# Chapter 1: Introduction

## Problem Statement

## Objective

* To design a Multi-Layer Feedforward Neural Network Model that recognizes Handwritten digits (one at a time) with high accuracy.
* To properly analyze and optimize the designed model using different model improvement techniques such as normalization, regularization, hyperparameter tuning, etc. and optimization algorithms like Adam and Minibatch Gradient Descent.
* To implement the components of the project from scratch (using no more than core python packages) so that the authors could have an in-depth understanding of Deep Learning.

# Chapter 2: Literature Review

# Chapter 3: Methodology

## Resource Requirements:

Various hardware and software resources required for the successful completion of this project are:

Hardware: PC with Multi core CPU

Software: Python Programming Language with Numpy, Matplotlib and Scipy as its core packages, Anaconda Navigator

## Process Model:

There are a huge number of process models that can be adopted to carry out a project. Yet, these existing models are either application development-centric or enterprise architecture focused or rooted in hardware or software development approaches. So they face a significant challenge when used with the unique lifecycle requirements of AI/ML projects. So there is a need of project management methodology that takes into account the various data-centric needs of AI while also keeping in mind the application-focused uses of the models and other artifacts produced during an AI lifecycle.

[CPMAI](https://www.cognilytica.com/cpmai-methodology/): The Cognitive Project Management for AI Methodology, developed by Cognilytica with dozens of other organizations, is one of the best methodologies adopted for a large number of real-world Big data and AI projects. CPMAI extends the well-known CRISP-DM methodology with AI and ML specific documents, processes, and tasks. The CPMAI methodology also incorporates the latest practices in Agile Methodologies and adds additional DataOps activities that aim to make CPMAI data-first, AI-relevant, highly iterative, and focused on the right tasks for operational success.

Since the methodology is for large organizations and is production oriented, many of its components are quite not feasible for a small project like this. So the methodology was adopted with necessary changes to best suit the needs of this project.

1. Project Understanding
2. Data Understanding
3. Data Preparation
4. Data Modeling
5. Model Evaluation

The methodology consists of 5 major phases:

* Project Understanding: This phase starts since the inception of the project idea. It then manifests to project selection followed by the understanding of the project characteristics and requirements. It then establishes the project among the *seven patterns of AI*. Along with the creation of the project plan, the scope and objective of the project is also defined in this phase.
* Data Understanding: In this phase the data requirements of the project are identified. The data are then looked for in different sources and checked for its authenticity. The data format is evaluated and the quality of the data is ensured. On completion, this phase is able to give a good description of the data to be used.
* Data Preparation: This phase includes the importing of the dataset and processing it so that the dataset becomes ready for training the model. Here, the data is cleaned, normalized and the label is encoded as necessary. Data Augmentation is carried out to upscale the data volume whenever required. Besides, the main data is partitioned into train, dev and test set in a certain manner that are used to train the model.
* Modeling: This phase helps to select a modeling technique and any required algorithms. Here, we make all the assumptions that are required to build a model including its architecture, generate test designs and train the model. Besides this phase also includes different approaches to optimize, retrain and scale the model. This phase combined with the evaluation phase takes majority of the lifespan of the project.
* Model Evaluation: This is one of the most crucial phases for designing a better model. The evaluation of the model gives feedback about how well the model is learning and generalizing the data. Here, various outcomes and visualizations of the model like accuracy, learning curve, loss curve etc helps to analyze the state of the model and gives intuitions about the what approach to select to better optimize the model. Once the problem is identified and a suitable approach is selected, this phase loops back to either data preparation phase for collection of more data or the modeling phase for redefining and retraining of the model with necessary changes.

## Tools and Techniques:

Design Tools:

Following are the design tools used by the authors in this project.

**MS Visio**: MS Visio is a diagram design tool used to create diagrams for varieties of field ranging from floor designs to software engineering designs and many more. In this project, the authors have used this tool to create DFD and the project Schedule (Gantt Chart).

**MS Word**: MS Word is a document design tool that provides an extended range of useful features. Various reports, including the proposal itself, are designed by the authors using this tool.

Implementation Tools:

The implementation for this project is totally based on the Jyputer platform, an ecosystem that enables others to build tools on top of it. Jyputer is a collaboration tool for writing and sharing code and text within the context of a web. The code runs on a server, that can be anywhere, and the results are turned into HTML incorporated into the page one is writing.

[Jupyter is built from three parts](https://www.oreilly.com/radar/what-is-jupyter/):

1. **Jupyter Notebook- The Front End**: It is a web-based interactive computational environment for creating Jupyter notebook documents. It allows one to write code and edit and run them in the notebooks.
2. **The Jupyter server**: It is a relatively simple application that runs on your laptop, or a multi-user server like JupyterHub.
3. **The kernel protocol**: It allows the server to offload the task of running code to a language-specific kernel.

**JupyterLab**: It is a Jupyter based IDE that offers all the familiar building blocks of the classic Jupyter Notebook (notebook, terminal, text editor, file browser, rich outputs, etc.) in a flexible and powerful user interface.

# Chapter 4: System Analysis

## Functional requirement

Functional requirements for a model describe what the model should and should not do. It also describes how the model should behave for a particular set of inputs. The functional requirements of the given model are as follows:

* The model should be able to classify any new image of a handwritten digit with maximum accuracy.
* The model should not classify any alphabets and characters as a digit.

## Data Description

MNIST Database

[MNIST Database](http://yann.lecun.com/exdb/mnist/) is a large database of handwritten digits images that is extensively used as a toy database for various image processing systems and machine learning projects in academics and industries. It was adopted from a larger database called NIST, with some manipulation on the images, by a team of researchers lead by Yann LeCun.

The database consists of four files:

* Train-images-idx3-ubyte.gz: training set images (9912422 bytes)
* Train-labels-idx1-ubyte.gz: training set labels (28881 bytes)
* T10k-images-idx3-ubyte.gz: test set images (1648877 bytes)
* T10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)

The data is stored in the idx file format designed for storing vectors and multidimensional matrices.

The training set consists of 60000 examples and the test set consists of 10000 examples. The first 5000 examples of the test set are taken from the original NIST training set. The last 5000 are taken from the original NIST test set. The first 5000 are cleaner and easier than the last 5000.

Processed MNIST Dataset

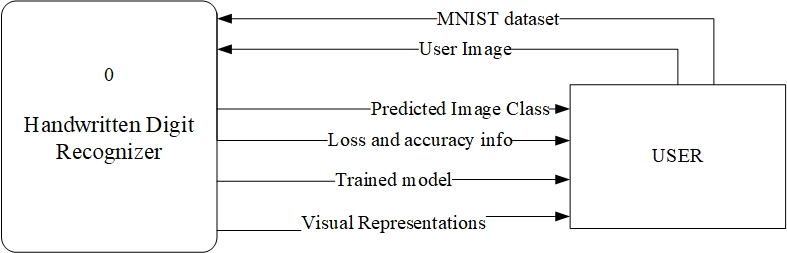
The dataset when imported is converted into a set of n-dimensional arrays.

|  |  |  |
| --- | --- | --- |
| Dataset | Datatype | Array Size |
| Training Set Image | <class 'numpy.ndarray'> | (60000, 28, 28) |
| Training Set Label | <class 'numpy.ndarray'> | (1, 60000) |
| Test Set Image | <class 'numpy.ndarray'> | (10000, 28, 28) |
| Test Set Label | <class 'numpy.ndarray'> | (1, 10000) |

After importing, the test set is split into development/validation set and test set through random selection of the data resulting in Train/ dev and test set. Then the input image is flattened and normalized. The output labels are transformed into one-hot representation. Finally, the dataset is ready and has the following form

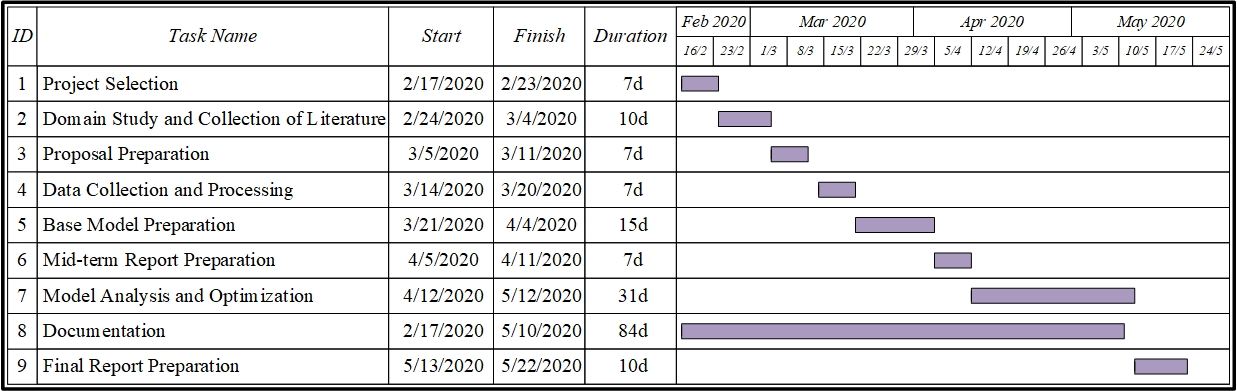
|  |  |  |
| --- | --- | --- |
| Dataset | Processed Array size | Original Array Size |
| Training Set Image | (784, 60000) | (60000, 28, 28) |
| Training Set Label | (11, 60000) | (1, 60000) |
| Dev Set Image | (784, 5000) | (5000, 28, 28) |
| Dev Set Label | (11, 5000) | (1, 10000) |
| Test Set Image | (784, 5000) | (5000, 28, 28) |
| Test Set Label | (11, 5000) | (1, 10000) |

Process Modeling



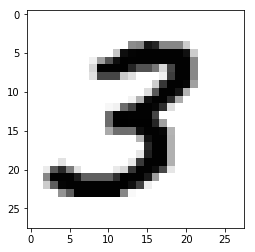
# Chapter 5: Project Schedule Plan

This project begun with the



# Chapter 6: Expected Output

The final model should be able to classify any new image of a handwritten digit with maximum accuracy. If the input image is of 3, the model should be able predict that its 3 as shown below.



Actual Label: 3, Predicted Label: 3

# Chapter 7: References