Automatic Classification of Medical Modality using Convolutional Neural Network

Moana Yamanouchi (27742407)

Supervised by: Dr. Mohammad Reza

Project Proposal

FIT3161 Computer Science Project 1

Monash University Malaysia

School of Information Technology

Abstract

With rapid technological advances in modalities, new algorithms and methods are required in order to differentiate the characteristics of the medical images within the large image collection. This task can be achieved using image classification which assigns multitude of input image features from dataset into a specific category through extraction and fusion. The two main key challenges for medical image classifications are 1) identifying the differences ranging from subtle to distinct among modalities and 2) the dataset provided in medical dataset is imbalanced and limited labelled dataset which will increase the risk of overfitting. Our proposed methodology is to use fine-tuned convolutional neural network using fusion technique to improve classification accuracy,

Keywords: Image classification, medical modality, convolutional neural network, fusion technique.

Contents

1. Introduction

Medical imaging is an essential process for diagnosis, monitoring and treatment to make accurate medical decisions. There are various types of medical imaging such as Computer Tomography (CT), Magnetic Resonance Imaging (MRI), X-ray and Ultrasound. Using images for comparative studies is an essential technique for diagnostic decision-making for health professionals based on their knowledge and analysing from a patient's data. In recent years, improvements in healthcare have been made due to the technological advancement of medical modality. The growing focus of development in medical devices and high demand for both medical and business field creates many high investment opportunities in the medical sector.

The most commonly used database for storing and accessing medical images is text-based Picture Archiving and Communication Systems (PACS). Although PACS search is useful for health professionals to search known identifiers and characteristics, it cannot retrieve information based on visual features[1]. Furthermore, PACS image retrieval becomes infeasible for large imaging datasets which is inevitable in modern healthcare as diverse ranges of imaging data is required and is compounded with development of technological innovations in medical imaging modality. With such large volume, manually labelling images become uneconomical and prone to human error.

To eliminate these limitations, content-based image retrieval (CBIR) was introduced[2] by assessing quantitative measures and objective features followed by extracting similar information from a feature representation at a pixel-level. This image search technique utilizes visual features which includes mainly colour, shape, and texture for medical images. The two main key components of CBIR are providing 1) suitable feature representation and 2) effective similarity measures. The main challenge for CBIR is the semantic gap between low level images and high level concept; machine learning has become a prominent technique to reduce this gap. Identifying an accurate feature representation and specifically classifying the images is an important technique in CBIR. This task can be achieved by using medical modality classification.

1.1 History

Convolutional Neural Networks are used in deep learning and in a plethora of computer vision tasks today. However, there is some history to how its architecture and layers work the way they do. In 1962, two researchers, Hubel and Weisel[3] were experimenting to find out to what stimulus, neurons respond to. They recorded from neurons in the visual cortex of a cat, as they moved a bright line across its retina. They had discovered that only certain neurons were fired (occurrence of an all-or-none action potential) only when light was on a particular position of the retina. The activity of the neurons that were activated, changed depending on the orientation of the line of light. Sometimes the neurons fired only when the line of light was moving in a particular direction. It was then discovered that neurons in the first few layers of a visual cortex will only pick up very simple features, like the orientation. This provided an overview of how the visual system operates. It constructs complex patterns in layers of understanding from simple features to more complex ones.

In 1982, a neuroscientist named David Marr conceptualized how images are reconstructed into 3D images in the visual cortex[4]. The first stage which he called *Primal Sketches*, which involved the use of optical flow and segmentation, to find the rough edges of objects in the image field. Followed by the second stage, *2.5D*, which is 2D with texture, depth and shade to provide further detail to the image. And the final stage, *3D*, reconstructing the entire scene.

Furthermore, in 1980, Kunihiko Fukushima had proposed and created a neural network for visual pattern recognition which could learn without any help and acquired an ability to recognize distinct patterns in an image based on geometric similarity of their shapes without being influenced by their positions. The network was nicknamed "Neocognitron". After the experiment was fully carried out, the self-organized network had a structure which was very similar to the hierarchy of the visual system that had been proposed by Hubel and Weisel. With all this ground-breaking research, there has been several biologically inspired CNN methodologies that have been proposed throughout the recent years, each with their own unique tweaks and incorporated techniques.

1.2 Overview

With rapid technological advances in modalities, new algorithms and methods are required in order to differentiate the characteristics of the medical images within the large image collection. This task can be achieved using image classification which assigns multitude of input image features from dataset into a specific category through extraction and fusion.

1.3 Problem Statement

A challenge for modality classification is identifying the differences ranging from subtle to distinct among modalities, along with the variation in the appearance of individual diseases which affects the application of classification techniques. For example, the distinct difference between X-ray and CT scan must be classified into two different modality classes. Whereas, the subtle differences between fractured and dislocated leg X-ray images needs to be classified into one modality class.

Another challenge is the imbalanced and limited labelled dataset. For example, in ImageClef 2016, training dataset of 6776 labelled medical images were provided, where approximately 42% of datasets were statistical figures, graphs, charts (GFIG). Whereas, ImageNet 2012 [5] consists of 1.2 million images. Statistically comparing these two datasets, if both accuracy resulted in same percentage, ImageNet will still obtain higher accuracy due to the difference in the size. The imbalanced and limited labelled data will lead to risk of overfitting which will be further discussed later on.

1.4 Motivation

Various research papers on image modality classification using convolutional neural network (CNN) and fusion techniques will be analyzed. The main goal is to accurately and efficiently classify the images regardless of the differences in visual characteristics.

The main objectives are:

- To critically summarize and evaluate previous research papers

- Explain the improvements of image modality classification in recent years
- To identify different types of CNN architecture and analyse its differences
- To identify different methods of fusion technique in CNN and analysis its advantages and disadvantages
- To include arguments/ideas while maintaining logical progression

1.5 ImageClef 2016 Database

For comparative analysis purpose based on accuracy results, information from ImageClef 2016 will be used in Section 4.2 and 4.4. Medical task in ImageClef is an ongoing evaluation campaign that is a part of the Conference and Labs of the Evaluation Forum (CLEF) initiated since 2004. Five subtasks that were provided in 2016 are:

- Compound Figure Detection
- Multi-label classification
- Figure separation
- Subfigure Classification
- Caption Prediction

Subfigure classification is, as shown above, is one of the subtasks which was already introduced to a similar subtask in 2011. The dataset used in modality classification are figures from multi-label classification task that is then distributed into labelled subfigures. The aim is to classify 30 heterogeneous classes ranging from various diagnosis images to generic biomedical illustrations.

1.6 Feature Extraction

The two main feature extractions are 1) handcrafted features and 2) automatic feature extraction using deep learning. Handcrafted features for modality image classification have been proposed such as intensity histogram, local binary pattern (LBP) and scale-invariant feature transform (SIFT) which are then are encoded into bag-of-words (BoW). For automatic feature extraction, amongst various deep learning techniques, convolutional neural network (CNN) has become a popular technique as it outperformed other conventional methods based on its high accuracy result and low test error rate

2. Background

2.1 Convolutional Neural Network

CNN is a deep learning model which is a type of artificial neural network that analyses data using mathematical model. It is widely used in medical image analysis as it simplifies the training process by implemented trained end-to-end using supervised learning algorithm.

Deep learning is a subset of Artificial Intelligence (AI) which constructs a machine learning model that primarily focuses on using appropriate hierarchical representation of the data to draw conclusions. It is an architecture that is used in various machine learning application especially for a more detail-oriented and accurate image classification. Artificial neural network (ANN) is a class of machine learning algorithm that was biologically inspired by the

mechanism of the human brain. In this section, the fundamental concept of feed-forward neural network will be introduced as a basis of deep model.

In a supervised learning algorithm, it consists of training dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ which consists of images x_i associated with labelled data y_i. Input feature x produces a predicted score $\hat{y}$ using a score function $\hat{y} = f(x_i, W)$ where matrix weight W is a parameter. The loss function $l(\hat{y}, y)$ is then applied to compare with the actual target labelled data y and $\hat{y}$ to compute its disagreement. The gradient of the loss function is then computed where W is updated using backpropagation by applying recursive application of chain rule by computing the gradient (partial derivative) f on x. Each neurons, therefore has a value of f which is then adjusted during backpropagation by multiplying the local gradient and its input value. An example would be classification, regression, object detection, semantic segmentation and image captioning.

In an unsupervised learning algorithm, it is trained to learn an underlying hidden structure of the data without using labelled data y. An example of unsupervised learning is clustering, dimensionality reduction, feature learning and density estimation. A common technique that is used to evaluate the quality is by applying feature extractor on a supervised dataset.

Training CNN using dataset for supervised learning algorithm will have space complexity $O(N^2)$ as it contains both N number of image x and N number of labelled data y in the dataset. Whereas, unsupervised learning algorithm will have space complexity of $O(N)$ as only image x is required in the dataset. Therefore, based on the comparison of space complexity, unsupervised learning algorithm will be more efficient as it requires less memory space.

2.1.1 Architecture

CNN consists of three main layers which are convolutional layer, pooling layer and fully-connected layer. It views inputs as images which is what distinguishes from other neural network. This assumption is taken as an advantage and structures the neurons in three-dimensional volume which significantly reduces the total number of parameters in the network.

