17044633_DL_hw3

March 11, 2018

1 Deep Learning: Homework 3

Name: Yuan Zhang
SN: 17044633

1.1 Q1: Understanding LSTM vs GRU (30 pts)

Before going deeper into your practical tasks, take some time to revise and make sure you understand the two major types of recurrent cells you will be using in this assignment: Long-Short Term Memory Units (LSTM) first introduced by Hochreiter and Schmidhuber [1997] and the more recent Gated Recurrent Units (GRU) by Cho et al. [2014]. Once you have done this, answer the following questions:

1. Can LSTMs (and respectively GRUs) just store the current input in the state (c_t for LSTM and h_t for GRU, in the class notation) for the next step? If so, give the gates activation that would enable this behaviour. If not, explain why not. [10 pts]
2. Can LSTMs (and respectively GRUs) just store a previous state into the current state and ignore the current input? If so, give the gates' activation that would enable this. If not, explain why not. [10 pts]
3. Are GRUs a special case of LSTMs? If so, give the expression of the GRU gates in term of LSTM's gates (o_t, i_t, f_t). If not, give a counter-example. Assume here the same input. [10 pts]

Answers:

1. For LSTM, the state update function is

$$c_t = f_1(W_1x_t + U_1h_{t-1} + b_1) * c_{t-1} + f_2(W_2x_t + U_2h_{t-1} + b_2) * f_3(W_3x_t + U_3h_{t-1} + b_3)$$

The main idea is to make forget(f1) gate small then the information of last state will be forgotten. Make input(f2) gate large then the information of input will be remembered.

There is one possible setting: $f_1 : 0 \quad f_2 : 1$

Then the update function becomes $c_t = f_3(W_3x_t + U_3h_{t-1} + b_3)$ and we can reconstruct x_t by c_t if f_3 and W_3 are reversible and $U_3 = 0$.

For GRU, the state update function is

$$h_t = (1 - f_1(W_1x_t + U_1h_{t-1} + b_1)) * f_2(W_2x_t + U_2(h_{t-1} * f_3(W_3x_t + U_3h_{t-1} + b_3)) + b_2) + f_1(W_1x_t + U_1h_{t-1} + b_1) * h$$

The main idea is to make forget(f1) gate small then the information of input can be remembered.

There is one possible setting: $f_1 : 0 \quad f_3 : 0$

Then the update function becomes $h_t = f_2(W_2x_t + b_2)$ and we can always reconstruct x_t by h_t if f_2 and W_2 are reversible.

2. For LSTM, the state update function is

$$c_t = f_1(W_1x_t + U_1h_{t-1} + b_1) * c_{t-1} + f_2(W_2x_t + U_2h_{t-1} + b_2) * f_3(W_3x_t + U_3h_{t-1} + b_3)$$

   The main idea is to make forget(f1) gate large then the information of last state can be keeped. Make input(f2) gate small then the information of input will be ignored.

There is one possible setting: $f_1 : 1 \quad f_2 : 0$

Then the update function becomes $c_t = c_{t-1}$ and we can always reconstruct c_{t-1} by c_t.

For GRU, the state update function is

$$h_t = (1 - f_1(W_1x_t + U_1h_{t-1} + b_1)) * f_2(W_2x_t + U_2(h_{t-1} * f_3(W_3x_t + U_3h_{t-1} + b_3)) + b_2) + f_1(W_1x_t + U_1h_{t-1} + b_1) * h$$

The main idea is to make forget(f1) gate large then the information of last state can be keeped.

There is one possible setting: $f_1 : 1$

Then the update function becomes $h_t = h_{t-1}$ and we can always reconstruct h_{t-1} by h_t.

3. No. The location of the input gate, or the corresponding reset gate is different. The LSTM unit computes the new memory content without any separate control of the amount of information flowing from the previous time step. On the other hand, the GRU controls the information flow from the previous activation when computing the new, candidate activation.

