Obfuscated Machine Learning

Aditya Dixit, Garvit Pugalia, Shaan Mathur, Sparsh Arora, Ritesh Chitalia, Yash Choudhary

May 2019

Contents

1 Introduction

1.1 Purpose

The purpose of this Software Requirement Specification (SRS) document is to present detailed requirements for the Obfuscated Machine Learning application. It will cover the overall description of the product, specific system features and requirements, and hardware/software interfaces. It will also specify dependencies, constraints, functional and non-functional requirements.

1.2 Document Conventions

This document is intended for the following:

- Developers and software engineers working in the team assigned to design this application.
- Potential clients that will be using and integrating the application into their own system.

1.3 Scope

Obfuscated Machine Learning is a Python package built on top of TensorFlow (one of the leading machine learning python packages) that intends to introduce a new technology in machine learning that allows the communication of inputs in a secure and efficient manner. Specifically, it aims to design neural networks that run on obfuscated inputs; in other words, it can efficiently pseudo-encrypt the input, send it securely across a network, and have the neural network run on the pseudo-encrypted input without decrypting. The basic idea behind the technique can be thought as: The neural network using a linear transformation with nontrivial kernel will accept inputs that are obfuscated, where obfuscation is summing the input vector with some kernel vector. This technology can be very useful in settings in which privacy is essential. For instance, a hospital may want to use a neural network to detect whether a particular patient's x-ray demonstrates signs of cancer, but privacy laws prevent the image itself from being sent over the network; here, the Obfuscation Machine Learning application can be used to efficiently transmit private data over the network using pseudo-encrypted neural network inputs.

1.4 Intended Audience

As the language of this document is technical, the intended audience for this document are for the software engineers, software , and management to view. For nontechnical audiences, this document should still be fairly accessible; nonetheless, the Glossary describes many of the technical terms used in this specification.

2 Overall Description

2.1 Product Perspective

The basic math involved behind the working of Obfuscation Machine Learning involves the following key concept:

Let $\mathcal{N} : R^m \to R^n$ be the function computed by a neural network. If the neural network is designed so that its first layer has more neurons than the second layer, then the kernel of the first linear transformation will be nontrivial. If there are k linearly independent vectors $\vec{y}_1, \ldots, \vec{y}_k \in R^m$ in the kernel, then for any input vector $\vec{x} \in R^m$ and any choice of parameters $\lambda_1, \ldots, \lambda_k \in R$,

$$\mathcal{N}(\vec{x}) = \mathcal{N}(\vec{x} + \lambda_1 \vec{y}_1 + \ldots + \lambda_k \vec{y}_k).$$

Therefore we can obfuscate the input vector by adding to it any kernel vector of our choosing.

Using this idea of augmenting inputs with linear combinations of kernel vectors, we aim to create a Python package built on top of TensorFlow to implement this security feature. For the general encryption application to work the project must be structured in such a way:

First, a prototype is constructed via TensorFlow to demonstrate that this idea indeed works. This prototype neural network will be trained using the MNIST database, which contains tens of thousands of labeled training data for images of handwritten digits. This will include an elementary client-server interaction in which the client sends obfuscated inputs to the server, which in turn runs the neural net and returns to the client the final answer.

Next, the ideas learned in building the prototype will be generalized to design the obfuscation framework. Here the integration with TensorFlow becomes a key point of discussion and development. Once the application is integrated with TensorFlow, the implementation of the design for the Python module can be initiated. This phase also involves unit and implementation testing.

The final phase involves practical use of the application by using the module in a particular use case and then comparing the performance of this use case implementation with standard encryption schemes to examine the boost of performance in settings where the input is encrypted by the client and decrypted by the server. It is more likely to see a performance boost in use cases involving high throughput of secure data to a remote Machine Learning (ML) service (remote ML server).

2.2 Product Functions

ID	PF1
Title	**Obfuscating an input**
Description	The user must be able to obfuscate input images using the computed kernel keys as parameters. The output must be in the same format as the input, and should be compatible with the neural network.
Priority	**High**
ID	PF2
Title	**Setting up a topology for the neural net**
Description	The prototype must set up the neural network, ensuring that the first layer has more neurons or nodes than the second layer. This ensures that the weight matrix has kernel keys.
Priority	**High**
ID	PF3
Title	**Acquiring the kernel keys**
Description	The prototype should be able to calculate the kernel of any provided matrix. This can be implemented as a separate module or extracted from existing matrix packages. The module should output kernel vectors that can be used for input obfuscation.
Priority	**High**

2.3 User Classes and Characteristics

The types of users using this application will be clients who require encryption and security in their machine learning systems. These users will usually deal with lots of private data and ideally have a remote machine learning server used for transmission. Some examples of such users would be hospitals that deal with highly private data transmission (like X-ray, test results) to machine learning servers or Defense Aviation companies that need to use machine learning systems on confidential data. If the user is utilizing a network for transmission of input to the neural network (in cases where there machine learning server is remote) then security of data becomes a huge concern and there is a risk of corruption or attack. In such user classes, complete obfuscation of the input helps in optimizing the encryption process especially in cases of high throughput input data.

The type of user using the application will impact the functionality of the application . The different types of users might be interested in the various functionalities offered by the main module: Obfuscating an input, setting up a topology for a neural net and acquiring the kernel keys (kernel vectors of the linear transformation). All user class can use every functionality of the module.

2.4 Operating Environment

Since the application is a python module built on top of TensorFlow, the user must have an operating environment that supports python and TensorFlow: these involve most commonly used OS environments like macOS, UNIX, Windows etc.

2.5 Design/Implementation Constraints

One constraint is that it is unknown if it is computationally hard to break this encryption scheme since we have not proved it. Because of this we refer to the process as obfuscation or pseudo-encryption.

Another constraint is the communication method between the client and server. The server must define a communication protocol for the user to follow to ensure the authenticity and security of the client connection. Since client-server communication is a critical part of the application handling this constraint is important.

Another constraint is the functioning of the neural net. The application only controls the inputs to the neural net not the underlying functionality hence the performance of the neural net will limit the performance of the application if it were to act as a bottleneck.

2.6 Uncertainties

The key uncertainty in this application is the integration process of the python module with the TensorFlow interface. There needs to be a design phase designated to design the architecture for the integration of the module with TensorFlow. There should be a complete analysis of the TensorFlow APIs and how the python module needs to interact with them to ensure complete efficiency.

Another uncertainty is the process to prove that the obfuscation technique used in this application is computationally hard to break into. There needs to be a solid mathematical base to prove the validity of this technique. Proving the technique as computationally hard will improve the credibility of the application.

3 External Interface Requirements

3.1 User Interfaces

The objective of this project is to build an 'Obfuscation' Library as a layer on top of Python/ TensorFlow. As such, this project does not have any user interfaces since the users will be programmers who wish to 'import' the library to be used for obfuscation in their own applications and networks.

3.2 Hardware and Communication Interfaces

As mentioned in Section 3.1 (User Interfaces), the objective of this project is to build an 'Obfuscation' library that will obfuscate an input image, perform some transformation on it, and then have it be recognized by the same neural net that recognizes the original image. In other words, this is a software build for use by other software applications and as such, hardware interfaces are not relevant.

3.3 Software Interfaces

Given that this project is a software for software applications, there are some integral software interfaces that the library must guarantee. Depending on their use, these interfaces can be broadly divided into two categories- Configuration-Time Interfaces (3.3.1) and Run-Time Interfaces (3.3.2)

3.3.1 Configuration-Time Interfaces

- **Setup Neural Network**: The programmers Neural Network must be passed as input so that the 'Obfuscation' library can use it to create an appropriate Neural Network that has the key property that each layer has less nodes than the previous one.
- **Obtain Neural Network Weight Matrix**: Unraveling the newly created Neural Network means obtaining its weight matrix (unique to every neural network).

- **Calculate Kernel Keys**: Once the Obfuscation Library knows the created Neural Networks weight matrix, it can calculate the corresponding kernel keys which will be used in the obfuscation of an input image at a later time.