Convolutional Layer

In convolutional layer, it consist a filter (or kernel) of array weight W which slides/convolves over the height and width of the input value using a dot product to produce two dimensional activation map. The filter is slide across the image depending on the stride length and padding. This map detects the specified visual feature. The network will initially learn low level features and as it progresses in subsequent layers, a more complex abstract feature becomes the input.

Parameter sharing scheme is implemented to control number of parameters which constrains neurons in each depth slice. This is due to the sparse connection between layers, causing the area of the filter to decrease. By constraining the weights, it will increase the algorithm efficiency and by using constant weights, it will significantly reduce its memory storage.

Pooling Layer

The main function of pooling layer is to downsample the spatial size of the image and number of parameters. The most commonly used operation is max-pooling which extracts the highest input value within the targeted filter to compile the highest activations.

Fully Connected Layer

The fully connected layer is the final layer of the network where, unlike pooling layer, each neuron in the input volume is fully connected to the next layer.

2.1.2 Feature Extraction

There are two main methods for feature extraction using CNN architecture[7]: 1) Global feature extractor obtains a holistic representation of the image by extracting the activation from fully-connected layers. 2) Local feature extractor focuses more on a specific area which is achieved by removing the fully-connected layers and accepts activation maps as local features. The features are then combined into single vector using encoding method such as VLAD encoding, BoW encoding or Fisher Vector encoding. Sum-pooling or max-pooling is also used as local feature aggregation for its efficient computation and non-parametric measures.

2.2 Fusion Technique

Fusion technique is a strategy to fusing different information to improve accuracy and robustness.

2.2.1 Early Fusion

Early fusion, also known as feature fusion, is a technique where all distinct features vectors are concatenated into a single feature vector. By combining these high dimensional feature vectors, it stores the effective discriminate information while simultaneously eliminates redundant information. Therefore, the total vector size would be the sum of the features dimensions due to the merging of all feature vectors. Although, this gives good performance, feature discrimination is neglected as it completely relies on the learning algorithm. Therefore, sum (or average) pooling or maximum pooling is implemented to extract only the significant feature values.

2.2.2 Late Fusion

Late fusion integrates separate classification scores. This method is achieved by using features that extracted from individual channel and feeding into a classifier to extract each individual classification score. The two disadvantages in late fusion is 1) number of classifiers is same as total number of features which is computationally infeasible and 2) connection between individual features do not reflect on the classification score due to the individual complementary cues. Regardless of these disadvantages, late fusion is still used for its accurate result.

2.3 Related Works

2.3.1 Types of CNN Architecture

AlexNet[8],introduced by Krizhevsky *et al.*, was the first large scale CNN achieved in image classification which outperformed all previous non-deep learning application in ILSVRC challenge in 2012 with top 5 error rate of 16% compared to 26%. In the year 2014, Simonyan and Zisserman introduced *VGGNet* which reinforced that the depth of the network correlates with the overall performance of hierarchical representation of the data.

In ILSVRC 2014, *GoogLeNet*[9] was the winner due to its high computational efficiency. The main contribution was the development of "Inception Model" by creating an efficient local network topology. The architecture uses parallel operations containing multiple receptive field sizes for convolution and pooling operation which then concatenates into single vector in the next layer. However, the total number of convolutional operation (which includes spatial location for each filer map and dot product operation) and the pooling operation would lead to high computational complexity. This lead to the improvement of the architecture where linear filter was replaced with "micro networks", a bottleneck layer that uses 1x1 convolutions that was investigated by Lin *et al.*[10]. By implementing the extra layer, it preserves spatial dimension while reducing its depth.

Compared to *AlexNet* both deeper CNNs, *VGGNet*[11] and *GoogLeNet*, has decreased its parameter, because smaller filters were applied while sustaining the effectiveness of the receptive field. However, this increases the total space complexity, because each node requires memory allocation for forward and backpropagation.

2.3.2 Analysis on ImageClef 2016

Biomedical Computer Science Group (BCSG) obtained the highest accuracy result of 88.43% by applying early fusion technique of four different classifiers: 1) applying early fusion of both visual and textual feature which were fed into to SVM classifier with RBF kernel, 2) transfer learning was applied where features were extracted from1) and fed into pre-trained *ResNet-152[12]*, 3) adopting global visual feature by using transfer learning on *ResNet-152[12]*and extracting its features from fully-connected layer. Furthermore, transition from higher to lower dimensional feature was applied using *Projection Learning Rule* and 4) applying matrix factorization technique called non-negative matrix factorization (NMF).

Early fusion of both textual and visual features has improved the accuracy of the result. By applying the current state-of-the-art CNN architecture, *ResNet-152[12]*, has improved its accuracy result. For further improvement, fine-tuning can be applied instead of transfer learning, because transfer learned features are reflective of the original dataset (which are usually images that do not link with medical modality images) which may not reflect on the subtle differences of medical images.

NovaSearch implemented CNN to evaluate its classification performance with such unbalance and limited dataset provided by ImageClef 2016. VGG-like model (inspired by *VGG[11]*) with dropout technique had the highest result out of all the runs with 65.31% accuracy[13]. Although, it has proven that CNN is an essential tool for image modality classification based on the accuracy percentage, overcoming the challenge of overfitting was not accomplished based on their result. In the pre-processing stage, only one channel was

used (output activation volume with depth (or channel) of 1) which excludes RGB images and will only view the images in a greyscale value. This may hinge the overall classification result as some modality images contain coloured features. Also by not using the depth dimension, it loses the purpose of using the CNN architecture.

2.3.3 Overcome Overfitting

Supervised learning algorithm generally requires large number of training dataset for an accurate classification result; however, acquiring such volume for medical images is difficult as it must be manually annotated by a professional which is cost and time inefficient. Therefore, class imbalance and limited dataset provided will cause overfitting due to poor generalization of the training dataset. To overcome this challenge, various techniques are applied.

Data Augmentation

Data augmentation is a common technique[14] used by artificially enlarging dataset. The traditional technique used involves cropping, flipping, distorting and adding noises to each image.

Transfer Learning
Transfer learning is a process which CNN is pre-trained on a labelled natural image dataset and is used to extract its features on other datasets. The two main strategies used are 1) pre-trained CNN is used as features extractors and 2) fine-tuning CNN [15].

Kumar *et al.[16]* compared two methods using fined-tuned CNN architecture, *AlexNet* and *GoogLeNet* 1) as an image feature extractor and apply early fusion technique by concatenating with the independent feature vector that is then uses one-to-one multiclass SVM, and 2) as a classifier using softmax probability. By fusing two different CNN architecture, it has shown that shallower architectures, like AlexNet, has the ability to learn generalized features, whereas deeper architectures, *GoogLeNet*, learn semantically relevant features.

Tajbakhsh *et al. [15]* has investigated the correlation between the depth of fine-tuning and accuracy of image classifiers where various level of tuning was tested, from shallow to deep, based on fine-tuning the number of convolutional layer. As a result, it has shown that the deeper fine-tuned CNN architecture had high accuracy rate.

Increase Dataset
Another simple common technique used to enrich dataset is to source additional image by extracting previous dataset[17]. For example, some teams[18, 19] from ImageClef 2016 has concatenated with ImageClef 2013 dataset from modality image classification.

One of the main challenges in medical imaging domain was insufficient amount of labelled data. This problem can be omitted by the use of unsupervised learning algorithm. An approach that is expected to have high impact in medical imaging is using Generative Adversarial Networks (GANs)[20] for synthetic data augmentation. GANs are an implicit density estimation that can learn generative model based on a game theory. The adversarial modelling framework consists of two deep CNNs, generator G and discriminator D. The goal

is to optimize G and improve its ability to synthesize images until D cannot distinguish between real and fake images.

Frid-Adar *et al.*[21] implemented a proposed model known as deep convolutional GAN (DCGAN) where both generator and discriminator architecture is modelled using deep CNN, to improve the overall performance of CNN for medical image classification.
DCGAN was used to create synthesized data of liver lesions on CT images with classification performance of 85.7% sensitivity and 92.4% specificity whereas classic data augmentation yields 78.6%.
If GAN is to be used in image modality classification to synthesize modality images, its architecture cannot use lesion region of interest (ROI) as automatic classification of modalities also requires discriminating distinct differences.

2.3.4 Classification Accuracy

The ensemble method by Kumar *et al.[16]* resulted in top-1 accuracy of 82.48%.The only image classification task that was applied in this paper was colonoscopy frame classification where is classifies either *informative* or *non-informative.* Therefore, this method cannot prove that this method will work on image modality classification. However, further studies[22] have tested on medical modality classification using ensemble method along with improving image pre-processing by applying cropping on images on deeper CNN architecture with a top-1 accuracy of 85.55% for *Inception-ResNet-V2* and 85.33% for the *Inception-v4* network. Although, this paper has proven a higher accuracy result compared to BCSG at ImageClef 2016, adjusting pre-processing stage can be time and cost inefficient as it requires manually cropping and resizing each images.

Akilan *et al.[23]* have shown that late fusion approach has enhanced the classification accuracy by fusing high level features from three multi-deep pre-trained CNN architectures, *AlexNet, VGG-16,* and *Inception-v3,* which were then used as features extractors. The feature fusion was applied by encoding high to low dimensional CNN features by reducing the rectified feature dimensionality. Then feature fusion was applied using various method of pooling function to compare its accuracy.

