In [1]:

```
Collecting tfa-nightly
  Downloading tfa_nightly-0.15.0.dev20210910145451-cp37-cp37m-many
linux_2_12_x86_64.manylinux2010_x86_64.whl (1.1 MB)
     |████████████████████████████████| 1.1 MB 11.8 MB/s eta 0:00:
01
Requirement already satisfied: typeguard>=2.7 in /usr/local/lib/py
thon3.7/dist-packages (from tfa-nightly) (2.7.1)
Installing collected packages: tfa-nightly
Successfully installed tfa-nightly-0.15.0.dev20210910145451
```

In [96]:

```
import tensorflow as tf
import os
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
  tf.config.experimental.set_memory_growth(gpu, True)

import tensorflow.keras as keras
import tensorflow_addons as tfa
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import pandas as pd
from scipy.interpolate import interp1d
from scipy.ndimage.interpolation import rotate, shift
import PIL
from PIL import Image
from imgaug import augmenters as iaa
from skimage import util as sku
import random
from itertools import chain
from itertools import permutations
from itertools import combinations
from itertools import cycle
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Input, Conv2D, Activation, MaxP
import json
from keras.preprocessing import image
import warnings
warnings.filterwarnings('ignore')

%load_ext tensorboard

print("Tensorflow version is: ", tf.__version__)
assert tf.__version__[0] == '2'
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Tensorflow version is:  2.6.0
```

In [3]:

```
Archive:  drive/MyDrive/A1_data.zip
  inflating: A1_data/Images.zip
  inflating: A1_data/README.md
  inflating: A1_data/__MACOSX/._README.md
  inflating: A1_data/S40AR_test_data.csv
```

In [4]:

```
Streaming output truncated to the last 5000 lines.
  inflating: A1_data/Images/Img_4611.jpg
  inflating: A1_data/Images/Img_789.jpg
  inflating: A1_data/Images/Img_7318.jpg
  inflating: A1_data/Images/Img_6006.jpg
  inflating: A1_data/Images/Img_9335.jpg
  inflating: A1_data/Images/Img_1769.jpg
  inflating: A1_data/Images/Img_8995.jpg
  inflating: A1_data/Images/Img_6760.jpg
  inflating: A1_data/Images/Img_9453.jpg
  inflating: A1_data/Images/Img_3618.jpg
  inflating: A1_data/Images/Img_4177.jpg
  inflating: A1_data/Images/Img_5269.jpg
  inflating: A1_data/Images/Img_2506.jpg
  inflating: A1_data/Images/Img_4163.jpg
  inflating: A1_data/Images/Img_2512.jpg
  inflating: A1_data/Images/Img_8981.jpg
  inflating: A1_data/Images/Img_6774.jpg
  inflating: A1_data/Images/Img_9447.jpg
```

In [4]:

Exploratory Data Analysis

Strategy

- Load the images and visualize a sample. Check the differences in dimensions of the images
- Getting the minimum and maximum values for the dimensions of images
- Look at the basic information of the dataframe, and check for missing values
- Checking the distribution of the action_classes and actions

In [5]:

```
img_dir = "A1_data/Images/"

data_df = pd.read_csv('A1_data/S40AR_train_data.csv')
```

Out[5]:

	FileName	action	action_class
0	Img_1000.jpg	walking_the_dog	Interacting_with_animal
1	Img_1001.jpg	riding_a_bike	other_activity
2	Img_1002.jpg	gardening	domestic_work
3	Img_1008.jpg	cooking	domestic_work
4	Img_1010.jpg	jumping	other_activity

In [5]:

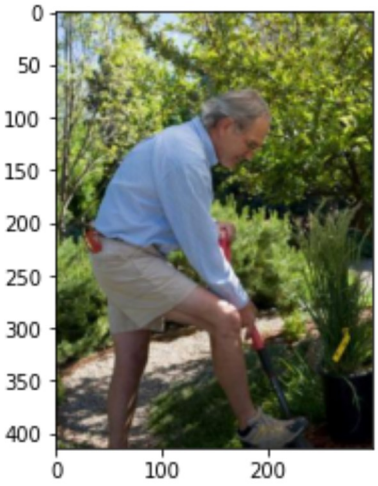
Now we can visualize the images to see if they've loaded correctly

In [6]:
```
for i in range(5):
    img=mpimg.imread(img_dir+data_df.head(5)['FileName'][i])
    print(data_df.head(5)['action'][i])
    print(data_df.head(5)['action_class'][i])
    imgplot = plt.imshow(img)
```

```
walking_the_dog
Interacting_with_animal
```

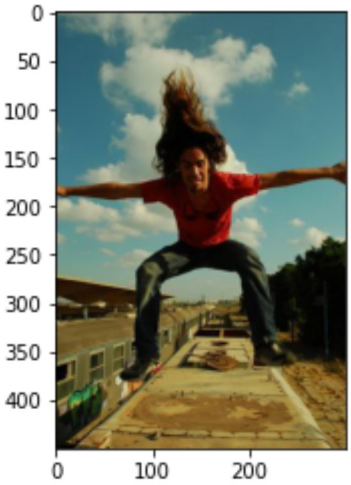
```
riding_a_bike
other_activity
```

```
gardening
domestic_work
```



```
cooking
domestic_work
```


```
jumping
other_activity
```



✓ Observations:

- Images loaded in correctly
- The images are all of different dimensions, resizing might be required
- Images are coloured, so there will be a 3-channel input into the model

In [6]:

Getting the max and min x and y values from each image

In [7]:

```
max_x = 0
max_y = 0
min_x = 1000
min_y = 1000

for i in range(len(data_df)):
    img=mpimg.imread(img_dir+data_df['FileName'][i])
    imgplot = plt.imshow(img)
    img_size = imgplot.get_size()
```

```
    img_x = img_size[1]
    img_y = img_size[0]

    if(img_x > max_x):
        max_x = img_x

    if(img_y > max_y):
        max_y = img_y

    if(img_x < min_x):
        min_x = img_x

    if(img_y < min_y):
        min_y = img_y

print("Max X: "+str(max_x))
print("Max Y: "+str(max_y))
print("Min X: "+str(min_x))
```

```
Max X: 961
Max Y: 960
Min X: 200
Min Y: 200
```

✓ Observations:

- The images vary quite greatly in dimensions, and for our model we want them all to have the same dimensions, while also retaining the characteristics of each image that makes them have an associated image. There are two major techniques for handling this, as pointed out over here - https://wandb.ai/ayush-thakur/dl-question-bank/reports/How-to-Handle-Images-of-Different-Sizes-in-a-Convolutional-Neural-Network--VmlldzoyMDk3NzQ (https://wandb.ai/ayush-thakur/dl-question-bank/reports/How-to-Handle-Images-of-Different-Sizes-in-a-Convolutional-Neural-Network--VmlldzoyMDk3NzQ)
- Either an average size compromise or a definitive size based on the baseline model will be used

In [10]:

Basic information from data frame

In [8]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3030 entries, 0 to 3029
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   FileName      3030 non-null   object
 1   action        3030 non-null   object
 2   action_class  3030 non-null   object
dtypes: object(3)
memory usage: 71.1+ KB
```

✓ Observations:

- There are no missing values in the dataset, so we don't have to do anything to deal with it

In [8]:

Class Distribution

Now we can look at the distribution of the different action classes, and each subclass within the action classes

In [9]:

```
# printing out the value counts for each column
for column in data_df.columns:
    print(column + ": " + str(data_df[column].unique()))
    print(data_df[column].value_counts())
```

```
FileName: ['Img_1000.jpg' 'Img_1001.jpg' 'Img_1002.jpg' ... 'Img_9
87.jpg'
 'Img_988.jpg' 'Img_995.jpg']
Img_4955.jpg    1
Img_7844.jpg    1
Img_1804.jpg    1
Img_3158.jpg    1
Img_5582.jpg    1
               ..
Img_8929.jpg    1
Img_5588.jpg    1
Img_4866.jpg    1
Img_2934.jpg    1
Img_1691.jpg    1
Name: FileName, Length: 3030, dtype: int64

action: ['walking_the_dog' 'riding_a_bike' 'gardening' 'cooking' '
jumping'
 'cutting_vegetables' 'watching_TV' 'cleaning_the_floor'
 'shooting_an_arrow' 'texting_message' 'playing_violin' 'feeding_a
_horse'
 'taking_photos' 'washing_dishes' 'riding_a_horse' 'rowing_a_boat'
 'playing_guitar' 'climbing' 'running' 'phoning' 'using_a_computer
']
riding_a_horse        196
jumping               195
climbing              195
riding_a_bike         193
walking_the_dog       193
playing_guitar        189
cooking               188
```

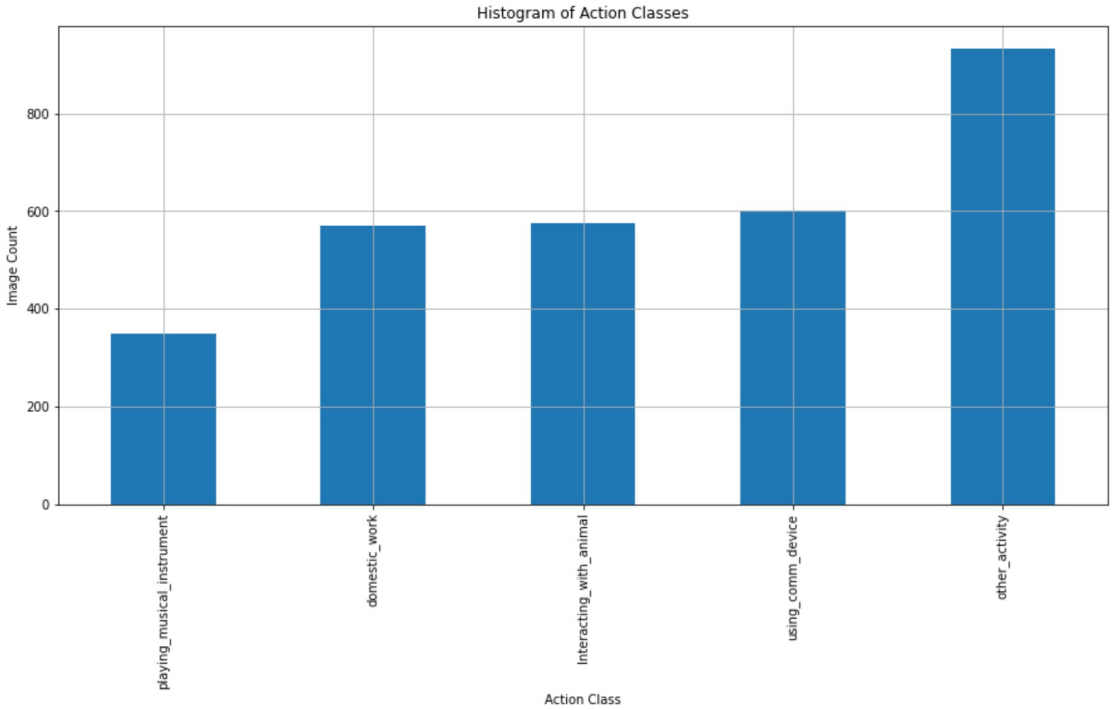
✓ Observations:

- There seems to be an uneven distribution, it's worth plotting this out to visualize the differences better

In [9]:

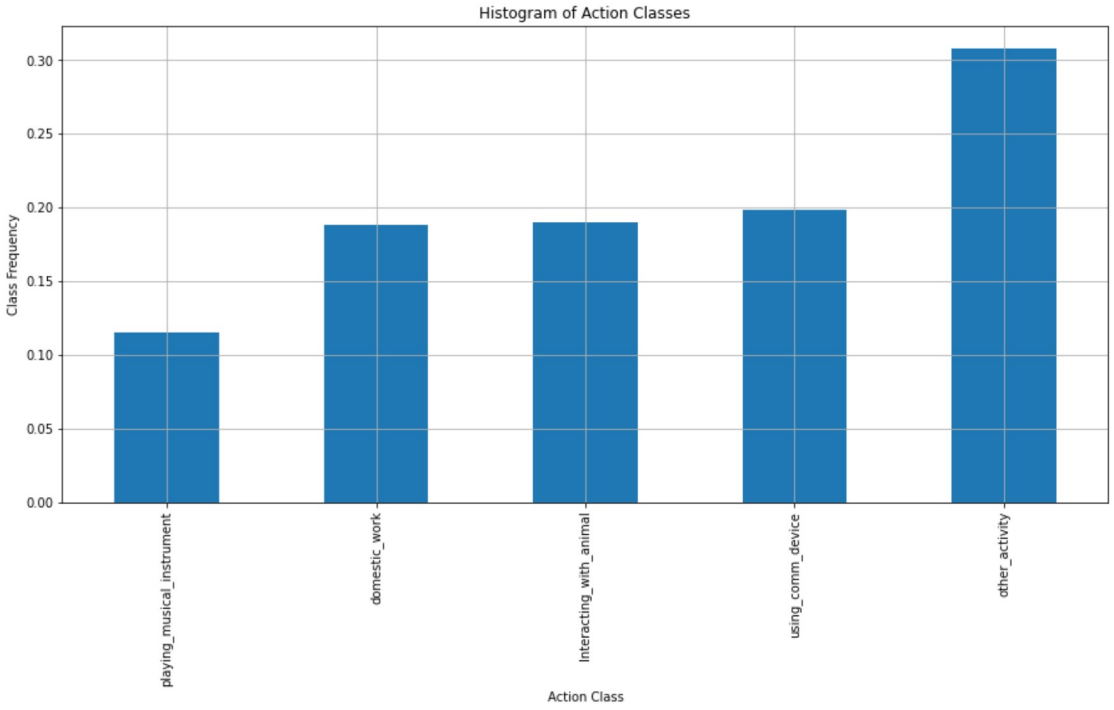
Distribution of Action Class

In [10]:
```
plt.figure(figsize=(15,7))
data_df.action_class.value_counts(ascending=True).plot(kind='bar',
plt.xlabel("Action Class")
plt.ylabel("Image Count")
plt.title("Histogram of Action Classes")
plt.show()
```




```
playing_musical_instrument    349
domestic_work                 570
Interacting_with_animal       576
using_comm_device             602
other_activity                933
Name: action_class, dtype: int64
```

In [11]:
```
plt.figure(figsize=(15,7))
data_df.action_class.value_counts(normalize=True, ascending=True).p
plt.xlabel("Action Class")
plt.ylabel("Class Frequency")
plt.title("Histogram of Action Classes")
plt.show()
```




```
playing_musical_instrument    0.115182
domestic_work                 0.188119
Interacting_with_animal       0.190099
using_comm_device             0.198680
other_activity                0.307921
Name: action_class, dtype: float64
```

✓ Observations:

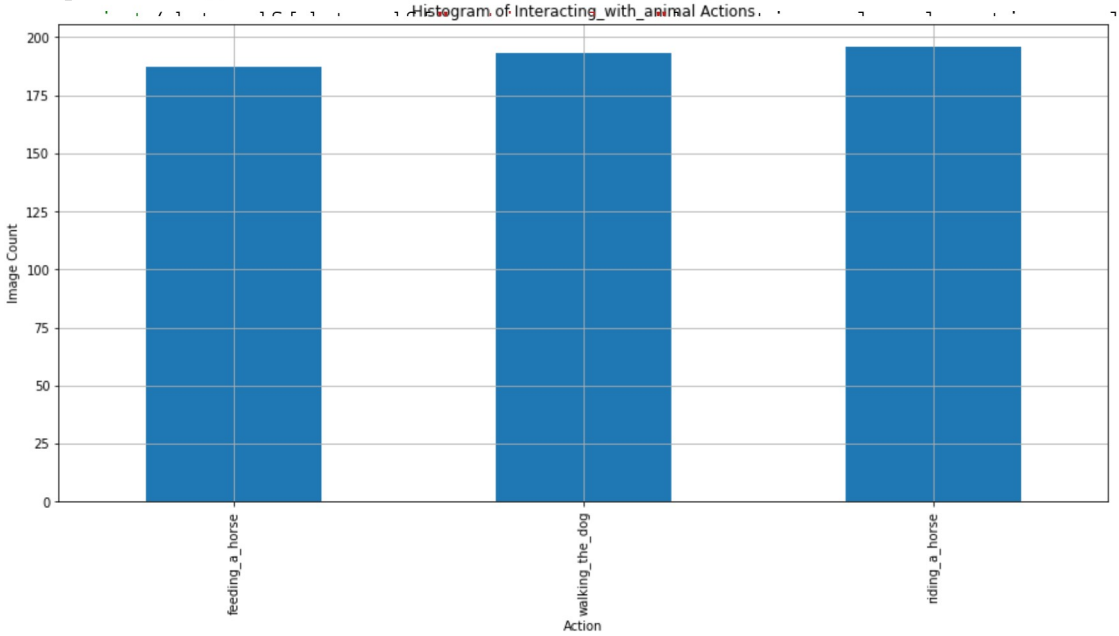
- There definitely is a class imbalance in this dataset, with "other_activity" having as many as 3 times the number of images as "playing musical instrument"
- The classes "domestic work", "interacting with animal" and "using_comm_device" are nearly evenly represented in this dataset

In [11]:
```
```

Distribution of Action

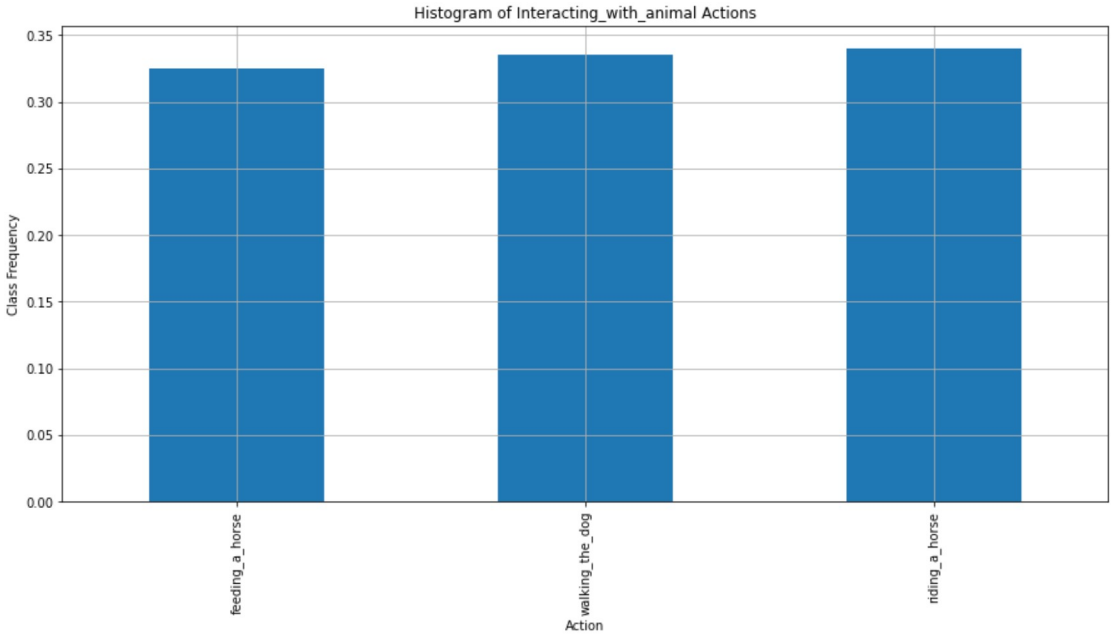
In [12]:
```
for action_class in data_df['action_class'].unique():

    plt.figure(figsize=(15,7))
    data_df[data_df["action_class"]==action_class].action.value_cou
    plt.xlabel("Action")
    #plt.xticks(rotation=35)
    plt.ylabel("Image Count")
    plt.title("Histogram of " + action_class + " Actions")
```

```
plt.show()
```



In [13]:

```
for action_class in data_df['action_class'].unique():

    plt.figure(figsize=(15,7))
    data_df[data_df["action_class"]==action_class].action.value_cou
    plt.xlabel("Action")
    #plt.xticks(rotation=35)
    plt.ylabel("Class Frequency")
    plt.title("Histogram of " + action_class + " Actions")
    plt.show()
    print(data_df[data_df["action_class"]==action_class].action.val
```



✓ Observations:

- Actions of action classes "playing_musical_instrument" and "interacting_with_animal" are well balanced
- The other 3 action classes have a lot of imbalance with their sub actions

In [13]:

Performance metric choice

- action_class: **Accuracy**. Even though there is an imbalance in class distribution, this is something we can accommodate for with data augmentation. Moreover, if we consider the distribution ignoring "other_activity", it is not too imbalanced. "other_activity" is a generic action_class for arguably a more diverse set of actions than the other action_classes. Thus, the standard accuracy metric should be good enough.
- action: **Top 5 Categorical Accuracy**. Chosen because there are 21 classes in total, with a wide variety of different actions, that can sometimes be interpreted differently. For example, "cooking", "washing dishes", and "cutting vegetables" are 3 activites that tend to have the person doing them in a similar posture. Our deep learning model may interpret this posture to be the primary determinant of classifying these. Moreover, sometimes there is overlap in the images themselves. For example, a person may be cutting vegetables, but have the stove on with food cooking and dishes in the sink all simultaneously. This can confuse the model, so having a "top 5" seems more appropriate.

Data Loader

Strategy

- Resize all the images to a unified size, and save to a new dataframe
- Take a sample of images, and try different augmentations on them, visualizing all the augmentations
- Determine the parameters (or range of parameters) for each augmentation within which the image is still usable
- Write a function to generate permutations and combinations of each of the augmentations for each image, preserving the label

In [14]:

Out[14]:

	FileName	action	action_class
0	Img_1000.jpg	walking_the_dog	Interacting_with_animal
1	Img_1001.jpg	riding_a_bike	other_activity
2	Img_1002.jpg	gardening	domestic_work
3	Img_1008.jpg	cooking	domestic_work
4	Img_1010.jpg	jumping	other_activity
...	...	...	...
3025	Img_977.jpg	texting_message	using_comm_device
3026	Img_980.jpg	feeding_a_horse	Interacting_with_animal
3027	Img_987.jpg	riding_a_bike	other_activity
3028	Img_988.jpg	jumping	other_activity
3029	Img_995.jpg	playing_guitar	playing_musical_instrument

3030 rows × 3 columns

In [14]:

Image Resizing

- 224x224 was chosen as it is a common image size to input for images that convey more complex data
- Furthermore, upsizing smaller images to a higher resolution might make them lose some details due to fuzziness
- Higher resolution images will also require a lot more computational time for network

In [15]:
```
# UNCOMMENT THE LINE BELOW TO CREATE THE NEW DIRECTORY
!mkdir "images_scaled"
for i in range(len(data_df)):
    img=mpimg.imread(img_dir+data_df['FileName'][i])
    #imgplot = plt.imshow(img)
    image_scaled = tf.image.convert_image_dtype(img, tf.float32)
    image_scaled = tf.image.resize(image_scaled, (224, 224))
```

In [16]:
```
data_df.action = pd.Categorical(data_df.action)
data_df["action_num"] = data_df.action.cat.codes

data_df.action_class = pd.Categorical(data_df.action_class)
data_df["action_class_num"] = data_df.action_class.cat.codes

data_df["FileName_scaled"] = "images_scaled/" + data_df['FileName']
```

In [17]:

Out[17]:

	FileName	action	action_class	action_num	action_class_nu
0	Img_1000.jpg	walking_the_dog	Interacting_with_animal	18	
1	Img_1001.jpg	riding_a_bike	other_activity	10	
2	Img_1002.jpg	gardening	domestic_work	5	
3	Img_1008.jpg	cooking	domestic_work	2	
4	Img_1010.jpg	jumping	other_activity	6	
...	...	...	...	...	
3025	Img_977.jpg	texting_message	using_comm_device	16	
3026	Img_980.jpg	feeding_a_horse	Interacting_with_animal	4	
3027	Img_987.jpg	riding_a_bike	other_activity	10	
3028	Img_988.jpg	jumping	other_activity	6	
3029	Img_995.jpg	playing_guitar	playing_musical_instrument	8	

3030 rows × 6 columns

Image augmentations

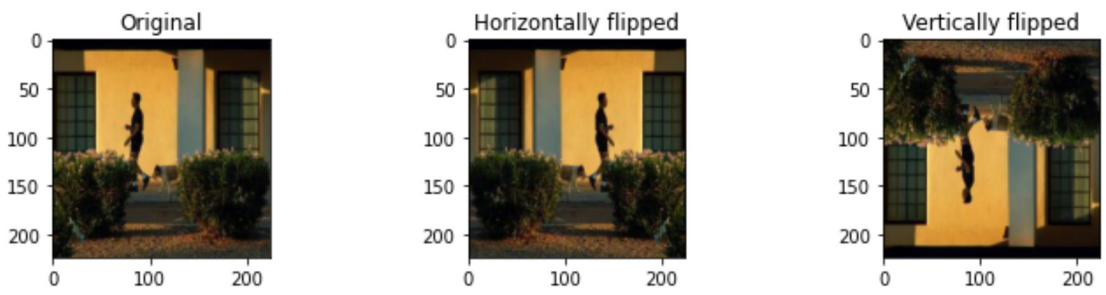
Strategy

- The best parameters were picked from each observation, and then combinations of all of them were looked at on images to see if the images still held up to being what theyre supposed to look like as per the label

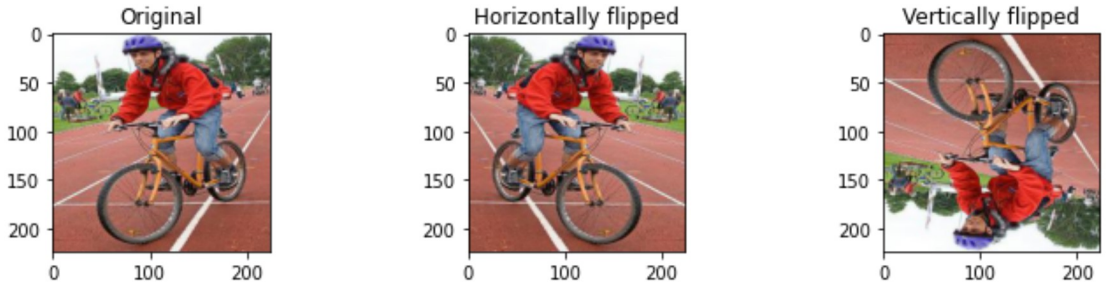
Horizontal/Vertical flipping

In [18]:
```
for i in range(5):
    #plt.subplots(1,3)
    plt.figure(1)
    plt.figure(figsize=(12,5))
    plt.subplot(231)
    plt.title("Original")
    img=mpimg.imread(data_df.head(5)['FileName_scaled'][i])
    print(data_df.head(5)['action'][i])
    print(data_df.head(5)['action_class'][i])
    plt.imshow(img)

    #plt.show()

    #Horizontally flipped
    hflip= iaa.Fliplr(p=1.0)
    hflipped_image= hflip.augment_image(img)

    #Vertically flipped
    vflip= iaa.Flipud(p=1.0)
    vflipped_image= vflip.augment_image(img)


    plt.subplot(232)
    plt.title("Horizontally flipped")
    plt.imshow(hflipped_image)
    plt.subplot(233)
    plt.imshow(vflipped_image)
    plt.title("Vertically flipped")
    plt.show()

    print("-----------------------------------------------------------

    #plt.show()
```

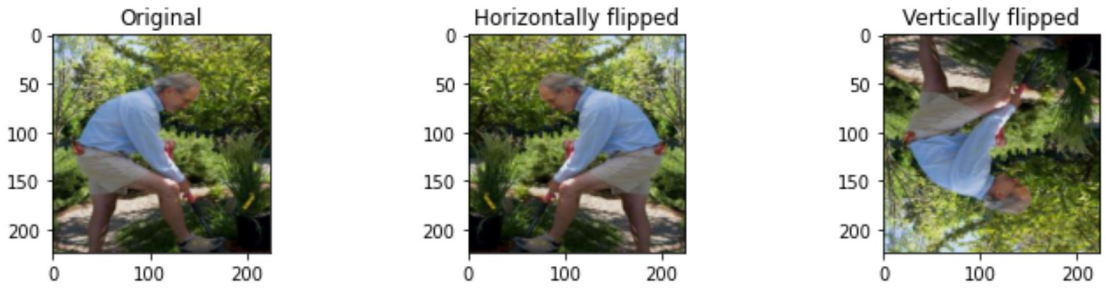
```
walking_the_dog
Interacting_with_animal

<Figure size 432x288 with 0 Axes>
```



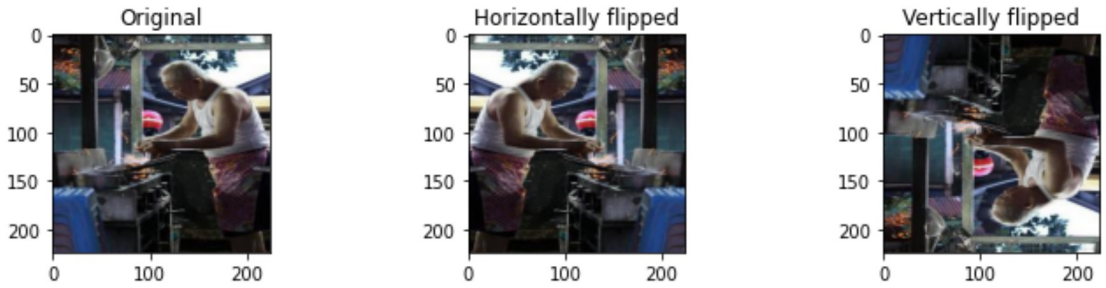
```
--------------------------------------------------------------------
--------------------------------------------------
riding_a_bike
other_activity

<Figure size 432x288 with 0 Axes>
```



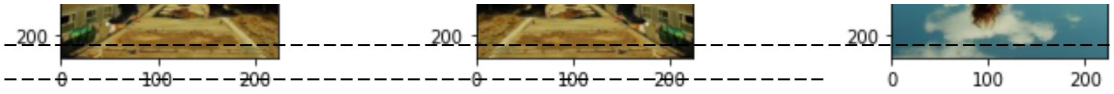

```
--------------------------------------------------------------------
--------------------------------------------------
gardening
domestic_work

<Figure size 432x288 with 0 Axes>
```




```
--------------------------------------------------------------------
--------------------------------------------------
cooking
domestic_work

<Figure size 432x288 with 0 Axes>
```




```
--------------------------------------------------------------------
--------------------------------------------------
jumping
other_activity

<Figure size 432x288 with 0 Axes>
```



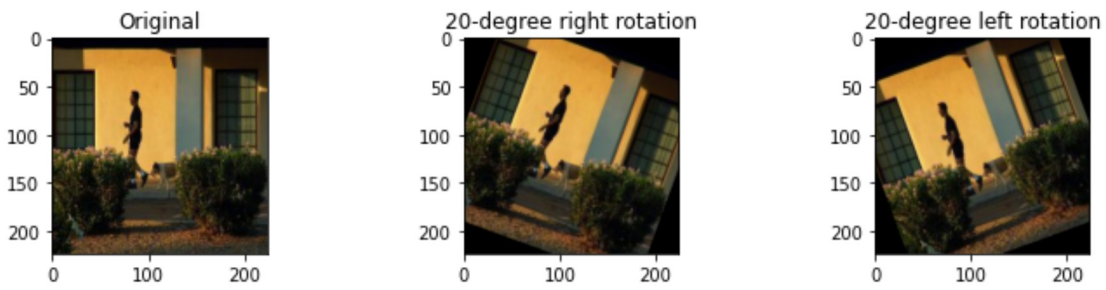
In [18]:

Rotation

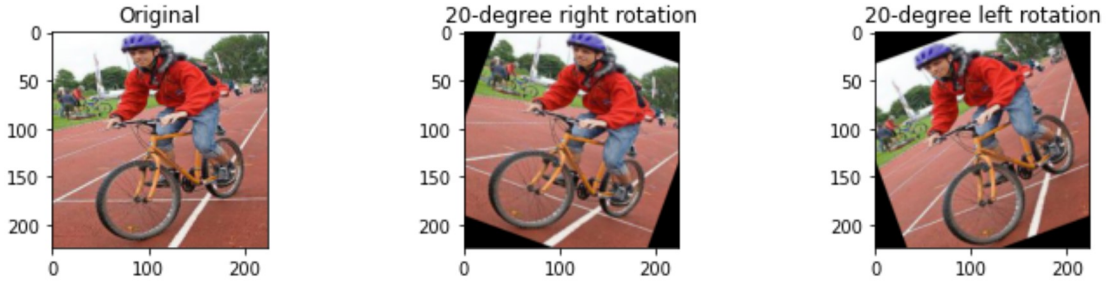
In [19]:

```
rotation_degree = 20
for i in range(5):
    #plt.subplots(1,3)
    plt.figure(1)
    plt.figure(figsize=(12,5))
    plt.subplot(231)
    plt.title("Original")
    img=mpimg.imread(data_df.head(5)['FileName_scaled'][i])
    print(data_df.head(5)['action'][i])
    print(data_df.head(5)['action_class'][i])
    plt.imshow(img)

    #plt.show()

    # clockwise rotation
    rot = iaa.Affine(rotate=(rotation_degree,rotation_degree))
    rot_clockwise_image_right = rot.augment_image(img)

    # clockwise rotation
    rot2 = iaa.Affine(rotate=(-rotation_degree,-rotation_degree))
    rot_clockwise_image_left = rot2.augment_image(img)


    plt.subplot(232)
    plt.title(str(rotation_degree)+"-degree right rotation")
    plt.imshow(rot_clockwise_image_right)

    plt.subplot(233)
    plt.imshow(rot_clockwise_image_left)
    plt.title(str(rotation_degree)+"-degree left rotation")
    plt.show()

    print("------------------------------------------------------------

    #plt.show()
```

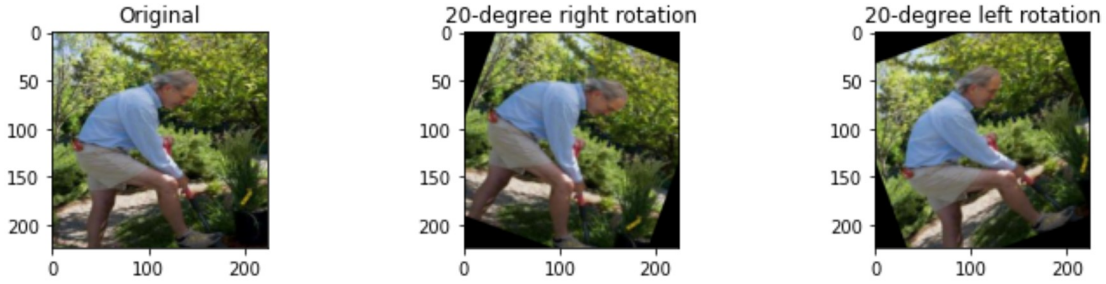
```
walking_the_dog
Interacting_with_animal

