Media Engineering and Technology Faculty
German University in Cairo

Thesis Title

Bachelor Thesis

Author:	Student Name
Supervisors:	Assoc. Prof. Mohammed Abdel-Megeed Salem
Submission Date:	15 May, 2017

Media Engineering and Technology Faculty
German University in Cairo

Thesis Title

Bachelor Thesis

Author:	Student Name
Supervisors:	Assoc. Prof. Mohammed Abdel-Megeed Salem
Submission Date:	15 May, 2017

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowledgment has been made in the text to all other material used

Student Name
15 May, 2017

Acknowledgments

Write your acknowledgment here....

Abstract

Here you should compose a summary of your work and results in one page.

Contents

List of Figures

Chapter 1

Introduction

Medical imaging is an undeniable tool that is used in diagnosing pathologies, planning surgeries and following up on surgeries. The automatic detection and localization of different bodies in medical images is a heavily tackled field currently due to its fast progression and fine results. 3D images that are worked on by these automatic mechanisms are usually acquired using computed tomography (CT) or magnetic resonance imaging(MRI) as in [14], and both at the same time, as in [5]. It can be noted however that there is more research implementing the Classification forest than the Regression forest.

Chapter 2

Background

2.1 Data Anonymization

Training a machine learning classifier requires a huge amount of data however, releasing such amount of person-specific data may pose a huge security risk to indvidual's privacy as it might open them up to Identity fraud,blackmail among other things. Removing excplicit identifying information such as Names and Email is a solution to prevent information leaking however it must be done in consideration of both the user's privacy and the quality of the data to still be able to use it to in training meaningful prediction models.

2.2 Machine Learning

Currently one of the trending fields in Artifical Intelligence (AI). Machine learning is generally considered as a subset of AI since it has the ability to learn and act without being excplicitly programmed. It works by generalizing certian patterns instead of storing the input data and trying to match it exactly[18]. For example, building an application to recognize cats and dogs using Machine Learning can be done by showing the application pictures of different cats and dogs and having the application remember "features" of the cats and dogs to be able to classify them later on. On the other hand using a traditional approach storing all available pictures of cats and dogs and then later matching pictures exactly with the stored pictures.

2.3 Neural Networks

Neural Networks were introduced in 1943 by Warren McCulloch and Walter Pitts when they presented the first model of the neuron, this discovery ignited the way for research in both biological neural networks and artifical neural networks (ANNs). In the last 10 years however ANNs have been gaining popularity among researchers.

ANNs consist of artifical neurons which is modeled after the simplest part of the central nervous system, each neuorn receives input from other neurons which makes them change their state depneding on the input value and finally, produce an output depending on that input and state.[22] If we look at the network as a whole we will find that it consists of multiple connections, each connection moving the output from neuron a to the next layer in the network to become the input of neuron b and b is passed onto the next layer as input to c and so on, each of these connections is assigned a weight w that is assigned randomly in the begining then it is iteratively changed, We can also add a bias term if want to influence the outputs to lean onto a certain output that is desirable for us.

2.3.1 Fully Connected Neural Networks

Fully Connected Neural Networks are the most basic neural network, they consist of 3 layers fully interconnected to each other, the three layers are, an inpurt layer, any number of hidden layers and an output layer. Every neuron starting from the first hidden layers is connected to atleast the following layers and maybe connected to any layer above it. The input is then propagated starting from the input layer until it reaches the output layer, each neuron in every layer has it's own weight,bias values depending on the importance assigned to it by the network. The activation function of each neuron then decides whether each neuron will be "spiked" or not. The most commonly used activation functions are Sigmoid function, RELU function, Leaky RELU function, Tanh function

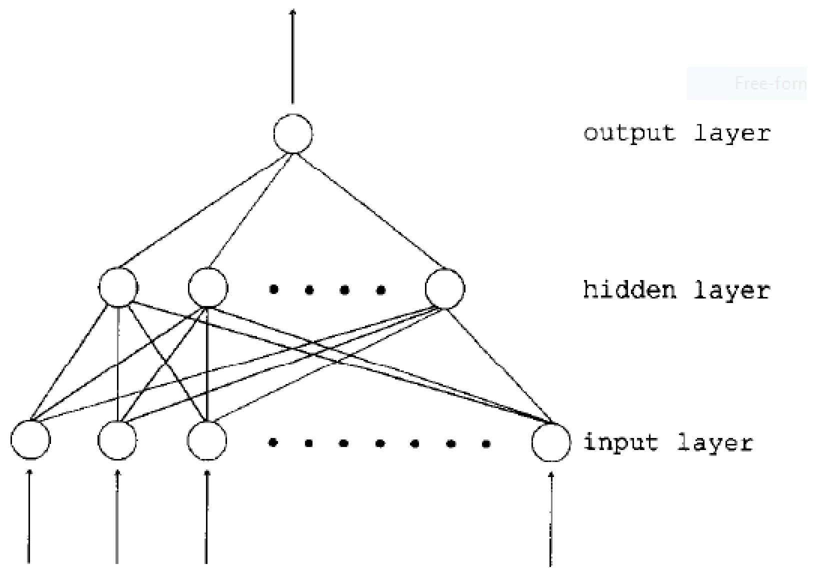


Figure 2.1: In the following figure is an example of a fully connected network from [20]

2.4 Homomorphic Encryption

After investigating what the latest advances in academic research regarding encryption had reached we came to the conclusion that Homomprhic encryption was the most suitable encryption scheme to use, To give a brief introduction, Homomorphic encryption is a form of encryption that allows computation on ciphertexts, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext. This makes homomorphic encryption the best candidate in Machine Learning as a service and cloud computing applications.
In this study we have compiled the three best implementations of Homomorphic Encryption to the best of our knowledge

2.4.1 PALISADE

Palisade is an open source highly protable cryptography library that supports Homomorphic Encryption. The project was previously funded by the NSA and is developed and maintained by the cryptography team at New Jersey Institute of Technology. At the time of writing this PALISADE doesn't support floating point operations as it doesn't support CKKS Homomorphic Encryption scheme

2.4.2 HELib

Homomorphic Encryption Library is the second open source library that we looked at, HELib is developed and maintained by Shai Halevi and Victor Shoup. HELib supports both BGV and CKKS encryption schemes which allows for operations on floating point numbers. The downside to using HELib is that it less maintained due to the low number of contributors.

2.4.3 Microsoft SEAL

SEAL is a library that is built by Microsoft's Cryptography team at Microsoft Research[19]. The advantage of SEAL is that it supports both BGV/BFV and CKKS as well being the most documented library of the three. It also has no dependencies making it modular in addition to having a CMAKE file which is used to build for various environments making it easy to compile and run.

It was also demonstrated that SEAL can be used in privacy preserving Machine Learning where the user data was encrypted on-device and the cloud server operated on the encrypted data providing full user annonymity.

2.5 Cross Platform Mobile Development

The mobile application sector is divided amongst several platforms however the two most dominating platforms are iOS and Android and as a result of this companies need to hire application developers experinced in both platforms or resort to hiring more development teams each assigned to a specific platform which also makes development time longer and causes inconsistency between the different platform applications which might cause user dissatisfaction. Consequently as a solution, Cross Platform development aims to develop 1 application for multiple platforms having almost the same visual and functional features as native platform applications.[2]

2.5.1 React Native

In 2015 Facebook announced it's new framework React Native which aims to develop native applications for iOS, Android, UWP using the same codebase written in Javascript which reduces the number of developers needed to develop an application and the effort needed for maintainence and code reviews. React Native also offers the ability to run platform-specific code in Swift, Java, Objective-C.[15] As of 2019, Facebook, Instagram, Uber, Skype and Pintrest are using React Native primarily.

Chapter 3

Literature Review

Searching for related works in published research, we found out that our preferred approach of using homomorphic encryption was only used and published in 1 implementation[12], it was done by Microsoft's Cryptography team where they wanted to use a cloud service to predict whether a patient would be re-admitted into a hosptial without violating his privacy by learning all his health related data. Their encryption architecture was as follows, the hosptial encrypts the data using the public key and then the encrypted data is sent to the cloud service provider where the prediction is generated and sent back to the hospital which decrypts it using the private key. The cloud service provider however doesn't have access to the private key making it unable to learn the data it was preforming the prediction on nor the prediction itself. However for this paper they assumed that they already had a trained model and they skip directly to the prediction stage which will not be our case since we need to train our model from scratch. Unfortunately the implementation itself wasn't published anywhere.

We managed to find and shortlist 3 compatible homomorphic encryption libraries and a brief description is given for each library along with it's advantages in the Background chapter 2

As for the Data Anonymization appraoch we only found two papers [7, 11] similar to the approach that we had in mind as we wanted to combine multiple annonymity approaches however, we found papers which used different techniques to achieve data anonymity [3, 21] however we will only be going through related papers.

In [7], they state different anonymity techniques and their use cases for each data type for example, if we have a dataset populated by employee's applying anonymization would probably combine using generalization which changes each entry in a column to an entry having a range instead of being exact, suppression is probably going to be used to sanitize the data from high risk data and finally perturbation we can add noise to the data by a certian value to hide the real value while keeping the relations between the values.

In [11], Smilliary to [7] Different anonymity techniques are laid out however in this paper they go a step further and try to generate an algorithm to preform data anonymization on any dataset regardless of it's type by laying out some rules that are traversed in order regarding the anonymization of columns in a dataset. For example, If a column contains categorical data that has a hierarchical tree then this column should be generalized by replacing each value by a value having a range, If a column contains categorical data but no hierarchical tree then it should be strictly removed.

Chapter 4

Methodology

Our goal was to explore whether privacy preserving ML would yield results similar to ML techniques without any privacy constraints and whether data obfuscation techniques can be enough to protect user data or can obfuscated data be de-anonymized. We had a couple of approaches in mind for the privacy preserving ML, the first was using known data anonymization techniques such as on device face bluring/automated metadata deleteion and finally the last approach which was more complicated was trying to use homomorphic encryption and passing encrypted data the the Machine Learning Model. However to be able to test these techniques we had to first collect enough data to be able to analyze our results and tweak our machine learning models so we decided to run an experiment on students in the GUC which involved developing a system that would gather data from user-submitted images and provide brand recognition and personal recommendation based on the each user's preferences. We also disclosed that all data gathered will be anonymized and will be deleted at the end of the bachelor thesis. As for the data obfuscation and data de-anonymization we had to generate our own dataset and classifier as this area hasn't been explored in recent research.

4.1 Privacy Preserving Machine Learning

4.1.1 Privacy Architecture

Here we detail the privacy architecture for both approaches using data anonymization and for using Homomorphic Encryption.

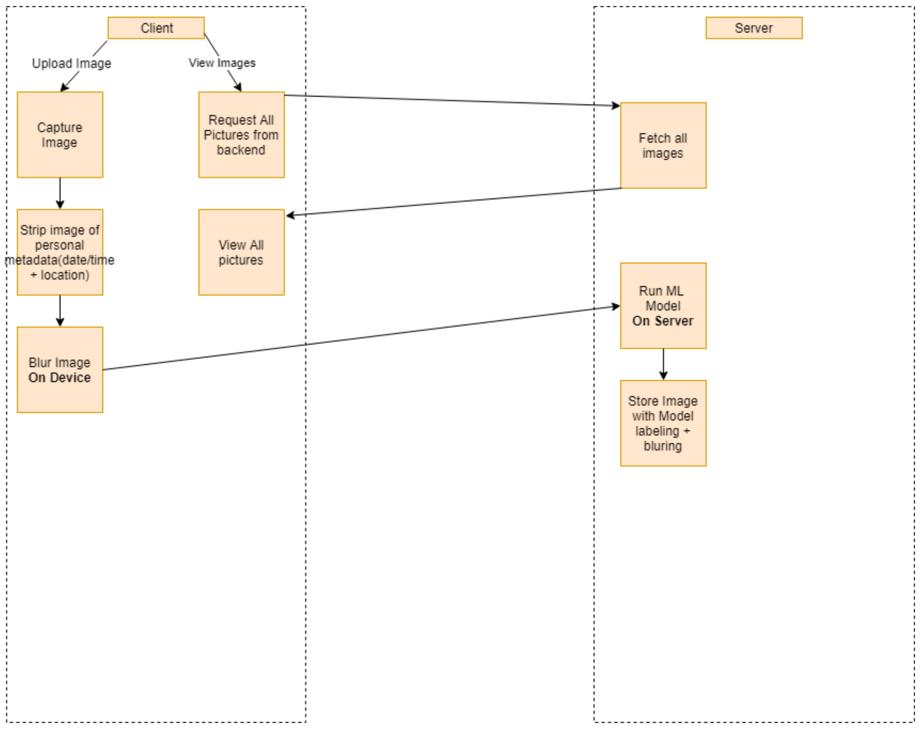

Client
Upload Image
View Images
Capture Image
Request All Pictures from backend
Strip image of personal metadata(date/time + location)
View All pictures
Blur Image **On Device**
Server
Fetch all images
Run ML Model **On Server**
Store Image with Model labeling + bluring

Figure 4.1: Anonymization Architecture

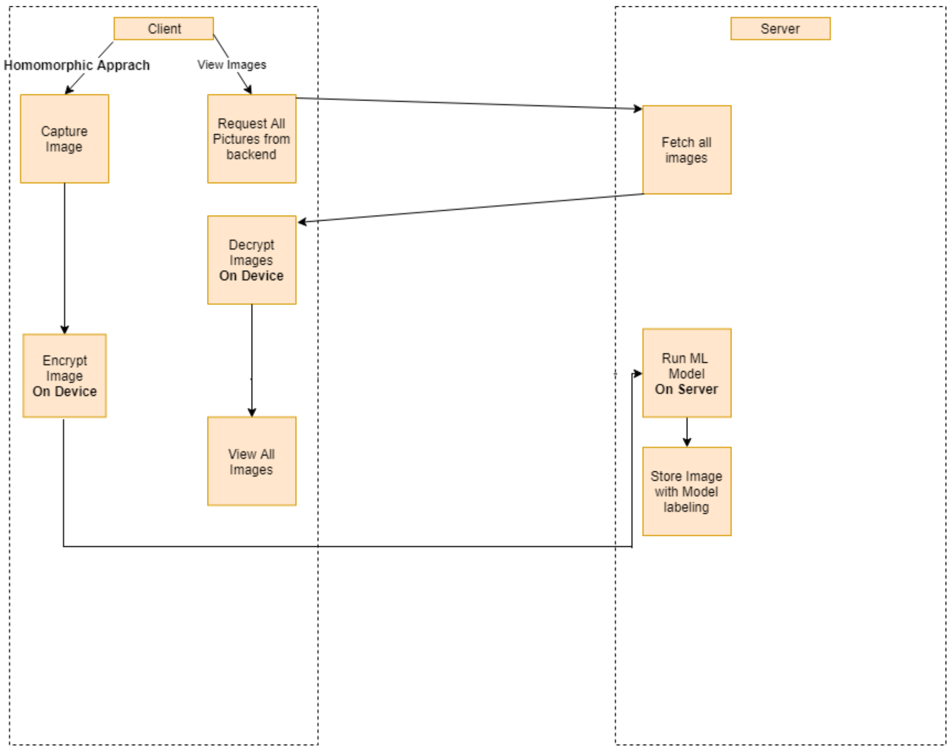

Client
Homomorphic Apprach
View Images
Capture Image
Request All Pictures from backend
Decrypt Images **On Device**
Encrypt Image **On Device**
View All Images
Server
Fetch all images
Run ML Model **On Server**
Store Image with Model labeling

Figure 4.2: Homomorphic Architecture

4.1.2 System Design

The system consisted of a cross-platform Instagram-like mobile application that was developed using Facebook's React-Native, As for the backend we opted for Google's Firebase as the backend provider as it provided multiple advantages over other competitors namely, seamless authentication, realtime database that was shared across all platforms, access to built-in on device and cloud basic Machine Learning models such as facial recognition and object recognition, the ability to deploy custom Machine Learning (ML) models within the same server and lastly, the ability to scale easily. The application was developed alongside 2 of my colleagues who were working on similar bachelor projects (personalization and brand recognition).

A social application was chosen to get the most user interaction which in turn would mean more user data to pass of the ML model. The application flow was as follows, users had to first sign-up to be able to use the application then, whenever a user signed up the application would prompt the user to verify his email address however this feature was later removed as the majority of the users weren't willing to verify their email-addresses and other wanted to use fake emails as a form of identity security, the user would also get a prompt to choose his Social Interaction Level which changes how his data is handled, the user would then get a daily notification to post one or more images/text posts while the experiment was running until the thesis submission date. Users also had the ability to "like" any image posted by any other user to further their preferences.

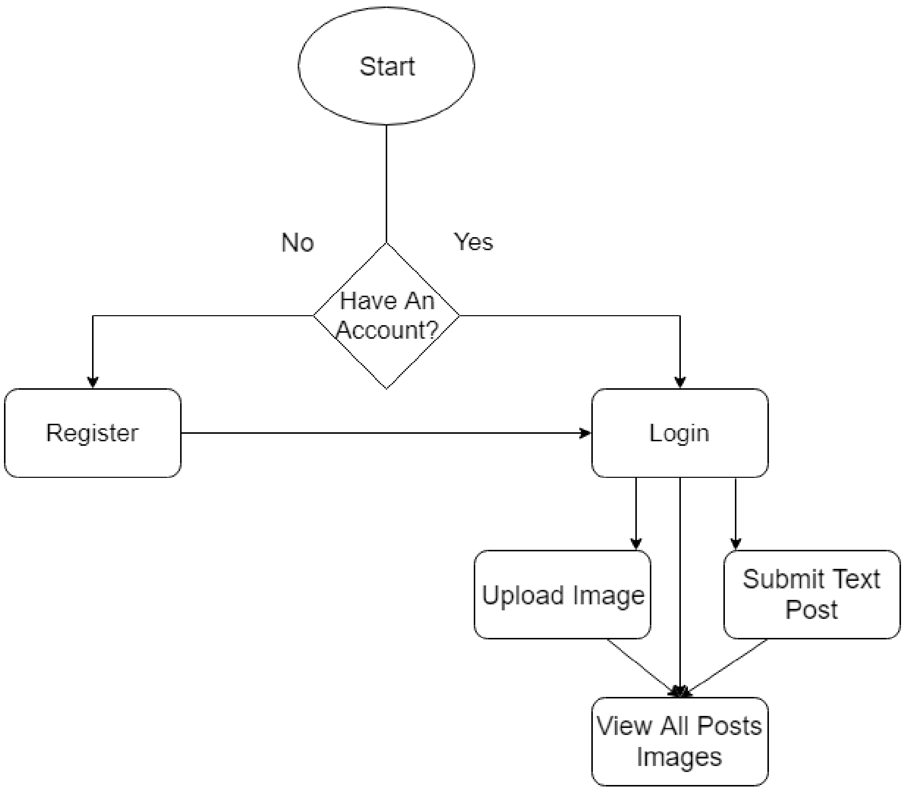


Figure 4.3: In the following figure is the flow of our system

We decided to use Google's Material Interface guidelines not only because it simplifies the user interface as much possible but also due the simplicity in implementation it offers. We chose bright colored buttons and cards in addition to being consistent with our design choices such as button placements,validation errors to make the experience more enjoyable and intuitive so we could increase user retention as much as we could.

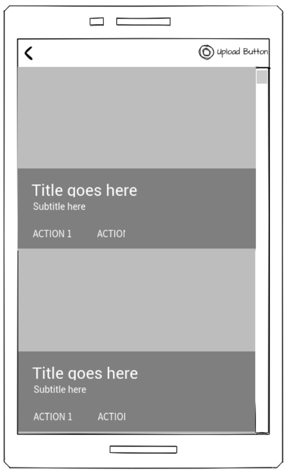


Figure 4.4: In the following figure is a wireframe of the user's timeline

We tried to follow an incremental approach while developing the application where we ranked all the features in order of priority from highest to lowest, we prioritized shipping the most important features first such as the Login/SignUp process and the ability for users to upload images to our database even if it wasn't shown to them in the UI so our data-collection would start as early as possible, followed by features of lessar importance such as improving the user experince and working on multiple security features for the backend and patching bugs. This was acceptable to the majority of the users since the audience that was targeted understood that the application was an experimental application rather than a production-ready application.

At the end of development, the application consisted of 5 different screens being, Login, Register, A Questionnaire, Top Photos and Timeline. The login and registration screens are self-explanatory, as for the questionnaire it was shown upon registration only and was used to put to gather intial preferences about tech items and clothing items the users prefered the most so my colleague would be able to start the personalization process, similary the top photos screen was also shown once following the registration of any user and the completion of the previously described questionnaire, the top photos screen asked the users to input their most favourite pictures that they would like to share whether it was images of themselves or clothing/tech items they liked. We also offered users an option to opt-out of reminding notifications while they were registering as some users reported their refusal to allow our application to get the notification permissions on android, however this was in a later build that wasn't ready so most users couldn't benefit from it.

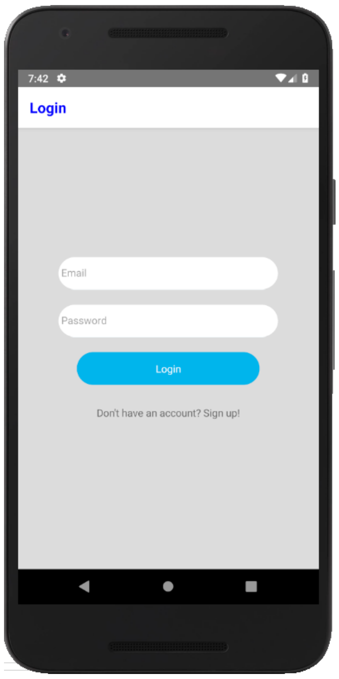


Figure 4.5: Login Screen

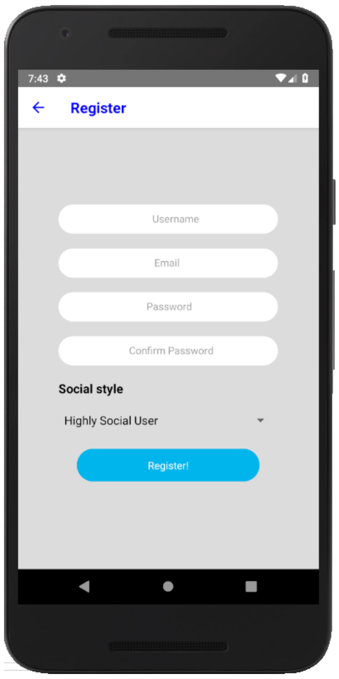


Figure 4.6: Registeration Screen

4.1.3 Homomorphic Scheme

We had previously outlined three potential libraries to use for Homomorphic Encryption(HELib,SEAL,PALISADE). We decided to use SEAL as it proved the most beneficial to us. Since the goal was to use Homomorphic Encryption on a mobile platform which was not natively supported by any library as of the time of writing this, We had to re-compile the SEAL library which was written in C++ and compile it using Javascript to be compatible with our React Native application. There was no direct conversion available to either convert the C++ code into Javascript or compile C++ directly into Javascript, however what we found was that React Native offers what is called Native Modules, Native Modules are basiaclly either native android code (Java) or native iOS code (Swift) that are linked to Javascript using what is called React Native Bridge which exposes native methods to Javascript. So our goal now to first compile the C++ SEAL library into Java first then link the produced Java library to Javascript using the React Native Bridge.

In general, to run C/C++ code using Java it is done through the Java Native Interface (JNI) which acts as a bridge between Java and C/C++ source file. This process was done as follows, First, we had to use the provided CMake file which is used to manage the building process in a compiler independent manner to import the source files into Android Studio and package them into a library, for this process a lot of errors were encountered due to compatibility issues with different versions of CMAKE. After managing to port the library into Android Studio and connecting it to our React Native application, the next step was to feed the encrypted data to our neural network however this was not possible as all pre-built Machine Learning libraries and APIs don't support this kind of data format as all of the libraries are designed to take the data directly without any change. We then figured out that it was necessary to build our neural network from scratch starting from the filters and kernels to the activation functions meaning we would essentially have to build our own library which proved to be too challenging and time consuming for this bachelor thesis since we had no prior knowledge of Machine Learning and Neural Networks.

4.2 Data Anonymization and Incremental Privacy

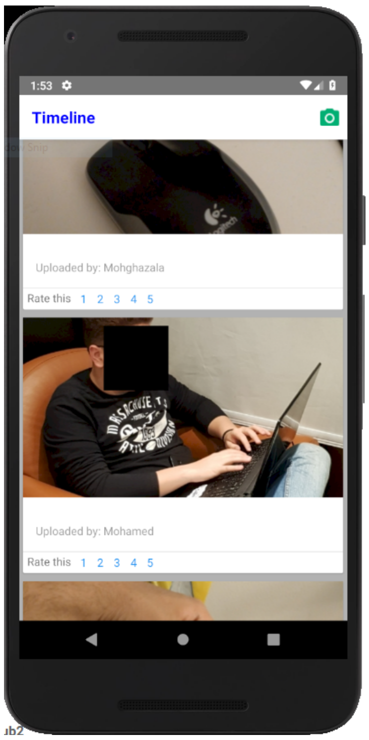
For this part we discuss how multiple Data Obfuscation techniques were used for user submitted data in the application we built instead of using Homomorphic Encryption. Secondly, what our approach for Incremental Privacy was, how we implemented it and applied it to the 2 closely related synthetic datasets that were generated, One was designed for friendship inference between students in the GUC and the other one was designed for location inferencing between students in similar tutorial groups in the GUC, which we will be discussing in later sections in full detail, to test how Incremental Privacy would affect our Machine Learning Models.

4.2.1 Data Anonymization

The main idea in this scheme was to allow the user to choose his own level of privacy by having the user choose his desired social interaction and privacy during the registration process which would affect his experience in the application later on. We defined 3 privacy/social levels for the users to choose from. The levels being, Highest Social Interaction, Medium Social Interaction user and lastly a Non-Social User. Each level then would have certain data anonymization techniques applied to his data. High Social Users had all their data as it as without any modifications to their images and text submitted, On the other hand Medium Interaction users had all their image run on a trained face detection machine learning model and applied blurring over resulting faces. These user's text posts however were posted as it as without any filtering. Lastly, the Non-Social users had the most privacy out of the three levels, For images they had the blurring and face detection algorithm run in addition to removing in metadata in images submitted such as location and date.

The model used for facial detection is an *on-device* model to ensure privacy and is provided by Google's firebase ML Kit, as for the blurring, All javascript image manipulation libraries weren't compatible with React Native hence, we were forced to build our own blurring algorithm using native Java which was pretty straight-forward and consisted of running Google's Firebase ML Kit to detect the faces in the input image then we applied the blurring algorithm on the bounding boxes returned from the face detection model finally, we passed the resulting image back to React Native using React Native Bridge.

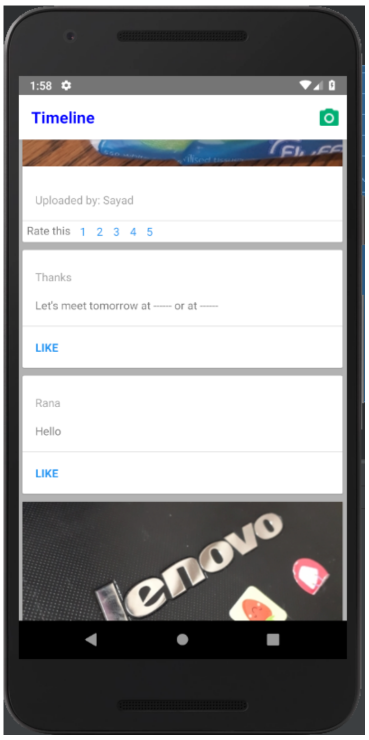
As for The text posts of these users was also filtered to prevent any locations,names and phone numbers from appearing using a Javascript Natural Language Processing library "Compromise"[6]. We have also added local GUC landmarks and buildings to the lexicon of the library which detects these words when they are used in any form.



Uploaded by: Mohghazala

Uploaded by: Mohamed

Figure 4.7: Test User with blurred face



Uploaded by: Sayad

Thanks

Let's meet tomorrow at —— or at ——

Rana

Hello

Figure 4.8: Censored text with location Data

4.2.2 Dataset Generation

Our initial goal was to test the Data Obfuscation and Incremental Privacy locally on students from the GUC however since there was no publicly available dataset from the university, we had to resort to generating our own synthetic Dataset. Synthetic Dataset generation is commonly used for research purposes as it can be customised to fit the needs of any particular research topic moreover, Institutions and Organizations have been hesitant to publish any results using any datasets that have been collected from their customers due the recent growing privacy concerns worldwide[9] espically in the EU after the latest General Data Proctetion Regulation (GDPR) laws have been passed. The generated data varies according to the application and research field and may often include text,social connections,images,graphs. The simplest way to generate such datasets is to create multiple scripts each handling a specific part of the problem the research is tackling and then combine all related rows of data.[1]

In this paper we chose to use Python to generate our datasets as it has multiple data manipulation libraries namely Numpy and Pandas which are both part of the SciPy package for Python[16].

For the student friendship prediction dataset we tried to make it as realistic as possible to avoid making it too easy to learn. We tried to model the data that could be generated from 1st year students in a 1 month duration, The design was as follows, each student had a name, 5 intrests chosen randomly out of 20 pre-determined intrests,a tutorial group that the student is assigned to and finally, each student had a number of “events” that represented his online social presence which could resemble tags in pictures,Facebook mentions,Facebook posts. To make it more realistic each event had a random number of students participating in it, these students were selected in 2 phases, the first phase was used in the 15 days of the modeling process where students were selected to be in other student’s events randomly which resemble the process that students usually go through when they first join an institution where they are introduced to a large amount of people as they are seeking to make new friends however as time passes each student’s social circle is further reduced to people who are considered friends rather than random people met in lectures, we tried to achieve this in the final 15 days of the 1 month modeling period by making the participants of the remaining events based on the mutual intrests of students and the number of their previous appearances in the first 15 days of the modeling process. After generating the data for 150 students we then used cross product to get a record for every student with all the others and finally, to decide whether every 2 students should be friends or not, we designed an equation with what we felt was the most realistic weights for each parameter described above and the final equation was as follows:

$$F = 0.15(CommonTutorial)+0.3(MutualIntrests/3)+0.50(MutualEvents/15)^2+0.5(CommonFavLoc) \tag{4.1}$$

55% of the weight was given if the number of shared events between the 2 students during the 1 month period was 15 or more events, 30% of weight was given if the students had atleast 3 mutual intrests and the last 15% were given if the students shared the same tutorial. We also considered the values $F >= 0.56$ to avoid having un-realistic relations. We ended up with 22,052 rows and we provide a sample table with values taken from the dataset below 4.1 We also added the result of the function to each row of the dataset for verification purposes but it is not shown in the table below due to space limitations.

Common_Tut	Common_Loc	Mutual_Events	Mutual_Intrests	Friendship	Id1	Id2
1	0	3	2	0	0	1
1	0	2	0	0	0	2
0	1	8	2	1	0	3
. .	. .	. .	. .	. .	. .	. .
1	0	3	2	0	148	147

Table 4.1: Friendship Dataset

As for the second dataset the goal was to explore whether we could provide location prediction for students inside the GUC and on holidays given their list of friends. Generating the second dataset was much easier since we had the data that resulted from the first dataset such as each student's friend list in addition to each student's tutorial number which was also present in the first dataset and how would the accuracy be impacted by the incremental obfuscation of their data as a form of data privacy protection. As for the location data we proceeded similar to the first dataset by generating a random classroom from the GUC that was assigned to each tutorial for 30 rooms. We managed to generate 4606 rows and a sample of the dataset is provided below

TimeSlot	Tutorial	Location	Student	fav_loc	friends	Id
Sat 1st	1	C7.301	Student0	9	371114	0
Sat 2nd	1	D4.304	Student0	9	371114	0
Sat 3rd	1	C5.304	Student0	9	371114	0
. .	. .	. .	. .	. .	. .	. .
Holiday	6	Paris	Student142	3	134135	142

Table 4.2: Location Prediction Dataset

Finally, we extracted 10 rows from the second dataset that weren't used in neither training nor testing to be used for the purpose of showing a live demo during the defense by hosting our models on a simple website and running the prediction on these extraced rows.

Chapter 5

Results

5.1 Experiment setting and Model Architecture

5.1.1 Application Experiment Setting

To test the performance of our data anonymization and our application generally, we needed to get the largest amount of real world users we can get. We managed to reach an agreement with the team teaching CSEN 603 which is the Software Engineering course in the GUC led by Dr.Aysha Alsafty to publish the application for the students enrolled in this course. In this course students are divided into groups of 10 and try to develop the best web-application for real clients. We encouraged teams to post pictures and text posts while they were working on their projects and to further incentivize students to participate we also stated that there would be an award for the most active team in the awards ceremony held at the end of semester. We also stated that all data collected will be deleted at the end of project and will be used only for the purposes of this research strictly to address their privacy concerns.

5.1.2 ML Model Architecture

For the models we decided to use Keras[4] which is a model-level library, providing high-level building blocks for developing deep learning models. Keras offers simple APIs for the users which makes it easy to learn and use espically for simple models. It also provides 3 backend enginges for to choose from TensorFlow/Theano/CNTK for the purposes of this research we decided to choose TensorFlow as it is widely adopted and has the highest support availability incase we encountered any problems. The models we built are fairly simple relative to other deep neural network models as the the relations weren't complex in both of the datasets.

The classifier for the first dataset consisted of 2 hidden dense layers each having 4 nodes, both of the hidden layers used "RELU" as their activation function and used

L2-Regularization to improve the loss function, the output layer was placed after the 2 hidden layers and had only 1 node with a sigmoid activation function. We configured to model to use “Binray Crossentropy” as the loss function as we are trying to classify whether 2 students are friends or not.

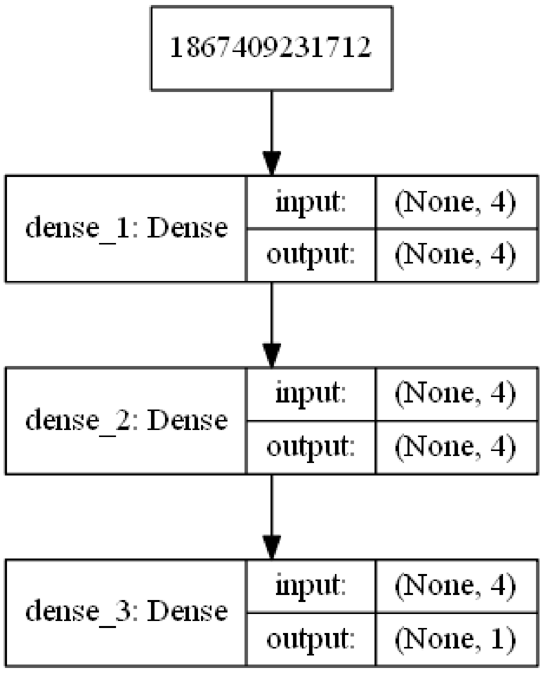


Figure 5.1: Model 1 Diagram

As for the second Model this one was more complex as we first had to use one hot encoding (OHE) to encode all room locations,date/timeSlot fields and student’s names as well which were all generated in text form which can’t be fed to the Model directly. What OHE does simply is that it counts the number of unique values in a column and then creates new columns equal to the numbers of these unique values in the column where every new column corresponds to 1 of the unique values, The new columns are filled by checking which unique value was present for this row and put a “1” in it’s corresponding newly generated column and “0”s in all other generated columns while the rest of the data for the row is unchanged. For this model we had to do multiple tweaks to the architecture of the model as we progressed with the data obfuscation however these details will be discussed thoroughly in the next chapter and we will only state the baseline architecture of the model in this section, The model consisted of 3 layers, The first layer is the input layer which had a dimensionality of 184 due to using OHE on the rows while the second layer is the only hidden layer which had 20 nodes and used a “RELU” activation function and also used L2-Regularization to minimize the loss function and prevent overfitting, finally the last layer which is the output layer consisted of 38 nodes which represented 38 unique room locations and used “Softmax” as it’s activation function.

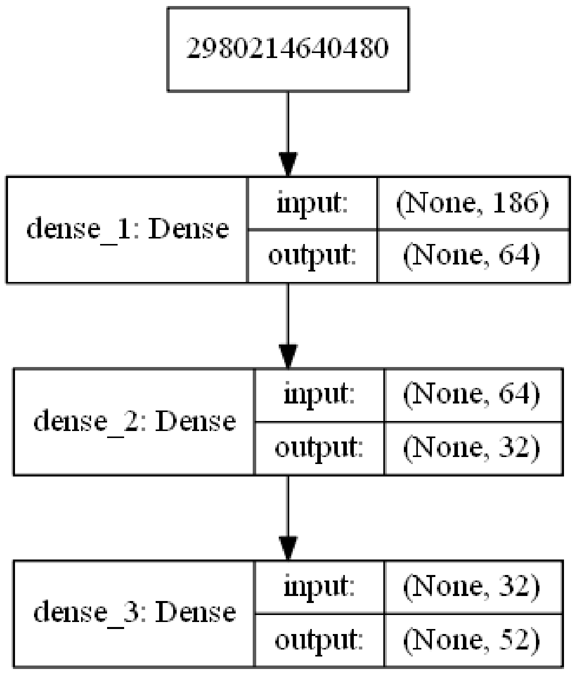


Figure 5.2: Model 2 Baseline Diagram

The 2 models used “Adam” as their optimizer as it can handle gradients with big differences which our dataset maybe suffering from, in addition to being able to handle noisy data and very easy to configure and use

5.2 Application Experiment Results

Evaluating the effectiveness of an experiment involving a mobile application generally involves measuring the time performance of different aspects of the backend of the application, performing analysis on the application’s users and finally analyze the collected data through the application.

5.2.1 Application Users

We managed to get 46 registerd user on our backend who consumed about 1.6GB of bandwidth of our 10GB Quota for downloads/uploads during a 2 month duration and 50MB out of a 1GB Quota of consistent storage on our fireback database, however about 5 of those were accounts made for testing the application throughout the semester thus the number of real world users is about 40. In this section we will be performing statistical analysis on various aspects of the user’s profiles such as users gender distribution which was used to get more accurate reccomendations for users, users choices of privacy and the results we got from the questionnaire that was provided inside the application after registration.

Regarding the gender distribution we offered 3 options being (Male,Female,Private) for each registering user, the statistics at the time of writing this excluding the test users was as follows, we had 20 Male users, 14 Female users, and 6 users who refused to specify their gender. We note that the low number of users who refused to share their gender shows that either most users don't see a risk by sharing their gender or they trust our application to share it. It is also noted that the higher number of Male users shows that Male users are more likely try out new applications to help with their testing especially that our testing was done in the Computer Science department where student might have an intrest to tryout new applications.

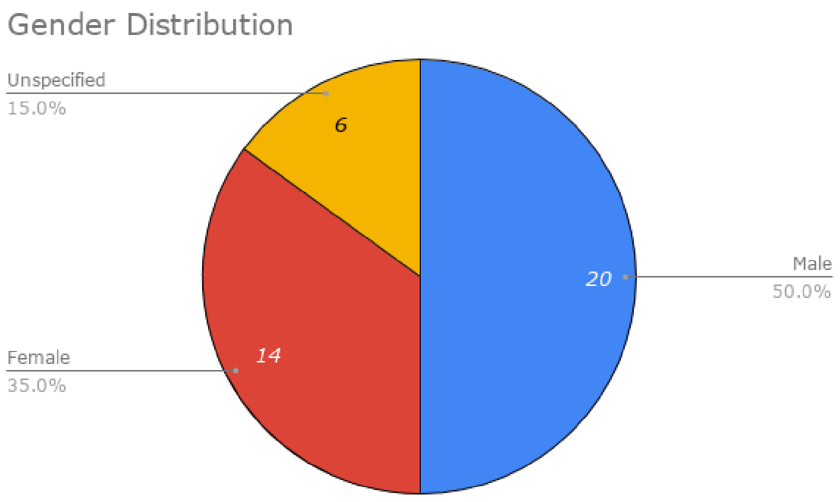


Figure 5.3: User's Gender Distribution

As for the Privacy level chosen by the users, we also offered 3 levels of privacy as mentioned in [4.2]. The privacy distribution was found to be as follows, The highest chosen setting was the medium privacy setting with 17 users choosing it followed by the high privacy setting chosen by 13 users and finally the lowest privacy setting was selected by 10 which was more than excepted. In the next chapter we will discuss these findings and numbers and what we can conclude from these values and what they mean to us and their correlation with other values.

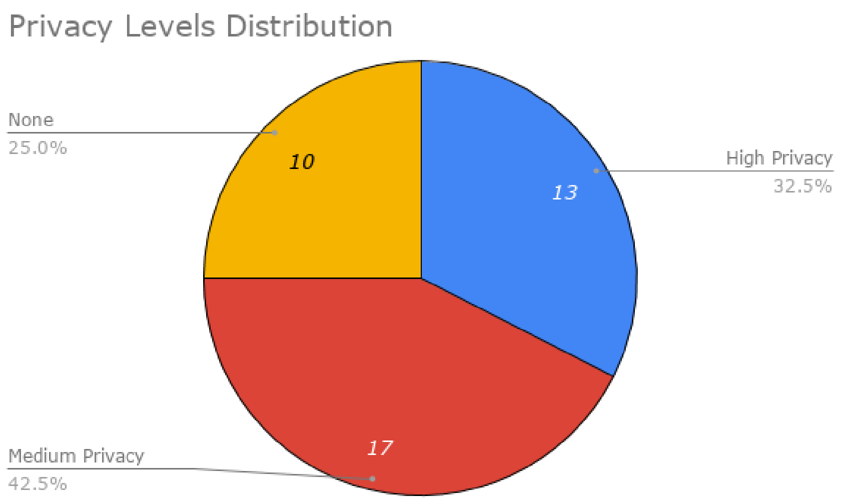


Figure 5.4: User's Privacy Level Distribution

Now moving onto the data collected from the application, our application was used in collecting data of various forms, images, text, and ratings. In total we had 103 data points distributed as follows, 67 Image Posts, 22 Text posts and 14 Image ratings.

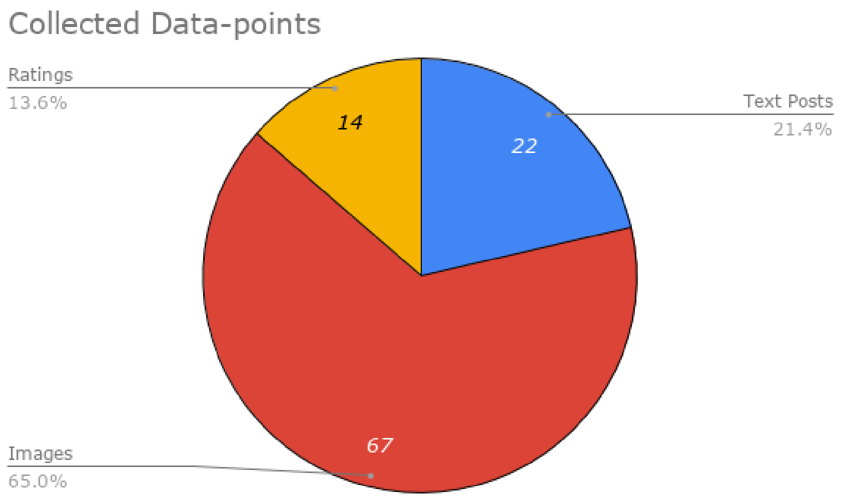


Figure 5.5: Submitted Data Distribution

5.2.2 Application Backend Performance

In this section we will discuss how long it takes for Images/Text to be posted to the backend, what features of faces the face detection algorithm can detect, how different modes of face detection behave and blur and how long it takes to blur these faces, how long it takes to load user's timeline and finally how much time it takes for the registration/login process.

The following timings were recorded using Moment.js which is the most popular date/-time manipulation library in Javascript. Login and registration are pretty fast, however to get an accurate timinig and mitigate any network issues we tested each operation 5 times and got the average. It is also noted that the amount of data transfered when logging-in or registering is minimal so network speed will not be a factor in this bench-marking process. The average time for logging in was 4488.1ms while the average time for registering was 5374.1ms which is 12% higher since logging in is just matching a record with no POST request while the registeration we perform validation checks on the front-end and on the backend. As for the loading the data from the backend database this is excepted to have the largest time and worst performance since we currently don't perform any data chunking and load all submitted Images/Text posts regardless of date submitted or if it is seen before whenever a user views his Timeline. The average time for loading the data was 14,795.4ms. We state the exact value for each trial in the table below.

	1st	**2**	**3**	**4**	**5**	**Average Time(ms)**
Login	**3553**	**2820.1**	8210	4240	3621	**4488.1**
Register	**5879.45**	**6092.38**	5455.367	**4727.55**	**4718.8**	5374.1
LoadTimeline	**15089**	**15463**	**14446**	**14793**	**14186**	**14795.4**

Table 5.1: Average Loading Time for backend operations

Now moving onto the face detection algorithm and blurring algorithm, the face detection model that firebase provided had 2 modes, a fast mode and an accurate mode, here we detail the functional differences between each of them.

For both the "ACCURATE" mode and the "FAST" mode the maximum number of faces we could detect was 6 faces in 1 image however, the "ACCURATE" mode provided more powerful detection features such as being able to detect partially obscured faces behind other objects as well as being to detect faces that have been rotated to the right or to the left relative to the device's camera in contrast to the "FAST" mode where the detected faces can only be rotated upwards or downards, however this trade-off comes at the cost of the detection speed. The full pipeline in the "FAST" mode takes 6579ms to detect up to 6 faces in 1 image and blur them incase a high privacy level is chosen and save it to the database while the "ACCURATE" mode pipeline takes up to 14,747.02 ms to process the same image. We repeat the time measuring process used above in the following table by averaging the time measured for uploading and blurring the same image 5 times.

	1	**2**	**3**	**4**	**5**	**Average Time(ms)**
FAST	**7092.1**	**6395.7**	**6524.4**	**6440.1**	**6444.5**	**6579.36**
ACCURATE	**14782**	**14931.5**	**14798.7**	**14501.9**	**14720.8**	**14747.02**

Table 5.2: Average Time for FAST and ACCURATE modes

The “ACCURATE” mode also offers facial features detection such as smile detection, face countor detection however these features weren’t useful for our purpose so we won’t be going into their details.

5.3 ML Models Results

In this section we will be evaluating the results of the 2 models we outlined earlier using the datasets generated in [4] however we will be applying tweaks to these datasets to simulate applying incremental privacy and consequently we will have some tweaks to the second model architecture to improve it’s performance in response to incremental privacy.

All the models described below were trained locally on a machine with the following hardware specifications CPU: Intel I5-4460 GPU: Nvidia GTX 1060 6GB RAM: 16GB

5.3.1 Friendship Prediction Model

This model is a very simple model to predict as it doesn’t have any complex features or relations moreover, the predicted value is generated by a function that we invented which is pretty easy for a (ANN) to learn and approximate. We split the dataset into an 80-20 training and validation split and started training using the model stated in [5.1] and started the training process with 50 epochs however we noticed that the model was overfitting heavily where we managed to get a training accuracy of 100%, We later discovered that this was caused due to 2 reasons first, high number of epochs compared to the dataset size as we had a high number of steps per epoch which caused our model to see the same data multiple times and secondly, our dataset as it suffered from a data-leak. A dataleak is when the training data contains the prediction that the model should learn, In our case we included the value of the friendship function which determined whether 2 students should be friends or not, to solve this we removed this column from the training and validation data and reduced the number of the epochs to 30 while keep the same steps per epoch.

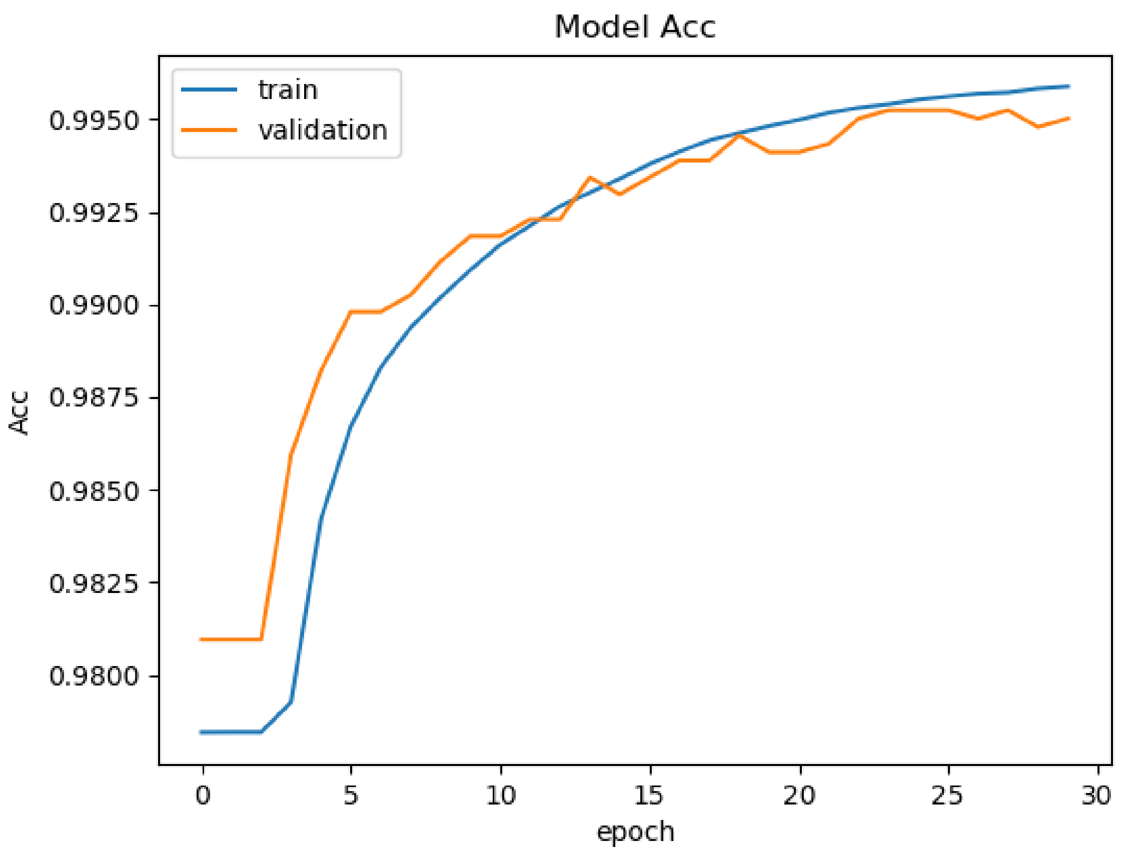
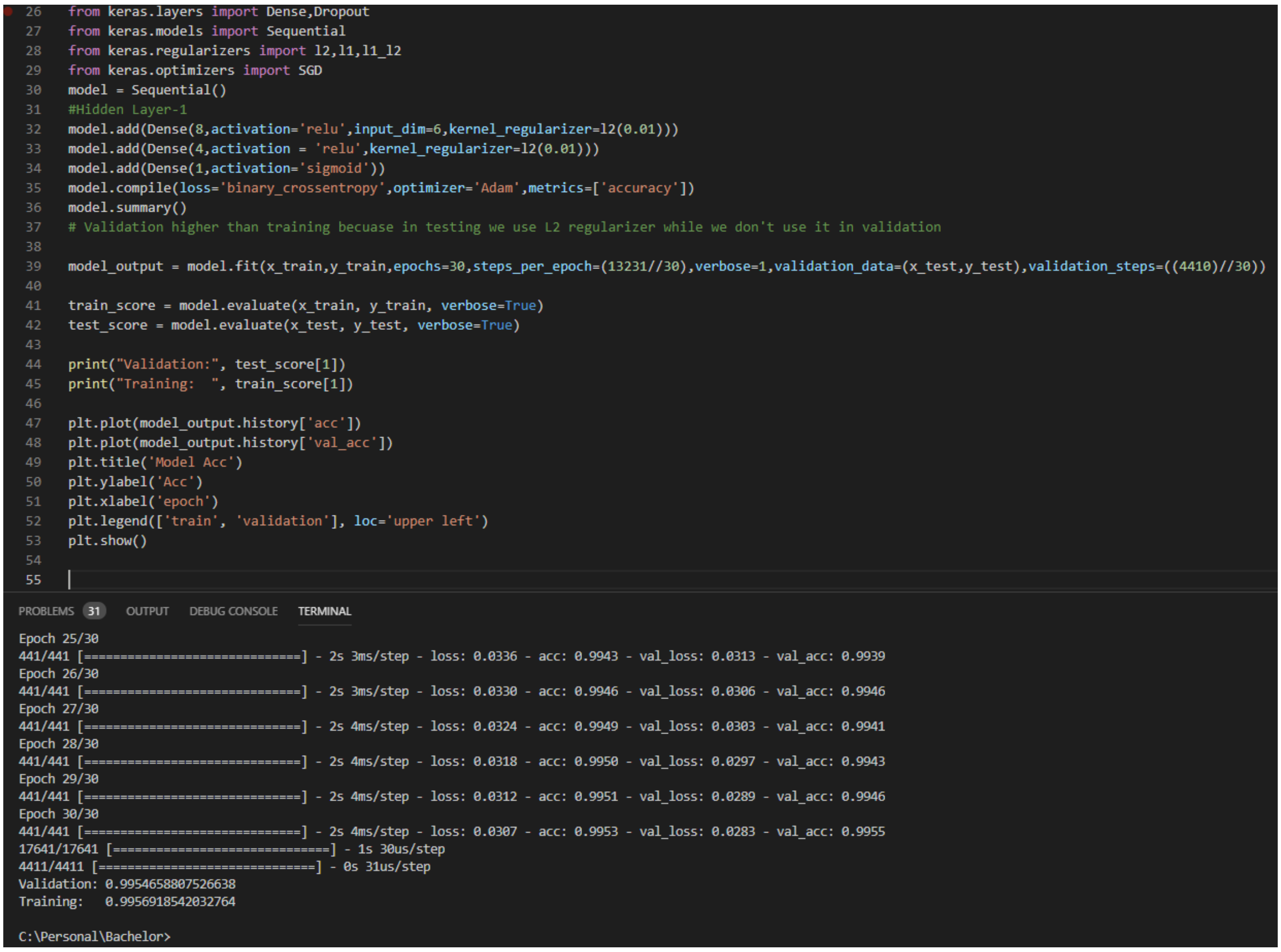


Figure 5.6: Training and validation accuracy

Looking at this curve it is noted that at some points the validation accuracy is higer than the testing accuracy this is caused due to following, Every Keras model has 2 operation modes, Training and Testing, in the training mode Regularization mechanisms such L1/L2 regularization and drop-out are used however, when Keras switches the model to testing mode all regularization mechanisms are turned off moreover, the reported training loss is actually the average of the losses over batches of training data and losses at the first epochs is generally higher than losses at later ones. On the other hand, the testing loss which is computed at the end of an epoch using the model as it is without considering previous batches thus, resulting in a lower loss. The above can also be verified by making the model predict the labels for the training and testing data after it is done trainig we can see it the trainig accuracy is actually higher. It is also noted that the fluctuations in both the training and validation accuracy are very small fluctuations in the range of 0.0020 however they may appear otherwise due to plotting the labels with 4 decimal places.



```
from keras.layers import Dense,Dropout
from keras.models import Sequential
from keras.regularizers import l2,l1,l1_l2
from keras.optimizers import SGD
model = Sequential()
#Hidden Layer-1
model.add(Dense(8,activation='relu',input_dim=6,kernel_regularizer=l2(0.01)))
model.add(Dense(4,activation = 'relu',kernel_regularizer=l2(0.01)))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='Adam',metrics=['accuracy'])
model.summary()
# Validation higher than training becuase in testing we use L2 regularizer while we don't use it in validation

model_output = model.fit(x_train,y_train,epochs=30,steps_per_epoch=(13231//30),verbose=1,validation_data=(x_test,y_test),validation_steps=((4410)//30))

train_score = model.evaluate(x_train, y_train, verbose=True)
test_score = model.evaluate(x_test, y_test, verbose=True)

print("Validation:", test_score[1])
print("Training:  ", train_score[1])

plt.plot(model_output.history['acc'])
plt.plot(model_output.history['val_acc'])
plt.title('Model Acc')
plt.ylabel('Acc')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

```

```
Epoch 25/30
441/441 [==============================] - 2s 3ms/step - loss: 0.0336 - acc: 0.9943 - val_loss: 0.0313 - val_acc: 0.9939
Epoch 26/30
441/441 [==============================] - 2s 3ms/step - loss: 0.0330 - acc: 0.9946 - val_loss: 0.0306 - val_acc: 0.9946
Epoch 27/30
441/441 [==============================] - 2s 4ms/step - loss: 0.0324 - acc: 0.9949 - val_loss: 0.0303 - val_acc: 0.9941
Epoch 28/30
441/441 [==============================] - 2s 4ms/step - loss: 0.0318 - acc: 0.9950 - val_loss: 0.0297 - val_acc: 0.9943
Epoch 29/30
441/441 [==============================] - 2s 4ms/step - loss: 0.0312 - acc: 0.9951 - val_loss: 0.0289 - val_acc: 0.9946
Epoch 30/30
441/441 [==============================] - 2s 4ms/step - loss: 0.0307 - acc: 0.9953 - val_loss: 0.0283 - val_acc: 0.9955
17641/17641 [==============================] - 1s 30us/step
4411/4411 [==============================] - 0s 31us/step
Validation: 0.9954658807526638
Training:   0.9956918542032764

C:\Personal\Bachelor>
```

Figure 5.7: Verification of [5.6]

We decided not to simulate any kind of incremental privacy for this model as this dataset was just created to see if we are on the right track and to help us in creating the second model which is our original goal

5.3.2 GUC Location Prediction Model

Chapter 6

Conclusion and Future Work

We have seen that Random forest can be used to do different tasks. This is definitely something that gives the random forest a lot of flexibility and power. It can be used for regression, classification and other goals. In our topic, we have seen that we can achieve our goal either by regression or classification as the review of the previous work has shown. What we have chosen to work with was the classification way as it was simpler to understand, implement and had more available resources and works of others that helped us a lot.

Appendix

Appendix A

Matlab Code

```
%This function splits the matrix "im" into 2 horizontal halves
%and gets the mean of each half then gets the mean of those
%
%Re
```

Bibliography

[1] Georgia Albuquerque, Thomas Lowe, and Marcus Magnor. Synthetic generation of high-dimensional datasets. *IEEE transactions on visualization and computer graphics*, 17(12):2317–2324, 2011.

[2] Oscar Axelsson and Fredrik Carlström. Evaluation targeting react native in comparison to native mobile development. 2016.

[3] Roberto J Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *21st International conference on data engineering (ICDE'05)*, pages 217–228. IEEE, 2005.

[4] François Chollet et al. Keras. https://keras.io, 2015.

[5] Chengwen Chu, Daniel L Belavỳ, Gabriele Armbrecht, Martin Bansmann, Dieter Felsenberg, and Guoyan Zheng. Fully automatic localization and segmentation of 3d vertebral bodies from ct/mr images via a learning-based method. *PloS one*, 10(11):e0143327, 2015.

[6] Compromise. https://github.com/spencermountain/compromise, March 2019.

[7] Graham Cormode and Divesh Srivastava. Anonymized data: generation, models, usage. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 1015–1018. ACM, 2009.

[8] Antonio Criminisi, Jamie Shotton, and Stefano Bucciarelli. Decision forests with long-range spatial context for organ localization in ct volumes. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 69–80. Citeseer, 2009.

[9] Ashish Dandekar, Remmy AM Zen, and Stéphane Bressan. A comparative study of synthetic dataset generation techniques. In *International Conference on Database and Expert Systems Applications*, pages 387–395. Springer, 2018.

[10] Adrien Depeursinge, Antonio Foncubierta-Rodriguez, Dimitri Van De Ville, and Henning Müller. Three-dimensional solid texture analysis in biomedical imaging: Review and opportunities. *Medical image analysis*, 18(1):176–196, 2014.

[11] Benjamin CM Fung, Ke Wang, and S Yu Philip. Anonymizing classification data for privacy preservation. *IEEE transactions on knowledge and data engineering*, 19(5):711–725, 2007.

[12] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210, 2016.

[13] Ben Glocker, Johannes Feulner, Antonio Criminisi, D Haynor, and Ender Konukoglu. Automatic localization and identification of vertebrae in arbitrary field-of-view ct scans. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2012*, pages 590–598, 2012.

[14] Ben Glocker, Darko Zikic, Ender Konukoglu, David R Haynor, and Antonio Criminisi. Vertebrae localization in pathological spine ct via dense classification from sparse annotations. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 262–270. Springer, 2013.

[15] Niclas Hansson and Tomas Vidhall. Effects on performance and usability for cross-platform application development using react native, 2016.

[16] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed ¡today¿].

[17] Dwarikanath Mahapatra. Analyzing training information from random forests for improved image segmentation. *IEEE Transactions on Image Processing*, 23(4):1504–1512, 2014.

[18] Iqbal Muhammad and Zhu Yan. Supervised machine learning approaches: A survey. *ICTACT Journal on Soft Computing*, 5(3), 2015.

[19] Microsoft SEAL (release 3.2). `https://github.com/Microsoft/SEAL`, February 2019. Microsoft Research, Redmond, WA.

[20] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62, 1997.

[21] Manolis Terrovitis, Nikos Mamoulis, and Panos Kalnis. Privacy-preserving anonymization of set-valued data. *Proceedings of the VLDB Endowment*, 1(1):115–125, 2008.

[22] Andreas Zell. *Simulation neuronaler netze*, volume 1. Addison-Wesley Bonn, 1994.