Pixel Recurrent Neural Network

Yue Han

Abstract

This paper discusses the article *Pixel Recurrent Neural Networks* and its contributions to generative models for images via Long Short Term Memory layers (LSTM). More specifically, row LSTM and diagonal Bidirectional LSTM (biLSTM) layers that aid the RNN to model local and long-range correlations (Aaron et.al, 2016). Furthermore, I will present how LSTM works and how LSTM's are used to counter vanishing and exploding gradients that regular RNN's face (Ibanez, 2022). Despite Pixel RNN's providing a useful way to model two-dimensional natural images, there are some restrictions and assumptions. It is dependent on the quality of the training dataset (natural and high resolution images), only up to 12 layers were tested in the original paper, it takes a while to train, and LSTM assumes the present timestep state depends on previous timestep (Fei, 2017). However, future directions include more testing using a larger dataset and more layers, 3D image modeling, video modeling, or changing the LSTM layers to change the structure to include Advanced LSTM (A-LSTM) or Sequencers proposed in (Fei & Gang,2017) and (Yuki & Masato, 2023).

1 Problem Statement

Pixel Recurrent Neural Networks aims to model the distribution of large quantities of natural images using a generative model. This is a challenging problem since images are high dimensional and require "context" when dealing with downsizing. In other words, the idea us to see the "big picture" without seeing the entire picture. Meaning, tractability, scalablity, and expressivity are needed. Previous attempts include VAE's that have an inference step that results in poor performance and original RNN's can have vanishing and exploding gradients. Thus, there always seems to be a trade-off making it difficult to find a solution that fulfills our needs.

2 Main Result

This paper has suggested two types of two-dimensional Long Short Term Memory (LSTM) layers that can be used in the Pixel RNN (Aaron et.al, 2016). The two types are row LSTM and Bidirectional LSTM (BiLSTM). As Aaron et.al (2016) states, this has resulted in the ability to model local and long-range correlations. Additionally, this use of LSTM's addresses a problem RNN's run into. RNN's have gradients that are used to update the weights but these gradients can vanish or explode (Ibanez, 2022). Ibanez states, that vanishing happens when there are many layers in the network (long data sequences which is particularly true for large images we use to train) and you have a small gradient update. The update that takes place deep in the network does not affect the earliest layers thus, the network can't learn. When the gradient is too large it leads to the learning being unreliable. This problem makes modeling long-range correlations difficult. However, by using LSTM we can solve this problem, we will address how later in the empirical studies section. Below I will give a brief overview of LSTM, row LSTM used in the pixel RNN, BiLSTM, and diagonal BiLSTM used in the pixel RNN with a series of figures. The main focus will be however, basic LSTM and gradients.

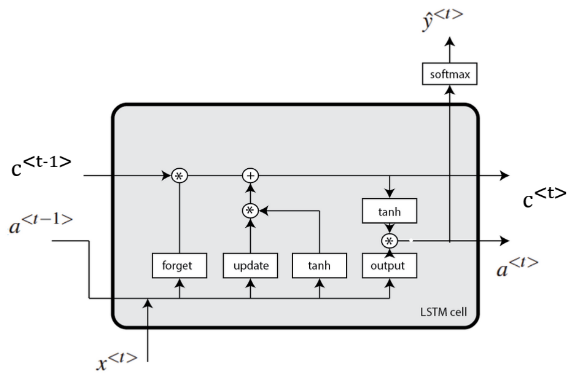

Figure 1: LSTM cell (Zvornicanin, 2022)

2.1 LSTM

LSTM stands for long short term memory, it is designed to be able to remember and forget. From 1, we see that a LSTM block contains the following gates, the forget gate f_t, the update/input gate u_t, and the output gate o_t. The forget gate chooses the info the current memory cell will get from the previous cell by assigning a value from 0 to 1 where 0 means forget all and 1 means remember all (Zvornicanin, 2022). More on this in the table below. The update gate controls if and what gets updated. While the output gate controls what is the next hidden state.

