - 374s 790ms/step - loss: 0.9805 - accuracy: 0.5
767 - val_loss: 0.9014 - val_accuracy: 0.6170
Epoch 20/35
473/473 [==============================] - 386s 817ms/step - loss: 0.9673 - accuracy: 0.5
855 - val_loss: 0.7797 - val_accuracy: 0.6795
Epoch 21/35
473/473 [==============================] - 393s 832ms/step - loss: 0.9476 - accuracy: 0.5
965 - val_loss: 0.7682 - val_accuracy: 0.6942
Epoch 22/35
473/473 [==============================] - 384s 811ms/step - loss: 0.9344 - accuracy: 0.6
040 - val_loss: 0.7964 - val_accuracy: 0.6839
Epoch 23/35
473/473 [==============================] - 398s 842ms/step - loss: 0.9355 - accuracy: 0.5
990 - val_loss: 0.7794 - val_accuracy: 0.6829
Epoch 24/35
473/473 [==============================] - 400s 847ms/step - loss: 0.9167 - accuracy: 0.6
114 - val_loss: 0.8163 - val_accuracy: 0.6761
Epoch 25/35
473/473 [==============================] - 397s 839ms/step - loss: 0.8996 - accuracy: 0.6
232 - val_loss: 0.7276 - val_accuracy: 0.7010
Epoch 26/35
473/473 [==============================] - 374s 792ms/step - loss: 0.8914 - accuracy: 0.6
186 - val_loss: 0.7255 - val_accuracy: 0.7038
Epoch 27/35
473/473 [==============================] - 387s 818ms/step - loss: 0.8759 - accuracy: 0.6
340 - val_loss: 0.7028 - val_accuracy: 0.7193
Epoch 28/35
473/473 [==============================] - 393s 831ms/step - loss: 0.8661 - accuracy: 0.6
396 - val_loss: 0.7405 - val_accuracy: 0.7034
Epoch 29/35
473/473 [==============================] - 393s 830ms/step - loss: 0.8557 - accuracy: 0.6
468 - val_loss: 0.7796 - val_accuracy: 0.6685
Epoch 30/35
473/473 [==============================] - 377s 798ms/step - loss: 0.8532 - accuracy: 0.6
463 - val_loss: 0.6893 - val_accuracy: 0.7286
Epoch 31/35
473/473 [==============================] - 393s 831ms/step - loss: 0.8338 - accuracy: 0.6
562 - val_loss: 0.6888 - val_accuracy: 0.7259
Epoch 32/35
473/473 [==============================] - 391s 826ms/step - loss: 0.8378 - accuracy: 0.6
551 - val_loss: 0.7200 - val_accuracy: 0.7111
Epoch 33/35
```

```
473/473 [==============================] - 376s 795ms/step - loss: 0.8247 - accuracy: 0.6
556 - val_loss: 0.6918 - val_accuracy: 0.7225
Epoch 34/35
473/473 [==============================] - 396s 837ms/step - loss: 0.8240 - accuracy: 0.6
628 - val_loss: 0.6807 - val_accuracy: 0.7346
Epoch 35/35
473/473 [==============================] - 393s 832ms/step - loss: 0.8056 - accuracy: 0.6
700 - val_loss: 0.7076 - val_accuracy: 0.7149
```

Evaluating the Model on Test Set

In [ ]:

```
# Write your code to evaluate the model performance on the test set
test_loss, test_accuracy = model3.evaluate(test_set)

# Print the test loss and accuracy
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
4/4 [==============================] - 1s 145ms/step - loss: 1.3860 - accuracy: 0.2500
Test Loss: 1.3859734535217285
Test Accuracy: 0.25
```

Observations and Insights:

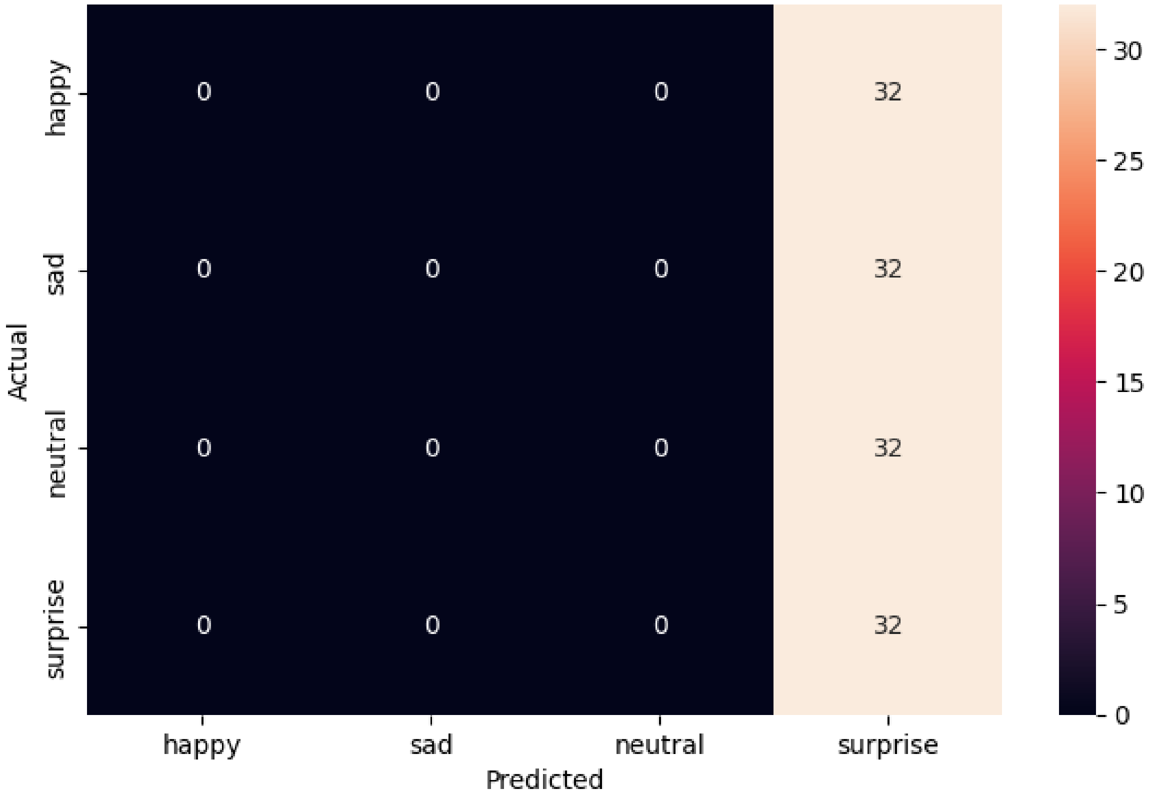
The model's test performance shows a loss of 1.386 and accuracy at 25%, essentially akin to random guessing across four classes, indicating it hasn't learned to differentiate them effectively. This could be due to overfitting, underfitting, inappropriate learning rate (0.003), or inadequate model architecture. Adjusting the learning rate, enhancing data preprocessing and augmentation, rebalancing the dataset for class imbalances, and tweaking the model architecture could improve results. It's also worth exploring different hyperparameters and considering transfer learning even for grayscale images. Utilizing confusion matrices and classification reports will provide deeper insights into class-specific performance, guiding targeted improvements.

Plotting the Confusion Matrix for the chosen final model

In [ ]:

```
# Plot the confusion matrix and generate a classification report for the model
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
test_set = datagen_test.flow_from_directory(folder_path + "test",
                                                              target_size = (img_size,i
mg_size),
                                                              color_mode = 'grayscale',
                                                              batch_size = 128,
                                                              class_mode = 'categorical
',
                                                              classes = ['happy', 'sad'
, 'neutral', 'surprise'],
                                                              shuffle = True)
test_images, test_labels = next(test_set)

# Write the name of your chosen model in the blank
pred = model3.predict(test_images)
pred_classes = np.argmax(pred, axis=1)
y_true = np.argmax(test_labels, axis = 1)

# Printing the classification report
print(classification_report(y_true, pred_classes, target_names=['happy', 'sad', 'neutral
', 'surprise'], zero_division=0))
# Plotting the heatmap using confusion matrix
cm = confusion_matrix(y_true, pred_classes)
plt.figure(figsize=(8, 5))
sns.heatmap(cm, annot=True, fmt='.0f', xticklabels=['happy', 'sad', 'neutral', 'surprise
'], yticklabels=['happy', 'sad', 'neutral', 'surprise'])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```

```
Found 128 images belonging to 4 classes.
4/4 [==============================] - 1s 133ms/step
              precision    recall  f1-score   support

       happy       0.00      0.00      0.00        32
         sad       0.00      0.00      0.00        32
     neutral       0.00      0.00      0.00        32
    surprise       0.25      1.00      0.40        32

    accuracy                           0.25       128
   macro avg       0.06      0.25      0.10       128
weighted avg       0.06      0.25      0.10       128
```



Observations and Insights:

The classification report reveals that the model exclusively predicts every test image as 'surprise', achieving only 25% accuracy. This suggests the model hasn't effectively learned to differentiate between the classes. All metrics for 'happy', 'sad', and 'neutral' are zero, indicating no correct predictions for these categories. To improve, consider revisiting the model architecture, enhancing data preprocessing and augmentation, and adjusting training parameters. Exploring class balancing techniques could also help, given the model's current bias towards one class. This scenario underscores the need for iterative model evaluation and refinement.

Conclusion:

After reviewing the models' performances, it's clear that our custom CNN and transfer learning methods using EfficientNet and ResNet didn't hit the mark, with each struggling to differentiate between the four classes effectively. The accuracy stalled at 25%, essentially guessing, which signals a gap in our approach. Despite the high hopes pinned on transfer learning's ability to tap into pre-trained networks, the results didn't match expectations for our specific grayscale image task.

The uniform underperformance across different models points to a need for a deeper dive into optimization strategies. This includes enhancing data augmentation to cover more ground, fine-tuning hyperparameters to find the sweet spot, and potentially tackling any dataset imbalances head-on.

Based on these insights, I'm leaning towards a blended solution. This would combine the feature extraction prowess of transfer learning with a tailored model architecture designed specifically for our dataset's nuances. Bolstering this with advanced data preprocessing and augmentation methods could be our best shot at

breaking through the current performance plateau.

Iterative testing and tweaking, guided by the comparative analysis of different models, will be crucial. This adaptive approach, rooted in continuous learning and refinement, stands out as our strategic pathway to developing a more accurate and reliable classification model.

Insights

Refined insights:

- What are the most meaningful insights from the data relevant to the problem?

Comparison of various techniques and their relative performance:

- How do different techniques perform? Which one is performing relatively better? Is there scope to improve the performance further?

Proposal for the final solution design:

- What model do you propose to be adopted? Why is this the best solution to adopt?