Machine learning using Homomorphic Encryption

Subhashish Dhar
BITS, Pilani
CSIS Department
h20190023@pilani.bits-pilani.ac.in

Sanchita Badkas
BITS, Pilani
CSIS Department
h20190520@pilani.bits-pilani.ac.in

Machine Learning (ML) has impacted various domains from retail industry to law enforcement bodies. The key ingredient to ML is data. The traditional process involved collecting the data at one place where the machine is trained. Later, the trained machine is fed with data for which predictions are to be made. However, with rise in the ML applications, the ML process is no longer centralized. Also, most of the time data owners are concerned about the privacy of their data which hinders the process of machine training. In order to address these concerns, privacy preserving ML was researched on using a technique called homomorphic encryption. This paper sheds light on how HE can help tackle the privacy concerns with sensitive data for ML applications.

Keywords—Homomorphic encryption, privacy preserving machine learning, secure multiparty machine learning)

I. Introduction

Machine learning has been around for a long time and has found it's way inside various domains. ML has been now providing aids to make business decisions that are relevant to current trends. ML is helping in field of finance by analysing patterns of various industries in order to predict the stock markets. Cyber crimes are detected using ML which helps the law enforcement bodies to take actions before the crime is committed. Face recognition is also an useful aid in various domains. But in order for the machine to understand the patterns it needs huge amount of data. Depending on the complexity of the problem, the need for data can reach upto few tera bytes as well. Data owners have a huge opportunity to monetize this need of data. However, most of the times the data is sensitive. For eg. A company who wants to get predictions for it's future stock value may not want to reveal it's financial records oto the model owners. This is what resulted into a research branch of privacy preserving machine learning. Homomorphic encryption plays a key role in it.

The paper is organized as follows,

Section II covers the various terminologies that are prerequisites for understanding the role of homomorphic encryption in machine learning applications Section III mentions a particular scenario in which HE would be used. Section IV talks about the related research that has been going around for privacy preserving ML. Section V mentions the datasets we have used for this particular paper. Section VI lists the python libraries we have used for the demo. Section VII and VIII talk about the experiment and it's result.

II. Background

A. Machine learning:

Machine Learning needs no introduction in today's day and age to highly data driven IT industries trying to optimise and personalise trends recommendations according to a person's choices. Machine learning is the ideal data analysis tool which automates analytical model building, build systems which can learn from data, identify patterns without having to explicitly code for any of them. Most IT industries having or generating large amounts of data use machine learning nowadays in conjunction with data mining to make intelligent use of this data at their disposal.

Various machine learning models are used for different applications like for classification, models like decision tree, naive Bayes classifier, Support Vector Machine etc. are used while random forests, polynomial regression, neural networks etc. are used for regression. The decision for choosing a model is highly dependent on various characteristics of data like linearity or correlation between the variables, number of hyperparameters in the model, available computing power etc. In our demonstrations, we have used the Neural Network for classification of datasets. This is a type of Supervised learning where the machine is fed with input data as well as corresponding correct output data known as 'labels'. The algorithm tries to find the correlation between the input and output and generalise it over unknown data to predict these labels.

Machine learning does have a steep learning curve and sometimes requires deep domain knowledge to make the most out of it. Several newer professions such as data scientists and

engineers have been formed for people who specialise in this field. But this limits the reach of this technology to much larger businesses which can afford to invest in the manpower required. Also, a lot of computing power is required to train these models when we are talking about huge datasets with millions of data points. This is especially true for Deep Learning applications where feature engineering is skipped and the task of making sense of the data is left up to the machine. Machine learning as a service was developed to alleviate these roadblocks, which is discussed in the next section.

B. Machine Learning as a Service(MLaaS):

Machine learning as a service (MLaaS) is introduced nowadays to combat these issues. They are various cloud-based platforms that cover data pre-processing, model training, and model evaluation, with further prediction. Prediction results can be obtained locally through REST APIs. These are third party providers which already have a trained model for your required domain and offer prediction services so that all the complexity of defining and training your own model is eliminated.

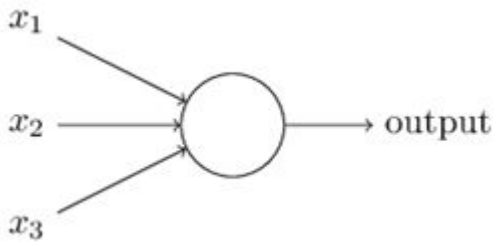
Amazon Machine Learning services, Azure Machine Learning, Google Cloud AI, and IBM Watson are four leading cloud MLaaS services that allow for fast model training and deployment. These service providers have ready-made solutions for the majority of popular machine learning applications, including recommender systems, forecasting, image and video analysis, advanced text analytics, machine translation, automated transcription, speech generation, and conversational agents.

