Olympic Sports Recognition

Chitra Lakhani, Vikas Janu

JK Lakshmipat University Jaipur 302026 India

ARTICLE INFO

Keywords:
Image processing
Neural Network
Image Recognition
Big data

ABSTRACT

This is a brief report about the process used for the image recognition of any Olympic sports. First, there is a brief introduction to the report, followed by a background on big data and image recognition. Then, the methodology is explained that is used to complete this project and achieve the objective of this project. The objective of this report is to give a report on the project which is the recognition of any Olympic sport from any image using image processing.

1. Introduction

Sports recognition using image processing is a field that involves using computer algorithms to identify and understand motion-related images or videos. It is an important element of research and development and has received great attention in recent years due to its many uses.

The main idea of image recognition is to use appropriate information from pictures or videos, such as players' positions, balls, and other objects, and use this information to determine what is happening during the game. This may include tasks such as tracking players, finding the ball, and recognizing the sports.

Use a variety of computer vision, such as object detection, segmentation, and classification, to perform these tasks. These techniques involve visual analysis of images or videos and use them to identify and track items of interest. It has many applications such as sports recognition using functional images, sports advertising, player analysis and sports education. For example, it can be used to follow the movements of the players and the ball during football and to give detailed information about the performances of the players. It can also be used to help coaches and players identify areas of improvement and develop better training strategies.

Overall, the experience of recognizing sports using pictures is exciting and evolving rapidly with the potential to change the way we understand and analyse images.

2. Related Work

2.1. Big Data Engineering

Big Data Engineering is the practise of planning, constructing, evaluating, and maintaining systems that can manage significant amounts of data. The expansion in the volume of data produced by companies, people, and organisations has given rise to this field. The infrastructure needed to store, process, and analyse this data is built by big data engineers.

Many different technologies and techniques are included in the field of big data engineering, such as distributed computing systems, cloud computing, Hadoop, spark, and no-SQL databases. With the help of these technologies, enormous amounts of data can be processed, frequently in real-time, to produce insights that can be used to inform business choices.

Building scalable and dependable systems that can manage the large volume, diversity, and velocity of data that enterprises generate is the main objective of big data engineering. Data modelling, data integration, data storage, data processing, and data analysis must all be thoroughly understood to do this. Programming languages like Java, Python, and Scala as well as software engineering techniques like version control, testing, and debugging must all be mastered by big data engineers.

Big Data engineers are in high demand, and this trend is anticipated to continue in the coming years. The demand for knowledgeable Big Data Engineers will only rise as more and more firms rely on data-driven insights to make choices.

2.2. Image Processing

The alteration of images is the focus of the branch of digital signal processing known as image processing. In order to obtain valuable information from digital photos, it involves techniques for image enhancement, analysis, and interpretation. Image processing has a wide range of uses in industries like entertainment, engineering, security, and medical. Picture enhancement, picture restoration, image segmentation, and image identification are some of the frequent tasks carried out by image processing algorithms.

Image enhancement is the process of enhancing an image's visual quality, such as by reducing noise, sharpening the edges, or modifying the brightness and contrast. Contrarily, image restoration seeks to improve deteriorated images caused by noise, blur, or other elements.

In order to detect objects of interest, image segmentation involves separating an image into several areas. This method is frequently applied in medical imaging to find lesions or tumours. Automatically identifying objects or patterns inside a picture is known as image recognition. Examples of this include face recognition and object detection in self-driving automobiles.

MATLAB, Python, and C++ are just a few of the many computer languages that can be used to develop image processing methods. Additionally, there are numerous specialized software programmes and libraries for image processing, including ImageJ, OpenCV, and scikit-image.

ORCID(s):

In conclusion, the field of image processing is crucial to many facets of contemporary life. Its methodologies are constantly changing in line with developments in artificial intelligence and computation, and its uses span from surveillance and entertainment to medical diagnosis and treatment.

3. Olympic Sports Recognition

The process of recognising objects, persons, or other things in digital photos or movies is known as image recognition. In addition to security, medical imaging, driverless vehicles, and augmented reality, it is a significant application of computer vision and machine learning.

Following are the steps followed for image recognition process:

1. Data gathering and preparation: Gathering and getting ready the data is the initial phase. In order to do this, a sizable dataset of photos must be gathered, each of which contains examples of the objects or entities the system is supposed to be able to identify. To make sure the system can use the dataset to learn from it efficiently, it must be appropriately curated and labelled.

2. Feature extraction: The system extracts feature from the photos in this step that are important for recognition. A feature may be simple, like colour or texture, or it may be complicated, like shapes and patterns. The attributes chosen depend on the application in question and how difficult the recognition task is to complete.

3. Training: To understand the connections between the features and the recognised objects or entities, the system is taught using machine learning methods, such as deep neural networks. The system is fed labelled data during the training phase, and the model's parameters are changed to improve performance.

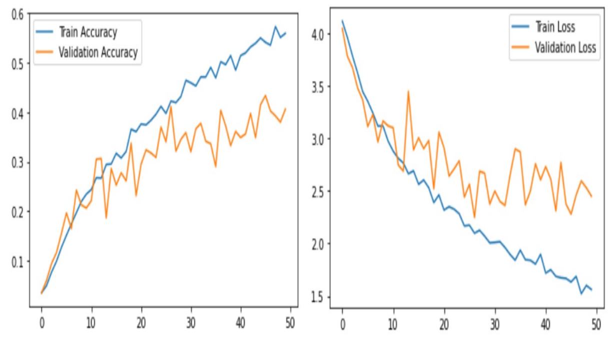
4. Testing and validation: Following training, the system is put to the test on a different collection of photos to gauge how well it performs. To determine the system's effectiveness, accuracy and precision are measured, and the findings are contrasted with the actual data.

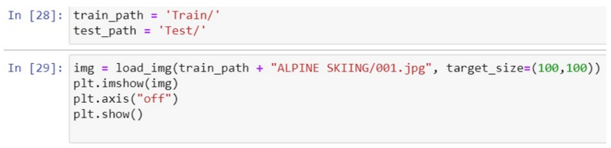
5. Deployment and refinement: The system can be implemented in the actual world following testing and validation. The system might need to be improved over time nevertheless as it runs across new problems and data. This entails tracking the system's performance over time, finding potential improvement areas, and updating the model and the data as necessary.

4. Evaluation and Results

The result of the model built in this project is given below:

As the number of epochs increase, accuracy of the model increases, and the loss is decreasing. Nevertheless, there is still only 60% accuracy, approximately, and there is still area for growth for the model. We can increase the number of inputs to the model to make it more precise and increase the number of epochs. Although, that would take longer of the model to train. This model took approximately 900s or 15





```
In [28]: train_path = 'Train/'
         test_path = 'Test/'

In [29]: img = load_img(train_path + "ALPINE SKIING/001.jpg", target_size=(100,100))
         plt.imshow(img)
         plt.axis("off")
         plt.show()
```

mins to train, and will take longer with more input data to train on and more epochs to run.

Appendix

A. About the Data

The type of data that is used is images. We will use the 100s of images from each of the sports played at Olympics to recognise which sports is which. The data is in images and the processing will be done on these images. The csv file contains the location of these images and the classification for them. We will train the model on the given data and then test that model for accuracy to recognise the sports.

B. Data Gathering

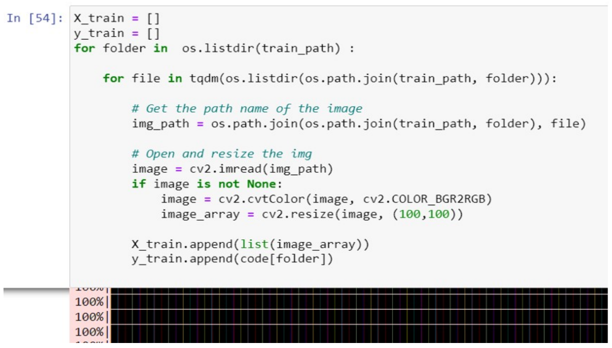
For building this model, the requirements are 100s of images of all the Olympic games that are played. For this dataset, we found 61 such games and have collected approximately 100 images for each of these games. The data collection was ethical and copyright free since the images were collected from free and no-copyright resources.

C. Data Preprocessing

The data is loaded for processing first. This is where we will start working in Jupyter notebook. We can view the data as follows to see if it is accessible:

After that, we can load by accessing the directory directly through the notebook with os library that is imported as follows:

Here, we have adjusted the size of all the images in order to make it easy for them to fit the model and given all



```
In [54]: X_train = []
         y_train = []
         for folder in  os.listdir(train_path) :

             for file in tqdm(os.listdir(os.path.join(train_path, folder))):

                 # Get the path name of the image
                 img_path = os.path.join(os.path.join(train_path, folder), file)

                 # Open and resize the img
                 image = cv2.imread(img_path)
                 if image is not None:
                     image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
                     image_array = cv2.resize(image, (100,100))

                 X_train.append(list(image_array))
                 y_train.append(code[folder])
```

```
100%|
100%|
100%|
```

the images the same size, 100x100, using the cv2.resize() function from the cv2 library. To confirm that the data has been imported:

```
In [55]: plt.figure(figsize=(20,20))
         for n , i in enumerate(list(np.random.randint(0,len(X_train),61))) :
             plt.subplot(8,8,n+1)
             plt.imshow(X_train[i])
             plt.axis('off')
             plt.title(getcode(y_train[i]))
```

D. Data Augnmentation

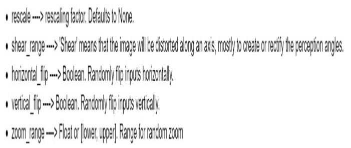
Data augmentation refers to the technique of artificially increasing the size of a training dataset by applying various transformations to the existing data. The goal of data augmentation is to improve the performance of a machine learning model by increasing the diversity of the training data, which helps the model to generalize better to new, unseen data. Some common features are:

In addition to image data, data augmentation can also be applied to other types of data, such as text or audio data. For example, in natural language processing (NLP), data augmentation techniques can include paraphrasing or altering the order of sentences in a document. By increasing the size and diversity of the training data, data augmentation can help to prevent overfitting and improve the robustness and generalization ability of the model. In this model, we have used ImageDataGenerator to apply augmentation with few of the above mentioned features.

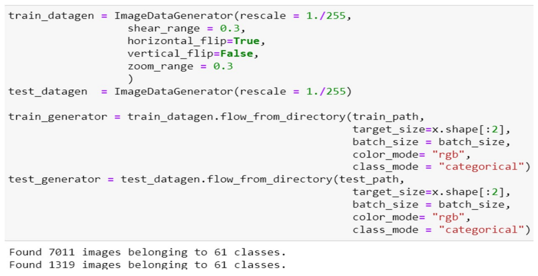
E. Model Building

The model used for this project is one the most common models used for images recognition, CNN (Convolutional Neural Network).

Convolutional Neural Networks (CNNs) are a type of artificial neural network that are commonly used for image recognition and computer vision tasks. CNNs are designed to automatically learn spatial hierarchies of features from the input data, allowing them to capture complex patterns and structures that are difficult to identify using traditional machine learning techniques.



- rescale ----> rescaling factor. Defaults to None.
- shear_range ----> 'Shear' means that the image will be distorted along an axis, mostly to create or rectify the perception angles.
- horizontal_flip ----> Boolean. Randomly flip inputs horizontally.
- vertical_flip ----> Boolean. Randomly flip inputs vertically.
- zoom_range ----> Float or [lower, upper]. Range for random zoom



```
train_datagen = ImageDataGenerator(rescale = 1./255,
                  shear_range = 0.3,
                  horizontal_flip=True,
                  vertical_flip=False,
                  zoom_range = 0.3
                  )
test_datagen  = ImageDataGenerator(rescale = 1./255)

train_generator = train_datagen.flow_from_directory(train_path,
                                                     target_size=x.shape[:2],
                                                     batch_size = batch_size,
                                                     color_mode= "rgb",
                                                     class_mode = "categorical")
test_generator = test_datagen.flow_from_directory(test_path,
                                                     target_size=x.shape[:2],
                                                     batch_size = batch_size,
                                                     color_mode= "rgb",
                                                     class_mode = "categorical")
Found 7011 images belonging to 61 classes.
Found 1319 images belonging to 61 classes.
```

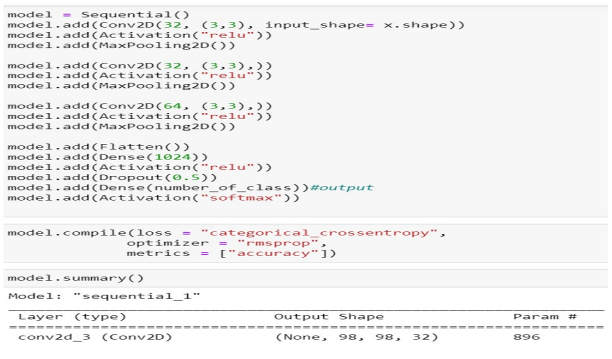
Working on CNN

The basic building block of a CNN is the convolutional layer. In this layer, a set of filters (also known as kernels) are applied to the input image, with each filter learning to detect a specific feature or pattern. The filters are typically small in size (e.g. 3x3 or 5x5), and they slide over the input image one pixel at a time, producing a 2D activation map (also known as a feature map) that highlights the regions of the input image that correspond to the learned feature.

The output of a convolutional layer is then passed through a non-linear activation function, such as ReLU (Rectified Linear Unit), which introduces non-linearity into the network and allows it to capture more complex patterns. The resulting feature maps are then fed into a pooling layer, which reduces the spatial size of the feature maps by taking the maximum or average value over small regions (e.g. 2x2) in each feature map. This pooling operation helps to make the network more robust to small variations in the input data, and reduces the number of parameters in the network, making it computationally more efficient.

The output of the pooling layer is then passed through another set of convolutional and pooling layers, creating a hierarchy of feature maps that capture increasingly complex patterns and structures in the input data. Finally, the output of the last convolutional layer is flattened into a 1D vector and passed through one or more fully connected layers, which perform a non-linear transformation of the features and produce the final output of the network.

During training, the parameters (i.e. weights and biases) of the network are learned using backpropagation, a gradient-based optimization algorithm that adjusts the parameters to minimize the difference between the network's output and the desired output (e.g. a class label for an image). The learning process involves iteratively feeding training examples through the network, computing the error between the predicted output and the true output, and updating the parameters of the network using the gradients of the error with respect to the parameters.

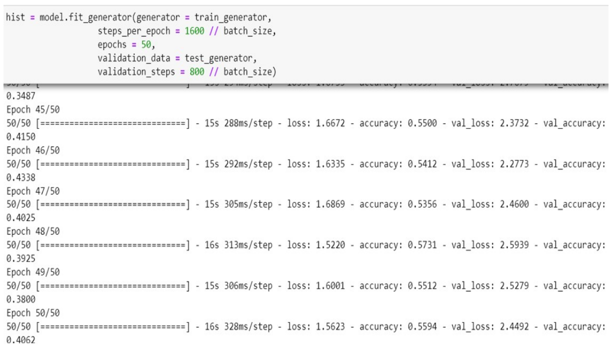


```
model = Sequential()
model.add(Conv2D(32, (3,3), input_shape= x.shape))
model.add(Activation("relu"))
model.add(MaxPooling2D())

model.add(Conv2D(32, (3,3),))
model.add(Activation("relu"))
model.add(MaxPooling2D())

model.add(Conv2D(64, (3,3),))
model.add(Activation("relu"))
model.add(MaxPooling2D())

model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(Dropout(0.5))
model.add(Dense(number_of_class))#output
model.add(Activation("softmax"))
```

```
model.compile(loss = "categorical_crossentropy",
              optimizer = "rmsprop",
              metrics = ["accuracy"])
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 98, 98, 32)        896
```



```
hist = model.fit_generator(generator = train_generator,
                   steps_per_epoch = 1600 // batch_size,
                   epochs = 50,
                   validation_data = test_generator,
                   validation_steps = 800 // batch_size)
```

```
0.3487
Epoch 45/50
50/50 [==============================] - 15s 288ms/step - loss: 1.6672 - accuracy: 0.5500 - val_loss: 2.3732 - val_accuracy:
0.4150
Epoch 46/50
50/50 [==============================] - 15s 292ms/step - loss: 1.6335 - accuracy: 0.5412 - val_loss: 2.2773 - val_accuracy:
0.4338
Epoch 47/50
50/50 [==============================] - 15s 305ms/step - loss: 1.6869 - accuracy: 0.5356 - val_loss: 2.4600 - val_accuracy:
0.4025
Epoch 48/50
50/50 [==============================] - 16s 313ms/step - loss: 1.5220 - accuracy: 0.5731 - val_loss: 2.5939 - val_accuracy:
0.3925
Epoch 49/50
50/50 [==============================] - 15s 306ms/step - loss: 1.6001 - accuracy: 0.5512 - val_loss: 2.5279 - val_accuracy:
0.3800
Epoch 50/50
50/50 [==============================] - 16s 328ms/step - loss: 1.5623 - accuracy: 0.5594 - val_loss: 2.4492 - val_accuracy:
0.4062
```

E.1. Model Building in Notebook

First we will build the model and then fit the generated data in it.

Model training:

F. Source Code

https://github.com/chitra-sl/Big-data-Project

G. IPR Certificate

We, Chitra S. Lakhani and Vikas Janu, with this certify that the project work submitted by us, entitled Olympic Sports Recognition, to our supervisors, Dr. Alok Kumar and Dr. Utsav Upadhyay, in partial fulfilment of the requirements for the course is a Bonafede work carried out by us and has not been previously submitted to any other course. We further certify that no part of this work shall be published, reproduced, or distributed in any form without the prior permission of our supervisors. We understand that any such unauthorized use of the project work may be considered a violation of academic ethics and result in severe penalties. We also affirm that the project work has been carried out under the ethical standards and guidelines set forth by our supervisors. We acknowledge that our supervisor has the right to make any modifications or revisions to the project work that may be deemed necessary. We also agree to abide by any additional terms and conditions as stipulated by our supervisor.

Date: 20th May, 2023

Chitra Lakhani
2020BTECHCSE024
JK Lakshmipat University, Jaipur

Vikas Janu
2020BTECHCSE083
JK Lakshmipat University, Jaipur