Privacy Preserving Neural Networks

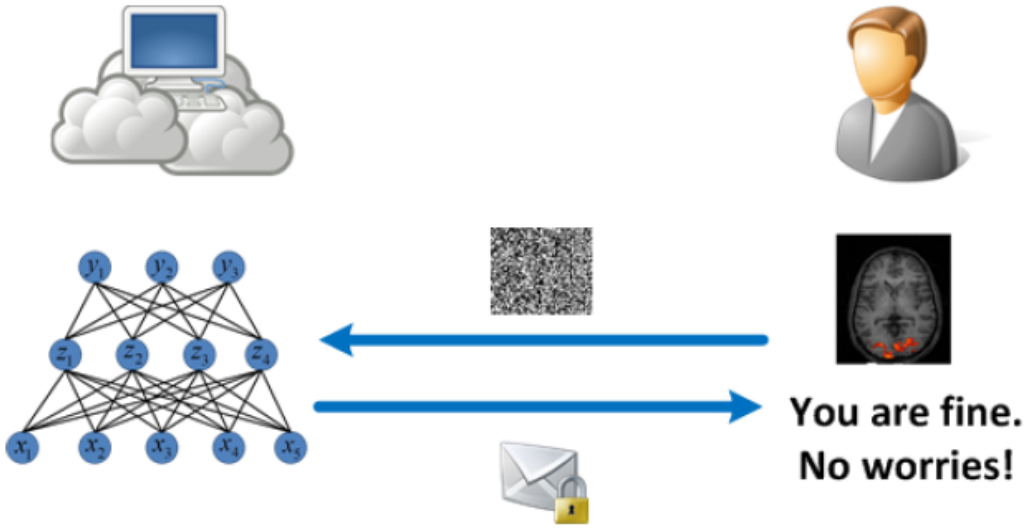
Shreya Garge

August 20, 2018

Machine Learning and Privacy

- Machine Learning : Learns from big data to think for itself and make predictions
- Presents a whole new threat to privacy by opening large amounts of data for analysis on a whole new scale.
- Will compromise the privacy of the individuals represented in the data sets.
- These models are also vulnerable to hackers in many ways :
  - Steal private confidential user data and use it with malicious intent.
  - Edit or inject into the training data to compromise the final results of the algorithm to influence decisions.
  - Steal the model itself for its intellectual value.
- Research in ML has been around for a long time, but research towards privacy and security in ML has started very recently and is still in its nascent stages.

Aim of this Project



Aim of this project - Preserving the privacy of a user's data while employing predictive models. user obtains a prediction on his image while :

- The server doesn't learn anything about the user's image or the final prediction.
- The client doesn't know anything about the model.

The Prediction Model : CNN

- Convolutional neural networks - proven to be the most effective for image recognition tasks.
- CNNs contain several layers of nodes (units inspired by the neuron in biological brain), first of which is a convolution layer.
- The model : weights and biases, learned by training the network on large sets of data.
- To protect the privacy - user encrypts image before sending for prediction.
- Using the same network is not possible - encrypted data will make no sense to the network.
- Need to implement a new network based on this model such that the operations being performed on the encrypted data make sense.
- HOW? - Homomorphic encryption.

Protocol specification

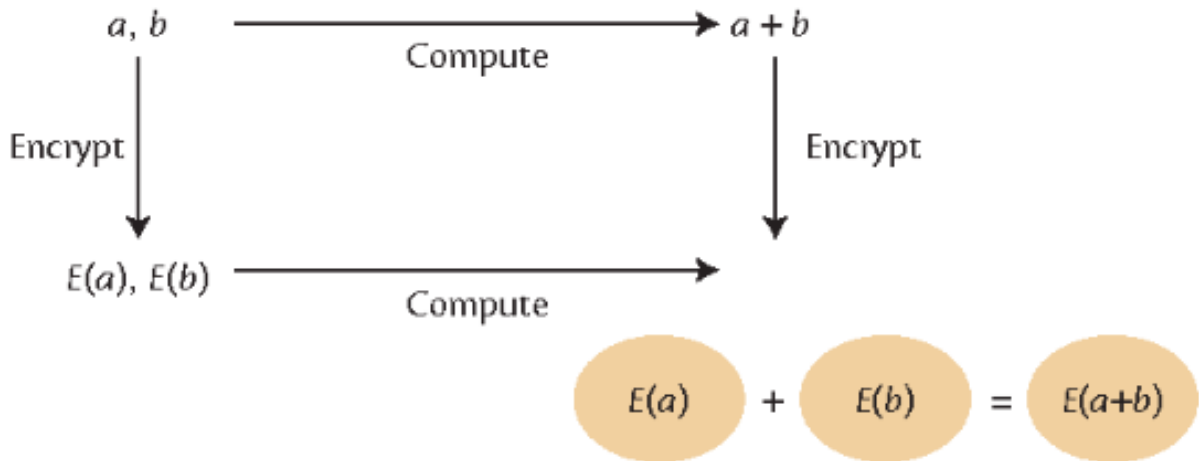
- Client sends the encrypted image, builds an object with the HE evaluation functions and parameters and sends it to the server.
- Server has the model. It runs needs to run the model on the encrypted image using homomorphic evaluations.
- The different layers in the prediction network are implemented on the server using only the HE evaluations provided by the client.
- Server has no access to the secret key, cannot decrypt at any stage.
- The encrypted result is sent back to the client, who then decrypts it and obtains the prediction.
- What is Homomorphic encryption and how do we implement the network using HE evaluations? -