2.3.5 Conclusion

From various paper reviewed in this paper, it is evident that implementing CNN architecture is a prominent technique for image modality classification. Based on our analogy, applying GoogLeNet is a suitable architecture because of 1x1 convolution layer. For example, when extracting feature of a small region (in order to detect the subtle differences between features), 1x1 convolution layer will preserve the image size while reducing the depth. Additionally, we believe that using fine-tuning method on the architecture may improve the overall classification accuracy.

We hypothesize that implementing the early fusion technique from Kumar *et al.* [16] by extracting feature dimension through different CNN architectures and using early fusion technique is suitable for image modality classification to be able to discriminate between distinct and subtle differences between image modalities.

3. Project Management Plan

1.1 Project Overview

This project is a part of our Final Year Project. Progress to complete the project will begin at the start of the second semester of 2018. The project will take place for approximately 12 weeks. There will be no budget for this project as all the resources that would be needed for this project will be provided by the university without any charge. The basis of this project is to create an ensemble of fine-tuned CNN that will be trained and used to classify various types of medical modalities. We will also be comparing the performance and accuracy of various types of fusion techniques to determine which best suits one for our imaging domain is. Towards the end of the project, we will need to present the results and observations of our project supervisor.

3.2 Project Scope

3.2.1 Project Deliverables

1. Image Modality Classification code in MATLAB for implementing:

    a. Data augmentation of training, validation and testing image sets.
    b. Fine-tuning the chosen CNN architecture.
    c. Feature extraction of images.
    d. Function of chosen method to avoid overfitting.
    e. Early and late fusion techniques.

2. Graphical User Interface (GUI) implementation in MATLAB of the code consisting of:
    a. Buttons of different functionality of the code. (e.g. pre-processing, classifying, training CNN, etc.)
    b. An axes/space to display input image.
    c. The ability to allow images to be inputted through dragging and dropping.
    d. A text area to display the predicted class of the input image to the user.
3. A project report, including the results and discussion of our implementation of the classifier.
4. A slide to briefly explain the results and discussion of our implementation of the classifier

3.2.2 Project Characteristics and Requirements

Criteria	**Description**	**Nature**
Have a user friendly GUI	Create a graphical user interface that succeeds in hiding the back end of the program from the user and is visually pleasing.	Functional
Have Drag and Drop functionality	Allows the user to drag and drop images to classify them for convenience.	Functional
Have buttons in GUI	Add buttons to perform functions that lead up to the complete CNN architecture. e.g. Pre-process, Train CNN, Classify, etc.	Functional
Visualize results from pressing buttons	Makes sure that the buttons produce a form of result/graph if needed. e.g. button to train a CNN will produce a graph, displaying its accuracy overtime.	Functional

Handle irrelevant images appropriately	Makes sure that the program is able to output "Invalid" when presented with an irrelevant image	Functional
Handle error when an input image is not chosen	Makes sure that the program is able to output "Invalid" when any functional program button is used without an input image provided	Functional
Handle errors when an input image with invalid dimensions are chosen	Makes sure that the program is able to output "Invalid" when an input of invalid dimensions are presented.	Functional
Handle errors when buttons aren't clicked on in the correct order	Makes sure the program recovers and restarts from an error if the user were to not follow the proper order the buttons are supposed to be clicked on	Functional
Add frames to visualize image	Makes sure a frame of a fixed dimension in present in the GUI to display the image that has been browsed/dragged in by the user.	Functional
Executed within memory limit	Ensure that the program can be ran within the machine memory space that has been allocate for the task.	Non-Functional
Time taken to respond	Ensure that the program does not take too long of a time to execute its specified functions for convenience.	Non-Functional
Reliability of program	Ensure that the program can classify images with a high accuracy rate.	Non-Functional
Availability of program	Ensure that the program's back end code can be accessed and edited with ease. (Properly labelling and orientating the code's design)	Non-Functional
Maintainability of program	Ensure that the program can be recovered/restarted with ease if an error/crash/unwanted outcome has occurred.	Non-Functional
Recoverability of program	Ensure to constantly save progress as program is being developed and storing the saved data in a backup storage in case any loss of progress occurs.	Non-Functional

3.2.3. Product user acceptance criteria

1. The user has the ability to upload data, where the system will determine whether the data is valid or invalid.
2. User has the choice to drag and drop images into the user interface to be used as input.
3. An axes/blank space for the image that was uploaded to be displayed to the user.
4. Variety of buttons that executes functions in the back end/code of the system.
5. User will be notified if buttons were clicked on in the wrong order of how the implementation is supposed to work. (e.g Clicking on “Classify” button before the “Pre-process” button.
6. User will be able to see the predicted class output of the system through a text area in the user interface.

3.3. Project Organisation

3.3.1. Process Model

Agile project management is a recent trend that is affecting IT project management. The approach is iterative and incremental to refine work items and deliver frequently. Unlike the traditional approach, this requires dynamic activities that is repeated until it is correct. Since this project has relatively high uncertainty of the outcome, an adaptive approach will be used

as we will iteratively be making robust decisions in the face of uncertainty, with an aim of reducing uncertainty as time progresses.

3.3.2. Project Responsibilities

All of this has been done in *Figure 1* in the Appendix that is attached to this document in the form of a Requirements Traceability Matrix. Do note however, that we are unsure of who will be responsible for each section as we have not started actually coding the program yet. This table may face changes in the future when the second semester starts as we will have a better clue on who will work on which section.

3.4. Management Process

3.4.1. Risk Management

The first step that we would take is to identify all possible and logical risks that could occur while developing the network. After doing so, we would then analyse each and every risk that have been identified to determine the root of its cause. This will then give us a better understanding the nature of the risks. We will then proceed to rank the risks by determining the risk magnitude, which is a combinational score of the likelihood of it from occurring and its consequences. By doing so, we can figure out which risks are very minor and which ones are very serious to warrant treatment. We then assess our highest ranking risks and come up with a plan to treat these risks to obtain acceptable risk levels. We can achieve this by creating risk mitigation strategies and prevention plans. All of the deliverables from the mentioned steps should be included in a project risk register. We can then utilize the project risk register to monitor over and track risks in our project, effectively minimizing the chances of risks of occurring through the project and reducing the consequences of a risk if it were to ever occur. We have created a risk register for our project and it is in the Appendix attached to this document labelled *Figure 2*.

3.4.2. Monitoring and Controlling Mechanisms

3.4.2.1. Communication and reporting plan

We have listed all major project functions and its details in *Figure 3* in the Appendix that is attached to this document in the form of a Communication Table. All of the major steps and deliverables in our project have been listed along with the method of documentation. We plan to mainly document most of the deliverables using Google Drive documents as they can be shared and edited easily by multiple members. It is also a cloud storage to ensure that our data is secured. Most of the project deliverables that require coding will be done in the MATLAB software. We are still unsure of who will handle each major section, which explains the reason for why the whole “Who is Responsible” section is just filled with “Project Team”. The schedules are also just rough estimations of how long we will spend to accomplish each major section of the project. All of the information in the attached Communication Table may vary in the future.

3.4.2.2. Review and audit mechanisms

For version control, we will use GitHub. We will commit our progress fairly often to keep track of our project version. It will also act as a backup storage for our code in case any file

corruptions occur during the duration of this project. As for quality assurance, we will make sure to have several meetings to review each other's individual work to check for any mistakes that may have been made such as spelling errors or any presence of bugs in the codes. Documentation on the codes will be done by adding comments next to any line of code that is necessary whereas documentation on everything else, such as progress reports or methodology changes will be documented in a word document in a shared Google Drive.

3.5. Schedule and Resource Requirements

3.5.1. Schedule

We have done this in Appendix *Figure 4* and *Figure 5* in the form of a Work Breakdown Structure along with its Gantt Chart with its dependencies in the Appendix that is attached to this document. We have also labelled the work packages of this project in a parenthesis next to the task's name. Do note however, that the schedule is very uncertain as the semester has not started and we cannot predict how long each and every task will take. For the Gantt Chart that is provided, we assumed that every task takes 2 days to accomplish.

3.5.2. Resource Requirements

We will require the MATLAB software license on the provided computer in order to code and execute our program. We will also need the computer that will be provided with to have sufficient memory space in order to run our program. The computer will also need to have a decent or better GPU as the whole overall process of our program is rather computationally expensive, meaning that the program will take a very long time to execute if the GPU is weak. We will also require a data set to work with in order to train, validate and test our program's classification accuracy.

