High Accuracy Approximation of Secure Multiparty Neural Network Training

Daniel Ho, Xin Wang, Wenting Zheng, Joseph E. Gonzalez, Raluca Ada Popa, Ion Stoica
UC Berkeley

1. Introduction

Deep neural network models [8, 12, 16, 18] are widely used in machine learning applications and have become the dominant approach to image classification, natural language processing, and advanced planning and reasoning. To produce high quality models, neural network training requires massive amounts of data. Unfortunately, many organizations do not possess the large labeled datasets required to train sufficiently deep neural networks. However, if multiple organizations can pool their data together and collaboratively train on the combined dataset, they could potentially produce a better model than could otherwise be produced by any of the individual organizations.

While collaboration will likely yield a model that is far more accurate, the process of pooling data introduces serious privacy concerns. Not only does pooling data reveal individual data contributor's sensitive information to every other participant, it also creates a high risk central data repository. Privacy laws in some countries also prohibit or have restrictive policies on data sharing. In the past, people have used anonymization [17, 20] to protect user privacy and to mitigate the risk of a compromised data pool. Unfortunately, previous attacks such as the Netflix attack [15] have also shown that it is possible to deanonymize such real world datasets.

Secure multiparty computation (MPC) [1, 6, 19] is a promising cryptographic technique that allows multiple participants to jointly compute a function without revealing each party's input to any other party. Recent systems [10, 14] utilize MPC to train neural networks, but an important challenge is that neural network's non-linear functions are difficult to be efficiently represented in the secure computation format. Prior work [5, 10, 14] introduced efficient approximations of some non-linear functions using low degree polynomials and piecewise linear approximation. However, these systems evaluate their approximations on small neural networks with very few layers and on easy datasets. For example, SecureML [14] uses a neural network with only two hidden layers and evaluates the accuracy on MNIST [13].

In this paper, we propose new approximations of widely used non-linear neural network functions using Taylor expansion and piecewise linear approximation. We take standard neural networks[8, 18] – convolutional neural networks (CNN) and recurrent neural networks (RNN) – and replace their non-linear functions with our approximations. We evaluate our approximations using challenging image datasets such as CIFAR-10 and CIFAR-100 [11] and large scale English-to-French translation data[2]. Our benchmarks show that these new approximations preserve the original model's accuracy.

2. System design

2.1 System overview

We use the same system setting and cryptographic techniques as SecureML [14], where the model is trained with the help of two *non-colluding* servers. This means that an attacker cannot compromise both servers at the same time. There are multiple parties $P_1 ... P_n$ who jointly compute a model using stochastic gradient descent. Each party splits its data into two random shares using an additive secret sharing scheme and sends each share to a single server. For example, if party P_1 has data d, it will generate two random elements s_1 and s_2 such that $d = s_1 + s_2$. Server 1 will receive s_1, and server 2 will receive s_2.

The gradient descent computation is compiled into a circuit-based representation composed of addition and multiplication gates. Addition requires one local addition, while multiplication requires some communication between the two servers using pre-computed multiplication triplets. SecureML shows that it is possible to further improve efficiency by converting between secret sharing and garbled circuits for comparison and division circuits.

We approximate non-linear functions using either Taylor expansion (i.e., polynomial approximation) or piecewise linear approximation. Polynomial approximations are computed using multiplication gates over finite field elements, while piecewise linear approximation is first converted into garbled circuit representation and converted back once the computation is complete. To improve efficiency, we want to approximate using either low degree polynomials or piecewise approximation with few segments as these methods will reduce the number of required gates.

2.2 Approximations of non-linear functions

We propose approximating the non-linear functions with Taylor expansion and piecewise linear approximation. We focus on approximating e^x and `tanh` shown in Table 1 since e^x is a building block for many functions including Sigmoid, Softmax function, etc which are widely used in neural network as the last layer for classification tasks. We are not

Approx.	Formula
SecureML approx. of e^x	ReLU
Deg. k Taylor approx. of e^x	$\sum_{i=0}^{k} \frac{x^i}{i!}$
4-part piecewise linear approx. of e^x	$max \begin{cases} 0 \\ 0.0077x + 0.155 \\ x+1 \\ 7.3891x - 7.3891 \end{cases}$
3-part piecewise linear approx. of tanh	$\begin{cases} -1 & x \leq -1 \\ x & -1 < x < 1 \\ 1 & 1 \leq x \end{cases}$

Table 1. Approximation formulae

approximating Softmax function directly since tasks we encounter nowadays often have a large number of different categories which makes multi-variable Taylor expansion difficult to use. Combined with division circuits, we can simply compose the approximation of e^x to form approximations of more complicated functions like Softmax. `tanh` is generally used in recurrent neural networks as a building block for LSTM [4] cell and GRU cell[3], etc. In this work, we focus on approximating `tanh` with piecewise linear approximations.

3. Evaluation

3.1 Convolutional Neural Networks

We first evaluate our approaches on convolutional neural networks, one of the most widely used neural network category. We train ResNet-18 and ResNet-32 [8] on CIFAR 10 and CIFAR 100 datasets [11] with gradient descent optimization using momentum 0.9 and batch sizes of 128. The initial learning rate is 0.1 and reduced by 10 every 20000 steps after 40000 steps of the initial learning rate. We use both Taylor expansion and piecewise linear approximation to approximate the Softmax function in the network and Table 2 shows the results obtained on both datasets. SecureML used the same update function as the activation function being approximated rather than computing the partial derivative of the new function; however, changing the gradient calculation in our experiments did not improve accuracy so all experiments are run using the correct gradient calculations.

