Privacy Preserving Inference with Convolutional Neural Network Ensemble

Alexander Xiong, Michael Nguyen, Andrew So

Research Mentor: Dr. Tingting Chen

Background

- Pairing machine learning models with cloud computing provides a solution to scaling the ever-increasing workloads.
- Popular ML algorithms are typically developed without rigorous security or privacy protection [2], [3].
- Works including CryptoNets, using homomorphic encryptions, have been proposed towards enabling inference as a service [4]-[6].

System Architecture

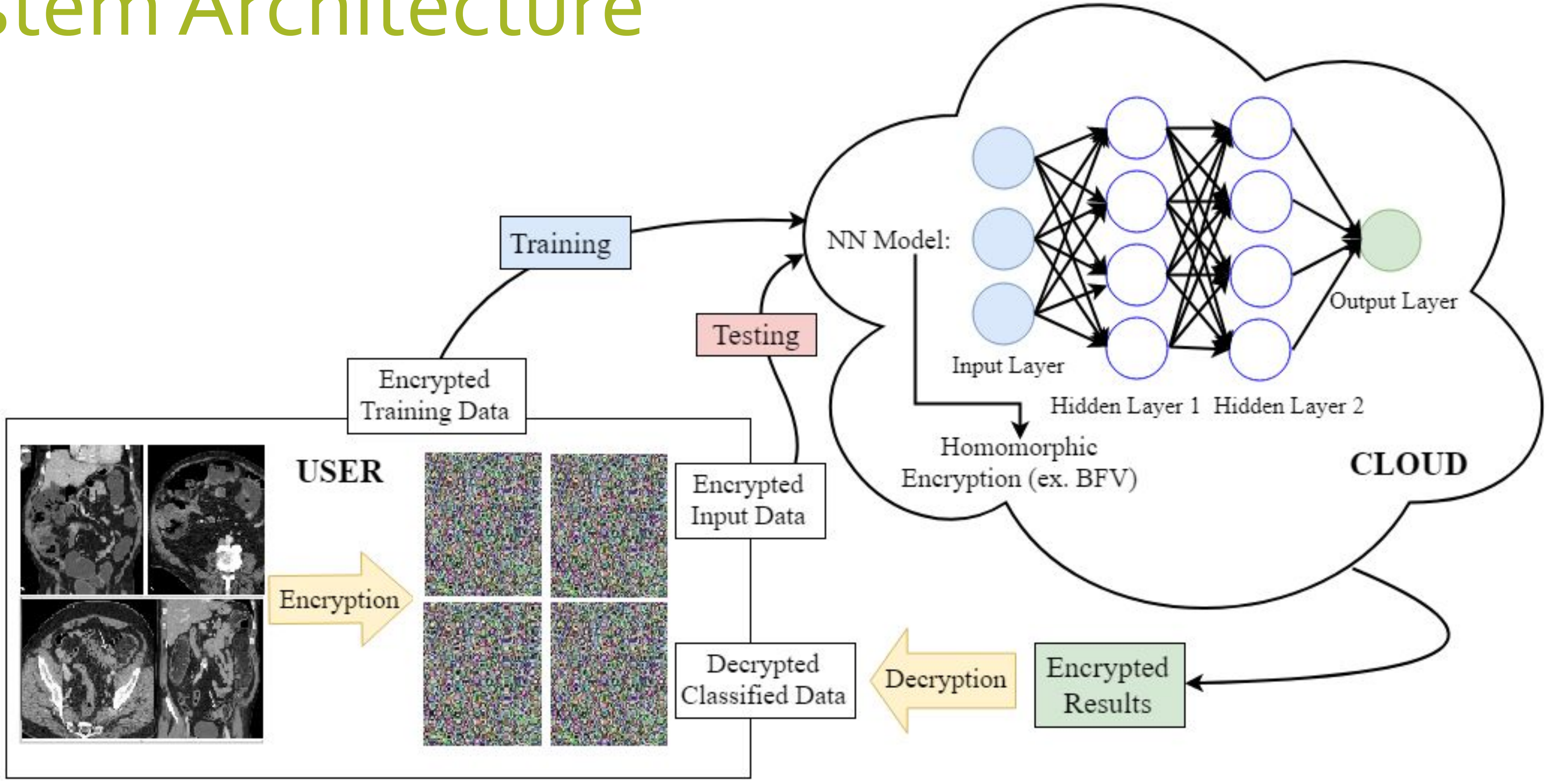


Fig 1. Architecture of Privacy Preserving Neural Network in a cloud setting

Background

Fully Homomorphic Encryption

$$\begin{tikzcd}
m \arrow[r, "f"] \arrow[d, leftrightarrow, "\text{Decrypt}", "\text{Encrypt}"'] & f(m) \arrow[d, leftrightarrow, "\text{Decrypt}", "\text{Encrypt}"'] \\
Enc(m) \arrow[r, "f"] & Enc(f(m))
\end{tikzcd}$$

Fig 2. Fully Homomorphic Encryption

- Encryption Scheme:
    - We use the Brakerski/Fan-Vercauteren (BFV) scheme, a lattice-based cryptographic scheme dependent on the Ring Learning With Errors problem [20]

Project Contribution

We further extend the capability of privacy preserving deep neural network inference, through a joint decision made by multiple deep neural network models on encrypted data, to address bias caused by unbalanced local training datasets.

In particular, we design and implement a privacy preserving prediction method through an ensemble of convolutional neural networks.