A counter-example: for GRU, set $f_1 = W_2 = b_2 = 0$, then $h_t = f_2(h_{t-1} * f_3(W_3x_t + U_3h_{t-1} + b_3))$. This mean we can control how much information in h_{t-1} we want to pass to h_t by different input x, which cannot be done by LSTM

1.2 Q3: Analyse the results (20 pts + 10 pts)

1. How does this compare with the results you obtained in the first assignment(DL1), when training a model that "sees" the entire image at once? Explain differences. [5 pts]

2. Let us take a look closer look at one of the trained models: say GRU (32). Plot the outputs of the RNN layer and hidden state over time. In particular, look at the first 3-5 time steps. Plot the input image along side. You can use `python plt.imshow(output_GRU_over_time)` for these, where `output_GRU_over_time.shape` is (T=28,hidden_units) dimensional. What do you observe? Show at least one pair of these plots to support your observation(s). [5 pts]

3. Now, look at the last 3-5 time steps. What do you observe? When is the classification decision made? To validate your answer to the second part of the question, provide the classification predictions for the last 5 time steps -- that is, pretend whatever the output of the GRU is at that time step is in fact the last output in the computation, and feed that into the classification mapping. [10 pts]

[Bonus] Let's looking inside the computation. Take one of the previous LSTM models (for simplicity pick one of the one-layer recurrence models) and track the status of the gates (o_t, i_t, f_t) over time. Note that if you used the provided RNNCell wrappers (BasicLSTMCell and co) in tensorflow, these will keep this information hidden, so you will need to implement your own version of this recurrence layer, or mirror the one in tensorflow, but now exposing these hidden variables. A bit of warning: this is not trivial and will require some thinking on the coding side, but it will also provide you with a more informative way of visulazing the inner computation. [10 pts]

Answers:

1. It depends on how to 'see' the entire image at once. If compared with CNN, the performance of RNN is worse. But it is better than other models in DL1 who see the entire image as a column. CNN utilizes the spatial location of pixels. RNN uses the oreder of the rows and other models haven't considered the spatial character.

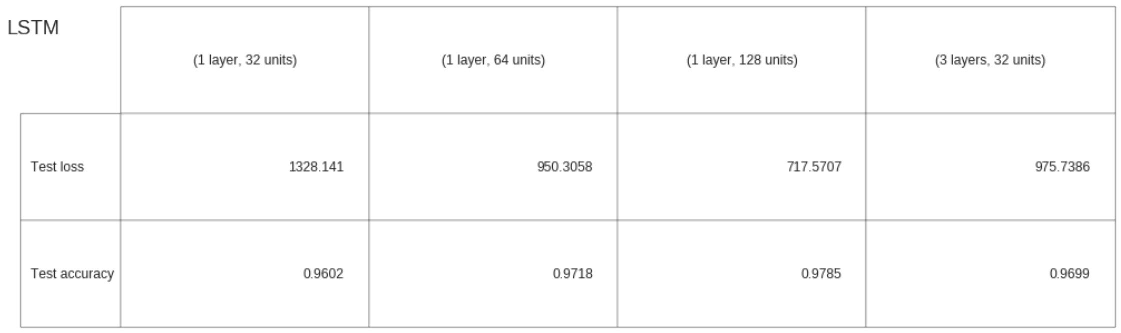
2. For the first 3-5 time steps, all the states are quite close and change slowly. Because the inputs for the first time steps are quite similar among different labels(like wihite background), so it is hard for the hidden states to behave differently for different labels. (The plots are shown in the "Analysis of results" cell.)

3. For the last 3-5 time steps, the hidden states are quite different among different labels. Because for these last time stpes, they have already been fed enough time steps' input and they already have some ideas which class the input sequence belongs to. So the hidden states behave quite differently for the later classification maaping. And we feed the last 1-5 time stpes' output into the classification mapping and get [0.959600,0.954700,0.919400,0.832600,0.723000] prediction accuracy on test data. We can see that the last 2 steps have similar accuracy. So the decision have been made after the last 2 steps. (The plots are shown in the "Analysis of results" cell.)

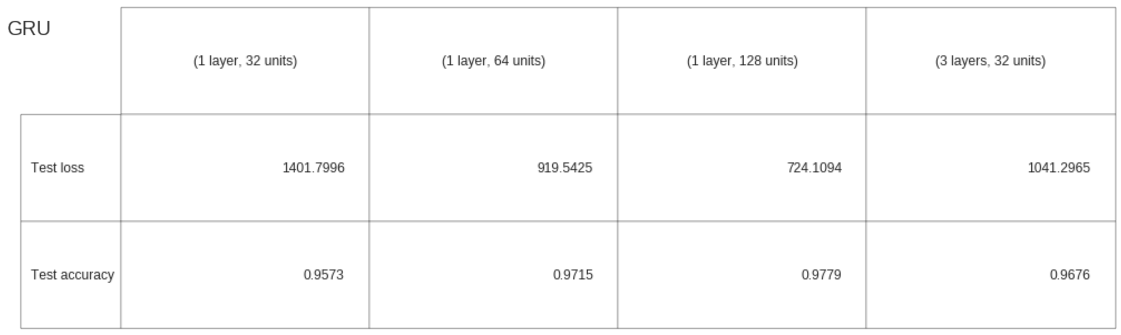
[Bonus] We can see that output gates have a similar trend as the outputs especially for the dark part, but ouput gates are darker in general. For the first time steps, all the demensions for different gates are quite close and change slowly which is the same as the outputs of LSTM. For the last time steps, some demensions for forget gates are large, which means these demensions need rememebr previous information which is very useful to the predictions. The input gates are quite acitevate for most time. (The plots are shown in the "Analysis of results" cell.)

2 Results

```
In [3]: plot_summary_table(experiment_results)
```



LSTM	(1 layer, 32 units)	(1 layer, 64 units)	(1 layer, 128 units)	(3 layers, 32 units)
Test loss	1328.141	950.3058	717.5707	975.7386
Test accuracy	0.9602	0.9718	0.9785	0.9699

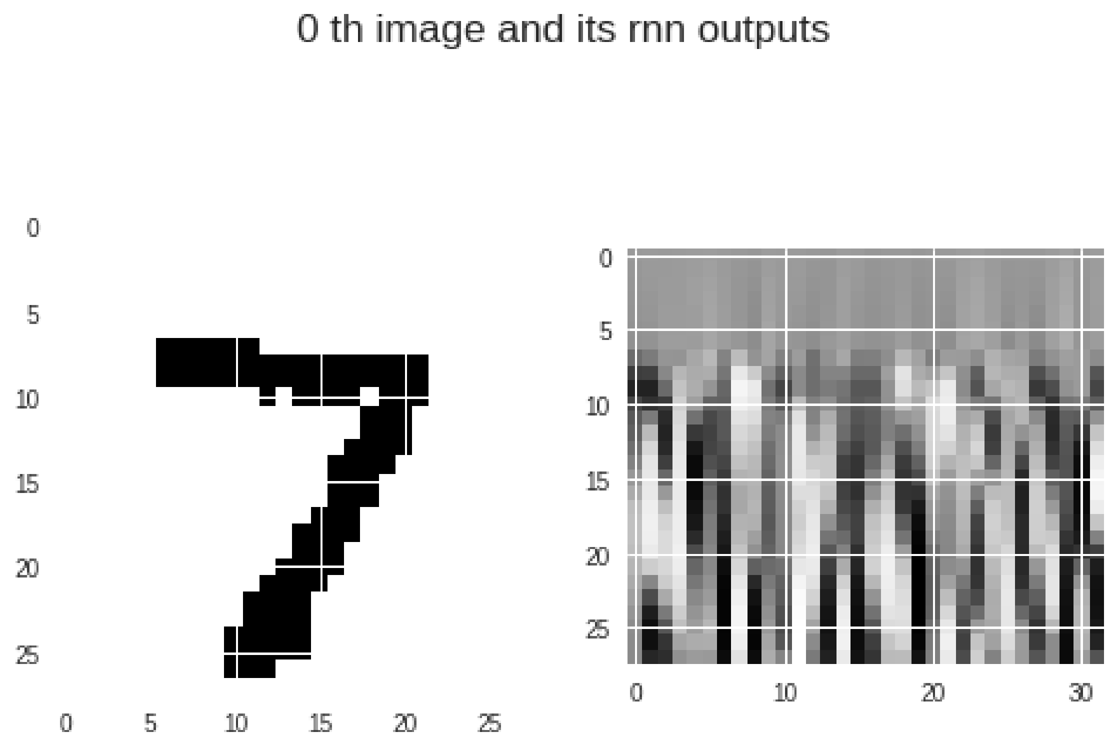


GRU	(1 layer, 32 units)	(1 layer, 64 units)	(1 layer, 128 units)	(3 layers, 32 units)
Test loss	1401.7996	919.5425	724.1094	1041.2965
Test accuracy	0.9573	0.9715	0.9779	0.9676

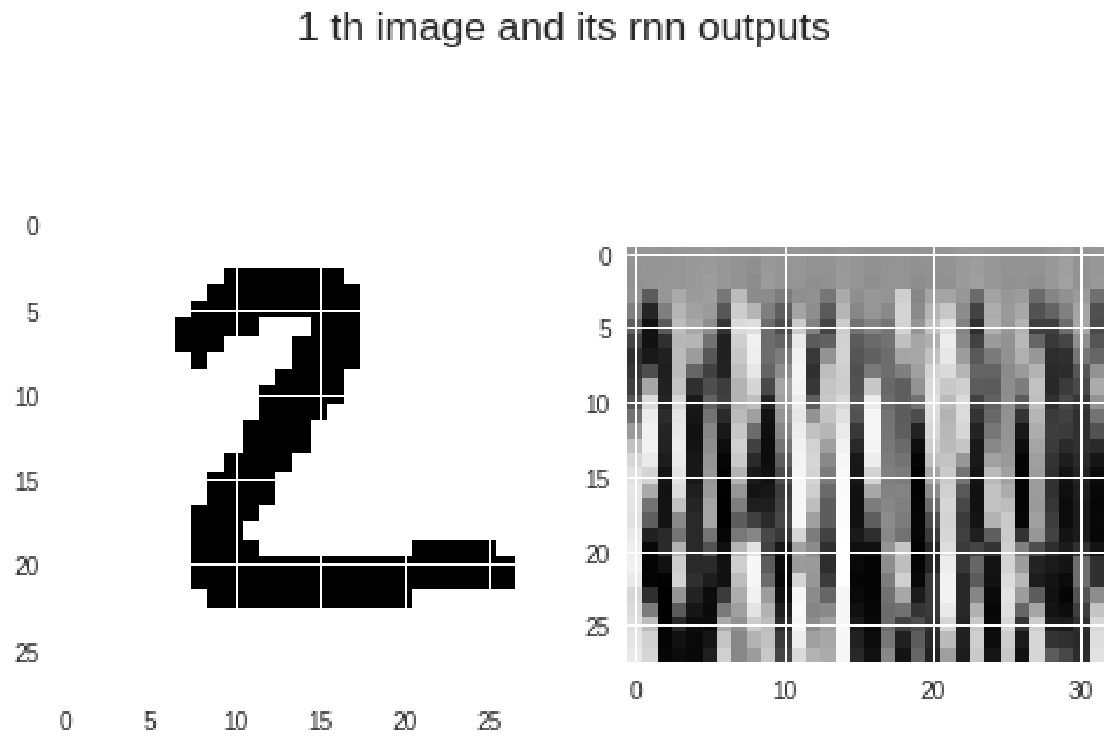
3 Analysis of results

```
In [5]: # Q3.2 & Q3.3
        # Look into the GRU computation over time
        binarized_input_image = np.reshape(binarize(eval_mnist.test.images),(-1,28,28))

        # plot one sample over time
        for i in range(2):
        # for i in range(output_GRU_over_time.shape[0]):
          fig, axes = plt.subplots(1, 2)
          filename = str(i) + ' th image and its rnn outputs'
          fig.suptitle(filename,fontsize="x-large")
          axes[0].imshow(binarized_input_image[i,:,:])
          axes[1].imshow(output_GRU_over_time[i,:,:])
          plt.show()
          name = raw_input("Press Enter to continue...")
```

