Private Text Classification with Convolutional Neural Networks

Samuel Adams[1], **David Melanson**[1], **Martine De Cock**[1,2]
[1] School of Engineering and Technology
University of Washington Tacoma
{sadams,mence40,mdecock}@uw.edu
[2] Dept. of Applied Math., Computer Science and Statistics
Ghent University
martine.decock@ugent.be

Abstract

Text classifiers are regularly applied to personal texts, leaving users of these classifiers vulnerable to privacy breaches. We propose a solution for privacy-preserving text classification that is based on Convolutional Neural Networks (CNNs) and Secure Multiparty Computation (MPC). Our method enables the inference of a class label for a personal text in such a way that (1) the owner of the personal text does not have to disclose their text to anyone in an unencrypted manner, and (2) the owner of the text classifier does not have to reveal the trained model parameters to the text owner or to anyone else. To demonstrate the feasibility of our protocol for practical private text classification, we implemented it in the PyTorch-based MPC framework CrypTen, using a well-known additive secret sharing scheme in the honest-but-curious setting. We test the runtime of our privacy-preserving text classifier, which is fast enough to be used in practice.

1 Introduction

Text classification is inherently a two-party computation problem between the owner of a text classifier and the owner of a personal text. Text classifiers are used for a wide variety of purposes, such as detection of spam and misinformation, sentiment analysis, tailored advertising, surveillance, etc. In these applications, the text classifier is often owned by a company or organization who wants to keep their model private because it offers a competitive advantage and/or it was trained on a proprietary dataset. Deep learning models in particular are powerful enough to memorize specific examples from the training data (Carlini et al., 2019), hence disclosing a trained model can leak very specific information about training data. Furthermore, in applications such as spam or misinformation detection, disclosing the model would help adversaries to develop strategies for evading detection.

The common practice nowadays is therefore for the owner of the personal text to disclose their text to the company or application developer. This in turn also raises serious privacy concerns, stemming from misuse of the data by the company for purposes beyond the originally professed scope. Furthermore, while most companies make reasonable efforts to keep data private, the data itself is a valuable asset that is routinely sold when companies undergo bankruptcy and/or it can become subject of accidental or intentional public exposure (Canny, 2002; Solon, 2018; Thompson and Warzel, 2019). Data breaches and intentional misuse of data have given rise to new laws and regulations, such as the European General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA), and, orthogonal to this, the use of privacy-enhancing technologies (PETs) for the development of algorithms that do not leak personal information about their inputs, thereby protecting the privacy of all the users involved.

In this paper we propose such an algorithm for private text classification that, as illustrated in Fig. 1, allows the owner of a personal text (*Bob*) to infer a label using *Alice*'s text classifier, without requiring Bob to disclose anything about his text in an unencrypted manner to Alice, and without requiring Alice to show her trained model parameters to anyone. To this end, we use Secure Multiparty Computation (MPC) (Cramer et al., 2015), an umbrella term for cryptographic approaches that allow two or more parties to jointly compute a specified output (the class label) from their private information (the classifier and the personal text) in a distributed fashion, without revealing the private information to each other.

Initial solutions for private classification of unstructured text with MPC have been proposed for logistic regression models and tree based models (Reich et al., 2019), and for Naive Bayes classifiers (Resende et al., 2021). All of this existing work relies on an MPC subprotocol for private feature extraction, in which a boolean word occurrence

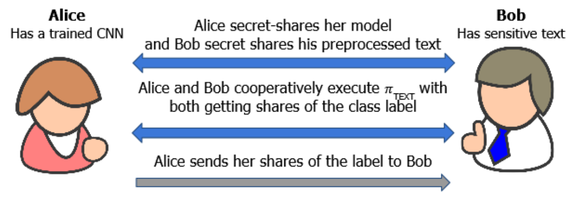
Alice
Has a trained CNN
Alice secret-shares her model
and Bob secret shares his preprocessed text
Bob
Has sensitive text
Alice and Bob cooperatively execute π_{TEXT} with
both getting shares of the class label
Alice sends her shares of the label to Bob

