In [1]:
```
import pandas as pd
import os, shutil
import time
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam, Nadam, SGD
from tensorflow.keras import regularizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Ba
from tensorflow.keras.layers import Conv2D, MaxPooling2D, AveragePooling2D
from tensorflow.keras import losses

print(tf.__version__)    #YOUR TF VERSION CHECK
```

```
2.9.0
```

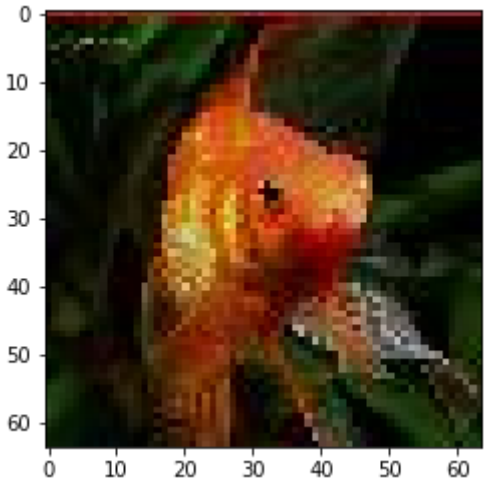
In [ ]:
```
#Train data has folders for each class but Validation has just one folder.
#This section of the code iterates over the Val_annots file, creates folder

df = pd.read_csv('tiny-imagenet-200/val/Val_annots.csv')

parent_dir = 'tiny-imagenet-200/val/'
for i in range(len(df)):
    image_path = os.path.join(parent_dir, df.loc[i]['Name'])
    if not os.path.exists(parent_dir + df.loc[i]['Name']):
        os.mkdir(image_path)
    shutil.copy(parent_dir + 'images/' + df.loc[i]['Image'], image_path)

shutil.rmtree(parent_dir + 'images)
```

The Tiny ImageNet Dataset can be downloaded from [https://image-net.org/download-images.php (https://image-net.org/download-images.php)](https://image-net.org/download-images.php)

In [2]:
```
# Define the image size and number of classes
img_size = 64
num_classes = 200

# Create the data generators for training and validation sets
train_datagen = ImageDataGenerator(
    rescale=1./255,
    horizontal_flip=True,
)

train_generator = train_datagen.flow_from_directory(
    'tiny-imagenet-200/train',
    target_size=(img_size, img_size),
    batch_size=64,
    class_mode='categorical'
)

val_datagen = ImageDataGenerator(
    rescale=1./255,
    horizontal_flip=True,
)

val_generator = val_datagen.flow_from_directory(
    'tiny-imagenet-200/val',
    target_size=(img_size, img_size),
    batch_size=64,
    class_mode='categorical'
)
```

```
Found 100000 images belonging to 200 classes.
Found 10000 images belonging to 200 classes.
```

In [3]:
```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt

#Displaying a sample train image of Fish category
image = Image.open('tiny-imagenet-200/train/n01443537/images/n01443537_4.JP
image_array = np.asarray(image)

plt.imshow(image_array)
plt.show()
```



As we can see the image is only of 64X64X3 size and is highly blured. Classification can be performed but expecting high validation accuracy can be challenging.

In [4]:
```
#Below class is to record the model training times
class TimeHistory(tf.keras.callbacks.Callback):
    def on_train_begin(self, logs={}):
        self.times = []

    def on_epoch_begin(self, batch, logs={}):
        self.epoch_time_start = time.time()

    def on_epoch_end(self, batch, logs={}):
        epoch_time_end = time.time()
        self.times.append(epoch_time_end - self.epoch_time_start)
```

DEEP CNN Models

1. Single Convolutional Layer NN model

In [5]:
```
# Define the CNN architecture
model1 = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(img_
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])
```

In [6]:
```
print(model1.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 flatten (Flatten)           (None, 30752)             0

 dense (Dense)               (None, 1024)              31491072

 dense_1 (Dense)             (None, 200)               205000

=================================================================
Total params: 31,696,968
Trainable params: 31,696,968
Non-trainable params: 0
_________________________________________________________________
None
```