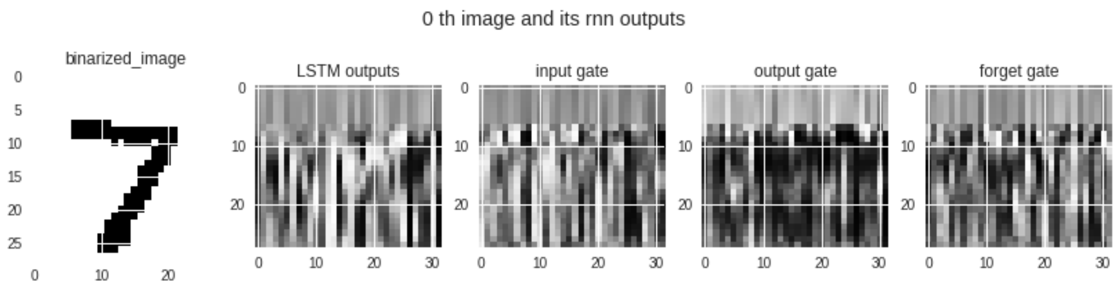


```
Press Enter to continue...
```

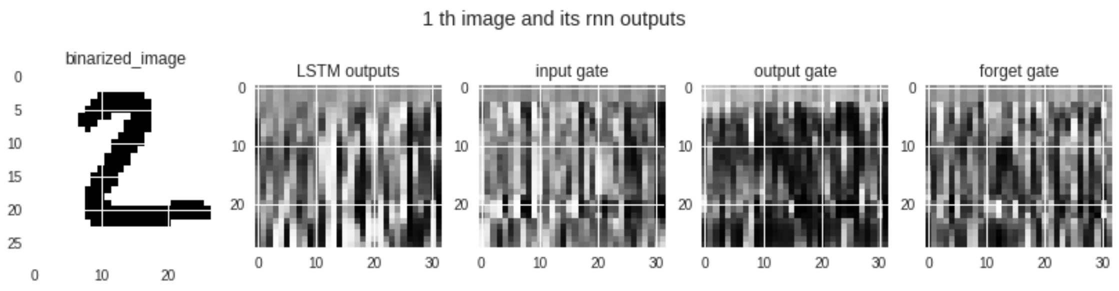


```
Press Enter to continue...
```

```
In [10]: # Bonus: plot gates
         title = ['LSTM outputs','input gate','output gate','forget gate']
         # Look into the GRU computation over time
         binarized_input_image = np.reshape(binarize(eval_mnist.test.images),(-1,28,28))
         # plot one sample over time
         for i in range(2):
         # for i in range(binarized_input_image.shape[0]):
           fig, axes = plt.subplots(1, 5,figsize=(14,16))
           filename = str(i) + ' th image and its rnn outputs'
           st = fig.suptitle(filename)
           axes[0].imshow(binarized_input_image[i,:,:])
           axes[0].set_title('binarized_image')
           for j in range(4):
             axes[j+1].imshow(values[j][:,i,:])
             axes[j+1].set_title(title[j])
           st.set_y(0.6)
           plt.show()
           name = raw_input("Press Enter to continue...")
```



```
Press Enter to continue...
```



```
Press Enter to continue...
```