S-LSTM-GAN: Shared Recurrent Neural Networks with Adversarial Training

Amit Adate and B. K. Tripathy

Abstract In this paper, we propose a new architecture *Shared-LSTM Generative Adversarial Network* (S-LSTM-GAN) that works on recurrent neural networks (RNNs) via an adversarial process and we apply it by training it on the handwritten digit database. We have successfully trained the network for the generator task of handwritten digit generation and the discriminator task of its classification. We demonstrate the potential of this architecture through conditional and quantifiable evaluation of its generated samples.

Keywords Generative adversarial networks · Recurrent neural networks · Adversarial training · Handwritten digit generation · Deep learning

1 Introduction

Generative adversarial networks (GANs) are a relatively new category of neural network architectures which were conceptualized with the aim of generating realistic data [1]. Their method involves training two neural networks, architectures with contrasting objectives, a generator, and a discriminator. The generator tries to produce samples that look authentic, and the discriminator tries to differentiate between the generated samples and real data. This methodology makes it possible to train deep models without expensive normalizing constants, and this framework has proven to produce highly realistic samples of data [2].

The GAN framework is the most popular architecture with successes in the line of research on adversarial training in deep learning [3] where the two-player minimax

A. Adate (✉) · B. K. Tripathy
VIT University, Vellore, India
e-mail: email2amitadate@gmail.com; adateamit.sanjay2014@vit.ac.in

B. K. Tripathy
e-mail: tripathybk@vit.ac.in

A. J. Kulkarni et al. (eds.), *Proceedings of the 2nd International Conference on Data Engineering and Communication Technology*, Advances in Intelligent Systems and Computing 828, https://doi.org/10.1007/978-981-13-1610-4_11

game is crafted carefully so that the convergence of the networks attains the optimal criteria. The initial work on GANs focused on generating images [4], however, GANs have a range of application domains like feature learning [5] and improved image generation [2].

In this work, we propose a hybrid model called as S-LSTM-GAN (Shared-LSTM-GAN) and we investigate the viability of using adversarial training on it for the task for handwritten digit generation. We would demonstrate the workings of two closely related neural networks, both variants of the recurrent neural unit, long short-term memory (LSTM) cells.

2 Background: LSTM

LSTM is a variant of the RNN and was introduced in [6]. The fundamental idea behind LSTM cell was the use of memory cell for retaining data for longer time and overcoming the limitations of recurrent neural networks. RNNs have problem with recognizing long-term dependencies like recognizing sequences of words which are quite apart from each other, and this problem is also referred to as the vanishing gradient problem. More technically speaking, the values in the matrix and multiple matrix multiplication are diminishing or becoming closer to zero and after a few time steps they vanish completely [7].

At far away time steps gradient is zero, and these gradients are not contributing to learning. In fact, vanishing gradient problem is not only limited to RNN. They are also observed in case of deep feed-forward neural networks.

3 S-LSTM-GAN: Shared Recurrent Networks with Adversarial Training

The proposed model consists of two different deep recurrent neural models with adversarial training. The adversaries are made up of two different architectures, but both sharing their weights. The generator (G) is trained to generate pixel values that are similar to the real data, while the discriminator (D) is trained to identify the generated data. The training can be modeled as a two-player minimax game for which the equilibrium is reached when the generator can consistently generate digits which the discriminator cannot identify from the real data. We define the following loss functions for the discriminator and generator, respectively, L_D and L_G

$$L_G = \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)}))) \tag{1}$$

$$L_D = \frac{1}{m}\sum_{i=1}^{m}\left[-\log(D(x^{(i)})) - (\log(D(G(z^{(i)}))))\right] \tag{2}$$

where $(z^{(i)})$ is the sequence of input vectors and $x^{(i)}$ is a sequence from the training data. k is the dimensionality of the input data.

The overall model is depicted in Fig. 1. In the figure, the pink portion of the model is the generator and the brown portion is the discriminator. For reasons of clarity, the image is split into quadrants here, but in our experiments the aim was to split the image into pixels in an effort to create a generator that could create digits by each pixel using the long-range memory of the LSTM cells. The architectures of the generator and the discriminator are elaborated in Figs. 2 and 3, respectively.

The input data for each cell in G comes from a random vector, merged with the output of the previous cell, similar to application in [8]. The dimensionality of the input data set is defined as the number of sections we have sampled the dataset into, we are essentially splitting the image into k sections and then feeding them to the LSTM layers sequentially.

The discriminator consists of a series of LSTM cells all sharing the weights with their adversaries, and they feed forward into a linear neural network which is twice the size of the individual LSTM cells. Further, a softmax layer with a linear score function was applied to perform the task of classification.

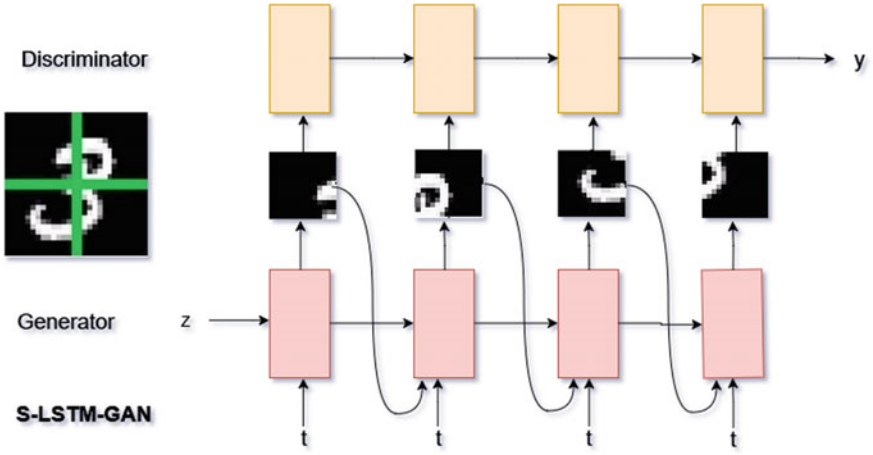

Fig. 1 S-LSTM-GAN: two recurrent neural sharing weights and training in parallel

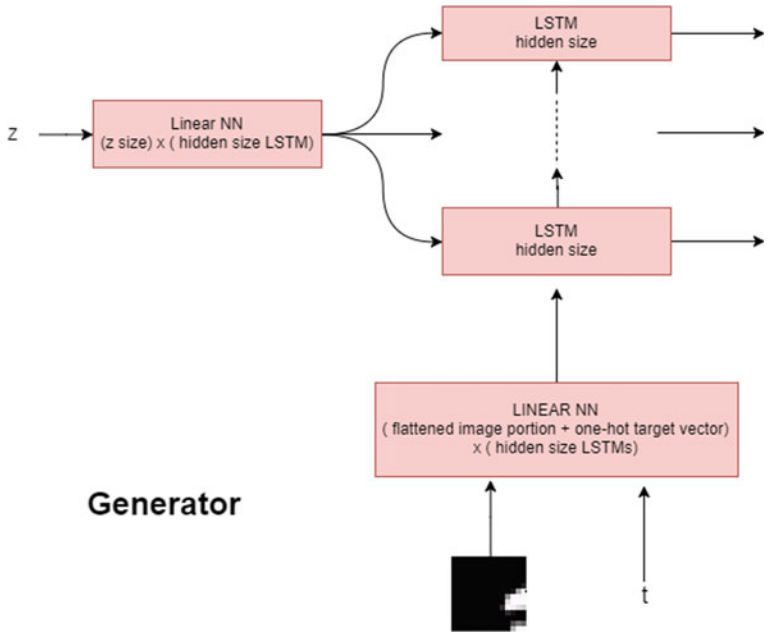

Fig. 2 S-LSTM-GAN: the generator architecture

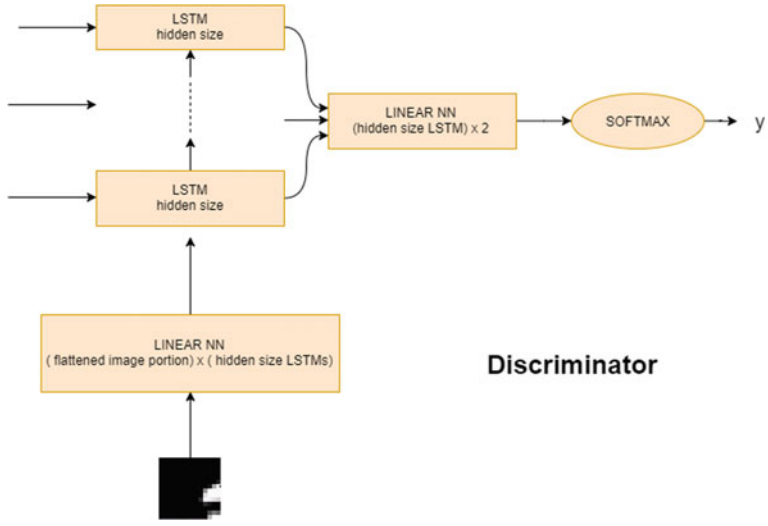

Fig. 3 S-LSTM-GAN: the discriminator architecture

4 Experimental Setup

We evaluated the above-mentioned architecture on the MNIST dataset [4] for two separate instances. Firstly with dividing it into four segments, refer Fig. 1 and secondly by dividing into 16 segments.

Model Layout Details: The number of layers in the LSTM network in the generator is six, while in the discriminator is two. They have 600 and 400 internal (hidden) units, respectively. Both architectures have unidirectional layout. In D, the cells feed forward to a fully connected network, with weights that are shared across each time step. The activation function used for each cell is sigmoid.

Dataset: We have imported the MNIST dataset, and its images contain gray levels due to anti-aliasing technique used by the normalization algorithm. The images were centered in an image by computing the center of mass of the pixels and translating the image so as to position this point at the center of the 28×28 field. We segment that image as per our instance, and this sets the dimensionality k for our model. We operate directly on the pixel values considering each image as a 28×28 matrix.

Training: Backpropagation and mini-batch stochastic gradient descent were used. For the initial instance of 4 segments, learning rate of 0.0001 for the second instance of 16 segments, learning rate of 0.0002 was used. The GPU used was Quadro P6000, and we trained the model for 10^7 epochs. For the generator, we recorded the loss as it changed alongside the increment in epochs. And for the discriminator, the metric we recorded was classification accuracy, best being one, against the rise in epochs. Refer Fig. 4.

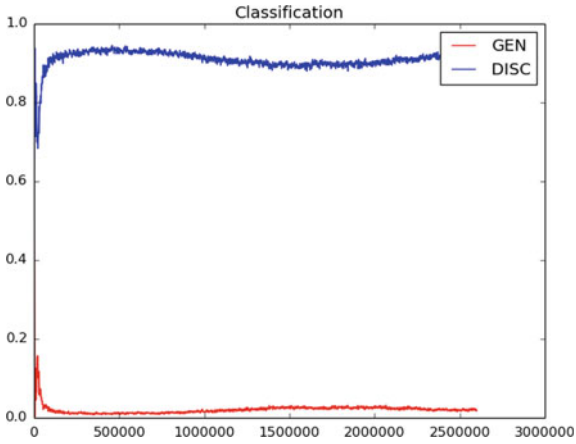

Fig. 4 S-LSTM-GAN: classification versus epochs during training with instance of four segments

At four segments, we had to train for the initial for 5×10^5 epochs to see the generator able to generate image that were giving a steady loss, and then the spike in loss started for the generator while the discriminator converges at loss = 0.25. We are generating a sample at every 10^5 epoch. Hence, our sample at the third checkpoint was having some distinguishable features compared to the rest.

At 16 segments, we got better results than the previous instance, and here the classification was comparatively cleaner, with just a few spikes. The generator showed signs of heavy spikes in losses in early epochs, but later converged upon a value. It is noted that the loss is higher than the previous instance, but it still yields a cleaner classification. Due to the high variance, no singular checkpoint was distinguishable but compared to its predecessor, and they were easily recognizable to the human eye.

5 Results

The results of our experimental study are presented in Figs. 8 and 9 for 4 time steps and 16 time steps, respectively. We have chosen a very small learning rate to help the model learn the pixel values with more variability and larger intensity span. Allowing multiple layers for the generator and allowing it to train for these many epochs help to generate handwritten digits with a high degree of classification.

Figure 5 reveals that to attain convergence we can reduce the number of iterations by 2×, for the task of building the classifier net only. Figure 7 displays the variance of the same model over the initial iterations of turbulent variance in the accuracy

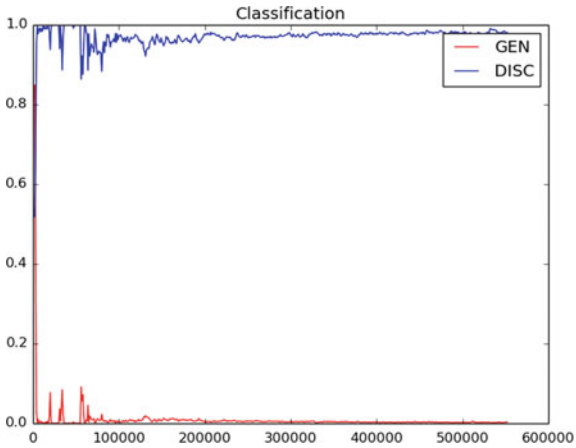

Fig. 5 S-LSTM-GAN: classification versus epochs during training with instance of 16 segments

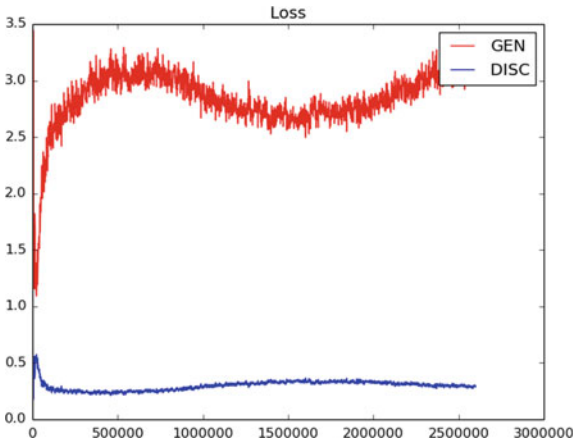

Fig. 6 S-LSTM-GAN: loss versus epochs during training with instance of four segments

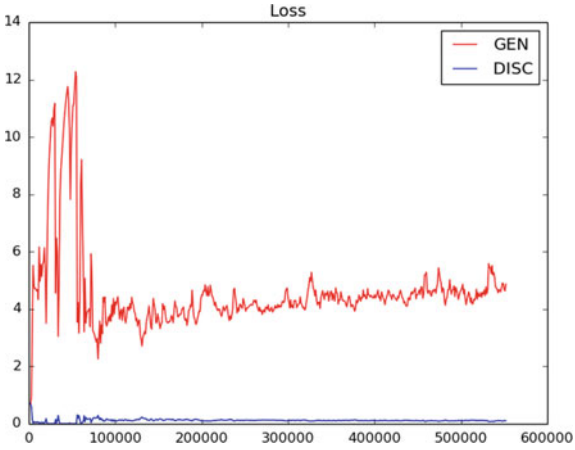

Fig. 7 S-LSTM-GAN: loss versus epochs during training with instance of 16 segments

of its digit generation. We maintain generator accuracy by varying dropout and the network depth of six layers.

Unlike the graph presented in Fig. 6, we have found that beyond the 16-segment instance, the generator would collapse at early iterations. We believe this is because the generator has to reduce the variance of the minimax objective over the iterations, and it will not be able to continue to do the following as implied in Fig. 7. Hence, if we were to segment the image ahead of 16 segments, the proposed model would fail.

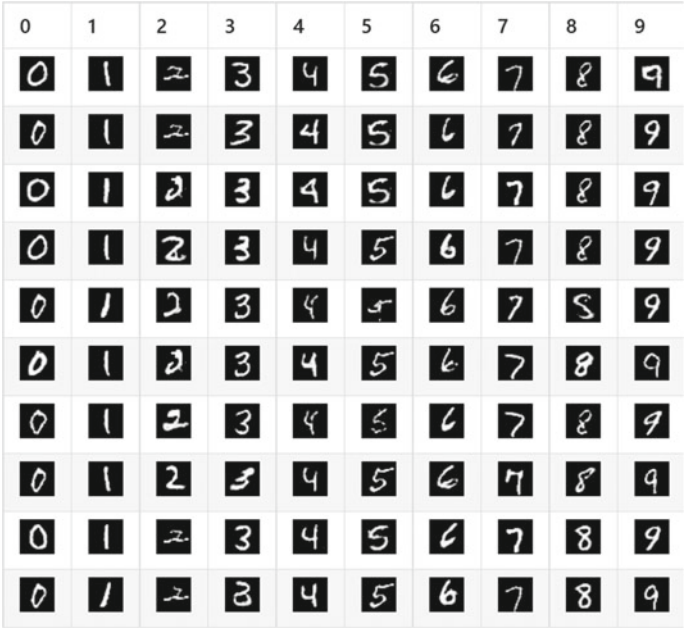

Fig. 8 S-LSTM-GAN: images generated during training with instance of four segments

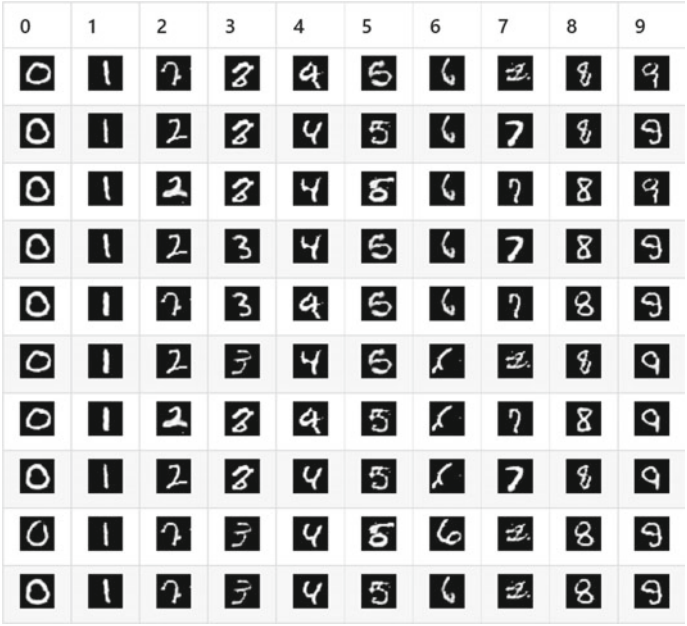

Fig. 9 S-LSTM-GAN: images generated during training with instance of 16 segments

We also found that beyond the 16-segment instance, the generator would collapse at early iterations. We believe this is because the generator has to reduce the variance of the minimax objective over the iterations, and it will not be able to continue to do the following as implied in Fig. 7. Hence, if we were to segment the image ahead of 16 segments, the proposed model would fail.

To view our code and the images generated by S-LSTM-GAN: https://github.com/amitadate/S-LSTM-GAN-MNIST.

6 Conclusion

In this paper, we have proposed a neural network model for the task of learning handwritten digits trained using an application based on generative adversarial networks. We believe that training could be accelerated greatly by developing better techniques for synchronizing the generator and the discriminator or by forming better partitioning of the data sample before or during training.

In future work, we will look at more sophisticated mechanisms for applying the similar approach toward the task of handwriting synthesis. For that amount of features, introducing multiple generators would be ideally the next step, and the discriminator would also be needed to be broadened as per the feature–label pairs.

References

1. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press. Book in preparation for MIT Press
2. Ledig C, Theis L, Huszar F, Caballero J, Aitken AP, Tejani A, Totz J, Wang Z, Shi W (2016) Photo-realistic single image super-resolution using a generative adversarial network. CoRR, arXiv:1609.04802
3. Ganin Y, Ustinova E, Ajakan H, Germain P, Larochelle H, Laviolette F, Marchand M, Lempitsky V (2016) Domain-adversarial training of neural networks. J Mach Learn Res 17(1):2030–2096
4. LeCun Y, Cortes C (2010) MNIST handwritten digit database
5. Donahue J, Jia Y, Vinyals O, Hoffman J, Zhang N, Tzeng E, Darrell T (2013) Decaf: A deep convolutional activation feature for generic visual recognition. CoRR, arXiv:1310.1531
6. Graves A, Schmidhuber J (2005) Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. 18:602–610
7. Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J (2015) LSTM: a search space odyssey. CoRR, arXiv:1503.04069
8. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780