Homomorphic Encryption

An encryption scheme is said to be homomorphic if mathematical operations can be applied directly to the cipher texts, preserving the operation.

$$Dec_{sk}(f(Enc_{pk}(a), Enc_{pk}(b))) = f(a, b)$$

$$Dec_{sk}(f(Enc_{pk}(a), b)) = f(a, b)$$



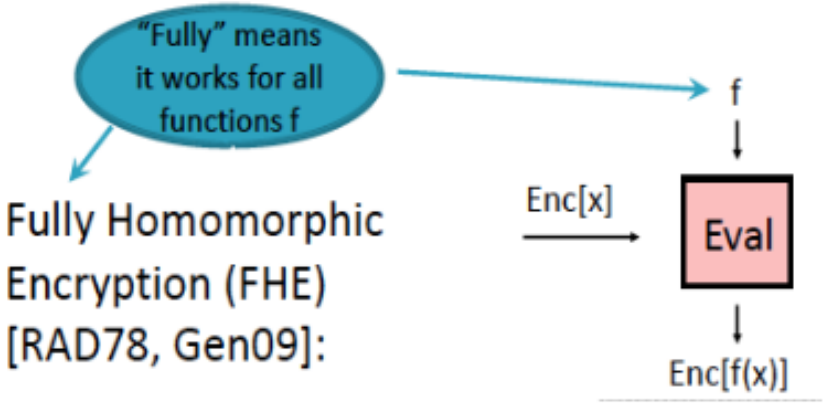
Homomorphic Encryption

- It is intuitive for deterministic encryptions, (eg naive RSA is multiplicatively homomorphic) - but the scheme would not be CPA secure.
- For semantic security, there needs to be some randomness in the encryption - called ‘Noise’.
- decryption removes the randomness - there is an upper limit of noise after which we cannot recover the message.
- The problem that arises - the noise is affected by the evaluations being performed.
- With the amount of operations involved in training a model or inferring data with an encrypted model, we quickly end up unable to decrypt the noise-polluted cipher text.

Fully Homomorphic Encryption

- ADD, MULT are Turing-complete (over any ring).
- If we can add and multiply encrypted bits - we can *AND* and *XOR* them. Thus we can express any function as a polynomial size circuit of these gates.
- An encryption scheme can be called FULLY homomorphic if it can "*encrypt 0 and 1, and ADD and MULTIPLY encrypted data*".
- Implementing FHE is feasible in theory.
- The first fully homomorphic algorithm : took 100 trillion times as long to perform calculations of encrypted data than plaintext.
- IND-CCA2 is impossible for HE: adversary can homomorphically tweak challenge ciphertext.
- IND - CCA1 FHE is still an open problem. IND-CCA1 is possible for SWHE.

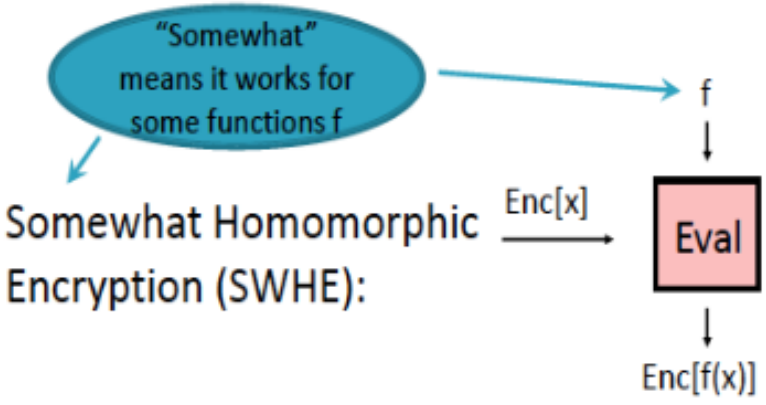
Fully Homomorphic



Fully Homomorphic Encryption (FHE) [RAD78, Gen09]:

- Arbitrary processing
- But computationally expensive.

