UMKC CS5542

**Team 3**

**Zeshan Ather, John Goza, Hongkun Jin, Mouqing Jin**

**Increment 2 Report and Software Overview**

**Framework Specification**

* **Significance**

By leveraging competing software models against each other we believe we can significantly enhance correct software prediction of the desired output, the score of the basketball game. we can summarize the long basketball video into short highlight video and that will attract the more audience and will be able to get more views ratings. Our aim is also to predict some analytics for audience about winning team, goal ratios etc.

* **Features:**

The feature vector that we used for this project is

1. Dunks
2. Free throws
3. Tapping
4. Shooting

**Approach**

* **Data Sources**

Our data source is different high deficiency basketball videos. We collected some videos and then generated frames from each video. We also collected some long videos trim that video to extract specific data frames for the training of our model





* **Data Specification:** Amount, Format, Training/Testing data, etc

We are using 4 categories of the data which are dunks, free throws, tapping and shooting. Each category contains 50 main frame images.

**For Shallow Learning** we are splitting our data into 0.7 and 0.3 for training and testing respectively.

**For Deep Learning** we are first converting .png image dataset into MNIST data format which are gray scale images. The common parameters for the model are

1. MNIST validation size is 40
2. Epochs: 500



* **Analytic Algorithms/Platform**
  + **Clustering**

In shallow learning we use K-Means clustering. The cluster size is 400 and number of iterations we used are 20

* + **Regression/Classification**

**In TensorFlow linear regression** is applied. We used MNIST converted format dataset. The paraments for model are

Learning rate 0.01 Batch size 28 Epoch 500

We applied **softmax regression** model with same dataset format

* + **Deep Learning**

**In TensorFlow** we applied the convoluted neural network model. For convoluted neural network model, we used 3 layers of filtering of 32,64 and 1024. We used the Adam optimizer for our CNN model.

* + **Shallow Learning Model**

In shallow learning we applied **Naïve base, random forest and decision trees**. For training and testing purposes we split our data in to 70% and 30% ratio respectively.

* + **Annotation**

We use the clarifai API to annotate our images

* **Expected Inputs/Outputs**

Our Input will be the long basketball video. We will separate the frames from our videos. We will train our image classification models with the extracted frames. Then find the main frames on the basis different actions and hot spot areas.  
Our output will be a summarized video with the highlights of the match.

**Related Work**

* Open Source Project

<https://github.com/gskielian/JPG-PNG-to-MNIST-NN-Format>

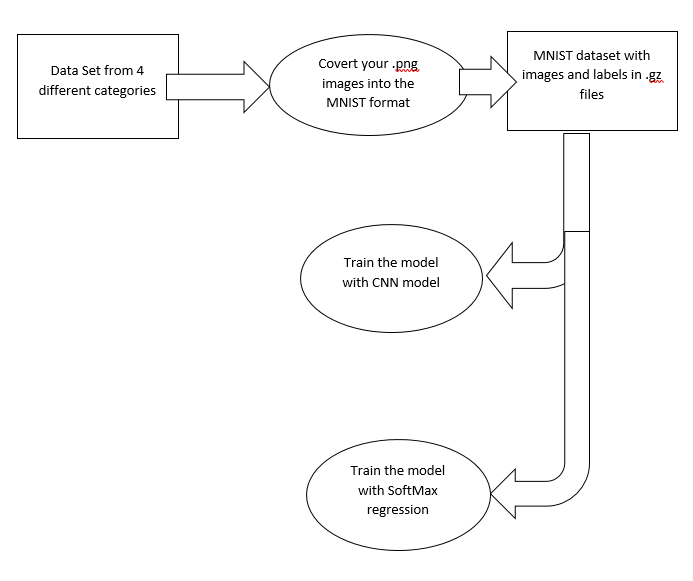
this helped us in reading our data in TensorFlow

**Application Specification & Implementation**

* System Specification (Big Data Analytics Server/Client)
  + Software Architecture

We used the TensorFlow model in PyCharm and got the accuracy curves in the Tensorbaord

* + Features, workflow, technologies
    - Activity Diagram (workflow, data, task)



* + - **Feature Specification**

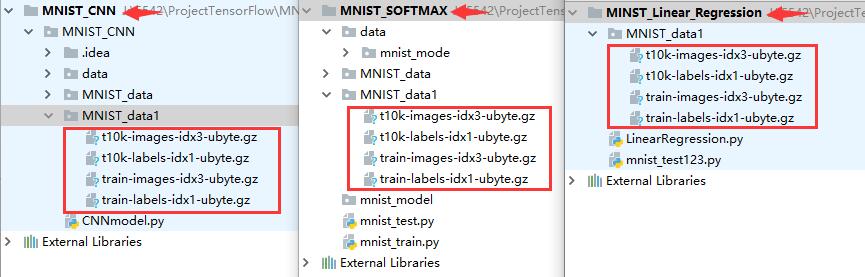
The feature vector set that we are handling in the project up till now are dribbling, dunks, tapping and free throws

**Operation Specification (Input/output, exceptions)**

**For the Tensor flow:**

Training and test in TensorFlow

Copy the MNIST data to each of TensorFlow approaches folder.



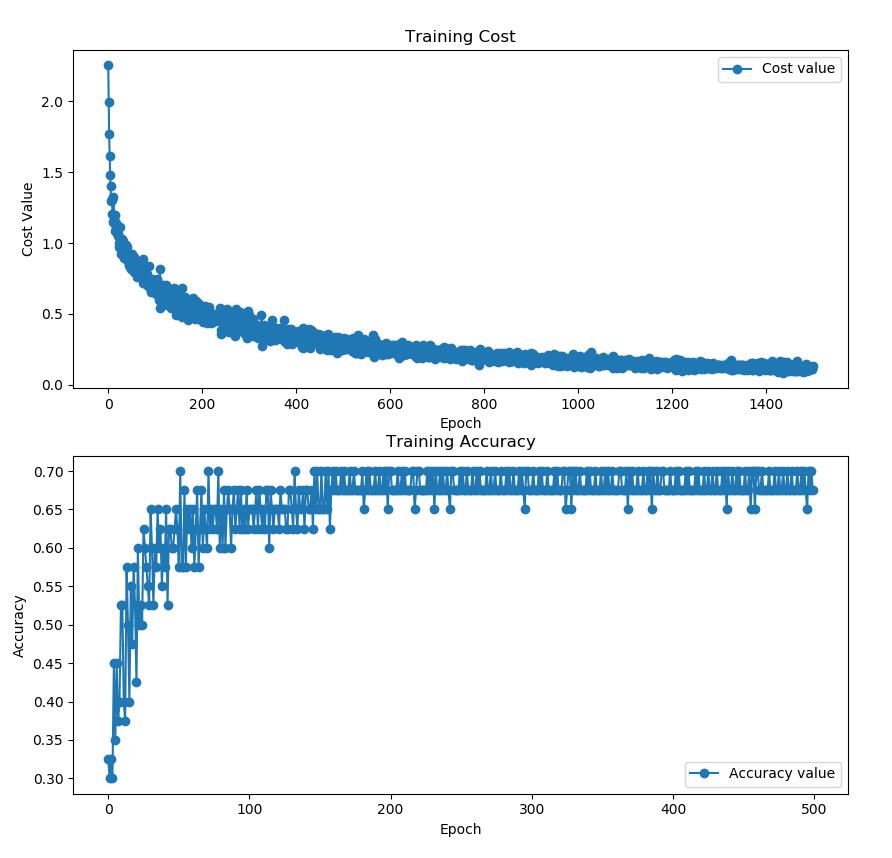
**CopyMnistData**

The common parameters for each method.

1. MNIST validation size is 40
2. Epochs: 500
   1. **Linear Regression**

2.1.1 Parameters setting:





**LR\_TraingCost\_Accuracy**

1. Running Time: 33.87053728103638 seconds
2. Accuracy: 0.675
   1. **SoftMax**

Parameters setting: None

There are two .py files to build model and test. We combine them in order to calculate the time the running cost.

Results

1. Running Time: 2.1413497924804688 seconds
2. Accuracy: 0.8888889
   1. **CNN**

**Parameters setting**:

(1) weight\_variable and bias\_variable are 0.1 and 0.1

(2) The Optimizer we use is: AdamOptimizer(1e-4)

(3) Filter size is: 32-64-1024

**Results**

1. Running Time: 64.0976209640503 seconds
2. Accuracy: 0.888889

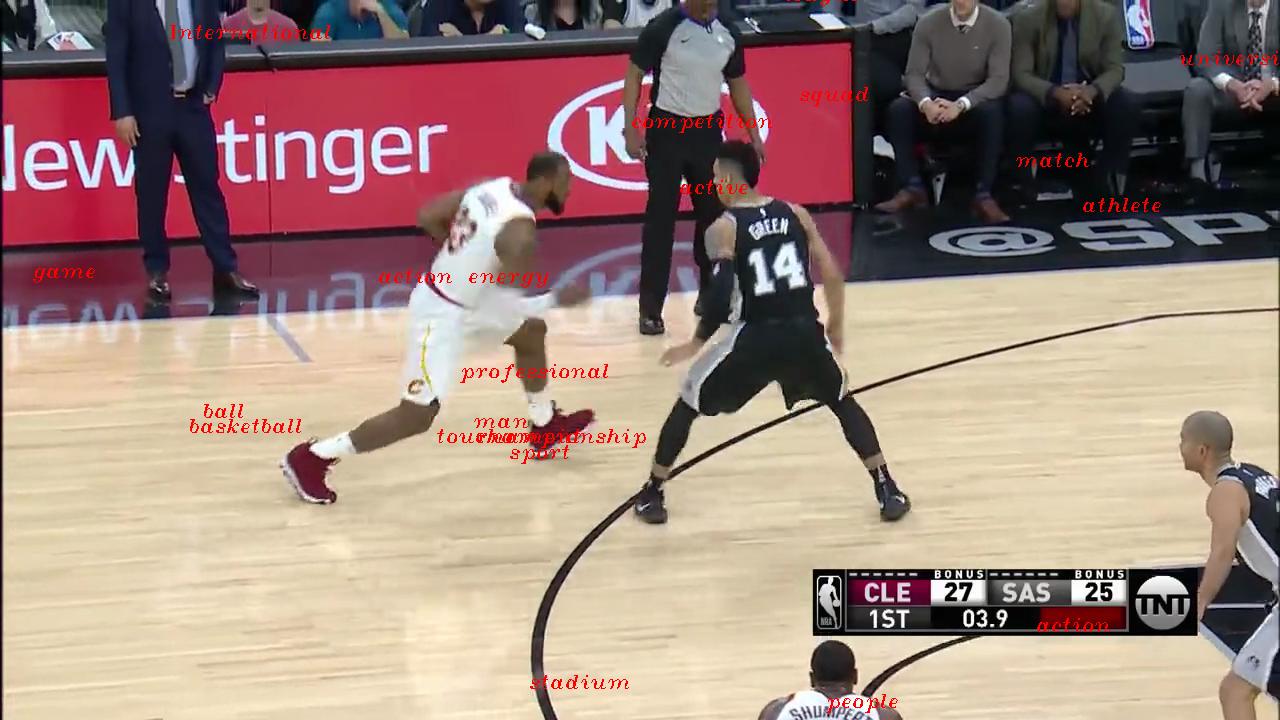
**Changing epochs**

We can see the results; the training accuracy always is 1 after step100. So, we changed epoch to 100 and see what happen.

1. Running Time: 15.982191562652588 seconds
2. Accuracy: 0.833333

**Annotation**

We annotated the images using the calarfai API’s



**Accuracy Comparison Chart**

|  |  |
| --- | --- |
| **Shallow Learning** | |
| Naïve Bayes | 33% |
| Decision Tree | 89% |
| Random Fores | 93% |
| **TensorFlow** | |
| CNN | 94% |
| Linear Regression | 67% |
| SoftMax Regression | 89% |

**Project Management**

1. **Plan & Project Timelines, Members, Task Responsibility**
   * **Implementation status report** 
     + **Work completed:**

We have completed the shallow learning classification using 3 different models **Naïve base, random forest and decision trees**

We have completed the regression classification in TensorFlow

We have implemented the deep learning convoluted neural network model for the classification of our images

We have used the clarafai APIs for the annotation of the images

* + - **Responsibility/Work Division**

Tasks is divided among the 2 groups

First group 3-1 is handling the data collection, shallow leaning models random forest, naïve base, TensorFlow softmax regression, claraifi API and documentation of the work

Second group is handling the deep learning CNN model, linear regression, shallow learning decision tree, data collection, and documentation of the work.

Task is distributed equally so everyone is actively participating.

* + **Work to be completed**
    - **Description**

In our future work we will work on  
Implementation of Web/Mobile app development

More deep learning model’s implementation with our data

Spark API implementation with android app

* + - Responsibility (Task)

We are working on how the task will be distributed among the group

* + **Issues/Concerns**

Migration of the datasets might be an issue

As we are training the model with gray scale images so identifying the different teams might be a challenge