4. External Design

4.1 ImageClef 2013 Database

In our study, we have considered 31 different image modality from subfigure classification dataset from ImageClef 2013 Collection. Medical task in ImageClef is an ongoing evaluation campaign that is a part of the Conference and Labs of the Evaluation Forum (CLEF) initiated since 2004. The class codes with descriptions are show in *Figure 1*:

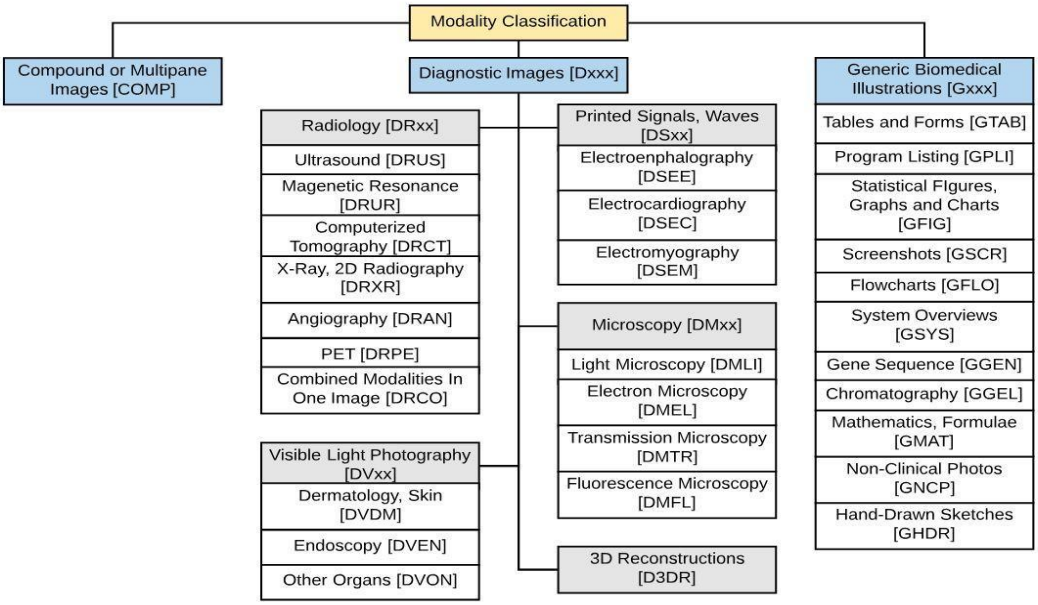


Figure 1: Class hierarchy for modality classification task (derived from [22])

The dataset that will be used to train the CNN is skewed and isn't even. The class with the lowest number of sample images is Mathematics and Formulaes [GMAT] with 20, which is 0.68% of the whole data set, while the class with the highest number of sample images is Compound or Multiplane Images [COMP] with 1105, which is 38.3% of the whole data set. Furthermore, the data set consists of 31 classes in total, which further proves that the training data set that has been provided is significantly skewed.

4.2 User Interface

The final product will be represented on GUI in MATLAB for:

- Visualization
- Simple implementation
- Classification of the image

The simple UI is shown in *Figure 2* which contains 4 functions:

1. Train CNN

The system will initiate the training process with the specified dataset. Once the training has been finished, a notification panel will appear showing "Training Finished". During the training:

- Progress of training will show according to number of epochs
- Estimated time of completion
- Graphical representation of training error

The user will also have an option to halt the training

2. Browse

The input data is loaded onto the system in order for the image to be classified

3. Pre-process

The image will resize according to the required size

4. Classify

After the system has predicted the class of the image, the result of the specified class will appear in the textbox “Predicted Class”

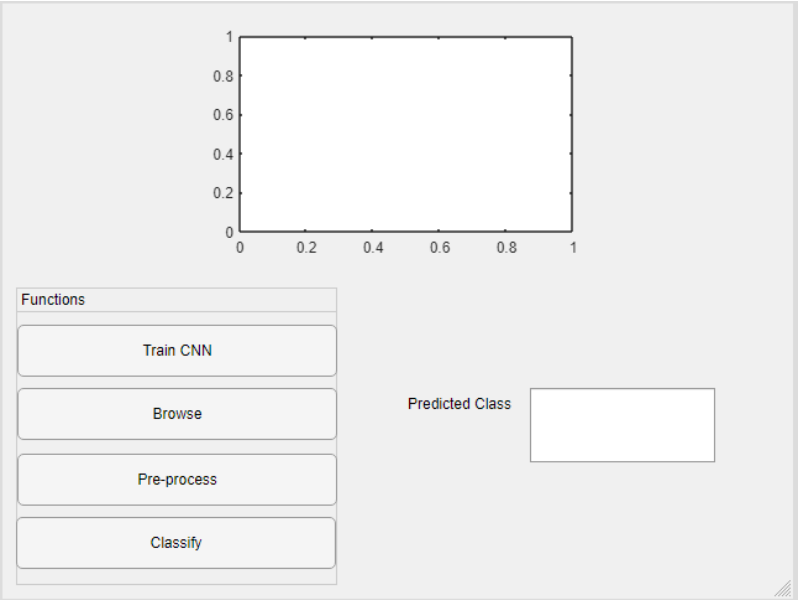


Figure 2: Simple User Interface

4.3 Performance

In order to enhance the accuracy of CNN, it requires a large increase in computation; therefore, general-purpose CPU does not become sufficient due to its lack of computational parallelism. The execution environment (GPU, multi-GPU, CPU and parallel) can be selected using`trainingOptions`. CNN is computationally expensive process due to the large volume of parameters and data in the network. Therefore, methods to parallelize and distributed algorithms becomes an essential technique to improve the time complexity:

4.3.1 Parallelism

- Multi-core CPU: assume that the network and dataset can fit into the memory of single machine with multiple cores. This can increase the speed of training process by 1) using multiple images per layer and 2) perform SGD of multiple mini-batch.
- GPU: used for computationally expensive task. The `trainNetwork` function also uses GPU by default, if available. This can be checked using `gpuDevice` and can specify its execution environment using `trainingOption` function.
- Clusters of computers with multiple CPUs and GPUs: where any computationally expensive task will use GPU.

4.3.2 Distributed Computing

- Data parallelism: data is distributed on multiple machines. This can be achieved in MATLAB by using the `train` function and setting `sim` parameter `'useParallel'` to `yes.`
- Model parallelism: model can be split into multiple machines

5. Methodology

5.1 Toolchains and Approaches

The algorithm that we have developed for this study will be implemented in MATLAB using its development tools and high-level language functions. In order to create, train and simulate the neural network, *Neural Network Toolbox* will be required which contains algorithms and pretrained models. To increase the processing speed,*Parallel Computing Toolbox* can be used to increase performance and training speed.

When implementing CNN for computer vision application *MatConvNet* can be used which is from MATLAB toolbox. By downloading the CNN library from *MatConvNet,* it achieves optimization of GPU computation. *MatConvNet* supports NVIDIA GPU which includes CUDA implementations. Wrappers and pre-trained models are also provided in the library; wrappers can use CNN architecture using SimpleNN for basic chains of blocks and DagNN for complex directed acyclic graphs(DAG). `v1_simplenn`function, for example is a SimpleNN that can be used for pre-training the model.

For simplicity, we will be training and fine-tuning AlexNet architecture for our study. The pretrained AlexNet has been trained from approximately a million images and is able to classify into 1000 classes. These training images are a subset of ImageNet database, which is used in ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). The pre-trained AlexNet network can be downloaded using alexnetfunction.

5.2 Stages

5.2.1 Load Data

The images can be unzipped and loaded on the to the datastore `imageDatastore`which will be stored as an object. The datastore is then divided into 70% training and 30% test data using `splitEachLabels` where it will individually be individually stored as datastores.

5.2.2 Data Pre-processing

The first layer of pretrained AlexNet is 227 x 227 x 3, where the input size of the image is 227 x 227 with an RGB color channel. Therefore, the new data must adjust to the required input size of the network. The image can be rescaled using imresize function.

5.2.3 Data Augmentation

To increase the dataset, data augmentation will be used using `augmentedImageDatastore.`To create a pre-processing option for data augmentation,

`imageDataAugmenter;` the options can be specified using different set properties for transformation:

reflection	`RandXReflection` and `RandYReflection`
rotation	`RandRotation`
translation	`RandXTranslation` and `RandYTranslation`

The `augmentedImageDatastore` is then used to read batches of augmented images to improve the efficiency of the process as it extracts random combination of transformation per batch for each iteration from the training data.

5.2.4 Fine-Tuning

Fine-tuning initially starts by transferring weights from the pre-trained AlexNet network.

layer	type	input	kernel	stride	pad	output
data	input	3 x 277 x 227	N/A	N/A	N/A	3 x 277 x 277
conv1	convolution	3 x 277 x 227	11 x 11	4	0	96 x 55 x 55
pool1	max pooling	96 x 55 x 55	3 x 3	2	0	96 x 27 x 27
conv2	convolution	96 x 27 x 27	5 x 5	1	2	256 x 27 x 27
pool2	max pooling	256 x 27 x 27	3 x 3	2	0	256 x 13 x 13
conv3	convolution	256 x 13 x 13	3 x 3	1	1	384 x 13 x 13
conv4	convolution	384 x 13 x 13	3 x 3	1	1	384 x 13 x 13
conv5	convolution	384 x 13 x 13	3 x 3	1	1	256 x 13 x 13
pool5	max pooling	256 x 13 x 13	3 x 3	2	0	256 x 6 x 6
fc6	fully connected	256 x 6 x 6	6 x 6	1	0	4096 x 1
fc7	fully connected	4096 x 1	1 x 1	1	0	4096 x 1
fc8	fully connected	4096 x 1	1 x 1	1	0	31 x 1

Table 1: AlexNet architecture used for our experiment where ‘fc8’ adjusts to 31 number of classes after feature extraction

The last fully connected layer will be replaced with a new fully connected layer, since the number of layer correlates with the number of classes required in the dataset. In our study, we will be dealing with 31 different modality classes; hence, the new fully connected layer will contain 31 neurons. This method can be implemented on MATLAB by extracting all layers excluding the last three layers by setting a range:

```
transferLayer = net.Layer(1:end-3)
```

The number of classes is then set to `numClasses = 5` and is then replaced with `layer = [ transferLayer, fullyConnectedLayer(numClasses,`

```
'WeightLearnRateFactor', 25, 'BiasLearnRateFactor',25)

softmaxLayer classificationLayer  ];
```

The training option also needs to be specified such as mini-batch size and validation data to calculate its training speed per iterations during training.

5.2.5 Feature Extraction

Feature extraction is computationally feasible and requires only few data to obtain reasonable result; a selected deep layer can be easily extracted on MATLAB `activations`method.

5.2.5.1 Multi-class classification with SVMs

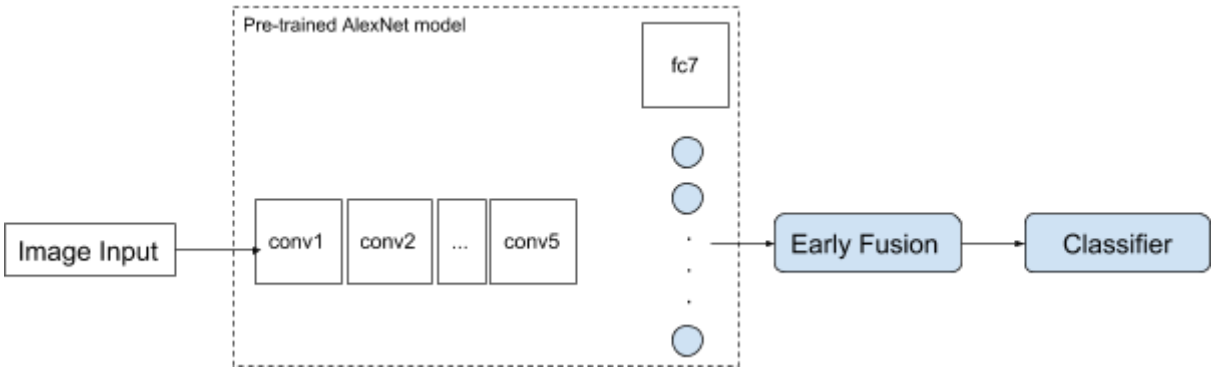
Support Vector Machine (SVM) is a popular technique for image classification in medical domain. It defines the details of the loss function by designing a hyperplane that classifies all the training vectors into two classes. The basic concept of SVM is to find the balance between regularization term and the training errors.

For this study, we will be implementing one-against-one method as it is more suitable for practical use than one-against-all method.[Medical image retrieval with probabilistic...] The one-against-one method calculates binary SVMs of all possible pairs of classes which constructs *N(N-1)/2* classifiers; therefore, the space complexity is $O(N^2)$. Whereas, one-against-all method is *O(N)* as it calculates a selected class against all the other classes that is separated by the hyperplane. Although one-against-one method has higher time complexity, we still be using this method for its higher classification accuracy (less error rate).

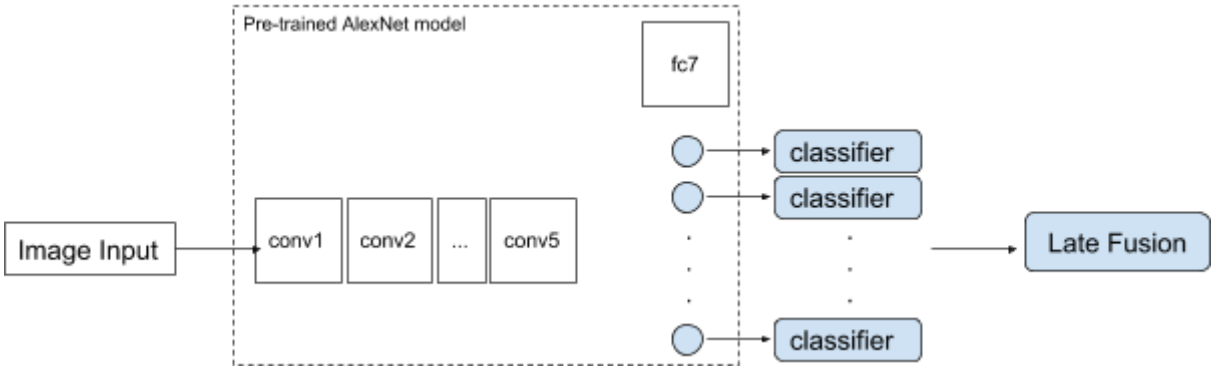
5.2.5.2 Fusion Technique

There are two different fusion techniques that can be used to train the SVM classifier shown in *Figure 3*:

1. Early fusion: fusing all the features extracted from CNN into single feature vectors which then used an input of the classifier to determine the class of the data.
2. Late fusion: each feature is fed into the classifier to obtain the classification score and is then fused to get the final score using mean, max, min and weight sum. This can be used in MATLAB using `mean, max, min` and `sum` function.



a) Early Fusion Approach



b) Late Fusion Approach

Figure 3: Fusion Technique on fine-tuned AlexNet

The features will be extracted from 'fc7' fully connected layer, prior to the final fully connected layer, which contains complex high level features. This can be extracted on MATLAB using:

```
featureExtractionLayer = 'fc7';

featureTraining = activations(convnet, trainingSet, featureExtractionLayer)    //
returns activations of the specified layer using network 'convnet' and data
'trainingSet'
```

Then to train the SVM classifier `fitcecoc` function is used which is a fit multiclass SVM model (or other classifier) which trains an error-correcting output codes (ecoc). The two parameters indicates the predictors and the class labels.

```
classifier = fitcecoc(featureTraining, trainingSet.Labels)
```

5.2.6 Improve Generalization and Avoid Overfitting

One of the main challenges that we assume that we will face, as mentioned previously, is overfitting. When there is a drastic difference in error value between training set and validation set, it identifies the network to be overfitting the data and has the inability to generalize the data. There are several techniques to avoid overfitting, but this section will be solely based on the simple techniques that can be used in MATLAB.

5.2.6.1 Retrain Network

Training several networks can ensure an accurate network generalization by taking the mean square error of each network to find the average accuracy. This can be implemented on MATLAB:

```
net = alexnet(20); // train network 10 times numberNN = 20

NN = cell(1, numberNN);

performance = zeros(1, numberNN) //set all the 20 values to
zero
```

```
for i = 1:numberNN      fprintf('Training %d/: %d\n', i, numberNN);   //training the first part of the dataset

NN{i} = train(netx1, t1);    y2 = NN{i}(x2);

  // calculate mean square error in second part of the dataset

      performance(i) = mse(net, t2, y2);

End
```

By using this method, each network will have different initial weight and biases with different ratio of training, validation and testing sets.

5.2.6.2 Multiple Neural Networks

Averaging the total output of several networks can also improve network generalization. It is similar to 5.2.6.1 *Retrain Network* except the performance of each tested network is calculated separately:

```
      performance = zeros(1, numberNN); y2Total = 0;

      for i = 1:numNN   neti = nets{i};   y2 = neti(x2);   performance(i) = mse(neti, t2, y2);  y2Total = y2Total + y2; end performance y2AveOutput = y2Total / numberNN; perfAveOutputs = mse(nets{1}, t2, y2AveOutput)
```

It is predicted that the overall performance is lower than individual performances.

5.2.6.3 Early Stopping Approach

If the validation error (as well as training set error) increases during the initial phase of training process, it is a sign of overfitting. If this occurs, the training will stop at a specific number of iterations (`net.trainParam.max_fail`) and returns the minimum value of weights and biases from the validation error. It is suggested that the training method should be slow as the network will use more parameters. If a minimum of test set error reaches before validation set error, it may indicate poor selection of validation set; hence poor division of dataset. This can be improved by changing the division function such as `dividerand`, `divideblock`, `divideint` and `divideind`; this can be altered by accessing into the division function `net.divideFcn`.

5.2.6.4 Freeze Initial Layers

By setting the learning rates to zero on the initial layers, the `trainNetwork`function will not compute the gradient of those layers; hence, will not update the parameters during the training process. By implementing this method, it may decrease the time complexity of the training process as well as avoid overfitting the new data. The freezing method can be implemented on MATLAB using `freezeWeight` to set the learning rates of selected range of layers to zero. Then, `createLgraphUsingConnection` function is used to reconnect the layers.

5.2.6.4 Regularization Approach

Regularization is also another method to improve generalization by modifying the loss function. The performance function can be identified through calculating the squared hinge loss SVM of the network. This function is further improved by extending using regularization *R(W)* which calculates the sum of all squared elements of *W*. This can be adjusted by multiplying the value with a hyperparameter λ to adjust the width of the margin:

$$L = \frac{1}{N}\sum_i^N L_i + \lambda R(W) \quad (1)$$

Where the average loss of L_i is data loss and $\lambda R(W)$ is the regularization loss. By implementing regularization, if there are same dot product with different weight vectors, the smaller and more diffused weight will be chosen.

In MATLAB, regularization performance can be adjusted using the performance ratio `net.performParam.regularization = 0.5` when retraining the previous network. The performance ratio parameter can be adjusted to identify the optimum value.

Rather than re-initializing the previous network, we can also use a standard method of regularization by using weight decay. L2 regularization is the most common regularization technique which penalizes the square magnitude of all weights in the objective function; where regularization term $R(W) = \sum w_i{}^2$ in equation (1):

$$L = \frac{1}{N}\sum_i^N L_i + \frac{\lambda}{2m} R \sum w_i{}^2 \quad (2)$$

This indicates that the every parameter w is decayed linearly towards zero for each training iteration. This can also be implemented in MATLAB using `setL2Factor`.

Bayesian Regularization

Bayesian regularization using Levenberg-Marquardt optimization calculates the probabilistic interpretation of network parameter which updates the weight and bias values. Although early stopping is one of the techniques to use for complexity control, it causes the bias to increase; this can be controlled using Bayesian regularization. Bayesian regularization network adds additional hyperparameters $\alpha$and β to equation (2):

$$L = \beta \frac{1}{N}\sum_i^N L_i + \alpha \frac{\lambda}{2m} R \sum w_i{}^2$$

The ratio β/α penalizes large weights; if $\alpha << \beta$, it will make small errors whereas if $\alpha >> \beta$, it will try to reduce the weight size. The posterior distribution can be then updated using probabilistic framework of David MacKay. The Bayesian regularization can be implemented in MATLAB using`trainbr` function where the inputs and targets falls in the range [-1,1]. If it exceeds the range, `mapminmax` or `mapstd` function can be used.

The advantages of using Bayesian regularization is 1) it will decrease the bias values and 2) it will have better performance as it does not require any subdivision of validation and training

set. However, due to the increase in the hyperparameter, there is a possibility of leading to overfitting. Hence, this method is still an option which may be used for this project.

Dropout Technique

Performing a dropout is another simple regularization technique by temporarily removing the neuron and thinning the network, where the neurons are chosen at random. This allows the number of parameters to decreases which increases the time complexity significantly. This can be applied on MATLAB using`dropoutLayer`.

5.3 Evaluation Approach

During the training process, a figure will display the training metrics per iteration. The network training can be specified using `trainingOptions`. The figure will plot:

- Training accuracy
- Smoothed training accuracy
- Validation accuracy
- Training loss, smoothed training loss, and validation loss

In the classification stage, the performance of a classifier will be analysed using confusion matrix using an in-built function `confusionmat`. Using this function we will identify all the number true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). Using the confusion matrix we will compute the specificity, accuracy and sensitivity rate.

Train-Test split Approach

In order to obtain high classification accuracy without overfitting, it needs to have a good measurement of generalization of the classifier. To achieve this, it is essential to avoid tuning the hyperparameters in the test set as it may significantly reduce its performance. In order to tune the hyperparameters, the training set can be subdivided into a training set and a validation set. The training set will be used for computing the gradient and adjusting the weights and biases. The validation set will be processed during the training process which is used for calculating the total validation error. It is assumed that the decrease in validation error means that the network is not overfitting the data. In MATLAB the main data division function that is used for training CNN is the function `dividerand`which divides the data randomly. The default ratio for training: testing: validation is 0.7 : 0.15 : 0.15 respectively; hence, based on our approach, we can retain its default value.

Cross-Validation Approach

We will be implementing cross-validation for hyperparameter tuning. Although it is computationally expensive, it is essential to use this technique for our model due to the skewed dataset. By using this approach, it will produce an estimation of the overall performance of the model. Cross-validation can be used in MATLAB using the `crossvalind` function.

Test Planning

6.1 Test coverage

The requirement for the test coverage criteria is to ensure every possible outcomes at least once. By validating all the branches/decision in the code, it ensures the efficiency of the code. The test coverage we will be using is decision coverage. The validity of the input data will be verified where system can detect valid and invalid input data. For example, if the input does not contain any data or the size and type of the image do not fit the requirement, the system will detect as 'invalid'. We will use decision coverage to make meet the criteria of meeting every point has been invoked at least once.

6.2 Test methods

Testing and evaluating on the classifier feature can be done using MATLAB in-built function where an example is shown below:

Firstly, set up test data:

```
folder = 'testData' testSet = imageDatastore(fullfile(folder,
categories), 'LabelSource', 'foldernames;);

testSet.ReadFcn = @readFunctionTrain;
```

The system will then extract features the test data and classifier:

```
featureTest = activations(net,testSet, featureExtractionLayer);
predictLabels = predict(classifier, featureTest);
```

Then, the system will predict the overall classification accuracy using:

```
confusionM = confusionmat(testSet,Labels, predictLabels);

confusionM = confusionM./sum(confusionM, 2);
mean(diag(confMat))
```

Then, the new trained model can be tested on test data:

1. Load a new image using `imread` function
2. Resize the image according the input size using `augmentedImageDatastore` function
3. Feature extraction using `activations` function
4. Classify the image using `predict` function

6.3 Sample Test Cases References

The test data that will be collected will be from subfigure classification dataset from ImageClef 2013 collection. The dataset provided contains 31 different classes with 2,597 images. There is a slight difference in class distribution between training and test data. For

example, GSCR in training set contains 0.77% whereas the testing set contains 3.13%. This may be something we need to be considering as it may have an effect when comparing the accuracy result for training and testing.

References

1. Kumar, A., et al., *Content-Based Medical Image Retrieval: A Survey of Applications to Multidimensional and Multimodality Data.* Journal of Digital Imaging, 2013. **26**(6): p. 1025-1039.
2. Chang, N.-S. and K.-S. Fu, *Query-by-pictorial-example.* IEEE Transactions on Software Engineering, 1980(6): p. 519-524.
3. Hubel, D.H. and T.N. Wiesel, *Receptive fields of single neurones in the cat's striate cortex.* The Journal of physiology, 1959. **148**(3): p. 574-591.
4. Marr, D., *A theory of cerebellar cortex.* The Journal of physiology, 1969. **202**(2): p. 437-470.
5. Russakovsky, O., et al., *ImageNet Large Scale Visual Recognition Challenge.* International Journal of Computer Vision, 2015. **115**(3): p. 211-252.
6. Khan, S. and S.-P. Yong. *A comparison of deep learning and hand crafted features in medical image modality classification*. in *Computer and Information Sciences (ICCOINS), 2016 3rd International Conference on*. 2016. IEEE.
7. Ergun, H., et al., *Early and Late Level Fusion of Deep Convolutional Neural Networks for Visual Concept Recognition.* International Journal of Semantic Computing, 2016. **10**(03): p. 379-397.
8. Krizhevsky, A., I. Sutskever, and G.E. Hinton. *Imagenet classification with deep convolutional neural networks*. in *Advances in neural information processing systems*. 2012.
9. Szegedy, C., et al. *Going deeper with convolutions*. 2015. Cvpr.
10. Lin, M., Q. Chen, and S. Yan, *Network in network.* arXiv preprint arXiv:1312.4400, 2013.
11. Simonyan, K. and A. Zisserman, *Very deep convolutional networks for large-scale image recognition.* arXiv preprint arXiv:1409.1556, 2014.
12. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
13. Semedo, D. and J. Magalhães. *NovaSearch at ImageCLEFmed 2016 Subfigure Classification Task*. in *CLEF (Working Notes)*. 2016.
14. Kumar, A., et al. *Subfigure and Multi-Label Classification using a Fine-Tuned Convolutional Neural Network*. in *CLEF (Working Notes)*. 2016.
15. Tajbakhsh, N., et al., *Convolutional neural networks for medical image analysis: Full training or fine tuning?* IEEE transactions on medical imaging, 2016. **35**(5): p. 1299-1312.
16. Kumar, A., et al., *An ensemble of fine-tuned convolutional neural networks for medical image classification.* IEEE journal of biomedical and health informatics, 2017. **21**(1): p. 31-40.
17. Seco, G., et al., *Overview of the ImageCLEF 2016 medical task.* Working Notes of CLEF, 2016.
18. Koitka, S. and C.M. Friedrich. *Traditional Feature Engineering and Deep Learning Approaches at Medical Classification Task of ImageCLEF 2016*. in *CLEF (Working Notes)*. 2016.
19. Valavanis, L., S. Stathopoulos, and T. Kalamboukis. *IPL at CLEF 2016 Medical Task*. in *CLEF (Working Notes)*. 2016.
20. Goodfellow, I., et al. *Generative adversarial nets*. in *Advances in neural information processing systems*. 2014.

21. Frid-Adar, M., et al., *GAN-based Synthetic Medical Image Augmentation for increased CNN Performance in Liver Lesion Classification.* arXiv preprint arXiv:1803.01229, 2018.
22. Koitka, S. and C.M. Friedrich. *Optimized Convolutional Neural Network Ensembles for Medical Subfigure Classification*. in *International Conference of the Cross-Language Evaluation Forum for European Languages*. 2017. Springer.
23. Akilan, T., et al. *A late fusion approach for harnessing multi-cnn model high-level features*. in *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*. 2017. IEEE.

8325 words

Appendix

Figure 1 (Requirements Traceability Matrix).

Requirements Traceability Matrix				
Project Name:	Final Year Project			
Project Team Members:	Moana and Bryan			
Project Description:	The main objective of this project is to create a CNN architecture using fusion techniques and finetuning to classify medical modalities. However, the final product would be to			
	create a program with a simple user interface with buttons that map to the functions that have been implemented in the back end.			
ID	Requirements	Assumptions	Category	Source
1	Able to drag and drop images	User should be able to drag and drop images into the interface	Functional	Project Team
2	User friendly user interface	There will be a easy-to-us user interface to use the program	Functional	Project Team
3	Have buttons	User interface will have buttons to perform functions	Functional	Project Team
4	Visualize results from pressing buttons	Extra windows will pop up to visualize data if needed. e.g. "Train CNN" button	Functional	Project Team
5	Handle irrelevant images appropriately	Program will output "Invalid" when an image out of the imaging domain is inputted when "Classify button is pressed	Functional	Project Team
6	Handle error when an input image is not chosen	Program will output "Invalid" when no image is inputted for any functional button	Functional	Project Team
7	Handle errors when an input image with invalid dimensions are chosen	Program will output "Invalid" when an image of invalid size is attempted to be inputted	Functional	Project Team
8	Handle errors when buttons aren't clicked on in the correct order	Program will restart/recover with an error message if the buttons are clicked on in the wrong order	Functional	Project Team
9	Add frames to visualize image	User interface will have a blank space to act as a frame for the inputted image	Functional	Project Team
10	Executed within memory limit	The program will be ran within the machine memory space that has been allocate for the task.	Non-Functional	Project Team
11	Time taken to respond	The program will not take too long of a time to execute its specified functions for convenience.	Non-Functional	Project Team
12	Reliability of program	The program will classify images with a high accuracy rate.	Non-Functional	Project Team
13	Availability of program	The program's back end code can be accessed and editted with ease	Non-Functional	Project Team
14	Maintainability of program	The program can be recovered/restarted with ease if an error/crash/unwanted outcome has occured.	Non-Functional	Project Team
15	Recoverability of program	Project team will save progress as program is being developed and storing the saved data in a backup storage	Non-Functional	Project Team

Figure 2 (Risk Register).

No.	Rank	Risk	Description	Category	Triggers	Root Cause	Potential Responses	Risk Owner	Probability	Impact	Score
1	5	Human error	Code error	internal -people		Miscommunication	Re-evalue and proof read the codes	Team	medium	low	15
2	4	Competitor threat	Unexpected same topic published by a researcher	external		Unavoidable	Make small changes to approach	Team	low	high	21
3	1	Scope	Misunderstand scope of the project	internal-people		Miscommunication	Re-evaluate the function of the application	Team	medium	high	35
4	3	Time inefficiency	Error in time estimation	internal-people	Realization of lack of time towards the deadline	Miscalculation of time allocation	Hold emergency meeting to address isse	Team	medium	medium	25
5	4	Lack of resource	Resource unavilability	internal		Lack of planning and reseraching the availability of resouces provided	Use a different method	Team	low	high	21
6	2	Malware infection in the system	Vulnerability of the system	external	System malfunction	Lack of security	High security management	Team	low	Very high	27
7	5	Inadequate planning	Lack of understanding of the project deliverables	internal-people	Problems during end-of-project meeting	Miscommunication		Team	medium	low	15

Legend	Numerical Scale:
Very low	1
low	3
medium	5
high	7
Very high	9

Figure 3 (Communication Table)

Delivery Info (What?)	Recipients (Who?)	Delivery Methods (How?)	Schedule (When)	Who's Responsible (Owner)
Features Extraction	Project Team and Project Supervisor	MATLAB and Team Meetings	Week 1-2	Project Team
Classification	Project Team and Project Supervisor	MATLAB and Team Meetings	Week 3-5	Project Team
Validation	Project Team and Project Supervisor	MATLAB and Team Meetings	Week 6-8	Project Team
Documentation	Project Team and Project Supervisor	Team Meetings and Google Drive Documents	Week 8-11	Project Team
Presentation	Project Team and Project Supervisor	Team Meetings and Google Drive Documents	Week 12	Project Team
Status Reports	Project Team and Project Supervisor	Team Meetings and Google Drive Documents	Twice every week	Project Team
Review Work Done	Project Team and Project Supervisor	Team Meetings and Google Drive Documents	Once every week	Project Team

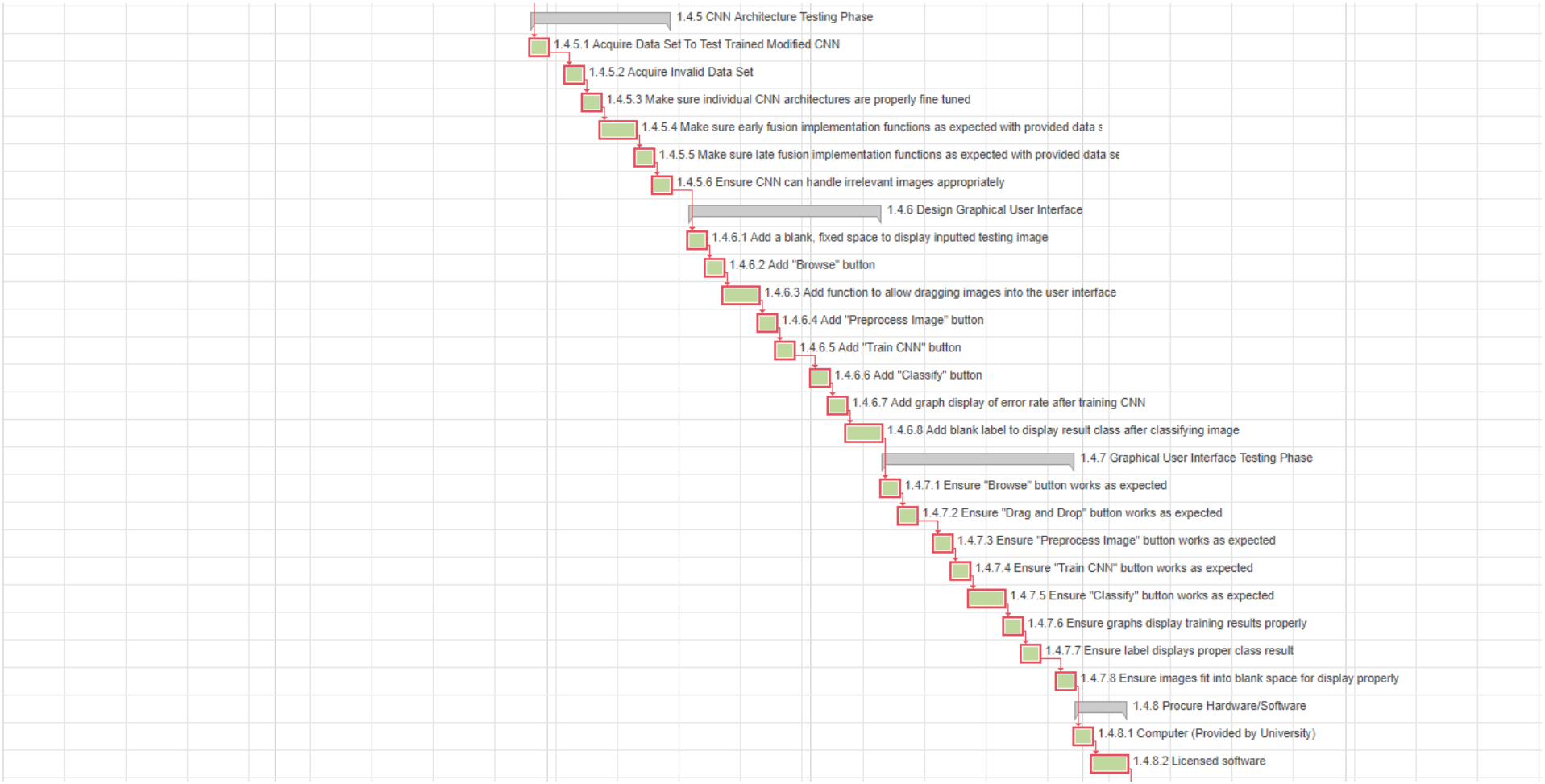
Figure 4 (Work Breakdown Structure)

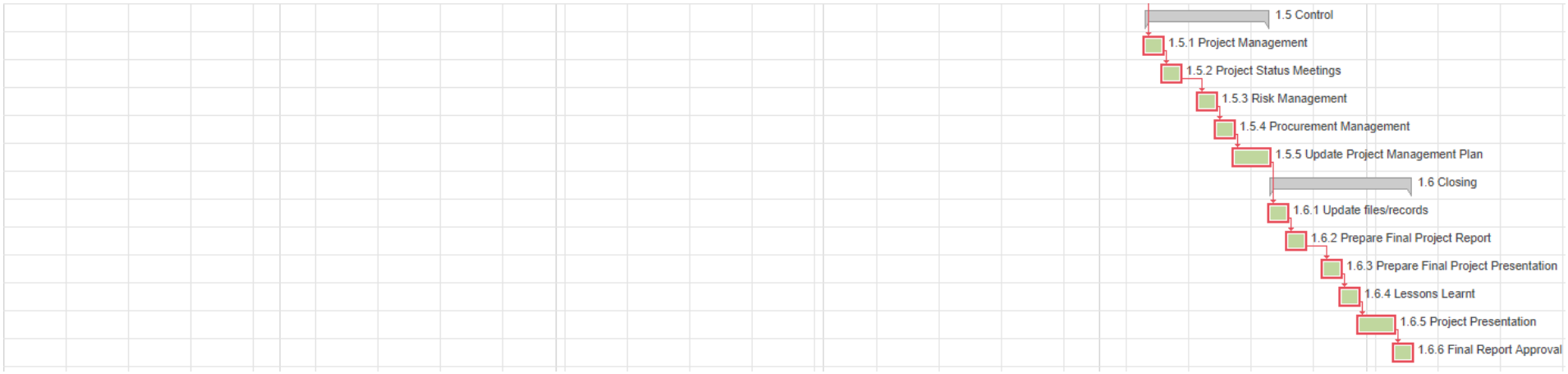
Task Name	Duration	Start	Finish	Predecessor	Assigned To	% Complete
1.0 Final Year Project	114d	07/01/18	12/05/18		Project Team	0%
1.1 Initiation	4d	07/01/18	07/04/18		Project Team	0%
1.1.1 Select Project Team	2d	07/01/18	07/02/18		Project Team	0%
1.1.2 Select Project Topic	2d	07/03/18	07/04/18	3	Project Team	0%
1.2 Planning	6d	07/05/18	07/12/18		Project Team	0%
1.2.1 Develop Scope Statement	2d	07/05/18	07/06/18	4	Project Team	0%
1.2.2 Project Team Meeting	2d	07/05/18	07/06/18	4	Project Team	0%
1.2.3 Create RTM	2d	07/09/18	07/10/18	7	Project Team	0%
1.2.4 Create WBS Gantt Chart	2d	07/11/18	07/12/18	8	Project Team	0%
1.3 Research	8d	07/13/18	07/24/18		Project Team	0%
1.3.1 Learn about CNNs	2d	07/13/18	07/16/18	9	Project Team	0%
1.3.2 Learn about Fine-Tuning on CNNs	2d	07/17/18	07/18/18	11	Project Team	0%
1.3.3 Learn about early fusion used in CNNs	2d	07/19/18	07/20/18	12	Project Team	0%
1.3.4 Learn about late fusion used in CNNs	2d	07/23/18	07/24/18	13	Project Team	0%
1.4 Executing	74d	07/25/18	11/05/18		Project Team	0%
1.4.1 Project Kickoff Meeting	2d	07/25/18	07/26/18	14	Project Team	0%
1.4.2 Concept	8d	07/27/18	08/07/18		Project Team	0%
1.4.2.1 Define requirements	2d	07/27/18	07/30/18	16	Project Team	0%
1.4.2.2 Define user requirements	2d	07/31/18	08/01/18	18	Project Team	0%
1.4.2.3 Define content requirements	2d	08/02/18	08/03/18	19	Project Team	0%
1.4.2.4 Define system requirements	2d	08/06/18	08/07/18	20	Project Team	0%
1.4.3 CNN Architecture Design	6d	08/08/18	08/15/18		Project Team	0%
1.4.3.1 Decide on what CNN architectures to use to fuse	2d	08/08/18	08/09/18	21	Project Team	0%
1.4.3.2 Decide on what fusion approach to use	2d	08/10/18	08/13/18	23	Project Team	0%
1.4.3.3 Decide on extra features/methods to implement	2d	08/14/18	08/15/18	24	Project Team	0%
1.4.4 CNN Architecture Development	10d	08/16/18	08/29/18		Project Team	0%
1.4.4.1 Acquiring CNN Architectures	2d	08/16/18	08/17/18	25	Project Team	0%
1.4.4.2 Acquiring Data Sets for training.	2d	08/20/18	08/21/18	27	Project Team	0%
1.4.4.3 Fine-tuning necessary CNNs.	2d	08/22/18	08/23/18	28	Project Team	0%
1.4.4.4 Creating error rate graphs to display after CNN training is complete	2d	08/24/18	08/27/18	29	Project Team	0%
1.4.4.5 Implementing fusion techniques to the CNN architecture	2d	08/28/18	08/29/18	30	Project Team	0%

1.4.5 CNN Architecture Testing Phase	12d	08/30/18	09/14/18		Project Team	0%
1.4.5.1 Acquire Data Set To Test Trained Modified CNN	2d	08/30/18	08/31/18	31	Project Team	0%
1.4.5.2 Acquire Invalid Data Set	2d	09/03/18	09/04/18	34	Project Team	0%
1.4.5.3 Make sure individual CNN architectures are properly fine tuned	2d	09/05/18	09/06/18	35	Project Team	0%
1.4.5.4 Make sure early fusion implementation functions as expected with provided data sets	2d	09/07/18	09/10/18	36	Project Team	0%
1.4.5.5 Make sure late fusion implementation functions as expected with provided data sets	2d	09/11/18	09/12/18	37	Project Team	0%
1.4.5.6 Ensure CNN can handle irrelevant images appropriately	2d	09/13/18	09/14/18	38	Project Team	0%
1.4.6 Design Graphical User Interface	16d	09/17/18	10/08/18		Project Team	0%
1.4.6.1 Add a blank, fixed space to display inputted testing image	2d	09/17/18	09/18/18	39	Project Team	0%
1.4.6.2 Add "Browse" button	2d	09/19/18	09/20/18	41	Project Team	0%
1.4.6.3 Add function to allow dragging images into the user interface	2d	09/21/18	09/24/18	42	Project Team	0%
1.4.6.4 Add "Preprocess Image" button	2d	09/25/18	09/26/18	43	Project Team	0%
1.4.6.5 Add "Train CNN" button	2d	09/27/18	09/28/18	44	Project Team	0%
1.4.6.6 Add "Classify" button	2d	10/01/18	10/02/18	45	Project Team	0%
1.4.6.7 Add graph display of error rate after training CNN	2d	10/03/18	10/04/18	46	Project Team	0%
1.4.6.8 Add blank label to display result class after classifying image	2d	10/05/18	10/08/18	47	Project Team	0%
1.4.7 Graphical User Interface Testing Phase	16d	10/09/18	10/30/18		Project Team	0%
1.4.7.1 Ensure "Browse" button works as expected	2d	10/09/18	10/10/18	48	Project Team	0%
1.4.7.2 Ensure "Drag and Drop" button works as expected	2d	10/11/18	10/12/18	50	Project Team	0%
1.4.7.3 Ensure "Preprocess Image" button works as expected	2d	10/15/18	10/16/18	51	Project Team	0%
1.4.7.4 Ensure "Train CNN" button works as expected	2d	10/17/18	10/18/18	52	Project Team	0%
1.4.7.5 Ensure "Classify" button works as expected	2d	10/19/18	10/22/18	53	Project Team	0%
1.4.7.6 Ensure graphs display training results properly	2d	10/23/18	10/24/18	54	Project Team	0%
1.4.7.7 Ensure label displays proper class result	2d	10/25/18	10/26/18	55	Project Team	0%
1.4.7.8 Ensure images fit into blank space for display properly	2d	10/29/18	10/30/18	56	Project Team	0%
1.4.8 Procure Hardware/Software	4d	10/31/18	11/05/18		Project Team	0%
1.4.8.1 Computer (Provided by University)	2d	10/31/18	11/01/18	57	Project Team	0%
1.4.8.2 Licensed software	2d	11/02/18	11/05/18	59	Project Team	0%
1.5 Control	10d	11/06/18	11/19/18		Project Team	0%
1.5.1 Project Management	2d	11/06/18	11/07/18	60	Project Team	0%
1.5.2 Project Status Meetings	2d	11/08/18	11/09/18	62	Project Team	0%

1.5.3 Risk Management	2d	11/12/18	11/13/18	63	Project Team	0%
1.5.4 Procurement Management	2d	11/14/18	11/15/18	64	Project Team	0%
1.5.5 Update Project Management Plan	2d	11/16/18	11/19/18	65	Project Team	0%
1.6 Closing	12d	11/20/18	12/05/18		Project Team	0%
1.6.1 Update files/records	2d	11/20/18	11/21/18	66	Project Team	0%
1.6.2 Prepare Final Project Report	2d	11/22/18	11/23/18	68	Project Team	0%
1.6.3 Prepare Final Project Presentation	2d	11/26/18	11/27/18	69	Project Team	0%
1.6.4 Lessons Learnt	2d	11/28/18	11/29/18	70	Project Team	0%
1.6.5 Project Presentation	2d	11/30/18	12/03/18	71	Project Team	0%
1.6.6 Final Report Approval	2d	12/04/18	12/05/18	72	Project Team	0%

Figure 5 (Gantt Chart)




1.5 Control
1.5.1 Project Management
1.5.2 Project Status Meetings
1.5.3 Risk Management
1.5.4 Procurement Management
1.5.5 Update Project Management Plan
1.6 Closing
1.6.1 Update files/records
1.6.2 Prepare Final Project Report
1.6.3 Prepare Final Project Presentation
1.6.4 Lessons Learnt
1.6.5 Project Presentation
1.6.6 Final Report Approval