We can see from the figure that our low degree polynomial approximations obtained comparable and even better results with original models without approximation. During the experiments, we made an interesting observation that *odd* degree Taylor approximations had worse accuracy. Our hypothesis is that the inputs to the approximation function could possibly be negative and greater than 1. In this case, the Taylor approximation for e^x would be negative and thus the value would clip at zero when we calculate the approximated softmax, bringing a significant fraction of errors.

Data	Type	Test Accuracy	
		ResNet18	ResNet32
CIFAR10	Without Approx.	91.2%	92.6%
	SecureML Approx.	28.9%	10.1%
	Taylor Approx. degree 2	**91.0**%	**92.2**%
	Taylor Approx. degree 3	56.0%	47.7%
	Taylor Approx. degree 4	91.6%	92.4%
	Taylor Approx. degree 5	80.4%	82.7%
	Linear Approx. 4-parts	**91.3**%	**91.9**%
	Linear Approx. 5-parts	91.5%	91.5%
	Linear Approx. 6-parts	91.6%	92.0%
	Linear Approx. 7-parts	91.5%	92.4%
CIFAR100	Without Approx.	67.8%	68.3%
	SecureML Approx.	1.1%	0.9%
	Taylor Approx. degree 2	**68.3**%	62.4%
	Taylor Approx. degree 4	66.9%	**67.1**%
	Linear Approx. 4-parts	**68.0**%	**68.3**%
	Linear Approx. 5-parts	67.1%	68.5%
	Linear Approx. 6-parts	68.0%	68.3%
	Linear Approx. 7-parts	66.7%	67.9%

Table 2. Results on CNNs

Since SecureML approximation uses *ReLU* function to approximate e^x and also clips negative values to zero, we hypothesize this explains its poor performances.

3.2 Recurrent Neural Networks

We continue to explore the effectiveness of our approaches on recurrent neural networks which is another popular type of neural networks. We use Sequence-to-Sequence (Seq2seq) model[18] modified from TensorFlow tutorial to speed up training for French translation[2]. We use piecewise linear approximation to approximate the `tanh` activation function and `Softmax` function. For `tanh`, we use 3-part piecewise linear approximation and for `Softmax`, we vary from 4 to 7 linear parts. As we can see from the results in Table 3, we can get very close perplexity to the original Seq2seq model without approximation.

Type	Perplexity
Without Approx.	10.38
Linear Approx. 4-parts	19.11
Linear Approx. 5-parts	17.11
Linear Approx. 6-parts	17.45
Linear Approx. 7-parts	**13.77**

Table 3. Results on RNNs

4. Conclusion and future work

In this paper, we showed efficient approximations of neural network functions that can preserve prediction accuracy. For future work, we would like to implement our approximations in an MPC framework and evaluate its efficiency. There are also other details that need to be considered, such as data precision's impact on prediction accuracy. As prior works [7, 9, 14] have shown, precision should not have a negative impact on accuracy.

References

[1] BEN-OR, M., GOLDWASSER, S., AND WIGDERSON, A. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing* (1988), ACM, pp. 1–10.

[2] BOJAR, O., CHATTERJEE, R., FEDERMANN, C., HADDOW, B., HUCK, M., HOKAMP, C., KOEHN, P., LOGACHEVA, V., MONZ, C., NEGRI, M., ET AL. Findings of the 2015 workshop on statistical machine translation. *EMNLP 2015* (2015), 1.

[3] CHO, K., VAN MERRIËNBOER, B., GULCEHRE, C., BAHDANAU, D., BOUGARES, F., SCHWENK, H., AND BENGIO, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[4] GERS, F. A., SCHMIDHUBER, J., AND CUMMINS, F. Learning to forget: Continual prediction with lstm.

[5] GILAD-BACHRACH, R., DOWLIN, N., LAINE, K., LAUTER, K., NAEHRIG, M., AND WERNSING, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning* (2016), pp. 201–210.

[6] GOLDREICH, O., MICALI, S., AND WIGDERSON, A. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing* (1987), ACM, pp. 218–229.

[7] HAN, S., MAO, H., AND DALLY, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[8] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[9] IANDOLA, F. N., HAN, S., MOSKEWICZ, M. W., ASHRAF, K., DALLY, W. J., AND KEUTZER, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360* (2016).

[10] KONEČNÝ, J., MCMAHAN, H. B., YU, F. X., RICHTÁRIK, P., SURESH, A. T., AND BACON, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[11] KRIZHEVSKY, A., AND HINTON, G. Learning multiple layers of features from tiny images.

[12] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[13] LECUN, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[14] MOHASSEL, P., AND ZHANG, Y. Secureml: A system for scalable privacy-preserving machine learning. *IACR Cryptology ePrint Archive 2017* (2017), 396.

[15] NARAYANAN, A., AND SHMATIKOV, V. How to break anonymity of the netflix prize dataset. *arXiv preprint cs/0610105* (2006).

[16] SILVER, D., HUANG, A., MADDISON, C. J., GUEZ, A., SIFRE, L., VAN DEN DRIESSCHE, G., SCHRITTWIESER, J., ANTONOGLOU, I., PANNEERSHELVAM, V., LANCTOT, M., ET AL. Mastering the game of go with deep neural networks and tree search. *Nature 529*, 7587 (2016), 484–489.

[17] SWEENEY, L. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10*, 05 (2002), 557–570.

[18] VINYALS, O., KAISER, Ł., KOO, T., PETROV, S., SUTSKEVER, I., AND HINTON, G. Grammar as a foreign language. In *Advances in Neural Information Processing Systems* (2015), pp. 2773–2781.

[19] YAO, A. C. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on* (1982), IEEE, pp. 160–164.

[20] ZHOU, B., PEI, J., AND LUK, W. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *ACM Sigkdd Explorations Newsletter 10*, 2 (2008), 12–22.