Somewhat Homomorphic



Somewhat Homomorphic Encryption (SWHE):

- Limited processing (eg, low degree polynomials only)
- Cheaper computationally.

Practical Homomorphic Encryption

- Although these concepts of HE have been around since 1978 [Rivest], it has found practical applications only after 'bootstrapping' was introduced in 2009 by Craig Gentry.
- Gentry introduced a Practical HE using ideal lattices - practical because the noise could be dealt with using bootstrapping.
- Bootstrapping - a recrypt step during which we decrypt a previous encryption inside a new encryption.
- allows for the elegant decryption-inside-encryption, without having to remove the outer encryption before decrypting the inside.
- The benefit: we are removing all existing noise in the cipher!
- Somewhat HE + bootstrapping gives somewhat practical Fully HE.

Building HE Part 1: The hard problem underlying HE schemes

The Ring Learning With Errors (RLWE) problem

- The polynomial Ring : $R_q = \mathbb{Z}_q[x]/\phi_{2^d}(x)$
- Discrete Gaussian error distribution χ with variance σ^2
- $s \sim R_q, \quad a \sim R_q, \quad e \sim \chi$
- Search RLWE problem : find s given noisy inner products $(a, b = s \cdot a + e \pmod q)$
- Decision RWLE problem : distinguish such (a_i, b_i) pairs from uniform $(a_i, \tilde{b_i})$ pairs.
- Hardness of these problems is because of the perturbation by noise.

Building HE part 2: The underlying construction

Polly Cracker

- *KeyGen:* secret = some point $(s_1, \ldots, s_n) \in \mathbb{Z}_q^n$. Public key: Polynomials : $f_i(x_1, \ldots, x_n) s.t. f_i(s_1, \ldots, s_n) = 0 \pmod{q}$
- *Encrypt:* From $g \sim f_i$ s.t. $g(s_1, \ldots, s_n) = 0 \pmod{q}$. Ciphertext is: $c(x_1, \ldots, x_n) = m + g(x_1, \ldots, x_n) \pmod{q}$.
- *Decrypt:* Evaluate ciphertext at the secret: $c(s_1, \ldots, s_n) = m \pmod{q}$.
- *ADD and MULT:* Output sum or product of ciphertext polynomials.
  This as such is very insecure, but when the encryptions evaluate to a small noise term instead of 0, this construction can be moved into the RLWE setting and lattice hardness can be achieved.
  This is the concept extending which, many HE schemes are constructed. One of them, the Fan-Vercauteren scheme, is used in our implementation.

The Fan-Vercauteren HE scheme

- plain texts $m \in R_t$, cipher texts $c \in (R_q \times R_q)$
- *KeyGen:* $a \sim R_q, e \sim \chi$
  $k_s \sim R_2, \quad \vec{k_p} = (k_{p1}, k_{p2}) := ([-(a \cdot k_s + e)]_q, a) \in (R_q \times R_q)$
- *Encryption:* $u, e_1, e_2 \sim \chi$ and $\Delta = \lfloor \frac{q}{t} \rceil$

  $\vec{c} = (c_1, c_2) := ([k_{p1} \cdot u + e_1 + \Delta \cdot m]_q, [k_{p2} \cdot u + e_2]_q) \in (R_q \times R_q)$

- *Decryption:*

$$m = \left[\left\lfloor \frac{t[c_1 + c_2 \cdot k_s]_q}{q} \right\rceil \right]_t \in R_t$$

The Fan-Vercauteren HE scheme

- *Addition:* $\vec{c_1} + \vec{c_2} = ([c_{11} + c_{12}]_q, [c_{12} + c_{22}]_q)$
- *Multiplication:*

$$\vec{c_1} \times \vec{c_2} = \left(\left[\left\lfloor \frac{t(c_{11} \cdot c_{21})}{q} \right\rceil \right]_q, \left[\left\lfloor \frac{t(c_{11} \cdot c_{22} + c_{12} \cdot c_{21})}{q} \right\rceil \right]_q, \left[\left\lfloor \frac{t(c_{12} \cdot c_{22})}{q} \right\rceil \right]_q \right)$$

  - Increases cipher text size
  - need to perform a 'relinearization' operation to compact the cipher text size back to 2.

