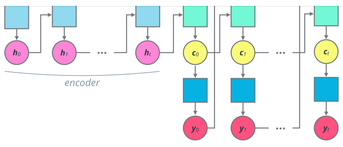


Encoder-Decoder: Unrolled

The first is called an Encoder, which accepts the source sentence, one word at a time, and captures its overall meaning in a single vector. This is simply the state vector at the last time step. Note that the encoder network is not used to produce any outputs.

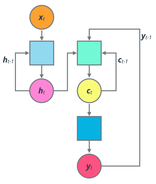
The second network is called a Decoder, which then interprets the final sentence vector and expands it into the corresponding sentence in the target language, again one word at a time.

The first time step for the decoder network is special. It is fed in the final sentence vector from the encoder h_t, and given a sentinel input to kickstart the process. The recurrent portion of the network produces a state vector c_0, and with that the fully-connected portion produces the first output word in the target language, y_0.

At each subsequent time step t, the decoder network uses its own previous state c_{t-1} as well as its own previous output y_{t-1} in order to produce the current output, y_t.

This process is typically continued for a fixed number of iterations, with the idea that the network will start producing special padding symbols after all meaningful words have been produced. Alternatively, the network may learn to output a stop symbol, such as a period (.), to indicate that the translation is complete.

If we roll back the time steps, we can see what the overall architecture looks like.



Encoder-Decoder: Schematic

This encoder-decoder design very popular for several sequence-to-sequence tasks, not just Machine Translation.

Now, there are several variations of this design that can be used to enhance the performance of the network.

- One option is to use different kinds of recurrent neural network units, such as LSTMs, GRUs etc. instead of vanilla RNN units. That allows the network to better analyze the input sequence, at the cost of additional model complexity.
- Another dimension to explore is how many recurrent layers to use. Each layer effectively incorporates information from the input sequence, producing a compact state vector at each time step. Additional layers can essentially incorporate information across these state vectors.
- Other more innovative approaches include adding in a backward encoder (bidirectional encoder-decoder model), feeding in the sentence vector to each time step of the decoder (attention mechanism), etc.

Feel free to experiment with these different approaches to see what architecture works best for your task. Keep in mind that these mechanisms typically add to the model complexity, which means you need more data and time to train the additional parameters.