<Figure size 432x288 with 0 Axes>
```



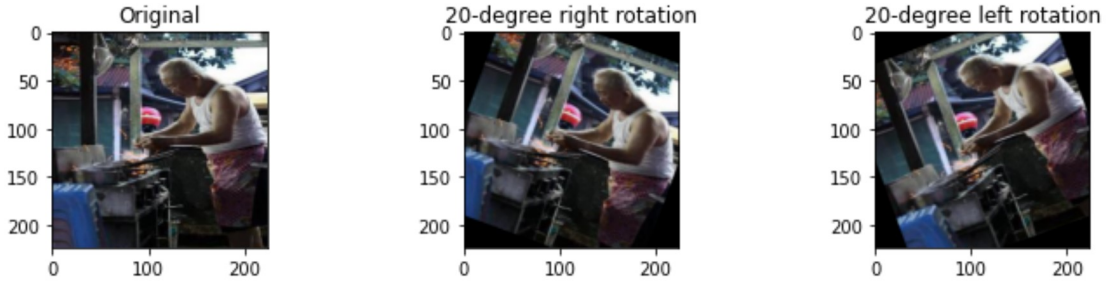
```
------------------------------------------------------------------
--------------------------------------------------
riding_a_bike
other_activity

<Figure size 432x288 with 0 Axes>
```



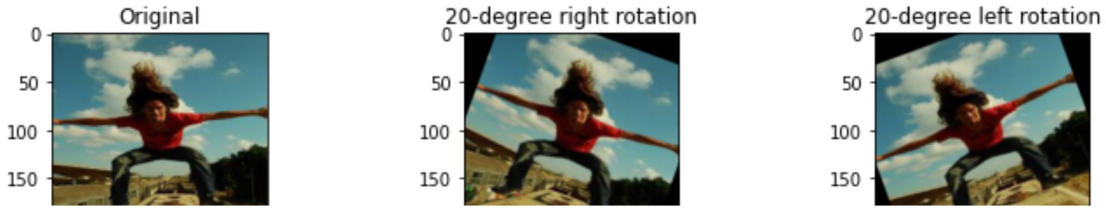

```
------------------------------------------------------------------
--------------------------------------------------
gardening
domestic_work

<Figure size 432x288 with 0 Axes>
```

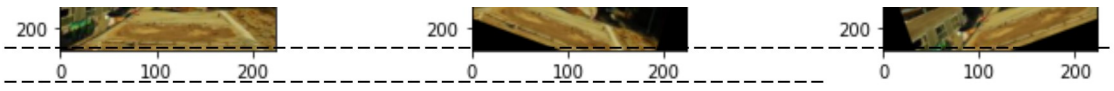



```
------------------------------------------------------------------
--------------------------------------------------
cooking
domestic_work

<Figure size 432x288 with 0 Axes>
```




```
------------------------------------------------------------------
--------------------------------------------------
jumping
other_activity

<Figure size 432x288 with 0 Axes>
```





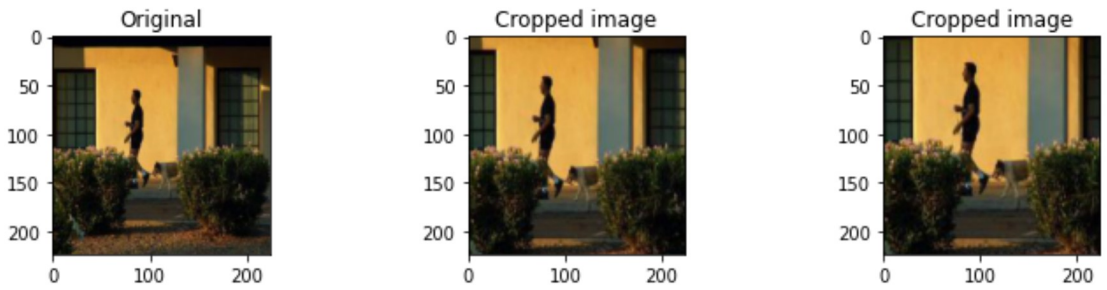
In [19]:

Image cropping

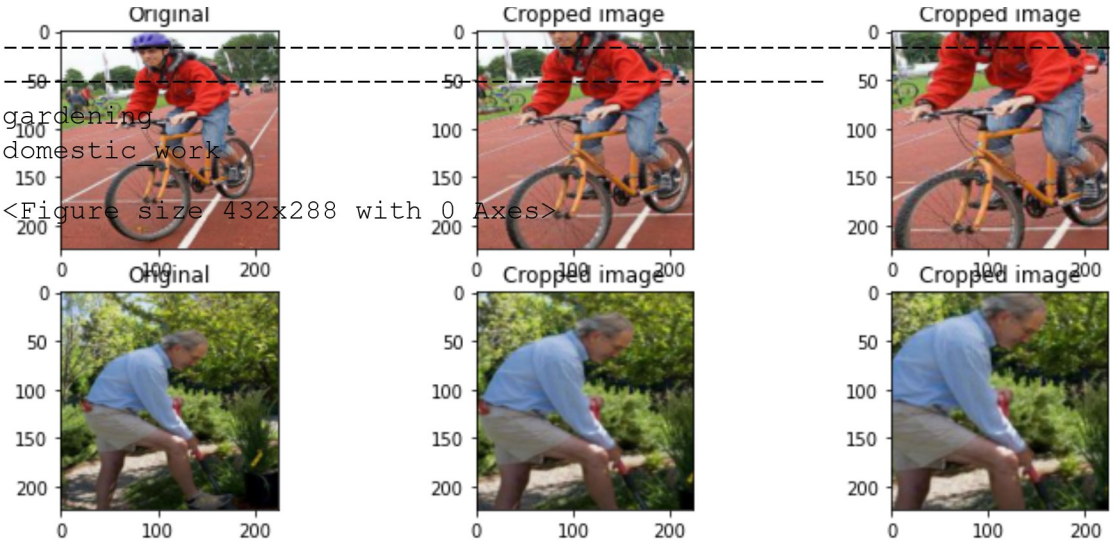
In [20]:

```
for i in range(5):


    plt.figure(1)
    plt.figure(figsize=(12,5))
    plt.subplot(231)
    plt.title("Original")
    img=mpimg.imread(data_df.head(5)['FileName_scaled'][i])
    print(data_df.head(5)['action'][i])
    print(data_df.head(5)['action_class'][i])
    plt.imshow(img)

    crop = iaa.Crop(percent=(0.1, 0.1)) # crop image
    crop_image1=crop.augment_image(img)

    crop = iaa.Crop(percent=(0.1, 0.2)) # crop image
    crop_image2=crop.augment_image(img)


    plt.subplot(232)
    plt.title("Cropped image")
    plt.imshow(crop_image1)

    plt.subplot(233)
    plt.title("Cropped image")
    plt.imshow(crop_image2)

    plt.show()

    print("-----------------------------------------------------------

    #plt.show()
```

```
walking_the_dog
Interacting_with_animal

<Figure size 432x288 with 0 Axes>
```




```
------------------------------------------------------------------
------------------------------------------------
riding_a_bike
other_activity

<Figure size 432x288 with 0 Axes>
```

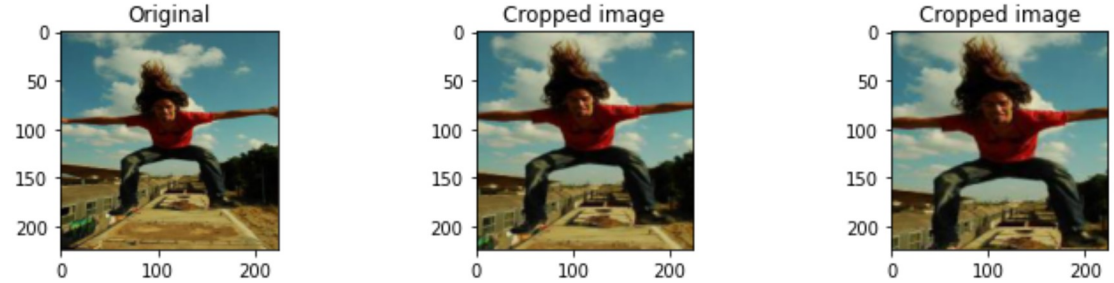



```
gardening
domestic_work
<Figure size 432x288 with 0 Axes>
```



```
cooking
domestic_work

<Figure size 432x288 with 0 Axes>
```


```
jumping
other_activity

<Figure size 432x288 with 0 Axes>
```



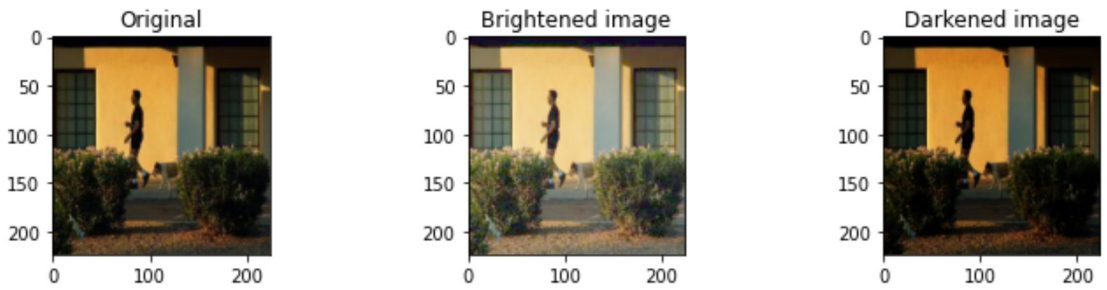
In [20]:

Gamma adjustment (brightening and darkening)

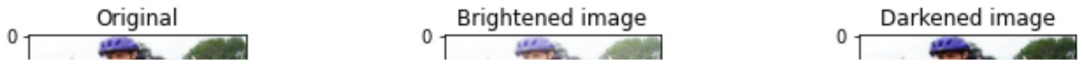
In [21]:

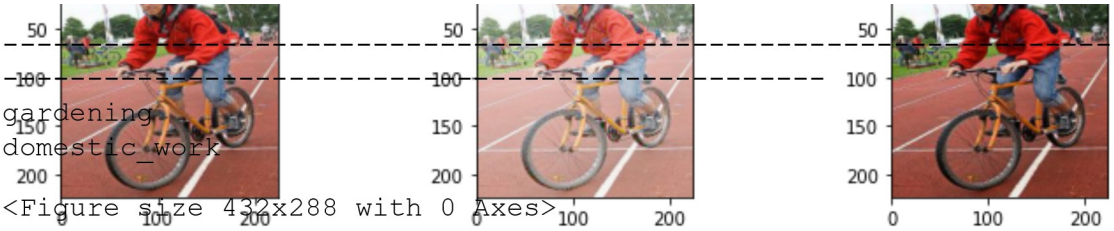
```
for i in range(5):


    plt.figure(1)
    plt.figure(figsize=(12,5))
    plt.subplot(231)
    plt.title("Original")
    img=mpimg.imread(data_df.head(5)['FileName_scaled'][i])
    print(data_df.head(5)['action'][i])
    print(data_df.head(5)['action_class'][i])
    plt.imshow(img)

   # bright
    contrast1=iaa.GammaContrast(gamma=0.5)
    brightened_image = contrast1.augment_image(img)

    #dark
    contrast2=iaa.GammaContrast(gamma=1.3)
    darkened_image = contrast2.augment_image(img)


    plt.subplot(232)
    plt.title("Brightened image")
    plt.imshow(brightened_image)

    plt.subplot(233)
    plt.title("Darkened image")
    plt.imshow(darkened_image)

    plt.show()

    print("-----------------------------------------------------------

    #plt.show()
```

```
walking_the_dog
Interacting_with_animal

<Figure size 432x288 with 0 Axes>
```

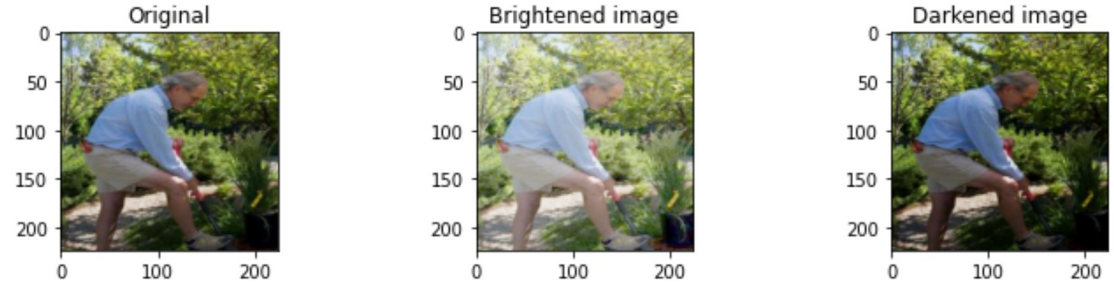



```
------------------------------------------------------------------
------------------------------------------------
riding_a_bike
other_activity

<Figure size 432x288 with 0 Axes>
```

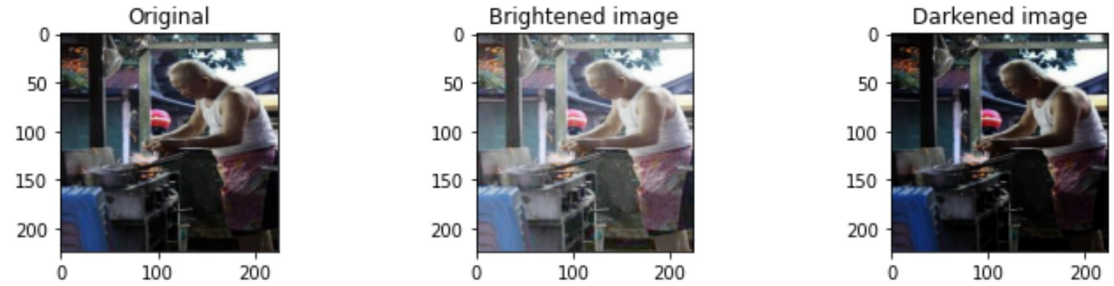




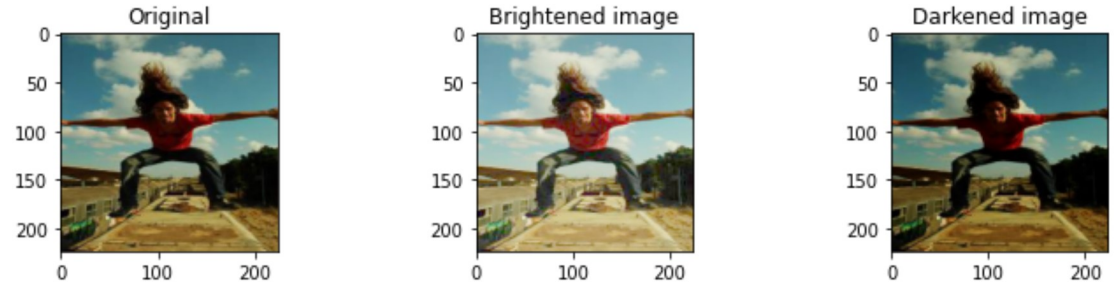
```
gardening
domestic_work
<Figure size 432x288 with 0 Axes>
```




```
--------------------------------------------------------------------
--------------------------------------------------
cooking
domestic_work

<Figure size 432x288 with 0 Axes>
```




```
--------------------------------------------------------------------
--------------------------------------------------
jumping
other_activity

<Figure size 432x288 with 0 Axes>
```




```
--------------------------------------------------------------------
--------------------------------------------------
```

In [21]:

Adding random noise

In [22]:

```
for i in range(5):


    plt.figure(1)
    plt.figure(figsize=(12,5))
    plt.subplot(221)
    plt.title("Original")
    img=mpimg.imread(data_df.head(5)['FileName_scaled'][i])
    print(data_df.head(5)['action'][i])
    print(data_df.head(5)['action_class'][i])
    plt.imshow(img)


    noisy_image= sku.random_noise(img)


    plt.subplot(222)
    plt.title("Noisy image")
    plt.imshow(noisy_image)

    plt.show()

    print("------------------------------------------------------------

    #plt.show()
```

```
walking_the_dog
Interacting_with_animal

<Figure size 432x288 with 0 Axes>
```


```
------------------------------------------------------------------------
------------------------------------------------
riding_a_bike
other_activity

<Figure size 432x288 with 0 Axes>
```

```
------------------------------------------------------------------------
------------------------------------------------
gardening
domestic_work

<Figure size 432x288 with 0 Axes>
```


```
------------------------------------------------------------------------
------------------------------------------------
cooking
domestic_work

<Figure size 432x288 with 0 Axes>
```


```
------------------------------------------------------------------------
------------------------------------------------
jumping
other_activity

<Figure size 432x288 with 0 Axes>
```


```
------------------------------------------------------------------------
------------------------------------------------
```

In [22]:

Trying random combinations together

In [23]:

```
for i in range(5):


    plt.figure(1)
    plt.figure(figsize=(12,5))
    plt.subplot(221)
    plt.title("Original")
    img=mpimg.imread(data_df.head(5)['FileName_scaled'][i])
    print(data_df.head(5)['action'][i])
    print(data_df.head(5)['action_class'][i])
    plt.imshow(img)

    # add noise
    noisy_image= sku.random_noise(img)

    # brighten
    contrast1=iaa.GammaContrast(gamma=0.5)
    brightened_image = contrast1.augment_image(noisy_image)

    # rotate
    rot = iaa.Affine(rotate=(rotation_degree,rotation_degree))
    rot_clockwise_image_right = rot.augment_image(brightened_image)

    # crop
    crop = iaa.Crop(percent=(0.1, 0.1))
    crop_image1=crop.augment_image(rot_clockwise_image_right)

    # flip horizontally
    hflip= iaa.Fliplr(p=1.0)
    hflipped_image= hflip.augment_image(crop_image1)

    plt.subplot(222)
    plt.title("Augmented image")
    plt.imshow(hflipped_image)

    plt.show()

    print("----------------------------------------------------------

    #plt.show()
```

```
walking_the_dog
Interacting_with_animal

<Figure size 432x288 with 0 Axes>
```

```
Clipping input data to the valid range for imshow with RGB data
([0..1] for floats or [0..255] for integers).

--------------------------------------------------------------------------
------------------------------------------------
riding_a_bike
other_activity

<Figure size 432x288 with 0 Axes>
```


```
--------------------------------------------------------------------------
------------------------------------------------
gardening
domestic_work

<Figure size 432x288 with 0 Axes>
```


```
Clipping input data to the valid range for imshow with RGB data
([0..1] for floats or [0..255] for integers).

--------------------------------------------------------------------------
------------------------------------------------
cooking
domestic_work

<Figure size 432x288 with 0 Axes>
```


```
--------------------------------------------------------------------------
------------------------------------------------
```

```
jumping
other_activity
<Figure size 432x288 with 0 Axes>
```


```
------------------------------------------------------------------------
------------------------------------------------
```

✓ Observations:

- The images seem to retain their essence in terms of their classification label, while still being different to the original, as we want the augmentations to be

In [23]:

Putting it all together

In [24]:

Out[24]:

	FileName	action	action_class	action_num	action_class_nu
0	Img_1000.jpg	walking_the_dog	Interacting_with_animal	18	
1	Img_1001.jpg	riding_a_bike	other_activity	10	
2	Img_1002.jpg	gardening	domestic_work	5	
3	Img_1008.jpg	cooking	domestic_work	2	
4	Img_1010.jpg	jumping	other_activity	6	
...	...	...	...	...	
3025	Img_977.jpg	texting_message	using_comm_device	16	
3026	Img_980.jpg	feeding_a_horse	Interacting_with_animal	4	
3027	Img_987.jpg	riding_a_bike	other_activity	10	
3028	Img_988.jpg	jumping	other_activity	6	
3029	Img_995.jpg	playing_guitar	playing_musical_instrument	8	

3030 rows × 6 columns

First we split the data into train, test, and validation sets

In [25]:

```
train_data_df, test_data_df = train_test_split(data_df, test_size=0
train_data_df, val_data_df = train_test_split(train_data_df, test_s
```

In [26]:

Out[26]:

	FileName	action	action_class	action_num	action_class_num
1494	Img_5261.jpg	feeding_a_horse	Interacting_with_animal	4	0
483	Img_2455.jpg	feeding_a_horse	Interacting_with_animal	4	0
1052	Img_4043.jpg	cooking	domestic_work	2	1
2695	Img_8659.jpg	climbing	other_activity	1	2
2702	Img_868.jpg	running	other_activity	13	2
...	...	...	...	...	...
1804	Img_6122.jpg	feeding_a_horse	Interacting_with_animal	4	0
2606	Img_8406.jpg	cooking	domestic_work	2	1
33	Img_1098.jpg	climbing	other_activity	1	2
1635	Img_5642.jpg	watching_TV	using_comm_device	20	4
687	Img_2982.jpg	watching_TV	using_comm_device	20	4

1704 rows × 6 columns

In [27]:

Out[27]:

	FileName	action	action_class	action_num	action_class_num
2704	Img_8683.jpg	shooting_an_arrow	other_activity	14	2
1095	Img_419.jpg	climbing	other_activity	1	2
2969	Img_9398.jpg	cooking	domestic_work	2	1
1316	Img_4789.jpg	walking_the_dog	Interacting_with_animal	18	0
881	Img_3555.jpg	walking_the_dog	Interacting_with_animal	18	0
...	...	...	...	...	...
2486	Img_808.jpg	riding_a_horse	Interacting_with_animal	11	0
2175	Img_7183.jpg	cooking	domestic_work	2	1
551	Img_2611.jpg	riding_a_horse	Interacting_with_animal	11	0
2666	Img_8559.jpg	washing_dishes	domestic_work	19	1
2964	Img_9389.jpg	running	other_activity	13	2

568 rows × 6 columns

In [28]:

Out[28]:

	FileName	action	action_class	action_num	action_class_n
535	Img_257.jpg	rowing_a_boat	other_activity	12	
1369	Img_4929.jpg	riding_a_bike	other_activity	10	
1171	Img_4372.jpg	climbing	other_activity	1	
2921	Img_9279.jpg	walking_the_dog	Interacting_with_animal	18	
736	Img_3148.jpg	texting_message	using_comm_device	16	
...	...	...	...	...	

	FileName	action	action_class	action_num	action_class_n
1076	Img_4112.jpg	riding_a_horse	Interacting_with_animal	11	
545	Img_2596.jpg	cleaning_the_floor	domestic_work	0	
1645	Img_5672.jpg	jumping	other_activity	6	
894	Img_3588.jpg	cooking	domestic_work	2	
1387	Img_4963.jpg	playing_violin	playing_musical_instrument	9	

In [29]:
```
augmentations = ['hflip', 'rotate', 'crop', 'gamma', 'noise']
```

In [30]:

Out[30]:
```
[('hflip',),
 ('rotate',),
 ('crop',),
 ('gamma',),
 ('noise',),
 ('hflip', 'rotate'),
 ('hflip', 'crop'),
 ('hflip', 'gamma'),
 ('hflip', 'noise'),
 ('rotate', 'crop'),
 ('rotate', 'gamma'),
 ('rotate', 'noise'),
 ('crop', 'gamma'),
 ('crop', 'noise'),
 ('gamma', 'noise'),
 ('hflip', 'rotate', 'crop'),
 ('hflip', 'rotate', 'gamma'),
 ('hflip', 'rotate', 'noise'),
 ('hflip', 'crop', 'gamma'),
 ('hflip', 'crop', 'noise'),
 ('hflip', 'gamma', 'noise'),
 ('rotate', 'crop', 'gamma'),
 ('rotate', 'crop', 'noise'),
 ('rotate', 'gamma', 'noise'),
 ('crop', 'gamma', 'noise'),
 ('hflip', 'rotate', 'crop', 'gamma'),
 ('hflip', 'rotate', 'crop', 'noise'),
 ('hflip', 'rotate', 'gamma', 'noise'),
 ('hflip', 'crop', 'gamma', 'noise'),
 ('rotate', 'crop', 'gamma', 'noise'),
 ('hflip', 'rotate', 'crop', 'gamma', 'noise')]
```

✓ Observations:

- There are 32 variations for each image, excluding the original

In [30]:

Writing the generator

For this assignment I wrote a custom data generator, with a modified data loading mechanism from the class in the week 5 lab, as well as the custom generator found here

(https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly)

The data generator has the capability to augment the images, as well as load the images with the two classes, to be fed into our Mutli-Task learning model.

In [66]:

```
class DataGenerator(keras.utils.Sequence):

    # Initializer
    def __init__(self, data_frame, batch_size = 32, dim = (224, 224
                 shuffle = True, create_folder = False,
                 augment = False, file_name = '',  data_mean=0, dat

        self.batch_size = batch_size
        self.shuffle = shuffle
        self.create_folder = create_folder
        self.file_name = file_name
        self.augment = augment
        self.dim = dim
        self.n_channels = n_channels
        self.data_frame = data_frame
        self.image_column = image_column

        # if the data is to be augmented, we call the augment funct
        if self.augment == True:
            self.data_frame = self.__augment()
        # otherwise read the dataframe as is
        else:
            self.data_frame = data_frame

        # We have 2 labels since we're going to be doing multi-clas
        self.action_class_label = data_frame['action_class_num'].va
        self.action_class_ids = np.arange(len(self.action_class_lab

        self.action_label = data_frame['action_num'].values.tolist(
        self.action_ids = np.arange(len(self.action_label)).tolist(

        # Data normalization parameters
        self.data_mean = data_mean
        self.data_std = data_std

        self.on_epoch_end()

    def __len__(self):
        'Denotes the number of batches per epoch'
        return int(np.floor(len(self.action_class_ids) / self.batch_

    def __augment(self):
        'Augmentation function'

        rows_list = []
        if self.create_folder == True:
            !mkdir "images_augmented"

        # Augmentation parameters
        rotation_degree = 20
        augmentations = ['hflip', 'rotate', 'crop', 'gamma', 'noise
        augmentation_combinations = list(chain(*[combinations(augme

        for index, row in self.data_frame.iterrows():
```

```
            # Reading in the image file
            img=mpimg.imread(row['FileName_scaled'])
            og_filename = "images_augmented/"+ (row['FileName_scale
            dict_line = {"FileName": og_filename, "action":row["act
            rows_list.append(dict_line)
            tf.keras.preprocessing.image.save_img(og_filename, img)

            # Looping over each augmentation combination from above
            for augmentation in augmentation_combinations:

                img=mpimg.imread(row['FileName_scaled'])

                # Horizontal flip
                if 'hflip' in augmentation:

                    hflip = iaa.Fliplr(p=1.0)
                    img = hflip.augment_image(img)

                # Rotation
                if 'rotate' in augmentation:

                    # randomly selectes clockwise or anti-clockwise
                    rotation_orientation = random.choice([(rotation_
                    rot = iaa.Affine(rotate=rotation_orientation)
                    img = rot.augment_image(img)

                # Cropping
                if 'crop' in augmentation:

                    crop = iaa.Crop(percent=(0.1, 0.1))
                    img = crop.augment_image(img)

                # Gamme
                if 'gamma' in augmentation:

                    # randomly selects brightening or darkening
                    gamma = random.choice([0.5, 1.3])
                    contrast_brighten = iaa.GammaContrast(gamma=gam
                    img = contrast_brighten.augment_image(img)

                # Adding noise
                if 'noise' in augmentation:

                    img = sku.random_noise(img)

                # Saving the augmented image
                augmented_filename = "images_augmented/"+ (row['Fil
                tf.keras.preprocessing.image.save_img(augmented_fil
                dict_line = {"FileName": augmented_filename, "actio
                rows_list.append(dict_line)

        # Creating a new dataframe from the augmented images and sa
        augmented_df = pd.DataFrame(rows_list)
        augmented_df.to_csv(self.file_name, index = False)

        # Changing the dataframe's labels to categorical numbers
        augmented_df.action = pd.Categorical(augmented_df.action)
        augmented_df["action_num"] = augmented_df.action.cat.codes

        augmented_df.action_class = pd.Categorical(augmented_df.act
```

```
        augmented_df["action_class_num"] = augmented_df.action_clas

        return augmented_df


    def __getitem__(self, index):
        'Generate one batch of data for the given index'
        # Generate indexes of the batch
        indexes = self.indexes[index*self.batch_size:(index+1)*self

        # Find list of IDs
        data_ids_temp = [self.action_class_ids[k] for k in indexes]

        action_class_label_temp = [self.action_class_label[k] for k
        action_label_temp = [self.action_label[k] for k in indexes]

        # Generate data
        X, [action_class, action] = self.__data_generation(data_ids

        return X, [keras.utils.to_categorical(action_class, num_cla
        #return X, [action_class, action]

    def on_epoch_end(self):
        'Updates indexes after each epoch'
        self.indexes = np.arange(len(self.action_ids))
        if self.shuffle == True:
            np.random.shuffle(self.indexes)


    def __data_generation(self, data_ids_temp, action_class_label_t
        'Generates data containing batch_size samples'

        # Initialization
        X = np.empty((self.batch_size, *self.dim))#, self.n_channel
        action_class = np.empty((self.batch_size), dtype=int)
        action = np.empty((self.batch_size), dtype=int)

        # Generate data
        for i, ids in enumerate(data_ids_temp):

            X[i,] = self.__read_data_instance(data_ids_temp[i])
            action_class[i] = action_class_label_temp[i]
            action[i] = action_label_temp[i]


        #return X, keras.utils.to_categorical(y, num_classes=self.n
        return X, [action_class, action]

    def __read_data_instance(self, pid):
        # Read an image
        filepath = self.data_frame.iloc[pid][self.image_column]

        data = Image.open(filepath)

        data = np.asarray(data)

        X = data

        # Input normalization
        X = (X - self.data_mean)/self.data_std
```

```
        return X
```

In [ ]:

In [ ]:
```
'''
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0

training_generator = DataGenerator(
                                    data_frame = train_data_df,
                                    batch_size = BATCH_SIZE,
                                    data_mean = data_mean,
                                    data_std = data_std,
                                    create_folder = True,
                                    dim = (224, 224, 3),
                                    shuffle = True,
                                    augment = True,
                                    file_name = 'drive/MyDrive/trai
                                    )

'''
```

In [31]:
```
# If the augmentation is already done, since it's a lengthy process
```

```
Streaming output truncated to the last 5000 lines.
  inflating: images_augmented/Img_8696_('hflip', 'rotate').jpg
  inflating: images_augmented/Img_4638_('hflip', 'gamma', 'noise
').jpg
  inflating: images_augmented/Img_4754_('crop',).jpg
  inflating: images_augmented/Img_3117_('gamma', 'noise').jpg
  inflating: images_augmented/Img_1669_('rotate', 'gamma', 'noise
').jpg
  inflating: images_augmented/Img_5405_('gamma', 'noise').jpg
  inflating: images_augmented/Img_8563.jpg
  inflating: images_augmented/Img_1752_('crop', 'noise').jpg
  inflating: images_augmented/Img_6834_('rotate', 'crop', 'gamma
').jpg
  inflating: images_augmented/Img_269_('crop',).jpg
  inflating: images_augmented/Img_5529.jpg
  inflating: images_augmented/Img_9022_('hflip', 'gamma', 'noise
').jpg
  inflating: images_augmented/Img_3594_('noise',).jpg
  inflating: images_augmented/Img_6376_('rotate', 'crop', 'noise
```

In [32]:

In [33]:
```
train_data_df.action = pd.Categorical(train_data_df.action)
train_data_df["action_num"] = train_data_df.action.cat.codes

train_data_df.action_class = pd.Categorical(train_data_df.action_cl
```

In [34]:

Out[34]:

	FileName	action	action_class	act
0	images_augmented/Img_5261.jpg	feeding_a_horse	Interacting_with_animal	

	FileName	action	action_class	act
1	images_augmented/Img_5261_('hflip',).jpg	feeding_a_horse	Interacting_with_animal	
2	images_augmented/Img_5261_('rotate',).jpg	feeding_a_horse	Interacting_with_animal	
3	images_augmented/Img_5261_('crop',).jpg	feeding_a_horse	Interacting_with_animal	
4	images_augmented/Img_5261_('gamma',).jpg	feeding_a_horse	Interacting_with_animal	
...	...	...	...	
54523	images_augmented/Img_2982_('hflip', 'rotate', ...	watching_TV	using_comm_device	
54524	images_augmented/Img_2982_('hflip', 'rotate', ...	watching_TV	using_comm_device	
54525	images_augmented/Img_2982_('hflip', 'crop', 'g...	watching_TV	using_comm_device	
54526	images_augmented/Img_2982_('rotate', 'crop', '...	watching_TV	using_comm_device	
54527	images_augmented/Img_2982_('hflip', 'rotate', ...	watching_TV	using_comm_device	

In [ ]:

Multi-task Learning Model

Strategy

- We first create an evaluation pipeline
- Then we find a baseline model from previous research, and add the aspect of multi-task learning to it

Model Evaluation Setup

In [39]:

```
class Logger(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs=None):
        action_class_accuracy = logs.get('action_class_accuracy')
        action_accuracy = logs.get('action_accuracy')
        val_action_class_accuracy = logs.get('val_action_class_accu
        val_action_accuracy = logs.get('val_action_accuracy')

        action_class_top_k = logs.get('action_class_top_k_categoric
        action_top_k = logs.get('action_top_k_categorical_accuracy'
        val_action_class_top_k = logs.get('val_action_class_top_k_c
        val_action_top_k = logs.get('val_action_top_k_categorical_a


        print('='*30, epoch + 1, '='*30)
        print("")
        print(f'action_class_accuracy: {action_class_accuracy:.2f},
        print(f'val_action_class_accuracy: {val_action_class_accura
        print("")
        print(f'action_class_top_k_categorical_accuracy : {action_c
        print(f'val_action_class_top_k_categorical_accuracy : {val_
```