Ensemble

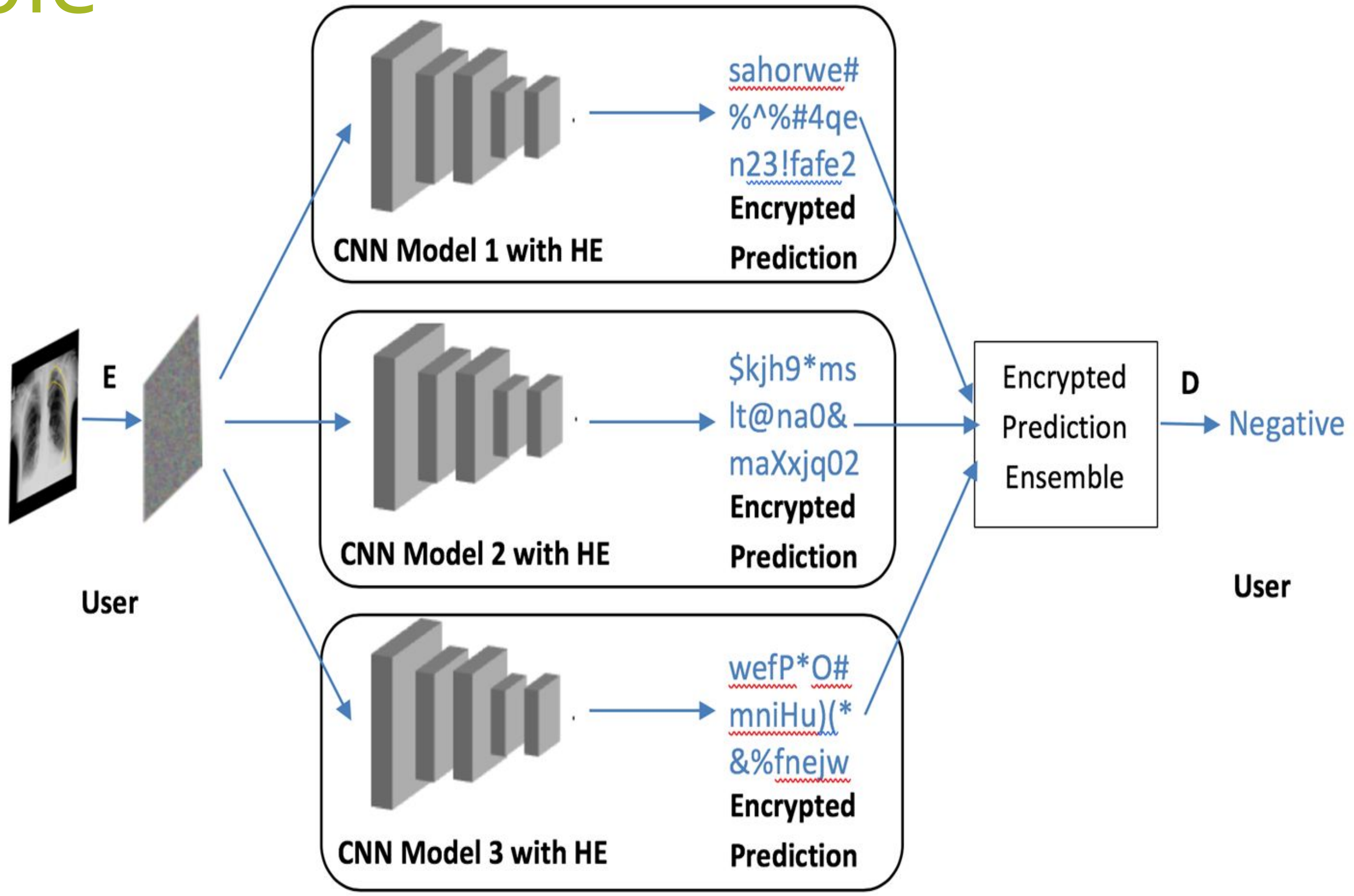
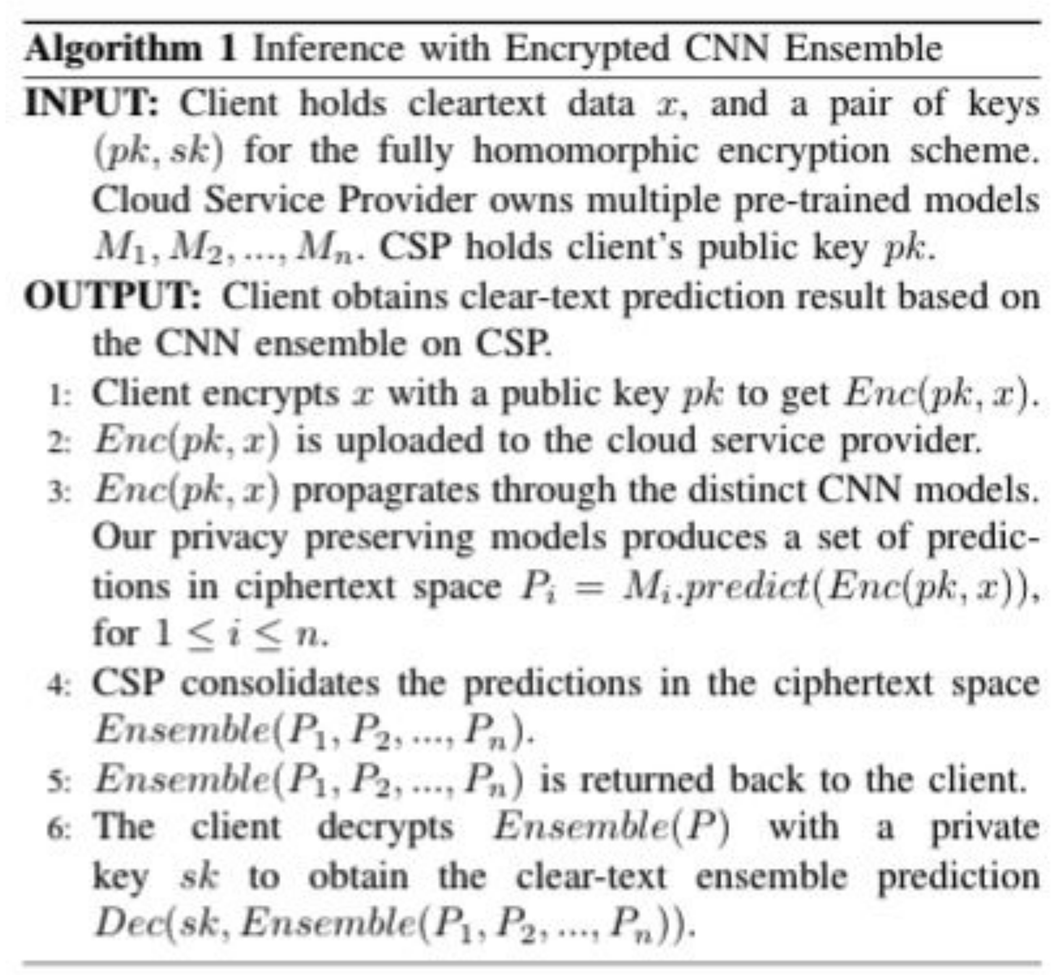


Fig. 3. Encrypted CNN Ensemble Diagram.

Security Model

- We assume that
  - All parties, i.e., the cloud service platform, certificate authority, and end-users are all honest but curious.
  - The parties do not fully trust each other.
  - End-users do not disclose private data or private key to the CSP.
  - The data that CSP receives and operates on are all encrypted.
- We will show that they can not learn the private data from the procedure of executing the privacy preserving neural network ensemble inference and from the encrypted intermediate results.

System & Security Model Design



```
Algorithm 1 Inference with Encrypted CNN Ensemble
INPUT: Client holds cleartext data $x$, and a pair of keys
    $(pk, sk)$ for the fully homomorphic encryption scheme.
    Cloud Service Provider owns multiple pre-trained models
    $M_1, M_2, ..., M_n$. CSP holds client's public key $pk$.
OUTPUT: Client obtains clear-text prediction result based on
    the CNN ensemble on CSP.
1: Client encrypts $x$ with a public key $pk$ to get $Enc(pk, x)$.
2: $Enc(pk, x)$ is uploaded to the cloud service provider.
3: $Enc(pk, x)$ propagrates through the distinct CNN models.
   Our privacy preserving models produces a set of predic-
   tions in ciphertext space $P_i = M_i.predict(Enc(pk, x))$,
   for $1 \leq i \leq n$.
4: CSP consolidates the predictions in the ciphertext space
   $Ensemble(P_1, P_2, ..., P_n)$.
5: $Ensemble(P_1, P_2, ..., P_n)$ is returned back to the client.
6: The client decrypts $Ensemble(P)$ with a private
   key $sk$ to obtain the clear-text ensemble prediction
   $Dec(sk, Ensemble(P_1, P_2, ..., P_n))$.
```

Fig 4. Algorithm 1. Inference with Encrypted CNN Ensemble

- The goal is to design an evaluation function in the ciphertext domain for a CNN ensemble that each pretrained model takes in a encrypted image as input test data.
- The ciphertext ensemble output can be decrypted by the user into the correct cleartext inference result.
- The challenge is to preserve the same properties in the ciphertext domain as in the cleartext domain for both the feed-forward inference process of individual models and the ensemble stage.

Security & Efficiency Analysis

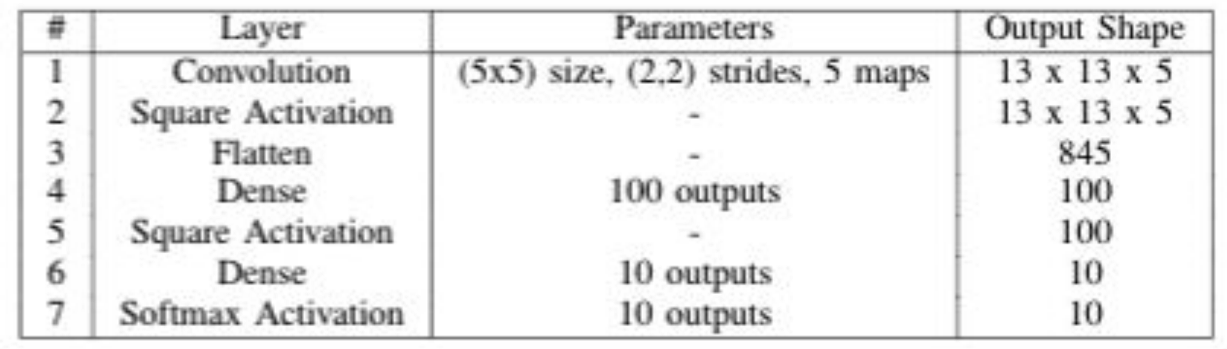
- In Algorithm 1, the message that CSP receives is the encrypted image data.
- CSP cannot learn the plaintext x, i.e., the pixel values of original image, due to the security of BFV which is based on the difficulty of solving the Ring Learning with Error (RLWE) problem [23].

- The messages transmitted between the user and CSP are Enc(pk, x) and Ensemble(P1, P2, … , Pn).
- Given the dimension of single grayscale image x as $m \times m$, the size of Enc(pk, x) is $m^2 \times n$ bytes, where n is determined by the key setup parameters in BFV (in this case, $n = 65544 \times 8$).
- The size of Ensemble(P1, P2, … , Pn) is $K \times n$, where K is the number of potential classes in the prediction results, and in our implementation, $K = 10$.

Technologies

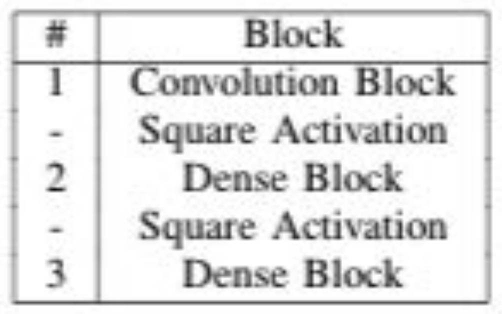
- We used C#, Python, and Keras to create our neural network which similarly follows the design set forth in CryptoNet.
- Using Python and Keras, we reconstructed the Keras equivalent of the CryptoNet neural network.
- In doing so, we are able to train our model in Keras and transfer our weights and biases into the CryptoNet.
- We also use C# to integrate the logic to produce the ensemble prediction in CryptoNet and its integration of Microsoft SEAL for homomorphic encryption functionalities.
- We use an 64 bit Intel Core i9-9900k @ 3.6 GHz with 32 GB of RAM and Nvidia GeForce RTX 2080 Ti as the base conditions to run our experiments.

Model Architecture



#	Layer	Parameters	Output Shape
1	Convolution	(5x5) size, (2,2) strides, 5 maps	13 x 13 x 5
2	Square Activation	-	13 x 13 x 5
3	Flatten	-	845
4	Dense	100 outputs	100
5	Square Activation	-	100
6	Dense	10 outputs	10
7	Softmax Activation	10 outputs	10

Table 1. Non-encrypted model architecture

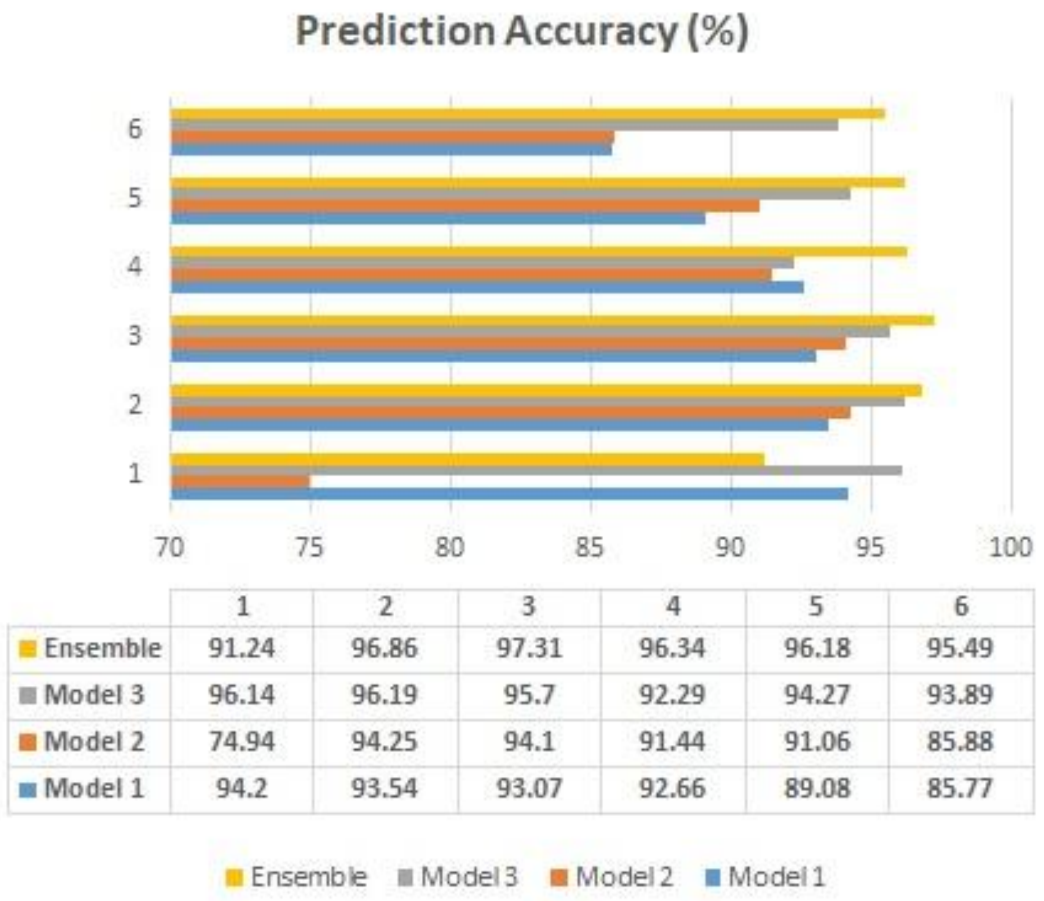


#	Block
1	Convolution Block
-	Square Activation
2	Dense Block
-	Square Activation
3	Dense Block

Table 2. Feedforward blocks

- The ensemble consists of 3 models, each of which has the architecture defined in Table 1.
- After training, the weights and biases are broken down into the 3 blocks as shown in Table 2.
- We add the activation function between each block without affecting the weights and biases.
- We integrate these 3 sets of weight and bias .csv files into CryptoNet's input such that the input can propagate through 3 different models and produce an average of the 3 resulting predictions.

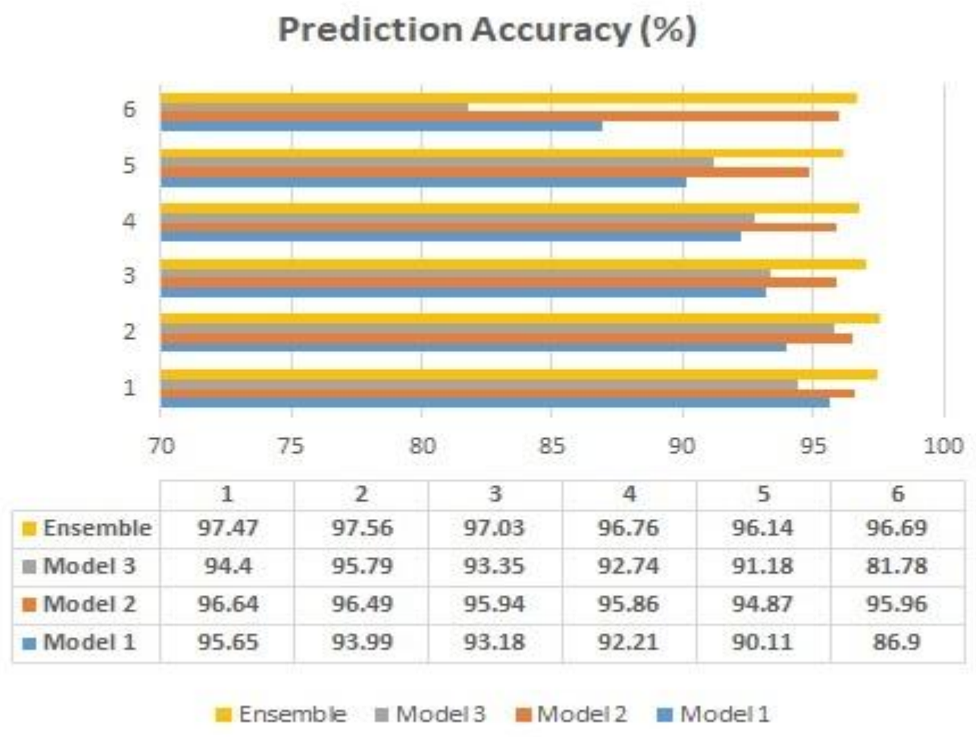
Experiment Results



	1	2	3	4	5	6
🟨 Ensemble	91.24	96.86	97.31	96.34	96.18	95.49
▨ Model 3	96.14	96.19	95.7	92.29	94.27	93.89
🟧 Model 2	74.94	94.25	94.1	91.44	91.06	85.88
🟦 Model 1	94.2	93.54	93.07	92.66	89.08	85.77

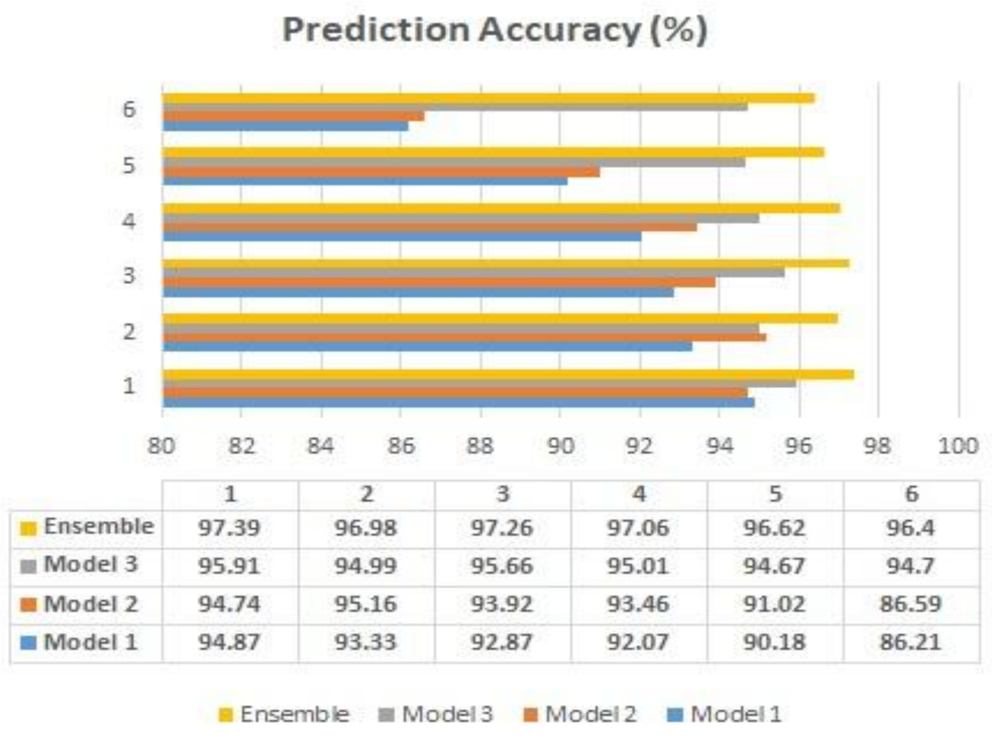
Fig 5. Accuracy w/ regular data split

- Test Accuracy of Encrypted CNN Ensemble compared with Individual Models.
- Using the regular data split method,majority of digit "0", "1", "2" images are grouped to train individual Model 1, most "3", "4", "5" images for Model 2,and majority of "6", "7", "8", "9" images for Model 3.
- For each data split method, we have 6 different settings which vary in the range of percentage for each label.
- To measure the accuracy of our trained models, we use 10k images.



	1	2	3	4	5	6
Ensemble	97.47	97.56	97.03	96.76	96.14	96.69
Model 3	94.4	95.79	93.35	92.74	91.18	81.78
Model 2	96.64	96.49	95.94	95.86	94.87	95.96
Model 1	95.65	93.99	93.18	92.21	90.11	86.9

Fig 6. Accuracy w/ likewise data split



	1	2	3	4	5	6
Ensemble	97.39	96.98	97.26	97.06	96.62	96.4
Model 3	95.91	94.99	95.66	95.01	94.67	94.7
Model 2	94.74	95.16	93.92	93.46	91.02	86.59
Model 1	94.87	93.33	92.87	92.07	90.18	86.21

Fig 7. Accuracy w/ random data split

- We group digits that look similar together to train one individual model.
- The particular partition used was ({"0","6","8"}, {"2","3","5","7"}, {"1","4","9"})

- We randomly partitioned the labels to train individual CNN models.
- The particular partition used was ({"0","6","8"}, {"2","3","5","7"}, {"1","4","9"})

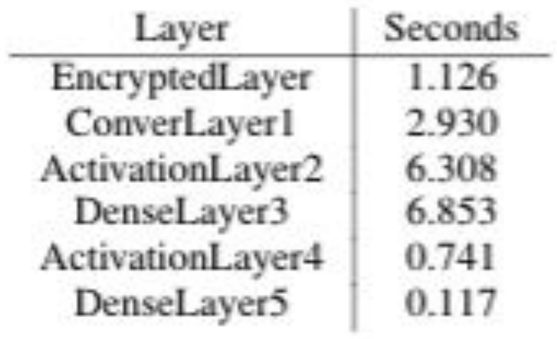
Accuracy Evaluation

- In Figures 4, 5, and 6, we observe that the test accuracy levels across different settings are consistently high, in the range of 96% - 97%.
- Except for one setting in the regular data split method, the ensemble produces a higher test accuracy than individual CNN models.
- This implies that building ensembles has its strengths in achieving lower bias and thus higher test accuracy.
- Our algorithm enables end-users to utilize more powerful deep learning models in a privacy-preserving manner.

Efficiency Evaluation

- As the computing time and resources are an important factor in deep learning, we evaluate the efficiency of our algorithm.
- We measured the time it takes to complete each layer for each batch in the model and averaged our overall runtime per layer per batch
- The most time-consuming layers in the encrypted CNN model are the Activation Layer, the Dense Layer, and the Convolutional Layer

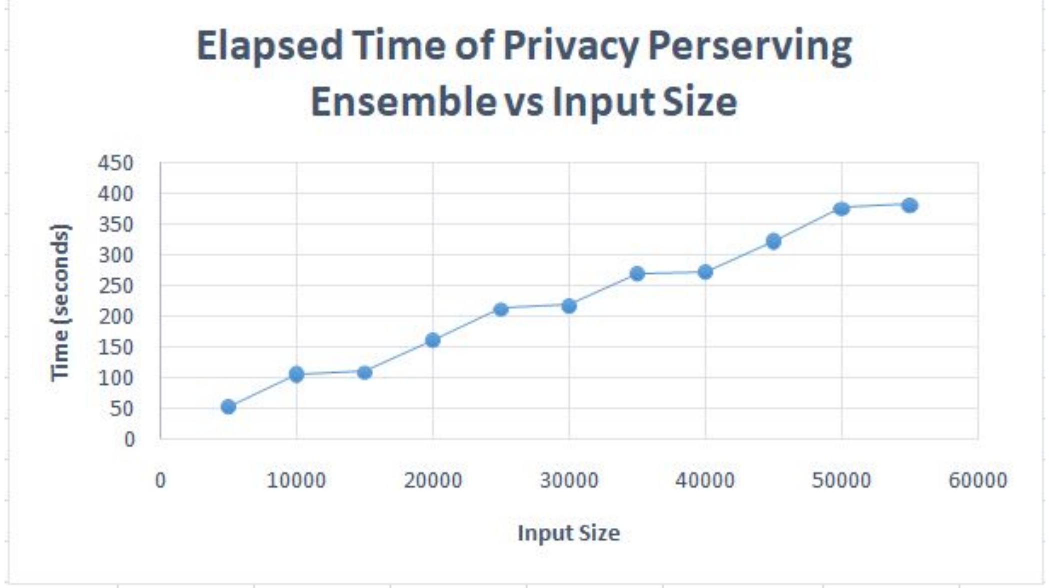
- We determined that total elapsed time increases linearly with regards to the input size.
- Overall, our algorithm can efficiently make inferences for 50,000 images in less than 7 minutes.



Layer	Seconds
EncryptedLayer	1.126
ConverLayer1	2.930
ActivationLayer2	6.308
DenseLayer3	6.853
ActivationLayer4	0.741
DenseLayer5	0.117

Table 3. Average Time elapsed in seconds per layer

Fig 7. Elapsed Time with Different Input Sizes



Conclusion

- This work advances the fully homomorphic encryption-based secure deep learning technique by enabling encrypted CNN ensembles.
- We present a privacy preserving CNN ensemble algorithm that takes encrypted images as input and produces encrypted prediction results based on an average ensemble of multiple pre-trained CNN models.
- Our encrypted CNN ensemble model preserves inference correctness (i.e., same prediction result as using clear text input) and user data privacy.
- We have also verified the accuracy and efficiency of our algorithm.