Qualitatively

Advantages/Disadvantages:

MLP

Advantage: Suitable for classification prediction problems where inputs are assigned a class or label, therefor have been proven to be effective in one of the most common problems, which is image recognition.

Disadvantage: On the other hand, there isn't a way to input sequences data with Importance of order and time.

RNN

Advantage: Were designed to work with sequence prediction problems which are a considerable part of the problems network solves, such as Text data, Speech data and Generative models.

Disadvantage: It became very inefficient to train a RNN on large data, due to the exploding and vanish gradients, which mean that the back-propagation effect can't go far enough to reach the early stage of layers.

LSTM

Advantage: Overcomes the problems of training a recurrent network and in turn has been used on a wide range of applications.

Disadvantage: The ability to have a network that can process data from early stage of layers requires more complex structure for each cell.

Time Complexity:

The time complexity reflected as the number of calculations we need to do in the forward function.
We will mark it in the following way: $in\_features = n$, $out\_features = h$, $first\ dimension\ of\ the\ input = d$, and ignore the bias as in our implementation. In addition, we assume the tanh(x) and sigmoid(x) functions are running in $T(x), S(x)$ respectively, and matrix multiplication of $matrix(n,m) * matrix(m,p)$ is $O(nmp)$.

MLP: sigmoid value of $matrix(d,n) * matrix(n,h)$ => $\boldsymbol{O(S(dnh))}$

RNN: tanh value of $matrix(d,n) * matrix(n,h) + matrix(d,h) * matrix(h,h)$
=> $\boldsymbol{O(T(dnh + dh^2))}$

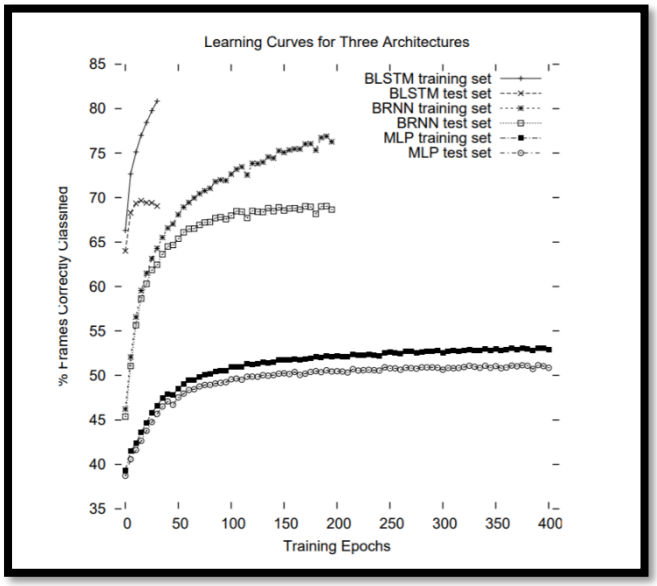
LSTM:
i: sigmoid value of $matrix(d,n) * matrix(n,h) + matrix(d,h) * matrix(h,h)$
=> $O\big(S(dnh + dh^2)\big)$
f: sigmoid value of $matrix(d,n) * matrix(n,h) + matrix(d,h) * matrix(h,h)$
=> $O\big(S(dnh + dh^2)\big)$
g: tanh value of $matrix(d,n) * matrix(n,h) + matrix(d,h) * matrix(h,h)$ =>
$O\big(T(dnh + dh^2)\big)$
o: sigmoid value of $matrix(d,n) * matrix(n,h) + matrix(d,h) * matrix(h,h)$
=> $O\big(S(dnh + dh^2)\big)$
c': $O(dh)$, h': $O(T(dh) + dh)$
=> $\boldsymbol{O(S(dnh + dh^2) + T(dnh + dh^2))}$

Quantitatively

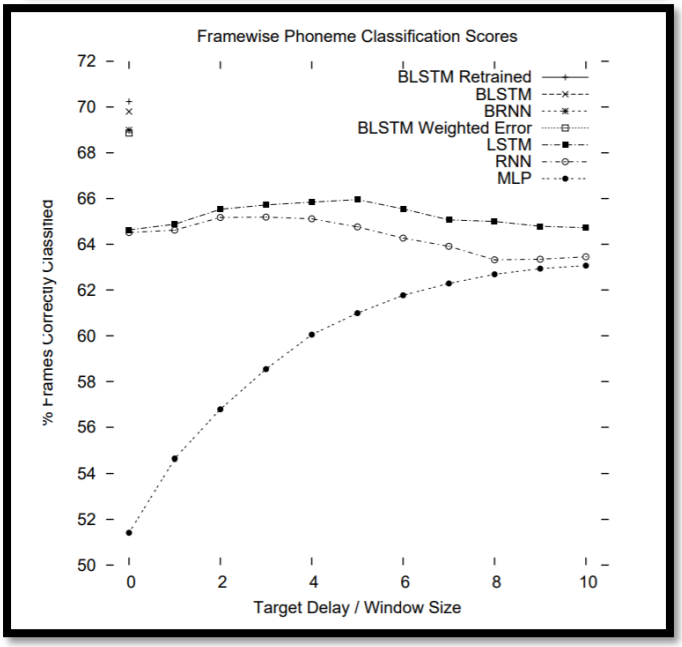
<u>Training results on a simple task</u>

The information was taken from chap 5 at: https://www.cs.toronto.edu/~graves/preprint.pdf

Error calculation method	Backpropagation
Initial weights	Random distribution with range $[-0.1, 0.1]$



As can be seen, LSTM was much faster to converge than RNN or MLP.



LSTM was little more accurate, the difference is not big maybe because long time dependencies were not tested at this experiment.