As this has happened, we’ve seen a growing number of attempts to augment RNNs with new properties. Four directions stand out as particularly exciting:

Neural Turing Machines

Attentional Interfaces

Adaptive Computation Time

Neural Programmers

Individually, these techniques are all potent extensions of RNNs, but the really striking thing is that they can be combined, and seem to just be points in a broader space. Further, they all rely on the same underlying trick---something called attention—to work.

Our guess is that these “augmented RNNs” will have an important role to play in extending deep learning’s capabilities over the coming years.

Neural Turing Machines combine a RNN with an external memory bank. Since vectors are the natural language of neural networks, the memory is an array of vectors:

Adaptive Computation Time

Standard RNNs do the same amount of computation for each time step. This seems unintuitive. Surely, one should think more when things are hard? It also limits RNNs to doing O(n) operations for a list of length n.

Adaptive Computation Time is a way for RNNs to do different amounts of computation each step. The big picture idea is simple: allow the RNN to do multiple steps of computation for each time step.

[图片]

There are a few more details, which were left out in the previous diagram. Here’s a complete diagram of a time step with three computation steps.

That’s a bit complicated, so let’s work through it step by step. At a high-level, we’re still running the RNN and outputting a weighted combination of the states:

[图片]

We have a total budget for the halting weights of 1, so we track that budget along the top. When it gets to less than epsilon, we stop.

When we stop, might have some left over halting budget because we stop when it gets to less than epsilon. What should we do with it? Technically, it’s being given to future steps but we don’t want to compute those, so we attribute it to the last step.

When training Adaptive Computation Time models, one adds a “ponder cost” term to the cost function. This penalizes the model for amount of computation it uses. The bigger you make this term, the more it will trade-off performance for lowering compute time.

Adaptive Computation Time is a very new idea, but we believe that it, along with similar ideas, will be very important.