2.2 LSTM Values

$u_t = \sigma(W_{uu}a^{<t-1>} + W_{ux}x^{<t>} + b_u)$

$f_t = \sigma(W_{ff}a^{<t-1>} + W_{fx}x^{<t>} + b_f)$

$o_t = \sigma(W_{oo}a^{<t-1>} + W_{ox}x^{<t>} + b_o)$

$\hat{c}^{<t>} = tanh(W_{cc}a^{<t-1>} + W_{cx}x^{<t>} + b_c) =$ potential memory cell value

$c^{<t>} = u_t \odot \hat{c}^{<t>} + f_t \odot c^{<t-1>} =$ current memory cell value

$a^{<t>} = o_t \odot tanh(c^{<t>}) =$ output value/hidden state

σ is the logistic sigmoid function and W is the weight matrix and $\odot$ is element-wise product

(Zvornicanin, 2022)

update/input gate	forget gate	result
0	0	erase
0	1	remember
1	0	overwrite
1	1	contribute to previous

(Grosse, 2017)

2.3 Row LSTM

Row LSTM is a unidirectional LSTM layer where one-dimensional convolution is applied row by row from top to bottom resulting in "triangular" context above the pixel 2(Aaron et.al, 2016). This means it does not get the entire context available unlike in diagonal BiLSTM.

2.4 BiLSTM

BiLSTM introduces another LSTM layer that reverses the input and combines the two outputs from the two layers 3 (Zvornicanin, 2022). That means there are two directions. This allows the current cell to get all context available to this point (past to present) 2 but this comes with the cost of slower training than LSTM.

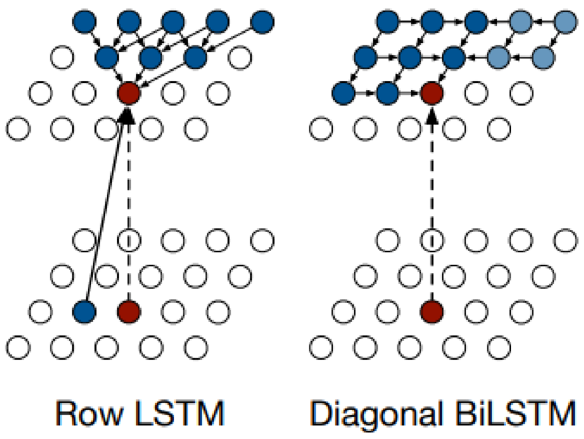

Figure 2: Row LSTM and Diagonal BiLSTM context (Aaron et.al, 2016)

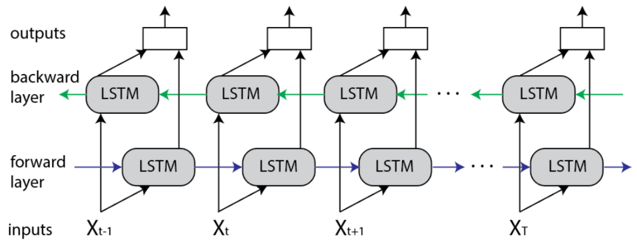

Figure 3: BiLSTM layer (Zvornicanin, 2022)

2.5 Diagonal BiLSTM

We adjust the input rows by 1 from the previous row resulting in a $n \times (2n-1)$ map (Aaron et.al, 2016). Allowing for the full context available to be acquired by the current pixel and still being parallel computation. For the two directions we do 1×1 convolution for input to state and then do 2 $\times 1$ kernel column-wise convolution for state to state (Aaron et.al, 2016). Then we get the output map and make it $n \times n$ (Aaron et.al, 2016). So we get the full context with small computations leading to a non-linear computation because of the small kernel (Ingham, 2019).

3 Examples and Counterexamples

3.1 Vanishing Gradient and LSTM

From earlier, we addressed that LSTM solves the vanishing gradient problem that RNN's face. We will dive into why this happens. Firstly for RNN during back propogation the gradient is:

(Arbel, 2020)

$$\frac{\delta E}{\delta W} = \sum_{t=1}^{T} \frac{\delta E_t}{\delta W} \to 0 \tag{1}$$

where E is the prediction error and W is weight matrix.

$$\frac{\delta E_k}{\delta W} = \frac{\delta E_k}{\delta a_k} \frac{\delta a_k}{\delta c_k} (\Pi_{t=2}^{k} \frac{\delta c_t}{\delta c_{t-1}}) \frac{\delta c_1}{\delta W} \tag{2}$$

Where a_k is output vector.

$$c_t = \sigma(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \tag{3}$$

$$\begin{aligned} \frac{\delta c_t}{\delta c_{t-1}} &= \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot \frac{\delta}{\delta c_{t-1}}(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \\ &= \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot (W_{rec}) \end{aligned} \tag{4}$$

and plugging this back into (2) we get

$$\frac{\delta E_k}{\delta W} = \frac{\delta E_k}{\delta a_k}\frac{\delta a_k}{\delta c_k}(\Pi_{t=2}^{k}\sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot (W_{rec}))\frac{\delta c_1}{\delta W} \tag{5}$$

When W_{rec} is very large it creates the exploding gradient problem (Arbel, 2020). When k is really large it can make the gradients vanish since $\sigma' < 1$.

