Homomorphic Encryption For Computer Vision

Rohan Virani
Sanjari Srivastava
Preey Shah

CS231N
Deep Learning for
Computer Vision

Motivation

- Applying machine learning, especially to problems involving medical/financial or other type of sensitive data, requires careful maintenance of data privacy and security.
- At the same time, the Prediction-as-a-Service paradigm is also gaining traction, in which a prediction service manages the cloud infrastructure needed to run a model at scale, and makes it available for online and batch prediction requests.
- Since legal requirements to preserve data privacy may prevent healthcare companies from using cloud-based machine learning solutions, there is a need for neural networks which can be applied directly on encrypted data; without decrypting it on the cloud.
- Recent advances in fully homomorphic encryption make this possible. A fully homomorphic encryption allows an arbitrary sequence of additive and multiplicative operations to be performed on encrypted data directly.

Prior Work

Most relevant: “Crypto-nets - neural networks over encrypted data,” Microsoft 2016 and “Privacy preserving vision via additive homomorphic encryption” Phong et al.

- Prove that it is possible to run neural networks on encrypted data without substantial loss of accuracy but at a cost to inference time on MNIST

Limitations: Failed to test method on more complex datasets and analyze how changes in architecture affect accuracies and inference times

Modifying Neural Nets

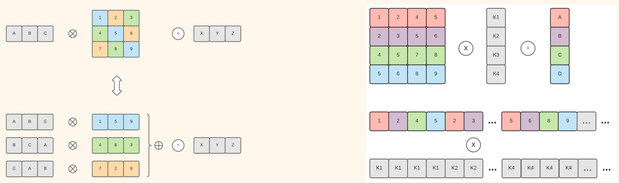


Image to Column method for performing encrypted convolutions

Results for Test Accuracy and Inference Times

We used two CNN architectures for conversion to encrypted space for homomorphic computation. The first architecture is a simple 1 layer convolutional neural network and two connected layers and tested on 5k samples of grayscale

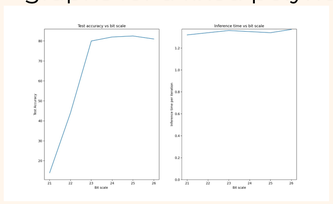
Dataset	Test Accuracy (Unencrypted)	Test Accuracy (Encrypted)	Total Inference Time in Hours	Poly mod degree Coeff Modulus
MNIST	97.14	98.90	1.23	8192 (32,26)
KMNIST	90.65	90.71	1.31	8192 (32,26)
Fashion MNIST	87.40	87.39	1.32	8192 (32,26)

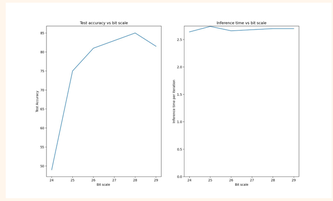
The second architecture is a two layer convolutional neural network with an average pooling layer and 2 fully connected layers

Dataset	Test Accuracy (Unencrypted)	Test Accuracy (Encrypted)	Total Inference Time in Hours	Poly mod degree Coeff Modulus
MNIST	94.72	94.78	4.27	16384 (45,30)
KMNIST	88.46	87.66	4.01	16384 (45,30)
Fashion MNIST	84.42	83.90	4.16	16384 (31,26)

Results for Bit Scale and Modulus

- We also modify the bit scale and polynomial modulus (CKKS encryption parameters) to see how this impacts test accuracies and inference times
- The bottom left graphs alter the bit scale for a fixed polynomial modulus of 8192 and the bottom right graphs for a fixed polynomial modulus of 16384





- Bit scale controls precision of encryption of the fractional part and polynomial modulus increases security guarantees

Experimental Approach

- We use the Tenseal Library built on Microsoft SEAL for writing neural networks via homomorphic additive and multiplicative operations
- We use the image to columns method for achieving convolution - and have to decrypt and re-encrypt the data for each convolutional layer
- We use the Halevi and Shoup Method for writing linear layers (loops over the diagonals of matrices and computes dot products)
- We convert all non-linear functions to their polynomial approximations so they can be computed in encrypted space
- ReLU is approximated by a square function and we used average pooling instead of max pooling
- We use the CKKS encryption scheme which uses approximate arithmetic so the output is slightly different to the input

Discussion and Future Work

- Test accuracies are similar or higher in the one layer conovlutional net because of the encryption scheme used - since CKKS outputs approximate results this adds natural noise to the output which guards against overfitting
- Inference times scale linearly with the number of convolutional and pooling layers due to the lack of GPU support in Microsoft SEAL based libraries
- Increasing the bit scale improves the test accuracies up until a certain point (26) after which performance plateaus while inference times are not impacted
- Increasing the polynomial modulus leads to inference time scaling linearly and thus there is a trade off between security guarantees and test inference times