A SecureNN Models

```
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(10, activation='softmax')
```

Figure 1: Network A used by Mohassel and Zhang [2017]

```
tf.keras.layers.Conv2D(16, 5, 1, 'same', activation='relu'),
tf.keras.layers.MaxPooling2D(2),
tf.keras.layers.Conv2D(16, 5, 1, 'same', activation='relu'),
tf.keras.layers.MaxPooling2D(2),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(100, activation='relu'),
tf.keras.layers.Dense(10, activation='softmax')
```

Figure 2: Network B used by Liu et al. [2017]

```
tf.keras.layers.Conv2D(20, 5, 1, 'valid', activation='relu'),
tf.keras.layers.MaxPooling2D(2),
tf.keras.layers.Conv2D(50, 5, 1, 'valid', activation='relu'),
tf.keras.layers.MaxPooling2D(2),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(100, activation='relu'),
tf.keras.layers.Dense(10, activation='softmax')
```

Figure 3: Network C used by LeCun et al. [1998]

```
tf.keras.layers.Conv2D(5, 5, 2, 'same', activation='relu'),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(100, activation='relu'),
tf.keras.layers.Dense(10, activation='softmax')
```

Figure 4: Network D used by Riazi et al. [2018]

B Communication

Table 1 shows the communication of our implementation in comparison to previous works. As Tan et al. [2021], we note that the implementation of Wagh et al. [2021] does not compute an appropriate gradient in the back-propagation, which limits the comparison.

C Fashion MNIST

We have run our implementation on Fashion MNIST for a more complete picture. Figure 5 shows our results.

Table 1: Comparison to previous work. Accuracy N/A means that the accuracy figures were not given or computed in a way that does not reflect the secure computation.

Network		Comm. per epoch (GB)	Acc. (# epochs)	Precision (f)
A	Wagh et al. [2021]	3	N/A	13
	Ours	26	97.9% (15)	16
	Ours	55	97.7% (15)	32
B	Wagh et al. [2021]	108	N/A	13
	Ours	20	93.6% (15)	16
	Ours	41	94.7% (15)	32
C	Wagh et al. [2021]	162	N/A	13
	Tan et al. [2021]	534	94.0% (5)	20
	Ours	352	94.9% (5)	16
	Ours	711	93.8% (5)	32
D	Wagh et al. [2021]	11	N/A	13
	Ours	41	96.8% (15)	16
	Ours	86	96.8% (15)	32

D Hyperparameter Settings

In the following we discuss our choice of hyperparameters.

Number of epochs As we found convergence after 100 epochs, we have run most of our benchmarks for 150 epochs, except for the comparison of optimizers where we stopped at 100.

Early stop We have not used early stop.

Mini-batch size We have used 128 throughout as it is a standard size. We briefly trialed 1024 as suggested by Li et al. [2017], but did not found any improvement.

Learning rate We have tried a number of learning rates as documented in the main paper. As a result, we settled for 0.01 for SGD in further benchmarks.

Learning rate decay/schedule We have not used either.

Random initialization The platform uses independent random initialization by design.

Dropout We have experimented with Dropout but not found any improvement.

Input preprocessing We have normalized the inputs to $[0, 1]$.

Test/training split We have used the usual MNIST split.

E Update Normalization

Agrawal et al. [2019] have suggested to normalize update gradients with AMSgrad (line 7 in Algorithm 4). However, Figure 6 shows that this does not improve the performance compared to SGD.

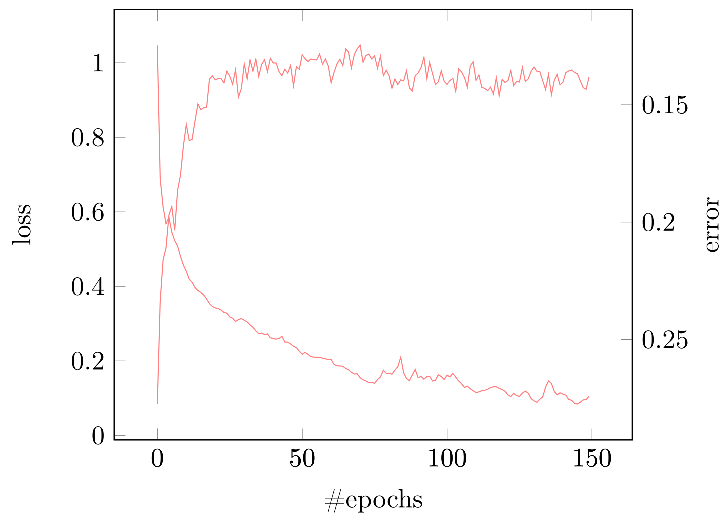

Figure 5: Network C with Fashion MNIST when running SGD with rate 0.01.

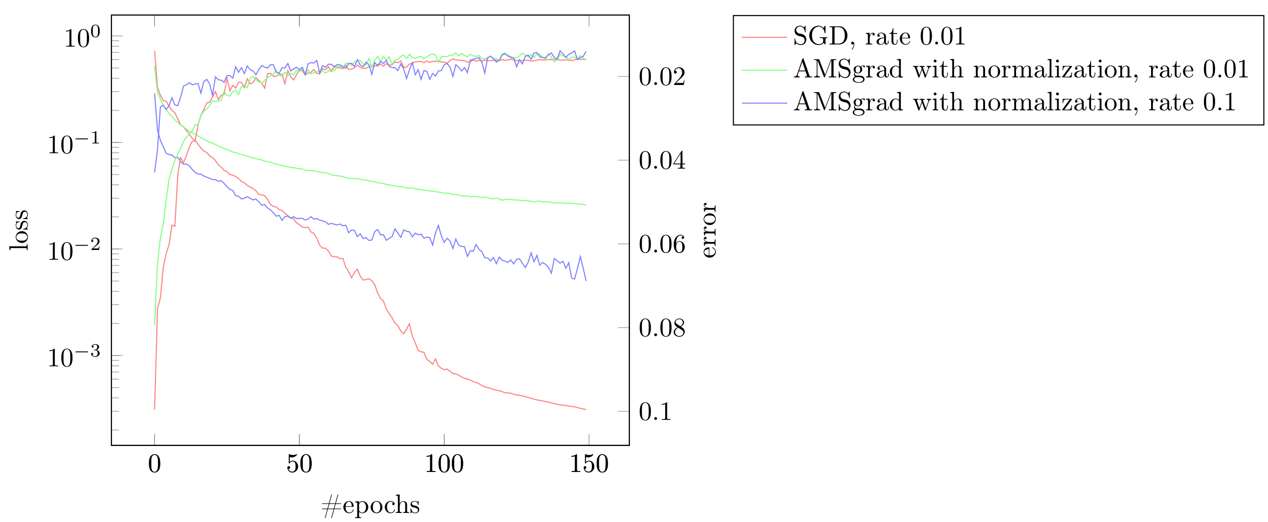

Figure 6: Loss and accuracy for network C, $f = 32$, and probabilistic truncation.

References

N. Agrawal, A. S. Shamsabadi, M. J. Kusner, and A. Gascón. QUOTIENT: Two-party secure neural network training and prediction. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 1231–1247. ACM Press, Nov. 2019. doi: 10.1145/3319535.3339819.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

H. Li, S. De, Z. Xu, C. Studer, H. Samet, and T. Goldstein. Training quantized nets: A deeper understanding. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/1c303b0eed3133200cf715285011b4e4-Paper.pdf`.

J. Liu, M. Juuti, Y. Lu, and N. Asokan. Oblivious neural network predictions via MiniONN transformations. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 2017*, pages 619–631. ACM Press, Oct. / Nov. 2017. doi: 10.1145/3133956.3134056.

P. Mohassel and Y. Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy*, pages 19–38. IEEE Computer Society Press, May 2017. doi: 10.1109/SP.2017.12.

M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In J. Kim, G.-J. Ahn, S. Kim, Y. Kim, J. López, and T. Kim, editors, *ASIACCS 18*, pages 707–721. ACM Press, Apr. 2018.

S. Tan, B. Knott, Y. Tian, and D. J. Wu. CryptGPU: Fast privacy-preserving machine learning on the GPU, 2021.

S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. *PoPETs*, 2021(1):188–208, Jan. 2021. doi: 10.2478/popets-2021-0011.