Low Latency Privacy Preserving Inference

Supplementary Material

1 Rings

In this work we consider commutative rings $\mathcal{R}$. A ring is a set which is equipped with addition and multiplication operations and satisfies several ring axioms such as $a + b = b + a$ for all $a, b \in \mathcal{R}$. A commutative ring is a ring in which the multiplication is commutative., i.e., $ab = ba$ for all $a, b \in \mathcal{R}$. Since all rings we consider are commutative, we use the term "ring" to refer to a commutative ring.

The set $\mathbb{Z}$ and the set $\mathbb{Z}_p$ of integers modulu p are rings. The elements of $\mathbb{Z}_p$ can be thought of as sets of the form $\{i + ap : a \in \mathbb{Z}\}$. The notation $k \in \mathbb{Z}_p$ refers to the set $\{k + ap : a \in \mathbb{Z}\}$. Alternatively, k is a representative of the the set $\{k + ap : a \in \mathbb{Z}\}$.

The set $\mathcal{R}[x]$ of polynomials with coefficients in a ring $\mathcal{R}$ is itself a ring. Thus, $\mathbb{Z}_p[x]$ is the ring of polynomials with coefficients in $\mathbb{Z}_p[x]$. Finally, we introduce the ring $\frac{\mathbb{Z}_p[x]}{x^n+1}$ which is the ring used in the BFV scheme. The elements of this ring can be thought of as sets of the form $\{r(x) + q(x)\,(x^n + 1) \mid q(x) \in \mathbb{Z}_p[x]\}$.

The notation $t(x) \in \frac{\mathbb{Z}_p[x]}{x^n+1}$ refers to the set $\{t(x) + q(x)\,(x^n + 1) \mid q(x) \in \mathbb{Z}_p[x]\}$. Conversely, we say that $t(x)$ is a representative of the set $\{t(x) + q(x)\,(x^n + 1) \mid q(x) \in \mathbb{Z}_p[x]\}$. For each element in $\frac{\mathbb{Z}_p[x]}{x^n+1}$, there is a representative in $\mathbb{Z}_p[x]$ with degree at most $n - 1$. Furthermore, any two non-equal polynomials of degree at most $n - 1$ in $\mathbb{Z}_p[x]$, are representatives of different elements in $\frac{\mathbb{Z}_p[x]}{x^n+1}$.

2 Parallel Scaling

The performance of the different solutions is affected by the amount of parallelism allowed. The hardware used for experimentation in this work has 8 cores. Therefore, we tested the performance of the different solutions with 1, 2, 4, and 8 cores to see how the performance varies. The results of these experiments are presented in Figure 1. These results show that at least up to 8 cores the performance of all methods scales linearly when tested on the MNIST data-set. This suggests that the latency can be further improved by using machines with higher core count. We note that the algorithm utilizes the cores well and therefore we do not expect large gains from running multiple queries simultaneously.

3 Lola-Dense

LoLa-Dense uses the same network layout as CryptoNets (see Figure 2) and has accuracy of 98.95%. However, it is implemented differently: the input to the network is a single dense message where the pixel values are mapped to coordinates in the encoded vector line after line . The first step in processing this message is breaking it into 25 messages corresponding to the 25 pixels in the convolution map to generate a convolution representation. Creating each message requires a single vector multiplication. This is performed by creating 25 masks. The first mask is a vector of zeros and ones that corresponds to a matrix of size 28×28 such that a one is in the (i, j) coordinate if the i, j pixel in the image appears as the upper left corner of the 5×5 window of the convolution layer. Multiplying point-wise the input vector by the mask creates the first message in the convolution representation hybrided with the interleaved representation. Similarly the other messages in the convolution representation are created.

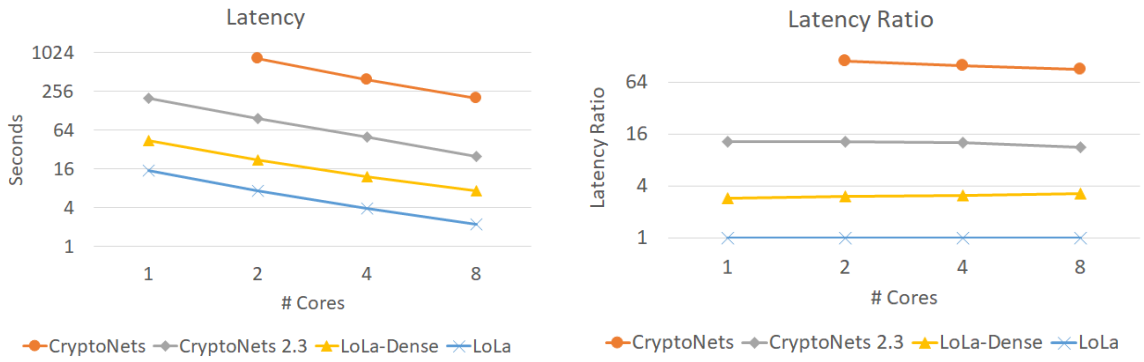

Figure 1: Left: latency of the different network implementations for the MNIST task with respect to the number of available cores. Right: ratio between the latency of each solution and the latency of LoLa

Note that all masks are shifts of each other which allows using the convolution representation-row major multiplication to implement the convolution layer. To do that, think of the 25 messages as a matrix and the weights of a map of the convolution layer as a sparse vector. Therefore, the outputs of the entire map can be computed using 25 multiplications (of each weight by the corresponding vector) and 24 additions. Note that there are 169 windows and all of them are computed simultaneously. However, the process repeats 5 times for the 5 maps of the convolution layer.

The result of the convolution layer are 5 messages, each one of them contains 169 results. They are united into a single vector by rotating the messages such that they will not have active values in the same locations and summing the results. At this point, a single message holds all the 845 values (169 windows $\times 5$ maps). This vector is squared, using a single multiplication operation, to implement the activation function that follows the convolution layer. This demonstrates one of the main differences between CryptoNets and LoLa; In CryptoNets, the activation layer requires 845 multiplication operations, whereas in LoLa it is a single multiplication. Even if we add the manipulation of the vector to place all values in a single message, as described above, we add only 4 rotations and 4 additions which are still much fewer operations than in CryptoNets.

Next, we apply a dense layer with 100 maps. LoLa-Dense uses messages of size $n = 16384$ where the 845 results of the previous layer, even though they are in interleaving representation, take fewer than 1024 dimensions. Therefore, 16 copies are stacked together which allows the use of the Stacked vector – Row Major multiplication method. This allows computing 16 out of the 100 maps in each operation and therefore, the entire dense layer is computed in 7 iterations resulting in 7 interleaved messages. By shifting the i^{th} message by $i-1$ positions, the active outputs in each of the messages are no longer in the same position and they are added together to form a single interleaved message that contains the 100 outputs. The following square activation requires a single point-wise-multiplication of this message. The final dense layer is applied using the Interleaved vector – Row Major method to generate 10 messages, each of which contains one of the 10 outputs.[1]

Overall, applying the entire network takes only 7.2 seconds on the same reference hardware which is 34.7$\times$ faster than CryptoNets and 3.4$\times$ faster than CryptoNets 2.3.

4 Secure CIFAR

The neural network used has the following layout: the input is a $3 \times 32 \times 32$ image (i) 3×3 linear convolution with stride of $(1, 1)$ and 128 output maps, (ii) 2×2 average pooling with $(2, 2)$ stride (iii)

[1]It is possible, if required, to combine them into a single message in order to save communication.

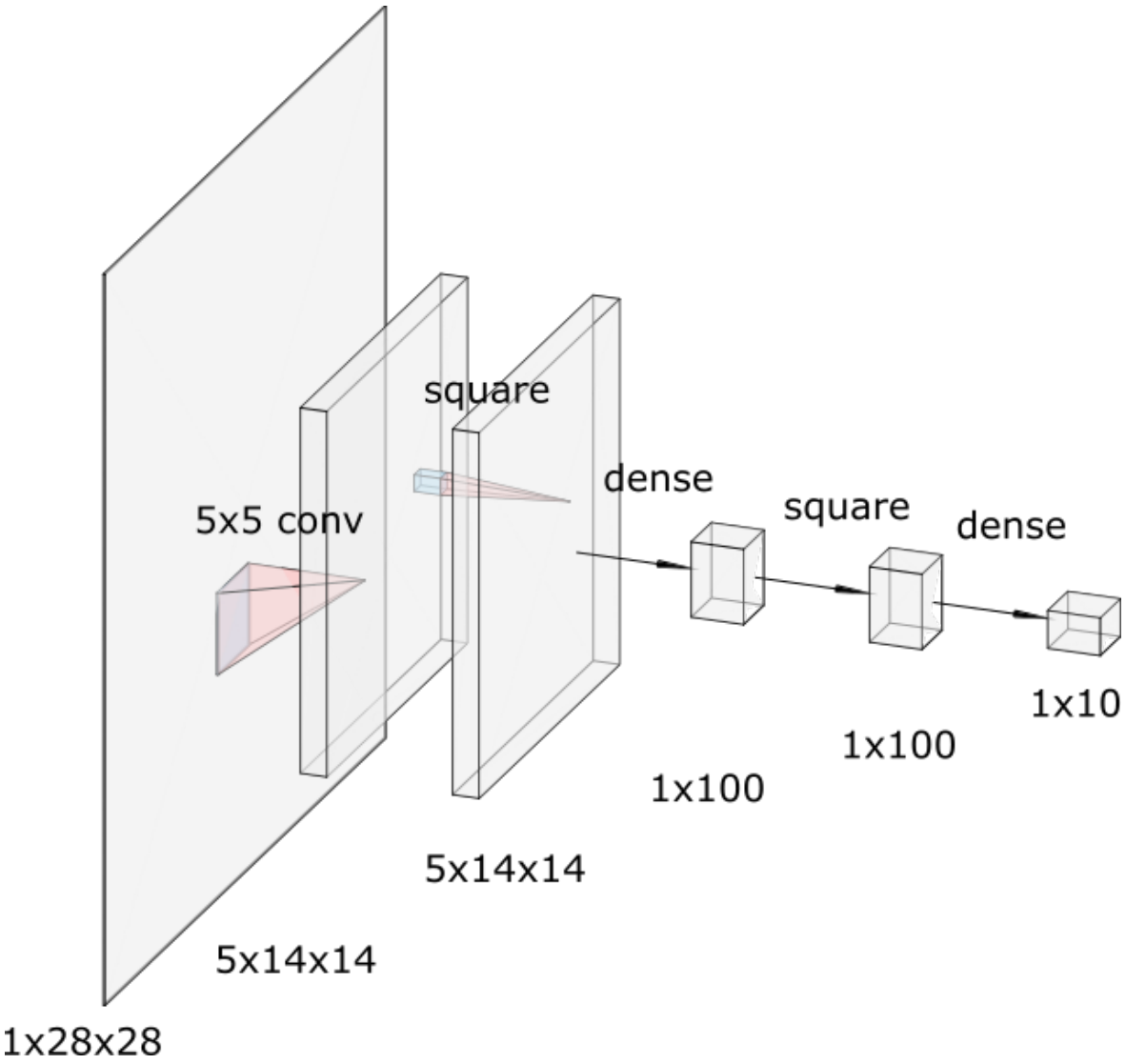

Figure 2: The structure of the network used for MNIST classification.

Layer	Input size	Representation	LoLa-Dense Operation
5×5 convolution layer	1×784	dense	mask input to create 25 messages
	25×169	convolution-interleave	convolution vector – row major mult'
	5×169	interleave	combine 5 messages into one
square layer	1×845	interleave	square
dense layer	1×845	interleave	stack 16 copies
	1×13520	stacked-interleave	stacked vector – row major mult'
	7×16	interleave	combine 7 messages into one
square layer	1×100	interleave	square
dense layer	1×100	interleave	interleaved vector – row major
output layer	10×1	sparse	

Table 1: Message size, message representation and operations in each layer of the LoLa-Dense inference solution on MNIST. The input size format is number of vectors $\times$ dimension

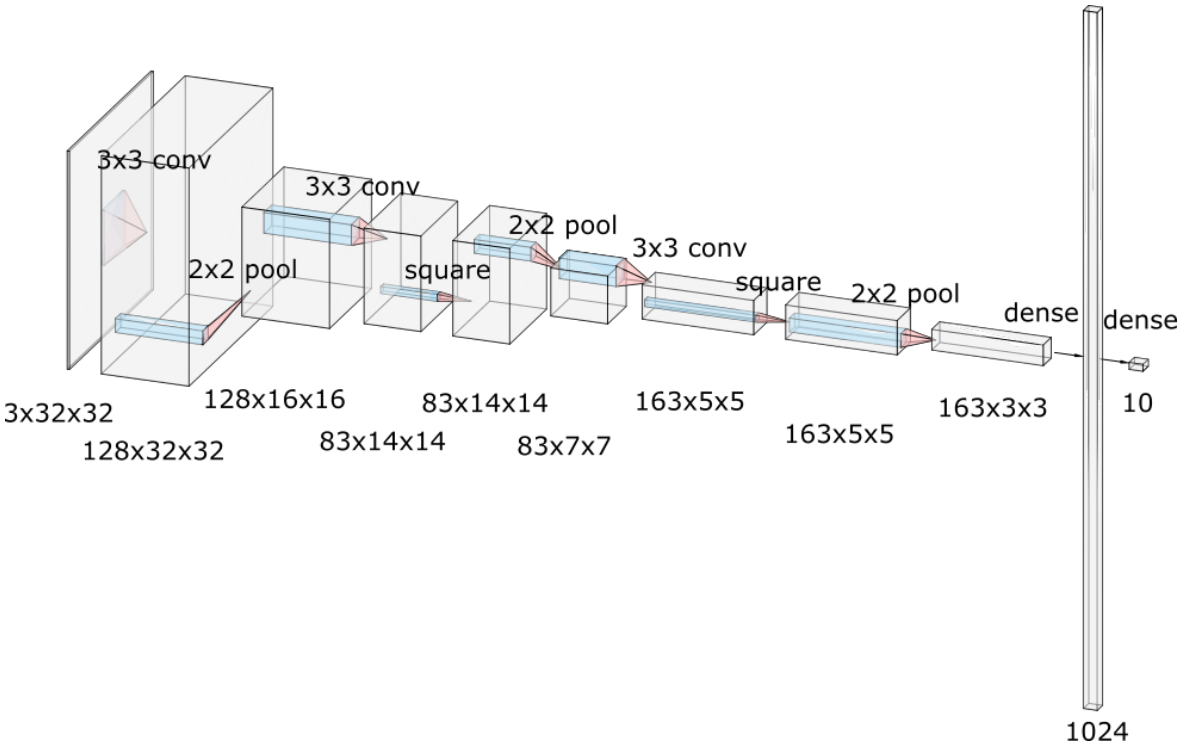

Figure 3: The structure of the network used for CIFAR classification.

Layer	Input size	Output format	Description
Preprocess	200×300	dense	apply convolution layers from Alex-Net
Encryption	4096	dense	image is encrypted into 1 message
dense layer	101	sparse	dense-vector row major multiplication

Table 2: Data representation changes for CalTech 101 task

3×3 convolution with $(1, 1)$ stride and 83 maps (iv) Square activation (v) 2×2 average pooling with $(2, 2)$ stride (vi) 3×3 convolution with $(1, 1)$ stride and 163 maps (vii) Square activation (vii) 2×2 average pooling with stride $(2, 2)$ (viii) fully connected layer with 1024 outputs (ix) fully connected layer with 10 outputs (x) softmax. ADAM was used for optimization Kingma & Ba (2014) together with dropouts after layers (vii) and (viii). We use zero-padding in layers (i) and (vii). See Figure 3 for an illustration of the network.

For inference, adjacent linear layers were collapsed to form the following structure: (i) $8 \times 8 \times 3$ convolutions with a stride of $(2, 2, 0)$ and 83 maps (ii) square activation (iii) $6 \times 6 \times 83$ convolution with stride $(2, 2, 0)$ and 163 maps (iv) square activation (v) dense layer with 10 output maps. See Figure 4 for an illustration.

5 CalTech-101

Table 2 shows the different data representations when using the method proposed for private inference on the CalTech-101 dataset using deep representations. [2]

[2]In Table 2 we use the terminology of dense vectors also in the first stage of applying Alex-Net before the encryption.

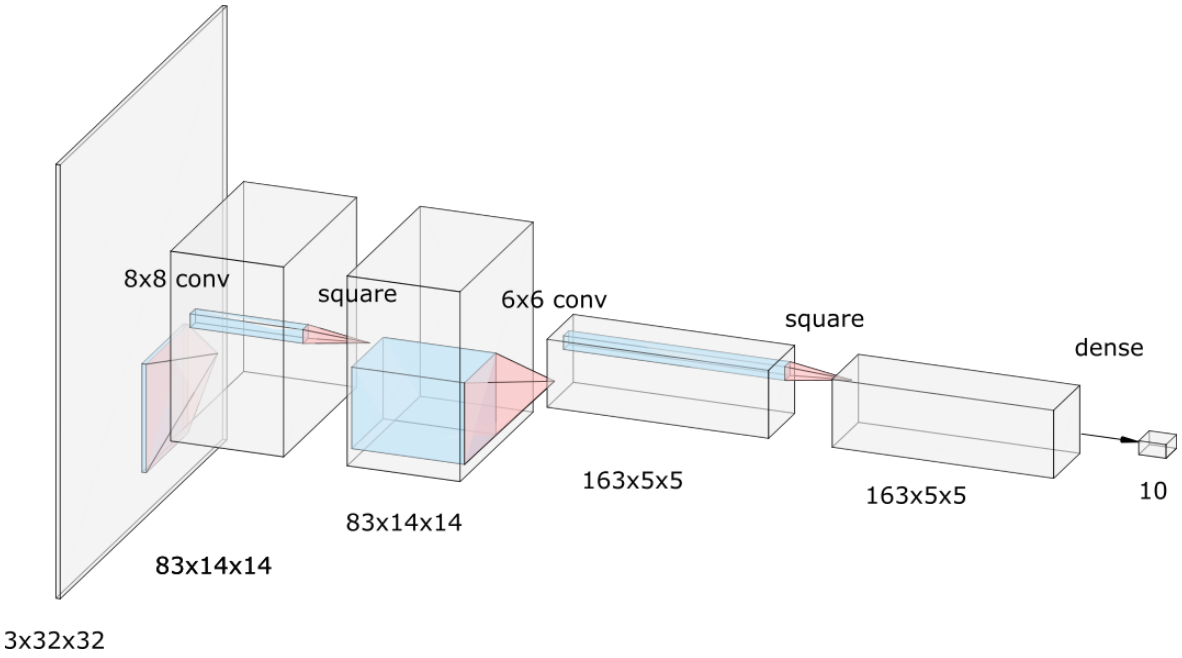

Figure 4: The structure of the network used for CIFAR classification after collapsing adjacent layers.

References

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.