**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**Jnana Sangama, Belagavi - 590018**



PROJECT ENTITLED

**“Virtual Musical Instruments using Neural Networks”**

Submitted in partial fulfilment of the requirements for the award of degree of

**BACHELOR OF ENGINEERING**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

For the Academic year 2017 - 2018

Submitted by:

|  |  |
| --- | --- |
| **Chandra Mahendra Vikram Singh** | **1MV14CS027** |
| **Himanshu Kumar** | **1MV14CS040** |
| **Jubin George Mathew** | **1MV14CS046** |
| **K Vishnudev** | **1MV14CS049** |

Project carried out at

**Sir M. Visvesvaraya Institute of Technology**

**Bangalore - 562157**

Under the guidance of

**Mrs. Sheela S Kathavate**

Assistant Professor, Dept of CSE

Sir M Visvesvaraya Institute of Technology, Bangalore



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**SIR M VISVESVARAYA INSTITUTE OF TECHNOLOGY**

HUNASAMARANAHALLI, BANGALORE - 562157

**SIR M VISVESVARAYA INSTITUTE OF TECHNOLOGY**

Krishnadeveraya Nagar, International Airport Road,   
Hunasamaranahalli, Bengaluru – 562157

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

****

**CERTIFICATE**

It is certified that the project work entitled "**Virtual Musical Instruments using Neural Networks**" is carried out **by Chandra Mahendra Vikram Singh (1MV14CS027), Himanshu Kumar (1MV14CS040), Jubin George Mathew (1MV14CS046), K Vishnudev (1MV14CS049)** bonafide student of **Sir M Visvesvaraya Institute of Technology** in partial fulfilment for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering of the **Visvesvaraya Technological University**, Belagavi during the year 2017-2018. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the course of Bachelor of Engineering.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Mrs. Sheela S Kathavate  Internal Guide  Assistant Prof., Dept of CSE  Sir MVIT | **Prof. Dilip K. Sen**  HoD  Dept of CSE  Sir MVIT | **Dr. V.R. Manjunath**  Principal  Sir MVIT |

**Name of the examiners                                                                     Signature with date**

1)

2)

**DECLARATION**

We hereby declare that the entire project work embodied in this dissertation has been carried out by us and no part has been submitted for any degree or diploma of any institution previously.

Place: Bangalore

Date:

Signature of Students:

|  |  |
| --- | --- |
| Chandra Mahendra Vikram Singh (1MV14CS027) | Himanshu Kumar  (1MV14CS040) |
| Jubin George Mathew  (1MV14CS046) | K Vishnudev  (1MV14CS049) |

**ACKNOWLEDGMENT**

It gives us immense pleasure to express our sincere gratitude to the management of **Sir M. Visvesvaraya Institute of Technology,** Bangalore for providing the opportunity and the resources to accomplish our project work in their premises.

On the path of learning, the presence of an experienced guide is indispensable and we would like to thank our guide **Mrs Sheela S Kathavate**, Assistant Professor, Dept. of CSE, for his invaluable help and guidance.

We would also like to convey our heartfelt and sincere thanks to **Dr. V.R. Manjunath,** Principal, Sir. MVIT for providing us with the infrastructure and facilities needed to develop our project.

We would also like to thank the staff of Department of Computer Science and Engineering and lab-in-charges for their co-operation and suggestions. Finally, we would like to thank all our friends for their help and suggestions without which completing this project would not have been possible.

|  |  |
| --- | --- |
| **Chandra Mahendra Vikram Singh** | 1MV14CS027 |
| **Himanshu Kumar** | 1MV14CS040 |
| **Jubin George Mathew** | 1MV14CS046 |
| **K Vishnudev** | 1MV14CS049 |

**ABSTRACT**

Music is an important entity in everyone’s life. Similarly, musical instruments have a paramount place in Music. The art of playing musical instruments have been traditionally carried on from generation to generation, from primitive stage to this modern age. As Music is a performing art, which is being creative and cannot be static, hence the gradual developments and experiments have always given new ideas to the modern generation.

This project highlights experimentation in Instrumental Music which led to digitalised instruments in this sector. Electronic Instruments are instruments which help to electrically produce a symphony of its own may it be an electric guitar in a rock concert or a keyboard in orchestra. This project introduces an end-to-end neural network model for playing the sound of a musical instrument based on a silent video of it being played. At a high level, the model consists of a convolutional neural network (CNN) to extract features from the raw video frames and neural autoregressive models to encode the spatiotemporal features of the video.

**Table of Contents**

[**INTRODUCTION** 1](#_Toc513714135)

[1.1 Scope Of Work 3](#_Toc513714136)

[1.2 Motivation 3](#_Toc513714137)

[**LITERATURE SURVEY** 4](#_Toc513714138)

[2.1 Convolutional Neural Networks 5](#_Toc513714139)

[2.2 Supervised Learning 7](#_Toc513714140)

[2.3 Data Acqusition And Augmentation 7](#_Toc513714141)

[2.4 Keras And Tensorflow 9](#_Toc513714142)

[**SYSTEM REQUIREMENTS**](#_Toc513714143) [**AND** **SPECIFICATIONS** 11](#_Toc513714144)

[3.1 Functional Requirements 12](#_Toc513714145)

[3.2 Non-Functional Requirements 12](#_Toc513714146)

[3.3 Other Non-Functional Requirements 13](#_Toc513714147)

[3.3.1 Safety And Security Requirements 13](#_Toc513714148)

[3.3.2 Software Quality Attributes 13](#_Toc513714149)

[3.4 Deveopment Requirements 14](#_Toc513714150)

[**DATASET AND FEATURES** 16](#_Toc513714151)

[4.1 Dataset 17](#_Toc513714152)

[4.2 Features 17](#_Toc513714153)

[**SYSTEM ANALYSIS & DESIGN** 19](#_Toc513714154)

[5.1 System Analysis 20](#_Toc513714155)

[5.1.1 System Development Life Cycle 20](#_Toc513714156)

[5.1.2 The Development Phases In Brief 21](#_Toc513714157)

[5.2 System Design 22](#_Toc513714158)

[5.2.2 High Level Design 23](#_Toc513714159)

[**IMPLEMENTATION** 24](#_Toc513714160)

[7.1 Implementation 25](#_Toc513714161)

[7.1.1 Model 25](#_Toc513714162)

[7.1.2 Real-Time Image Capture 26](#_Toc513714163)

[7.1.3 Training And Saving The Model 26](#_Toc513714164)

[7.1.4 Imports And Hyper Parameters 27](#_Toc513714165)

[7.1.5 Loading The Model And Prediction 28](#_Toc513714166)

[7.1.6 Model Upload And Download 29](#_Toc513714167)

[7.2 Execution Phase 29](#_Toc513714168)

[7.2.1 Imports And Constants 29](#_Toc513714169)

[7.2.2 User Interface 30](#_Toc513714170)

[**EXECUTION AND TEST CASES** 31](#_Toc513714171)

[8.1 Execution 32](#_Toc513714172)

[8.1.1 Test Runs 33](#_Toc513714173)

[8.1.2 Observations And Graphs 34](#_Toc513714174)

[**CONCLUSION AND FUTURE ENHANCEMENTS** 35](#_Toc513714175)

[9.1. Conclusion 36](#_Toc513714176)

[9.2 Future Work 36](#_Toc513714177)

[9.3 Project Activity 36](#_Toc513714178)

[**REFERENCES** 38](#_Toc513714179)

**TABLE OF FIGURES**

[Figure 1: CNN Layers 6](#_Toc513714192)

[Figure 2: Neural Network 6](#_Toc513714193)

[Figure 3: A Sample Convolutional Neural Network 7](#_Toc513714194)

[Figure 4: Gaussian noise 8](#_Toc513714195)

[Figure 5: Lighting Property Changes 9](#_Toc513714196)

[Figure 6: Tensorflow Backend 10](#_Toc513714197)

[Figure 7: Keys Layout 18](#_Toc513714198)

[Figure 8: Overview of the System 18](#_Toc513714199)

[Figure 9: Software Development Life Cycle 20](#_Toc513714200)

[Figure 10: Custom CNN Model 22](#_Toc513714201)

[Figure 11: High Level System Architecture 23](file:///E:\BE_Report_FYP.docx#_Toc513714202)

[Figure 12: User Interface in Tkinter of the Paper Piano layout 30](file:///E:\BE_Report_FYP.docx#_Toc513714203)

**CHAPTER 1**

# **INTRODUCTION**

**CHAPTER 1**

**INTRODUCTION**

Nowadays, machine learning resides in every domain of the industry like online advertising, recommendation system, search engine, machine locomotive, pattern recognition etc. Also due to its pronounced effectiveness its applicability keeps growing bigger. Machine learning is an application of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

In machine learning, a concept called convolutional neural network (CNN, or ConvNet) is used extensively in image and video recognition. It is a class of deep, feed-forward artificial neural networks that has successfully been applied to analysing visual imagery. CNNs use a variation of multilayer perceptron’s designed to require minimal pre-processing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage. They have applications in image and video recognition, recommender systems and natural language processing.

This project redefines the way one uses music instruments to generate sound to compose music. A keyboard instrument is a musical instrument played using a keyboard, a row of levers which are pressed by the fingers. The most common of these are the piano, organ, and various electronic keyboards, including synthesizers and digital pianos.

## 1.1 SCOPE OF WORK

Scope of the project is to develop and implement a working model to detect the movement of hands and play music accordingly. The focus is to develop a cost effective and a portable alternative to existing musical instruments that are expensive and bulky. The setup includes an optical device to capture video for detection of motion. This project is limited only to keyboard instruments and other instruments are out of scope of this project. Accuracy of the model, quality of capturing device and nondeterministic variation in processing time may widely affect the performance of this model. The project is expected to be completed within a period 4 - 6 months. This time span will involve building a custom hardware package from existing embedded system to give an independent kit to practice music.

## 1.2 MOTIVATION

Recent years have marked a sharp increase in the number of ways in which people interact with electronic devices. Keyboards and computer mouse devices are not the only way users interact with their computers. This project aims to investigate and demonstrate a human-computer interaction application in which a user can interact with the layout of a piano on a sheet of paper using computer vision. The image is acquired using a general-purpose camera from a single direction. The work presents detail about methods of piano key detection and labelling, hand movement detection, fingertip detection and tracking, and touch interaction. The detected key press then triggers a sound. The paper covers the details of the development of the virtual piano from concept to realization.

**CHAPTER 2**

# **LITERATURE SURVEY**

**CHAPTER 2**

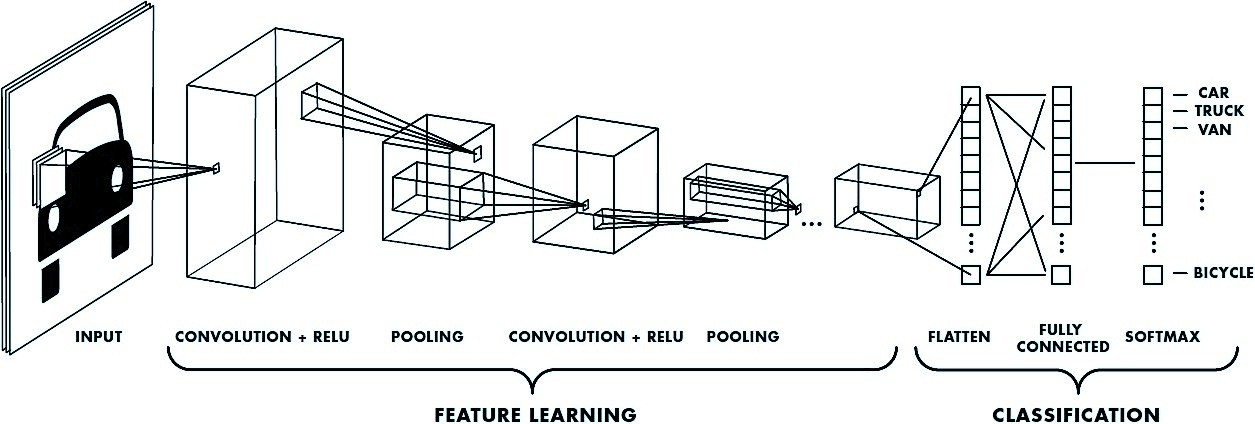
**LITERATURE SURVEY**

There are wide range of instruments for music generation, they are either expensive or colossal to carry and sometimes both. So, this model tries to develop a system to mimic multiple instruments or input devices using the same environment in a more economic manner. Different Input Output devices such as keyboard, mouse or gesture could be used as input for electronic gadgets. They also suffer the same set of limitations as musical instruments. Thus to overcome such limitations, we have an elegant solution which involves building a generic learning model which can be trained for any specific device and the same can be achieved by passing the input received from these devices to a Convolutional neural network.

## 2.1 CONVOLUTIONAL NEURAL NETWORKS

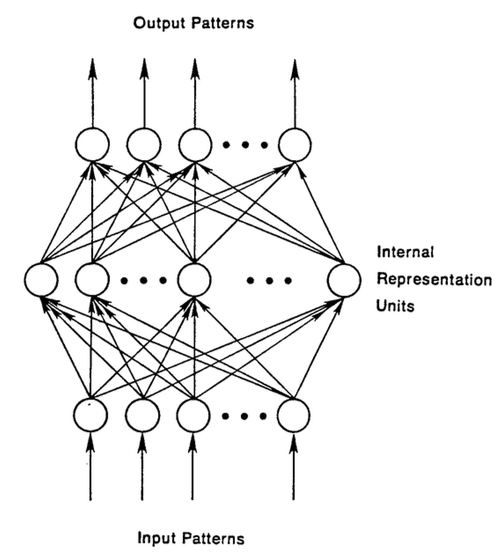
Convolutional Neural Networks (CNN, or ConvNet) are a class of deep-forward artificial neural networks that have been used for image analysis. These networks use convolution layers in its core. The images are broken down into smaller units, which are a set of learnable filters called kernels. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.



**Figure 1: CNN Layers**

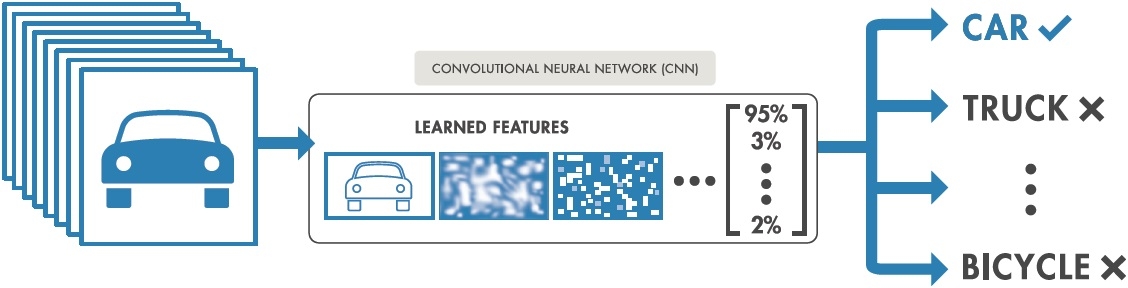
Recurrent neural networks (RNN) [2] were created in the 1980s but have recently gained popularity with advances to network designs and increased computational power of graphics processing units. They are particularly useful with sequential data because each neuron or unit can use internal memory to maintain information about previous inputs.



**Figure 2: Neural Network**

## 2.2 SUPERVISED LEARNING

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way (see inductive bias).



**Figure 3: A Sample Convolutional Neural Network**

## 2.3 DATA ACQUSITION AND AUGMENTATION

Deep learning uses Neural Nets with a lot of hidden layers (dozens in today’s state of the art), and requires large amounts of training data. These models have been particularly effective in gaining insight and approaching human level accuracy in perceptual tasks like vision, speech, language processing. The theory and mathematical foundations were laid several decades ago.

Primarily two phenomena have contributed to the rise of machine learning:

a) Availability of huge data-sets/training examples in multiple domains

b) Advances in raw compute power and the rise of efficient parallel hardware.

Some basic but powerful data augmentation techniques include the following:

* **Gaussian noise**

Over-fitting usually happens when your neural network tries to learn high frequency features (patterns that occur a lot) that may not be useful. Gaussian noise, which has zero mean, essentially has data points in all frequencies, effectively distorting the high frequency features. This also means that lower frequency components (usually, your intended data) are also distorted, but your neural network can learn to look past that. Adding just the right amount of noise can enhance the learning capability.



**Figure 4: Gaussian noise**

* **Lighting conditions**

Changing lighting conditions using programming technique includes conversion of RGB images to HSV. The HSV model describes colour similar to the colours perceived by human eye. RGB defines colour in terms of combination of primary colours. In situations where colour illustration plays an integral role, the HSV colour model is often preferred over the RGB model. 'Hue' represents the colour, 'Saturation' represents the amount to which that respective colour is mixed with white and 'Value' represents the amount to which that respective colour is mixed with black (Grey level).

In RGB, we cannot separate colour information from luminance. HSV or Hue Saturation Value is used to separate image luminance from colour information.

Different lighting conditions can be emulated by tweaking the HSV values.



**Figure 5: Lighting Property Changes**

## 2.4 KERAS AND TENSORFLOW

**Keras** is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Features of Keras are:

* Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
* Supports both convolutional networks and recurrent networks, as well as combinations of the two.
* Runs seamlessly on CPU and GPU.

**TensorFlow** is an open source software library for machine learning across a range of tasks, and developed by Google to meet their needs for systems capable of building and training neural networks to detect and decipher patterns and correlations, analogous to the learning and reasoning which humans use. It is currently used for both research and production at Google products, ‍ often replacing the role of its closed-source predecessor, DistBelief. TensorFlow was originally developed by the Google Brain team for internal Google use before being released under the Apache 2.0 open source license on November 9, 2015.

TensorFlow is Google Brain's second generation machine learning system, released as open source software on November 9, 2015. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, MacOS, Windows, and mobile computing platforms including Android and iOS.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations which such neural networks perform on multidimensional data arrays. These multidimensional arrays are referred to as "tensors". In June 2016, Google's Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

TensorFlow provides a Python API, as well as C++, Haskell, Java, Go, and Rust APIs. In addition, there is a 3rd party package for R.



**Figure 6: Tensorflow Backend**

**CHAPTER 3**

# **SYSTEM REQUIREMENTS**

# **AND SPECIFICATIONS**

**CHAPTER 3**

**SYSTEM REQUIREMENTS SPECIFICATION**

The software requirements specification document enlists all necessary requirements that are required for the project development. To derive the requirements, we need to have clear and thorough understanding of the products to be developed. This is prepared after detailed communications with the project team and customer.

## 3.1 FUNCTIONAL REQUIREMENTS

The Functional Requirements Specification documents the operations and activities that a system must be able to perform.

1. Live video feed as input
2. Placing piano layout inside capture frame
3. Registering keypress events of the piano layout

## 3.2 NON-FUNCTIONAL REQUIREMENTS

The modeling, automatic implementation and runtime verification of constraints in component-based applications. Constraints have been assuming an ever more relevant role in modeling distributed systems as long as business rules implementation, design-by-contract practice, and fault-tolerance requirements are concerned. Nevertheless, component developers are not sufficiently supported by existing tools to model and implement such features, we propose a methodology and a set of tools that enable developers both to model component constraints and to generate automatically component skeletons that already implement such constraints. The methodology has been extended to support implementation even in case of legacy components.

## 3.3 OTHER NON-FUNCTIONAL REQUIREMENTS

### **3.3.1 Safety and Security Requirements**

Many software-intensive systems have significant safety and security ramifications and need to have their associated safety- and security-related requirements properly engineered. For example, it has been observed by several consultants, researchers, and authors that inadequate requirements are a major cause of accidents involving software-intensive systems. Yet in practice, there is very little interaction between the requirements, safety and security disciplines and little collaboration between their respective communities. Most requirements engineers know little about safety and security engineering, and most safety and security engineers know little about requirements engineering. Also, safety and security engineering typically concentrates on architectures and designs rather than requirements because hazard and threat analysis typically depend on the identification of vulnerable hardware and software components, the exploitation of which can cause accidents and enable successful attacks.

### **3.3.2 Software Quality Attributes**

Following factors are used to measure software development quality. Each attribute can be used to measure the product performance. These attributes can be used for Quality assurance as well as Quality control. Quality Assurance activities are oriented towards prevention of introduction of defects and Quality control activities are aimed at detecting defects in products and services.

**Reliability**

Measure if product is reliable enough to sustain in any condition give consistently correct results. Product reliability is measured in terms of working of project under different working environment and different conditions.

**Maintainability**

Different versions of the product should be easy to maintain. For development it should be easy to add code to existing system, should be easy to upgrade for new features and new technologies time to time. Maintenance should be cost effective and easy. System be easy to maintain and correcting defects or making a change in the software

**Usability**

This can be measured in terms of ease of use. Application should be user friendly. The system must be Easy to use for input preparation, operation, and interpretation of output, and provide consistent user interface standards or conventions with our other frequently used systems. They should be easy for new or infrequent users to learn to use the system.

**Portability**

This can be measured in terms of Costing issues related to porting, Technical issues related to porting, Behavioral issues related to porting

**Correctness**

Application should be correct in terms of its functionality, calculations used internally and the navigation should be correct. This means application should adhere to functional requirements.

**Efficiency**

To Major system quality attribute. Measured in terms of time required to complete any task given to the system. For example, system should utilize processor capacity, disk space and memory efficiently. If system is using all the available resources, then user will get degraded performance failing the system for efficiency. If system is not efficient then it cannot be used in real time applications.

## 3.4 DEVEOPMENT REQUIREMENTS

**Model Training Requirement**

* Google Collaboratory - It's a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.

**Hardware Requirement**

PROCESSOR : Intel i3 or higher

RAM : 4 GB or higher (8 GB Recommended)

MONITOR : 13” or higher

HARD DISK : 8 GB or higher

PIANO LAYOUT : 5 Key Layout (Can be scaled)

WEBCAM : 720p webcam

**Software Requirement**

OPERATING SYSTEM : LINUX

PROGRAMMING LANGUAGE : PYTHON

IDE : SUBLIME TEXT

**CHAPTER 4**

# **DATASET AND FEATURES**

**CHAPTER 4**

**DATASET AND FEATURES**

## 4.1 DATASET

The dataset used in the project was generated from the live video feed as input. The fetched video feed was processed to generate the image frames for the respective key classes of the paper piano.

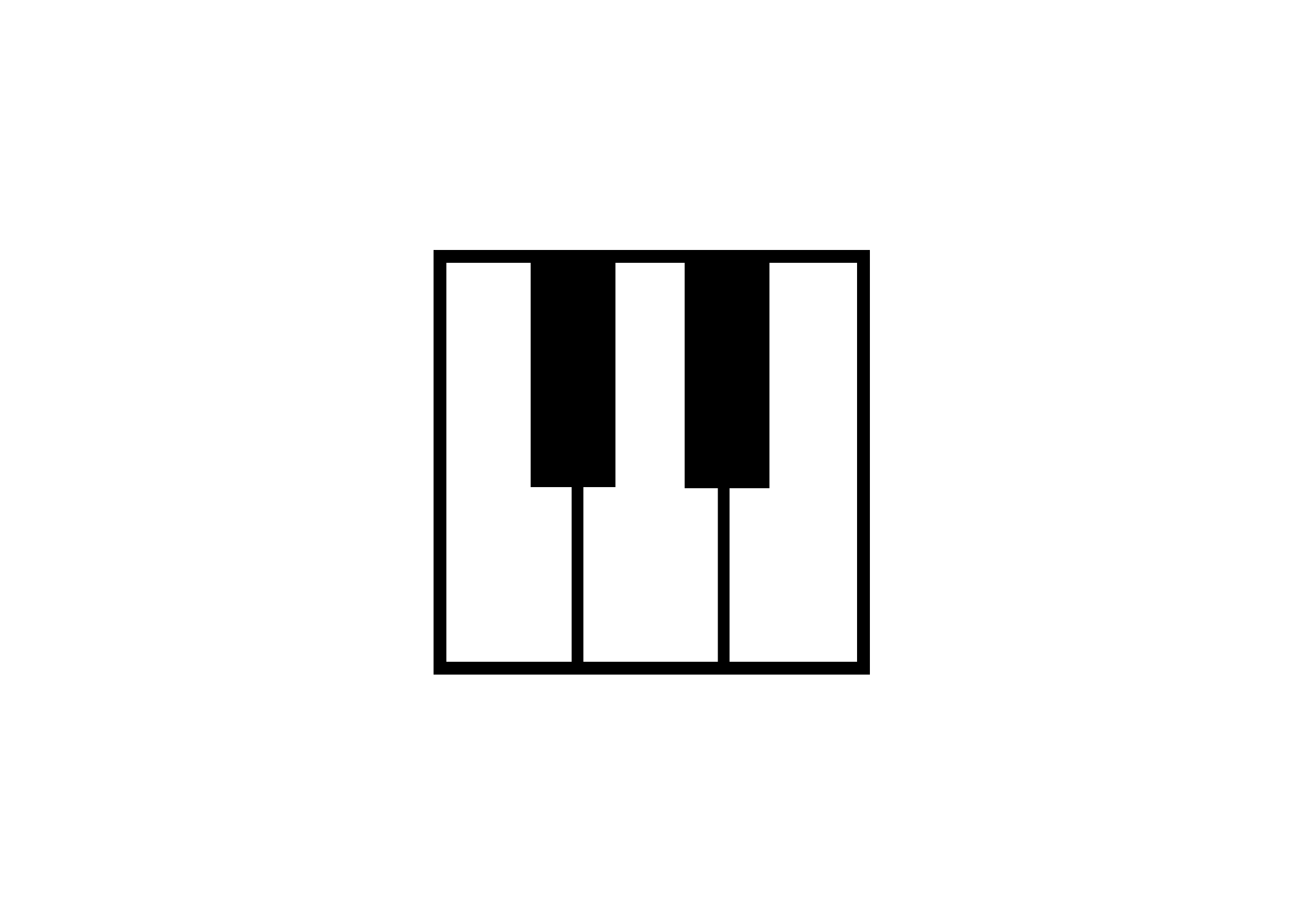
|  |  |  |  |
| --- | --- | --- | --- |
| CLASS | TEST SET 1 | TEST SET 2 | TEST SET 3 |
| KEY1 | 234 | 198 | 213 |
| KEY2 | 172 | 156 | 128 |
| KEY3 | 131 | 146 | 192 |
| KEY4 | 106 | 110 | 89 |
| KEY5 | 123 | 93 | 76 |
| KEY6 | 159 | 92 | 132 |

Table 1 : Test Set for all classes

## 4.2 FEATURES

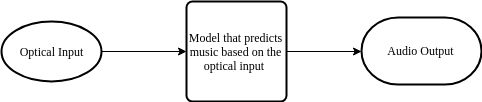
The Neural Network identifies all features during the learning phase itself. These features range from few to some crores in number based on its complexity.

Here we use white keys, black keys, layout of the piano as primary hyper features to enable successful learning. Thereafter multiple classes are used accordingly to successfully classify key press events of the paper layout.



**Figure 7: Keys Layout**

The system is performing a visual input to scan paper piano layout in order to play the music.



**Figure 8: Overview of the System**

**CHAPTER 5**

# **SYSTEM ANALYSIS & DESIGN**

**CHAPTER 5**

**SYSTEM ANALYSIS AND DESIGN**

## 5.1 SYSTEM ANALYSIS

### **5.1.1 System Development Life Cycle**

The SDLC is an application systems approach to development of information system. The tools of SDLC are using diagrams so it will be easier to understand, its stages related to each other. When changes occur in all phases of the system then it does not repeat again, SDLC phase is simpler.

**Figure 9: Software Development Life Cycle**

### **5.1.2 The Development Phases in Brief**

**a.) Planning Phase**

The Planning phase began with figuring out a way to develop a framework with organized deep reinforcement learning modules working at an environment agnostic level. And thus, to implement this model we planned to build an environment which will emulate and learn to play Games which run in the Atari Environment, and with our observations and models acquired by training the model we will be able to implement the concept of Reinforcement Learning by the means of Markov’s Decision Process into our systems and learn to play the game perfectly.

**b.) Analysis Phase**

This phase began with the analysis of the requirements needed to build the working model of our autonomous game playing system, by preparing an abstract and carrying out a literature survey and realizing about the different approaches to go about and develop the system. The analysis revealed that the primary difficulty arises due to insufficient exploration, resulting in an agent being unable to learn robust value functions. Intrinsically motivated agents can explore new behavior for its own sake rather than to directly solve problems. Such intrinsic behaviors could eventually help the agent solve tasks posed by the environment. We present hierarchical-DQN (h-DQN), a framework to integrate hierarchical value functions, operating at different temporal scales, with intrinsically motivated deep reinforcement learning.

**c.) Design Phase**

We constructed our design by partitioning the design of the whole system into two levels, the high level design and the low level design. In the High level we presented a system represented in its most abstract level which consisted of the methodologies and necessary approaches to go about designing the environment, agents and the convolutional neural networks. In the lower layer of the design we have focus mainly on the build and the detailed design of the Convolutional Neural Network and how it could perhaps be used to train the model using the tensors.

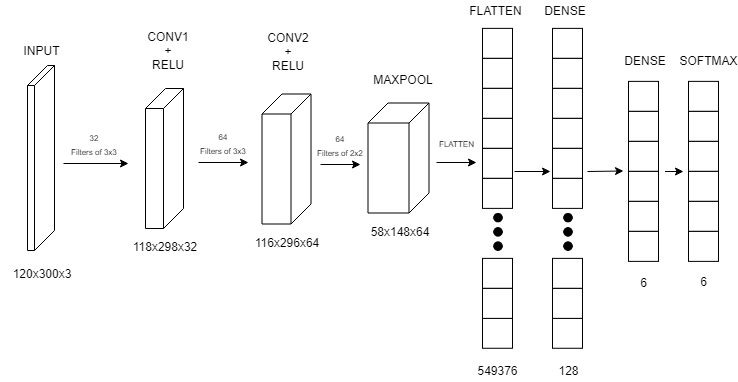
**d.) Implementation Phase**

The Implementation phase in order to develop a working model was divided into three segments, namely, common, simulator, & Train-Atari. In the first segment we handle parameters which are common throughout all the games running in the Atari environment, Simulator segment is used to define the part of the model which handles the objects related to the different simulators individually catering to the environment of the specific game. The train-Atari segment is the part of our model which is actually involved with the training process where the machine learns to play the game.

**e.) Use Phase**

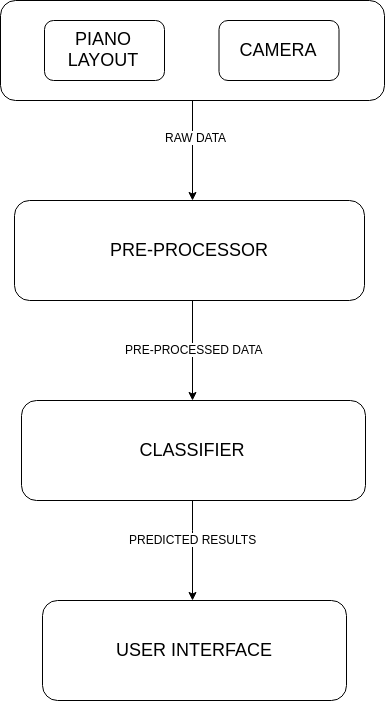
After the implementation phase we move on to the use phase where we have trained our model to train and play two games, Pong and Space Invader perfectly achieving almost perfect score every time. These use phases were used to test the model we can further extend our model to train different environments and use them in different classes of Applications such as Robotics, Finance Sector, Medical Sector (E.g. Amblyopia)

## 5.2 SYSTEM DESIGN

**5.2.1 Low Level Design**

**Figure 10: Custom CNN Model**

### **5.2.2 High Level Design**

****

**Figure 11: High Level System Architecture**

**CHAPTER 7**

# **IMPLEMENTATION**

**CHAPTER 7**

**IMPLEMENTATION**

## 7.1 IMPLEMENTATION

The design carried out in order to develop the model for being able to train and test and imply supervised reinforcement learning was translated into a working model by implementing our design process which was broken down into different segments of programmed source code. Each of these segments were responsible for handling an aspect of the working model. We have described the working of each of the segments along with the code snippets and implementation in the following subsections.

### **7.1.1 Model**

Our neural network model consists of 2 layers of Conv2D layers stacked sequentially followed by MaxPooling2D, Dropout, Flatten and Dense layers. The Conv2D layers uses relu activation function. The Dense layer uses softmax activation function. Before training the model, the learning process is configured using the compile method of the Keras model class, which takes the type of optimizer, type of loss and other metrics as parameters. The code snippet given below shows the model segment of our system.

.  
.  
model = Sequential()  
model.add(Conv2D(32, kernel\_size=(3, 3),  
                 activation='relu',  
                 input\_shape=input\_shape))  
model.add(Conv2D(64, (3, 3), activation='relu'))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Dropout(0.25))  
model.add(Flatten())  
model.add(Dense(128, activation='relu'))  
model.add(Dropout(0.5))  
model.add(Dense(num\_classes, activation='softmax'))  
  
model.compile(loss=keras.losses.categorical\_crossentropy,  
              optimizer=keras.optimizers.Adadelta(),  
              metrics=['accuracy'])  
.  
.  
.

### **7.1.2 Real-time Image Capture**

The video frame is captured with the help of **VideoCapture()** which reads image from a specific buffer of the memory. Continuous images are fetched using **VideoCapture.read()**. The captured frames of images are displayed using **cv2.imread()**. Keyboard events are captured with the help of the **cv2.waitKey()** which is detecting the ESC key for exiting out of the infinite loop. **VideoCapture.release()** is used for freeing the device which is being used for capturing the images. **cv2.destroyAllWindows()** is used to close all the opened windows which is used to display the captured frames of images.

import numpy as np  
import cv2  
  
# setting the video device capture to device 1 i. e external webcam  
cap = cv2.VideoCapture(1)  
  
while(True):  
        # read the frames from the device  
    ret, frame = cap.read()  
  
    .  
    .  
    .  
  
    # display the image captured from the device  
    cv2.imshow('frame',gray)  
  
    # taking the keyboard events  
    k = cv2.waitKey(33)  
    if k == 27:  
        # ESC key pressed - break the Loop  
        break  
  
# releasing the video device which is being used for data capturing.  
cap.release()  
  
# destroy all the windows.  
cv2.destroyAllWindows()

### **7.1.3 Training and saving the model**

Dataset is divided into batches of 20 images each and is clustered as such via batchGenerator function. After successful batching, data is reshaped and normalized to ease the process of training. Now the normalized data is trained using the fit function of Keras.models class.

Thus after all the epochs are completed for training the model, it is then saved to the local user system with the help of to\_json and save\_weights functions of Keras.models class which returns a representation of the model as a JSON string and saves the weights of the model as a HDF5 file respectively.

.  
.  
# number of epoches the model should be training.  
for e in range(epochs):  
  
    # fetching batched of images 20 each  
    for x\_train, y\_train in batchGenerator():  
  
        # reshaping of the training data  
        x\_train = x\_train.reshape(x\_train.shape[0], img\_rows, img\_cols, 3)  
        input\_shape = (img\_rows, img\_cols, 3)  
  
    # normalization of the training data  
    x\_train = x\_train.astype('float32')  
    x\_train /= 255  
  
    # convert class vectors to binary class matrices  
    y\_train = keras.utils.to\_categorical(y\_train, num\_classes)  
  
    # training of the model  
    history = model.fit(x\_train, y\_train,  
              epochs=1,  
              verbose=0,  
              validation\_split=0.05)  
  
# saving the model into the local system.  
model\_json = model.to\_json()  
with open("model.json", "w") as json\_file:  
    json\_file.write(model\_json)  
  
model.save\_weights("model.h5")  
.  
.  
.

### **7.1.4 Imports and Hyper parameters**

The **OS** library is used to fetch images from the generated dataset and to access some low level services of operating system’s interfaces of the for the same cause. The high-level Neural Network library Keras is used for creating our CNN model, and it runs on top of TensorFlow backend. We are using linear stacked layer model called Sequential model. Layers used here include Dense, Dropout, Flatten, Conv2D and MaxPooling2D layers. We are setting the hyper parameters for the network as batch size of 20 images.

import os  
  
import keras  
from keras.models import Sequential  
from keras.layers import Dense, Dropout, Flatten  
from keras.layers import Conv2D, MaxPooling2D  
from keras import backend as K  
  
batch\_size = 20  
num\_classes = 6  
epochs = 20  
img\_rows, img\_cols = 120, 300  
input\_shape = (img\_rows, img\_cols, 3)  
.  
.  
.

**7.1.5 Loading the model and prediction**

For loading the model we use **model\_from\_json** and **load\_weights** functions of the Keras.models class for loading back the model configuration from the json file and loading the different weights of the network respectively. The model is then configured with the help of the **compile** method of the Keras.models class. Now the data which is to be predicted is reshaped and normalised before it is send for prediction. Now for prediction we use either **evaluate** or **predict** method of the Keras.models class, which gives us a vector of probabilities a particular sample belonging to a specific class.

from keras.models import model\_from\_json  
.  
.  
.  
  
# load json and create model  
json\_file = open('model.json', 'r')  
loaded\_model\_json = json\_file.read()  
json\_file.close()  
  
loaded\_model = model\_from\_json(loaded\_model\_json)  
  
# load weights into new model  
loaded\_model.load\_weights("model.h5")  
  
loaded\_model.compile(loss=keras.losses.categorical\_crossentropy,  
              optimizer=keras.optimizers.Adadelta(),  
              metrics=['accuracy'])  
  
# reshaping the data  
x\_test = x\_test.reshape(x\_test.shape[0], img\_rows, img\_cols, 3)  
  
# normalization of the new data.  
x\_test = x\_test.astype('float32')  
x\_test /= 255  
y\_test = keras.utils.to\_categorical(y\_test, num\_classes)  
  
# prediction of the results.  
score = model.evaluate(x\_test, y\_test, verbose=0)

# or.

score = model.predict(x\_test)

**7.1.6 Model Upload and Download**

Because of the scarcity of resources for computation the model is trained in the cloud in the google collab servers. To save the model from their servers to a local machine or our Google Drive we use the **pydrive** module provided by google for storage purposes. The object of **drive.CreateFile** is called to create a pydrive object to which the file is specified which is to be uploaded using **SetContentFile** and then uploaded using the **Upload** method of the pydrive object.

from pydrive.auth import GoogleAuth  
from pydrive.drive import GoogleDrive  
from google.colab import auth  
from oauth2client.client import GoogleCredentials  
.  
.  
.

# Create & upload model.h5 file.  
uploaded = drive.CreateFile()  
uploaded.SetContentFile('model.h5')  
uploaded.Upload()  
  
# Create & upload model.json file.  
uploaded = drive.CreateFile()  
uploaded.SetContentFile('model.json')  
uploaded.Upload()

## 7.2 EXECUTION PHASE

### **7.2.1 Imports and Constants**

The interface of the system is developed using **Tkinter python library.**

import simpleaudio as sa

import time

from threading import Thread

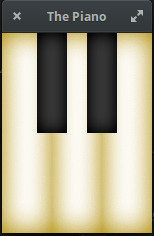
try:

  from Tkinter import Tk, Frame, BOTH, Label, PhotoImage

except ImportError:

  from tkinter import Tk, Frame, BOTH, Label, PhotoImage

### **7.2.2 User Interface**

Yahape kuch likhna h asdbkasbvkbaskcbaskjbcjkabsckj

**Figure 12: User Interface in Tkinter of the Paper Piano layout**

**CHAPTER 8**

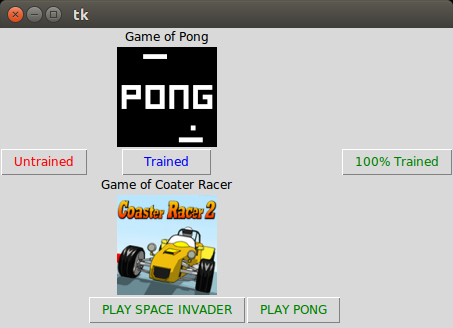
# **EXECUTION AND TEST CASES**

**CHAPTER 8**

**EXECUTION AND TEST CASES**

## 8.1 EXECUTION

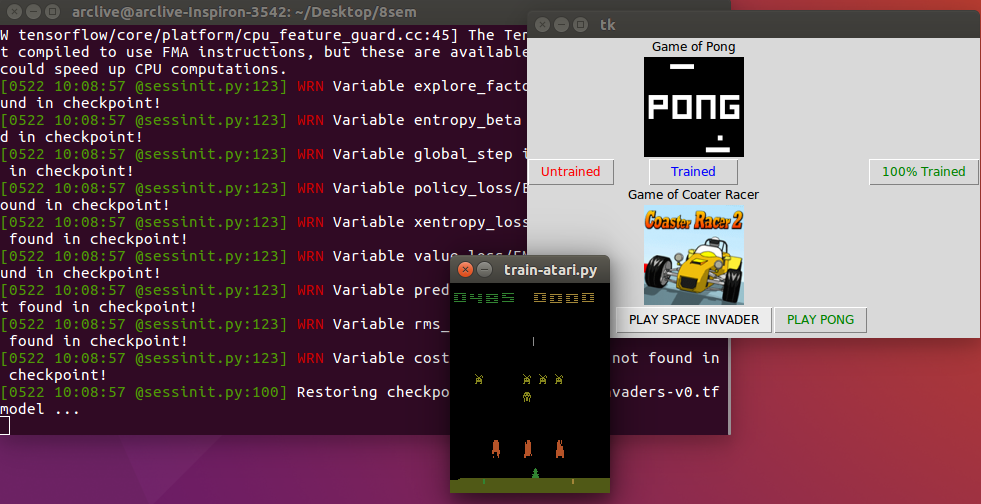
The User Interface which is displayed for the user to interact with the working, completely trained, untrained and partially trained models running in the system.



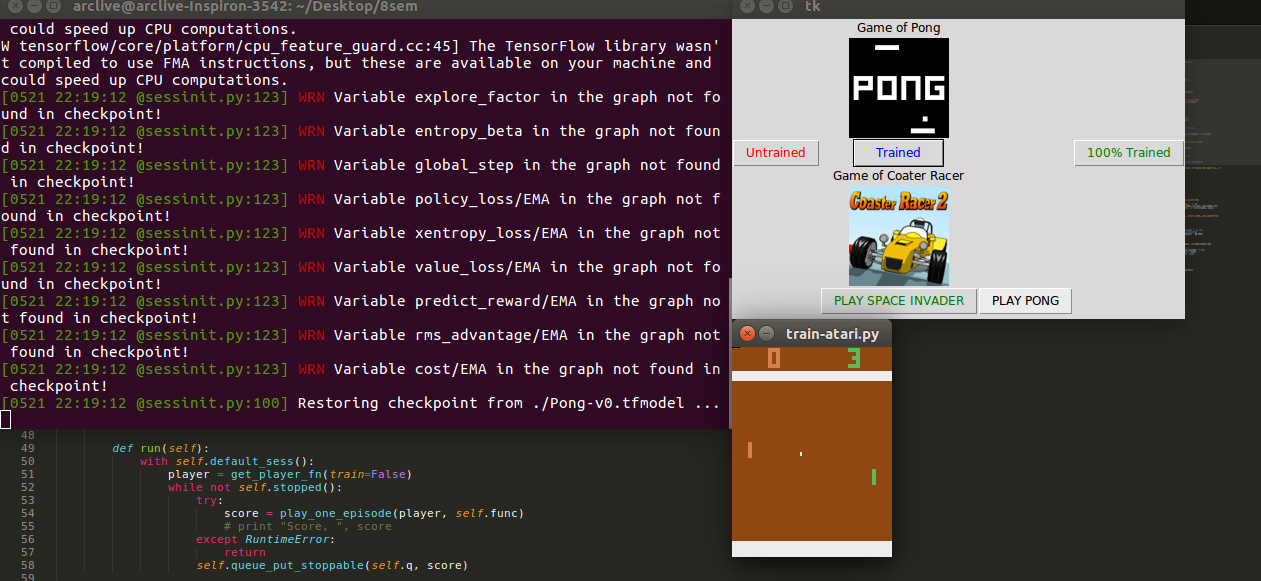
**Fig 8: Snapshot of the GUI**

## 8.1.1 Test Runs

We have captured the screenshots of the test runs performed on two games as shown below.

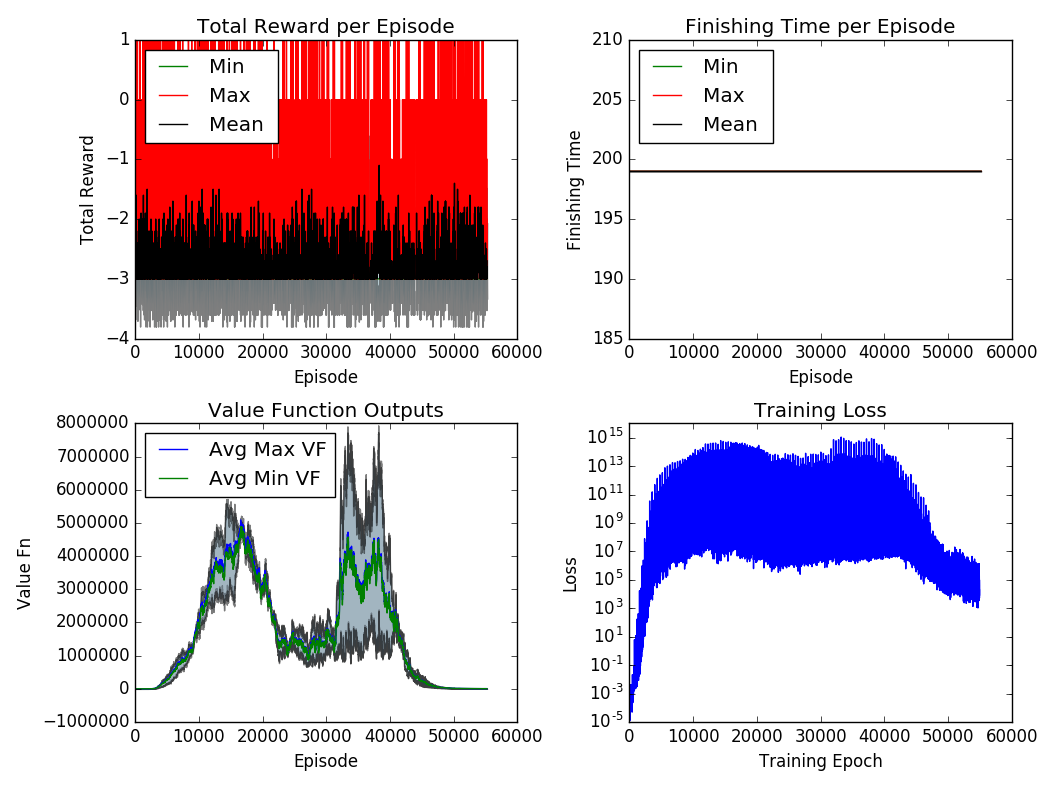


**Fig 9: Snap of the final testing phase of the Space Invader Game**

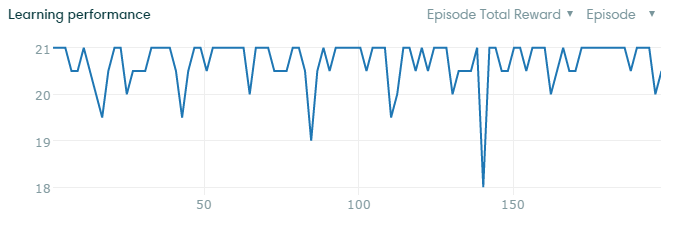


**Fig 10: Snap of the final testing phase of the Pong Game**

### **8.1.2 Observations and Graphs**



**Fig 11: Quality Graphs**



**Fig 12: Runtime Graph**

**CHAPTER 9**

# **CONCLUSION AND FUTURE ENHANCEMENTS**

**CHAPTER 9**

**CONCLUSION AND FUTURE ENCHANCEMENTS**

## 9.1. CONCLUSION

This project uses Convolutional Neural Networks model for detecting hand motion of the end user and hence playing the audio in correspondence with the musical instrument from the video feed. This model uses convolutional neural network which parses each video frame and extracts respective feature set to feed the model. The model doesn’t require explicit manual labelling, which helps to avoid the overhead. In future, we will be working on improving its latency in order to make it more realistic and expeditious. This would lead to an increase in its real time applications and gradually an alternative for bulky and costly instruments.

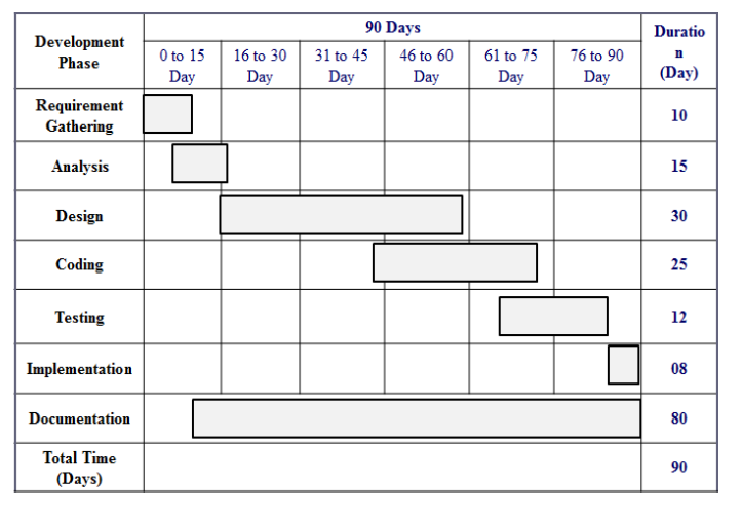
This is a difficult problem due to four aspects.

## 9.2 FUTURE WORK

## 9.3 PROJECT ACTIVITY

Project Activity Gantt Chart

This project is completed in a total duration of three months, along with three monthly project reviews which were conducted in the month of March, April and May.



# **REFERENCES**

**REFERENCES**