Classification of Encrypted Word Embeddings using Recurrent Neural Networks

Robert Podschwadt
rpodschwadt1@student.gsu.edu
Georgia State University
Atlanta, Georgia

Daniel Takabi
takabi@gsu.edu
Georgia State University
Atlanta, Georgia

ABSTRACT

Deep learning has made many exciting applications possible and given the popularity of social networks and user generated content everyday there is no shortage of data for these applications. The content generated by the users is written or spoken in natural language which needs to be processed by computers. Recurrent Neural Networks (RNNs) are a popular choice for language processing due to their ability to process sequential data. On the other hand, this data is some of the most privacy sensitive information. Therefore, privacy-preserving methods for natural language processing are crucial. In this paper, we focus on settings where a client has private data and wants to use machine learning as a service (MLaaS) to perform classification on the data without the need to disclose the data to the entity offering the service. We employ homomorphic encryption techniques to achieve this. Homomorphic encryption allows for data being processed without it being decrypted thereby protecting the users privacy. Although homomorphic encryption has been used for privacy-preserving machine learning, most of the work has been focused on image processing and convolutional neural networks (CNNs), but RNNs have not been studied. In this work, we use homomorphic encryption to build privacy-preserving RNNs for natural language processing tasks. We show that RNNs can be run over encrypted data without loss in accuracy compared to a plaintext implementation by evaluating our system on a sentinment classification task on the IMDb movie review dataset.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Natural language processing**; **Neural networks**;

KEYWORDS

privacy preserving machine learning, recurrent neural networks, homomorphic encryption

1 INTRODUCTION

Artificial neural networks have been very successful and popular over the last few years in a variety of domains. CNNs have shown better than human performance in image classification tasks [13, 38] and have also been applied to language processing tasks[16]. RNNs, another type of neural networks, are specifically designed to work with sequences. Unlike other types of networks, RNNs take the output of the previous sequence step into consideration. There are different types of RNN architectures such as Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU) and a simple fully connected variant or Elman Network [15]. In this work we work with Elman Networks and unless specified otherwise will use the term RNN instead of Elman Network. Recurrent architectures are very popular in natural language processing (NLP) due to the sequential nature of language. There are many different sub-fields in NLP. In this work we investigate the task of sentiment classification.

Many companies have built a business around offering MLaaS. In MLaaS the model is hosted in the cloud. The service provider has the infrastructure and know-how to build the models. The client owns the data and sends it to the provider (also called server) for processing.

A concern for the client of MLaaS is the privacy of the data. To process the data the server needs access to the data. This is often unwanted or unacceptable depending on the sensitivity of the data. There are three main techniques for preserving the privacy of the data while still allowing for ML algorithms to work: 1) Secure Multiparty Computation (SMC), 2) Differential Privacy (DP) and 3) Homomorphic Encryption (HE).

In previous work a variety different machine learning algorithm have been adapted for privacy preserving processing such as linear regression [29], linear classifiers [4, 17], decision trees [1, 4] or neural networks [14, 29, 32]. Solutions based on SMC [29, 32] come with a huge communication overhead.

We propose an approach that is based on homomorphic encryption and recurrent neural networks. It does not require interactive communication between client and server like SMC approaches but in the case of longer sequences, we use interactive communication to control the noise introduced by HE. Very little prior work deals with recurrent neural networks. Much of the work is done on CNNs in the image domain [10, 14, 22] and more. [39] perform encrypted speech recognition which is an NLP task but the model used is also a CNN. Badwai et al. [2] research privacy preserving text classification which is the task that we also us in the this paper but the authors do not use an RNN. To the best of our knowledge, there is only one prior paper working with a recurrent architecture. Qian and Lei propose a system [26] that is capable of implementing LSTM networks based on TFHE [7]. Their LSTM model suffers from a small drop in accuracy though when running on encrypted data. Our solution is able to maintain the same accuracy as the plain text model. We present a solution that can process RNNs with arbitrary length input sequences in a privacy preserving manner and introduce a way of using word embeddings with encrypted data. To ensure the privacy of the data we rely on the CKKS [6] crypto scheme. We evaluate our system on a text classification task. The basic idea of our proposed approach is running RNNs on the encrypted data by taking advantage of HE schemes. The server hosts the trained model, the client transmits the encrypted data for

processing and receives an encrypted result. The training of the model is done on plaintext. In this work, we make the following main contributions:

