Sports Classification

Description

- Classify 6 types of sports using 3 different Deep learning algorithms to get the best accuracy.

Dataset

- Sports Dataset Consists of
  - Basketball
  - Football
  - Rowing
  - Swimming
  - Tennis
  - Yoga

Data Preprocessing

Data Augmentation

- Using Image Data generator to make Data augmentation such as (Zooming - shifting - Rotation - etc)

```python
TRAIN_PATH = '/content/Train'
VAL_PATH = '/content/Valid'


train_data_generator = image.ImageDataGenerator(
                                    rotation_range=30,
                                    zoom_range=0.15,
                                    width_shift_range=0.2,
                                    height_shift_range=0.2,
                                    )


val_data_generator = image.ImageDataGenerator()

train_generator = train_data_generator.flow_from_directory(directory= TRAIN_PATH,
                                                target_size=IMAGE_SIZE,
                                                color_mode= 'rgb',
                                                class_mode= 'categorical',
                                                batch_size= BATCH_SIZE)


val_generator = val_data_generator.flow_from_directory(directory= VAL_PATH,
                                                target_size=IMAGE_SIZE,
                                                color_mode= 'rgb',
                                                class_mode= 'categorical',
                                                batch_size= BATCH_SIZE)
```

Python

```
Found 1345 images belonging to 6 classes.
Found 336 images belonging to 6 classes.
```

Models

First Model (Normal Architecture)

- Model consists of 5 Convolution Layers.

```
model = Sequential()
model.add(Conv2D(input_shape=(224,224,3),filters=8,kernel_size=(3,3),padding="valid", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters=16, kernel_size=(3,3), padding="valid", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters=32, kernel_size=(3,3), padding="valid", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters=64, kernel_size=(3,3), padding="valid", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2)))
```

Python

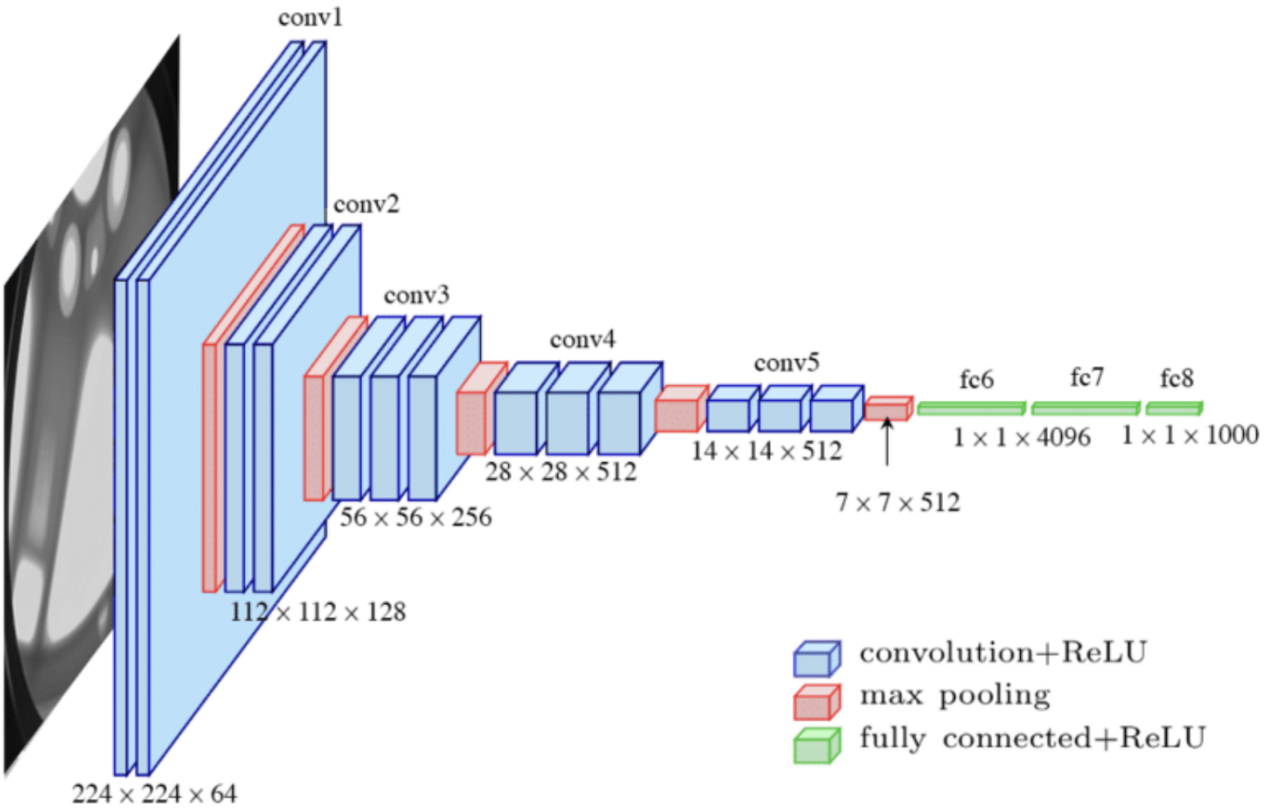
```
model.add(Flatten())
model.add(Dense(units=6, activation="softmax"))
```

Python

Model Accuracy

```
.
43/43 [==============================] - ETA: 0s - loss: 0.5667 - accuracy: 0.8037WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
43/43 [==============================] - 22s 516ms/step - loss: 0.5667 - accuracy: 0.8037 - val_loss: 0.5881 - val_accuracy: 0.8065 - lr: 1.0000e-10
```

Second Model (VGG16)



- VGG16, as its name suggests, is a 16-layer deep neural network. VGG16 is thus a relatively extensive network with a total of 138 million parameters—it's huge even by today's standards. However, the simplicity of the VGGNet16 architecture is its main attraction.

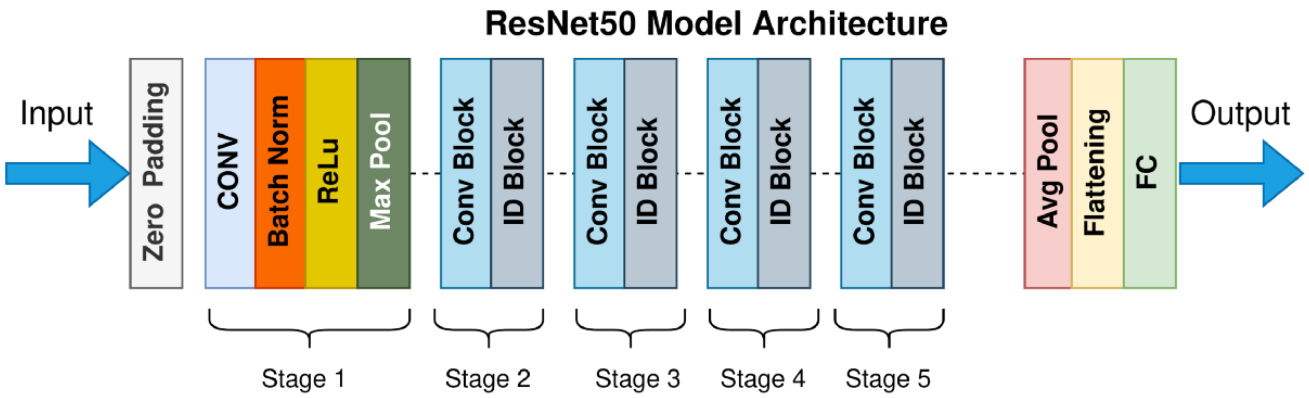
```
model = Sequential()
model.add(Conv2D(input_shape=(224,224,3),filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(Conv2D(filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(Flatten())
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=6, activation="softmax"))
```

Model Accuracy

```
Epoch 26: val_loss did not improve from 0.40784
43/43 [==============================] - 28s 638ms/step - loss: 0.3897 - accuracy: 0.8654 - val_loss: 0.4212 - val_accuracy: 0.8363 - lr: 1.0000e-16
Epoch 27/50
43/43 [==============================] - ETA: 0s - loss: 0.4031 - accuracy: 0.8610
Epoch 27: val_loss did not improve from 0.40784
43/43 [==============================] - 27s 630ms/step - loss: 0.4031 - accuracy: 0.8610 - val_loss: 0.4212 - val_accuracy: 0.8363 - lr: 1.0000e-16
Epoch 27: early stopping
```

Third Model (Resnet50)



- ResNet-50 has an architecture based on the model depicted above, but with one important difference. The 50-layer ResNet uses a bottleneck design for the building block.

```python
def convolutional_block(X, f, filters, stage, block, s=2):

    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    F1, F2, F3 = filters

    X_shortcut = X

    X = Conv2D(filters=F1, kernel_size=(1, 1), strides=(s, s), padding='valid', name=conv_name_base + '2a', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2a')(X)
    X = Activation('relu')(X)

    X = Conv2D(filters=F2, kernel_size=(f, f), strides=(1, 1), padding='same', name=conv_name_base + '2b', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2b')(X)
    X = Activation('relu')(X)

    X = Conv2D(filters=F3, kernel_size=(1, 1), strides=(1, 1), padding='valid', name=conv_name_base + '2c', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2c')(X)

    X_shortcut = Conv2D(filters=F3, kernel_size=(1, 1), strides=(s, s), padding='valid', name=conv_name_base + '1', kernel_initializer=glorot_uniform(seed=0))(X_shortcut)
    X_shortcut = BatchNormalization(axis=3, name=bn_name_base + '1')(X_shortcut)

    X = Add()([X, X_shortcut])
    X = Activation('relu')(X)

    return X
```

Model

```
def ResNet50(input_shape=(224, 224, 3)):

    X_input = Input(input_shape)

    X = ZeroPadding2D((3, 3))(X_input)

    X = Conv2D(64, (7, 7), strides=(2, 2), name='conv1', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name='bn_conv1')(X)
    X = Activation('relu')(X)
    X = MaxPooling2D((3, 3), strides=(2, 2))(X)

    X = convolutional_block(X, f=3, filters=[64, 64, 256], stage=2, block='a', s=1)
    X = identity_block(X, 3, [64, 64, 256], stage=2, block='b')
    X = identity_block(X, 3, [64, 64, 256], stage=2, block='c')


    X = convolutional_block(X, f=3, filters=[128, 128, 512], stage=3, block='a', s=2)
    X = identity_block(X, 3, [128, 128, 512], stage=3, block='b')
    X = identity_block(X, 3, [128, 128, 512], stage=3, block='c')
    X = identity_block(X, 3, [128, 128, 512], stage=3, block='d')

    X = convolutional_block(X, f=3, filters=[256, 256, 1024], stage=4, block='a', s=2)
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='b')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='c')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='d')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='e')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='f')

    X = X = convolutional_block(X, f=3, filters=[512, 512, 2048], stage=5, block='a', s=2)
    X = identity_block(X, 3, [512, 512, 2048], stage=5, block='b')
    X = identity_block(X, 3, [512, 512, 2048], stage=5, block='c')

    X = AveragePooling2D(pool_size=(2, 2), padding='same')(X)

    model = Model(inputs=X_input, outputs=X, name='ResNet50')
```

Model Accuracy

```
Epoch 1/30
43/43 [==============================] - 23s 533ms/step - loss: 0.0595 - accuracy: 0.9866 - val_loss: 0.2071 - val_accuracy: 0.9315 - lr: 1.0000e-05
Epoch 2/30
43/43 [==============================] - 23s 534ms/step - loss: 0.0717 - accuracy: 0.9784 - val_loss: 0.2000 - val_accuracy: 0.9405 - lr: 1.0000e-05
Epoch 3/30
43/43 [==============================] - 25s 578ms/step - loss: 0.0552 - accuracy: 0.9844 - val_loss: 0.2142 - val_accuracy: 0.9286 - lr: 1.0000e-05
Epoch 4/30
43/43 [==============================] - 23s 543ms/step - loss: 0.0586 - accuracy: 0.9829 - val_loss: 0.2048 - val_accuracy: 0.9345 - lr: 1.0000e-05
Epoch 5/30
43/43 [==============================] - 23s 539ms/step - loss: 0.0584 - accuracy: 0.9851 - val_loss: 0.2010 - val_accuracy: 0.9315 - lr: 1.0000e-06
Epoch 6/30
43/43 [==============================] - 24s 550ms/step - loss: 0.0499 - accuracy: 0.9859 - val_loss: 0.2000 - val_accuracy: 0.9315 - lr: 1.0000e-06
Epoch 7/30
43/43 [==============================] - 23s 539ms/step - loss: 0.0545 - accuracy: 0.9851 - val_loss: 0.2001 - val_accuracy: 0.9315 - lr: 1.0000e-07
Epoch 8/30
43/43 [==============================] - 23s 533ms/step - loss: 0.0489 - accuracy: 0.9859 - val_loss: 0.2000 - val_accuracy: 0.9315 - lr: 1.0000e-07
Epoch 9/30
43/43 [==============================] - 25s 571ms/step - loss: 0.0425 - accuracy: 0.9903 - val_loss: 0.2000 - val_accuracy: 0.9315 - lr: 1.0000e-08
Epoch 10/30
43/43 [==============================] - 23s 534ms/step - loss: 0.0437 - accuracy: 0.9903 - val_loss: 0.2000 - val_accuracy: 0.9315 - lr: 1.0000e-08
Epoch 11/30
43/43 [==============================] - 23s 535ms/step - loss: 0.0520 - accuracy: 0.9859 - val_loss: 0.2000 - val_accuracy: 0.9315 - lr: 1.0000e-09
Epoch 12/30
43/43 [==============================] - 23s 528ms/step - loss: 0.0501 - accuracy: 0.9844 - val_loss: 0.2000 - val_accuracy: 0.9315 - lr: 1.0000e-09
Epoch 13/30
43/43 [==============================] - 23s 543ms/step - loss: 0.0596 - accuracy: 0.9814 - val_loss: 0.2000 - val_accuracy: 0.9315 - lr: 1.0000e-10
```

Best Model in accuracy is Resnet50