SPORTS IMAGE CLASSIFICATION KAGGLE CHALLENGE

Purva Chiniya(17115068) - pchiniya@ee.iitr.ac.in ,purvachiniya15@gmail.com

Objective: To build a deep learning-based image classifier with the ability to best classify any image based on the 22 sports classes in the dataset.

Motivation: This work is a derivative of one of my research problems on Single Shot detectors for object detection. https://github.com/PurvaChiniya/Single_shot_detector

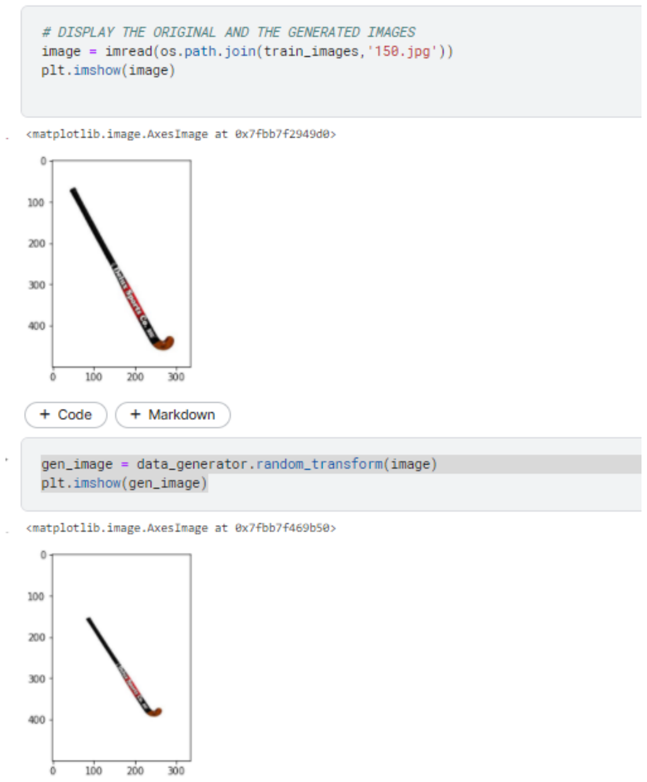
Proposal: The purpose of the project is to create a deep learning model to help classify sports images to its respective categorical sports accurately. The project consists of 2 modules:

1. CNN based model: trains the model on a simple 3 layer convolutional network to best understand the working.
2. VGG 16 transfer learning Model:finetunes the already trained vgg16 network on imagenet dataset to our model by changing the last few layers.

Workflow:

1. Data Preprocessing: The data images are preprocessed by normalizing and applying data augmentation, we did not need to do reshape here because the mean of dimensions was already (224,224).

2. Data Augmentation: All the test data cannot be loaded into the memory at once, and the number of images per sport need to be limited per batch of loading.
   That's why data augmentation is used at this step. Data Augmentation is used to normalize the image data and do normal operations like : rotations, horizontal shift, flips and rgb color variation.



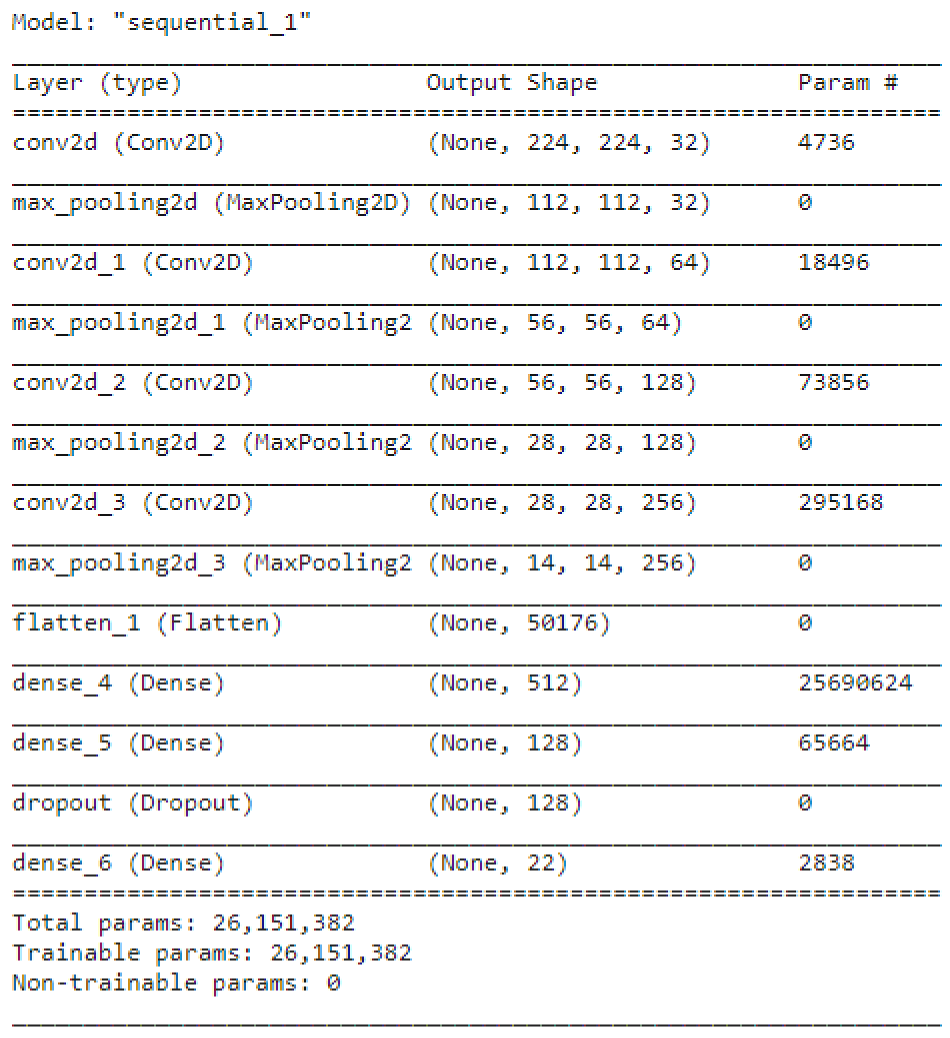
```
# DISPLAY THE ORIGINAL AND THE GENERATED IMAGES
image = imread(os.path.join(train_images,'150.jpg'))
plt.imshow(image)
```

```
<matplotlib.image.AxesImage at 0x7fbb7f2949d0>
```

```
gen_image = data_generator.random_transform(image)
plt.imshow(gen_image)
```

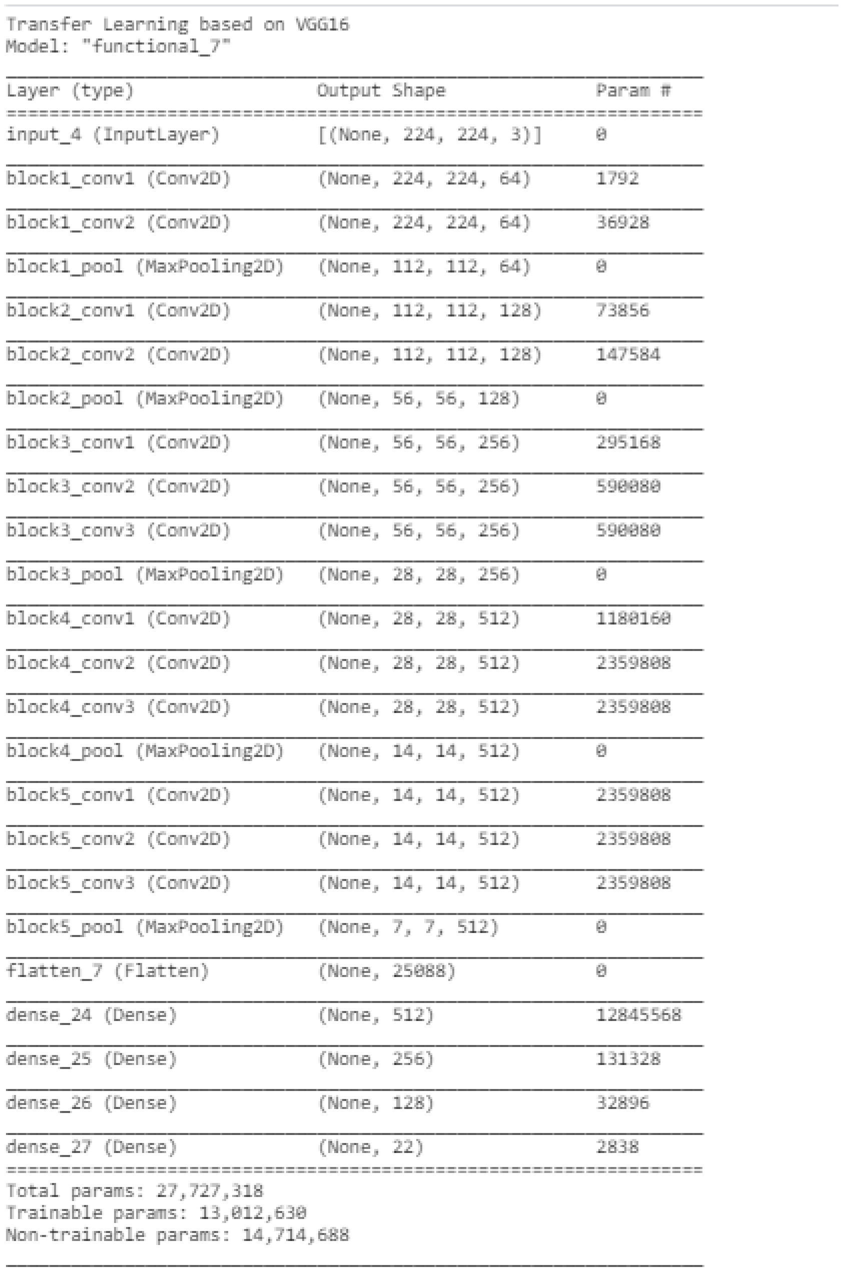
```
<matplotlib.image.AxesImage at 0x7fbb7f469b50>
```

3. Model Training: I tried two different approaches to make the classifier, first is a 3 layer convolutional network and second is imagenet trained vgg16 model, fine tuned to sports dataset.
   CNN Model - Trained on 3 layers of convolutional cells and max pooling and relu activation. Relu is used as the activation function. Model gives a training accuracy of 69.13 % for 35 epochs of training.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 224, 224, 32)      4736
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 112, 112, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 56, 56, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 256)       295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 256)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 50176)             0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               25690624
_________________________________________________________________
dense_5 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 22)                2838
=================================================================
Total params: 26,151,382
Trainable params: 26,151,382
Non-trainable params: 0
_________________________________________________________________
```

VGG 16 fine tuned model-
This is a transfer learning model. Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. So I finetuned our data on imagenet weights from the vgg16 model and finally added the classification layers at the bottom. Adam optimizer is used here as Adam performs faster and returns better accuracy. This model is trained for 30 epochs but gives barely 6.5% as accuracy.



```
Transfer Learning based on VGG16
Model: "functional_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten_7 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_24 (Dense)             (None, 512)               12845568
_________________________________________________________________
dense_25 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_26 (Dense)             (None, 128)               32896
_________________________________________________________________
dense_27 (Dense)             (None, 22)                2838
=================================================================
Total params: 27,727,318
Trainable params: 13,012,630
Non-trainable params: 14,714,688
_________________________________________________________________
```

4. Model Prediction: Prediction is done on the test dataset using the same data generator we used during training, and results are finally saved to a csv file.

Future Work:

1. To find more datasets for the 2nd model and tune them more to achieve better performance.
2. Try ensemble learning from both these models.