- We propose an approach that combines RNNs, specifically Elman Networks, and homomorphic encryption to perform inference over encrypted data in natural language processing tasks.
- We present an innovative approach to work with word embeddings for encrypted data.
- We perform thorough benchmarking of our system both with respect to run time performance and communication cost. Our results demonstrate that we are able to run RNNs over encrypted data without sacrificing accuracy and with reasonable performance and communication cost.

1.1 Threat Model and Problem Statement

In this paper, we apply privacy preserving machine learning techniques based on HE to RNNs. We focus on a client server setting such as MLaaS in which the client has full control over the data and the server has full control over the model. We assume that the model has been trained on plaintext data and the server offers inference as a service to the clients. The clients want to use the inference service and wish to keep their data private while the server wishes to keep its model private .

Threat Model: We assume that all parties are honest but curios. They will not deviate from the protocol but will attempt to learn any information possible in the process. The server does not share information about the architecture of the model with the client. The client encrypts the data and sends it to the server for processing. If it is possible, the server will process the data and send back the final result in encrypted format. In some cases data will be sent back to the client where it is decrypted, encrypted again to remove the built-up noise and sent back to the server to continue processing. In addition to the privacy of the data we have the goal to achieve accurate predictions. This means the predictions made on encrypted data should be as close as possible to predictions made on plaintext data.

2 BACKGROUND

2.1 Homomorphic Encryption

Homomorphic encryption (HE) schemes are similar to other asymmetric encryption schemes as in they have a public key pk for encrypting (Enc) data and a private or secret key sk for decryption (Dec). Additionally, HE schemes also have a so-called evaluation function, $Eval$. This evaluation function allows the evaluation of a circuit C over encrypted data without the need for decryption. Given a set of plaintexts $\{m_i\}_0^n$ and their encryption $\{c_i\}_0^n = Enc(pk, \{m_i\}_0^n)$ the circuit C can be evaluated as:

$Dec(sk, Eval(pk, C, c_0, \cdots, c_n)) = C(m_0, \cdots, m_n)$.

Most modern HE schemes are based on the ring learning with errors problem (RLWE). Roughly speaking to encrypt a plaintext some noise is added and the decryption process is the removal of that noise. For more details see Brakerski et al. [5] and Cheon et al. [6]. When operations are performed on the ciphertexts the noise grows and when it passes a certain threshold the ciphertext can not be decrypted correctly anymore. Multiplications add much more noise than additions. A way of controlling the noise is to use so-called leveled homomorphic encryption (LHE). LHE allows a for certain number of multiplications based on the parameters chosen for the encryption scheme and evaluating circuits of a known depth. Computation cost can be mitigated in some cases by using single instruction multiple data (SIMD) techniques introduced by Smart and Vercauteren [34].

2.2 Recurrent Neural Networks

In contrast to fully connected or convolutional neural networks, which are feed forward only, recurrent neural networks feed some part of their hidden state back into themselves.

There are many different types of recurrent neural network cells with Long short-term Memory (LSTM) [23] and Gated Recurrent Unit (GRU) [8] being the most popular ones. These cells are more complex than simple RNNs which we are focusing on this paper. While LSTM and GRU lead to better performance we focus on the simpler RNN type due to lower computational complexity.

The RNN used in this paper consists of three main components *input* (x_t), *hidden state* (s_t) and *output* (o) of the network at time step t. The s_t for one neuron is calculated by following formula: $s_t = f(x_t \cdot w + s_{t-1} \cdot v)$ where f is the activation function and $\cdot$ is the vector dot product. Tanh ($\frac{e^{-x}-e^{-x}}{e^{-x}+e^{-x}}$) is the most common activation functions used in RNNs.

2.3 Polynomial Approximation: Theoretical Foundation

One of the major limitations of homomorphic encryption is the limited set of operations that can be performed. CKKS supports addition and multiplication. Division is supported only for plaintext divisors. Basically this allows us to evaluate only polynomials. Tanh, a popular activation function in RNNs, can not be expressed as a polynomial. This means we can not evaluate it over encrypted data. A way to circumvent this is to find a polynomial approximation.

Hesamifard et al. [21] use an approach that is based on Chebyshev polynomials. Given the family of all continuous real valued functions X on a non-empty compact space $C(X)$ and let μ be a finite measure on X. The authors define $f, g \in C(X)$ as $\langle f, g \rangle = \int_X fg d\mu$. To generate Chebyshev polynomials they use $d\mu = \frac{dx}{\sqrt{1-x^2}}$ as the measure on $[-1, 1]$. For better computational performance we want to stick to low degree polynomials.

2.4 NLP with Neural Networks

Recurrent neural networks are widely used for addressing challenges in Natural Language Processing. Recurrent neural network reached state-of-the-art performance for different tasks such as: Speech Recognition, [19] and [18], Generating Image Descriptions, [25] and [36], Machine Translation, [3] and [11], Language Modeling, [28] and [35]. The implementation of an NLP pipeline using RNNs can be broken done into four major parts: 1) Designing the network, 2) Encoding the data, 3) Training the model and 4) Inference of new instances.

In the next section we will look at the individual steps in detail and describe the changes that are necessary for computation in a privacy preserving setting.

3 THE PROPOSED PRIVACY-PRESERVING CLASSIFICATION FOR RECURRENT NEURAL NETWORKS

Looking at the components of the RNN pipeline described in Section 2.4 we determine what changes need to be made to adhere to the constraints of homomorphic encryption.

Network Design. As long as we only use fully connected and recurrent layers the only consideration we need to make are the activation functions that are being used. All other operations inside an RNN can be performed over encrypted data using HE schemes. However, it is not possible to implement common activation functions within current HE schemes. We aim to find the best low degree polynomial approximation to replace the activation functions within the RNN.

Data Encoding. In this paper, we use word embeddings as an encoding scheme for textual data. We describe our approach to handling embeddings in more detail in Section 3.1.

Model Training. In this paper, we assume that the training of the model is performed by the server on plain training data.

Inference. This is the part of the pipeline in our system that is run on encrypted data. At no point during this process is the data decrypted on the server thus ensuring its privacy is protected. During processing by the model, the encrypted data accumulates noise. We describe a way of circumventing the problem of the noise crossing the threshold after which correct decryption is no longer possible in Section 3.2. Once the data has been processed by the entire network, the result of the classification is sent back to the client. The result of the classification is still encrypted and needs to be decrypted by the client.

A variety of activation functions have been proposed as replacements for common activation functions used in NNs. Dowlin et. al [14] use polynomials of degree 2 to substitute the Sigmoid function in CNNs and Shortell and Shokoufandeh [33] use polynomial of degree 3 to approximate the natural logarithm function. Hesamifard et. al [21] use Chebyshev polynomials to approximate activation functions such as ReLU, Sigmoid and Tanh. We will be using the approach of [21] to approximate Tanh which is the most popular activation function in RNNs. The Softmax function can not be performed over encrypted data but since it is typically used as the very last function of neural network, we move it to the client side. The server computes the neural network all the way to the inputs of the Softmax function. The the Softmax function is performed by the client after decryption to obtain the classification results.

3.1 Encrypted word embeddings

Word embeddings are a way to turn words into real valued vectors. The embedding layer basically is a lookup table that maps any word in a dictionary to a real valued vector. The lookup of an embedding for a given word cannot be performed efficiently in HE schemes. We address this problem by moving the embedding layer out of the RNN and to the client where it can be performed in plaintext. After performing the embedding lookup, the client encrypts the embeddings and sends the result to the server. To enhance the the privacy of the model, the model owner can use one of the many pretrained embeddings such as GloVe [30], Elmo [31], Bert [12] or XLNet [37] and share those with the client .

3.2 Noise growth in HE

In an RNN architecture, a sequence is processed by feeding its entries into a fully connected layer which also takes the output of that layer produced for the previous sequence entry. The current output and the previous output are combined into the new output. Due to the noise build-up in HE we need to keep track of the number of operations performed on ciphertexts. To process a sequence of length n with an RNN layer the resulting ciphertext needs to pass the layer n times. That means n dot products and activation functions are applied. It is not always possible to process all of the sequence entries due to the noise that is accumulated. Our approach is to send the encrypted data back to the client where it is decrypted and re-encrypted thereby removing the built up noise.

3.3 Implementation

We use CKKS to protect the privacy of the client data. The server trains a plaintext model and shares the embedding matrix with the client. The activation in the model needs to be compatible with HE. This is achieved by approximating Tanh using the method by Hesamifard et al. [21]. The client performs the embedding process and encrypts the result. The encrypted embeddings are sent to the server where it is processed. When the noise, built up during computation, reaches the limit it the data is sent back to client where it is decrypted, thereby removing all noise, rencrypted and sent back to the server. Once the model is completed processed the server sends the still encrypted resutl back to the client where it can be decrypted. We implement our proposed solution in C++11. We train the model using *Keras* [9] and the homomorphic encryption primitives are provided by HElib [20]. On the plaintext, we tried different activation functions and found out that Tahn and Tanh approximations work best. Other activation functions such as x^2 or the linear function cause the model not to train properly. We find that best replacement for our purposes is: $-0.0016357430301874 8x^3 + 0.249476365628036x$.

4 EXPERIMENTAL RESULTS

The experiments were performed on a Ubuntu 18.04 64bit machine with an AMD Ryzen 5 2600 @ 3.5GHz processor and 32GB of RAM. The IMDb [27] dataset contains 50,000 movie reviews labeled as either positive or negative of which 25,000 are used as training and 25,000 as test data. The tokenization is performed by Keras. We train a model to perform sentiment classification which is classifying a review as either positive or negative. Out of the 25,000 training instances we use 2,000 as validation data for hyperparameter tuning. We use a vocabulary of the top 20,000 words. We pad or truncate the reviews to be 200 words long. Our model consists of an embedding layer that turns words in the reviews into real valued vectors of dimension 128. The embedding matrix is randomly initialized and updated during the training process. The embedding layer is followed by and RNN layer with 128 units. We use the Tanh approximation from Section 3.3 as activation function. The last layer is a fully connected layer with two units and Softmax activation. The training is performed on the plain data using *Keras* and yields 86.47% accuracy on the unseen test data. We achieve the same accuracy on the encrypted data.

We extract the learned weights and run experiments with different batch sizes. In our experiments the noise growth exceeds the

Table 1: Data transferred during encrypted classification

Batchsize	Embeddings	Noisy ciphertext	Refreshed ciphertext	Batch
1	125MB	0.939MB	0.623MB	135MB
4	287MB	2.2MB	1.5MB	312MB
32	1,843MB	14MB	9MB	2,004MB
64	3,548MB	27MB	18MB	3,863MB
128	7,065MB	54MB	35MB	7,869MB
256	14,336MB	106MB	70MB	15,568MB

workable threshold after 27 timesteps. This means we need to add communication between client and server seven times to refresh the noise in order to classify the IMDb sequences of length 200.

The amount of data that needs to be transmitted depends on the batch size. The encrypted embeddings are larger than the plaintext data by a factor of 1,280. See Table 1 for different batch sizes. The *Embeddings* column is the amount of data that is initially transferred from the client to server. *Noisy ciphertext* gives the size of the data the server sends to the client to be refreshed and *Refreshed ciphertext* is the reencrypted answer. These are the values for only one refresh operation. The *Batch* column is the total amount of data transferred between client and server during classification of one batch which requires seven refresh rounds.

The amount of data that needs to be transmitted initially makes up the largest portion of the transfer. To run our network seven noise removal communications are required. At a batch size of 256 the server sends 106MB to the client and the client responds with 70MB. One round of noise removal therefore requires 176 MB to be transferred. All seven rounds take 1,232MB. Which is less than 10% of the initial transfer. The increase in size of the ciphertexts is nearly linear. Smaller ciphertexts sizes carry more overhead per instance than larger ones.

Table 2 lists the execution time for different batch sizes. The times are given for encrypted, plain data and for the actual time it takes to processes the batch as well as the resulting time per instance. The noise removal is not performed by the client though. It is simulated on the server. The measurements also do not include the encryption and transfer of the embeddings. We can see that increasing the batch size leads to lower per instance classification time. The effect is lost when increasing the batch size from 128 to 256. On the plain data we still can see improvement after that point. To get an accurate comparison the plain text measurements are performed on the same implementation as the encrypted experiments. It looks like the growth in execution time for the encrypted values is exponential while the plain version appears to be logarithmic. Our implementation performs best on encrypted data with a batch size of 128 and worst with a batch size of one if we look at the time per sample. The overhead is smallest though for one instance per batch. Here the encrypted version is 40 times slower than the plain version. For our optimal batch size of 128 the encrypted version is 92 times slower. This is due to the different growth rates of execution time for the encrypted and plain data.

Table 2: Run times of inference on IMDb test set.

Batch Size	Encrypted (sample/batch)	Plain (sample/batch)
1	70.6s / 70.6s	1.83s / 1.8s
4	20.2s / 80.7s	0.496s / 1.99s
32	5.8s / 184.6s	0.072s / 2.30s
64	4.3s / 272.7s	0.055s / 3.52s
128	4.2s / 547.6s	0.046s / 5.89s
256	6.5s / 1658.7s	0.039s / 9.96s

5 RELATED WORK

Badwai et al. [2] presented PrivFT a system for privacy preserving text classification built on Facebooks *fasttext* [24] (Joulin et al. [24]). The main difference to our work is that we use a recurrent architecture. In PrivFT the embedding operation is also not outsourced to the client. The client needs to one-hot encode each word, encrypt it and send it to server where the embedding operation is performed as a matrix multiplication. The message size is similar. The inference time for single instance on the IMDb is higher in our scenario but using larger batch sizes allows us to get a lower per instance time. In contrast to our work, PrivFT features schemes for training on encrypted data and a *CKKS* implementation with GPU acceleration. Lou and Jiang created SHE [26] a privacy preserving neural network framework based on TFHE. It offers support for LSTM cells. The authors replace the computationally expensive and high noise introducing matrix operations normally required by LSTMs with much cheaper shift operations. Zhang et al. [39] perform a different NLP task namely encrypted speech recognition based on a CNN. The last step of the network that matches the output to actual text is performed on the client side.

6 CONCLUSION

In this paper, we present an approach that allows the use of recurrent neural networks on homomorphically encrypted data based on the CKKS scheme. We present a solution to perform NLP tasks over encrypted data using recurrent neural networks, in our case sentiment analysis on the IMDb dataset. We are able to achieve this with no loss in accuracy compared to the plaintext model. This is made possible by introducing communication between client and server to refresh the noise. We trade network traffic for the ability efficiently use word embeddings. Our future work aims at investigating other recurrent architectures such as LSTM and GRU.

REFERENCES

[1] Louis J. M. Aslett, Pedro M. Esperança, and Chris C. Holmes. 2015. Encrypted statistical machine learning: new privacy preserving methods. *CoRR* (2015).

[2] Ahmad Al Badawi, Luong Hoang, Chan Fook Mun, Kim Laine, and Khin Mi Mi Aung. 2019. PrivFT: Private and Fast Text Classification with Homomorphic Encryption. *arXiv preprint arXiv:1908.06972* (2019).

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1409.0473

[4] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. 2015. Machine Learning Classification over Encrypted Data. In *22nd Annual Network and Distributed System Security Symposium, NDSS, San Diego, California, USA*.

[5] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2012. (Leveled) Fully Homomorphic Encryption Without Bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS '12)*. ACM, New York, NY, USA, 309–325.

[6] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic Encryption for Arithmetic of Approximate Numbers. In *Advances in Cryptology – ASIACRYPT 2017*, Tsuyoshi Takagi and Thomas Peyrin (Eds.). Springer International Publishing, Cham, 409–437.

[7] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. August 2016. TFHE: Fast Fully Homomorphic Encryption Library. https://tfhe.github.io/tfhe/.

[8] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1724–1734. https://doi.org/10.3115/v1/D14-1179

[9] François Chollet et al. 2017. Keras. https://github.com/fchollet/keras.

[10] Edward Chou, Josh Beal, Daniel Levy, Serena Yeung, Albert Haque, and Li Fei-Fei. 2018. Faster CryptoNets: Leveraging sparsity for real-world encrypted inference. *arXiv preprint arXiv:1811.09953* (2018).

[11] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[13] Terrance Devries and Graham W. Taylor. 2017. Improved Regularization of Convolutional Neural Networks with Cutout. *CoRR* abs/1708.04552 (2017). arXiv:1708.04552 http://arxiv.org/abs/1708.04552

[14] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter Michael Naehrig, and John Wernsing. 2016. *CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy*. Technical Report MSR-TR-2016-3.

[15] Jeffrey L. Elman. 1990. Finding structure in time. *COGNITIVE SCIENCE* 14, 2 (1990), 179–211.

[16] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1243–1252.

[17] Thore Graepel, Kristin Lauter, and Michael Naehrig. 2013. ML Confidential: Machine Learning on Encrypted Data. In *Proceedings of the 15th International Conference on Information Security and Cryptology (ICISC'12)*. Springer-Verlag.

[18] Alex Graves and Navdeep Jaitly. 2014. Towards End-to-end Speech Recognition with Recurrent Neural Networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14)*. JMLR.org, II–1764–II–1772. http://dl.acm.org/citation.cfm?id=3044805.3045089

[19] A. Graves, A. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 6645–6649. https://doi.org/10.1109/ICASSP.2013.6638947

[20] Shai Halevi and Victor Shoup. 2014. Algorithms in HElib. In *Advances in Cryptology - CRYPTO - 34th Annual Cryptology Conference, CA, USA, Proceedings*.

[21] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. 2016. CryptoDL: Towards Deep Learning over Encrypted Data. In *Annual Computer Security Applications Conference (ACSAC)*.

[22] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. 2019. Deep Neural Networks Classification over Encrypted Data. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*. ACM, 97–108.

[23] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 arXiv:https://doi.org/10.1162/neco.1997.9.8.1735

[24] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, 427–431. https://www.aclweb.org/anthology/E17-2068

[25] Andrej Karpathy and Li Fei-Fei. 2015. Deep Visual-Semantic Alignments for Generating Image Descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[26] Qian Lou and Lei Jiang. 2019. SHE: A Fast and Accurate Deep Neural Network for Encrypted Data. In *Advances in Neural Information Processing Systems*. 10035–10043.

[27] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, 142–150. http://www.aclweb.org/anthology/P11-1015

[28] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

[29] P. Mohassel and Y. Zhang. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *IEEE Symposium on Security and Privacy (SP)*.

[30] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[31] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

[32] M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. 2018. Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications. *CoRR* abs/1801.03239 (2018). arXiv:1801.03239 http://arxiv.org/abs/1801.03239

[33] Thomas Shortell and Ali Shokoufandeh. 2015. Secure Signal Processing Using Fully Homomorphic Encryption. In *Advanced Concepts for Intelligent Vision Systems - 16th International Conference, ACIVS, Italy, Proceedings*.

[34] N. P. Smart and F. Vercauteren. 2014. Fully homomorphic SIMD operations. *Designs, Codes and Cryptography* 71, 1 (01 Apr 2014), 57–81. https://doi.org/10.1007/s10623-012-9720-4

[35] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM Neural Networks for Language Modeling. In *INTERSPEECH*.

[36] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Francis Bach and David Blei (Eds.), Vol. 37. PMLR, Lille, France, 2048–2057. http://proceedings.mlr.press/v37/xuc15.html

[37] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237* (2019).

[38] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide Residual Networks. *CoRR* abs/1605.07146 (2016). arXiv:1605.07146 http://arxiv.org/abs/1605.07146

[39] S. Zhang, Y. Gong, and D. Yu. 2019. Encrypted Speech Recognition Using Deep Polynomial Networks. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5691–5695. https://doi.org/10.1109/ICASSP.2019.8683721