3.3.2 Run-Time Interfaces

- **Parse Input Image**: The library must be able to parse the original image supplied by the client/ user as input that it will later perform obfuscations on.
- **Perform Input Image Obfuscation**: As is clearly defined by the mathematics in this document, obfuscation means performing numerous and random linear transformations of the kernel values on the input image. In Layman's terms, this means changing the byte-by-byte/ pixel-by-pixel representation of the image, making the image unrecognizable to humans and most machines, hence the term- 'Obfuscation'
- **Test Neural Network with the Obfuscated Image**: Once the client has completed obfuscation of the input image, it is sent across the network to the server. On the server side, the obfuscated image is then passed to the same neural network that was created during the setup of the topography. The beauty of the transformations performed is that these transformations are completely invisible to the created neural network. When run with the obfuscated image, the neural network will recognize even the most 'brutally' obfuscated images, that a human couldn't dream of catching. Voila! This is the heart of the pseudo-encryption.

4 System Features

4.1 Prototype Application Features

ID	FR1
Title	**Download the Fashion-MNIST dataset**
Description	The prototype must be able to download the Fashion-MNIST dataset using the Tensorflow library. This was an arbitrary choice to display the functionality of the system. In fact, the product should be able to handle multiple datasets.
Requirement	To set up a test neural network and demonstrate obfuscation.
Dependency	N/A
ID	FR2
Title	**Set up topology of neural network**
Description	The prototype must set up the neural network, ensuring that the first layer has more neurons or nodes than the second layer. As proven before, this mathematically insures that the weights matrix has a kernel.
Requirement	To ensure that the weight matrix has a kernel for obfuscation.
Dependency	N/A

ID	FR3
Title	**Train neural network**
Description	The prototype must train the neural network with the Fashion-MNIST dataset. The Tensorflow package will be used extensively within the process, and the result should be a neural network with the properties needed for obfuscation.
Requirement	To acquire a test weight matrix, extract kernel vectors, and demonstrate the effect of obfuscation on real input data.
Dependency	FR1, FR2

ID	FR4
Title	**Extract weight matrix from neural network**
Description	The prototype must be able to extract the weight matrix from any given neural network. This will be implemented as a separate module, and incorporated into the final product design.
Requirement	To compute the kernel of a neural network and obfuscate input data accordingly.
Dependency	FR3

ID	FR5
Title	**Compute kernel keys**
Description	The prototype should be able to calculate the kernel of any provided matrix. This can be implemented as a separate module or extracted from existing matrix packages. The module should output kernel vectors (or kernel keys) that can be used for input obfuscation.
Requirement	To obfuscate the input while maintaining the correct output from the neural network.
Dependency	FR4

ID	FR6
Title	**Obfuscate input image with kernel keys**
Description	The user must be able to obfuscate input images using the previously computed kernel keys as parameters. The output must be in the same format as the input, and should be compatible with the neural network.
Requirement	To test the neural and demonstrate mathematical concepts behind the product.
Dependency	FR5

ID	FR7
Title	**Test neural network on original image**
Description	The prototype must test the neural network on the original image, and use the result as a baseline for comparison with the obfuscated image. This will be handled by the Tensorflow package completely, and doesn't depend on the obfuscation module.

Requirement	To use as a control and test whether obfuscated images produces the same result.
Dependency	FR3
ID	FR8
Title	**Test neural network on obfuscated image**
Description	The prototype must also test the neural network on the obfuscated image. The output from the neural network should be the same as with the original image.
Requirement	To ensure that the obfuscation doesn't change the output of the neural network.
Dependency	FR3, FR5
ID	FR9
Title	**Allow repeatability with different input images**
Description	The previous process should be repeated with various input images. Every test should result in the same output from the neural network for the original and the obfuscated image. The input images will be in the same format (as accepted by the neural network), however, they must cover a wide range of the 10 possible classifications.
Requirement	To ensure that obfuscation doesn't modify neural network output for a wide range of possible input images.
Dependency	FR6, FR7, FR8
ID	FR10
Title	**Allow side-by-side viewing of two images**
Description	The prototype should allow the user to view the input image and the obfuscated image side-by-side to understand the impact of obfuscation. This is an added functionality for demonstration purposes, and will not be provided in the final product.
Requirement	To understand the effect of obfuscation and demonstrate product functionality.
Dependency	FR6

4.2 Prototype Client-Server Features

The following features are minimalistic and focus on the key functions in the client-server model. More detailed implementation and design constraints will be established throughout the development process.

ID	FR11
Title	**Obfuscate input image (with public kernel keys)**

Description	With the kernel keys available to the client, they should be able to use the obfuscation module to modify the input data. The result should be in the same format as the original data, and compatible with the neural network.
Requirement	To transfer input data to the server-side neural network in an encrypted, secure manner.
Dependency	FR1-9
ID	FR12
Title	**Securely transfer image to server**
Description	The client should be able to communicate with the server-side neural network to transfer images and receive results. This will not be included in the product features as it should be pre-established by the system integrating the obfuscation module. For the prototype, it will help demonstrate the client-server model.
Requirement	To demonstrate the client-server model overarching the obfuscation module.
Dependency	N/A
ID	FR13
Title	**Test image on server-side neural network**
Description	The server should receive the inputted data and run the neural network. This feature will also be pre-established by the system that integrates the module. For the prototype, it will help demonstrate the obfuscation within the client-server format.
Requirement	To view the effect of obfuscation within the client-server model.
Dependency	FR11, FR12
ID	FR14
Title	**Return neural network result to client**
Description	The server should return the result of the neural network back to the client. This feature will also be pre-established by the system's original communication interfaces.
Requirement	To ensure that obfuscation doesn't affect the output of the neural network.
Dependency	N/A

4.3 Product Features

The final product features are heavily dependent on work with the prototype. Unfortunately, this means that these features cannot be covered in full detail. The following three features establish the basic functionality of the obfuscation module in a broad sense.

ID	FR15
Title	**Setting up the neural network**
Description	The client should be able to use the obfuscation module to properly set up the neural network. The module should ensure that the first layer of the neural network has more neurons/nodes than the second layer. This will be an extension to the already existing Tensorflow topology features for neural networks.
Requirement	To ensure that the neural network weight matrix has a kernel, and the obfuscated images produce the same results as the original.
Dependency	N/A

ID	FR16
Title	**Extracting kernel keys**
Description	The client should be able to use the obfuscation module to extract the kernel keys from the existing neural network. The function will work under the assumption that the neural network was properly initialized. This will require some form of client-server communication, which doesn't need to be secure. Our model should ensure that the kernel keys cannot be used to decrypt the obfuscated data.
Requirement	To obfuscate the input data correctly and ensure that data cannot be intercepted and interpreted.
Dependency	N/A

ID	FR17
Title	**Obfuscating Input Data**
Description	The client should be able to use the obfuscation module to correctly obfuscate the input data based on the extracted kernel keys. The kernel vectors require coefficients that can be customized or inputted by the client itself. This can be considered a second layer of key encryption.
Requirement	To securely transfer input data to the server-side neural network without affecting the outputted classification.
Dependency	N/A

5 Nonfunctional Requirements

5.1 Performance Requirements

ID	PR1
Title	**Topology setup**

Description	The choice of topology setup of a neural network affects the accuracy of the network itself. These different setups also affect how quickly a neural network converges to the right solution. Thus, it is necessary for performance requirements that a suitable topology be chosen such that the efficiency of the neural network is optimal.
Priority	Medium

5.2 Security Requirements

ID	CR1
Title	**Encrypted data transfer**
Description	The data transfer between the client and the server should be secure. The encrypted data that is being sent over should be unrecognizable and securely sent over the network. Since this application promises security of data transfer as its primary function, it is imperative that the system protects the privacy of the data being communicated over the network.
Priority	High

5.3 Software Quality Attributes

ID	QR1
Title	**Reliability**
Description	The obfuscation should be reliable i.e. the encrypted input image should be non-recognizable by a human. This is the main objective of this application and thus it is extremely important to ensure that the system protects the privacy of the data being communicated over the network. A failure in this attribute could be dangerous for this product.
Priority	High
ID	QR2
Title	**Correctness**
Description	The obfuscation should also be correct. The input image should be encrypted in a way such that it is not only unrecognizable by a human but also the same image when processed by the receiving neural network. A failure in this attribute could render the application meaningless and unnecessary.
Priority	High

Glossary

Fashion-MNIST A large database of images of clothes with 10 labelled classes.. 5

kernel keys Kernel vectors of the first linear transformation in the neural network.. 3

Machine Learning (ML) It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. 3

pseudo-encrypt It is not yet clear whether the obfuscation scheme employed can be deemed as encryption. Some mathematical analysis has to be done to determine the exact conditions under which the obfuscation is actual encryption. 2