Using these systems naturally brings us to the question of data privacy. When building Machine Learning as a Service solutions (MLaaS) complying with latest regulations like GDPR, its extremely critical to maintain privacy of both the model and the data especially in health or in finance where data is extremely sensitive. The model parameters are a business asset while data is personal data which is tightly regulated.

Our work here looks at one of these libraries, PySyft from Openmined which aims to solve this problem. It uses a form of homomorphic encryption to encrypt the training data and if required also the model. The model is trained on this encrypted data. This ensures that private sensitive information like finance records or medical records can be safely shared with MLaaS services for the purpose of building a domain specific model, and get predictions. Also MLaaS service providers can use the same logic to encrypt their model training weights, which affects the performance of the model. This is mostly a trade secret which the business would like to keep secret for competitive advantage.

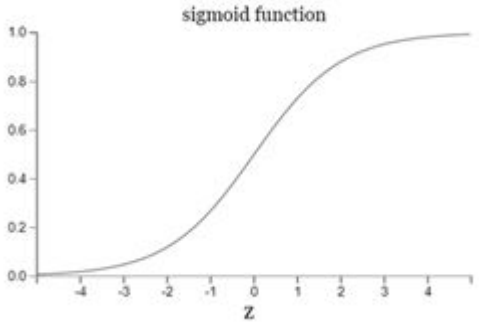
C. Neural Networks:

Neural network is a machine learning technique used primarily to classify objects by analysing and learning from training examples. This model is inspired by the densely interconnected network of brain cells called neurons and uses the machine equivalent “perceptron”. A perceptron is a mathematical function which takes several binary inputs, x1,x2,…, and produces a single binary output:



The perceptron calculates the weighted sums of all the input values, then transforms them as an output via activation function. A type of neuron with sigmoid activation function is specially used in classification problems. Sigmoid neurons are similar to perceptrons but modified so that small changes in their weights and bias cause only a small change in their output.

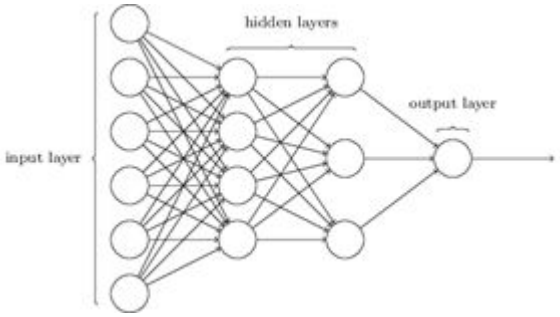
$$\sigma(z) \equiv \frac{1}{1+e^{-z}}.$$



This output between 0 and 1 is used as a confidence value of the prediction of the network, where value equal to or above 0.5 may be taken as true and below 0.5 taken as false in case of binary classification.

A neural net consists of numerous of these simple neurons that are densely interconnected in a hierarchical and non-linear structure consisting of several layers, where each layer includes several neural units.

The initial layer of these neurons, known as input units,



receive information about training examples that the network will attempt to learn about. Other units sitting on the last layer of the network and generating the final classification output are known as output units. The layers in between the input units and output units are called hidden layers. Most neural networks are fully connected, which means each hidden unit and each output unit is connected to every unit in the layers either side. The connections between one unit and another are represented by a number called a weight, which can be either positive or negative.

In their most general form, inputs fed into the network via the input units trigger the layers of hidden units, which in turn arrive at the output units. Such networks are called feedforward networks. During training neural networks learn things by a feedback process called backpropagation. The exact algorithm may be out of scope of this paper but what it essentially does is compare the output a network produces with the actual label. It uses any difference found between them to modify the weights of the connections between the neurons in the network, starting from the output layer to the hidden layer to the input layer.

D. Homomorphic Encryption:

Homomorphic encryption (HE) was proposed originally in [1], as a way to perform different operations on encrypted data. Homomorphism is a structure preserving transformation between two algebraic structures of the same type.

For example, consider a map

$\Phi : Z \rightarrow Z_{11}$, such that $\Phi(z) := z \pmod{11}$.

This map Φ, preserves both the additive and multiplicative structure of the integers in the sense that for every $z1,z2 \in Z$, we get

$$\Phi(z_1 + z_2) = \Phi(z_1) \oplus \Phi(z_2)$$

and

$$\Phi(z_1 \cdot z_2) = \Phi(z_1) \otimes \Phi(z_2)$$

where $\oplus$ and $\otimes$ are the addition and multiplication operations in Z_{11}. The map Φ is a ring homomorphism between the rings Z and Z_{11}.

Thus, HE can be used for computation of encrypted data without decrypting it. Due to this characteristic HE finds its application in privacy preserving computations for machine learning services over cloud.

E. Secure Multiparty Computation:

In the demonstrations, PySyft uses a form of encryption called secure multi party computation. SMPC 'encrypts' a value by essentially distributing it among 2 or more parties. This also enables PySyft to support federated learning techniques where the training data is shared between 2 or more workers, each worker trains the model with their own share of data, and in the end all model weights are aggregated and the mean is taken to get the model weight for the whole data. In order to decrypt a variable, all parties must agree and add their shares.

- Encryption:

This algorithm only works on integer values within the mathematical space of integer quotient ring including numbers between 1 – Q-1 where Q is a large prime number. Quoting the example given in the paper,

Given Q a large prime number and encrypting integer x
encrypt(x):
    a's share = randomly select number between -Q to Q
    b's share = randomly select number between -Q to Q
    c's share = (x-a's share-b's share)% Q

- Decryption:

Decryption is sum the shares together and take the modulus of the result (mod Q)

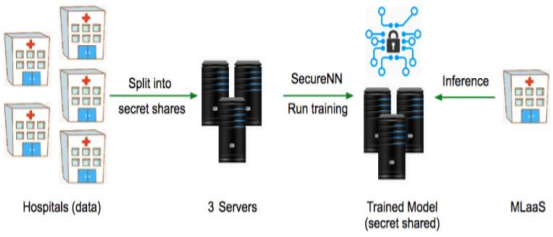
decrypt(a,b,c):
    (a+b+c)%Q

If we try to decrypt without every share, we will not get the original value x. These shares act like private keys, all of which are required to decrypt the integer value.

SMPC allows computation on the encrypted variables as homomorphic encryption. It supports addition, multiplication, and comparisons. PySyft library provides a wrapper over encrypted tensors, so that they can be used to train models using normal machine learning libraries such as PyTorch and Tensorflow. All parties in this federated learning system depend on a mutually selected party which generates the random shares and acts as "crypto_provider". It does not take part in the training process but needs to be trusted by all other shareholders to be fair.

F. *SecureNN - 3-Party Secure Computation for Neural Network Training:*

The SecureNN protocol[6] is used in PySyft library. As stated in [7] neural networks use various building blocks like matrix multiplications, max pooling, rectified linear etc. The secure NN protocol provides three-party secure computations for each of the building blocks. Using these blocks we can build secure protocols for training neural networks in a way that none of the parties involved would learn anything about the data of the other owners.



The protocol aims at making secure computations for neural networks that are scalable, provide high performance and security. They have modelled the problem as a set of M data owners who wish to execute training over their joint data using N servers. The security requirement of this model is to ensure no party learns anything about the data belonging to others. The protocol can be explained easily by the diagram from [6].

A better understanding of the protocol is explained in the following section.

III. Scenario

Consider a company 'Crypto' who is into developing Machine Learning as Service solutions for health industries. The data required to train this model developed as a part of the solution comes from various health organizations and facilities. However, the health records contain sensitive data which the organizations don't wish to expose. Let's assume two such data providers as 'Alice' and 'Bob'. Alice and Bob are also the customers of the MLaaS solutions. One solution can be that the 'Crypto' sends it's model to Alice, who trains it on her data and sends it back. 'Crypto' then sends the trained model to Bob who trains it on his data and sends it back to 'Crypto'. The accuracy of the model should ideally be increased after being exposed to more data from Bob. However, there are two problems with this scenarios,

- "Crypto" can check the parameters of the trained machine and make deductions about the data that "Alice" has, hence hampering Alice's privacy.
- "Bob" gets a half trained model, which he can train on his data and use without having to pay to 'Crypto', hence hampering Crypto's privacy.

Thus, in this scenario all the parties involved will have a 'share', which they'll exchange with one another. After the exchange, the scenario would end up with Alice and Bob, having three parts of shares,

- Their own share,
- The other party's share.
- The model owner's share.

Let's assume the parties with following shares:

Alice - AA

Bob - BB

Crypto - CC

After the exchange the resulting scenario would change to,

Alice - ABC

Bob - ABC

After Bob and Alice have trained the encrypted model they would send it back to 'Crypto' for decrypting. Thus, Alice and Bob don't learn about the model parameters and the Crypto isn't able to make assumptions about Alice's or Bob's data individually. Diagrammatically it can be represented as [8]

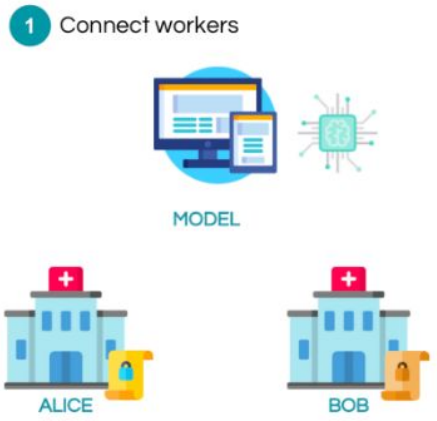


Fig : Step 1: Connecting workers

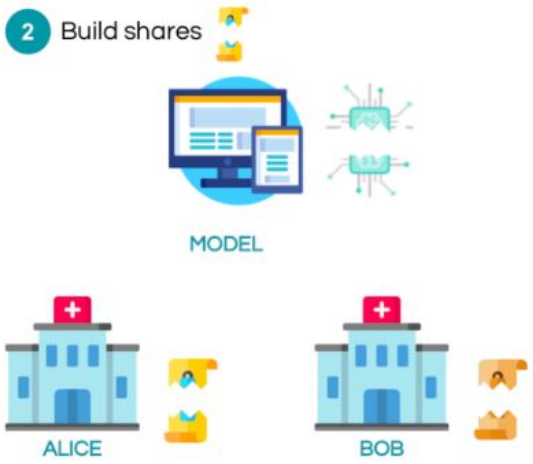


Fig : Step 2: Building Shares

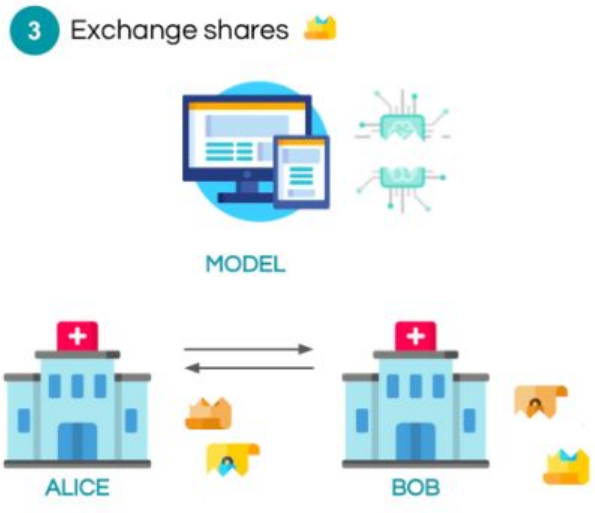


Fig : Step 3: Exchange of shares

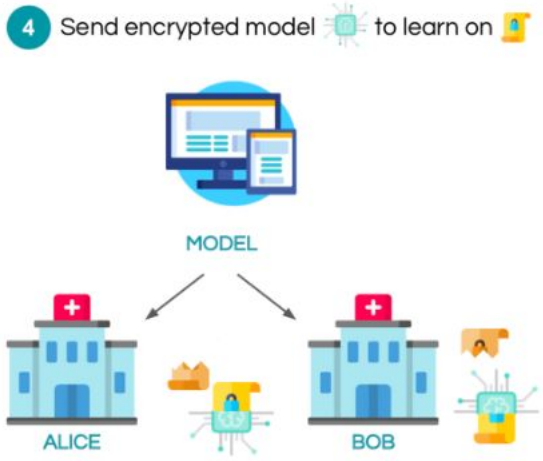


Fig : Step 4 : Distributing the shares.

IV. Literature survey

Homomorphic encryption was initially proposed in 1978 by Rivest[1], however no solid scheme was proposed for it until in 2009 by Gentry[2]. As different schemes were developed, many drawbacks of those schemes came in light. The major drawback of fully homomorphic encryption schemes proposed after 2009 was the time needed to perform the computations. This gave rise to different HE types which can be found in tada, which is the major motivation for our project. As per the paper, HE schemes can be classified into three major types as follows:

- Fully Homomorphic Encryption (FHE)

FHE represents the strongest notion of HE as it allows an arbitrary number of operations over encrypted data. The depth of circuits performing the operations is unbounded.

- Levelled Homomorphic Encryption

Unlike FHE the depth of circuits is bounded in this type.

- Partially Homomorphic Encryption

This scheme only supports one type of evaluation circuits, addition or multiplication and hence can be termed as the weakest notion compared to the other two.

Since, FHE leads to heavy computation overhead and PHE is weak, we choose LHE for our project. Various drawbacks of the above listed techniques can be found in

As HE's popularity grew with increasing concerns of data privacy, homomorphic encryption made its way into ML. Combining the machine learning algorithms with HE leads to different sets of problems. Since our project revolves around the neural network model of ML we'll be focusing on the problems encountered with this particular model which have been highlighted.

- Loss of accuracy over multiple operations:

As the number of iterations increase, the precision loss becomes more intense. In order to tackle this problem we can either use FHE but the amount of time needed to train the machine increases significantly. This can be resolved by decreasing the amount of the data used for training the machine. This method works when the patterns in the data aren't very complicated. As the complexity increases, the amount of data required to train the machine increases and it's not feasible to use FHE. Another approach can be reducing the number of iterations/epochs. This reduces the number of computations that are performed over encrypted data. Since the depth is limited, levelled Homomorphic encryption can be used to reduce the training time for that ML algorithm.

- Complexity of activation function:

Various activation functions used in neural networks can be sigmoidal, rectified linear, hyperbolic tangent etc. A

complete review of these functions can be found in [3]. Functions like sigmoidal are nonlinear polynomial functions which means more time for computation. This time is compounded heavily when we use encrypted data to train the machine. In order to overcome this shortcoming we can reduce the complexity of the activation function we use, which would affect the learning of the machine in the training phase. The algorithm might not be able to learn several non linear relationships in the data due to use of simplified activation functions. But, if the data amount is small, we can get away with using complex activation functions as the amount of time required to train the machine would be more, yet affordable. Other different approaches have been discussed in [4]

- Using 2-party communication for computation:

Using this method, requires constant communications between the model and data owners. In order to avoid the loss of precision and to have the ability to use complex activation functions the overall training process of the machine is broken into parts which are communicated to and fro between the 2 parties in a series of steps. The first step includes data owners encrypting their data and sending it to the model owner. The model owner computes the weights of the neurons in the first layer and provides that encrypted output along with the functions to the data owner. The data owner decrypts the data and performs the necessary computations on it, encrypts it and sends to the model owner. This process is recursively carried out for all the layers. A major drawback of this method is that significant information about the machine is revealed to the data owner. A solution to it was proposed which involved a cloud server on which the encrypted data is present. The model owner would now communicate with the cloud in the similar fashion stated above. Thus, the burden of computation is now shifted to the cloud reducing the time required for training of the machine. Also, since data owners won't receive intermediate results no part of the model is exposed to him ensuring privacy of the model owner.

But due to involvement of a 3rd party, researchers have preferred modifying the machine learning algorithms to use lower degree polynomials in order to work efficiently with encrypted data. Most of the research points out the fact that a machine trained with plain unencrypted data would have higher accuracy over the one trained using encrypted data. But the need of privacy trumps the need of accuracy. Various researches have been able to achieve results where the difference between the accuracy of machines trained using plain data and encrypted data was less, making the whole algorithm feasible.

V. Datasets

We have used two types of datasets for our demo,

- Wine Data Set:

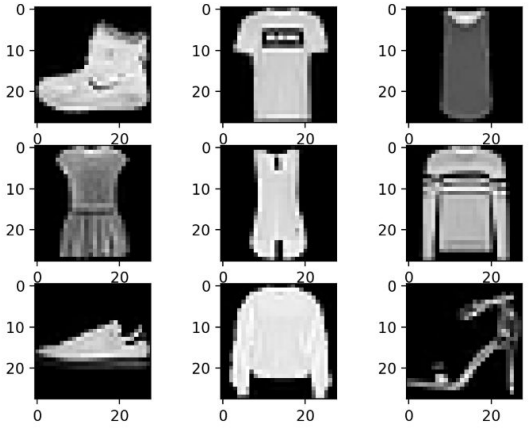
This dataset is represented as numerical tabular data of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. It contains 13 parameters on which different wines have been categorised.

The attributes are Alcohol, Malic acid, Ash, Alkalinity of ash, Magnesium, Total phenols, Flavonoids, Non-Flavonoid phenols, Proanthocyanidins, Color intensity, Hue, OD280/OD315 of diluted wines and Proline

Each instance is classified into 3 classes among class 1, class 2 or class 3.

- Fashion MNIST Data set:

Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes which are T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot



VI. Libraries

A. Pysyft:

PySyft is a python library developed by OpenMined for secure, privacy preserving ML. PySyft extends the capabilities of the commonly used machine learning library - Tensorflow and Torch. Pytorch and keras, the libraries built as wrappers over Torch and Tensorflow respectively have been used to build Pysyft.

The framework mainly focuses on three applications,

- Federated Learning:

Instead of having a central repository of data, the data is spread across different owners like Alice, Bob, Carol and Davis. The ML algorithm is trained parallely at each data owner and the results are sent back to the server where they are combined to update the model with all the information it has learnt.

- Differential Privacy

When Alice, Bob, Carol and many more are tracked for their data like their fitness data generated via body sensors the dataset with the patterns is publicly shared with the machine learning models but the information of the individual person is withheld.

- Secured Multi-Party Computation

SMPC allows different parties to perform computations over inputs while maintaining it's privacy. SMPC allows users to eradicate the need of a trusted third party for computations. Parties with private inputs compute functions without revealing their data. This is the model we have used in our project.

B. PyTorch

PyTorch is a library developed by Facebook built upon Torch which is used for ML applications like computer vision and natural language processing. Pytorch helps provide tensor computing with strong acceleration via GPUs.

VII. EXPERIMENT

A. Wine Classification on encrypted training data and model:

We used the wine dataset mentioned above as a representative of encrypted machine learning applications on tabular numerical data. This demonstrates the ability to encrypt tabular training data and create a classification model using general ML library pytorch. This training set is also indicative of medical records stored in excel sheets or relational databases.

B. Fashion MNIST dataset:

As specified above, this is an image dataset. The machine is trained over encrypted images which is representative of the X-rays which can be used to detect a particular disease or image of part of a skin to detect type of skin infection etc.

VIII. RESULT

A. Wine Classification on encrypted training data and model:

The outcome is a model capable of classifying a wine according to one of the 3 classes given its 13 observed parameters. The noteworthy thing is that the model achieved over 90% accuracy in only 5 epochs with encrypted training data. The predictions were also generated on encrypted data.

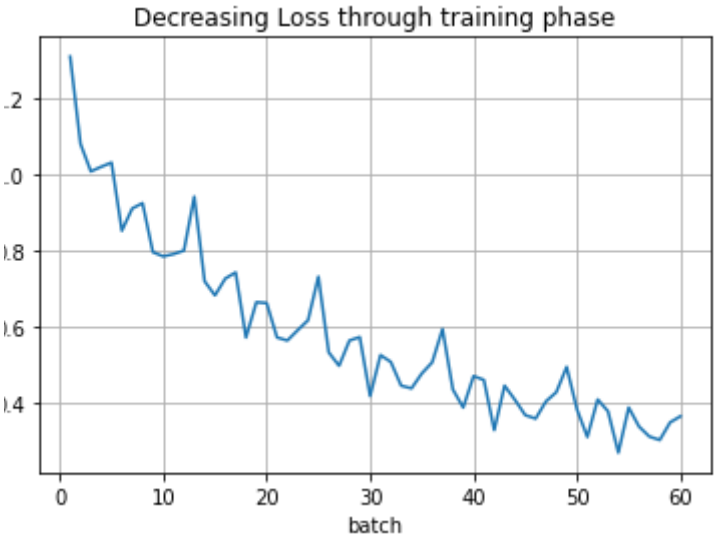


Fig : Decreasing loss with increasing training for wine dataset

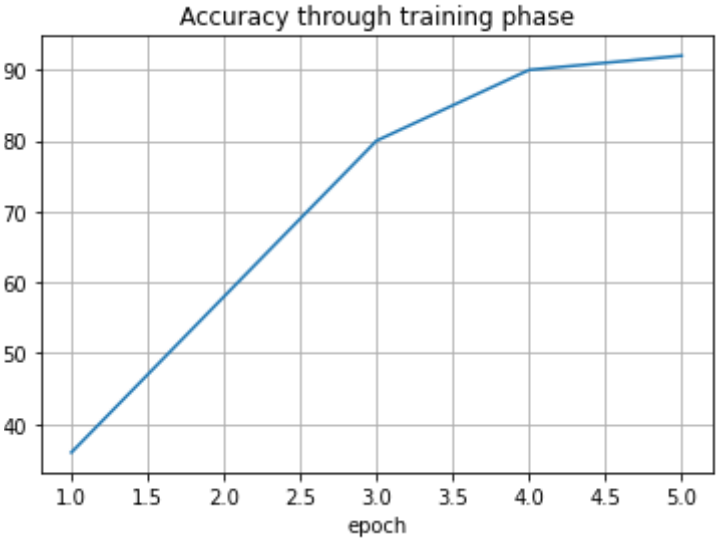


Fig : Increasing accuracy with more iterations/epochs for wine dataset

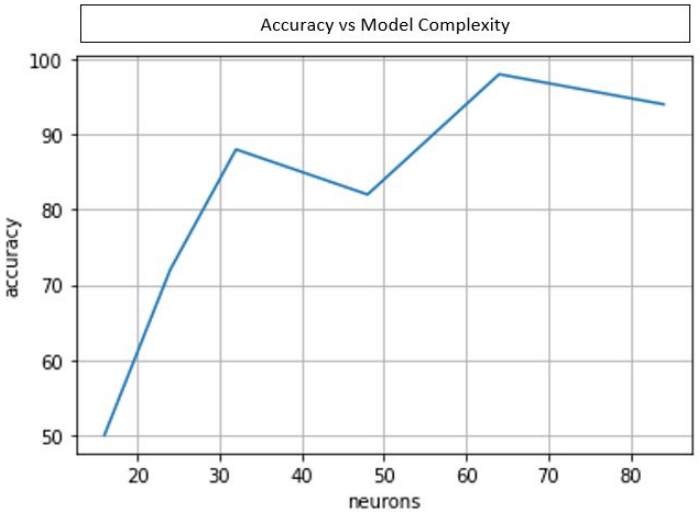


Fig : Accuracy vs complexity of the model. Accuracy increases till a point after which the risk of overfitting takes increases.

