**SYSTEM REQUIREMENTS**

**HARDWARE REQUIREMENTS:**

* Processor Type - Pentium - IV
* Speed - 2.4 GHZ
* Ram - 8 GB
* Hard disk - 552 GB SSD

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows 11
* Software Programming Package - MATLAB R2022a

**SYSTEM USED:**

Name of the Company: MI

Version: 21H2

**DESCRIPTION OF TOOLS USED:**

The project is mainly implemented in MATLAB using several packages. Deep learning modules ,which are constructed with certain tools are used in our project to classify age, gender and race.The tools are:

* Matlab R2022a
* Deep learning toolbox
* Deep network designer

**MATLAB**

MATLAB (Matrix Laboratory) is a matrix-oriented language for technical computing. It is not only used for computation, but also for visualization and programming in an easy-to-use environment. It is an interpreted language (not compiled) that was conceived to provide easy access to matrix and linear algebra software that was written in FORTRAN. One of the main features of MATLAB is that it is oriented toward numerical computing, instead of symbolic computing (as e.g., Maple software, Mathematical).The software comes in the form of a core program and addition a libraries or toolboxes. A toolbox is a collection of MATLAB functions (called M-functions or M-files) that extend the capability of the core environment to solve specific topic problems. MATLAB is optimized to be relatively fast when performing array operations, so it is important to take this into account to write suitable instructions, for example, to avoid unnecessary ’for’ loops that process individual array elements.

**DEEP LEARNING TOOLBOX**

Deep Learning Toolbox provides a framework for designing and implementing deep neural networks with algorithms, pre-trained models, and apps. You can use convolutional neural networks (Con-Nets, CNNs) and long short-term memory (LSTM) networks to perform classification and regression on image, time-series, and text data. You can build network architectures such as generative adversarial networks (GANs) and Siamese networks using automatic differentiation, custom training loops, and shared weights.

With the Deep Network Designer app, you can design, analyze, and train networks graphically. The Experiment Manager app helps you manage multiple deep learning experiments, keep track of training parameters, analyze results, and compare code from different experiments. You can visualize layer activations and graphically monitor training progress.

Create new deep networks for image classification and regression tasks by defining the network architecture and training the network from scratch. You can also use transfer learning to take advantage of the knowledge provided by a pretrained network to learn new patterns in new data. Fine-tuning a pretrained image classification network with transfer learning is typically much faster and easier than training from scratch. Using pretrained deep networks enables you to quickly learn new tasks without defining and training a new network, having millions of images, or having a powerful GPU.

After defining the network architecture, you must define training parameters using the training option function. You can then train the network using train network. Use the trained network to predict class labels or numeric responses.

You can train a convolutional neural network on a CPU, a GPU, multiple CPUs or GPUs, or in parallel on a cluster or in the cloud. Training on a GPU or in parallel requires Parallel Computing Toolbox™. Using a GPU requires a supported GPU device (for information on supported devices, see GPU Support by Release (Parallel Computing Toolbox)). Specify the execution environment using the training Options function.

**DEEP NETWORK DESIGNER**

The Deep Network Designer app lets you build, visualize, edit, and train deep learning networks. Using this app, you can:

* Build, import, edit, and combine networks.
* Load pretrained networks and edit them for transfer learning.
* View and edit layer properties and add new layers and connections.
* Analyze the network to ensure that the network architecture is defined correctly, and detect problems before training.
* Import and visualize datastores and image data for training and validation.
* Apply augmentations to image classification training data and visualize the distribution of the class labels.
* Train networks and monitor training with plots of accuracy, loss, and validation metrics.
* Export trained networks to the workspace or to Simulink®.
* Generate MATLAB® code for building and training networks and create experiments for hyperparameter tuning using Experiment Manager.