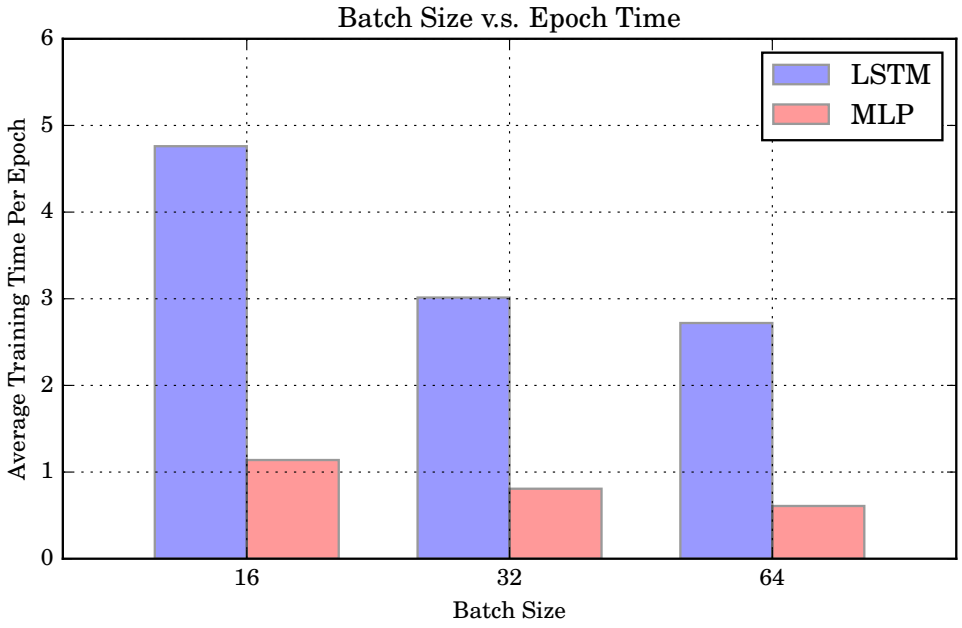
10-605 Machine Learning with Large Datasets - Fall 2016
Assignment 5 Solutions*

1.