B. Fashion MNIST dataset:

We have just used 640 images out of the 44,000 images in the actual dataset. Using the limited amount of data and just 3 epochs we see the machine is able to identify the type of clothing accessory with accuracy of about 60%.

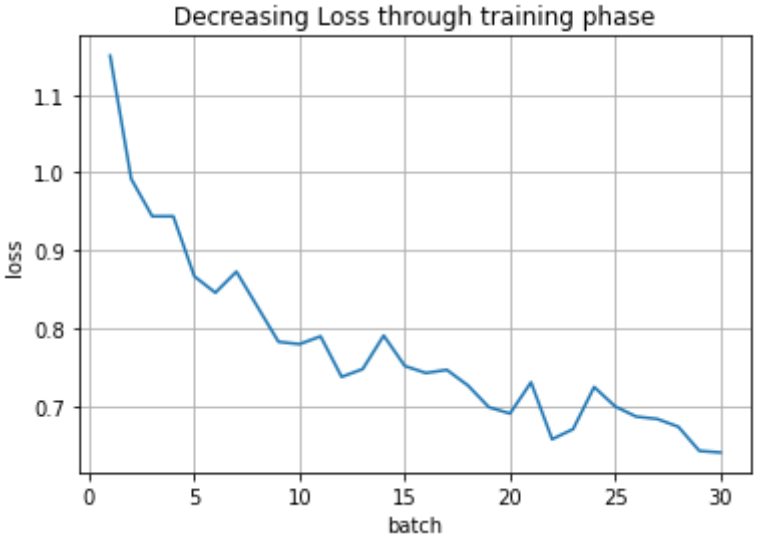


Fig : Decreasing loss with training for Fashion MNIST dataset

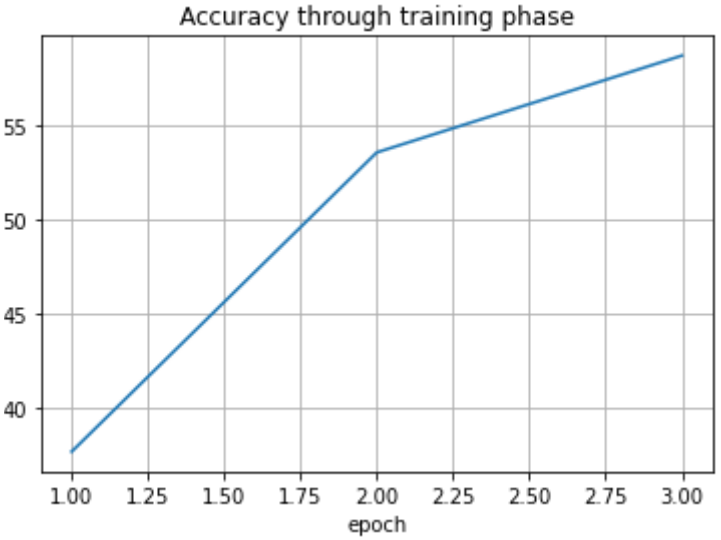


Fig: Increasing accuracy with just 3 epochs.

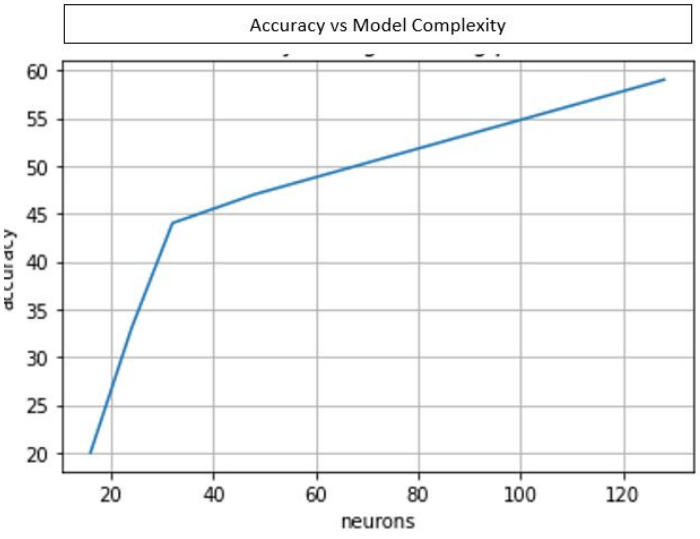


Fig: Accuracy vs model complexity for Fashion dataset.

Also, we would like to highlight the fact that the time taken for training the machine for simple wine dataset to complex fashion MNIST dataset is significantly higher when encrypted data is used.

The following graphs show the results, for wine dataset the accuracy achieved is similar or both types of data, but time taken for the machine to learn the patterns is higher for encrypted data.

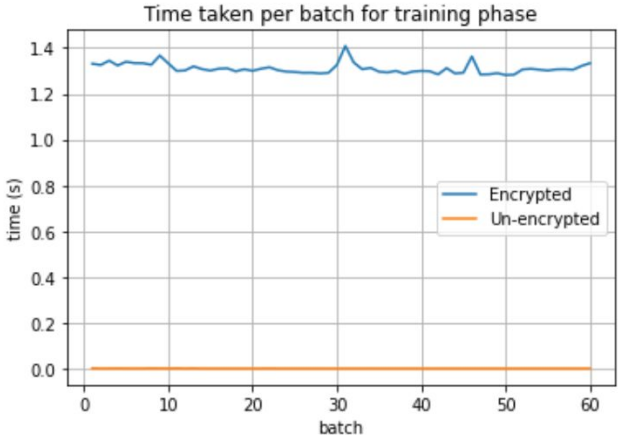


Fig : Time taken for machine to get trained on encrypted vs unencrypted data

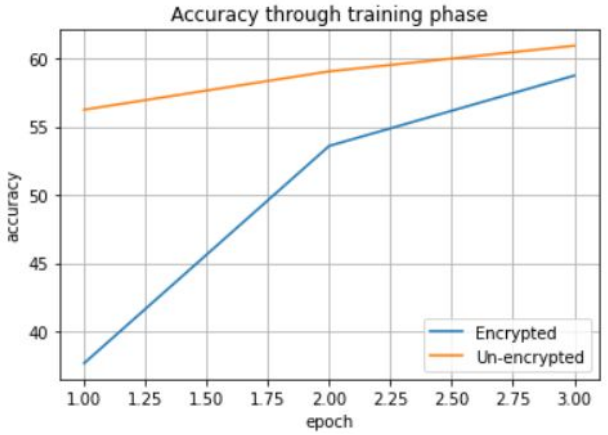


Fig: Accuracy for unencrypted data is higher than encrypted one.

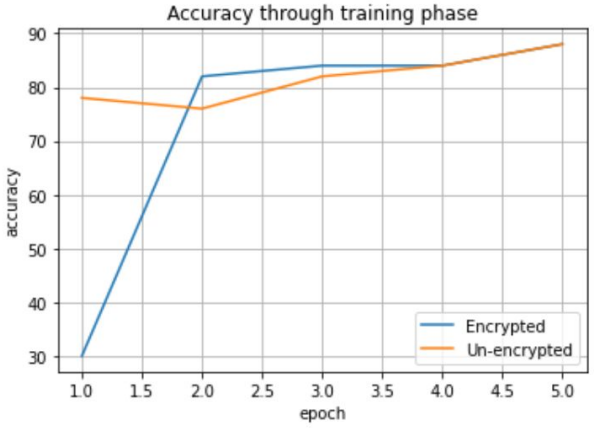


Fig : Accuracy vs Epoch for encrypted vs unencrypted data.

Similarly, we got the following results for the Fashion MNIST dataset.

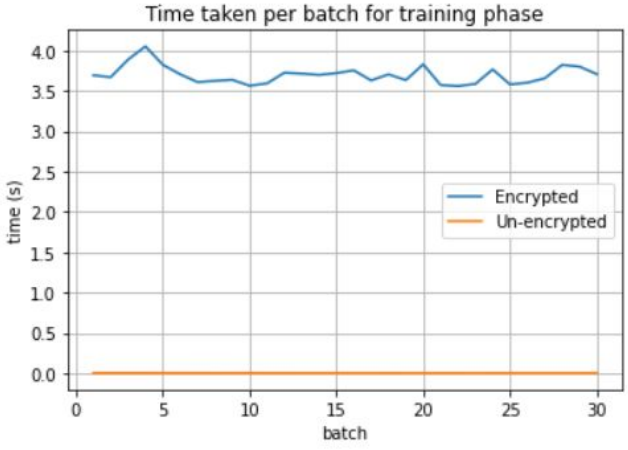


Fig : Time taken per batch for encrypted vs plain data.

References

[1]Databank And Privacy Homomorphisms by Rivest, Ronald L, Adleman, Len, and Dertouzos, Michael L in 1978.

[2]Fully homomorphic encryption using ideal lattices by Gentry, Craig in 2009

[3]A Review of Activation Function for Artificial Neural Network by Andrinandrasana David Rasamoelina, Fouzia Adjailia, Peter Sincak in 2020

[4]Efficient Evaluation of Activation Functions over Encrypted Data by Patricia Thaine, Sergey Gorbunov and Gerald Penn

[5]A generic framework for privacy preserving deep learning by Theo Ryffel, Andrew Trask, Morten Dahl, Jason Mancuso and Daniel Rueckert in 2018

[6] SecureNN: 3-Party Secure Computation for Neural Network Training by Sameer Wagh, Divya Gupta, and Nishanth Chandran in 2018

[7]CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy by Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing in 2016

[8] Openmined Pysyft Tutorial