In [7]:
```
time_callback = TimeHistory()

# Compile the model with an optimizer, a loss function, and evaluation metr
model1.compile(optimizer=Adam(learning_rate = 0.0001),
               loss='categorical_crossentropy',
               metrics=['accuracy'])

# Train the model on the training data and validate it on the validation da
with tf.device('/device:GPU:0'):
    history = model1.fit(train_generator,
                         validation_data=val_generator,
                         epochs=15,
                         callbacks=[time_callback, tf.keras.callbacks.Tensor
```

```
Epoch 1/15
   6/1563 [..............................] - ETA: 36s - loss: 5.3947 - ac
curacy: 0.0104    WARNING:tensorflow:Callback method `on_train_batch_end`
is slow compared to the batch time (batch time: 0.0093s vs `on_train_batc
h_end` time: 0.0169s). Check your callbacks.
1563/1563 [==============================] - 38s 24ms/step - loss: 4.7895
- accuracy: 0.0600 - val_loss: 4.4842 - val_accuracy: 0.0993
Epoch 2/15
1563/1563 [==============================] - 36s 23ms/step - loss: 4.3215
- accuracy: 0.1191 - val_loss: 4.2536 - val_accuracy: 0.1265
Epoch 3/15
1563/1563 [==============================] - 36s 23ms/step - loss: 4.0804
- accuracy: 0.1529 - val_loss: 4.1041 - val_accuracy: 0.1457
Epoch 4/15
1563/1563 [==============================] - 36s 23ms/step - loss: 3.8904
- accuracy: 0.1814 - val_loss: 3.9833 - val_accuracy: 0.1618
Epoch 5/15
1563/1563 [==============================] - 36s 23ms/step - loss: 3.7341
- accuracy: 0.2067 - val_loss: 3.8976 - val_accuracy: 0.1720
Epoch 6/15
1563/1563 [==============================] - 36s 23ms/step - loss: 3.5990
- accuracy: 0.2288 - val_loss: 3.8270 - val_accuracy: 0.1868
Epoch 7/15
1563/1563 [==============================] - 36s 23ms/step - loss: 3.4708
- accuracy: 0.2517 - val_loss: 3.7874 - val_accuracy: 0.1942
Epoch 8/15
1563/1563 [==============================] - 37s 24ms/step - loss: 3.3522
- accuracy: 0.2710 - val_loss: 3.7486 - val_accuracy: 0.1979
Epoch 9/15
1563/1563 [==============================] - 37s 24ms/step - loss: 3.2396
- accuracy: 0.2926 - val_loss: 3.7062 - val_accuracy: 0.2034
Epoch 10/15
1563/1563 [==============================] - 37s 24ms/step - loss: 3.1341
- accuracy: 0.3120 - val_loss: 3.7092 - val_accuracy: 0.2050
Epoch 11/15
1563/1563 [==============================] - 38s 24ms/step - loss: 3.0311
- accuracy: 0.3301 - val_loss: 3.6921 - val_accuracy: 0.2084
Epoch 12/15
1563/1563 [==============================] - 37s 24ms/step - loss: 2.9279
- accuracy: 0.3504 - val_loss: 3.6742 - val_accuracy: 0.2143
Epoch 13/15
1563/1563 [==============================] - 37s 24ms/step - loss: 2.8303
- accuracy: 0.3698 - val_loss: 3.6729 - val_accuracy: 0.2114
Epoch 14/15
1563/1563 [==============================] - 37s 23ms/step - loss: 2.7354
- accuracy: 0.3883 - val_loss: 3.6693 - val_accuracy: 0.2207
Epoch 15/15
1563/1563 [==============================] - 36s 23ms/step - loss: 2.6385
- accuracy: 0.4084 - val_loss: 3.6795 - val_accuracy: 0.2143
```

In [8]:
```
total_time = sum(time_callback.times)
print(f"Total execution time: {total_time:.2f} seconds")
```

```
Total execution time: 552.10 seconds
```

2. Two Convolutional Layer NN model

In [9]:
```
# Define the CNN architecture
model2 = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(img_
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel_regularize
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])
```

In [10]:
```
print(model2.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 62, 62, 32)        896

 max_pooling2d_1 (MaxPooling  (None, 31, 31, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 29, 29, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 12544)             0

 dense_2 (Dense)             (None, 1024)              12846080

 dense_3 (Dense)             (None, 200)               205000

=================================================================
Total params: 13,070,472
Trainable params: 13,070,472
Non-trainable params: 0
_________________________________________________________________
None
```

In [11]:
```
time_callback = TimeHistory()

# Compile the model with an optimizer, a loss function, and evaluation metr
model2.compile(optimizer=Adam(learning_rate=0.0001),
               loss='categorical_crossentropy',
               metrics=['accuracy'])

# Train the model on the training data and validate it on the validation da
with tf.device('/device:GPU:0'):
    history = model2.fit(train_generator,
                         validation_data=val_generator,
                         epochs=15,
                         callbacks=[time_callback, tf.keras.callbacks.Tensor
```

```
Epoch 1/15
   6/1563 [..............................] - ETA: 41s - loss: 5.7219 - ac
curacy: 0.0026   WARNING:tensorflow:Callback method `on_train_batch_end`
is slow compared to the batch time (batch time: 0.0118s vs `on_train_batc
h_end` time: 0.0183s). Check your callbacks.
1563/1563 [==============================] - 31s 20ms/step - loss: 4.9267
- accuracy: 0.0650 - val_loss: 4.5511 - val_accuracy: 0.1037
Epoch 2/15
1563/1563 [==============================] - 34s 22ms/step - loss: 4.3696
- accuracy: 0.1274 - val_loss: 4.2424 - val_accuracy: 0.1458
Epoch 3/15
1563/1563 [==============================] - 32s 21ms/step - loss: 4.0945
- accuracy: 0.1645 - val_loss: 4.0543 - val_accuracy: 0.1671
Epoch 4/15
1563/1563 [==============================] - 31s 20ms/step - loss: 3.8943
- accuracy: 0.1944 - val_loss: 3.9258 - val_accuracy: 0.1874
Epoch 5/15
1563/1563 [==============================] - 31s 20ms/step - loss: 3.7441
- accuracy: 0.2176 - val_loss: 3.8419 - val_accuracy: 0.1994
Epoch 6/15
1563/1563 [==============================] - 33s 21ms/step - loss: 3.6196
- accuracy: 0.2376 - val_loss: 3.7395 - val_accuracy: 0.2154
Epoch 7/15
1563/1563 [==============================] - 33s 21ms/step - loss: 3.5122
- accuracy: 0.2550 - val_loss: 3.6906 - val_accuracy: 0.2245
Epoch 8/15
1563/1563 [==============================] - 32s 20ms/step - loss: 3.4221
- accuracy: 0.2710 - val_loss: 3.6632 - val_accuracy: 0.2292
Epoch 9/15
1563/1563 [==============================] - 32s 20ms/step - loss: 3.3374
- accuracy: 0.2852 - val_loss: 3.6227 - val_accuracy: 0.2397
Epoch 10/15
1563/1563 [==============================] - 32s 20ms/step - loss: 3.2593
- accuracy: 0.2984 - val_loss: 3.6215 - val_accuracy: 0.2384
Epoch 11/15
1563/1563 [==============================] - 32s 20ms/step - loss: 3.1897
- accuracy: 0.3123 - val_loss: 3.5856 - val_accuracy: 0.2469
Epoch 12/15
1563/1563 [==============================] - 32s 20ms/step - loss: 3.1169
- accuracy: 0.3241 - val_loss: 3.5701 - val_accuracy: 0.2502
Epoch 13/15
1563/1563 [==============================] - 32s 20ms/step - loss: 3.0455
- accuracy: 0.3387 - val_loss: 3.5597 - val_accuracy: 0.2494
Epoch 14/15
1563/1563 [==============================] - 32s 20ms/step - loss: 2.9837
- accuracy: 0.3494 - val_loss: 3.5479 - val_accuracy: 0.2540
Epoch 15/15
1563/1563 [==============================] - 32s 20ms/step - loss: 2.9102
- accuracy: 0.3644 - val_loss: 3.5542 - val_accuracy: 0.2500
```

In [12]:
```
total_time = sum(time_callback.times)
print(f"Total execution time: {total_time:.2f} seconds")
```

```
Total execution time: 478.81 seconds
```

3. DEEP CNN Model 1

In [13]:
```
model3 = Sequential()

model3.add(Conv2D(64, (3, 3), strides=(1,1), padding='same', input_shape=(i
model3.add(BatchNormalization())
model3.add(Conv2D(64, (3, 3), strides=(1,1), padding='same', kernel_regular
model3.add(BatchNormalization())
model3.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same'))
model3.add(Activation('relu'))

model3.add(Conv2D(128, (3, 3), strides=(1,1), padding='same', kernel_regula
model3.add(BatchNormalization())
model3.add(Conv2D(128, (3, 3), strides=(1,1), padding='same', kernel_regula
model3.add(BatchNormalization())
model3.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same'))
model3.add(Activation('relu'))

model3.add(Conv2D(128, (3, 3), strides=(1,1), padding='same', kernel_regula
model3.add(BatchNormalization())
model3.add(Conv2D(128, (3, 3), strides=(1,1), padding='same', kernel_regula
model3.add(BatchNormalization())
model3.add(Activation('relu'))
model3.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same'))

model3.add(Conv2D(128, (3, 3), strides=(1,1), padding='same', kernel_regula
model3.add(BatchNormalization())
model3.add(Conv2D(128, (3, 3), strides=(1,1), padding='same', kernel_regula
model3.add(BatchNormalization())
model3.add(Activation('relu'))
model3.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same'))

model3.add(Conv2D(256, (3, 3), strides=(1,1), padding='same', kernel_regula
model3.add(BatchNormalization())
model3.add(Conv2D(512, (3, 3), strides=(1,1), padding='same', kernel_regula
model3.add(BatchNormalization())
model3.add(Activation('relu'))
model3.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same'))

model3.add(Flatten())
model3.add(Dense(4096))
model3.add(BatchNormalization())
model3.add(Activation('relu'))

model3.add(Dense(1024))
model3.add(BatchNormalization())
model3.add(Activation('relu'))

model3.add(Dense(num_classes))

model3.add(Activation('softmax'))

time_callback = TimeHistory()

model3.compile(loss=losses.categorical_crossentropy,
optimizer=Adam(learning_rate=0.0001), metrics=['accuracy'])
```

In [14]:
```
print(model3.summary())
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 64, 64, 64)        1792

 batch_normalization (BatchN  (None, 64, 64, 64)       256
 ormalization)

 conv2d_4 (Conv2D)           (None, 64, 64, 64)        36928

 batch_normalization_1 (Batc  (None, 64, 64, 64)       256
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 32, 32, 64)       0
 2D)

 activation (Activation)     (None, 32, 32, 64)        0

 conv2d_5 (Conv2D)           (None, 32, 32, 128)       73856

 batch_normalization_2 (Batc  (None, 32, 32, 128)      512
 hNormalization)

 conv2d_6 (Conv2D)           (None, 32, 32, 128)       147584

 batch_normalization_3 (Batc  (None, 32, 32, 128)      512
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 16, 16, 128)      0
 2D)

 activation_1 (Activation)   (None, 16, 16, 128)       0

 conv2d_7 (Conv2D)           (None, 16, 16, 128)       147584

 batch_normalization_4 (Batc  (None, 16, 16, 128)      512
 hNormalization)

 conv2d_8 (Conv2D)           (None, 16, 16, 128)       147584

 batch_normalization_5 (Batc  (None, 16, 16, 128)      512
 hNormalization)

 activation_2 (Activation)   (None, 16, 16, 128)       0

 max_pooling2d_5 (MaxPooling  (None, 8, 8, 128)        0
 2D)

 conv2d_9 (Conv2D)           (None, 8, 8, 128)         147584

 batch_normalization_6 (Batc  (None, 8, 8, 128)        512
 hNormalization)

 conv2d_10 (Conv2D)          (None, 8, 8, 128)         147584

 batch_normalization_7 (Batc  (None, 8, 8, 128)        512
```

```
hNormalization)

activation_3 (Activation)   (None, 8, 8, 128)         0

max_pooling2d_6 (MaxPooling  (None, 4, 4, 128)        0
2D)

conv2d_11 (Conv2D)          (None, 4, 4, 256)         295168

batch_normalization_8 (Batc  (None, 4, 4, 256)        1024
hNormalization)

conv2d_12 (Conv2D)          (None, 4, 4, 512)         1180160

batch_normalization_9 (Batc  (None, 4, 4, 512)        2048
hNormalization)

activation_4 (Activation)   (None, 4, 4, 512)         0

max_pooling2d_7 (MaxPooling  (None, 2, 2, 512)        0
2D)

flatten_2 (Flatten)         (None, 2048)              0

dense_4 (Dense)             (None, 4096)              8392704

batch_normalization_10 (Bat  (None, 4096)             16384
chNormalization)

activation_5 (Activation)   (None, 4096)              0

dense_5 (Dense)             (None, 1024)              4195328

batch_normalization_11 (Bat  (None, 1024)             4096
chNormalization)

activation_6 (Activation)   (None, 1024)              0

dense_6 (Dense)             (None, 200)               205000

activation_7 (Activation)   (None, 200)               0

=================================================================
Total params: 15,145,992
Trainable params: 15,132,424
Non-trainable params: 13,568
_________________________________________________________________
None
```

In [15]:
```
# Train the model on the training data and validate it on the validation da
with tf.device('/device:GPU:0'):
    history = model3.fit(train_generator,
                         validation_data=val_generator,
                         epochs=15,
                         callbacks=[time_callback, tf.keras.callbacks.Tensor
```

```
Epoch 1/15
   6/1563 [..............................] - ETA: 3:42 - loss: 9.4461 - a
ccuracy: 0.0052WARNING:tensorflow:Callback method `on_train_batch_end` is
slow compared to the batch time (batch time: 0.0620s vs `on_train_batch_e
nd` time: 0.1052s). Check your callbacks.
1563/1563 [==============================] - 197s 125ms/step - loss: 7.12
46 - accuracy: 0.1654 - val_loss: 6.1606 - val_accuracy: 0.2170
Epoch 2/15
1563/1563 [==============================] - 196s 125ms/step - loss: 5.33
15 - accuracy: 0.2849 - val_loss: 4.9999 - val_accuracy: 0.2842
Epoch 3/15
1563/1563 [==============================] - 196s 125ms/step - loss: 4.35
45 - accuracy: 0.3569 - val_loss: 4.3981 - val_accuracy: 0.3104
Epoch 4/15
1563/1563 [==============================] - 196s 125ms/step - loss: 3.72
88 - accuracy: 0.4108 - val_loss: 4.1502 - val_accuracy: 0.3201
Epoch 5/15
1563/1563 [==============================] - 196s 125ms/step - loss: 3.28
35 - accuracy: 0.4561 - val_loss: 3.6812 - val_accuracy: 0.3706
Epoch 6/15
1563/1563 [==============================] - 196s 125ms/step - loss: 2.93
88 - accuracy: 0.4983 - val_loss: 3.4901 - val_accuracy: 0.3859
Epoch 7/15
1563/1563 [==============================] - 196s 125ms/step - loss: 2.64
81 - accuracy: 0.5429 - val_loss: 3.3602 - val_accuracy: 0.4059
Epoch 8/15
1563/1563 [==============================] - 208s 133ms/step - loss: 2.39
32 - accuracy: 0.5840 - val_loss: 3.3655 - val_accuracy: 0.3955
Epoch 9/15
1563/1563 [==============================] - 210s 135ms/step - loss: 2.15
91 - accuracy: 0.6242 - val_loss: 3.3649 - val_accuracy: 0.3959
Epoch 10/15
1563/1563 [==============================] - 210s 135ms/step - loss: 1.94
17 - accuracy: 0.6687 - val_loss: 3.3642 - val_accuracy: 0.3958
Epoch 11/15
1563/1563 [==============================] - 210s 135ms/step - loss: 1.74
43 - accuracy: 0.7098 - val_loss: 3.3743 - val_accuracy: 0.3960
Epoch 12/15
1563/1563 [==============================] - 211s 135ms/step - loss: 1.55
86 - accuracy: 0.7505 - val_loss: 3.3587 - val_accuracy: 0.4036
Epoch 13/15
1563/1563 [==============================] - 211s 135ms/step - loss: 1.40
00 - accuracy: 0.7860 - val_loss: 3.5995 - val_accuracy: 0.3777
Epoch 14/15
1563/1563 [==============================] - 211s 135ms/step - loss: 1.26
16 - accuracy: 0.8176 - val_loss: 3.4843 - val_accuracy: 0.3940
Epoch 15/15
1563/1563 [==============================] - 211s 135ms/step - loss: 1.14
72 - accuracy: 0.8426 - val_loss: 3.5772 - val_accuracy: 0.3954
```

In [16]:
```
total_time = sum(time_callback.times)
print(f"Total execution time: {total_time:.2f} seconds")
```

```
Total execution time: 3054.00 seconds
```

4. Deep CNN Model 2

In [17]:
```
# Import required libraries
import tensorflow as tf
from tensorflow.keras import layers

# Define the input shape
input_shape = (64, 64, 3)

# Define the model layers
model4 = tf.keras.Sequential([
    layers.Conv2D(64, (3,3), padding='same', activation='relu', input_shape
    layers.Conv2D(64, (3,3), padding='same', activation='relu', kernel_regu
    layers.MaxPooling2D((2,2)),
    layers.BatchNormalization(),

    layers.Conv2D(128, (3,3), padding='same', activation='relu', kernel_reg
    layers.MaxPooling2D((2,2)),
    layers.BatchNormalization(),

    layers.Conv2D(256, (3,3), padding='same', activation='relu', kernel_reg
    layers.Conv2D(256, (3,3), padding='same', activation='relu', kernel_reg
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),

    layers.Dense(1024, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(512, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(200, activation='softmax')
])

time_callback = TimeHistory()

# Compile the model
model4.compile(optimizer=Adam(learning_rate=0.0001),
               loss='categorical_crossentropy',
               metrics=['accuracy'])
```

In [18]:
```
# Print the model summary
print(model4.summary())
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_13 (Conv2D)          (None, 64, 64, 64)        1792

 conv2d_14 (Conv2D)          (None, 64, 64, 64)        36928

 max_pooling2d_8 (MaxPooling  (None, 32, 32, 64)       0
 2D)

 batch_normalization_12 (Bat  (None, 32, 32, 64)       256
 chNormalization)

 conv2d_15 (Conv2D)          (None, 32, 32, 128)       73856

 max_pooling2d_9 (MaxPooling  (None, 16, 16, 128)      0
 2D)

 batch_normalization_13 (Bat  (None, 16, 16, 128)      512
 chNormalization)

 conv2d_16 (Conv2D)          (None, 16, 16, 256)       295168

 conv2d_17 (Conv2D)          (None, 16, 16, 256)       590080

 max_pooling2d_10 (MaxPoolin  (None, 8, 8, 256)        0
 g2D)

 flatten_3 (Flatten)         (None, 16384)             0

 dense_7 (Dense)             (None, 1024)              16778240

 dropout (Dropout)           (None, 1024)              0

 dense_8 (Dense)             (None, 512)               524800

 dropout_1 (Dropout)         (None, 512)               0

 dense_9 (Dense)             (None, 200)               102600

=================================================================
Total params: 18,404,232
Trainable params: 18,403,848
Non-trainable params: 384
_________________________________________________________________
None
```

In [19]:
```
# Train the model
with tf.device('/device:GPU:0'):
    history = model4.fit(train_generator,
                         epochs=25,
                         validation_data=val_generator,
                         callbacks=[time_callback, tf.keras.callbacks.Tensor
```

```
Epoch 1/25
   6/1563 [..............................] - ETA: 2:26 - loss: 11.5748 -
accuracy: 0.0052  WARNING:tensorflow:Callback method `on_train_batch_end`
is slow compared to the batch time (batch time: 0.0323s vs `on_train_batc
h_end` time: 0.0623s). Check your callbacks.
1563/1563 [==============================] - 151s 96ms/step - loss: 7.193
8 - accuracy: 0.0339 - val_loss: 5.0122 - val_accuracy: 0.1006
Epoch 2/25
1563/1563 [==============================] - 150s 96ms/step - loss: 4.632
5 - accuracy: 0.1095 - val_loss: 4.0970 - val_accuracy: 0.1766
Epoch 3/25
1563/1563 [==============================] - 149s 96ms/step - loss: 4.022
9 - accuracy: 0.1744 - val_loss: 3.6336 - val_accuracy: 0.2409
Epoch 4/25
1563/1563 [==============================] - 149s 96ms/step - loss: 3.696
1 - accuracy: 0.2189 - val_loss: 3.4759 - val_accuracy: 0.2642
Epoch 5/25
1563/1563 [==============================] - 149s 96ms/step - loss: 3.468
3 - accuracy: 0.2555 - val_loss: 3.2908 - val_accuracy: 0.2893
Epoch 6/25
1563/1563 [==============================] - 150s 96ms/step - loss: 3.292
9 - accuracy: 0.2885 - val_loss: 3.1674 - val_accuracy: 0.3149
Epoch 7/25
1563/1563 [==============================] - 150s 96ms/step - loss: 3.150
1 - accuracy: 0.3148 - val_loss: 3.0611 - val_accuracy: 0.3317
Epoch 8/25
1563/1563 [==============================] - 150s 96ms/step - loss: 3.017
9 - accuracy: 0.3410 - val_loss: 2.9928 - val_accuracy: 0.3503
Epoch 9/25
1563/1563 [==============================] - 150s 96ms/step - loss: 2.911
4 - accuracy: 0.3603 - val_loss: 2.9234 - val_accuracy: 0.3588
Epoch 10/25
1563/1563 [==============================] - 150s 96ms/step - loss: 2.805
3 - accuracy: 0.3821 - val_loss: 2.8927 - val_accuracy: 0.3678
Epoch 11/25
1563/1563 [==============================] - 150s 96ms/step - loss: 2.715
8 - accuracy: 0.3999 - val_loss: 2.8264 - val_accuracy: 0.3791
Epoch 12/25
1563/1563 [==============================] - 150s 96ms/step - loss: 2.632
5 - accuracy: 0.4184 - val_loss: 2.7976 - val_accuracy: 0.3892
Epoch 13/25
1563/1563 [==============================] - 150s 96ms/step - loss: 2.551
7 - accuracy: 0.4352 - val_loss: 2.7569 - val_accuracy: 0.4004
Epoch 14/25
1563/1563 [==============================] - 150s 96ms/step - loss: 2.476
2 - accuracy: 0.4508 - val_loss: 2.8391 - val_accuracy: 0.3837
Epoch 15/25
1563/1563 [==============================] - 150s 96ms/step - loss: 2.399
6 - accuracy: 0.4689 - val_loss: 2.7409 - val_accuracy: 0.3989
Epoch 16/25
1563/1563 [==============================] - 150s 96ms/step - loss: 2.330
1 - accuracy: 0.4836 - val_loss: 2.7665 - val_accuracy: 0.4049
Epoch 17/25
1563/1563 [==============================] - 150s 96ms/step - loss: 2.267
6 - accuracy: 0.4976 - val_loss: 2.7119 - val_accuracy: 0.4177
Epoch 18/25
1563/1563 [==============================] - 150s 96ms/step - loss: 2.197
```

```
2 - accuracy: 0.5127 - val_loss: 2.6823 - val_accuracy: 0.4169
Epoch 19/25
1563/1563 [==============================] - 150s 96ms/step - loss: 2.142
2 - accuracy: 0.5253 - val_loss: 2.7598 - val_accuracy: 0.4093
Epoch 20/25
1563/1563 [==============================] - 150s 96ms/step - loss: 2.067
9 - accuracy: 0.5410 - val_loss: 2.7221 - val_accuracy: 0.4182
Epoch 21/25
1563/1563 [==============================] - 150s 96ms/step - loss: 2.009
6 - accuracy: 0.5550 - val_loss: 2.8340 - val_accuracy: 0.3975
Epoch 22/25
1563/1563 [==============================] - 150s 96ms/step - loss: 1.953
9 - accuracy: 0.5672 - val_loss: 2.8361 - val_accuracy: 0.4000
Epoch 23/25
1563/1563 [==============================] - 149s 95ms/step - loss: 1.894
8 - accuracy: 0.5814 - val_loss: 2.7627 - val_accuracy: 0.4200
Epoch 24/25
1563/1563 [==============================] - 149s 95ms/step - loss: 1.842
2 - accuracy: 0.5928 - val_loss: 2.7544 - val_accuracy: 0.4185
Epoch 25/25
1563/1563 [==============================] - 150s 96ms/step - loss: 1.778
9 - accuracy: 0.6089 - val_loss: 2.7473 - val_accuracy: 0.4240
```

In [20]:
```
total_time = sum(time_callback.times)
print(f"Total execution time: {total_time:.2f} seconds")
```

```
Total execution time: 3744.04 seconds
```

5) ResNet Model 1- ResNet50

In [5]:
```
resnet50_model = tf.keras.applications.ResNet50(
    weights='imagenet',
    include_top=False,
    input_shape=(img_size, img_size, 3)
)
```

In [6]:
```
x = resnet50_model.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dropout(0.5)(x)
predictions = layers.Dense(num_classes, activation='softmax')(x)

model5 = models.Model(inputs = resnet50_model.input, outputs=predictions)
```

In [7]:
```
time_callback = TimeHistory()

# Compile the model
model5.compile(optimizer=Adam(learning_rate=0.0001),
               loss='categorical_crossentropy',
               metrics=['accuracy'])
```

In [8]:
```
print(model5.summary())
```

```
 conv2_block3_out (Activation)  (None, 16, 16, 256)  0           ['conv2_
block3_add[0][0]']

 conv3_block1_1_conv (Conv2D)   (None, 8, 8, 128)    32896       ['conv2_
block3_out[0][0]']

 conv3_block1_1_bn (BatchNormal  (None, 8, 8, 128)   512         ['conv3_
block1_1_conv[0][0]']
 ization)

 conv3_block1_1_relu (Activatio  (None, 8, 8, 128)   0           ['conv3_
block1_1_bn[0][0]']
 n)

 conv3_block1_2_conv (Conv2D)   (None, 8, 8, 128)    147584      ['conv3_
block1_1_relu[0][0]']

 conv3_block1_2_bn (BatchNormal  (None, 8, 8, 128)   512         ['conv3_
block1_2_conv[0][0]']
```

In [9]:
```
# Train the model
with tf.device('/device:GPU:0'):
    history = model5.fit(train_generator,
                         epochs=15,
                         validation_data=val_generator,
                         callbacks=[time_callback, tf.keras.callbacks.Tensor
```

```
Epoch 1/15
   6/1563 [..............................] - ETA: 3:31 - loss: 6.8956 - a
ccuracy: 0.0000e+00WARNING:tensorflow:Callback method `on_train_batch_end
` is slow compared to the batch time (batch time: 0.0998s vs `on_train_ba
tch_end` time: 0.1228s). Check your callbacks.
1563/1563 [==============================] - 228s 143ms/step - loss: 3.47
13 - accuracy: 0.2689 - val_loss: 2.3051 - val_accuracy: 0.4529
Epoch 2/15
1563/1563 [==============================] - 222s 142ms/step - loss: 2.16
66 - accuracy: 0.4841 - val_loss: 2.0099 - val_accuracy: 0.5088
Epoch 3/15
1563/1563 [==============================] - 221s 141ms/step - loss: 1.72
59 - accuracy: 0.5727 - val_loss: 1.9419 - val_accuracy: 0.5334
Epoch 4/15
1563/1563 [==============================] - 219s 140ms/step - loss: 1.41
61 - accuracy: 0.6404 - val_loss: 1.9479 - val_accuracy: 0.5355
Epoch 5/15
1563/1563 [==============================] - 220s 140ms/step - loss: 1.17
95 - accuracy: 0.6917 - val_loss: 1.8960 - val_accuracy: 0.5548
Epoch 6/15
1563/1563 [==============================] - 220s 141ms/step - loss: 0.97
78 - accuracy: 0.7399 - val_loss: 1.9514 - val_accuracy: 0.5486
Epoch 7/15
1563/1563 [==============================] - 220s 141ms/step - loss: 0.81
71 - accuracy: 0.7795 - val_loss: 2.0127 - val_accuracy: 0.5491
Epoch 8/15
1563/1563 [==============================] - 221s 142ms/step - loss: 0.68
78 - accuracy: 0.8100 - val_loss: 2.0410 - val_accuracy: 0.5512
Epoch 9/15
1563/1563 [==============================] - 221s 142ms/step - loss: 0.58
61 - accuracy: 0.8357 - val_loss: 2.1752 - val_accuracy: 0.5480
Epoch 10/15
1563/1563 [==============================] - 222s 142ms/step - loss: 0.50
17 - accuracy: 0.8576 - val_loss: 2.2624 - val_accuracy: 0.5412
Epoch 11/15
1563/1563 [==============================] - 221s 141ms/step - loss: 0.43
99 - accuracy: 0.8738 - val_loss: 2.2292 - val_accuracy: 0.5575
Epoch 12/15
1563/1563 [==============================] - 221s 141ms/step - loss: 0.38
21 - accuracy: 0.8886 - val_loss: 2.2521 - val_accuracy: 0.5526
Epoch 13/15
1563/1563 [==============================] - 221s 141ms/step - loss: 0.34
24 - accuracy: 0.8990 - val_loss: 2.3511 - val_accuracy: 0.5531
Epoch 14/15
1563/1563 [==============================] - 222s 142ms/step - loss: 0.30
40 - accuracy: 0.9095 - val_loss: 2.4762 - val_accuracy: 0.5334
Epoch 15/15
1563/1563 [==============================] - 222s 142ms/step - loss: 0.29
01 - accuracy: 0.9126 - val_loss: 2.4808 - val_accuracy: 0.5395
```

In [10]:
```
total_time = sum(time_callback.times)
print(f"Total execution time: {total_time:.2f} seconds")
```

```
Total execution time: 3320.05 seconds
```

6) ResNet Model 2 - ResNet101

In [11]:
```
resnet101_model = tf.keras.applications.ResNet101(
    include_top=False,
    weights="imagenet",
    input_shape=(img_size, img_size, 3),
)
```

In [12]:
```
x = resnet101_model.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dropout(0.5)(x)
predictions = layers.Dense(num_classes, activation='softmax')(x)

model6 = models.Model(inputs = resnet101_model.input, outputs=predictions)
```

In [13]:
```
time_callback = TimeHistory()

# Compile the model
model6.compile(optimizer=Adam(learning_rate=0.0001),
               loss='categorical_crossentropy',
               metrics=['accuracy'])
```

In [14]:
```
print(model6.summary())
```

```
Model: "model_1"
__________________________________________________________________________
_______________________
 Layer (type)                Output Shape              Param #     Connecte
d to
==========================================================================
=======================
 input_2 (InputLayer)        [(None, 64, 64, 3)]       0           []

 conv1_pad (ZeroPadding2D)   (None, 70, 70, 3)         0           ['input_
2[0][0]']

 conv1_conv (Conv2D)         (None, 32, 32, 64)        9472        ['conv1_
pad[0][0]']

 conv1_bn (BatchNormalization)  (None, 32, 32, 64)     256         ['conv1_
conv[0][0]']

 conv1_relu (Activation)     (None, 32, 32, 64)        0           ['conv1_
```

In [15]:
```
# Train the model
with tf.device('/device:GPU:0'):
    history = model6.fit(train_generator,
                         epochs=15,
                         validation_data=val_generator,
                         callbacks=[time_callback, tf.keras.callbacks.Tensor
```

```
Epoch 1/15
1563/1563 [==============================] - 381s 237ms/step - loss: 3.43
22 - accuracy: 0.2747 - val_loss: 2.2370 - val_accuracy: 0.4663
Epoch 2/15
1563/1563 [==============================] - 365s 233ms/step - loss: 2.08
05 - accuracy: 0.5015 - val_loss: 2.0029 - val_accuracy: 0.5235
Epoch 3/15
1563/1563 [==============================] - 363s 232ms/step - loss: 1.62
27 - accuracy: 0.5986 - val_loss: 1.8773 - val_accuracy: 0.5395
Epoch 4/15
1563/1563 [==============================] - 362s 232ms/step - loss: 1.30
10 - accuracy: 0.6671 - val_loss: 1.9223 - val_accuracy: 0.5489
Epoch 5/15
1563/1563 [==============================] - 362s 232ms/step - loss: 1.05
09 - accuracy: 0.7246 - val_loss: 1.9876 - val_accuracy: 0.5406
Epoch 6/15
1563/1563 [==============================] - 364s 233ms/step - loss: 0.84
78 - accuracy: 0.7735 - val_loss: 2.0723 - val_accuracy: 0.5532
Epoch 7/15
1563/1563 [==============================] - 362s 232ms/step - loss: 0.68
97 - accuracy: 0.8113 - val_loss: 2.0589 - val_accuracy: 0.5498
Epoch 8/15
1563/1563 [==============================] - 363s 233ms/step - loss: 0.56
99 - accuracy: 0.8409 - val_loss: 2.1778 - val_accuracy: 0.5487
Epoch 9/15
1563/1563 [==============================] - 362s 232ms/step - loss: 0.48
21 - accuracy: 0.8636 - val_loss: 2.1881 - val_accuracy: 0.5507
Epoch 10/15
1563/1563 [==============================] - 363s 233ms/step - loss: 0.42
13 - accuracy: 0.8800 - val_loss: 2.2289 - val_accuracy: 0.5505
Epoch 11/15
1563/1563 [==============================] - 364s 233ms/step - loss: 0.36
73 - accuracy: 0.8935 - val_loss: 2.3453 - val_accuracy: 0.5432
Epoch 12/15
1563/1563 [==============================] - 361s 231ms/step - loss: 0.31
72 - accuracy: 0.9058 - val_loss: 2.4139 - val_accuracy: 0.5423
Epoch 13/15
1563/1563 [==============================] - 361s 231ms/step - loss: 0.29
44 - accuracy: 0.9131 - val_loss: 2.5502 - val_accuracy: 0.5351
Epoch 14/15
1563/1563 [==============================] - 366s 234ms/step - loss: 0.26
22 - accuracy: 0.9216 - val_loss: 2.4279 - val_accuracy: 0.5477
Epoch 15/15
1563/1563 [==============================] - 363s 232ms/step - loss: 0.23
90 - accuracy: 0.9280 - val_loss: 2.5798 - val_accuracy: 0.5337
```

In [16]:
```
total_time = sum(time_callback.times)
print(f"Total execution time: {total_time:.2f} seconds")
```

```
Total execution time: 5463.40 seconds
```

In [ ]:
```

```