Figure 1: Private text classification process

Quantity of reviews by star rating				
1	2	3	4	5
23,783	6,890	8,308	46,693	46,410
negative			positive	

Table 1: Distribution of reviews by star rating

vector is created that indicates which words from Alice's predefined vocabulary occur in Bob's text. In contrast with this, and in line with the fact that deep learning is current state-of-the-art for many tasks in natural language processing (NLP), we propose to perform MPC-based private text classification with a Convolutional Neural Network (CNN) that has been trained to extract relevant features automatically. To the best of our knowledge this has not been explored yet in the literature.

The MPC protocols for text classification with CNNs that we use in this paper, are very similar to existing MPC protocols for image classification with CNNs (Agrawal et al., 2019; Dalskov et al., 2020; Juvekar et al., 2018; Kumar et al., 2020; Mishra et al., 2020). The main distinguishing aspect is that image classification is based on 2-dimensional (2D) CNNs, while for text classification it is common to use 1-dimensional (1D) CNNs in which the filters move in only one direction. Existing MPC protocols for inference with 2D CNNs can be straightforwardly adjusted to 1D CNNs. Our main contribution is therefore in designing the first application of MPC protocols for CNN-based text classification, and in particular in addressing a question that has remained open in the literature thus far, namely to what extent such provably secure protocols can enable *accurate* and *fast* text classification.

Among deep learning architectures, our choice for CNNs is deliberate. CNNs have been successfully applied for text classification (Kim, 2014) and offer the important advantage of being computationally less intensive than Long Short-Term Memory (LSTM) networks or other state-of-the-art architectures. As such, CNNs are an excellent "MPC-friendly" starting point to explore deep learning based private text classification. In Sec. 2 we describe a multi-channel CNN architecture that we designed for sentiment analysis of product reviews. The trained CNN F is owned by Alice in Fig. 1.

In Sec. 3 we describe our MPC protocol π_{TEXT} for private text classification of Bob's text with Alice's CNN F. As the first step in this process, Bob prepares his text using preprocessing steps that are publicly known and do not depend on Alice's model F nor on the data that Alice used to train F in any way. In our prototype, the preprocessing consists of converting the text using a publicly available sentence transformer (Reimers and Gurevych, 2019). Below we refer to both Alice's CNN model parameters and Bob's preprocessed text simply as "data". At the start of π_{TEXT}, Alice and Bob send each other encrypted shares of their data. Subsequently both parties engage in MPC computations and exchange intermediate encrypted results, without learning anything about the values of the data. At the end of π_{TEXT}, Alice and Bob each have "shares" of the inferred class label. The true class label is revealed only when these shares are combined, e.g. when Alice sends her shares to Bob.

In Sec. 4 we present results obtained with an implementation of π_{TEXT} in CrypTen (Knott et al., 2020), a recently proposed MPC framework built upon PyTorch. CrypTen realizes MPC through a well-known additive secret sharing scheme (see Sec. 3) that guards against honest-but-curious adversaries. A party corrupted by such an adversary still follows the instructions of the MPC protocol but attempts to learn information about the data from the intermediate values and communications between the parties. A correctly designed MPC protocol (as π_{TEXT}) prevents such attacks from being successful. To implement π_{TEXT}, we adapted and extended existing functionality for private image classification in CrypTen to text classification with 1-dimensional (1D) convolutional layers and 1D max pooling. Our results in Tab. 2 demonstrate the practicality of π_{TEXT} as the runtimes are low enough to be used in practice.

2 Text Classifier

Dataset. We trained Alice's model on the Software portion of the Amazon Customer Reviews Dataset.[1]

[1] https://s3.amazonaws.com/amazon-reviews-pds/readme.html

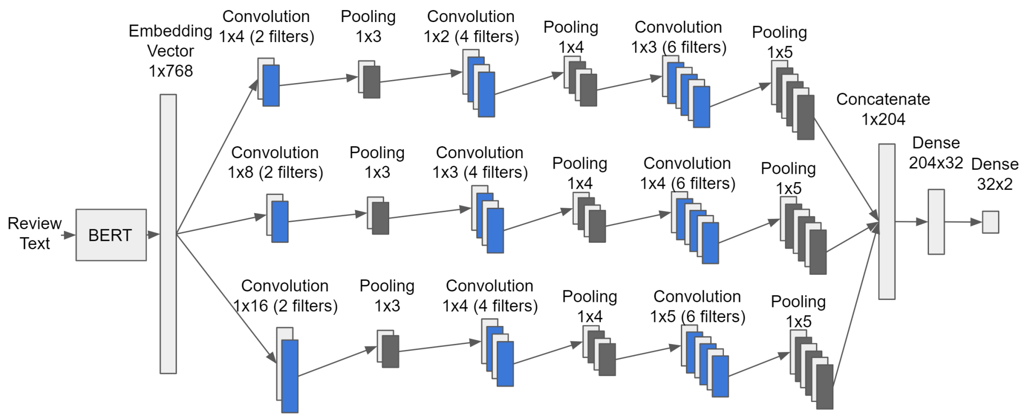
Review Text
BERT
Embedding Vector 1x768
Convolution 1x4 (2 filters)
Pooling 1x3
Convolution 1x2 (4 filters)
Pooling 1x4
Convolution 1x3 (6 filters)
Pooling 1x5
Convolution 1x8 (2 filters)
Pooling 1x3
Convolution 1x3 (4 filters)
Pooling 1x4
Convolution 1x4 (6 filters)
Pooling 1x5
Convolution 1x16 (2 filters)
Pooling 1x3
Convolution 1x4 (4 filters)
Pooling 1x4
Convolution 1x5 (6 filters)
Pooling 1x5
Concatenate 1x204
Dense 204x32
Dense 32x2

Figure 2: The structure of Alice's CNN

The data consists of product reviews for software offered on Amazon's storefront. There are 132,084 reviews total, each with an associated star rating ranging from one to five (see Tab. 1). The average length of a review is 63.6 words. We split the data into 80% for training and 20% for validation, using a stratified split to ensure the distribution of the star ratings is identical in both sets. As described below, we trained a binary classifier for sentiment analysis over this dataset, treating reviews with star ratings ≥ 4 as positive instances, and the rest as negative. We note that, with reproducibility of results in mind, we used a publicly available dataset. During deployment of our private text classification solution, it can be replaced by proprietary text data.

Model Architecture. The text classifier consists of two parts: a public sentence transformer and the private CNN belonging to Alice. The sentence transformer we employ is `stsb-distilbert-base` (Reimers and Gurevych, 2019), a model fine-tuned to produce sentence vectors. This takes as input the raw text and outputs a 768 dimensional embedding vector. As this transformer is public, Bob can also use the sentence transformer to prepare his data for classification by Alice's model. Alice's model consists of three parallel series of convolution and pooling layers, followed by fully-connected layers, ending in a sigmoid activated classification layer (see Fig. 2). This model is then trained with a learning rate of 0.002 with a batch size of 50, and loss is calculated using cross entropy. A dropout of 0.25 is added to the concatenation layer. The trained model obtains 85% accuracy on the validation data.

3 Private Text Classification

Additive Secret Sharing MPC Scheme. Protocols for Secure Multiparty Computation (MPC) enable a set of parties to jointly compute the output of a function over the private inputs of each party, without requiring any of the parties to disclose their own private inputs (Evans et al., 2018). MPC is concerned with the protocol execution coming under attack by an adversary which may corrupt one or more of the parties. In this paper we assume a set-up with two parties (Alice and Bob), one of which may be corrupted by an honest-but-curious adversary. "Honest-but-curious" means that a corrupted party still follows the instructions of the protocol, but the adversary attempts to learn private information from the internal state of the corrupted party and the messages that it receives. MPC protocols that are secure against honest-but-curious adversaries prevent such leakage of information.

In the additive secret sharing MPC scheme that we use, all computations are done on integers modulo q, i.e., in a ring $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$. To map real numbers, such as the trained CNN model parameters, into this ring, we use a fixed point representation with 16 bits of precision for the mantissa. A value x is secret-shared over $\mathbb{Z}_q$ between parties Alice and Bob by picking $x_A, x_B \in \mathbb{Z}_q$ uniformly at random subject to the constraint that $x = x_A + x_B \mod q$, and then revealing x_A to Alice and x_B to Bob. We denote this secret sharing by $[\![x]\!]$, which can be thought of as a shorthand for (x_A, x_B).

Secret-sharing based MPC works by first having the parties split their respective data in secret shares and send some of these shares to each other. To any individual party, their shares of x look like random noise. Only when all of the shares are combined together is the true value of x revealed.

When Alice and Bob have secret-shared numbers $[\![x]\!]$ and $[\![y]\!]$, they can straightforwardly compute $[\![x+y]\!]$ by adding their own shares. To compute $[\![x \cdot y]\!]$, Alice and Bob run a secure multiplication protocol π_{mul} that requires communication between the parties, using a well-known technique with Beaver Triples (Beaver, 1997). These triples can be generated by a trusted initializer (TI) that distributes correlated randomness to Alice and Bob and otherwise does not participate in the computations at all. This is the technique used in CrypTen (Knott et al., 2020) and adopted in the experiments in Sec. 4. Such a TI can be thought of as a third party, next to Alice and Bob, making the setting used in our experiments effectively a three-party configuration. In case a TI is not available or desirable, the required triples can be pre-computed by Alice and Bob in an online phase using an MPC protocol to emulate the TI (Damgård et al., 2012); this method is currently not yet supported in CrypTen.

Building on the cryptographic primitives for addition and multiplication, MPC protocols for other operations have been developed in the literature, many of which we use in turn to build our protocol π_{TEXT}. Of note is the secure comparison protocol π_{comp}, which works by computing the difference of two secret-shared numbers $[\![x]\!]$ and $[\![y]\!]$ and extracting the most significant bit, effectively returning $[\![1]\!]$ if $x < y$, and $[\![0]\!]$ otherwise (more details in (Knott et al., 2020)).

MPC Protocol π_{TEXT}. At the start of the protocol for private text classification, Alice has a CNN model with trained model parameters F, and Bob has an embedding vector D of his text obtained after preprocessing.

- Bob secret-shares vector D with Alice
- Alice secret-shares CNN parameters F with Bob
- Alice and Bob jointly perform computations on the secret sharings $[\![F]\!]$ and $[\![D]\!]$, following the computational graph of $[\![F]\!]$ on $[\![D]\!]$. To this end, for each operation on each layer of the CNN, Alice and Bob execute the appropriate MPC subprotocols, which include:
  - π_{Conv1D}: This operation reduces to the multiplication of secret-shared filter weights with secret-shared input values, and is thus trivially implemented with π_{mul}.
  - $\pi_{MaxPool1D}$: The MaxPool operation is performed through pairwise comparisons with π_{comp} of secret-shared values in a tournament style to determine secret shares of the maximum value for each window.
  - π_{ReLU}: Alice and Bob compute the ReLU activation function of a secret-shared number $[\![x]\!]$ by computing $\pi_{mult}\left([\![x]\!], \pi_{comp}\left([\![x]\!], 0\right)\right)$.
  - π_{Dense}: Application of dense layers reduces to multiplication of matrices with secret-shared numbers, which is an extension of π_{mul}.
  - $\pi_{Sigmoid}$: Alice and Bob compute the sigmoid activation function of a secret-shared number $[\![x]\!]$ using the approximation described by (Knott et al., 2020).
- Alice and Bob execute π_{comp} on the secret-shared output of the CNN to obtain secret shares of the class label.

4 Results

We implemented π_{TEXT} in CrypTen, taking advantage of CrypTen's architecture, which allows to overwrite Torch functions with MPC protocols. All subprotocols needed for π_{TEXT} already existed in the branch crypten-v0.1, barring the π_{Conv1D} and $\pi_{MaxPool1D}$ protocols. We ported π_{Conv1D} from the master branch of CrypTen, and we constructed $\pi_{MaxPool1D}$ ourselves.

In our tests, using $\pi_{Sigmoid}$ to approximate the Sigmoid function made no statistically significant impact on the accuracies. On a test set containing 20,417 instances, using $\pi_{Sigmoid}$ resulted in 17,351 correct classifications, whereas its plaintext counterpart provided 17,350 correct classifications, i.e. an accuracy of 85% across the line. In other words, our use of MPC does not cause accuracy loss. The code we used to run our tests is located on our fork of the CrypTen repository,[2] on the branch "crypten-v0.1".

For our tests, we set up three VMs on Azure, namely one for Alice, one for Bob, and one for the TI. The VMs are Standard L16s_v2, with 16 vCPUs, 128 GiB memory, and an expected network bandwidth of 6400 Mbps. In our experiments, despite having access to large amounts of memory, the parties never used more than 5 GiB at a time.

The TI never sees Alice's or Bob's data. Since the TI's activities are independent of the data that is used as input for the MPC protocols, MPC protocols can be separated in an off-line phase – during which the TI generates correlated randomness and distributes it to Alice and Bob – and an online

[2]https://github.com/SamuelDAdams/CrypTen

L16s: Private Text Classifier Runtimes		
# Instances	Sequential (sec)	Batched (sec)
200	151	21
400	296	44
600	445	65
800	596	88
1000	734	110

Table 2: Runtime results (in seconds) to privately classify "#Instances" using π_{TEXT} on Standard L16s_v2 VMs. "Sequential" denotes the protocol is run one instance at a time, and "Batched" shows the runtime to classify all instances in a single batch. All results are an average over 5 runs.

phase – during which Alice and Bob execute the MPC protocols. In CrypTen this separation is not made, hence the results in Tab. 2 include both the offline and the online phase.

Tab. 2 shows the runtime results of π_{TEXT} when privately classifying various amounts of instances (texts). On average, it takes roughly 0.74 seconds to classify a single instance using the sequential method, and 0.11 seconds using the batching method. Batching the classification task outperforms classifying data sequentially by a wide margin. This is because when batching, the parties can also batch communication rounds during the protocol execution, reducing the communication complexity. The time it takes to classify text is reasonably low, and would lend itself to real life applications, showing that MPC-based private text classification with deep learning models is feasible in practice.

5 Conclusion and Future Work

We have presented and evaluated the first application of MPC-based privacy-preserving text classification with CNNs. Our solution involves model owner Alice, who has a multi-channel CNN for text classification, and text owner Bob who transforms his text into an embedding vector using a publicly available BERT model. Next Alice and Bob secret share their inputs and run an MPC protocol to label Bob's text with Alice's model in a provably privacy-preserving manner. Our protocol takes ~ 0.74 sec to classify a text when run in sequential mode, and ~ 0.11 sec when run in batch mode on Standard L16s_v2 Azure virtual machines with an expected bandwidth of 6400 Mbps. These runtimes are independent of the length of the text, as the embedding vector has a fixed length.

Our work serves as a baseline for MPC-based text classification with deep learning that can be improved upon in many ways. From an NLP point of view, we have assumed that Bob can preprocess his text based on public knowledge, in particular with a sentence transformer model that is in no way dependent on Alice's training data or model parameters. While many text preprocessing algorithms are standard and publicly available to all parties (e.g. for stemming, tokenization, etc.), others may involve proprietary information. Pre-trained versions of language representation models that are publicly available can for instance be fine-tuned by model builders (such as Alice) on task specific data, leading to proprietary word or sentence embedding models that the model owners would not want to disclose. The development of effective and efficient MPC protocols for converting raw text into embedding vectors with state-of-the-art transformer architectures is an open problem.

From a security perspective, there exist a variety of MPC schemes beyond the one we considered in this paper, designed for different numbers of parties and offering various level of security that correspond to different threat models and come with different computational costs. Regarding threat models, besides honest-but-curious adversaries, there exist MPC schemes that protect against malicious adversaries that corrupt parties to deviate from the protocol instructions. Regarding the *number* of parties, some of the most efficient MPC schemes have been developed for three computing servers, out of which at most one is corrupted (i.e. honest majority) by an honest-but-curious or a malicious adversary. While text classification is inherently a (dishonest majority) two-party computation problem between the model owner Alice and the text owner Bob, MLaaS (machine learning as a service) set-ups in which Alice and Bob secret share with, and outsource the computations to, three servers in the cloud are growing in popularity in the privacy-preserving ML literature because of their efficiency (Dalskov et al., 2020; Kumar et al., 2020; Riazi et al., 2018; Wagh et al., 2019; Patra and Suresh, 2020). The use of these different MPC schemes for privacy-preserving text classification is an interesting direction for future work.

Acknowledgements. The authors thank Microsoft for the generous donation of cloud computing credits through the UW Azure Cloud Computing Credits for Research program.

References

Nitin Agrawal, Ali Shahin Shamsabadi, Matt J Kusner, and Adrià Gascón. 2019. QUOTIENT: two-party secure neural network training and prediction. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1231–1247.

Donald Beaver. 1997. Commodity-based cryptography. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 446–455.

John Canny. 2002. Collaborative filtering with privacy. In *Proceedings of 2002 IEEE Symposium on Security and Privacy*, pages 45–57.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium*, pages 267–284.

Ronald Cramer, Ivan Damgard, and Jesper Nielsen. 2015. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press.

Anders Dalskov, Daniel Escudero, and Marcel Keller. 2020. Secure evaluation of quantized neural networks. *Proceedings on Privacy Enhancing Technologies*, 2020(4):355–375.

Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. 2012. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, pages 643–662. Springer.

David Evans, Vladimir Kolesnikov, and Mike Rosulek. 2018. A pragmatic introduction to secure multiparty computation. *Foundations and Trends in Privacy and Security*, 2(2-3):70–246.

Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium*, pages 1651–1669.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. 2020. CrypTen: Secure multi-party computation meets machine learning. In *NeurIPS Workshop on Privacy-Preserving Machine Learning*.

Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. 2020. CrypTFlow: Secure TensorFlow inference. In *41st IEEE Symposium on Security and Privacy*, pages 336–353.

Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. 2020. Delphi: A cryptographic inference service for neural networks. In *29th USENIX Security Symposium*, pages 2505–2522.

Arpita Patra and Ajith Suresh. 2020. BLAZE: Blazing fast privacy-preserving machine learning. *arXiv preprint arXiv:2005.09042*.

Devin Reich, Ariel Todoki, Rafael Dowsley, Martine De Cock, and Anderson Nascimento. 2019. Privacy-preserving classification of personal text messages with secure multi-party computation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3752–3764.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3973–3983.

Amanda Resende, Davis Railsback, Rafael Dowsley, Anderson C. A. Nascimento, and Diego F. Aranha. 2021. Fast privacy-preserving text classification based on secure multiparty computation. Cryptology ePrint Archive, Report 2021/069.

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. In *Asia Conference on Computer and Communications Security*, pages 707–721.

Olivia Solon. 2018. Facebook says Cambridge Analytica may have gained 37M more users' data. *The Guardian*, Apr 4.

Stuart A. Thompson and Charlie Warzel. 2019. Twelve million phones, one dataset, zero privacy. *The New York Times*, Dec 19.

Sameer Wagh, Divya Gupta, and Nishanth Chandran. 2019. SecureNN: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 2019(3):26–49.