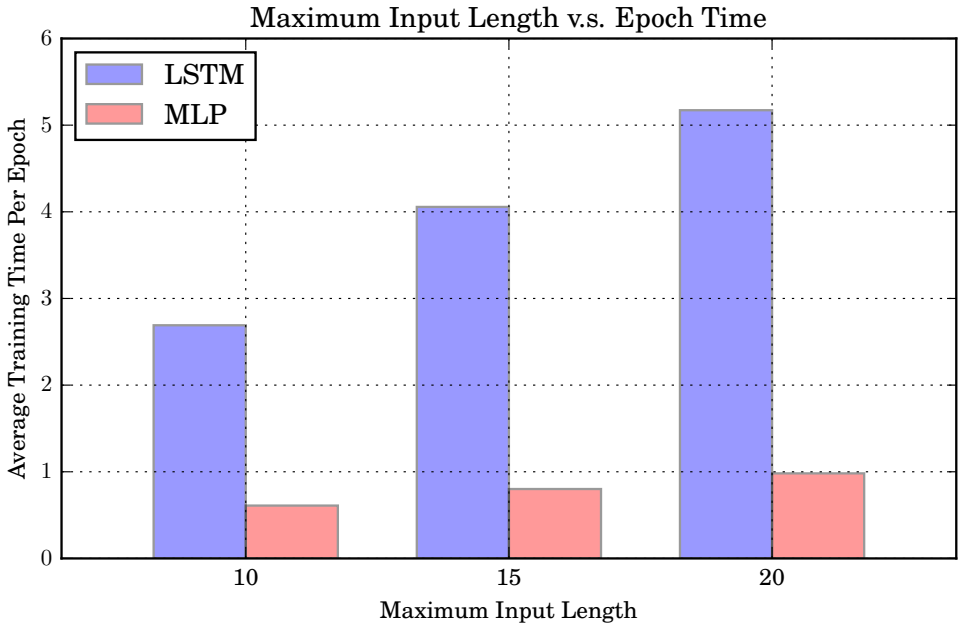


Form the plot we can see that, for both MLP and LSTM, as we increase the batch size, the average training time per epoch starts decreasing.
For large batch size, we are loading more examples to the memory at one batch. Because of advanced techniques like vectorization and tiling the computational costs among different batch sizes are relatively close to each other, as long as the examples can fit into memory. Yet for the fixed training dataset, a large batch size would lead us to a small number of batches in one epoch. Therefore, increasing the batch size could decrease the average training time per epoch.
Note it is important to mention the point about efficient implementation of numpy operations here. Even though we do fewer steps, the increased size could've negated the effect of the fewer operations. Say you wrote a simple triple for loop for matrix multiplication instead of the numpy ops the time to do one pass over the network would also go up substantially (this is because of matrix multiplication is bandwidth bound). But because of the way matrix multiplication is implemented in numpy the amount of time that it takes to multiply size $|2S|$ is not significantly

*Thanks to Bowen Deng for sharing his report for the solutions

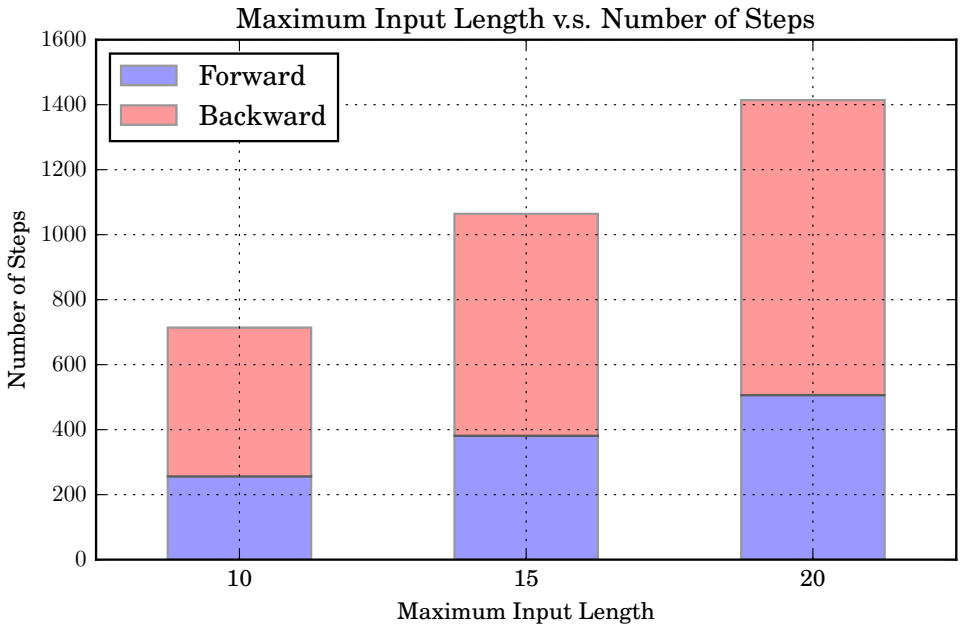
larger than multiplying size $|S|$ matrices (This is because of the increased computational workload and highly optimized cache aware implementation using tiling and vectorization).

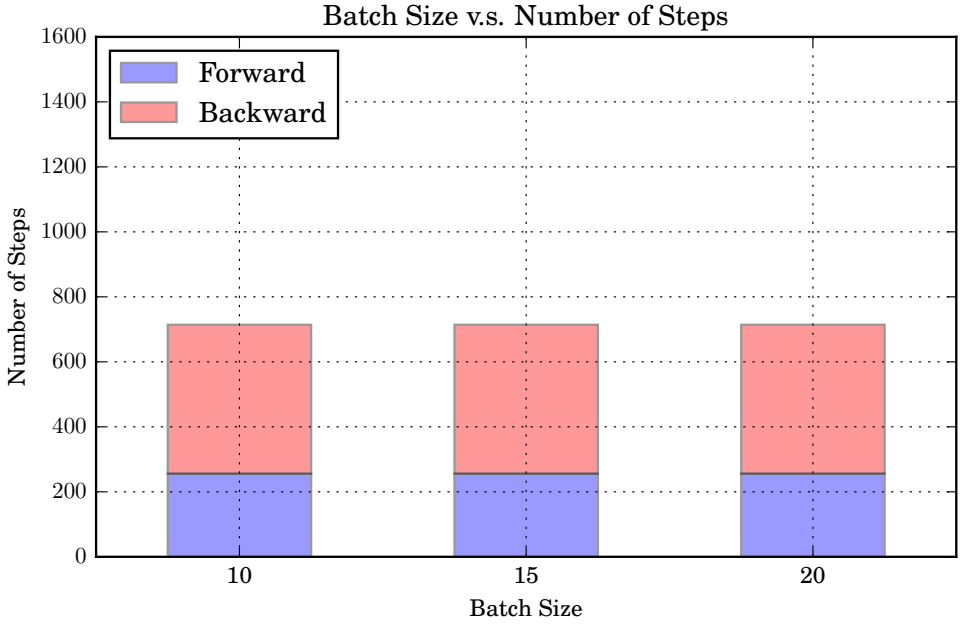
2.



