Assignment 3

1 LSTM task

The task is to implement an LSTM and apply it to recognizing digits from MNIST dataset. This task extends the exercise that we did at the lab sessions.

2 Data

Examples in MNIST datasets are 28x28 images, how do we get any RNN to work on them? We can make the RNN process i-th row at i-th time step. This is exactly what we have done in the lab session and what is required in this task. Here is example code that iterates through MNIST in this fashion:

```
from tensorflow.examples.tutorials.mnist import input_data
mb_size = 64
steps_n = 28
input_n = 784 / steps_n
mnist = input_data.read_data_sets("/tmp/data/", one_hot=True)
while True:
    batch_x, batch_y = mnist.train.next_batch(mb_size)
    batch_x = batch_x.reshape((mb_size, steps_n, input_n))
```

3 Basic implementation(Part 1)

The first part requires you to implement LSTM and apply it to MNIST dataset. The whole classification network should consist of two parts. The first is LSTM processing batches of pixels. You should take the last hidden state produced by LSTM and process it further with some neural network that will produce final distribution on classes.

- Your implementation should score at least 97% of accuracy on the test set.
- The training should work on minibatches.
- Use MNIST dataset from Tensorflow. The validation set should have size 5000 examples.
- You should do validation on the validation set every epoch.
- In this basic version the LSTM can have only one layer.
- Train the network using cross entropy loss.

You are not allowed to use RNN API from Tensorflow or copy any code from online sources.

4 Additional features(Part 2)

- Make your network work on inputs of different lengths. To generate sensible data of different lengths we will do some data augmentation. Implement augmentation of input images using cropping. Sample the width and the height of the cropped image uniformly from the interval $[24, 28]$. That way each cropped image will have size from 576 to 784. As in the basic version we will supply 28 pixels from the image at each time step. The size might not be divisible by 28, so you might have to add some padding at the last time step. As the input to the second part of the network you should choose the hidden state at the last time step. The number of time steps might be different for different examples(in the basic version, the number of time steps was always 28)
- Implement deep LSTM(many layers). It should be easy to change the number of layers and the size of hidden state in each of those layers.

5 Logs

You should save the logs from your run, containing train and validation error for subsequent epochs as well as timestamps. Also, you should store a checkpoint to show the validation error during code inspection.

6 Scoring

Part 1 and Part 2 are worth 50% each.

7 Additional solution features (nonobligatory)

If you want, you can additionally implement:

- add tensorboard visualization of training process(i.e. training loss, accuracy, validation loss, accuracy)
- implement deep bidirectional LSTM
- implement some form of gradient clipping(you might also show information about gradient magnitude to the the tensorboard visualization)
- implement dropout in the form described by this paper: `https://arxiv.org/abs/1409.2329`

8 Note

Implementing features from the nonobligatory list might help you getting full score even in case of minor drawbacks.

9 Deadline

You should submit your solution by email by 23:59 on 13.06.2017 to `maciej.klimek@deepsense.io` with email title "[GSN] Assignment 3". You can use the same email to ask question about this task in case something is not clear. Your code will be inspected during lab session on 14.06.