Moving the prediction network into the encrypted realm

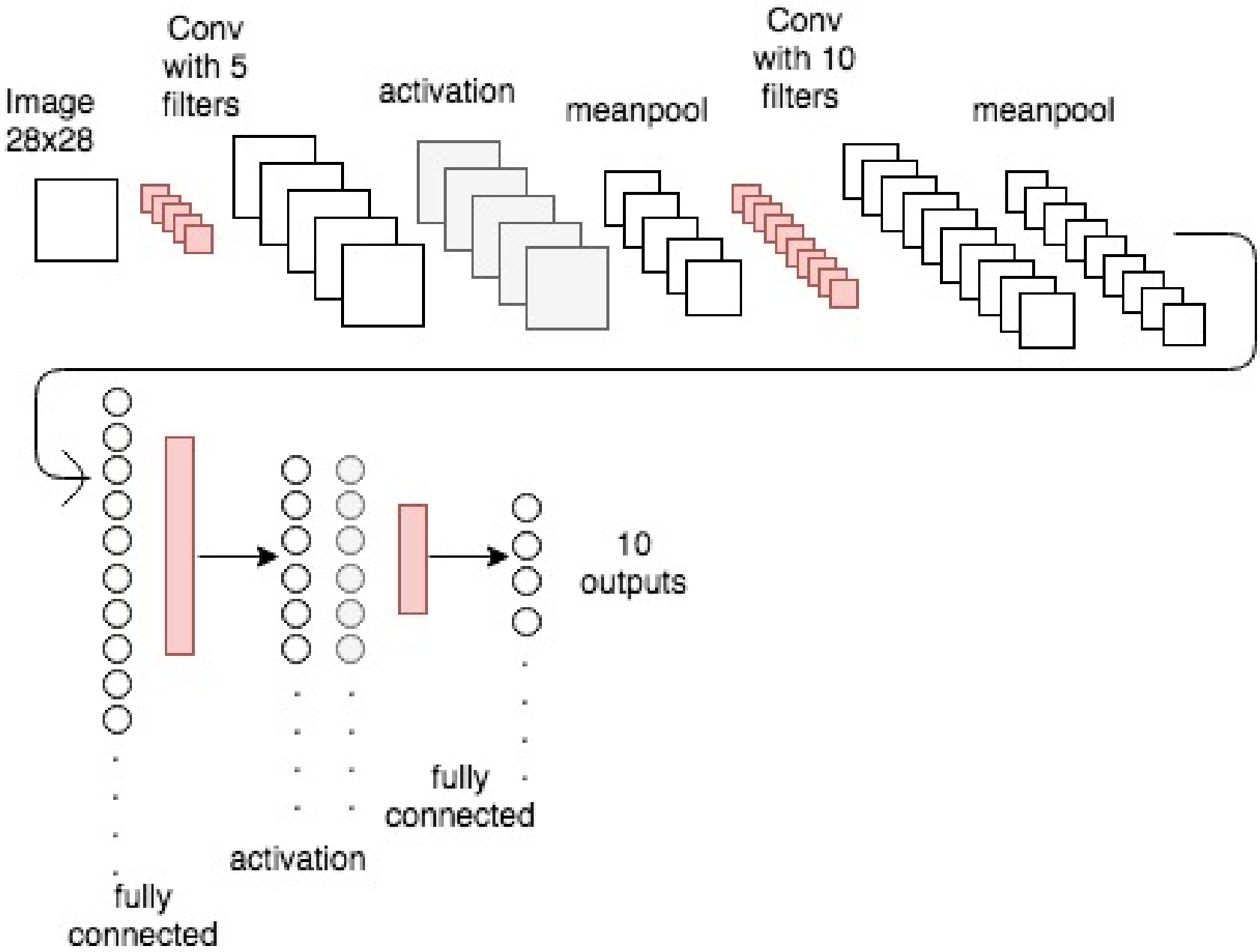


Figure: The implemented network

Encoding

- atomic constructs in Neural network : Rational numbers
- atomic constructs for HE : polynomials in R_t
- encoding scheme : maps one to the other in a way that preserves the addition and multiplication operations.
- converted to binary notation, which are taken as co-efficients of the polynomial
- Rationals - broken down to integral and fractional parts and integral part is stored in the lowest, fractional in the highest degree part of the polynomial with the signs of the co-efficients changed.
  eg : the rational 26.75 would be represented as the polynomial $-1x^{1023} - 1x^{1022} + 1x^4 + 1x^3 + 1x^1$.

Parameter selection

1. $m \in R_t = \mathbb{Z}_q[x]/\phi_{2^d}(x)$ where $\phi_{2^d}(x) = x^{2^{d-1}} + 1$. d, called *Polynomial modulus*. - effects security, cipher text size.
2. t, called *plain modulus*. - effects plain text size.
3. $c \in (R_q \times R_q)$. The parameter q, called *coefficient modulus* - more noise budget, lower security

$$\textit{noise budget in fresh encryption} \simeq \textit{log}_2(q/t)\ \textit{bits}$$

$$\textit{noise budget consumption in multiplication} \simeq \textit{log}_2(t)\ \textit{bits}$$

- If t is too small, decrypting will give wrong results if the co-efficient values go beyond the range of $\mathbb{Z}_t$ under repeated operations.
- Hence, parameters should be chosen considering all trade-offs.

Empirical results