Form the plot we can see that, for both MLP and LSTM, as we increase the maximum input length M, the average training time per epoch starts increasing.
Although the average training time per epoch is increased for both architectures, the reasons are slightly different. For the MLP architecture, by increasing M, we are explicitly adding more parameters to $W^{(1)}$. And for similar reasons to answer 1 this does not significantly increase the time. For the LSTM architecture, increasing M didn't explicitly adding any parameters to our models. However, it added more computation steps to the architecture, and such change increased the number of intermediate variables of our models. Thus, for both architectures, as we increase M, we are increasing the computational complexity of the models. So the average training time per epoch increased.

3.





From the plots we can see that, for LSTM, as we increase the maximum input length M, the number of forward and backward steps are increased. However, the changes of batch size have no effect on the number of steps. The reason is the same as *Question 1* and *Question 2*. Because of vectorization, the batch size only changed the size of the matrices we computed for each batch, the total number of matrix operations didn't change. But the maximum input length M determined how many intermediate variables needed to compute for each batch, thus, increasing M caused the increasing of computation steps.

4.

```
S = exp(Q.dot(D.T))
A = S / row-sum(S)
```

Where the last step uses broadcasting.

5.

Let $z = \sum_{i=1}^{1000} x_i w_i$. Thus,

$$\frac{df}{dz} = \frac{tanh(z)}{dz} = 1 - tanh^2(z) \in (0, 1)$$

To avoid the gradients being too small, say, we want

$$E[z] = 1$$

Since,

$$1 - tanh^2(1) \approx 0.41997434161$$

Assume, x_i, w_i for all $i = 1, ...1000$ are independent of each other. Therefore,

$$\begin{aligned} E[z] &= E[\sum_{i=1}^{1000} x_i w_i] \\ &= \sum_{i=1}^{1000} (E[x_i w_i]) \\ &= \sum_{i=1}^{1000} (E[x_i] E[w_i]) \\ &= \sum_{i=1}^{1000} \left(E[x_i] \frac{a}{2} \right) \\ &= \frac{a}{2} \sum_{i=1}^{1000} E[x_i] \end{aligned}$$

One-hot

$$\begin{aligned} E[z] &= \frac{a}{2} E[x_{i0}] \\ &= \frac{a}{2} \end{aligned}$$

Therefore, we choose

$$a = 2$$

Small

For the small case that $x_i \in [0, 1]$ for all i, let's assume $E[x_i] = 0.5$ for all i. Thus,

$$\begin{aligned} E[z] &= \frac{a}{2} \sum_{i=1}^{1000} E[x_i] \\ &= \frac{a}{2} \sum_{i=1}^{1000} 0.5 \\ &= \frac{a}{2} * 500 \\ &= 250a \end{aligned}$$

Therefore, we choose

$$a = \frac{1}{250} = 0.004$$

Big

For the big case that $x_i \in [1000, 1001]$ for all i, let's assume $E[x_i] = 1000.5$ for all i. Thus,

$$\begin{aligned} E[z] &= \frac{a}{2} \sum_{i=1}^{1000} E[x_i] \\ &= \frac{a}{2} \sum_{i=1}^{1000} 1000.5 \\ &= \frac{a}{2} * 1000500 \\ &= 500250a \end{aligned}$$

Therefore, we choose

$$a = \frac{1}{500250} \approx 0.000001999$$