```
        print("")
```

In [40]:

```
def get_callbacks(model_name):

    checkpoint_path = model_name + "/cp.ckpt"

    all_callbacks = [
                tf.keras.callbacks.EarlyStopping(
                   monitor="val_action_top_k_categorical_accuracy"
                   min_delta=0,
                   patience=5,
                   verbose=1,
                   restore_best_weights=True,
                ),
                 tf.keras.callbacks.EarlyStopping(
                   monitor="val_action_class_accuracy",
                   min_delta=0,
                   patience=5,
                   verbose=1,
                   restore_best_weights=True,
                ),

                 tf.keras.callbacks.ModelCheckpoint(checkpoint_path
                                                    save_weights_only=
                                                    verbose=1,
                                                    save_best_only = T
                 Logger(),
                 tf.keras.callbacks.TensorBoard(log_dir = './logs')
    ]
```

In [41]:

```
def plotter(history_hold, metric = 'accuracy', loss='loss', ylim_me
    cycol = cycle('bgrcmk')
    plt.figure(figsize=(10,5))
    for name, item in history_hold.items():
        metric_train = item[metric]
        metric_val = item['val_' + metric]
        loss_train = item[loss]
        loss_val = item['val_' + loss]

        x_train = np.arange(0,len(loss_val))

        c=next(cycol)
        plt.style.use("ggplot")
        plt.subplot(1,2,1)
        plt.plot(x_train, metric_train, c+'-', label=name+'_train')
        plt.plot(x_train, metric_val, c+'--', label=name+'_val')

        plt.legend()
        plt.xlim([1, xlim_max])
        plt.ylim(ylim_metric)
        plt.xlabel('Epoch')
        plt.ylabel(metric)
        plt.grid(True)
        plt.title(metric)


        plt.subplot(1,2,2)
```

```
        plt.plot(x_train, loss_train, c+'-', label=name+'_train')
        plt.plot(x_train, loss_val, c+'--', label=name+'_val')

        plt.legend()
        plt.xlim([1, xlim_max])
        # plt.ylim(ylim_loss)
        plt.xlabel('Epoch')
        plt.ylabel(loss)
        plt.grid(True)
```

In [ ]:

In [ ]:

Resnet Implementation

Implementation taken mostly as-is from the week 6 lab, with a few minor tweaks to add additional flexibility with parameters

In [ ]:

```
class ResidualBlock(tf.keras.layers.Layer):

    # Initialize components of the model
    def __init__(self, filter_num, stride=1, reg_lambda=0.0, kernel
        super(ResidualBlock, self).__init__()
        self.conv1 = tf.keras.layers.Conv2D(filters=filter_num,
                                            kernel_size = kernel_si
                                            strides=stride,
                                            kernel_initializer=tf.k
                                            kernel_regularizer=tf.k
                                            padding="same")
        self.bn1 = tf.keras.layers.BatchNormalization(momentum=momen
        self.conv2 = tf.keras.layers.Conv2D(filters=filter_num,
                                            kernel_size = kernel_si
                                            strides=1,
                                            kernel_initializer=tf.k
                                            kernel_regularizer=tf.k
                                            padding="same")
        self.bn2 = tf.keras.layers.BatchNormalization(momentum=momen
        if stride != 1:
            self.downsample = tf.keras.Sequential()
            self.downsample.add(tf.keras.layers.Conv2D(filters=filt
                                                       kernel_size=
                                                       kernel_initi
                                                       kernel_regul
                                                       strides=stri
            self.downsample.add(tf.keras.layers.BatchNormalization(
        else:
            self.downsample = lambda x: x

    # Define the forward function
    def call(self, inputs, training=None, **kwargs):
        residual = self.downsample(inputs)

        x = self.conv1(inputs)
        x = self.bn1(x, training=training)
        x = tf.nn.relu(x)
        x = self.conv2(x)
        x = self.bn2(x, training=training)
```

```
        output = tf.nn.relu(tf.keras.layers.add([residual, x]))

        return output

    def get_config(self):

        config = super().get_config().copy()
        config.update({
            'conv1': self.conv1,
            'bn1': self.bn1,
            'conv2': self.conv2,
            'bn2': self.bn2,
            'downsample': self.downsample,
        })
```

In [ ]:

In [ ]:

Baseline Model

The baseline model selected is the original ResNet paper used on ImageNet. This was selected to test whether the original architecture can generalize well to other image recognition problems, and with a limited set of data

In [ ]:
```
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                    data_frame = train_data_df,
                                    batch_size = BATCH_SIZE,
                                    data_mean = data_mean,
                                    data_std = data_std,
                                    create_folder = False,
                                    dim = (224, 224, 3),
                                    shuffle = True,
                                    augment = False,
                                    file_name = 'drive/MyDrive/trai
                                    )

validation_generator = DataGenerator(
                                    data_frame = val_data_df,
                                    batch_size = BATCH_SIZE,
                                    data_mean = data_mean,
                                    data_std = data_std,
                                    create_folder = False,
                                    dim = (224, 224, 3),
                                    shuffle = True,
                                    augment = False,
                                    file_name = 'drive/MyDrive/vali
                                    image_column = "FileName_scaled
```

In [ ]:
```
filters = [64, 128, 256, 512]
```

```
block_size = [3,4,6,3]
reg_lambda = 0.00

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
 initial_learning_rate = 0.1, decay_steps = len(train_data_df)//BAT(
  staircase=False)
```

In [ ]:

```
first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")

conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
                kernel_regularizer=tf.keras.regularizers.l2(reg_lam
                padding = "same",
                name = "conv_1")(input_)


bn_1 = BatchNormalization(momentum=0.9)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)

for nFilters, nBlocks in zip(filters, block_size):

    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re

    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l2
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(flat_1)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l2(reg
                    kernel_initializer=tf.keras.initializers.Random
                    name =  "action")(flat_1)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
```

```
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
           'accuracy',
           tf.keras.metrics.TopKCategoricalAccuracy(
                k=5,
                name="top_k_categorical_accuracy",
                dtype=None
           )

           ] #tfa.metrics.F1Score(num_classes = 5, average="weighte




)
```

```
Model: "model_1"
__________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
input (InputLayer)              [(None, 224, 224, 3) 0
__________________________________________________________________________________
conv_1 (Conv2D)                 (None, 224, 224, 64) 9472        input[0][0]
__________________________________________________________________________________
batch_normalization_37 (BatchNo (None, 224, 224, 64) 256         conv_1[0][0]
__________________________________________________________________________________
residual_block_16 (ResidualBloc (None, 112, 112, 64) 78784       batch_normalization_37[0][0]
__________________________________________________________________________________
residual_block_17 (ResidualBloc (None, 112, 112, 64) 74368       residual_block_16[0][0]
__________________________________________________________________________________
residual_block_18 (ResidualBloc (None, 112, 112, 64) 74368       residual_block_17[0][0]
__________________________________________________________________________________
residual_block_19 (ResidualBloc (None, 56, 56, 128)  231296      residual_block_18[0][0]
__________________________________________________________________________________
residual_block_20 (ResidualBloc (None, 56, 56, 128)  296192      residual_block_19[0][0]
__________________________________________________________________________________
residual_block_21 (ResidualBloc (None, 56, 56, 128)  296192      residual_block_20[0][0]
```

```
__________________________________________________________________________
_____________________________________
residual_block_22 (ResidualBloc (None, 56, 56, 128)  296192      r
esidual_block_21[0][0]
__________________________________________________________________________
_____________________________________
residual_block_23 (ResidualBloc (None, 28, 28, 256)  921344      r
esidual_block_22[0][0]
__________________________________________________________________________
_____________________________________
residual_block_24 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_23[0][0]
__________________________________________________________________________
_____________________________________
residual_block_25 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_24[0][0]
__________________________________________________________________________
_____________________________________
residual_block_26 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_25[0][0]
__________________________________________________________________________
_____________________________________
residual_block_27 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_26[0][0]
__________________________________________________________________________
_____________________________________
residual_block_28 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_27[0][0]
__________________________________________________________________________
_____________________________________
residual_block_29 (ResidualBloc (None, 14, 14, 512)  3677696     r
esidual_block_28[0][0]
__________________________________________________________________________
_____________________________________
residual_block_30 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_29[0][0]
__________________________________________________________________________
_____________________________________
residual_block_31 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_30[0][0]
__________________________________________________________________________
_____________________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_31[0][0]
__________________________________________________________________________
_____________________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
__________________________________________________________________________
_____________________________________
action_class (Dense)            (None, 5)            2565        f
lat_1[0][0]
__________________________________________________________________________
_____________________________________
action (Dense)                  (None, 21)           10773       f
lat_1[0][0]
==========================================================================
==================================
Total params: 21,327,962
Trainable params: 21,310,810
```

```
Non-trainable params: 17,152
_________________________________________________________________
```

In [ ]:

In [ ]:

```
<IPython.core.display.Javascript object>
```

In [ ]:

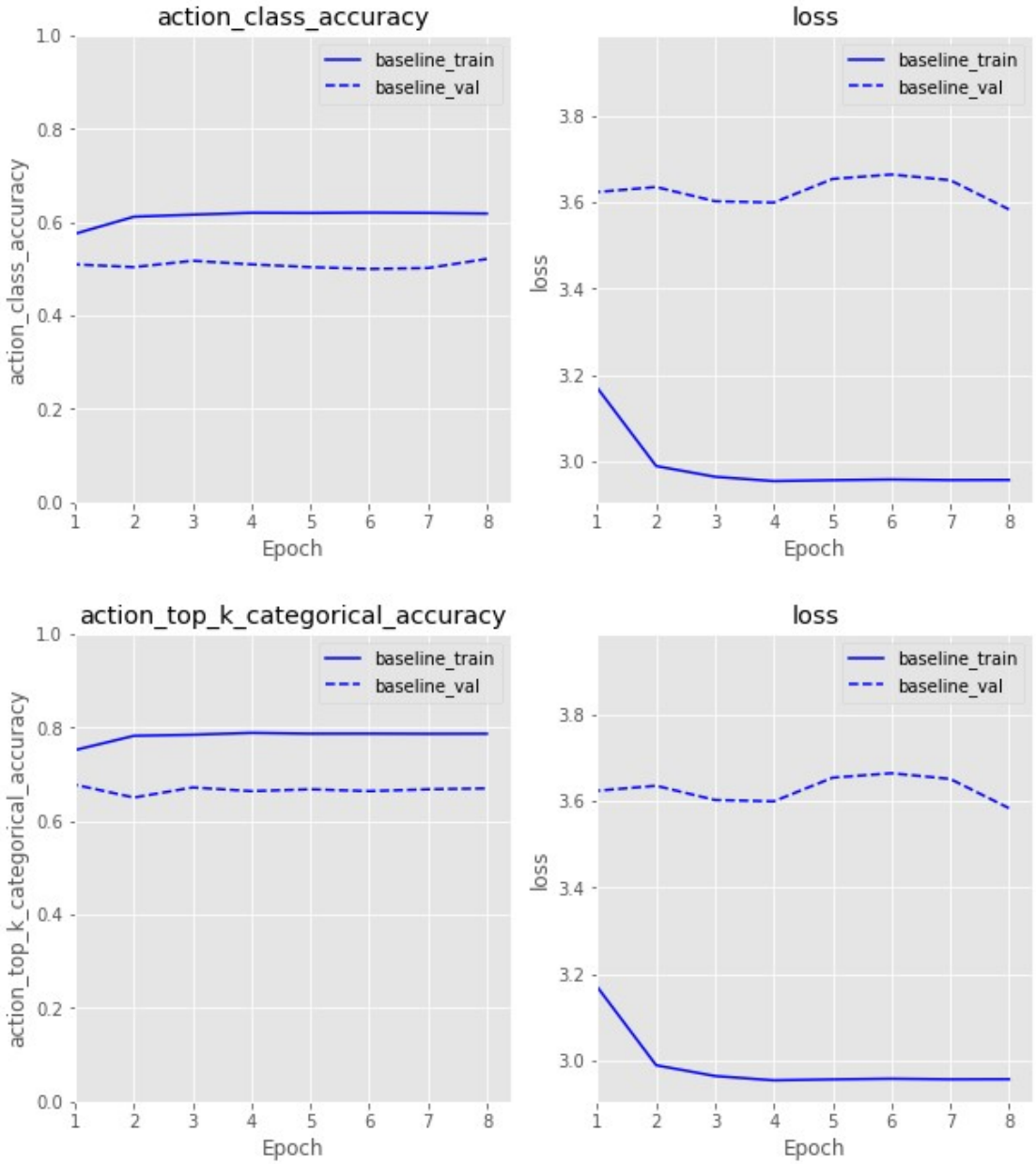
In [ ]:
```
model_name = "baseline"


model_results[model_name] = model.fit(
            training_generator,
            validation_data = validation_generator,
            steps_per_epoch = len(train_data_df)//BATCH_SIZE,
            validation_steps = len(val_data_df)//BATCH_SIZE,
            epochs = 200,
            callbacks = get_callbacks(model_name),
            verbose = True
```

```
Epoch 1/200
  6/852 [..............................] - ETA: 8:12 - loss: 12.25
21 - action_class_loss: 8.1074 - action_loss: 4.1447 - action_clas
s_accuracy: 0.2396 - action_class_top_k_categorical_accuracy: 1.00
00 - action_accuracy: 0.0885 - action_top_k_categorical_accuracy:
0.3047WARNING:tensorflow:Callback method `on_train_batch_end` is s
low compared to the batch time (batch time: 0.1595s vs `on_train_b
atch_end` time: 0.2643s). Check your callbacks.
852/852 [==============================] - 349s 384ms/step - loss:
3.9387 - action_class_loss: 1.3736 - action_loss: 2.5651 - action_
class_accuracy: 0.4430 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.2137 - action_top_k_categorical_accura
cy: 0.5898 - val_loss: 3.7318 - val_action_class_loss: 1.2612 - va
l_action_loss: 2.4705 - val_action_class_accuracy: 0.4961 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.2559 - val_action_top_k_categorical_accuracy: 0.6406

Epoch 00001: val_loss improved from inf to 3.73177, saving model t
o baseline/cp.ckpt
```

In [ ]:
```
# Save model results
model_results[model_name] = model_results[model_name].history

with open("drive/MyDrive/model_results.json", "w") as fp:
```

In [ ]:
```
# Save model weights
```

In [ ]:
```
plotter(model_results, metric = 'action_class_accuracy', ylim_metri
```



✓ Observations:

- Action_class seems to be underfitting
- Action seems to be overfitting
- The model performance plateaus very quickly
- Momentum might be too high which is why it's having difficulty estimating the unknown target function. Decreasing the momentum and ensuring the optimization happens in smaller steps might be useful

In [ ]:

Experimentation - Momentum

Strategy

- First, we do a random search and halve the momentum to 0.4. Then we do a semi-grid search based on the performance evaluations

Halving the momentum to 0.4

In [ ]:
```
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                        data_frame = train_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/trai
                                        )

validation_generator = DataGenerator(
                                        data_frame = val_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/vali
                                        image_column = "FileName_scaled
                                        )
```

In [ ]:
```
filters = [64, 128, 256, 512]
block_size = [3,4,6,3]
reg_lambda = 0.00

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
 initial_learning_rate = 0.1, decay_steps = len(train_data_df)//BAT
  staircase=False)
```

In [ ]:
```

first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")

conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
                kernel_regularizer=tf.keras.regularizers.l2(reg_lam
                padding = "same",
```

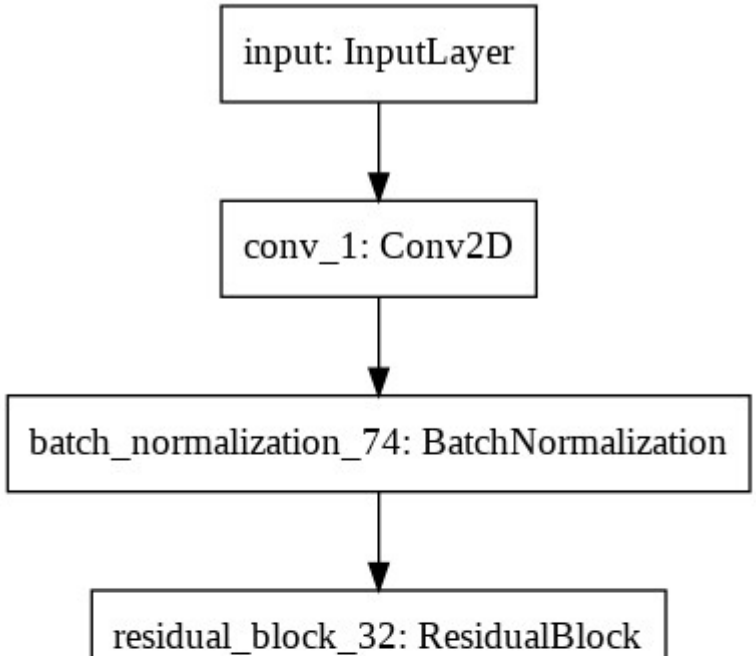
```
                name = "conv_1")(input_)


bn_1 = BatchNormalization(momentum=0.4)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)

for nFilters, nBlocks in zip(filters, block_size):

    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re

    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l2
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(flat_1)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l2(reg
                    kernel_initializer=tf.keras.initializers.Random
                    name =  "action")(flat_1)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
           'accuracy',
           tf.keras.metrics.TopKCategoricalAccuracy(
                k=5,
                name="top_k_categorical_accuracy",
                dtype=None
           )

           ]



)
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input (InputLayer)           [(None, 224, 224, 3)]     0         
_________________________________________________________________
conv_1 (Conv2D)              (None, 224, 224, 64)      9472      
_________________________________________________________________
batch_normalization_74 (Batch (None, 224, 224, 64)     256       
_________________________________________________________________
residual_block_32 (ResidualBl (None, 112, 112, 64)     78784     
_________________________________________________________________
residual_block_33 (ResidualBl (None, 112, 112, 64)     74368     
_________________________________________________________________
residual_block_34 (ResidualBl (None, 112, 112, 64)     74368     
_________________________________________________________________
residual_block_35 (ResidualBl (None, 56, 56, 128)      231296    
_________________________________________________________________
residual_block_36 (ResidualBl (None, 56, 56, 128)      296192    
_________________________________________________________________
residual_block_37 (ResidualBl (None, 56, 56, 128)      296192    
_________________________________________________________________
residual_block_38 (ResidualBl (None, 56, 56, 128)      296192    
_________________________________________________________________
residual_block_39 (ResidualBl (None, 28, 28, 256)      921344    
_________________________________________________________________
residual_block_40 (ResidualBl (None, 28, 28, 256)      1182208   
_________________________________________________________________
residual_block_41 (ResidualBl (None, 28, 28, 256)      1182208   
_________________________________________________________________
residual_block_42 (ResidualBl (None, 28, 28, 256)      1182208   
```

```
esidual_block_41[0][0]
__________________________________________________________________________
_____________________________________
residual_block_43 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_42[0][0]
__________________________________________________________________________
_____________________________________
residual_block_44 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_43[0][0]
__________________________________________________________________________
_____________________________________
residual_block_45 (ResidualBloc (None, 14, 14, 512)  3677696     r
esidual_block_44[0][0]
__________________________________________________________________________
_____________________________________
residual_block_46 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_45[0][0]
__________________________________________________________________________
_____________________________________
residual_block_47 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_46[0][0]
__________________________________________________________________________
_____________________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_47[0][0]
__________________________________________________________________________
_____________________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
__________________________________________________________________________
_____________________________________
action_class (Dense)            (None, 5)            2565        f
lat_1[0][0]
__________________________________________________________________________
_____________________________________
action (Dense)                  (None, 21)           10773       f
lat_1[0][0]
==========================================================================
================================
Total params: 21,327,962
Trainable params: 21,310,810
Non-trainable params: 17,152
__________________________________________________________________________
_____________________________________
```

In [ ]:

Out[43]:



In [ ]:

In [ ]:

```
Reusing TensorBoard on port 6006 (pid 413), started 0:59:46 ago.
(Use '!kill 413' to kill it.)

<IPython.core.display.Javascript object>
```

In [ ]:

```
model_name = "momentum_04"


model_results[model_name] = model.fit(
            training_generator,
            validation_data = validation_generator,
            steps_per_epoch = len(train_data_df)//BATCH_SIZE,
            validation_steps = len(val_data_df)//BATCH_SIZE,
            epochs = 200,
            callbacks = get_callbacks(model_name),
            verbose = True
```

```
Epoch 1/200
  6/852 [..............................] - ETA: 8:19 - loss: 8.170
6 - action_class_loss: 4.0014 - action_loss: 4.1692 - action_class
_accuracy: 0.2604 - action_class_top_k_categorical_accuracy: 1.000
0 - action_accuracy: 0.0391 - action_top_k_categorical_accuracy:
0.2708WARNING:tensorflow:Callback method `on_train_batch_end` is s
low compared to the batch time (batch time: 0.1584s vs `on_train_b
atch_end` time: 0.3354s). Check your callbacks.

WARNING:tensorflow:Callback method `on_train_batch_end` is slow co
mpared to the batch time (batch time: 0.1584s vs `on_train_batch_e
nd` time: 0.3354s). Check your callbacks.
```

```
852/852 [==============================] - 332s 384ms/step - loss:
3.8758 - action_class_loss: 1.3360 - action_loss: 2.5398 - action_
class_accuracy: 0.4518 - action_class_top_k_categorical_accuracy:
```

In [ ]:

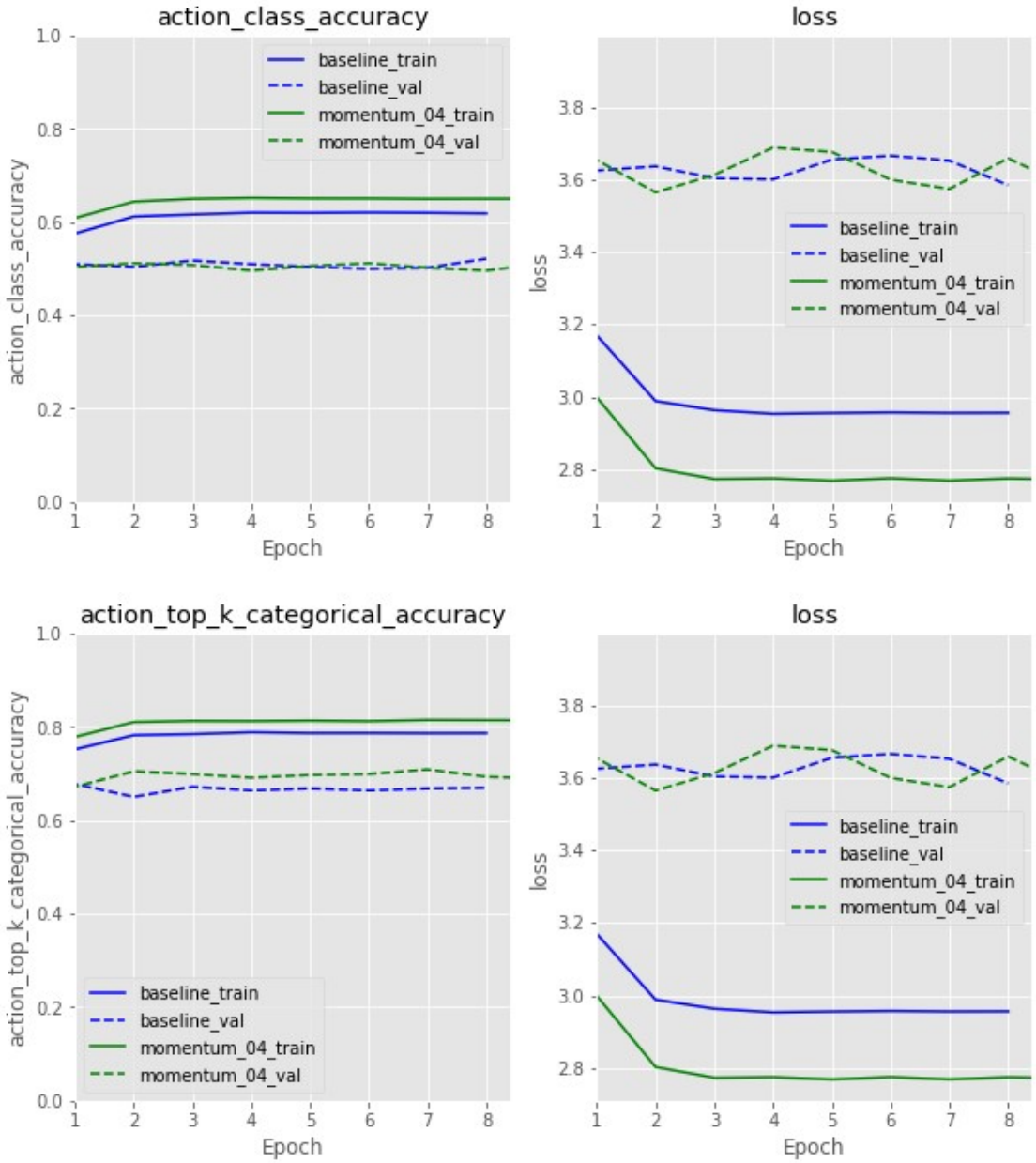
```
# Save model results
model_results[model_name] = model_results[model_name].history

!rm -r "drive/MyDrive/model_results.json"
with open("drive/MyDrive/model_results.json", "w") as fp:
```

In [ ]:

```
# Save model weights
```

```
WARNING:absl:Found untraced functions such as conv2d_72_layer_call
_and_return_conditional_losses, conv2d_72_layer_call_fn, conv2d_73
_layer_call_and_return_conditional_losses, conv2d_73_layer_call_f
n, conv2d_75_layer_call_and_return_conditional_losses while saving
(showing 5 of 160). These functions will not be directly callable
after loading.

INFO:tensorflow:Assets written to: drive/MyDrive/momentum_04_model
/assets

INFO:tensorflow:Assets written to: drive/MyDrive/momentum_04_model
/assets
```

In [ ]:
```
plotter(model_results, metric = 'action_class_accuracy', ylim_metri
```



✓ **Observations:**

- Acion_class is still underfitting in general, and is overfitting when compared to validation set.
- Slight increase in action metric, but it is still overfitting
- Might need to reduce the momentum even further

In [ ]:

In [ ]:

Reducing the momentum further to 0.1

In [ ]:
```
file_ = open('drive/MyDrive/model_results.json')
model_results = json.load(file_)
```

In [ ]:
```
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                        data_frame = train_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/trai
                                        )

validation_generator = DataGenerator(
                                        data_frame = val_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/vali
                                        image_column = "FileName_scaled
                                        )
```

In [ ]:
```
filters = [64, 128, 256, 512]
block_size = [3,4,6,3]
reg_lambda = 0.00

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
 initial_learning_rate = 0.1, decay_steps = len(train_data_df)//BAT
  staircase=False)
```

In [ ]:
```

first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")

conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
                kernel_regularizer=tf.keras.regularizers.l2(reg_lam
                padding = "same",
                name = "conv_1")(input_)


bn_1 = BatchNormalization(momentum=0.1)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)
```

```
for nFilters, nBlocks in zip(filters, block_size):

    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re

    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l2
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(flat_1)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l2(reg
                    kernel_initializer=tf.keras.initializers.Random
                    name =  "action")(flat_1)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
           'accuracy',
           tf.keras.metrics.TopKCategoricalAccuracy(
                k=5,
                name="top_k_categorical_accuracy",
                dtype=None
           )

           ]

)
```

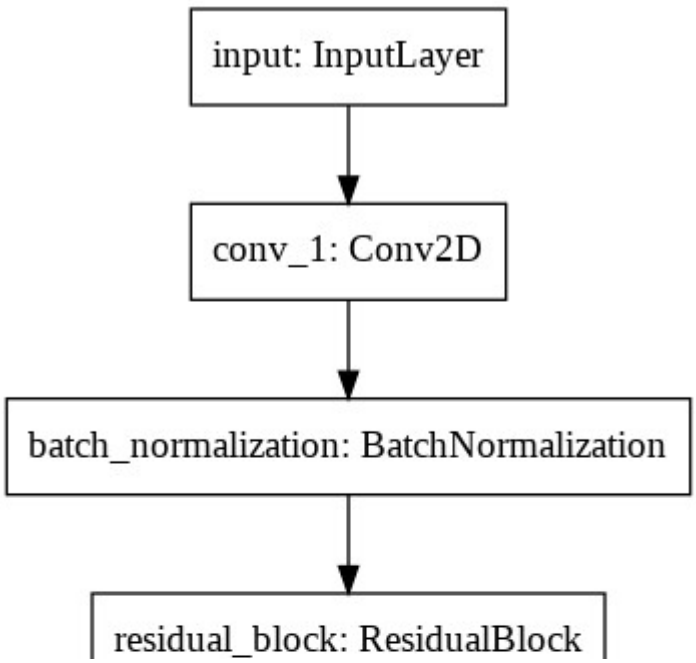
```
Model: "model"
__________________________________________________________________
_________________________________
Layer (type)                 Output Shape              Param #     C
onnected to
==================================================================
=================================
input (InputLayer)           [(None, 224, 224, 3) 0
```

```
__________________________________________________________________________________
_____________________________________
conv_1 (Conv2D)                 (None, 224, 224, 64) 9472        i
nput[0][0]
__________________________________________________________________________________
_____________________________________
batch_normalization (BatchNorma (None, 224, 224, 64) 256         c
onv_1[0][0]
__________________________________________________________________________________
_____________________________________
residual_block (ResidualBlock)  (None, 112, 112, 64) 78784       b
atch_normalization[0][0]
__________________________________________________________________________________
_____________________________________
residual_block_1 (ResidualBlock (None, 112, 112, 64) 74368       r
esidual_block[0][0]
__________________________________________________________________________________
_____________________________________
residual_block_2 (ResidualBlock (None, 112, 112, 64) 74368       r
esidual_block_1[0][0]
__________________________________________________________________________________
_____________________________________
residual_block_3 (ResidualBlock (None, 56, 56, 128)  231296      r
esidual_block_2[0][0]
__________________________________________________________________________________
_____________________________________
residual_block_4 (ResidualBlock (None, 56, 56, 128)  296192      r
esidual_block_3[0][0]
__________________________________________________________________________________
_____________________________________
residual_block_5 (ResidualBlock (None, 56, 56, 128)  296192      r
esidual_block_4[0][0]
__________________________________________________________________________________
_____________________________________
residual_block_6 (ResidualBlock (None, 56, 56, 128)  296192      r
esidual_block_5[0][0]
__________________________________________________________________________________
_____________________________________
residual_block_7 (ResidualBlock (None, 28, 28, 256)  921344      r
esidual_block_6[0][0]
__________________________________________________________________________________
_____________________________________
residual_block_8 (ResidualBlock (None, 28, 28, 256)  1182208     r
esidual_block_7[0][0]
__________________________________________________________________________________
_____________________________________
residual_block_9 (ResidualBlock (None, 28, 28, 256)  1182208     r
esidual_block_8[0][0]
__________________________________________________________________________________
_____________________________________
residual_block_10 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_9[0][0]
__________________________________________________________________________________
_____________________________________
residual_block_11 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_10[0][0]
__________________________________________________________________________________
_____________________________________
residual_block_12 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_11[0][0]
```

```
_________________________________________________________________
residual_block_13 (ResidualBloc (None, 14, 14, 512)  3677696     r
esidual_block_12[0][0]
_________________________________________________________________
residual_block_14 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_13[0][0]
_________________________________________________________________
residual_block_15 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_14[0][0]
_________________________________________________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_15[0][0]
_________________________________________________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
_________________________________________________________________
action_class (Dense)            (None, 5)            2565        f
lat_1[0][0]
_________________________________________________________________
action (Dense)                  (None, 21)           10773       f
lat_1[0][0]
=================================================================
Total params: 21,327,962
Trainable params: 21,310,810
Non-trainable params: 17,152
_________________________________________________________________
```

In [ ]:

Out[34]:



In [ ]:

```
<IPython.core.display.Javascript object>
```

In [ ]:

```
model_name = "momentum_01"


model_results[model_name] = model.fit(
            training_generator,
            validation_data = validation_generator,
            steps_per_epoch = len(train_data_df)//BATCH_SIZE,
            validation_steps = len(val_data_df)//BATCH_SIZE,
            epochs = 200,
            callbacks = get_callbacks(model_name),
            verbose = True
```

```
Epoch 1/200
  6/852 [..............................] - ETA: 7:51 - loss: 12.81
90 - action_class_loss: 7.9169 - action_loss: 4.9021 - action_clas
s_accuracy: 0.2214 - action_class_top_k_categorical_accuracy: 1.00
00 - action_accuracy: 0.0677 - action_top_k_categorical_accuracy:
0.3151WARNING:tensorflow:Callback method `on_train_batch_end` is s
low compared to the batch time (batch time: 0.1481s vs `on_train_b
atch_end` time: 0.2650s). Check your callbacks.
852/852 [==============================] - 346s 381ms/step - loss:
3.9843 - action_class_loss: 1.3938 - action_loss: 2.5905 - action_
class_accuracy: 0.4361 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.2065 - action_top_k_categorical_accura
cy: 0.5865 - val_loss: 3.9198 - val_action_class_loss: 1.3370 - va
l_action_loss: 2.5828 - val_action_class_accuracy: 0.4277 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.2344 - val_action_top_k_categorical_accuracy: 0.6055

Epoch 00001: val_loss improved from inf to 3.91979, saving model t
o momentum_01/cp.ckpt
```

In [ ]:

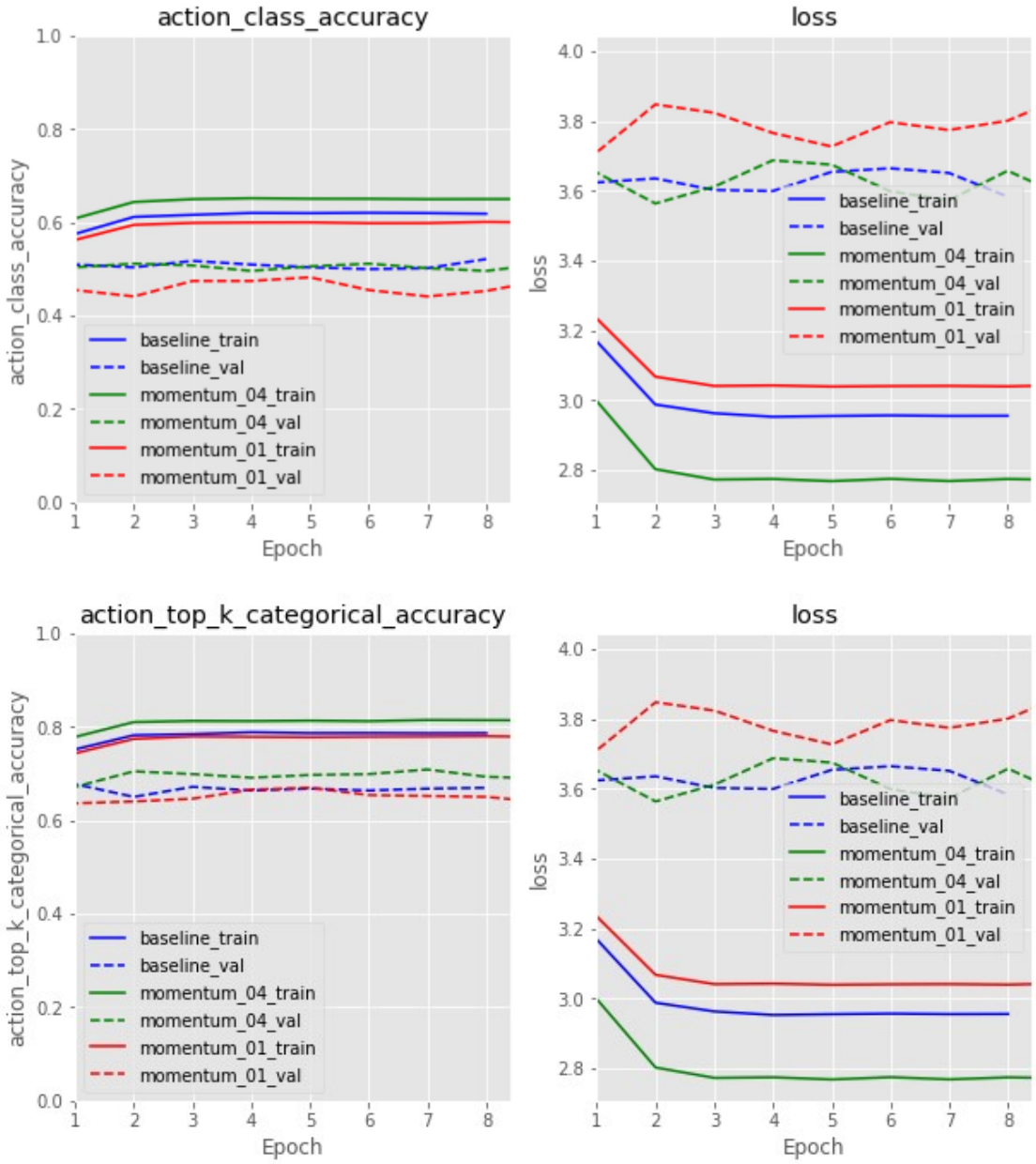
```
# Save model results
model_results[model_name] = model_results[model_name].history

!rm -r "drive/MyDrive/model_results.json"
with open("drive/MyDrive/model_results.json", "w") as fp:
```

In [ ]:

```
# Save model weights
```

```
WARNING:absl:Found untraced functions such as conv2d_layer_call_f
n, conv2d_layer_call_and_return_conditional_losses, conv2d_1_layer
_call_fn, conv2d_1_layer_call_and_return_conditional_losses, conv2
d_3_layer_call_fn while saving (showing 5 of 160). These functions
will not be directly callable after loading.

INFO:tensorflow:Assets written to: drive/MyDrive/momentum_01_model
/assets

INFO:tensorflow:Assets written to: drive/MyDrive/momentum_01_model
/assets
```

In [ ]:
```
plotter(model_results, metric = 'action_class_accuracy', ylim_metri
```



✓ **Observations:**

- Performance overall is worse than the baseline model
- It might not be the case that there is a problem with the optimization step size.
- We might need to increase the overall complexity of the model

In [ ]:

In [ ]:

Experimentation - Increasing Model Complexiy

Strategy

- First we try adding more hidden layers to the MLP

- Then we add more ResNet blocks to the model architecture
- Then we evaluate the performance from both of these and continue to tune

Adding more hidden layers to the MLP

- 4 hidden layers are added to the MLP, with 4096 neurons each. While this may be a lot of additionaly hidden layers to add to an already complex model, it is done in order to test the approach. It is expected from this that the model will perform well on the training data and not on the validation data, overfitting significantly

In [ ]:
```
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                        data_frame = train_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/trai
                                        )

validation_generator = DataGenerator(
                                        data_frame = val_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/vali
                                        image_column = "FileName_scaled
                                        )
```

In [ ]:
```
filters = [64, 128, 256, 512]
block_size = [3,4,6,3]
reg_lambda = 0.00

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
 initial_learning_rate = 0.1, decay_steps = len(train_data_df)//BAT
  staircase=False)
```

In [ ]:
```
first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")
```

```
conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
                kernel_regularizer=tf.keras.regularizers.l2(reg_lam
                padding = "same",
                name = "conv_1")(input_)


bn_1 = BatchNormalization(momentum=0.4)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)

for nFilters, nBlocks in zip(filters, block_size):

    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re

    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)
hidden_1 = Dense(4096, activation = tf.nn.relu, name = "hidden_1")(
hidden_2 = Dense(4096, activation = tf.nn.relu, name = "hidden_2")(
hidden_3 = Dense(4096, activation = tf.nn.relu, name = "hidden_3")(
hidden_4 = Dense(4096, activation = tf.nn.relu, name = "hidden_4")(

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l2
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(hidden_4)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l2(reg
                    kernel_initializer=tf.keras.initializers.Random
                    name =  "action")(hidden_4)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
           'accuracy',
           tf.keras.metrics.TopKCategoricalAccuracy(
                k=5,
```

```
                name="top_k_categorical_accuracy",
                dtype=None
            )

            ]



)
```

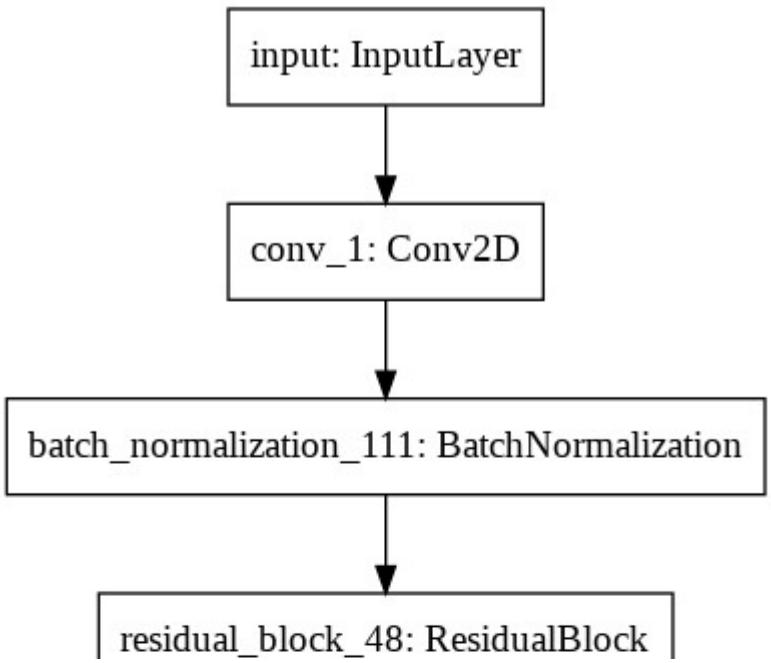
```
Model: "model_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   C
onnected to
==================================================================
input (InputLayer)           [(None, 224, 224, 3)      0
_________________________________________________________________
conv_1 (Conv2D)              (None, 224, 224, 64)      9472      i
nput[0][0]
_________________________________________________________________
batch_normalization_111 (BatchN (None, 224, 224, 64)   256       c
onv_1[0][0]
_________________________________________________________________
residual_block_48 (ResidualBloc (None, 112, 112, 64)   78784     b
atch_normalization_111[0][0]
_________________________________________________________________
residual_block_49 (ResidualBloc (None, 112, 112, 64)   74368     r
esidual_block_48[0][0]
_________________________________________________________________
residual_block_50 (ResidualBloc (None, 112, 112, 64)   74368     r
esidual_block_49[0][0]
_________________________________________________________________
residual_block_51 (ResidualBloc (None, 56, 56, 128)    231296    r
esidual_block_50[0][0]
_________________________________________________________________
residual_block_52 (ResidualBloc (None, 56, 56, 128)    296192    r
esidual_block_51[0][0]
_________________________________________________________________
residual_block_53 (ResidualBloc (None, 56, 56, 128)    296192    r
esidual_block_52[0][0]
_________________________________________________________________
residual_block_54 (ResidualBloc (None, 56, 56, 128)    296192    r
esidual_block_53[0][0]
_________________________________________________________________
residual_block_55 (ResidualBloc (None, 28, 28, 256)    921344    r
esidual_block_54[0][0]
_________________________________________________________________
```

```
_____________________________________
residual_block_56 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_55[0][0]
_____________________________________
_____________________________________
residual_block_57 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_56[0][0]
_____________________________________
_____________________________________
residual_block_58 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_57[0][0]
_____________________________________
_____________________________________
residual_block_59 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_58[0][0]
_____________________________________
_____________________________________
residual_block_60 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_59[0][0]
_____________________________________
_____________________________________
residual_block_61 (ResidualBloc (None, 14, 14, 512)  3677696     r
esidual_block_60[0][0]
_____________________________________
_____________________________________
residual_block_62 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_61[0][0]
_____________________________________
_____________________________________
residual_block_63 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_62[0][0]
_____________________________________
_____________________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_63[0][0]
_____________________________________
_____________________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
_____________________________________
_____________________________________
hidden_1 (Dense)                (None, 4096)         2101248     f
lat_1[0][0]
_____________________________________
_____________________________________
hidden_2 (Dense)                (None, 4096)         16781312    h
idden_1[0][0]
_____________________________________
_____________________________________
hidden_3 (Dense)                (None, 4096)         16781312    h
idden_2[0][0]
_____________________________________
_____________________________________
hidden_4 (Dense)                (None, 4096)         16781312    h
idden_3[0][0]
_____________________________________
_____________________________________
action_class (Dense)            (None, 5)            20485       h
idden_4[0][0]
_____________________________________
```

```
_________________________________
action (Dense)                  (None, 21)           86037       h
idden_4[0][0]
==================================================================
================================
Total params: 73,866,330
Trainable params: 73,849,178
Non-trainable params: 17,152
__________________________________________________________________
```

In [ ]:

Out[52]:



In [ ]:

```
<IPython.core.display.Javascript object>
```

In [ ]:
```
file_ = open('drive/MyDrive/model_results.json')
```

In [ ]:
```
model_name = "hidden_dense"


model_results[model_name] = model.fit(
            training_generator,
            validation_data = validation_generator,
            steps_per_epoch = len(train_data_df)//BATCH_SIZE,
            validation_steps = len(val_data_df)//BATCH_SIZE,
            epochs = 200,
            callbacks = get_callbacks(model_name),
            verbose = True
```

```
Epoch 1/200
  6/852 [..............................] - ETA: 7:40 - loss: 10.62
26 - action_class_loss: 5.8698 - action_loss: 4.7528 - action_clas
s_accuracy: 0.2292 - action_class_top_k_categorical_accuracy: 1.00
00 - action_accuracy: 0.0651 - action_top_k_categorical_accuracy:
0.2474WARNING:tensorflow:Callback method `on_train_batch_end` is s
low compared to the batch time (batch time: 0.1427s vs `on_train_b
atch_end` time: 0.3201s). Check your callbacks.
852/852 [==============================] - 330s 383ms/step - loss:
4.0414 - action_class_loss: 1.3721 - action_loss: 2.6693 - action_
class_accuracy: 0.4243 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.1750 - action_top_k_categorical_accura
cy: 0.5425 - val_loss: 3.7942 - val_action_class_loss: 1.2537 - va
l_action_loss: 2.5405 - val_action_class_accuracy: 0.4883 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.2539 - val_action_top_k_categorical_accuracy: 0.6309

Epoch 00001: val_loss improved from inf to 3.79417, saving model t
o hidden_dense/cp.ckpt
============================== 1 ==============================

action_class_accuracy: 0.42, action_accuracy: 0.18
val_action_class_accuracy: 0.49, val_action_accuracy: 0.25

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.54
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.63

Epoch 2/200
852/852 [==============================] - 325s 381ms/step - loss:
3.3543 - action_class_loss: 1.0980 - action_loss: 2.2563 - action_
class_accuracy: 0.5486 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.2863 - action_top_k_categorical_accura
cy: 0.7050 - val_loss: 3.7631 - val_action_class_loss: 1.2700 - va
l_action_loss: 2.4932 - val_action_class_accuracy: 0.4824 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.2500 - val_action_top_k_categorical_accuracy: 0.6367

Epoch 00002: val_loss improved from 3.79417 to 3.76314, saving mod
el to hidden_dense/cp.ckpt
============================== 2 ==============================

action_class_accuracy: 0.55, action_accuracy: 0.29
val_action_class_accuracy: 0.48, val_action_accuracy: 0.25

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.71
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.64

Epoch 3/200
852/852 [==============================] - 325s 381ms/step - loss:
3.1527 - action_class_loss: 1.0153 - action_loss: 2.1374 - action_
class_accuracy: 0.5896 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.3182 - action_top_k_categorical_accura
cy: 0.7414 - val_loss: 3.8034 - val_action_class_loss: 1.2880 - va
l_action_loss: 2.5154 - val_action_class_accuracy: 0.4824 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.2520 - val_action_top_k_categorical_accuracy: 0.6484
```

```
Epoch 00003: val_loss did not improve from 3.76314
============================== 3 ==============================

action_class_accuracy: 0.59, action_accuracy: 0.32
val_action_class_accuracy: 0.48, val_action_accuracy: 0.25

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.74
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.65

Epoch 4/200
852/852 [==============================] - 325s 381ms/step - loss:
3.1197 - action_class_loss: 1.0010 - action_loss: 2.1186 - action_
class_accuracy: 0.5937 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.3207 - action_top_k_categorical_accura
cy: 0.7483 - val_loss: 3.8821 - val_action_class_loss: 1.3183 - va
l_action_loss: 2.5637 - val_action_class_accuracy: 0.4727 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.2461 - val_action_top_k_categorical_accuracy: 0.6367
Restoring model weights from the end of the best epoch.

Epoch 00004: val_loss did not improve from 3.76314
============================== 4 ==============================
```

In [ ]:
```
# Save model results
model_results[model_name] = model_results[model_name].history

!rm -r "drive/MyDrive/model_results.json"
with open("drive/MyDrive/model_results.json", "w") as fp:
```

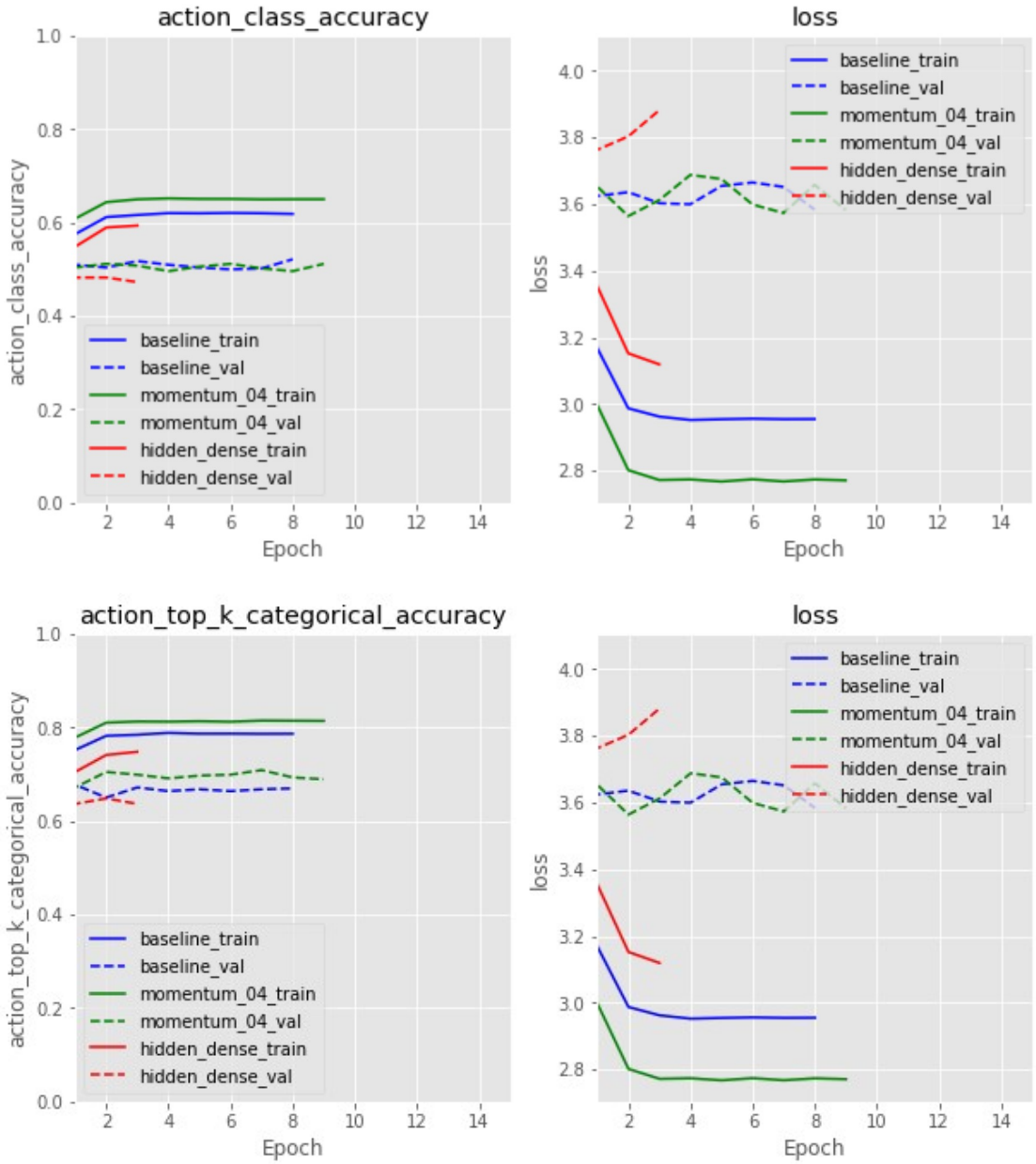
In [ ]:
```
# Save model weights
```

```
WARNING:absl:Found untraced functions such as conv2d_108_layer_cal
l_and_return_conditional_losses, conv2d_108_layer_call_fn, conv2d_
109_layer_call_and_return_conditional_losses, conv2d_109_layer_cal
l_fn, conv2d_111_layer_call_and_return_conditional_losses while sa
ving (showing 5 of 160). These functions will not be directly call
able after loading.

INFO:tensorflow:Assets written to: drive/MyDrive/hidden_dense_mode
l/assets

INFO:tensorflow:Assets written to: drive/MyDrive/hidden_dense_mode
l/assets
```

In [42]:
```
file_ = open('drive/MyDrive/model_results.json')
```

In [43]:
```
names = { 'baseline', 'momentum_04', 'hidden_dense'}
```

In [44]:
```
plotter(model_results_subset, metric = 'action_class_accuracy', yli
```



✓ **Observations:**

- The model surprisingly did not perform any better on the training set, let alone the validation set. Perhaps the complexity of the MLP is just right for this task already.
- We might have more luck adding more ResNet blocks

In [44]:

In [44]:

Adding more ResNet blocks

In [ ]:
```
BATCH_SIZE = 64

data_mean = 0.
```

```
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                        data_frame = train_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/trai
                                        )

validation_generator = DataGenerator(
                                        data_frame = val_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/vali
                                        image_column = "FileName_scaled
                                        )
```

In [ ]:

```
filters = [64, 128, 256, 512]
block_size = [6,6,6,6]
reg_lambda = 0.00

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
 initial_learning_rate = 0.1, decay_steps = len(train_data_df)//BAT
  staircase=False)
```

In [ ]:

```
first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")

conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
                kernel_regularizer=tf.keras.regularizers.l2(reg_lam
                padding = "same",
                name = "conv_1")(input_)


bn_1 = BatchNormalization(momentum=0.4)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)

for nFilters, nBlocks in zip(filters, block_size):

    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
```

```
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re

    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l2
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(flat_1)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l2(reg
                    kernel_initializer=tf.keras.initializers.Random
                    name =  "action")(flat_1)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
           'accuracy',
           tf.keras.metrics.TopKCategoricalAccuracy(
                k=5,
                name="top_k_categorical_accuracy",
                dtype=None
           )

           ]




)
```

```
Model: "model_7"
__________________________________________________________________________
_____________________________
Layer (type)                    Output Shape         Param #     C
onnected to
==========================================================================
==============================
input (InputLayer)              [(None, 224, 224, 3) 0
__________________________________________________________________________
_____________________________
```

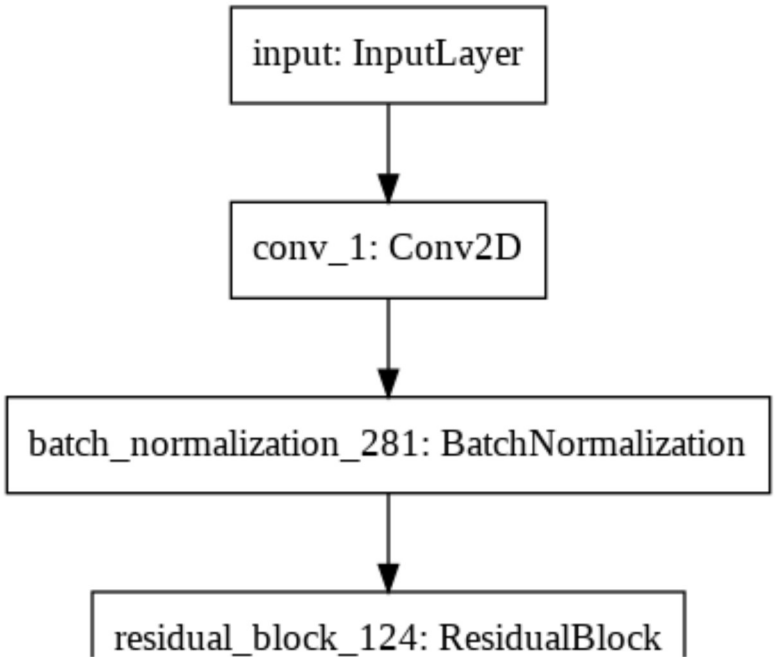
```
conv_1 (Conv2D)                 (None, 224, 224, 64) 9472        i
nput[0][0]
__________________________________________________________________________________
_____
batch_normalization_281 (BatchN (None, 224, 224, 64) 256         c
onv_1[0][0]
__________________________________________________________________________________
_____
residual_block_124 (ResidualBlo (None, 112, 112, 64) 78784       b
atch_normalization_281[0][0]
__________________________________________________________________________________
_____
residual_block_125 (ResidualBlo (None, 112, 112, 64) 74368       r
esidual_block_124[0][0]
__________________________________________________________________________________
_____
residual_block_126 (ResidualBlo (None, 112, 112, 64) 74368       r
esidual_block_125[0][0]
__________________________________________________________________________________
_____
residual_block_127 (ResidualBlo (None, 112, 112, 64) 74368       r
esidual_block_126[0][0]
__________________________________________________________________________________
_____
residual_block_128 (ResidualBlo (None, 112, 112, 64) 74368       r
esidual_block_127[0][0]
__________________________________________________________________________________
_____
residual_block_129 (ResidualBlo (None, 112, 112, 64) 74368       r
esidual_block_128[0][0]
__________________________________________________________________________________
_____
residual_block_130 (ResidualBlo (None, 56, 56, 128)  231296      r
esidual_block_129[0][0]
__________________________________________________________________________________
_____
residual_block_131 (ResidualBlo (None, 56, 56, 128)  296192      r
esidual_block_130[0][0]
__________________________________________________________________________________
_____
residual_block_132 (ResidualBlo (None, 56, 56, 128)  296192      r
esidual_block_131[0][0]
__________________________________________________________________________________
_____
residual_block_133 (ResidualBlo (None, 56, 56, 128)  296192      r
esidual_block_132[0][0]
__________________________________________________________________________________
_____
residual_block_134 (ResidualBlo (None, 56, 56, 128)  296192      r
esidual_block_133[0][0]
__________________________________________________________________________________
_____
residual_block_135 (ResidualBlo (None, 56, 56, 128)  296192      r
esidual_block_134[0][0]
__________________________________________________________________________________
_____
residual_block_136 (ResidualBlo (None, 28, 28, 256)  921344      r
esidual_block_135[0][0]
__________________________________________________________________________________
_____
```

```
residual_block_137 (ResidualBlo (None, 28, 28, 256)  1182208     r
esidual_block_136[0][0]
__________________________________________________________________________
_____________________________
residual_block_138 (ResidualBlo (None, 28, 28, 256)  1182208     r
esidual_block_137[0][0]
__________________________________________________________________________
_____________________________
residual_block_139 (ResidualBlo (None, 28, 28, 256)  1182208     r
esidual_block_138[0][0]
__________________________________________________________________________
_____________________________
residual_block_140 (ResidualBlo (None, 28, 28, 256)  1182208     r
esidual_block_139[0][0]
__________________________________________________________________________
_____________________________
residual_block_141 (ResidualBlo (None, 28, 28, 256)  1182208     r
esidual_block_140[0][0]
__________________________________________________________________________
_____________________________
residual_block_142 (ResidualBlo (None, 14, 14, 512)  3677696     r
esidual_block_141[0][0]
__________________________________________________________________________
_____________________________
residual_block_143 (ResidualBlo (None, 14, 14, 512)  4723712     r
esidual_block_142[0][0]
__________________________________________________________________________
_____________________________
residual_block_144 (ResidualBlo (None, 14, 14, 512)  4723712     r
esidual_block_143[0][0]
__________________________________________________________________________
_____________________________
residual_block_145 (ResidualBlo (None, 14, 14, 512)  4723712     r
esidual_block_144[0][0]
__________________________________________________________________________
_____________________________
residual_block_146 (ResidualBlo (None, 14, 14, 512)  4723712     r
esidual_block_145[0][0]
__________________________________________________________________________
_____________________________
residual_block_147 (ResidualBlo (None, 14, 14, 512)  4723712     r
esidual_block_146[0][0]
__________________________________________________________________________
_____________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_147[0][0]
__________________________________________________________________________
_____________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
__________________________________________________________________________
_____________________________
action_class (Dense)            (None, 5)            2565        f
lat_1[0][0]
__________________________________________________________________________
_____________________________
action (Dense)                  (None, 21)           10773       f
lat_1[0][0]
==========================================================================
==============================
```

```
Total params: 36,314,586
Trainable params: 36,289,498
Non-trainable params: 25,088
_________________________________________________________________
```

In [ ]:

Out[81]:



In [ ]:

In [ ]:

```
Reusing TensorBoard on port 6006 (pid 2517), started 0:03:00 ago.
(Use '!kill 2517' to kill it.)

<IPython.core.display.Javascript object>
```

In [ ]:

```
model_name = "hidden_conv"


model_results[model_name] = model.fit(
            training_generator,
            validation_data = validation_generator,
            steps_per_epoch = len(train_data_df)//BATCH_SIZE,
            validation_steps = len(val_data_df)//BATCH_SIZE,
            epochs = 200,
            callbacks = get_callbacks(model_name),
            verbose = True
```

```
Epoch 1/200
  6/852 [..............................] - ETA: 12:37 - loss: 20.4
095 - action_class_loss: 11.5818 - action_loss: 8.8276 - action_cl
ass_accuracy: 0.2188 - action_class_top_k_categorical_accuracy: 1.
0000 - action_accuracy: 0.0573 - action_top_k_categorical_accurac
y: 0.2656WARNING:tensorflow:Callback method `on_train_batch_end` i
s slow compared to the batch time (batch time: 0.1997s vs `on_trai
n_batch_end` time: 0.5794s). Check your callbacks.
```

```
WARNING:tensorflow:Callback method `on_train_batch_end` is slow co
mpared to the batch time (batch time: 0.1997s vs `on_train_batch_e
nd` time: 0.3794s). Check your callbacks.
852/852 [==============================] - 475s 551ms/step - loss:
4.2870 - action_class_loss: 1.5056 - action_loss: 2.7813 - action_
class_accuracy: 0.3974 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.1687 - action_top_k_categorical_accura
cy: 0.5147 - val_loss: 3.8341 - val_action_class_loss: 1.2881 - va
l_action_loss: 2.5460 - val_action_class_accuracy: 0.4727 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
```

In [ ]:
```
# Save model results
model_results[model_name] = model_results[model_name].history

!rm -r "drive/MyDrive/model_results.json"
with open("drive/MyDrive/model_results.json", "w") as fp:
```

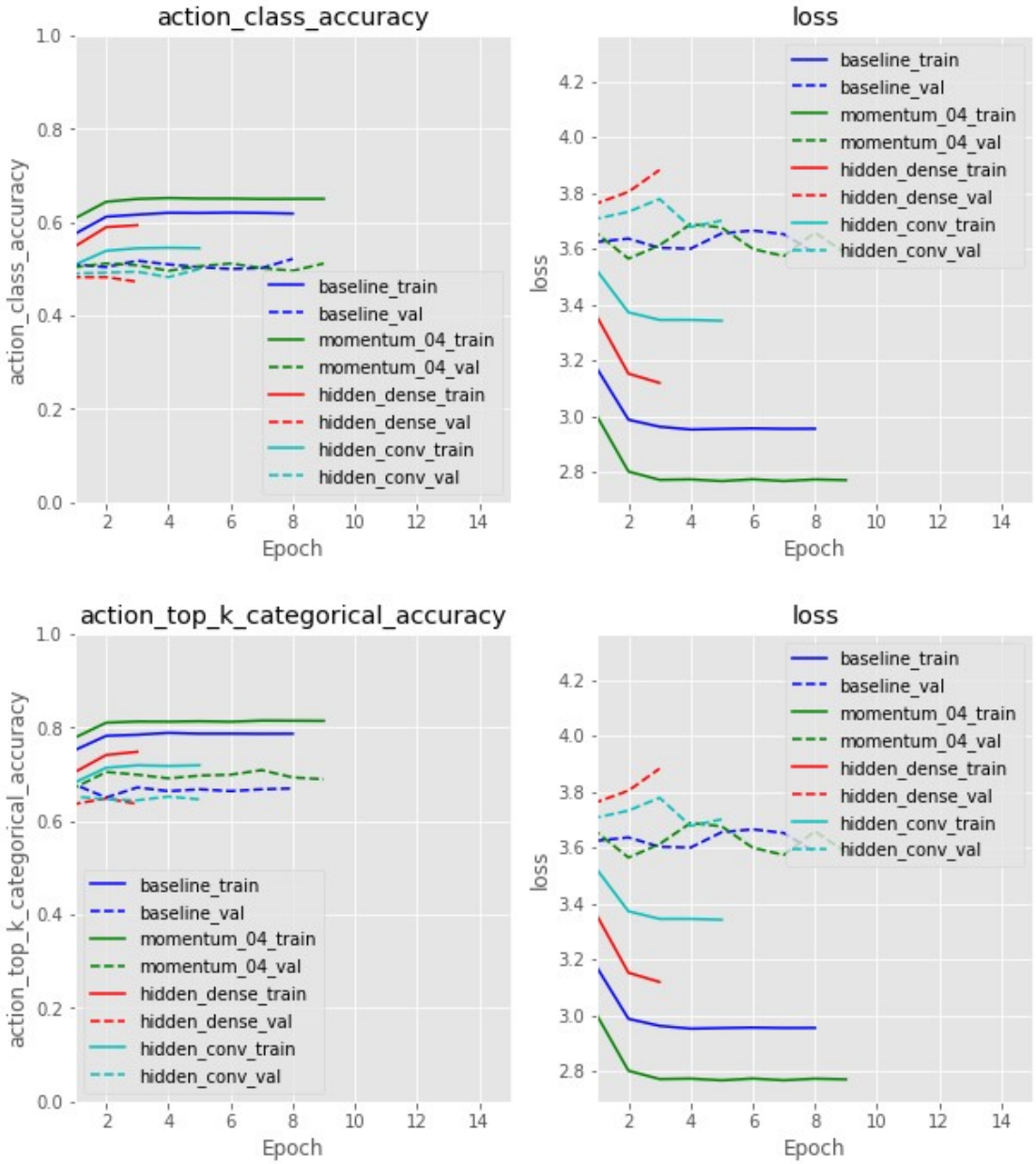
In [ ]:
```
# Save model weights
```

In [ ]:
```
```

In [ ]:
```
file_ = open('drive/MyDrive/model_results.json')
```

In [45]:
```
names = { 'baseline', 'momentum_04', 'hidden_dense', 'hidden_conv'}
```

In [46]:
```
plotter(model_results_subset, metric = 'action_class_accuracy', yli
```



✓ **Observations:**

- Again, there doesn't seem to be any positive impact on the performance. It is actually performance worse now.
- Increasing the model complexity may not be what's needed here to boos performance. It might be worth trying to tune the model to use different optimizers

In [ ]:

In [ ]:

Experimentation - Changing Optimizers

Strategy

- Testing two additional optimizers: Adam and RMSprop. Both are **adaptive** optimizers that don't need a learning rate schedule specified. Adam is also used commonly for image classification

Optimizer - Adam

In [ ]:
```
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                        data_frame = train_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/trai
                                        )

validation_generator = DataGenerator(
                                        data_frame = val_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/vali
                                        image_column = "FileName_scaled
                                        )
```

In [ ]:
```
filters = [64, 128, 256, 512]
block_size = [3,4,6,3]
reg_lambda = 0.00

#lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
# initial_learning_rate = 0.1, decay_steps = len(train_data_df)//BA
#  staircase=False)
```

In [ ]:
```

first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")

conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
                kernel_regularizer=tf.keras.regularizers.l2(reg_lam
```

```
                padding = "same",
                name = "conv_1")(input_)


bn_1 = BatchNormalization(momentum=0.4)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)

for nFilters, nBlocks in zip(filters, block_size):

    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re

    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l2
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(flat_1)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l2(reg
                    kernel_initializer=tf.keras.initializers.Random
                    name =  "action")(flat_1)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
           'accuracy',
           tf.keras.metrics.TopKCategoricalAccuracy(
                k=5,
                name="top_k_categorical_accuracy",
                dtype=None
           )

           ]

)
```

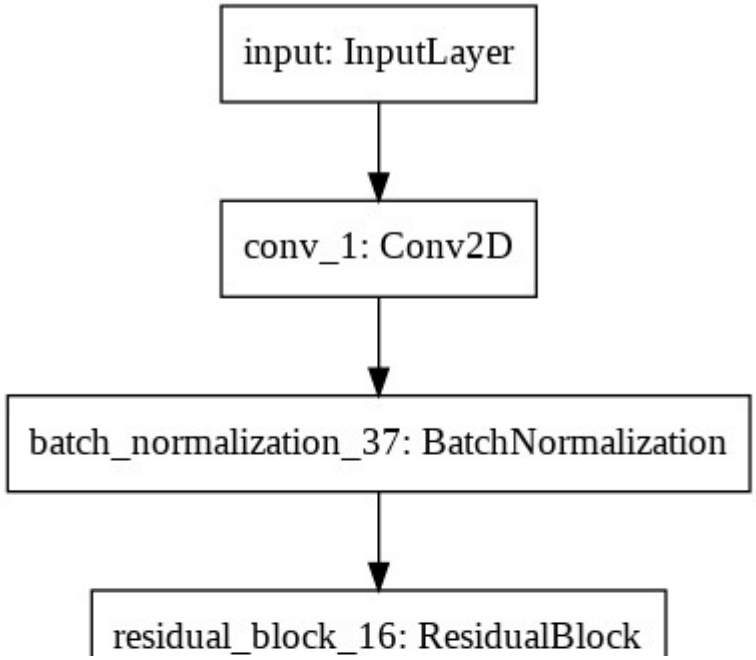
```
model.summary()
Model: "model_1"
```

```
          _
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input (InputLayer)              [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
conv_1 (Conv2D)                 (None, 224, 224, 64) 9472        input[0][0]
__________________________________________________________________________________________________
batch_normalization_37 (BatchNo (None, 224, 224, 64) 256         conv_1[0][0]
__________________________________________________________________________________________________
residual_block_16 (ResidualBloc (None, 112, 112, 64) 78784       batch_normalization_37[0][0]
__________________________________________________________________________________________________
residual_block_17 (ResidualBloc (None, 112, 112, 64) 74368       residual_block_16[0][0]
__________________________________________________________________________________________________
residual_block_18 (ResidualBloc (None, 112, 112, 64) 74368       residual_block_17[0][0]
__________________________________________________________________________________________________
residual_block_19 (ResidualBloc (None, 56, 56, 128)  231296      residual_block_18[0][0]
__________________________________________________________________________________________________
residual_block_20 (ResidualBloc (None, 56, 56, 128)  296192      residual_block_19[0][0]
__________________________________________________________________________________________________
residual_block_21 (ResidualBloc (None, 56, 56, 128)  296192      residual_block_20[0][0]
__________________________________________________________________________________________________
residual_block_22 (ResidualBloc (None, 56, 56, 128)  296192      residual_block_21[0][0]
__________________________________________________________________________________________________
residual_block_23 (ResidualBloc (None, 28, 28, 256)  921344      residual_block_22[0][0]
__________________________________________________________________________________________________
residual_block_24 (ResidualBloc (None, 28, 28, 256)  1182208     residual_block_23[0][0]
__________________________________________________________________________________________________
residual_block_25 (ResidualBloc (None, 28, 28, 256)  1182208     residual_block_24[0][0]
__________________________________________________________________________________________________
residual_block_26 (ResidualBloc (None, 28, 28, 256)  1182208     residual_block_25[0][0]
__________________________________________________________________________________________________
```

```
_____________________________________
residual_block_27 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_26[0][0]
_____________________________________
_____________________________________
residual_block_28 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_27[0][0]
_____________________________________
_____________________________________
residual_block_29 (ResidualBloc (None, 14, 14, 512)  3677696     r
esidual_block_28[0][0]
_____________________________________
_____________________________________
residual_block_30 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_29[0][0]
_____________________________________
_____________________________________
residual_block_31 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_30[0][0]
_____________________________________
_____________________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_31[0][0]
_____________________________________
_____________________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
_____________________________________
_____________________________________
action_class (Dense)            (None, 5)            2565        f
lat_1[0][0]
_____________________________________
_____________________________________
action (Dense)                  (None, 21)           10773       f
lat_1[0][0]
==================================================================
================================
Total params: 21,327,962
Trainable params: 21,310,810
Non-trainable params: 17,152
_____________________________________
_____________________________________
```

In [ ]:

Out[45]:



In [ ]:

In [ ]:

```
Reusing TensorBoard on port 6006 (pid 441), started 1:25:52 ago.
(Use '!kill 441' to kill it.)

<IPython.core.display.Javascript object>
```

In [ ]:

```
model_name = "optimizer_adam_no_lr"


model_results[model_name] = model.fit(
            training_generator,
            validation_data = validation_generator,
            steps_per_epoch = len(train_data_df)//BATCH_SIZE,
            validation_steps = len(val_data_df)//BATCH_SIZE,
            epochs = 200,
            callbacks = get_callbacks(model_name),
            verbose = True
```

```
Epoch 1/200
  6/852 [..............................] - ETA: 12:55 - loss: 5.39
44 - action_class_loss: 2.0331 - action_loss: 3.3613 - action_clas
s_accuracy: 0.2370 - action_class_top_k_categorical_accuracy: 1.00
00 - action_accuracy: 0.0599 - action_top_k_categorical_accuracy:
0.3125WARNING:tensorflow:Callback method `on_train_batch_end` is s
low compared to the batch time (batch time: 0.2252s vs `on_train_b
atch_end` time: 0.5962s). Check your callbacks.

WARNING:tensorflow:Callback method `on_train_batch_end` is slow co
mpared to the batch time (batch time: 0.2252s vs `on_train_batch_e
nd` time: 0.5962s). Check your callbacks.
```

```
852/852 [==============================] - 554s 645ms/step - loss:
3.7163 - action_class_loss: 1.2757 - action_loss: 2.4406 - action_
class_accuracy: 0.4728 - action_class_top_k_categorical_accuracy:
```

In [ ]:
```
# Save model results
model_results[model_name] = model_results[model_name].history

!rm -r "drive/MyDrive/model_results.json"
with open("drive/MyDrive/model_results.json", "w") as fp:
```

In [ ]:
```
# Save model weights
```

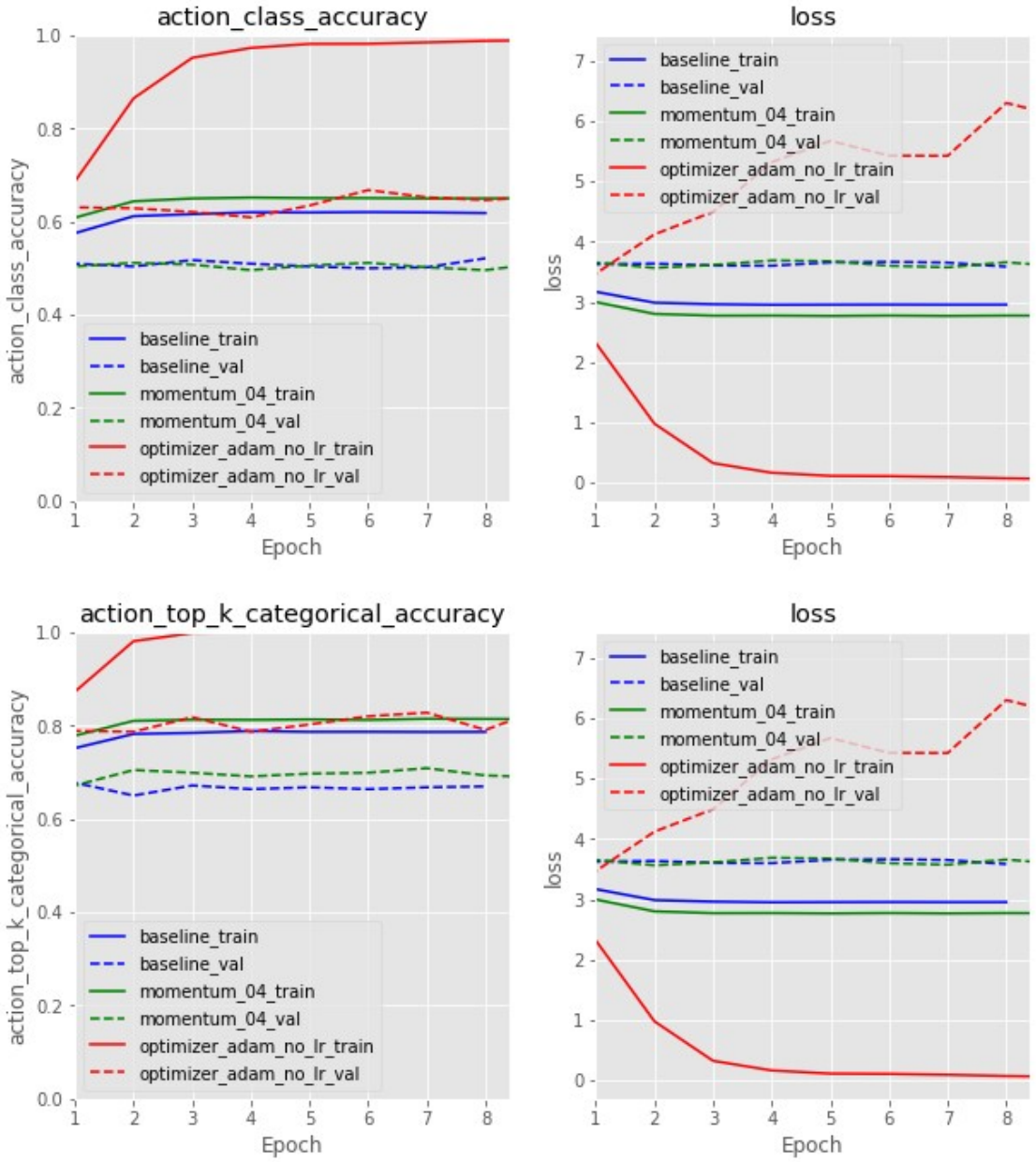
```
WARNING:absl:Found untraced functions such as conv2d_36_layer_call
_and_return_conditional_losses, conv2d_36_layer_call_fn, conv2d_37
_layer_call_and_return_conditional_losses, conv2d_37_layer_call_f
n, conv2d_39_layer_call_and_return_conditional_losses while saving
(showing 5 of 160). These functions will not be directly callable
after loading.

INFO:tensorflow:Assets written to: drive/MyDrive/optimizer_adam_no
_lr_model/assets

INFO:tensorflow:Assets written to: drive/MyDrive/optimizer_adam_no
_lr_model/assets
```

In [ ]:
```
```

Out[52]:
```
dict_keys(['baseline', 'momentum_04', 'momentum_01', 'momentum_02
', 'hidden_dense', 'hidden_conv', 'optimizer_adam', 'optimizer_ada
m_no_lr'])
```

In [ ]:
```
names = { 'baseline', 'momentum_04', 'optimizer_adam_no_lr' }
```

In [ ]:
```
plotter(model_results_subset, metric = 'action_class_accuracy', yli
```



In [ ]:

✓ **Observations:**

- The model seems to be overfitting for both tasks, but more so for action_class than action
- Regularization can potentially help here

Optimizer - RMSProp

In [ ]:
```
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                        data_frame = train_data_df,
```

```
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/trai
                                        )

validation_generator = DataGenerator(
                                        data_frame = val_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/vali
                                        image_column = "FileName_scaled
                                        )
```

In [ ]:

```
filters = [64, 128, 256, 512]
block_size = [3,4,6,3]
reg_lambda = 0.00

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
 initial_learning_rate = 0.1, decay_steps = len(train_data_df)//BAT
  staircase=False)
```

In [ ]:

```
first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")

conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
                kernel_regularizer=tf.keras.regularizers.l2(reg_lam
                padding = "same",
                name = "conv_1")(input_)


bn_1 = BatchNormalization(momentum=0.4)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)

for nFilters, nBlocks in zip(filters, block_size):

    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
```

```
    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l2
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(flat_1)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l2(reg
                    kernel_initializer=tf.keras.initializers.Random
                    name =  "action")(flat_1)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
           'accuracy',
           tf.keras.metrics.TopKCategoricalAccuracy(
                k=5,
                name="top_k_categorical_accuracy",
                dtype=None
           )

           ]

)
```

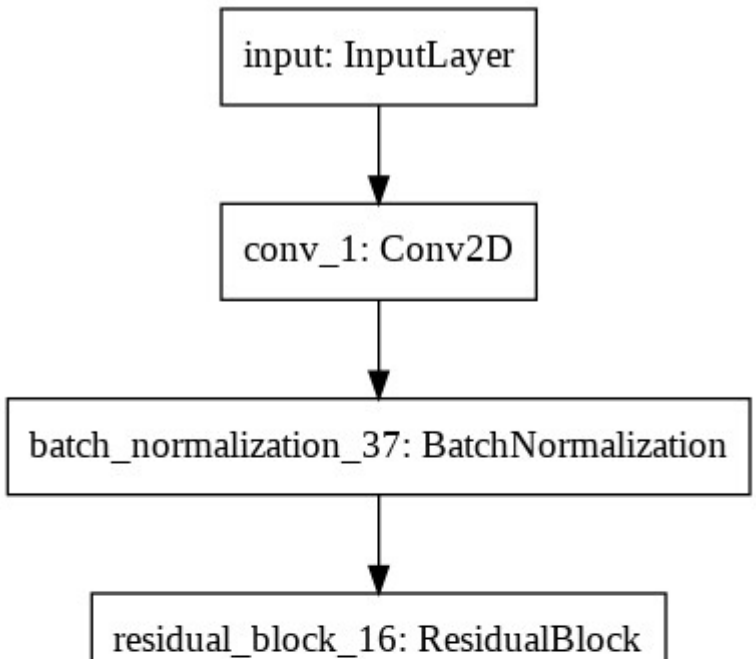
```
Model: "model_1"
__________________________________________________________________
_______________________________
Layer (type)                 Output Shape              Param #     C
onnected to
==================================================================
===============================
input (InputLayer)           [(None, 224, 224, 3) 0
__________________________________________________________________
_______________________________
conv_1 (Conv2D)              (None, 224, 224, 64) 9472        i
nput[0][0]
__________________________________________________________________
_______________________________
batch_normalization_37 (BatchNo (None, 224, 224, 64) 256         c
onv_1[0][0]
```

```
__________________________________________________________________________________________
residual_block_16 (ResidualBloc (None, 112, 112, 64) 78784       batch_normalization_37[0][0]
__________________________________________________________________________________________
residual_block_17 (ResidualBloc (None, 112, 112, 64) 74368       residual_block_16[0][0]
__________________________________________________________________________________________
residual_block_18 (ResidualBloc (None, 112, 112, 64) 74368       residual_block_17[0][0]
__________________________________________________________________________________________
residual_block_19 (ResidualBloc (None, 56, 56, 128)  231296      residual_block_18[0][0]
__________________________________________________________________________________________
residual_block_20 (ResidualBloc (None, 56, 56, 128)  296192      residual_block_19[0][0]
__________________________________________________________________________________________
residual_block_21 (ResidualBloc (None, 56, 56, 128)  296192      residual_block_20[0][0]
__________________________________________________________________________________________
residual_block_22 (ResidualBloc (None, 56, 56, 128)  296192      residual_block_21[0][0]
__________________________________________________________________________________________
residual_block_23 (ResidualBloc (None, 28, 28, 256)  921344      residual_block_22[0][0]
__________________________________________________________________________________________
residual_block_24 (ResidualBloc (None, 28, 28, 256)  1182208     residual_block_23[0][0]
__________________________________________________________________________________________
residual_block_25 (ResidualBloc (None, 28, 28, 256)  1182208     residual_block_24[0][0]
__________________________________________________________________________________________
residual_block_26 (ResidualBloc (None, 28, 28, 256)  1182208     residual_block_25[0][0]
__________________________________________________________________________________________
residual_block_27 (ResidualBloc (None, 28, 28, 256)  1182208     residual_block_26[0][0]
__________________________________________________________________________________________
residual_block_28 (ResidualBloc (None, 28, 28, 256)  1182208     residual_block_27[0][0]
__________________________________________________________________________________________
residual_block_29 (ResidualBloc (None, 14, 14, 512)  3677696     residual_block_28[0][0]
__________________________________________________________________________________________
residual_block_30 (ResidualBloc (None, 14, 14, 512)  4723712     residual_block_29[0][0]
```

```
__________________________________________________________________________________________
residual_block_31 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_30[0][0]
__________________________________________________________________________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_31[0][0]
__________________________________________________________________________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
__________________________________________________________________________________________
action_class (Dense)            (None, 5)            2565        f
lat_1[0][0]
__________________________________________________________________________________________
action (Dense)                  (None, 21)           10773       f
lat_1[0][0]
==========================================================================================
Total params: 21,327,962
Trainable params: 21,310,810
Non-trainable params: 17,152
__________________________________________________________________________________________
```

In [ ]:

Out[42]:



In [ ]:

```
<IPython.core.display.Javascript object>
```

In [ ]:

```
#file_ = open('drive/MyDrive/model_results.json')
```

In [ ]:

```
model_name = "optimizer_rmsprop"


model_results[model_name] = model.fit(
```

```
            training_generator,
            validation_data = validation_generator,
            steps_per_epoch = len(train_data_df)//BATCH_SIZE,
            validation_steps = len(val_data_df)//BATCH_SIZE,
            epochs = 200,
            callbacks = get_callbacks(model_name),
            verbose = True
            `
```

```
Epoch 1/200
  6/852 [..............................] - ETA: 11:59 - loss: 6.33
42 - action_class_loss: 2.5594 - action_loss: 3.7748 - action_clas
s_accuracy: 0.1927 - action_class_top_k_categorical_accuracy: 1.00
00 - action_accuracy: 0.0495 - action_top_k_categorical_accuracy:
0.2552WARNING:tensorflow:Callback method `on_train_batch_end` is s
low compared to the batch time (batch time: 0.3248s vs `on_train_b
atch_end` time: 0.3766s). Check your callbacks.
852/852 [==============================] - 580s 651ms/step - loss:
3.9282 - action_class_loss: 1.3556 - action_loss: 2.5726 - action_
class_accuracy: 0.4317 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.2093 - action_top_k_categorical_accura
cy: 0.5857 - val_loss: 4.1435 - val_action_class_loss: 1.5091 - va
l_action_loss: 2.6344 - val_action_class_accuracy: 0.4434 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.2793 - val_action_top_k_categorical_accuracy: 0.6523

Epoch 00001: val_loss improved from inf to 4.14350, saving model t
o optimizer_rmsprop/cp.ckpt
```

In [ ]:

```
# Save model results
model_results[model_name] = model_results[model_name].history

!rm -r "drive/MyDrive/model_results.json"
with open("drive/MyDrive/model_results.json", "w") as fp:
```

In [ ]:

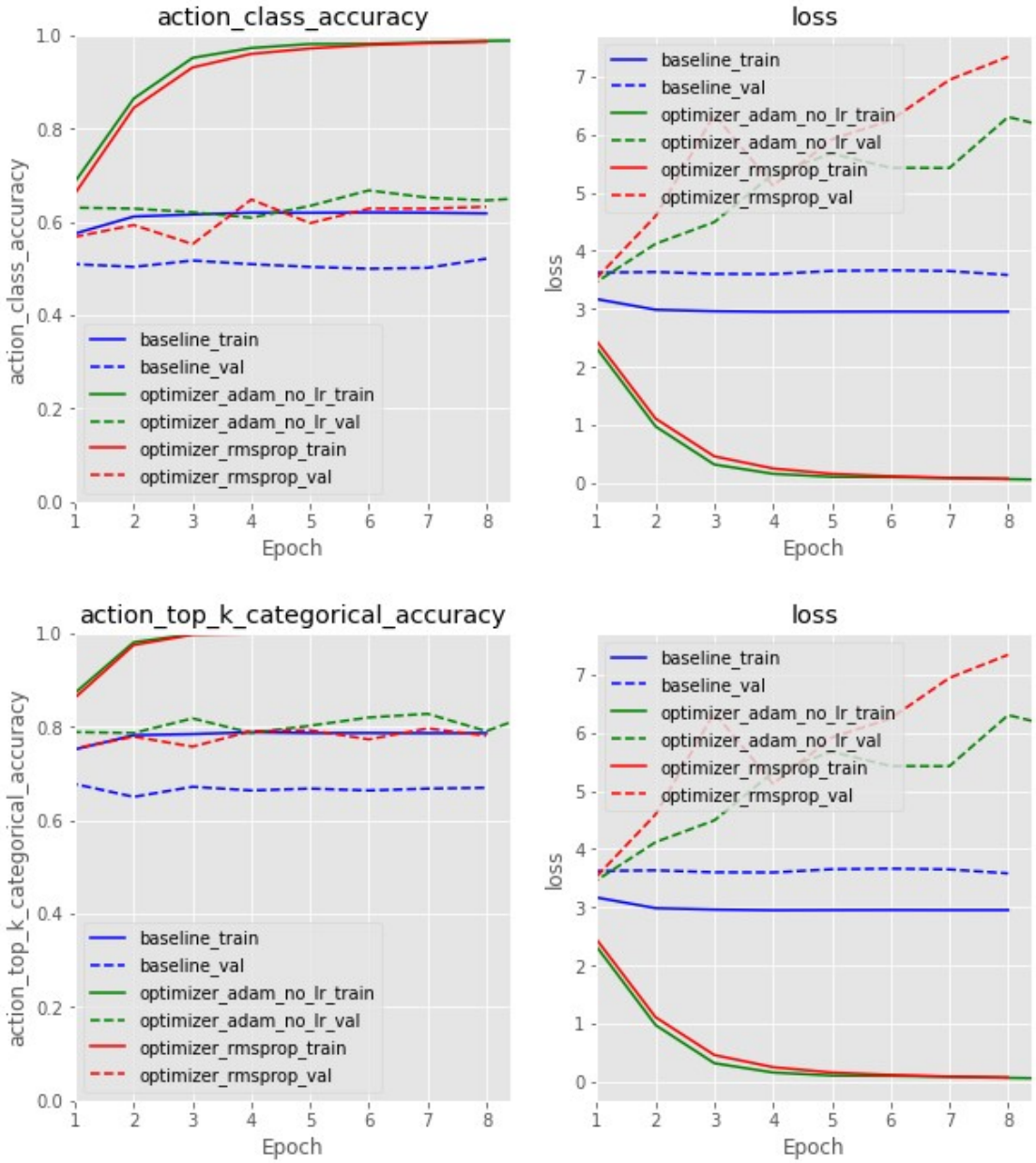
```
# Save model weights
```

```
WARNING:absl:Found untraced functions such as conv2d_36_layer_call
_and_return_conditional_losses, conv2d_36_layer_call_fn, conv2d_37
_layer_call_and_return_conditional_losses, conv2d_37_layer_call_f
n, conv2d_39_layer_call_and_return_conditional_losses while saving
(showing 5 of 160). These functions will not be directly callable
after loading.

INFO:tensorflow:Assets written to: drive/MyDrive/optimizer_rmsprop
_model/assets

INFO:tensorflow:Assets written to: drive/MyDrive/optimizer_rmsprop
_model/assets
```

In [ ]:

```
names = { 'optimizer_adam_no_lr', 'optimizer_rmsprop', 'baseline'
```

In [ ]:
```
plotter(model_results_subset, metric = 'action_class_accuracy', yli
```



In [ ]:
```
""
```

✓ **Observations:**

- Just like Adamn, the model is overfitting
- In terms of relative performance, RMSprop is almost on par with adam, however we will stick to adamn because as mentioned before it is used more commonly for image classification

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Experimentation - Regularization

Strategy

- First we will try to reduce the batch size, since that tends to have a regularizing effect on the model
- Then we will try to use L1 and L2 regularization, and do some tuning to the lambda parameter

Reducing the batch size

In [ ]:
```
BATCH_SIZE = 16

data_mean = 0.
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                        data_frame = train_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/trai
                                        )

validation_generator = DataGenerator(
                                        data_frame = val_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/vali
                                        image_column = "FileName_scaled
                                        )
```

In [ ]:
```
filters = [64, 128, 256, 512]
block_size = [3,4,6,3]
reg_lambda = 0.00


optimizer_ = tf.keras.optimizers.Adam()
```

In [ ]:

```
first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")

conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
                kernel_regularizer=tf.keras.regularizers.l2(reg_lam
                padding = "same",
                name = "conv_1")(input_)


bn_1 = BatchNormalization(momentum=0.4)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)

for nFilters, nBlocks in zip(filters, block_size):

    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re

    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l2
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(flat_1)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l2(reg
                    kernel_initializer=tf.keras.initializers.Randoml
                    name =  "action")(flat_1)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
```

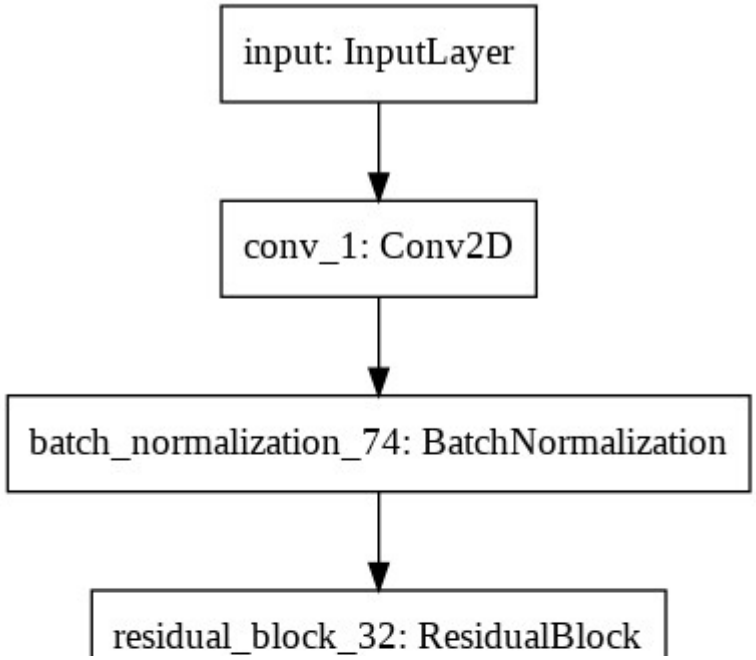
```
            'accuracy',
            tf.keras.metrics.TopKCategoricalAccuracy(
                  k=5,
                  name="top_k_categorical_accuracy",
                  dtype=None
            )

            ]



)
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   C
onnected to
==================================================================
input (InputLayer)           [(None, 224, 224, 3)      0
_________________________________________________________________
conv_1 (Conv2D)              (None, 224, 224, 64)      9472      i
nput[0][0]
_________________________________________________________________
batch_normalization_74 (BatchNo (None, 224, 224, 64)   256       c
onv_1[0][0]
_________________________________________________________________
residual_block_32 (ResidualBloc (None, 112, 112, 64)   78784     b
atch_normalization_74[0][0]
_________________________________________________________________
residual_block_33 (ResidualBloc (None, 112, 112, 64)   74368     r
esidual_block_32[0][0]
_________________________________________________________________
residual_block_34 (ResidualBloc (None, 112, 112, 64)   74368     r
esidual_block_33[0][0]
_________________________________________________________________
residual_block_35 (ResidualBloc (None, 56, 56, 128)    231296    r
esidual_block_34[0][0]
_________________________________________________________________
residual_block_36 (ResidualBloc (None, 56, 56, 128)    296192    r
esidual_block_35[0][0]
_________________________________________________________________
residual_block_37 (ResidualBloc (None, 56, 56, 128)    296192    r
esidual_block_36[0][0]
_________________________________________________________________
residual_block_38 (ResidualBloc (None, 56, 56, 128)    296192    r
esidual_block_37[0][0]
_________________________________________________________________
```

```
residual_block_39 (ResidualBloc (None, 28, 28, 256)  921344      r
esidual_block_38[0][0]
__________________________________________________________________________
_____________________________
residual_block_40 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_39[0][0]
__________________________________________________________________________
_____________________________
residual_block_41 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_40[0][0]
__________________________________________________________________________
_____________________________
residual_block_42 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_41[0][0]
__________________________________________________________________________
_____________________________
residual_block_43 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_42[0][0]
__________________________________________________________________________
_____________________________
residual_block_44 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_43[0][0]
__________________________________________________________________________
_____________________________
residual_block_45 (ResidualBloc (None, 14, 14, 512)  3677696     r
esidual_block_44[0][0]
__________________________________________________________________________
_____________________________
residual_block_46 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_45[0][0]
__________________________________________________________________________
_____________________________
residual_block_47 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_46[0][0]
__________________________________________________________________________
_____________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_47[0][0]
__________________________________________________________________________
_____________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
__________________________________________________________________________
_____________________________
action_class (Dense)            (None, 5)            2565        f
lat_1[0][0]
__________________________________________________________________________
_____________________________
action (Dense)                  (None, 21)           10773       f
lat_1[0][0]
==========================================================================
================================
Total params: 21,327,962
Trainable params: 21,310,810
Non-trainable params: 17,152
__________________________________________________________________________
_____________________________
```

In [ ]:

Out[60]:



In [ ]:

In [ ]:

```
Reusing TensorBoard on port 6006 (pid 441), started 3:24:52 ago.
(Use '!kill 441' to kill it.)

<IPython.core.display.Javascript object>
```

In [ ]:

```
model_name = "reg_batch_size"


model_results[model_name] = model.fit(
            training_generator,
            validation_data = validation_generator,
            steps_per_epoch = len(train_data_df)//BATCH_SIZE,
            validation_steps = len(val_data_df)//BATCH_SIZE,
            epochs = 200,
            callbacks = get_callbacks(model_name),
            verbose = True
```

```
Epoch 1/200
   6/3408 [..............................] - ETA: 21:55 - loss: 6.
3666 - action_class_loss: 2.4174 - action_loss: 3.9492 - action_cl
ass_accuracy: 0.1979 - action_class_top_k_categorical_accuracy: 1.
0000 - action_accuracy: 0.0625 - action_top_k_categorical_accurac
y: 0.2708WARNING:tensorflow:Callback method `on_train_batch_end` i
s slow compared to the batch time (batch time: 0.0733s vs `on_trai
n_batch_end` time: 0.2167s). Check your callbacks.

WARNING:tensorflow:Callback method `on_train_batch_end` is slow co
mpared to the batch time (batch time: 0.0733s vs `on_train_batch_e
nd` time: 0.2167s). Check your callbacks.
```

```
3408/3408 [==============================] - 623s 181ms/step - los
s: 3.9137 - action_class_loss: 1.3422 - action_loss: 2.5715 - acti
on_class_accuracy: 0.4344 - action_class_top_k_categorical_accurac
```

In [ ]:
```
# Save model results
model_results[model_name] = model_results[model_name].history

!rm -r "drive/MyDrive/model_results.json"
with open("drive/MyDrive/model_results.json", "w") as fp:
```

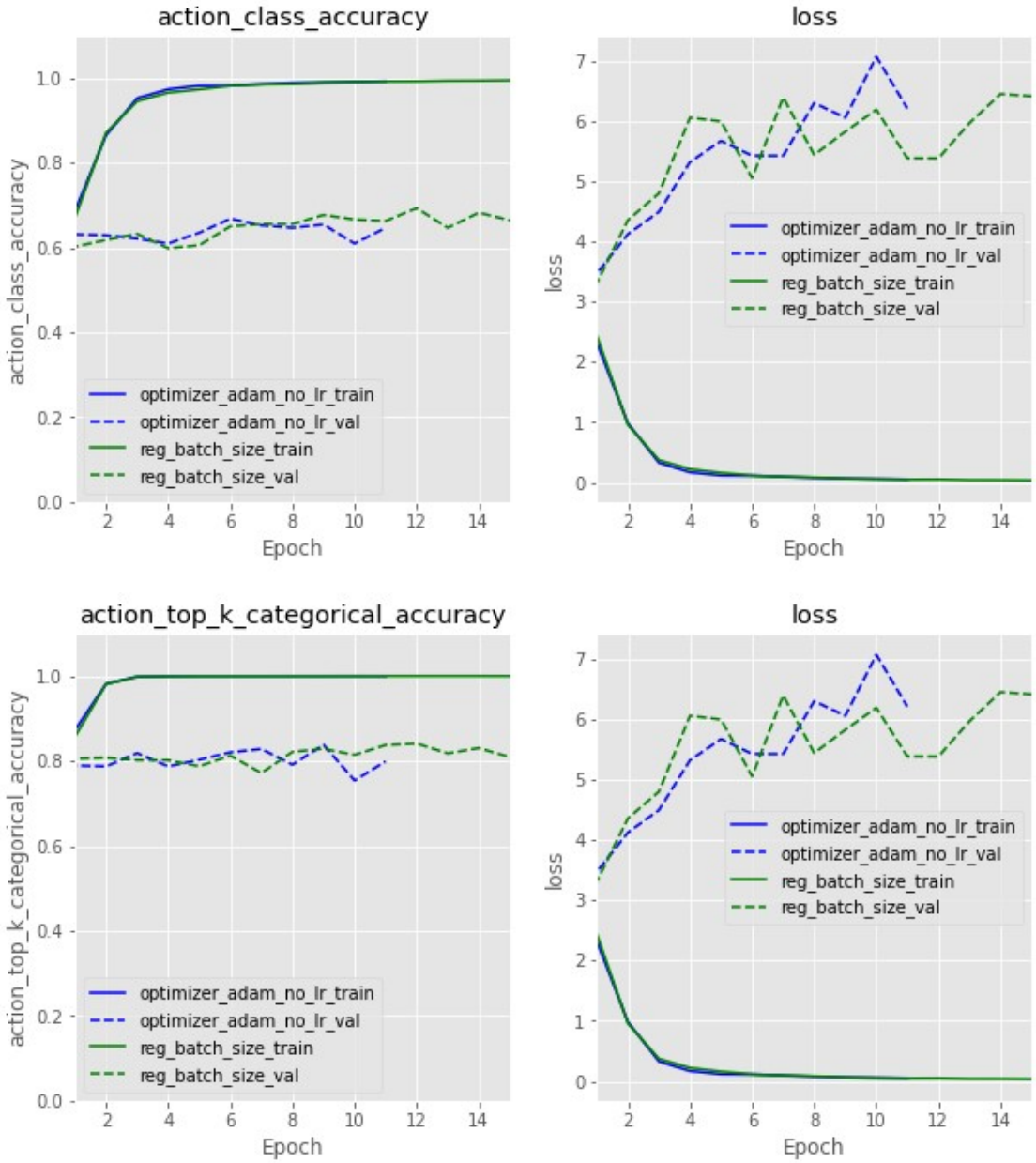
In [ ]:
```
# Save model weights
```

```
WARNING:absl:Found untraced functions such as conv2d_72_layer_call
_and_return_conditional_losses, conv2d_72_layer_call_fn, conv2d_73
_layer_call_and_return_conditional_losses, conv2d_73_layer_call_f
n, conv2d_75_layer_call_and_return_conditional_losses while saving
(showing 5 of 160). These functions will not be directly callable
after loading.

INFO:tensorflow:Assets written to: drive/MyDrive/reg_batch_size_mo
del/assets

INFO:tensorflow:Assets written to: drive/MyDrive/reg_batch_size_mo
del/assets
```

In [ ]:
```
file_ = open('drive/MyDrive/model_results.json')
```

In [47]:
```
names = { 'optimizer_adam_no_lr', 'reg_batch_size' }
```

In [48]:
```
plotter(model_results_subset, metric = 'action_class_accuracy', yli
```



✓ **Observations:**

- The reduction of the batch size 4x from 64 to 16 results in a much longer training time because the number of steps for training increase by a factor of the same amount
- There is no considerable difference in performance

Regularization - L2 with lambda = 0.01

Strategy

- We start with default value of 0.01 from the [documentation] (https://www.tensorflow.org/api_docs/python/tf/keras/regularizers/L2 (https://www.tensorflow.org/api_docs/python/tf/keras/regularizers/L2))
- We then do a semi-grid search based on the performance

In [ ]:
```
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                        data_frame = train_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/trai
                                        )

validation_generator = DataGenerator(
                                        data_frame = val_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/vali
                                        image_column = "FileName_scaled
                                        )
```

In [ ]:
```
filters = [64, 128, 256, 512]
block_size = [3,4,6,3]
reg_lambda = 0.01
```

In [ ]:
```
first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")

conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
                kernel_regularizer=tf.keras.regularizers.l2(reg_lam
                padding = "same",
                name = "conv_1")(input_)


bn_1 = BatchNormalization(momentum=0.4)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)

for nFilters, nBlocks in zip(filters, block_size):
```

```
    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re

    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l2
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(flat_1)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l2(reg
                    kernel_initializer=tf.keras.initializers.Random
                    name =  "action")(flat_1)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
           'accuracy',
           tf.keras.metrics.TopKCategoricalAccuracy(
                k=5,
                name="top_k_categorical_accuracy",
                dtype=None
           )

           ]

)
```

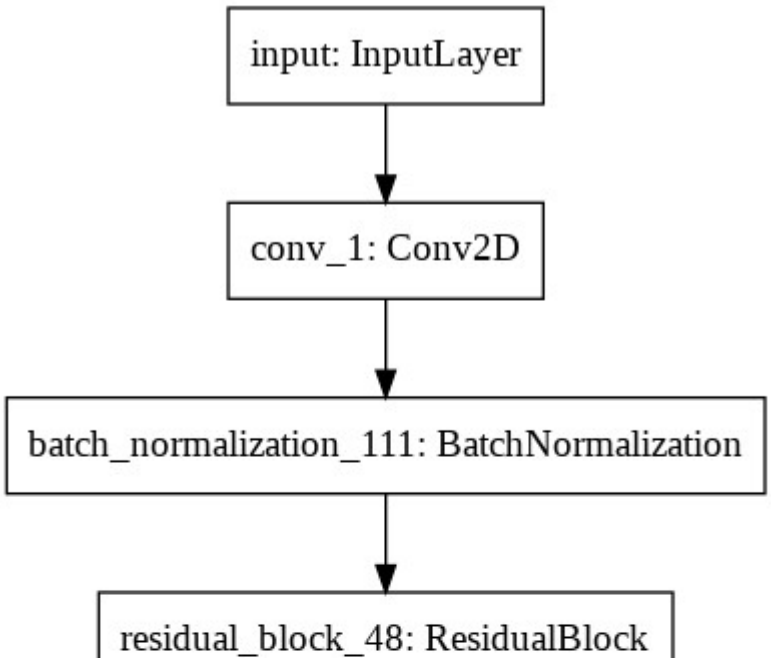
```
Model: "model_3"
__________________________________________________________________________
_____________________________________
Layer (type)                 Output Shape              Param #     C
onnected to
==========================================================================
================================
input (InputLayer)           [(None, 224, 224, 3)  0
__________________________________________________________________________
_____________________________________
```

```
conv_1 (Conv2D)                 (None, 224, 224, 64) 9472        i
nput[0][0]
_____________________________________
_____________________________________
batch_normalization_111 (BatchN (None, 224, 224, 64) 256         c
onv_1[0][0]
_____________________________________
_____________________________________
residual_block_48 (ResidualBloc (None, 112, 112, 64) 78784       b
atch_normalization_111[0][0]
_____________________________________
_____________________________________
residual_block_49 (ResidualBloc (None, 112, 112, 64) 74368       r
esidual_block_48[0][0]
_____________________________________
_____________________________________
residual_block_50 (ResidualBloc (None, 112, 112, 64) 74368       r
esidual_block_49[0][0]
_____________________________________
_____________________________________
residual_block_51 (ResidualBloc (None, 56, 56, 128)  231296      r
esidual_block_50[0][0]
_____________________________________
_____________________________________
residual_block_52 (ResidualBloc (None, 56, 56, 128)  296192      r
esidual_block_51[0][0]
_____________________________________
_____________________________________
residual_block_53 (ResidualBloc (None, 56, 56, 128)  296192      r
esidual_block_52[0][0]
_____________________________________
_____________________________________
residual_block_54 (ResidualBloc (None, 56, 56, 128)  296192      r
esidual_block_53[0][0]
_____________________________________
_____________________________________
residual_block_55 (ResidualBloc (None, 28, 28, 256)  921344      r
esidual_block_54[0][0]
_____________________________________
_____________________________________
residual_block_56 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_55[0][0]
_____________________________________
_____________________________________
residual_block_57 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_56[0][0]
_____________________________________
_____________________________________
residual_block_58 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_57[0][0]
_____________________________________
_____________________________________
residual_block_59 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_58[0][0]
_____________________________________
_____________________________________
residual_block_60 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_59[0][0]
_____________________________________
_____________________________________
```

```
residual_block_61 (ResidualBloc (None, 14, 14, 512)  3677696     r
esidual_block_60[0][0]
__________________________________________________________________________________
_______________________________
residual_block_62 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_61[0][0]
__________________________________________________________________________________
_______________________________
residual_block_63 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_62[0][0]
__________________________________________________________________________________
_______________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_63[0][0]
__________________________________________________________________________________
_______________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
__________________________________________________________________________________
_______________________________
action_class (Dense)            (None, 5)            2565        f
lat_1[0][0]
__________________________________________________________________________________
_______________________________
action (Dense)                  (None, 21)           10773       f
lat_1[0][0]
==================================================================
================================
Total params: 21,327,962
Trainable params: 21,310,810
Non-trainable params: 17,152
__________________________________________________________________________________
```

In [ ]:

Out[58]:



In [ ]:

```
rm: cannot remove 'logs': No such file or directory
```

In [ ]:

```
Reusing TensorBoard on port 6006 (pid 619), started 1:43:17 ago.
(Use '!kill 619' to kill it.)

<IPython.core.display.Javascript object>
```

In [ ]:

```
model_name = "reg_labmda_0p01"


model_results[model_name] = model.fit(
            training_generator,
            validation_data = validation_generator,
            steps_per_epoch = len(train_data_df)//BATCH_SIZE,
            validation_steps = len(val_data_df)//BATCH_SIZE,
            epochs = 200,
            callbacks = get_callbacks(model_name),
            verbose = True
```

```
Epoch 1/200
  6/852 [..............................] - ETA: 12:16 - loss: 504.
7764 - action_class_loss: 1.9089 - action_loss: 3.4557 - action_cl
ass_accuracy: 0.1849 - action_class_top_k_categorical_accuracy: 1.
0000 - action_accuracy: 0.0495 - action_top_k_categorical_accurac
y: 0.2943WARNING:tensorflow:Callback method `on_train_batch_end` i
s slow compared to the batch time (batch time: 0.2301s vs `on_trai
n_batch_end` time: 0.5500s). Check your callbacks.

WARNING:tensorflow:Callback method `on_train_batch_end` is slow co
mpared to the batch time (batch time: 0.2301s vs `on_train_batch_e
nd` time: 0.5500s). Check your callbacks.

852/852 [==============================] - 554s 646ms/step - loss:
29.3000 - action_class_loss: 1.4761 - action_loss: 2.8556 - action
_class_accuracy: 0.3638 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.1309 - action_top_k_categorical_accura
cy: 0.4613 - val_loss: 5.3385 - val_action_class_loss: 1.6575 - va
l_action_loss: 2.9781 - val_action_class_accuracy: 0.3027 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
```

In [ ]:

```
# Save model results
model_results[model_name] = model_results[model_name].history

!rm -r "drive/MyDrive/model_results.json"
with open("drive/MyDrive/model_results.json", "w") as fp:
```

In [ ]:

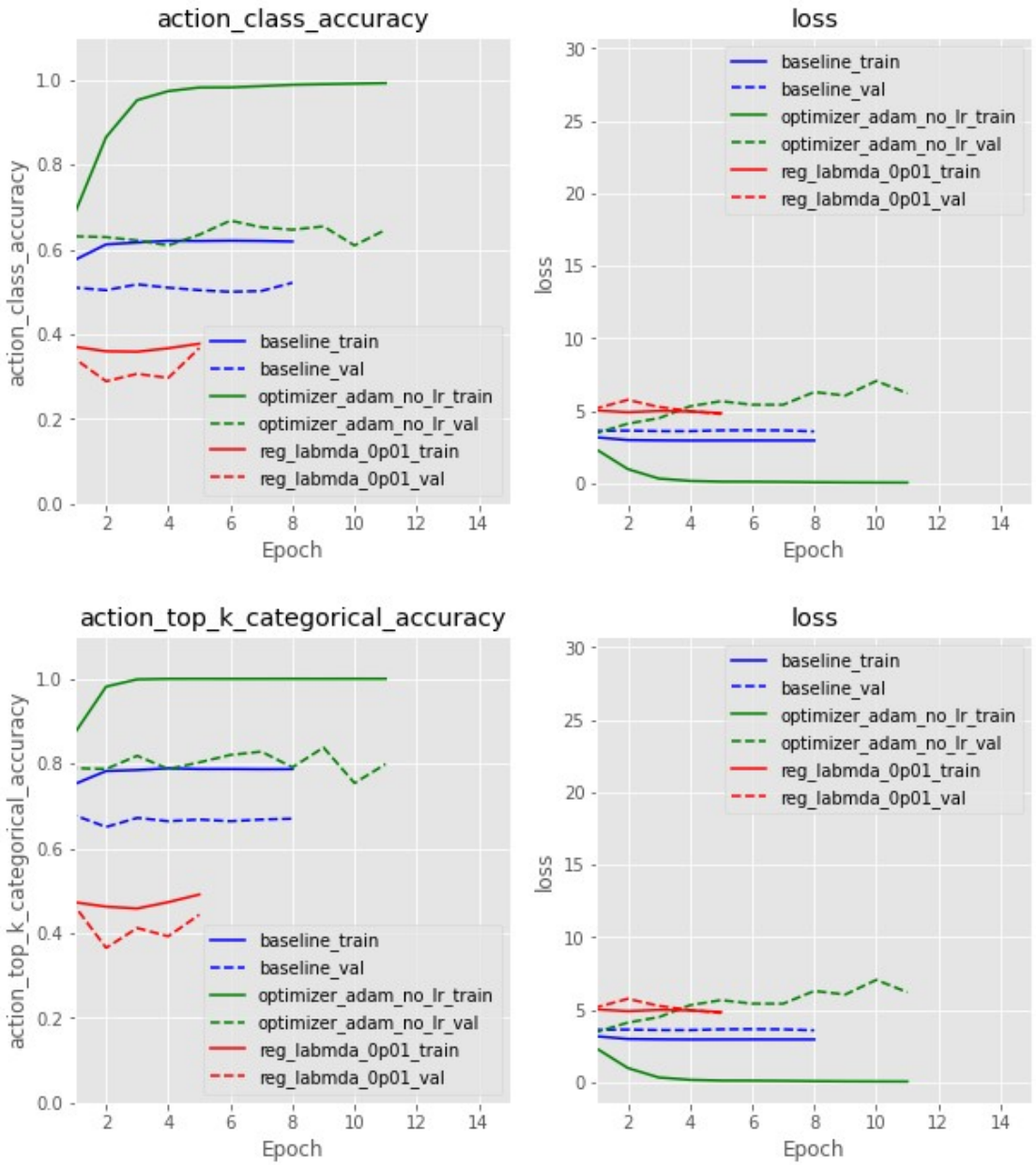
```
# Save model weights
```

```
WARNING:absl:Found untraced functions such as conv2d_108_layer_cal
l_and_return_conditional_losses, conv2d_108_layer_call_fn, conv2d_
109_layer_call_and_return_conditional_losses, conv2d_109_layer_cal
l_fn, conv2d_111_layer_call_and_return_conditional_losses while sa
ving (showing 5 of 160). These functions will not be directly call
able after loading.

INFO:tensorflow:Assets written to: drive/MyDrive/reg_labmda_0p01_m
odel/assets
```

```
INFO:tensorflow:Assets written to: drive/MyDrive/reg_labmda_0p01_m
```

In [49]:
```
names = { 'baseline', 'optimizer_adam_no_lr', 'reg_labmda_0p01' }
```

In [50]:
```
plotter(model_results_subset, metric = 'action_class_accuracy', yli
```



✓ **Observations:**

- Performance is considerably worse, even more than baseline.
- This means that we need to decrease the lambda parameter further to minimize the regularizing impact

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Regularization - L2 with lambda = 0.001

- The lambda value is decreased ten-fold to assess if that increases performance

In [ ]:
```
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                        data_frame = train_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/trai
                                        )

validation_generator = DataGenerator(
                                        data_frame = val_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/vali
                                        image_column = "FileName_scaled
                                        )
```

In [ ]:
```
filters = [64, 128, 256, 512]
block_size = [3,4,6,3]
reg_lambda = 0.001
```

In [ ]:
```
first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")

conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
```

```
                kernel_regularizer=tf.keras.regularizers.l2(reg_lam
                padding = "same",
                name = "conv_1")(input_)


bn_1 = BatchNormalization(momentum=0.4)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)

for nFilters, nBlocks in zip(filters, block_size):

    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re

    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l2
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(flat_1)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l2(reg
                    kernel_initializer=tf.keras.initializers.Random
                    name =  "action")(flat_1)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
           'accuracy',
           tf.keras.metrics.TopKCategoricalAccuracy(
                k=5,
                name="top_k_categorical_accuracy",
                dtype=None
           )

           ]

)
```

```
Model: "model_4"
```

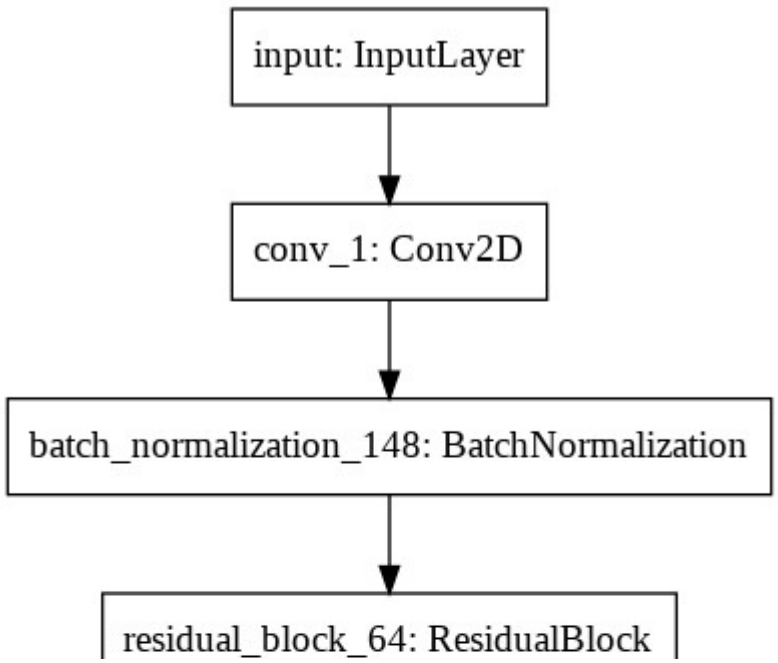
Model: "model_4"

Layer (type)	Output Shape	Param #	Connected to
input (InputLayer)	[(None, 224, 224, 3)	0	
conv_1 (Conv2D)	(None, 224, 224, 64)	9472	input[0][0]
batch_normalization_148 (BatchN	(None, 224, 224, 64)	256	conv_1[0][0]
residual_block_64 (ResidualBloc	(None, 112, 112, 64)	78784	batch_normalization_148[0][0]
residual_block_65 (ResidualBloc	(None, 112, 112, 64)	74368	residual_block_64[0][0]
residual_block_66 (ResidualBloc	(None, 112, 112, 64)	74368	residual_block_65[0][0]
residual_block_67 (ResidualBloc	(None, 56, 56, 128)	231296	residual_block_66[0][0]
residual_block_68 (ResidualBloc	(None, 56, 56, 128)	296192	residual_block_67[0][0]
residual_block_69 (ResidualBloc	(None, 56, 56, 128)	296192	residual_block_68[0][0]
residual_block_70 (ResidualBloc	(None, 56, 56, 128)	296192	residual_block_69[0][0]
residual_block_71 (ResidualBloc	(None, 28, 28, 256)	921344	residual_block_70[0][0]
residual_block_72 (ResidualBloc	(None, 28, 28, 256)	1182208	residual_block_71[0][0]
residual_block_73 (ResidualBloc	(None, 28, 28, 256)	1182208	residual_block_72[0][0]
residual_block_74 (ResidualBloc	(None, 28, 28, 256)	1182208	residual_block_73[0][0]

```
_________________________________
residual_block_75 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_74[0][0]
__________________________________________________________________________
_________________________________
residual_block_76 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_75[0][0]
__________________________________________________________________________
_________________________________
residual_block_77 (ResidualBloc (None, 14, 14, 512)  3677696     r
esidual_block_76[0][0]
__________________________________________________________________________
_________________________________
residual_block_78 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_77[0][0]
__________________________________________________________________________
_________________________________
residual_block_79 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_78[0][0]
__________________________________________________________________________
_________________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_79[0][0]
__________________________________________________________________________
_________________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
__________________________________________________________________________
_________________________________
action_class (Dense)            (None, 5)            2565        f
lat_1[0][0]
__________________________________________________________________________
_________________________________
action (Dense)                  (None, 21)           10773       f
lat_1[0][0]
==========================================================================
================================
Total params: 21,327,962
Trainable params: 21,310,810
Non-trainable params: 17,152
__________________________________________________________________________
_________________________________
```

In [ ]:

Out[70]:



In [ ]:

In [ ]:

```
<IPython.core.display.Javascript object>
```

In [ ]:

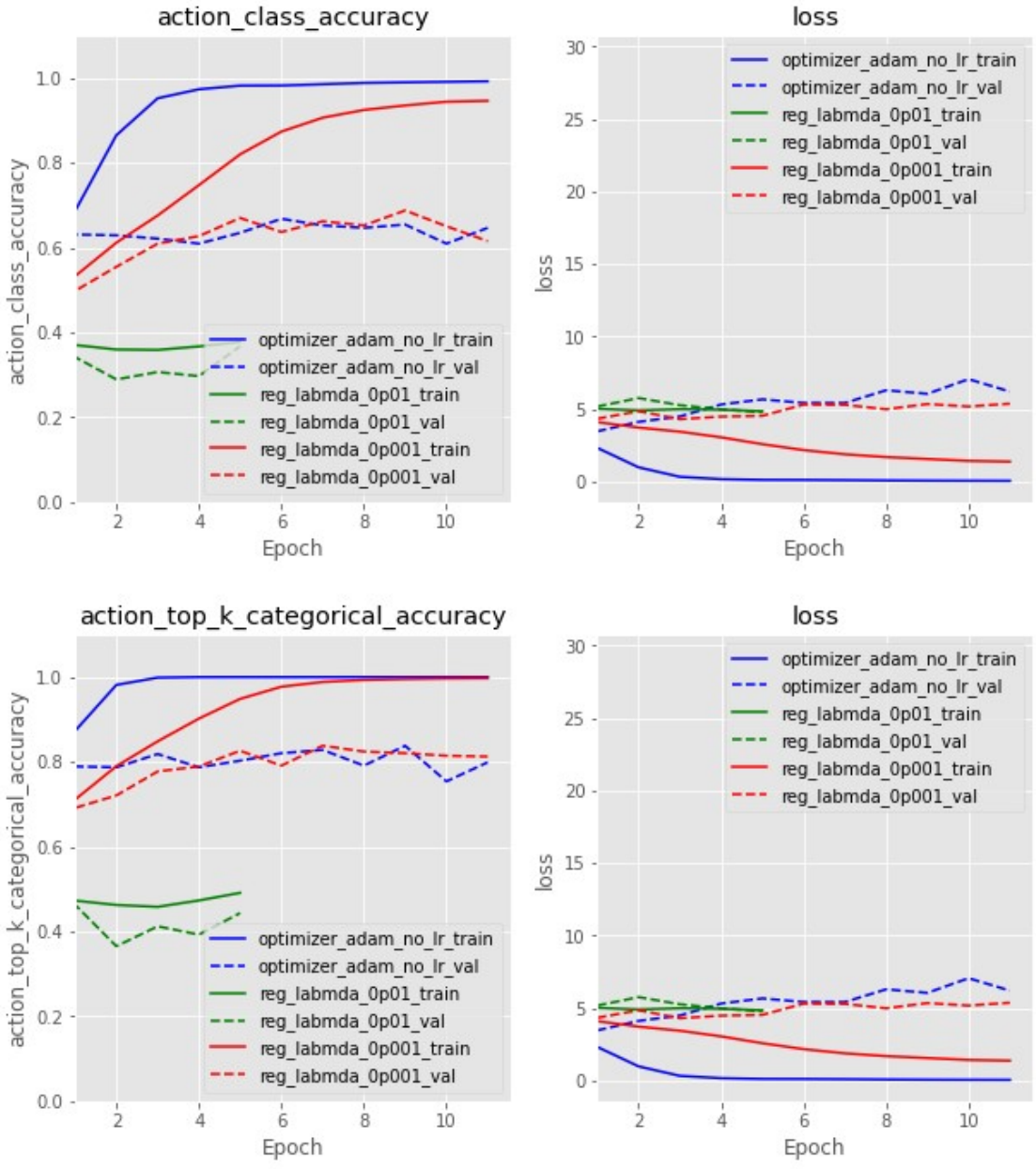
```
model_name = "reg_labmda_0p001"


model_results[model_name] = model.fit(
            training_generator,
            validation_data = validation_generator,
            steps_per_epoch = len(train_data_df)//BATCH_SIZE,
            validation_steps = len(val_data_df)//BATCH_SIZE,
            epochs = 200,
            callbacks = get_callbacks(model_name),
            verbose = True
```

```
Epoch 1/200
  6/852 [..............................] - ETA: 12:21 - loss: 58.0
462 - action_class_loss: 2.0803 - action_loss: 3.6221 - action_cla
ss_accuracy: 0.2370 - action_class_top_k_categorical_accuracy: 1.0
000 - action_accuracy: 0.0521 - action_top_k_categorical_accuracy:
0.2526WARNING:tensorflow:Callback method `on_train_batch_end` is s
low compared to the batch time (batch time: 0.2306s vs `on_train_b
atch_end` time: 0.5522s). Check your callbacks.

WARNING:tensorflow:Callback method `on_train_batch_end` is slow co
mpared to the batch time (batch time: 0.2306s vs `on_train_batch_e
nd` time: 0.5522s). Check your callbacks.
```

In [ ]:
```
file_ = open('drive/MyDrive/model_results.json')
```

In [51]:
```
names = { 'optimizer_adam_no_lr', 'reg_labmda_0p01', 'reg_labmda_0p
```

In [ ]:
```
plotter(model_results_subset, metric = 'action_class_accuracy', yli
```



In [ ]:

✓ **Observations:**

- While the model is overftitting, and performance evaluation is still around the same value at the end of the epochs, it is not overfitting as quickly, as can be seen from the plots. We can try to increase the lambda value from here

In [ ]:

Regularization - L2 with lambda = 0.005

- We increase the value of lambda to approximately halfway between 0.01 and 0.001

In [ ]:
```
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                        data_frame = train_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/trai
                                        )

validation_generator = DataGenerator(
                                        data_frame = val_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/vali
                                        image_column = "FileName_scaled
                                        )
```

In [ ]:
```
filters = [64, 128, 256, 512]
block_size = [3,4,6,3]
reg_lambda = 0.005

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
 initial_learning_rate = 0.1, decay_steps = len(train_data_df)//BAT
  staircase=False)
```

In [ ]:
```

first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")

conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
                kernel_regularizer=tf.keras.regularizers.l2(reg_lam
                padding = "same",
```

```
                name = "conv_1")(input_)


bn_1 = BatchNormalization(momentum=0.4)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)

for nFilters, nBlocks in zip(filters, block_size):

    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re

    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l2
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(flat_1)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l2(reg
                    kernel_initializer=tf.keras.initializers.Random
                    name =  "action")(flat_1)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
           'accuracy',
           tf.keras.metrics.TopKCategoricalAccuracy(
                k=5,
                name="top_k_categorical_accuracy",
                dtype=None
           )

           ]

)
```

```
model.summary()
```

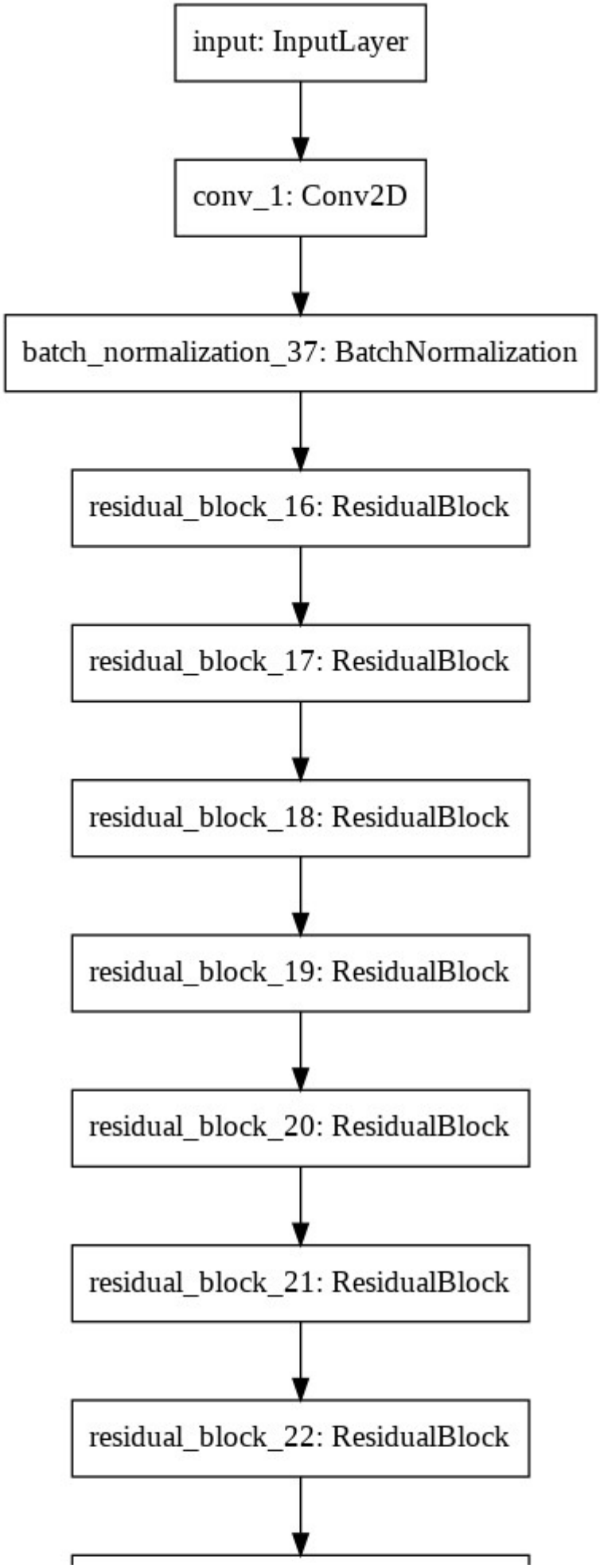
```
Model: "model_1"
_________________________________________________________________
```

```
_________________________________________
Layer (type)                 Output Shape              Param #     C
onnected to
==================================================================
==================================
input (InputLayer)           [(None, 224, 224, 3)      0
__________________________________________________________________
_________________________________________
conv_1 (Conv2D)              (None, 224, 224, 64)      9472        i
nput[0][0]
__________________________________________________________________
_________________________________________
batch_normalization_37 (BatchNo (None, 224, 224, 64)   256         c
onv_1[0][0]
__________________________________________________________________
_________________________________________
residual_block_16 (ResidualBloc (None, 112, 112, 64)   78784       b
atch_normalization_37[0][0]
__________________________________________________________________
_________________________________________
residual_block_17 (ResidualBloc (None, 112, 112, 64)   74368       r
esidual_block_16[0][0]
__________________________________________________________________
_________________________________________
residual_block_18 (ResidualBloc (None, 112, 112, 64)   74368       r
esidual_block_17[0][0]
__________________________________________________________________
_________________________________________
residual_block_19 (ResidualBloc (None, 56, 56, 128)    231296      r
esidual_block_18[0][0]
__________________________________________________________________
_________________________________________
residual_block_20 (ResidualBloc (None, 56, 56, 128)    296192      r
esidual_block_19[0][0]
__________________________________________________________________
_________________________________________
residual_block_21 (ResidualBloc (None, 56, 56, 128)    296192      r
esidual_block_20[0][0]
__________________________________________________________________
_________________________________________
residual_block_22 (ResidualBloc (None, 56, 56, 128)    296192      r
esidual_block_21[0][0]
__________________________________________________________________
_________________________________________
residual_block_23 (ResidualBloc (None, 28, 28, 256)    921344      r
esidual_block_22[0][0]
__________________________________________________________________
_________________________________________
residual_block_24 (ResidualBloc (None, 28, 28, 256)    1182208     r
esidual_block_23[0][0]
__________________________________________________________________
_________________________________________
residual_block_25 (ResidualBloc (None, 28, 28, 256)    1182208     r
esidual_block_24[0][0]
__________________________________________________________________
_________________________________________
residual_block_26 (ResidualBloc (None, 28, 28, 256)    1182208     r
esidual_block_25[0][0]
__________________________________________________________________
_________________________________________
```

```
residual_block_27 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_26[0][0]
__________________________________________________________________________
_____________________________
residual_block_28 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_27[0][0]
__________________________________________________________________________
_____________________________
residual_block_29 (ResidualBloc (None, 14, 14, 512)  3677696     r
esidual_block_28[0][0]
__________________________________________________________________________
_____________________________
residual_block_30 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_29[0][0]
__________________________________________________________________________
_____________________________
residual_block_31 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_30[0][0]
__________________________________________________________________________
_____________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_31[0][0]
__________________________________________________________________________
_____________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
__________________________________________________________________________
_____________________________
action_class (Dense)            (None, 5)            2565        f
lat_1[0][0]
__________________________________________________________________________
_____________________________
action (Dense)                  (None, 21)           10773       f
lat_1[0][0]
==========================================================================
================================
Total params: 21,327,962
Trainable params: 21,310,810
Non-trainable params: 17,152
__________________________________________________________________________
```

In [ ]:

Out[44]:

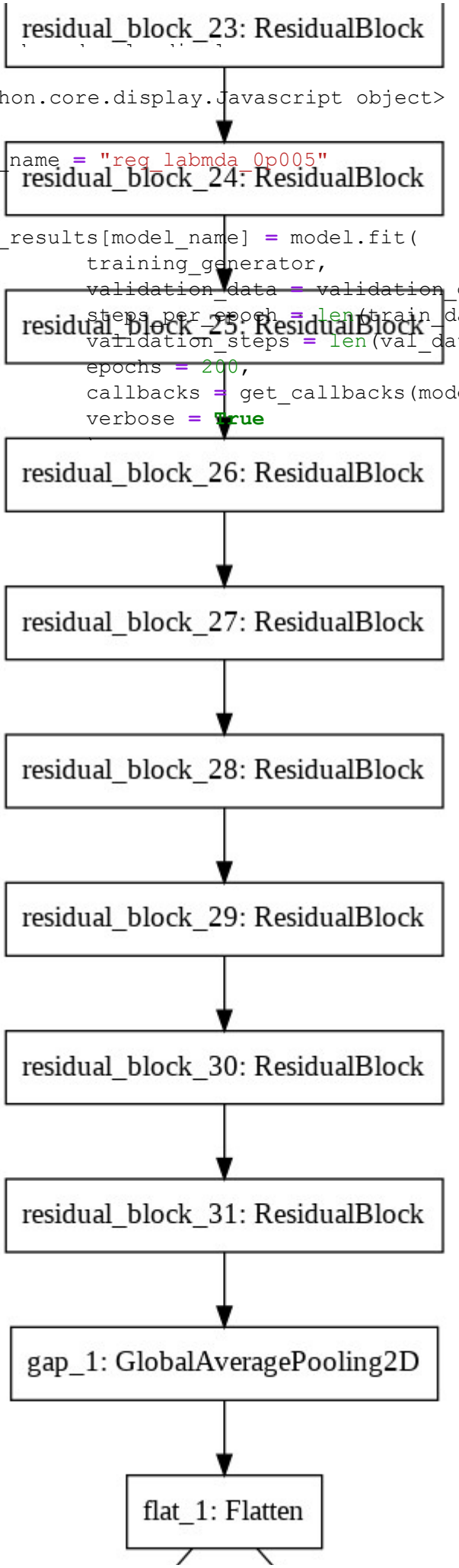

input: InputLayer
conv_1: Conv2D
batch_normalization_37: BatchNormalization
residual_block_16: ResidualBlock
residual_block_17: ResidualBlock
residual_block_18: ResidualBlock
residual_block_19: ResidualBlock
residual_block_20: ResidualBlock
residual_block_21: ResidualBlock
residual_block_22: ResidualBlock

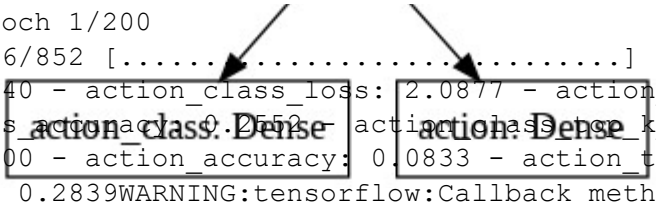
In [ ]:

<IPython.core.display.Javascript object>

In [ ]:

```
model_name = "reg_labmda_0p005"

model_results[model_name] = model.fit(
        training_generator,
        validation_data = validation_generator,
        steps_per_epoch = len(train_data_df)//BATCH_SIZE,
        validation_steps = len(val_data_df)//BATCH_SIZE,
        epochs = 200,
        callbacks = get_callbacks(model_name),
        verbose = True
```






```
Epoch 1/200
  6/852 [..............................] - ETA: 11:57 - loss: 258.
5840 - action_class_loss: 2.0877 - action_loss: 3.3607 - action_cl
ass_accuracy: 0.2552 - action_class_top_k_categorical_accuracy: 1.
0000 - action_accuracy: 0.0833 - action_top_k_categorical_accurac
y: 0.2839WARNING:tensorflow:Callback method `on_train_batch_end` i
s slow compared to the batch time (batch time: 0.2316s vs `on_trai
n_batch_end` time: 0.5309s). Check your callbacks.
852/852 [==============================] - 555s 645ms/step - loss:
20.0061 - action_class_loss: 1.4455 - action_loss: 2.7996 - action
_class_accuracy: 0.3857 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.1410 - action_top_k_categorical_accura
cy: 0.4905 - val_loss: 5.0512 - val_action_class_loss: 1.4840 - va
l_action_loss: 2.8109 - val_action_class_accuracy: 0.4062 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.1562 - val_action_top_k_categorical_accuracy: 0.5059

Epoch 00001: val_loss improved from inf to 5.05122, saving model t
o reg_labmda_0p005/cp.ckpt
============================== 1 ==============================

action_class_accuracy: 0.39, action_accuracy: 0.14
val_action_class_accuracy: 0.41, val_action_accuracy: 0.16

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.49
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.51

Epoch 2/200
852/852 [==============================] - 547s 641ms/step - loss:
4.7475 - action_class_loss: 1.3766 - action_loss: 2.6712 - action_
class_accuracy: 0.4220 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.1791 - action_top_k_categorical_accura
cy: 0.5529 - val_loss: 4.7240 - val_action_class_loss: 1.3553 - va
l_action_loss: 2.6988 - val_action_class_accuracy: 0.4219 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.1816 - val_action_top_k_categorical_accuracy: 0.5547

Epoch 00002: val_loss improved from 5.05122 to 4.72398, saving mod
el to reg_labmda_0p005/cp.ckpt
============================== 2 ==============================

action_class_accuracy: 0.42, action_accuracy: 0.18
val_action_class_accuracy: 0.42, val_action_accuracy: 0.18

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.55
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.55

Epoch 3/200
852/852 [==============================] - 547s 641ms/step - loss:
4.7074 - action_class_loss: 1.3358 - action_loss: 2.5851 - action_
class_accuracy: 0.4386 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.2033 - action_top_k_categorical_accura
cy: 0.5916 - val_loss: 5.5277 - val_action_class_loss: 1.4949 - va
l_action_loss: 3.1532 - val_action_class_accuracy: 0.4492 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.1875 - val_action_top_k_categorical_accuracy: 0.5918
```

```
Epoch 00003: val_loss did not improve from 4.72398
============================== 3 ==============================

action_class_accuracy: 0.44, action_accuracy: 0.20
val_action_class_accuracy: 0.45, val_action_accuracy: 0.19

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.59
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.59

Epoch 4/200
852/852 [==============================] - 546s 641ms/step - loss:
4.5361 - action_class_loss: 1.2779 - action_loss: 2.4679 - action_
class_accuracy: 0.4689 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.2368 - action_top_k_categorical_accura
cy: 0.6375 - val_loss: 4.7931 - val_action_class_loss: 1.3704 - va
l_action_loss: 2.5901 - val_action_class_accuracy: 0.4805 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.1973 - val_action_top_k_categorical_accuracy: 0.6230

Epoch 00004: val_loss did not improve from 4.72398
============================== 4 ==============================

action_class_accuracy: 0.47, action_accuracy: 0.24
val_action_class_accuracy: 0.48, val_action_accuracy: 0.20

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.64
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.62

Epoch 5/200
852/852 [==============================] - 546s 641ms/step - loss:
4.4434 - action_class_loss: 1.2331 - action_loss: 2.3782 - action_
class_accuracy: 0.4932 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.2688 - action_top_k_categorical_accura
cy: 0.6693 - val_loss: 5.1668 - val_action_class_loss: 1.2840 - va
l_action_loss: 3.0444 - val_action_class_accuracy: 0.5234 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.2441 - val_action_top_k_categorical_accuracy: 0.6484

Epoch 00005: val_loss did not improve from 4.72398
============================== 5 ==============================

action_class_accuracy: 0.49, action_accuracy: 0.27
val_action_class_accuracy: 0.52, val_action_accuracy: 0.24

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.67
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.65

Epoch 6/200
852/852 [==============================] - 546s 641ms/step - loss:
4.3312 - action_class_loss: 1.1765 - action_loss: 2.2861 - action_
class_accuracy: 0.5231 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.2941 - action_top_k_categorical_accura
cy: 0.6989 - val_loss: 4.3686 - val_action_class_loss: 1.1855 - va
```

```
l_action_loss: 2.3582 - val_action_class_accuracy: 0.5332 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.2754 - val_action_top_k_categorical_accuracy: 0.7070

Epoch 00006: val_loss improved from 4.72398 to 4.36861, saving mod
el to reg_labmda_0p005/cp.ckpt
============================== 6 ==============================

action_class_accuracy: 0.52, action_accuracy: 0.29
val_action_class_accuracy: 0.53, val_action_accuracy: 0.28

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.70
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.71

Epoch 7/200
852/852 [==============================] - 546s 641ms/step - loss:
4.0793 - action_class_loss: 1.0901 - action_loss: 2.1365 - action_
class_accuracy: 0.5682 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.3383 - action_top_k_categorical_accura
cy: 0.7404 - val_loss: 7.2998 - val_action_class_loss: 2.5582 - va
l_action_loss: 3.9758 - val_action_class_accuracy: 0.5137 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.2715 - val_action_top_k_categorical_accuracy: 0.6680

Epoch 00007: val_loss did not improve from 4.36861
============================== 7 ==============================

action_class_accuracy: 0.57, action_accuracy: 0.34
val_action_class_accuracy: 0.51, val_action_accuracy: 0.27

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.74
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.67

Epoch 8/200
852/852 [==============================] - 546s 641ms/step - loss:
3.8462 - action_class_loss: 1.0092 - action_loss: 1.9899 - action_
class_accuracy: 0.6085 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.3795 - action_top_k_categorical_accura
cy: 0.7794 - val_loss: 4.4414 - val_action_class_loss: 1.2581 - va
l_action_loss: 2.3576 - val_action_class_accuracy: 0.5664 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.3301 - val_action_top_k_categorical_accuracy: 0.7285

Epoch 00008: val_loss did not improve from 4.36861
============================== 8 ==============================

action_class_accuracy: 0.61, action_accuracy: 0.38
val_action_class_accuracy: 0.57, val_action_accuracy: 0.33

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.78
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.73

Epoch 9/200
852/852 [==============================] - 547s 642ms/step - loss:
```

```
3.5843 - action_class_loss: 0.9269 - action_loss: 1.8304 - action_
class_accuracy: 0.6456 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.4273 - action_top_k_categorical_accura
cy: 0.8142 - val_loss: 3.8903 - val_action_class_loss: 1.0697 - va
l_action_loss: 2.0367 - val_action_class_accuracy: 0.5977 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.3809 - val_action_top_k_categorical_accuracy: 0.7520

Epoch 00009: val_loss improved from 4.36861 to 3.89027, saving mod
el to reg_labmda_0p005/cp.ckpt
============================== 9 ==============================

action_class_accuracy: 0.65, action_accuracy: 0.43
val_action_class_accuracy: 0.60, val_action_accuracy: 0.38

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.81
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.75

Epoch 10/200
852/852 [==============================] - 547s 642ms/step - loss:
3.2859 - action_class_loss: 0.8257 - action_loss: 1.6408 - action_
class_accuracy: 0.6898 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.4823 - action_top_k_categorical_accura
cy: 0.8528 - val_loss: 4.0846 - val_action_class_loss: 1.1509 - va
l_action_loss: 2.1007 - val_action_class_accuracy: 0.6113 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.3965 - val_action_top_k_categorical_accuracy: 0.7715

Epoch 00010: val_loss did not improve from 3.89027
============================== 10 ==============================

action_class_accuracy: 0.69, action_accuracy: 0.48
val_action_class_accuracy: 0.61, val_action_accuracy: 0.40

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.85
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.77

Epoch 11/200
852/852 [==============================] - 547s 642ms/step - loss:
3.0195 - action_class_loss: 0.7200 - action_loss: 1.4553 - action_
class_accuracy: 0.7306 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.5357 - action_top_k_categorical_accura
cy: 0.8855 - val_loss: 4.1522 - val_action_class_loss: 1.1711 - va
l_action_loss: 2.1416 - val_action_class_accuracy: 0.6094 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.4277 - val_action_top_k_categorical_accuracy: 0.7539

Epoch 00011: val_loss did not improve from 3.89027
============================== 11 ==============================

action_class_accuracy: 0.73, action_accuracy: 0.54
val_action_class_accuracy: 0.61, val_action_accuracy: 0.43

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.89
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
```

```
_k_categorical_accuracy: 0.75

Epoch 12/200
852/852 [==============================] - 547s 642ms/step - loss:
2.7886 - action_class_loss: 0.6177 - action_loss: 1.2914 - action_
class_accuracy: 0.7729 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.5858 - action_top_k_categorical_accura
cy: 0.9139 - val_loss: 4.1367 - val_action_class_loss: 1.1699 - va
l_action_loss: 2.0520 - val_action_class_accuracy: 0.6602 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.4297 - val_action_top_k_categorical_accuracy: 0.8047

Epoch 00012: val_loss did not improve from 3.89027
============================== 12 ==============================

action_class_accuracy: 0.77, action_accuracy: 0.59
val_action_class_accuracy: 0.66, val_action_accuracy: 0.43

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.91
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.80

Epoch 13/200
852/852 [==============================] - 547s 642ms/step - loss:
2.5758 - action_class_loss: 0.5281 - action_loss: 1.1375 - action_
class_accuracy: 0.8072 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.6330 - action_top_k_categorical_accura
cy: 0.9356 - val_loss: 4.5365 - val_action_class_loss: 1.3630 - va
l_action_loss: 2.2372 - val_action_class_accuracy: 0.6250 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.4160 - val_action_top_k_categorical_accuracy: 0.7812

Epoch 00013: val_loss did not improve from 3.89027
============================== 13 ==============================

action_class_accuracy: 0.81, action_accuracy: 0.63
val_action_class_accuracy: 0.62, val_action_accuracy: 0.42

action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.94
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.78

Epoch 14/200
852/852 [==============================] - 547s 642ms/step - loss:
2.4293 - action_class_loss: 0.4631 - action_loss: 1.0249 - action_
class_accuracy: 0.8316 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.6688 - action_top_k_categorical_accura
cy: 0.9505 - val_loss: 4.4483 - val_action_class_loss: 1.2791 - va
l_action_loss: 2.1966 - val_action_class_accuracy: 0.5996 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
y: 0.4199 - val_action_top_k_categorical_accuracy: 0.7812

Epoch 00014: val_loss did not improve from 3.89027
============================== 14 ==============================

action_class_accuracy: 0.83, action_accuracy: 0.67
val_action_class_accuracy: 0.60, val_action_accuracy: 0.42
```

```
action_class_top_k_categorical_accuracy : 1.00, action_top_k_categ
orical_accuracy: 0.95
val_action_class_top_k_categorical_accuracy : 1.00, val_action_top
_k_categorical_accuracy: 0.78

Epoch 15/200
```

In [ ]:

```
# Save model results
model_results[model_name] = model_results[model_name].history

!rm -r "drive/MyDrive/model_results.json"
with open("drive/MyDrive/model_results.json", "w") as fp:
```

In [ ]:

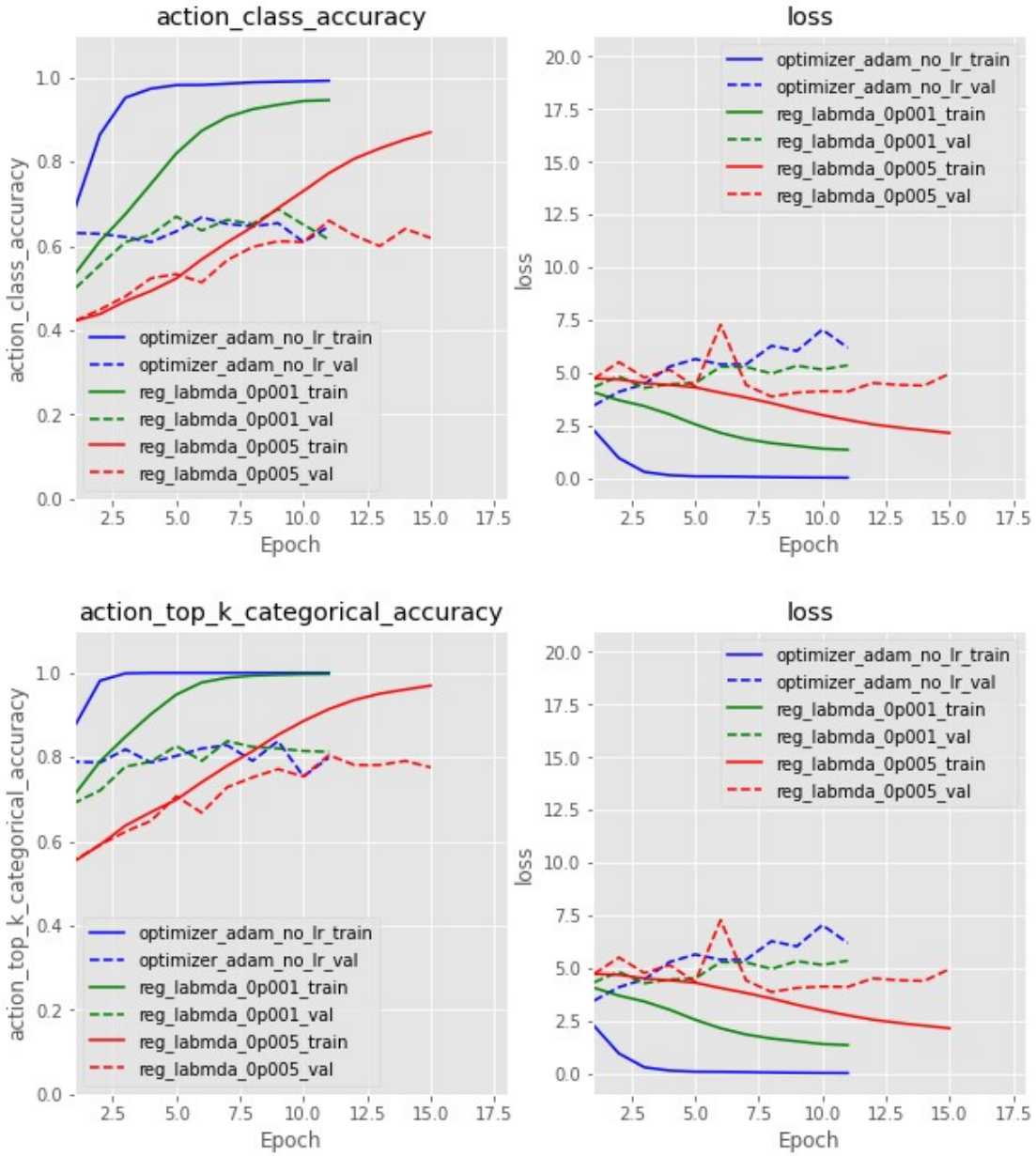
```
# Save model weights
```

```
WARNING:absl:Found untraced functions such as conv2d_36_layer_call
_fn, conv2d_36_layer_call_and_return_conditional_losses, conv2d_37
_layer_call_fn, conv2d_37_layer_call_and_return_conditional_losse
s, conv2d_39_layer_call_fn while saving (showing 5 of 160). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: drive/MyDrive/reg_labmda_0p005_
model/assets

INFO:tensorflow:Assets written to: drive/MyDrive/reg_labmda_0p005_
model/assets
```

In [52]:

```
names = { 'optimizer_adam_no_lr', 'reg_labmda_0p005' , 'reg_labmda_
```

In [53]:
```
plotter(model_results_subset, metric = 'action_class_accuracy', yli
plotter(model_results_subset, metric = 'action_top_k_categorical_ac
```



In [ ]:

✓ **Observations:**

- There is less overfitting, but it still does overfit towards the end. Might need to increae the the reg_lambda parameter, as well as give the model a higher patience parameter in the early stopping in the callbacks

In [ ]:

In [ ]:

In [ ]:

Regularization - L2 with lambda = 0.0075

In [ ]:
```
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                        data_frame = train_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/trai
                                        )

validation_generator = DataGenerator(
                                        data_frame = val_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/vali
                                        image_column = "FileName_scaled
                                        )
```

In [ ]:
```
filters = [64, 128, 256, 512]
block_size = [3,4,6,3]
reg_lambda = 0.0075

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
 initial_learning_rate = 0.1, decay_steps = len(train_data_df)//BAT
  staircase=False)
```

In [ ]:
```

first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")

conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
                kernel_regularizer=tf.keras.regularizers.l2(reg_lam
                padding = "same",
                name = "conv_1")(input_)
```

```
bn_1 = BatchNormalization(momentum=0.4)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)

for nFilters, nBlocks in zip(filters, block_size):

    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re

    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l2
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(flat_1)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l2(reg
                    kernel_initializer=tf.keras.initializers.Random
                    name =  "action")(flat_1)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
           'accuracy',
           tf.keras.metrics.TopKCategoricalAccuracy(
                k=5,
                name="top_k_categorical_accuracy",
                dtype=None
           )

           ]

)
```

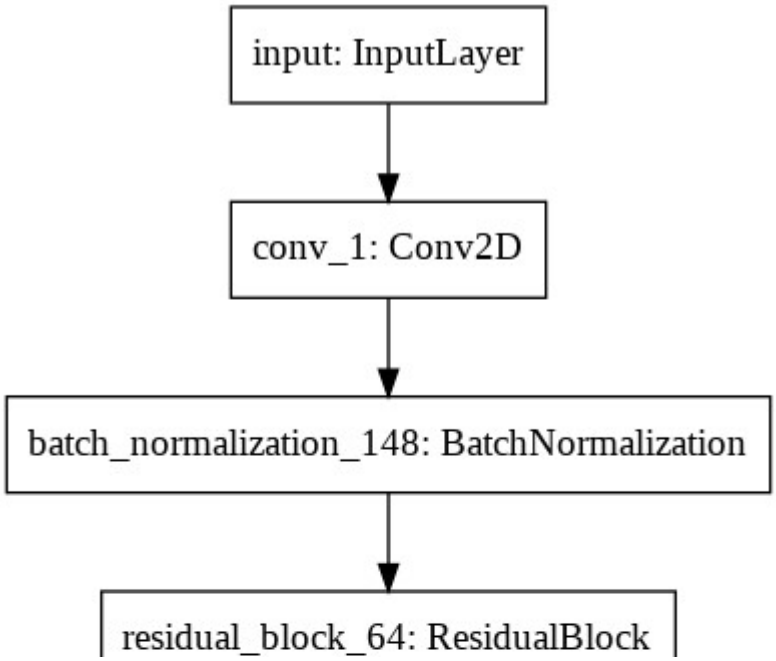
```
Model: "model_4"
__________________________________________________________________________
_____________________________________
Layer (type)                 Output Shape              Param #     C
onnected to
```

```
==================================================================
================================
input (InputLayer)              [(None, 224, 224, 3) 0

__________________________________
conv_1 (Conv2D)                 (None, 224, 224, 64) 9472        i
nput[0][0]

__________________________________
batch_normalization_148 (BatchN (None, 224, 224, 64) 256         c
onv_1[0][0]

__________________________________
residual_block_64 (ResidualBloc (None, 112, 112, 64) 78784       b
atch_normalization_148[0][0]

__________________________________
residual_block_65 (ResidualBloc (None, 112, 112, 64) 74368       r
esidual_block_64[0][0]

__________________________________
residual_block_66 (ResidualBloc (None, 112, 112, 64) 74368       r
esidual_block_65[0][0]

__________________________________
residual_block_67 (ResidualBloc (None, 56, 56, 128)  231296      r
esidual_block_66[0][0]

__________________________________
residual_block_68 (ResidualBloc (None, 56, 56, 128)  296192      r
esidual_block_67[0][0]

__________________________________
residual_block_69 (ResidualBloc (None, 56, 56, 128)  296192      r
esidual_block_68[0][0]

__________________________________
residual_block_70 (ResidualBloc (None, 56, 56, 128)  296192      r
esidual_block_69[0][0]

__________________________________
residual_block_71 (ResidualBloc (None, 28, 28, 256)  921344      r
esidual_block_70[0][0]

__________________________________
residual_block_72 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_71[0][0]

__________________________________
residual_block_73 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_72[0][0]

__________________________________
residual_block_74 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_73[0][0]

__________________________________
residual_block_75 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_74[0][0]

__________________________________
```

```
_____________________________________
residual_block_76 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_75[0][0]
_____________________________________
_____________________________________
residual_block_77 (ResidualBloc (None, 14, 14, 512)  3677696     r
esidual_block_76[0][0]
_____________________________________
_____________________________________
residual_block_78 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_77[0][0]
_____________________________________
_____________________________________
residual_block_79 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_78[0][0]
_____________________________________
_____________________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_79[0][0]
_____________________________________
_____________________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
_____________________________________
_____________________________________
action_class (Dense)            (None, 5)            2565        f
lat_1[0][0]
_____________________________________
_____________________________________
action (Dense)                  (None, 21)           10773       f
lat_1[0][0]
==================================================================
================================
Total params: 21,327,962
Trainable params: 21,310,810
Non-trainable params: 17,152
_____________________________________
```

In [ ]:

Out[72]:



In [ ]:

In [ ]:

```
<IPython.core.display.Javascript object>
```

In [ ]:
```
model_name = "reg_labmda_0p0075"


model_results[model_name] = model.fit(
            training_generator,
            validation_data = validation_generator,
            steps_per_epoch = len(train_data_df)//BATCH_SIZE,
            validation_steps = len(val_data_df)//BATCH_SIZE,
            epochs = 200,
            callbacks = get_callbacks(model_name),
            verbose = True
```

```
Epoch 1/200
  6/852 [..............................] - ETA: 11:42 - loss: 382.
1844 - action_class_loss: 1.9836 - action_loss: 3.3542 - action_cl
ass_accuracy: 0.2734 - action_class_top_k_categorical_accuracy: 1.
0000 - action_accuracy: 0.0625 - action_top_k_categorical_accurac
y: 0.2943WARNING:tensorflow:Callback method `on_train_batch_end` i
s slow compared to the batch time (batch time: 0.2292s vs `on_trai
n_batch_end` time: 0.5163s). Check your callbacks.

WARNING:tensorflow:Callback method `on_train_batch_end` is slow co
mpared to the batch time (batch time: 0.2292s vs `on_train_batch_e
nd` time: 0.5163s). Check your callbacks.

852/852 [==============================] - 554s 644ms/step - loss:
24.5107 - action_class_loss: 1.4761 - action_loss: 2.8614 - action
_class_accuracy: 0.3681 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.1234 - action_top_k_categorical_accura
cy: 0.4584 - val_loss: 5.1680 - val_action_class_loss: 1.4663 - va
l_action_loss: 2.8688 - val_action_class_accuracy: 0.3809 - val_ac
tion class top k categorical accuracy: 1.0000 - val action accurac
```

In [ ]:
```
# Save model results
model_results[model_name] = model_results[model_name].history

!rm -r "drive/MyDrive/model_results.json"
with open("drive/MyDrive/model_results.json", "w") as fp:
```

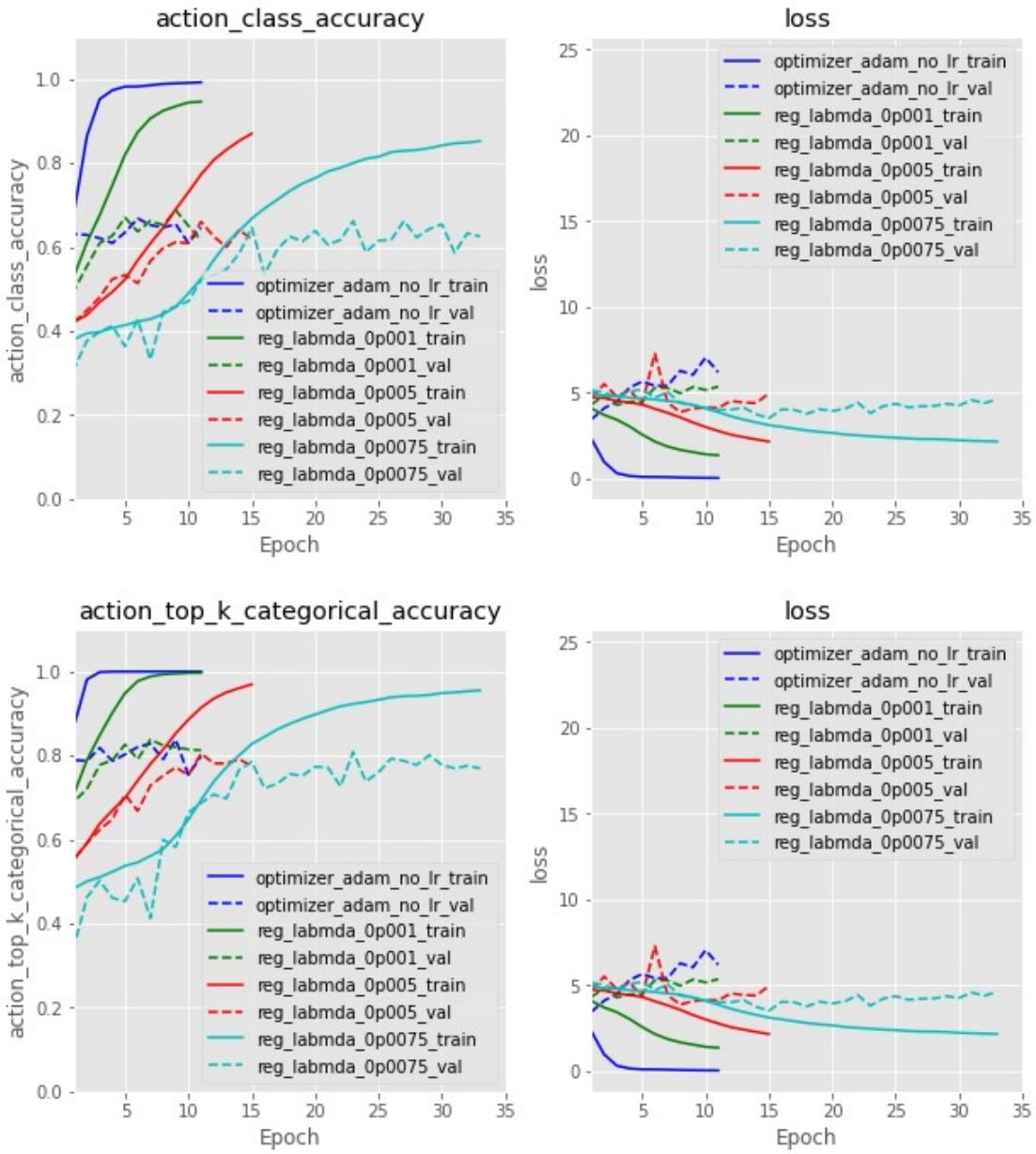
In [ ]:
```
# Save model weights
```

```
WARNING:absl:Found untraced functions such as conv2d_144_layer_cal
l_fn, conv2d_144_layer_call_and_return_conditional_losses, conv2d_
145_layer_call_fn, conv2d_145_layer_call_and_return_conditional_lo
sses, conv2d_147_layer_call_fn while saving (showing 5 of 160). Th
ese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: drive/MyDrive/reg_labmda_0p0075
_model/assets

INFO:tensorflow:Assets written to: drive/MyDrive/reg_labmda_0p0075
_model/assets
```

In [ ]:
```
file_ = open('drive/MyDrive/model_results.json')
```

In [ ]:
```
names = { 'optimizer_adam_no_lr', 'reg_labmda_0p005' , 'reg_labmda_
```

In [ ]:
```
plotter(model_results_subset, metric = 'action_class_accuracy', yli
plotter(model_results_subset, metric = 'action_top_k_categorical_ac
```



✓ **Observations:**

- Still overfitting with lots of epochs, and performance plateaus with approximately the same valuues.
- Might be worth trying a different regularization technique like L1

In [ ]:

In [ ]:

Regularization - L1 with lambda = 0.001

In [ ]:

In [ ]:

```
class ResidualBlock(tf.keras.layers.Layer):

    # Initialize components of the model
    def __init__(self, filter_num, stride=1, reg_lambda=0.0, kernel
        super(ResidualBlock, self).__init__()
        self.conv1 = tf.keras.layers.Conv2D(filters=filter_num,
                                            kernel_size = kernel_si
                                            strides=stride,
                                            kernel_initializer=tf.k
                                            kernel_regularizer=tf.k
                                            padding="same")
        self.bn1 = tf.keras.layers.BatchNormalization(momentum=mome
        self.conv2 = tf.keras.layers.Conv2D(filters=filter_num,
                                            kernel_size = kernel_si
                                            strides=1,
                                            kernel_initializer=tf.k
                                            kernel_regularizer=tf.k
                                            padding="same")
        self.bn2 = tf.keras.layers.BatchNormalization(momentum=mome
        if stride != 1:
            self.downsample = tf.keras.Sequential()
            self.downsample.add(tf.keras.layers.Conv2D(filters=filt
                                                       kernel_size=
                                                       kernel_initi
                                                       kernel_regul
                                                       strides=stri
            self.downsample.add(tf.keras.layers.BatchNormalization(
        else:
            self.downsample = lambda x: x

    # Define the forward function
    def call(self, inputs, training=None, **kwargs):
        residual = self.downsample(inputs)

        x = self.conv1(inputs)
        x = self.bn1(x, training=training)
        x = tf.nn.relu(x)
        x = self.conv2(x)
        x = self.bn2(x, training=training)

        output = tf.nn.relu(tf.keras.layers.add([residual, x]))

        return output

    def get_config(self):

        config = super().get_config().copy()
        config.update({
            'conv1': self.conv1,
            'bn1': self.bn1,
            'conv2': self.conv2,
            'bn2': self.bn2,
            'downsample': self.downsample,
```

```
        })
```

In [ ]:

```
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                        data_frame = train_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/trai
                                        )

validation_generator = DataGenerator(
                                        data_frame = val_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/vali
                                        image_column = "FileName_scaled
                                        )
```

In [ ]:

```
filters = [64, 128, 256, 512]
block_size = [3,4,6,3]
reg_lambda = 0.001

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
 initial_learning_rate = 0.1, decay_steps = len(train_data_df)//BAT
  staircase=False)
```

In [ ]:

```
first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")

conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
                kernel_regularizer=tf.keras.regularizers.l1(reg_lam
                padding = "same",
                name = "conv_1")(input_)


bn_1 = BatchNormalization(momentum=0.4)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)
```

```
for nFilters, nBlocks in zip(filters, block_size):

    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re

    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l1
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(flat_1)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l1(reg
                    kernel_initializer=tf.keras.initializers.Random
                    name =  "action")(flat_1)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
           'accuracy',
           tf.keras.metrics.TopKCategoricalAccuracy(
                k=5,
                name="top_k_categorical_accuracy",
                dtype=None
           )

           ]

)
```

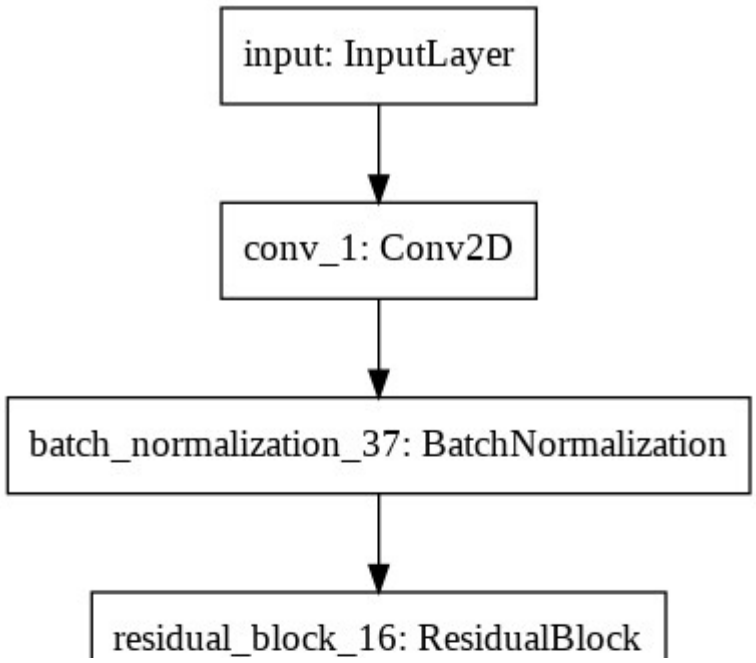
```
Model: "model_1"
__________________________________________________________________
_________________________________
Layer (type)                 Output Shape              Param #     C
onnected to
==================================================================
================================
```

```
input (InputLayer)              [(None, 224, 224, 3) 0
__________________________________________________________________________
__________________________________
conv_1 (Conv2D)                 (None, 224, 224, 64) 9472        i
nput[0][0]
__________________________________________________________________________
__________________________________
batch_normalization_37 (BatchNo (None, 224, 224, 64) 256         c
onv_1[0][0]
__________________________________________________________________________
__________________________________
residual_block_16 (ResidualBloc (None, 112, 112, 64) 78784       b
atch_normalization_37[0][0]
__________________________________________________________________________
__________________________________
residual_block_17 (ResidualBloc (None, 112, 112, 64) 74368       r
esidual_block_16[0][0]
__________________________________________________________________________
__________________________________
residual_block_18 (ResidualBloc (None, 112, 112, 64) 74368       r
esidual_block_17[0][0]
__________________________________________________________________________
__________________________________
residual_block_19 (ResidualBloc (None, 56, 56, 128)  231296      r
esidual_block_18[0][0]
__________________________________________________________________________
__________________________________
residual_block_20 (ResidualBloc (None, 56, 56, 128)  296192      r
esidual_block_19[0][0]
__________________________________________________________________________
__________________________________
residual_block_21 (ResidualBloc (None, 56, 56, 128)  296192      r
esidual_block_20[0][0]
__________________________________________________________________________
__________________________________
residual_block_22 (ResidualBloc (None, 56, 56, 128)  296192      r
esidual_block_21[0][0]
__________________________________________________________________________
__________________________________
residual_block_23 (ResidualBloc (None, 28, 28, 256)  921344      r
esidual_block_22[0][0]
__________________________________________________________________________
__________________________________
residual_block_24 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_23[0][0]
__________________________________________________________________________
__________________________________
residual_block_25 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_24[0][0]
__________________________________________________________________________
__________________________________
residual_block_26 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_25[0][0]
__________________________________________________________________________
__________________________________
residual_block_27 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_26[0][0]
__________________________________________________________________________
__________________________________
residual_block_28 (ResidualBloc (None, 28, 28, 256)  1182208     r
```

```
esidual_block_27[0][0]
__________________________________________________________________________
_____________________________
residual_block_29 (ResidualBloc (None, 14, 14, 512)  3677696     r
esidual_block_28[0][0]
__________________________________________________________________________
_____________________________
residual_block_30 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_29[0][0]
__________________________________________________________________________
_____________________________
residual_block_31 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_30[0][0]
__________________________________________________________________________
_____________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_31[0][0]
__________________________________________________________________________
_____________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
__________________________________________________________________________
_____________________________
action_class (Dense)            (None, 5)            2565        f
lat_1[0][0]
__________________________________________________________________________
_____________________________
action (Dense)                  (None, 21)           10773       f
lat_1[0][0]
==========================================================================
================================
Total params: 21,327,962
Trainable params: 21,310,810
Non-trainable params: 17,152
__________________________________________________________________________
```

In [ ]:

Out[46]:



In [ ]:

```
<IPython.core.display.Javascript object>
```

In [ ]:

```
model_name = "l1_reg_labmda_0p001"


model_results[model_name] = model.fit(
            training_generator,
            validation_data = validation_generator,
            steps_per_epoch = len(train_data_df)//BATCH_SIZE,
            validation_steps = len(val_data_df)//BATCH_SIZE,
            epochs = 200,
            callbacks = get_callbacks(model_name),
            verbose = True
```

```
Epoch 1/200
  6/852 [..............................] - ETA: 8:24 - loss: 810.7
603 - action_class_loss: 1.8098 - action_loss: 3.2528 - action_cla
ss_accuracy: 0.2344 - action_class_top_k_categorical_accuracy: 1.0
000 - action_accuracy: 0.0729 - action_top_k_categorical_accuracy:
0.3203WARNING:tensorflow:Callback method `on_train_batch_end` is s
low compared to the batch time (batch time: 0.1450s vs `on_train_b
atch_end` time: 0.3557s). Check your callbacks.
852/852 [==============================] - 329s 381ms/step - loss:
57.5056 - action_class_loss: 1.5573 - action_loss: 2.9903 - action
_class_accuracy: 0.3186 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.0948 - action_top_k_categorical_accura
cy: 0.3894 - val_loss: 17.6507 - val_action_class_loss: 1.6043 - v
al_action_loss: 2.9364 - val_action_class_accuracy: 0.3418 - val_a
ction_class_top_k_categorical_accuracy: 1.0000 - val_action_accura
cy: 0.1191 - val_action_top_k_categorical_accuracy: 0.4062

Epoch 00001: val_loss improved from inf to 17.65068, saving model
to l1_reg_labmda_0p001/cp.ckpt
```

In [ ]:

```
# Save model results
model_results[model_name] = model_results[model_name].history

!rm -r "drive/MyDrive/model_results.json"
with open("drive/MyDrive/model_results.json", "w") as fp:
```

In [ ]:

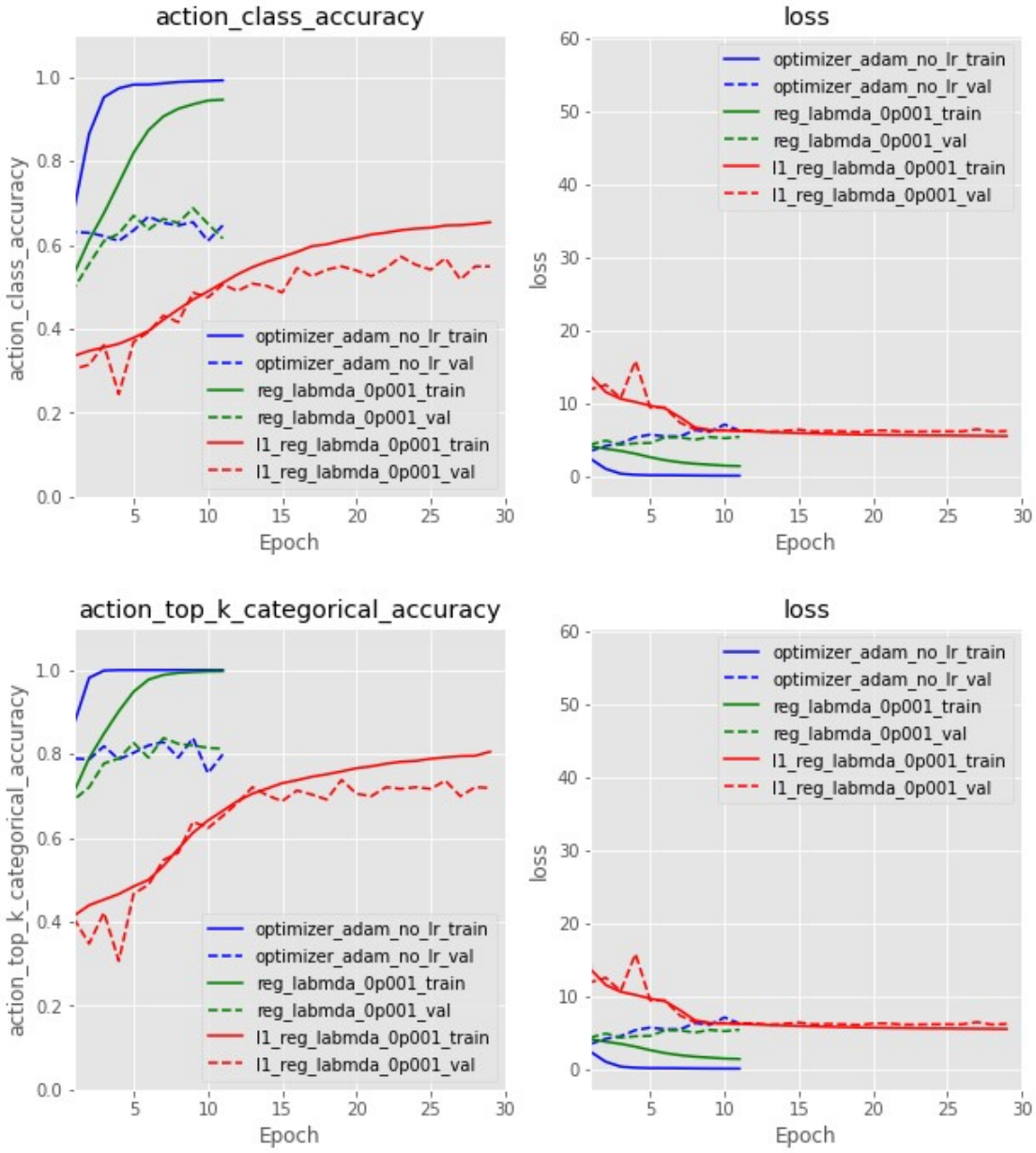
```
# Save model weights
```

```
WARNING:absl:Found untraced functions such as conv2d_36_layer_call
_fn, conv2d_36_layer_call_and_return_conditional_losses, conv2d_37
_layer_call_fn, conv2d_37_layer_call_and_return_conditional_losse
s, conv2d_39_layer_call_fn while saving (showing 5 of 160). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: drive/MyDrive/l1_reg_labmda_0p0
01_model/assets

INFO:tensorflow:Assets written to: drive/MyDrive/l1_reg_labmda_0p0
01_model/assets
```

In [ ]:

```
names = { 'optimizer_adam_no_lr', 'l1_reg_labmda_0p001', 'reg_labmd
model_results_subset = { key:value for key,value in model_results.i
```

In [ ]:
```
plotter(model_results_subset, metric = 'action_class_accuracy', yli
plotter(model_results_subset, metric = 'action_top_k_categorical_ac
```



In [ ]:

In [ ]:

✓ Observations:

- The model is significantly underfitting
- We might have to try to tune a different parameter, since the model's performance seems to be hitting a ceiling

In [ ]:

In [ ]:

Experimentation - Activation Function

- We try Leaky ReLU, which has a small slope for negative values instead of a flat slope

like ReLU. It is used mroe commonly for image classification tasks and tends to have better performance

In [ ]:

```
# add kernel size and momentum as parameters here, then remove this

class ResidualBlock(tf.keras.layers.Layer):

    # Initialize components of the model
    def __init__(self, filter_num, stride=1, reg_lambda=0.0, kernel
        super(ResidualBlock, self).__init__()
        self.conv1 = tf.keras.layers.Conv2D(filters=filter_num,
                                            kernel_size = kernel_si
                                            strides=stride,
                                            kernel_initializer=tf.k
                                            kernel_regularizer=tf.k
                                            padding="same")
        self.bn1 = tf.keras.layers.BatchNormalization(momentum=momen
        self.conv2 = tf.keras.layers.Conv2D(filters=filter_num,
                                            kernel_size = kernel_si
                                            strides=1,
                                            kernel_initializer=tf.k
                                            kernel_regularizer=tf.k
                                            padding="same")
        self.bn2 = tf.keras.layers.BatchNormalization(momentum=momen
        if stride != 1:
            self.downsample = tf.keras.Sequential()
            self.downsample.add(tf.keras.layers.Conv2D(filters=filt
                                                       kernel_size=
                                                       kernel_initi
                                                       kernel_regul
                                                       strides=stri
            self.downsample.add(tf.keras.layers.BatchNormalization(
        else:
            self.downsample = lambda x: x

    # Define the forward function
    def call(self, inputs, training=None, **kwargs):
        residual = self.downsample(inputs)

        x = self.conv1(inputs)
        x = self.bn1(x, training=training)
        x = tf.nn.leaky_relu(x)
        x = self.conv2(x)
        x = self.bn2(x, training=training)

        output = tf.nn.leaky_relu(tf.keras.layers.add([residual, x]

        return output

    def get_config(self):

        config = super().get_config().copy()
        config.update({
            'conv1': self.conv1,
            'bn1': self.bn1,
            'conv2': self.conv2,
            'bn2': self.bn2,
            'downsample': self.downsample,
        })
```

In [ ]:
```
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0
prefix=''
training_generator = DataGenerator(
                                        data_frame = train_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/trai
                                        )

validation_generator = DataGenerator(
                                        data_frame = val_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/vali
                                        image_column = "FileName_scaled
                                        )
```

In [ ]:
```
filters = [64, 128, 256, 512]
block_size = [3,4,6,3]
reg_lambda = 0.005

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
 initial_learning_rate = 0.1, decay_steps = len(train_data_df)//BAT
  staircase=False)
```

In [ ]:
```
first_time = True


input_ = Input(shape = (224, 224, 3), name = "input")

conv_1 = Conv2D(filters = 64,
                kernel_size = (7,7),
                strides = 1,
                kernel_initializer = tf.keras.initializers.RandomNo
                kernel_regularizer=tf.keras.regularizers.l2(reg_lam
                padding = "same",
                name = "conv_1")(input_)


bn_1 = BatchNormalization(momentum=0.4)(conv_1)

#maxpool_1 = MaxPool2D(pool_size = (2,2))(bn_1)
```

```
for nFilters, nBlocks in zip(filters, block_size):

    if first_time == True:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re
        first_time = False
    else:
        res_x = ResidualBlock(nFilters, stride = 2, reg_lambda = re

    for _ in range(1, nBlocks):

        res_x = ResidualBlock(nFilters, stride = 1, reg_lambda = re

        #maxpool_x = MaxPool2D(pool_size = (2,2))(res_x)

#reshape_1 = Reshape(res_x.shape[1:], name="reshape_1")(res_x)

gap_1 = GlobalAveragePooling2D(name = "gap_1")(res_x)


flat_1 = Flatten(name = "flat_1")(gap_1)

action_class = Dense(5, activation=tf.nn.softmax,
                        kernel_regularizer=tf.keras.regularizers.l2
                        kernel_initializer=tf.keras.initializers.Ra
                        name = "action_class")(flat_1)

action = Dense(21,  activation=tf.nn.softmax,
                    kernel_regularizer=tf.keras.regularizers.l2(reg
                    kernel_initializer=tf.keras.initializers.Random
                    name =  "action")(flat_1)

model = tf.keras.models.Model(input_, [action_class, action])

model.compile(
loss = {
    'action_class': 'categorical_crossentropy',
    'action': 'categorical_crossentropy'
},
optimizer = optimizer_,
metrics = [
           'accuracy',
           tf.keras.metrics.TopKCategoricalAccuracy(
                k=5,
                name="top_k_categorical_accuracy",
                dtype=None
           )

           ]

)
```

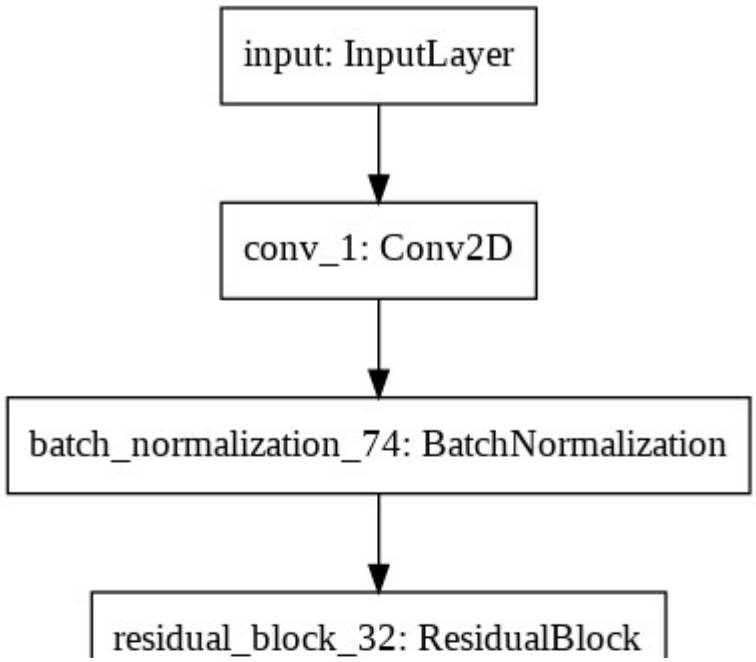
```
Model: "model_2"
__________________________________________________________________
_______________________________
Layer (type)                 Output Shape              Param #     C
onnected to
==================================================================
===============================
```

```
input (InputLayer)              [(None, 224, 224, 3) 0
__________________________________________________________________________
__________________________________
conv_1 (Conv2D)                 (None, 224, 224, 64) 9472        i
nput[0][0]
__________________________________________________________________________
__________________________________
batch_normalization_74 (BatchNo (None, 224, 224, 64) 256         c
onv_1[0][0]
__________________________________________________________________________
__________________________________
residual_block_32 (ResidualBloc (None, 112, 112, 64) 78784       b
atch_normalization_74[0][0]
__________________________________________________________________________
__________________________________
residual_block_33 (ResidualBloc (None, 112, 112, 64) 74368       r
esidual_block_32[0][0]
__________________________________________________________________________
__________________________________
residual_block_34 (ResidualBloc (None, 112, 112, 64) 74368       r
esidual_block_33[0][0]
__________________________________________________________________________
__________________________________
residual_block_35 (ResidualBloc (None, 56, 56, 128)  231296      r
esidual_block_34[0][0]
__________________________________________________________________________
__________________________________
residual_block_36 (ResidualBloc (None, 56, 56, 128)  296192      r
esidual_block_35[0][0]
__________________________________________________________________________
__________________________________
residual_block_37 (ResidualBloc (None, 56, 56, 128)  296192      r
esidual_block_36[0][0]
__________________________________________________________________________
__________________________________
residual_block_38 (ResidualBloc (None, 56, 56, 128)  296192      r
esidual_block_37[0][0]
__________________________________________________________________________
__________________________________
residual_block_39 (ResidualBloc (None, 28, 28, 256)  921344      r
esidual_block_38[0][0]
__________________________________________________________________________
__________________________________
residual_block_40 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_39[0][0]
__________________________________________________________________________
__________________________________
residual_block_41 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_40[0][0]
__________________________________________________________________________
__________________________________
residual_block_42 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_41[0][0]
__________________________________________________________________________
__________________________________
residual_block_43 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_42[0][0]
__________________________________________________________________________
__________________________________
residual_block_44 (ResidualBloc (None, 28, 28, 256)  1182208     r
```

```
esidual_block_43[0][0]
__________________________________________________________________________
_____________________________
residual_block_45 (ResidualBloc (None, 14, 14, 512)  3677696     r
esidual_block_44[0][0]
__________________________________________________________________________
_____________________________
residual_block_46 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_45[0][0]
__________________________________________________________________________
_____________________________
residual_block_47 (ResidualBloc (None, 14, 14, 512)  4723712     r
esidual_block_46[0][0]
__________________________________________________________________________
_____________________________
gap_1 (GlobalAveragePooling2D)  (None, 512)          0           r
esidual_block_47[0][0]
__________________________________________________________________________
_____________________________
flat_1 (Flatten)                (None, 512)          0           g
ap_1[0][0]
__________________________________________________________________________
_____________________________
action_class (Dense)            (None, 5)            2565        f
lat_1[0][0]
__________________________________________________________________________
_____________________________
action (Dense)                  (None, 21)           10773       f
lat_1[0][0]
==========================================================================
==============================
Total params: 21,327,962
Trainable params: 21,310,810
Non-trainable params: 17,152
__________________________________________________________________________
```

In [ ]:

Out[69]:



In [ ]:
```
!rm -r logs
```

In [ ]:

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 830), started 2:57:48 ago.
(Use '!kill 830' to kill it.)
```

```
<IPython.core.display.Javascript object>
```

In [ ]:

```
model_name = "activation_leaky_relu"


model_results[model_name] = model.fit(
            training_generator,
            validation_data = validation_generator,
            steps_per_epoch = len(train_data_df)//BATCH_SIZE,
            validation_steps = len(val_data_df)//BATCH_SIZE,
            epochs = 200,
            callbacks = get_callbacks(model_name),
            verbose = True
```

```
Epoch 1/200
  6/852 [..............................] - ETA: 8:34 - loss: 259.1
831 - action_class_loss: 2.0968 - action_loss: 3.5281 - action_cla
ss_accuracy: 0.2708 - action_class_top_k_categorical_accuracy: 1.0
000 - action_accuracy: 0.0755 - action_top_k_categorical_accuracy:
0.3229WARNING:tensorflow:Callback method `on_train_batch_end` is s
low compared to the batch time (batch time: 0.1472s vs `on_train_b
atch_end` time: 0.3634s). Check your callbacks.

WARNING:tensorflow:Callback method `on_train_batch_end` is slow co
mpared to the batch time (batch time: 0.1472s vs `on_train_batch_e
nd` time: 0.3634s). Check your callbacks.

852/852 [==============================] - 331s 383ms/step - loss:
19.2723 - action_class_loss: 1.4481 - action_loss: 2.8004 - action
_class_accuracy: 0.3799 - action_class_top_k_categorical_accuracy:
1.0000 - action_accuracy: 0.1463 - action_top_k_categorical_accura
cy: 0.4914 - val_loss: 4.7852 - val_action_class_loss: 1.3889 - va
l_action_loss: 2.7201 - val_action_class_accuracy: 0.4043 - val_ac
tion_class_top_k_categorical_accuracy: 1.0000 - val_action_accurac
```

In [ ]:

```
# Save model results
model_results[model_name] = model_results[model_name].history

!rm -r "drive/MyDrive/model_results.json"
with open("drive/MyDrive/model_results.json", "w") as fp:
```

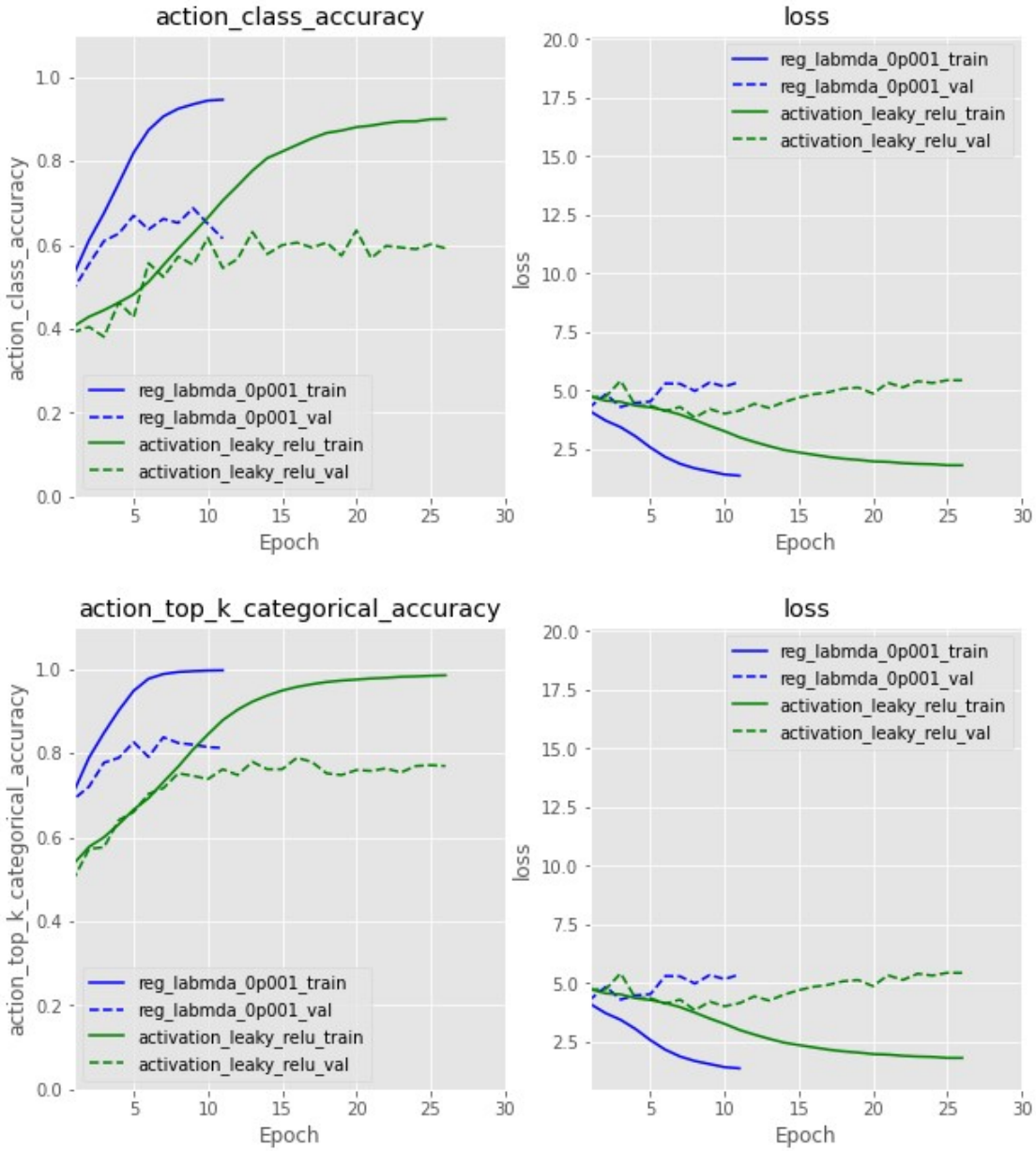
In [ ]:

```
# Save model weights
```

```
WARNING:absl:Found untraced functions such as conv2d_72_layer_call
_fn, conv2d_72_layer_call_and_return_conditional_losses, conv2d_73
_layer_call_fn, conv2d_73_layer_call_and_return_conditional_losse
s, conv2d_75_layer_call_fn while saving (showing 5 of 160). These
functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: drive/MyDrive/activation_leaky_
relu_model/assets

INFO:tensorflow:Assets written to: drive/MyDrive/activation_leaky_
relu_model/assets
```

In [60]:
```
names = { 'reg_labmda_0p001', 'activation_leaky_relu'}
```

In [61]:
```
plotter(model_results_subset, metric = 'action_class_accuracy', yli
plotter(model_results_subset, metric = 'action_top_k_categorical_ac
```



✓ Observations:

- There is no improvement in performance, in fact it does slightly worse

In [141]:

In [ ]:

In [ ]:

Test Data Evaluation

In [62]:

Out[62]:

	FileName	action	action_class	action_num	action_class_n
535	Img_257.jpg	rowing_a_boat	other_activity	12	
1369	Img_4929.jpg	riding_a_bike	other_activity	10	
1171	Img_4372.jpg	climbing	other_activity	1	
2921	Img_9279.jpg	walking_the_dog	Interacting_with_animal	18	
736	Img_3148.jpg	texting_message	using_comm_device	16	
...	...	...	...	...	
1076	Img_4112.jpg	riding_a_horse	Interacting_with_animal	11	
545	Img_2596.jpg	cleaning_the_floor	domestic_work	0	
1645	Img_5672.jpg	jumping	other_activity	6	
894	Img_3588.jpg	cooking	domestic_work	2	
1387	Img_4963.jpg	playing_violin	playing_musical_instrument	9	

758 rows × 6 columns

In [63]:

```
best_model = tf.keras.models.load_model("drive/MyDrive/reg_labmda_0

best_model.summary()
```

```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   C
onnected to
==================================================================
input (InputLayer)           [(None, 224, 224, 3)      0
_________________________________________________________________
conv_1 (Conv2D)              (None, 224, 224, 64)      9472      i
nput[0][0]
_________________________________________________________________
batch_normalization_148 (BatchN (None, 224, 224, 64)   256       c
onv_1[0][0]
_________________________________________________________________
residual_block_64 (ResidualBloc (None, 112, 112, 64)   78784     b
atch_normalization_148[0][0]
_________________________________________________________________
residual_block_65 (ResidualBloc (None, 112, 112, 64)   74368     r
esidual_block_64[0][0]
_________________________________________________________________
residual_block_66 (ResidualBloc (None, 112, 112, 64)   74368     r
esidual_block_65[0][0]
_________________________________________________________________
residual_block_67 (ResidualBloc (None, 56, 56, 128)    231296    r
esidual_block_66[0][0]
_________________________________________________________________
residual_block_68 (ResidualBloc (None, 56, 56, 128)    296192    r
esidual_block_67[0][0]
_________________________________________________________________
residual_block_69 (ResidualBloc (None, 56, 56, 128)    296192    r
esidual_block_68[0][0]
_________________________________________________________________
residual_block_70 (ResidualBloc (None, 56, 56, 128)    296192    r
esidual_block_69[0][0]
_________________________________________________________________
residual_block_71 (ResidualBloc (None, 28, 28, 256)    921344    r
esidual_block_70[0][0]
_________________________________________________________________
residual_block_72 (ResidualBloc (None, 28, 28, 256)    1182208   r
esidual_block_71[0][0]
_________________________________________________________________
residual_block_73 (ResidualBloc (None, 28, 28, 256)    1182208   r
esidual_block_72[0][0]
_________________________________________________________________
residual_block_74 (ResidualBloc (None, 28, 28, 256)    1182208   r
esidual_block_73[0][0]
```

```
_________________________________________________________________
residual_block_75 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_74[0][0]
_________________________________________________________________
residual_block_76 (ResidualBloc (None, 28, 28, 256)  1182208     r
esidual_block_75[0][0]
_________________________________________________________________
residual_block_77 (ResidualBloc (None, 14, 14, 512)  3677696     r
esidual_block_76[0][0]
```

In [69]:
```
BATCH_SIZE = 64

data_mean = 0.
data_std = 255.0
testing_generator = DataGenerator(
                                        data_frame = test_data_df,
                                        batch_size = BATCH_SIZE,
                                        data_mean = data_mean,
                                        data_std = data_std,
                                        create_folder = False,
                                        dim = (224, 224, 3),
                                        shuffle = True,
                                        augment = False,
                                        file_name = 'drive/MyDrive/test_
                                        image_column = "FileName_scaled
                                        )
```

In [142]:
```
results = best_model.evaluate(testing_generator, batch_size = BATCH_
```

```
11/11 [==============================] - 2s 152ms/step - loss: 4.8
139 - action_class_loss: 1.4272 - action_loss: 2.2995 - action_cla
ss_accuracy: 0.6406 - action_class_top_k_categorical_accuracy: 1.0
000 - action_accuracy: 0.4915 - action_top_k_categorical_accuracy:
0.8281
```

In [143]:
```

```

Out[143]:
```
[4.813885688781738,
 1.4271916151046753,
 2.299529790878296,
 0.640625,
 1.0,
 0.49147728085517883,
 0.828125]
```

In [144]:
```

```

In [146]:
```
for i in range(len(results)):
    print(metrics[i] + ": " + str(results[i]))
```

```
loss: 4.813885688781738
action_class_loss: 1.4271916151046753
```

✓ **Observations:**

- This matches the validation set performance

In [ ]:

In [ ]:

Making Predictions

In [77]:

In [78]:

Out[78]:

	FileName
0	Img_1005.jpg
1	Img_1012.jpg
2	Img_1014.jpg
3	Img_1015.jpg
4	Img_102.jpg
...	...
2094	Img_985.jpg
2095	Img_986.jpg
2096	Img_993.jpg
2097	Img_994.jpg
2098	Img_996.jpg

2099 rows × 1 columns

First we need to resize all the images to 224x224

In [79]:

```
!mkdir "predict_images_scaled"
for i in range(len(predict_df)):
    img=mpimg.imread(img_dir+predict_df['FileName'][i])
    #imgplot = plt.imshow(img)
    image_scaled = tf.image.convert_image_dtype(img, tf.float32)
    image_scaled = tf.image.resize(image_scaled, (224, 224))
```

In [80]:

In [81]:

Out[81]:

	FileName	FileName_scaled

	FileName	FileName_scaled
0	Img_1005.jpg	predict_images_scaled/Img_1005.jpg
1	Img_1012.jpg	predict_images_scaled/Img_1012.jpg
2	Img_1014.jpg	predict_images_scaled/Img_1014.jpg
3	Img_1015.jpg	predict_images_scaled/Img_1015.jpg
4	Img_102.jpg	predict_images_scaled/Img_102.jpg
...	...	...
2094	Img_985.jpg	predict_images_scaled/Img_985.jpg
2095	Img_986.jpg	predict_images_scaled/Img_986.jpg
2096	Img_993.jpg	predict_images_scaled/Img_993.jpg
2097	Img_994.jpg	predict_images_scaled/Img_994.jpg
2098	Img_996.jpg	predict_images_scaled/Img_996.jpg

Now we need to link the prediction values with their associated class labels for both tasks

In [165]:
```
action_labels = {}
for i in range(21):
    action_labels[i] = train_data_df["action"][train_data_df["actio

action_class_labels = {}
for i in range(5):
```

In [166]:

Out[166]:
```
{0: 'cleaning_the_floor',
 1: 'climbing',
 2: 'cooking',
 3: 'cutting_vegetables',
 4: 'feeding_a_horse',
 5: 'gardening',
 6: 'jumping',
 7: 'phoning',
 8: 'playing_guitar',
 9: 'playing_violin',
 10: 'riding_a_bike',
 11: 'riding_a_horse',
 12: 'rowing_a_boat',
 13: 'running',
 14: 'shooting_an_arrow',
 15: 'taking_photos',
 16: 'texting_message',
 17: 'using_a_computer',
 18: 'walking_the_dog',
 19: 'washing_dishes',
 20: 'watching_TV'}
```

In [167]:

Out[167]:

```
{0: 'Interacting_with_animal',
```

Now we can make the predictions

In [168]:
```
rows_list = []
for i in range(len(predict_df)):
    img = image.load_img(predict_df['FileName_scaled'][i])
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x /= 255
    #images = np.vstack([x])

    prediction = best_model.predict(x)

    dict_line = {"FileName": predict_df['FileName_scaled'][i], "act
    rows_list.append(dict_line)

predictions_df = pd.DataFrame(rows_list)
predictions_df.to_csv("predictions.csv", index = False)
```

In [ ]:

In [171]:

In [ ]: