Artificial Intelligence with MATLAB

Introduction

The objective of this blog is to explain how MATLAB can be used in artificial intelligence applications by taking the example of creating an AI model for **automatic no ball detection in cricket using MATLAB.** This can be done with a prior knowledge of deep learning, computer vision , and neural networks.

Deep Learning

Deep learning is a AI function that imitates the working of the human brain in processing data and decision making. Deep learning again consists of neural networks. The two types of neural networks are **artificial neural networks and convolution neural networks.** Convolution neural networks are used for image data processing.

Computer Vision

Computer vision is the function of artificial intelligence that trains the computer to visualise and interpret the surrounding world. Images captured from cameras can be interpreted and classified accurately by the computer due to computer vision. Computer vision is important in many AI applications like face recognition and text recognition.

Need for automatic no ball detection

Nowadays during cricket matches no ball is declared by capturing the position of the front foot of the bowler and then it is manually analysed by the third umpire. When the third umpire finds that the front foot of the bowler is not within the crease he will signal it as no ball by manually using a buzzer or siren.

Though the system is quite effective and it may not seem highly necessary it has a few benefits and can be considered. The declaration of no ball can be done faster with this technology. It also reduces the probability of human mistakes. This can be considered as a step towards automating third umpire decisions in the future of cricket.

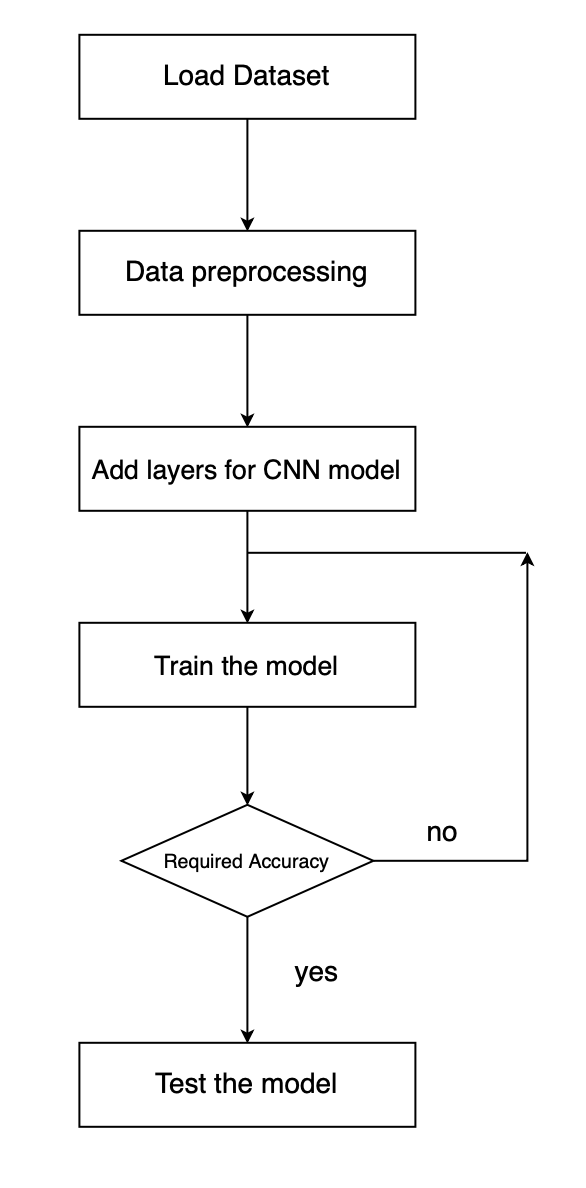
The AI model will analyse the image of the bowlers foot position and declare no ball or fair delivery . It can give automatic output by a buzzer sound, display, etc.

AI model for automatic no ball detection

We need to create a classification type convolution neural network since there are only two categories no ball and fair delivery. The data is binary not continuous.

For this we will need MATLAB deep learning toolbox.

We can create the model by writing the code manually or using deep network designer create a neural network using the graphical user interface and then generate a automatic code.

Algorithm

MATLAB Code

%% Create and Train a Deep Learning Model

% Script for creating and training a deep learning network with the following

% properties:

%%

%

% Number of layers: 7

% Number of connections: 6

% Training setup file: /Users/apurvaaddula/Desktop/trainingSetup\_2020\_11\_02\_\_19\_08\_14.mat

%

%%

% Run this script to create the network layers, import training and validation

% data, and train the network. The network layers are stored in the workspace

% variable |layers|. The trained network is stored in the workspace variable |net|.

%

% To learn more, see <matlab:helpview('deeplearning','generate\_matlab\_code')

% Generate MATLAB Code From Deep Network Designer>.

%

% Auto-generated by MATLAB on 02-Nov-2020 19:08:17

%% Load Initial Parameters

% Load parameters for network initialization. For transfer learning, the network

% initialization parameters are the parameters of the initial pretrained network.

trainingSetup = load("/Users/apurvaaddula/Desktop/trainingSetup\_2020\_11\_02\_\_19\_08\_14.mat");

%% Import Data

% Import training and validation data.

imdsTrain = imageDatastore("/Users/apurvaaddula/Desktop/DATASET","IncludeSubfolders",true,"LabelSource","foldernames");

[imdsTrain, imdsValidation] = splitEachLabel(imdsTrain,0.7,"randomized");

%% Augmentation Settings

imageAugmenter = imageDataAugmenter(...

"RandRotation",[-360 360],...

"RandScale",[1 2],...

"RandXReflection",true);

% Resize the images to match the network input layer.

augimdsTrain = augmentedImageDatastore([28 28 3],imdsTrain,"DataAugmentation",imageAugmenter);

augimdsValidation = augmentedImageDatastore([28 28 3],imdsValidation);

%% Set Training Options

% Specify options to use when training.

opts = trainingOptions("sgdm",...

"ExecutionEnvironment","auto",...

"InitialLearnRate",0.01,...

"MaxEpochs",5,...

"Shuffle","every-epoch",...

"Plots","training-progress",...

"ValidationData",augimdsValidation);

%% Create Array of Layers

layers = [

imageInputLayer([28 28 3],"Name","imageinput")

convolution2dLayer([3 3],32,"Name","conv","Padding","same")

batchNormalizationLayer("Name","batchnorm")

reluLayer("Name","relu")

fullyConnectedLayer(2,"Name","fc")

softmaxLayer("Name","softmax")

classificationLayer("Name","classoutput")];

%% Train Network

% Train the network using the specified options and training data.

[net, traininfo] = trainNetwork(augimdsTrain,layers,opts);

References:

<https://lms.matlabhelper.com/blog/red-blood-cell-counter-using-image-segmentation/>

<https://www.investopedia.com/terms/d/deep-learning.asp>

<https://in.mathworks.com/help/deeplearning/ug/create-simple-deep-learning-network-for-classification.html>

<https://www.sas.com/en_us/insights/analytics/computer-vision.html>

<https://www.mathworks.com/help/deeplearning/gs/create-simple-image-classification-network-using-deep-network-designer.html>

Image source:

<http://idlesummers.com/post.php?postid=1944>