For LSTM.

$$\begin{aligned} \frac{\delta c_t}{\delta c_{t-1}} &= \frac{\delta}{\delta c_{t-1}}[c_{t-1} \odot f_t + \hat{c_t} \odot u_t] \\ &= \frac{\delta}{\delta c_{t-1}}[c_{t-1} \odot f_t] + \frac{\delta}{\delta c_{t-1}}[\hat{c}_t \odot u_t] \\ &= \frac{\delta f_t}{\delta c_{t-1}} \cdot c_{t-1} + \frac{\delta c_{t-1}}{\delta c_{t-1}} \cdot f_t + \frac{\delta u_t}{\delta c_{t-1}} \cdot \hat{c}_t + \frac{\delta \hat{c}_t}{\delta c_{t-1}} \cdot u_t \end{aligned} \tag{6}$$

calculating it further

$$\begin{aligned} \frac{\delta f_t}{\delta c_{t-1}} \cdot c_{t-1} &= \frac{\delta}{\delta c_{t-1}}[\sigma(W_{ff}a_{t-1} + W_{fx}x_t + b_f)] \\ &= \sigma'[W_{ff}a_{t-1} + W_{fx}x_t + b_f]\frac{\delta a_t}{\delta c_{t-1}} \cdot c_{t-1} \\ &= \sigma'(W_{ff}a_{t-1} + W_{fx}x_t + b_f) \cdot W_{ff} \cdot \Gamma_o \odot tanh'(c_{t-1}) \cdot c_{t-1} \end{aligned} \tag{7}$$

Similarly we calculate

$$\frac{\delta c_{t-1}}{\delta c_{t-1}} \cdot f_t = f_t \tag{8}$$

$$\frac{\delta u_t}{\delta c_{t-1}} \cdot \hat{c}_t = \sigma'(W_{uu}a_{t-1} + W_{ux}x_t + b_u) \cdot W_{uu} \cdot \Gamma_o \odot tanh'(c_{t-1}) \cdot \hat{c}_{t-1} \tag{9}$$

$$\frac{\delta \hat{c}_t}{\delta c_{t-1}} \cdot u_t = tanh'(W_{cc}a_{t-1} + W_{cx}x_t + b_c) \cdot W_{cc} \cdot \Gamma_o \odot tanh'(c_{t-1}) \cdot \hat{u}_t \tag{10}$$

Then plugging (7-10) into (6) we get

$$\begin{aligned} \frac{\delta c_t}{\delta c_{t-1}} &= \sigma'(W_{ff}a_{t-1} + W_{fx}x_t + b_f) \cdot W_{ff} \cdot \Gamma_o \odot tanh'(c_{t-1}) \cdot c_{t-1} + f_t \\ &+ \sigma'(W_{uu}a_{t-1} + W_{ux}x_t + b_u) \cdot W_{uu} \cdot \Gamma_o \odot tanh'(c_{t-1}) \cdot \hat{c}_{t-1} + \\ &tanh'(W_{cc}a_{t-1} + W_{cx}x_t + b_c) \cdot W_{cc} \cdot \Gamma_o \odot tanh'(c_{t-1}) \cdot \hat{u}_t \\ &= P_t + Q_t + R_t + S_t \end{aligned} \tag{11}$$

And we can plug (11) this back into (2)

$$\frac{\delta E_k}{\delta W} = \frac{\delta E_k}{\delta a_k}\frac{\delta a_k}{\delta c_k}(\Pi_{t=2}^{k} P_t + Q_t + R_t + S_t)\frac{\delta c_1}{\delta W} \not\rightarrow 0 \tag{12}$$

This is because (11) can be influenced by the forget gate thus making us able to prevent the gradient from vanishing. Additionally, it is additive, thus making it easier to maintaing the function from vanishing. Similarly for exploding gradients we can control it using LSTM.

4 Empirical Studies

We can show RNN's can vanish and explode with the following examples.

4.1 Exploding Gradient Example

From (5) let $W_{rec} >> \sigma'$ such that it overwhelms σ' then $(\Pi_{t=2}^{k}\sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot (W_{rec}))$ will be extremely large resulting in (5) to be extremely large then the weight update $W \leftarrow W - \alpha\frac{\delta E}{\delta W}$ will be extremely different and unreliable.

4.2 Vanishing Gradient Example

From (5) letting σ' extremely small then the product $(\Pi_{t=2}^{k}\sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot (W_{rec})) \rightarrow 0$ then (2) which is $\frac{\delta E_k}{\delta W} \rightarrow 0$ then (1) which is the summation $\sum_{t=1}^{T}\frac{\delta E_t}{\delta W} \rightarrow 0$ and so the update for the weights will become $W \leftarrow W - \alpha\frac{\delta E}{\delta W} \approx W$ which means there will be no useful learning that can take place in a reasonable time frame.

4.3 LSTM Gradient Example

We can further prove LSTM prevents vanishing and exploding gradients by providing the following numerical example:

From (12) let $P_t \approx 0.2$, $Q_t \approx 0.5$, $R_t \approx 0.2$ and $S_t \approx 0.1$ then $\Pi_{t=2}^{k}P_t + Q_t + R_t + S_t \approx \Pi_{t=2}^{k}1 \not\rightarrow 0$ thus making (12) not vanish or explode.

We can then conclude that LSTM's contain the ability to prevent vanishing and exploding gradients. Furthermore, it also allows us to see a broader context, as stated earlier, thus we can see the importance of the inclusion of LSTM in Pixel RNN.

5 Limitations

The dataset used has quality natural images for training and testing. This is a reasonable restriction as these images are easily accessible and are in large quantity. Additionally, only up to 12 layers were used in the original article, thus more research needs to be done with larger datasets to know how the Pixel RNN reacts (Aaron et.al , 2016). Another restriction is time, the training takes a long time which is usually the case for generative models especially for images. However, this training time can be improved upon as a future direction. Another assumption takes place for LSTM, LSTM assumes that the current state relies on the past timestep (Fei, 2017). This restricts the model's ability to model temporal information (Fei, 2017). This can affect expansion into future topics such as video.

6 Future Directions

There are some future endeavors that may be interesting to explore. For example, we could advance to 3D images or video modeling in the future. The Pixel RNN focuses on 2D images, however, this can be seen as a building block for more advanced imaging. Additionally, we could improve the Pixel RNN structure. We could use Advanced LSTM to allow for better temporal context modeling as in (Fei, 2017). Or we could use sequencers which is deep LSTM to increase accuracy (Yuki & Masato, 2023). Furthermore, research can be done on larger datasets since only up to 12 layers were tested in the original article.

7 References

Aaron van den Oord, Nal Kalchbrenner, & Koray Kavukcuoglu. (2016). Pixel Recurrent Neural Networks.

Arbel, N. (2020, May 16). How LSTM networks solve the problem of vanishing gradients. Medium. Retrieved April 2023, from https://medium.datadriveninvestor.com/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577

Fei Tao, & Gang Liu. (2017). Advanced LSTM: A Study about Better Time Dependency Modeling in Emotion Recognition.

Grosse, R. (2017). Lecture 15: Exploding and Vanishing Gradients. Retrieved April 2023, from https://www.cs.toronto.edu/ rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20 Vanishing%20Gradients.pdf

Ibanez, W. by: D. (2022, November 11). Encoder-decoder models for natural language processing. Baeldung on Computer Science. Retrieved April 2023, from https://www.baeldung.com/cs/nlp-encoder-decoder-models#1-what-are-vanishingexploding-gradients

Ingham, F. (2019, May 1). Day 4: Pixel recurrent neural networks. Medium. Retrieved April 2023, from https://medium.com/a-paper-a-day-will-have-you-screaming-hurray/day-4-pixel-recurrent-neural-networks-1b3201d8932d

Olah, Christopher. Understanding LSTM networks. Understanding LSTM Networks – colah's blog. (2015). Retrieved April 2023, from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Yuki Tatsunami, & Masato Taki. (2023). Sequencer: Deep LSTM for Image Classification.

Zvornicanin, W. by: E. (2022, November 6). Differences between bidirectional and Unidirectional LSTM. Baeldung on Computer Science. Retrieved April 2023, from https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm