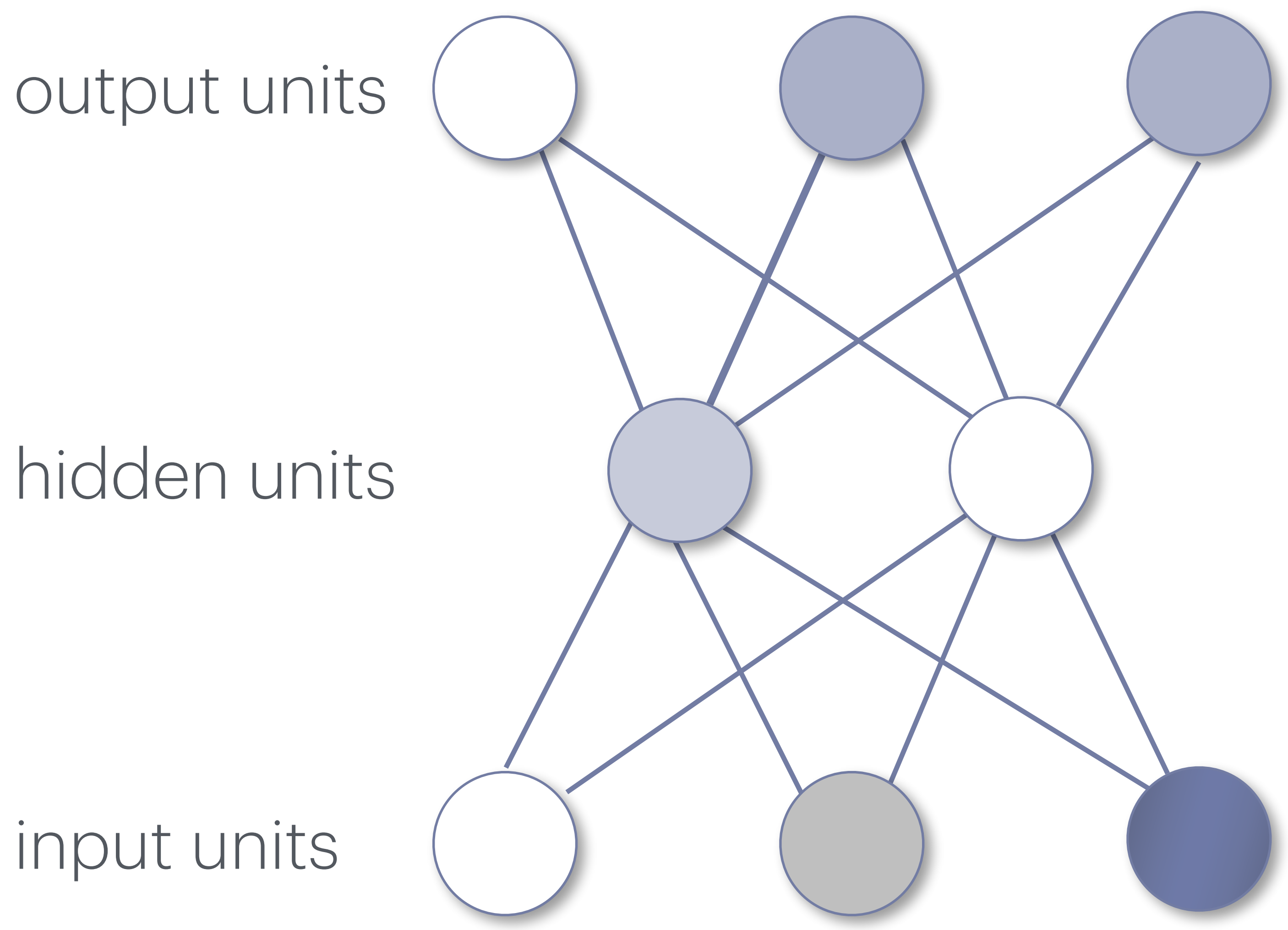
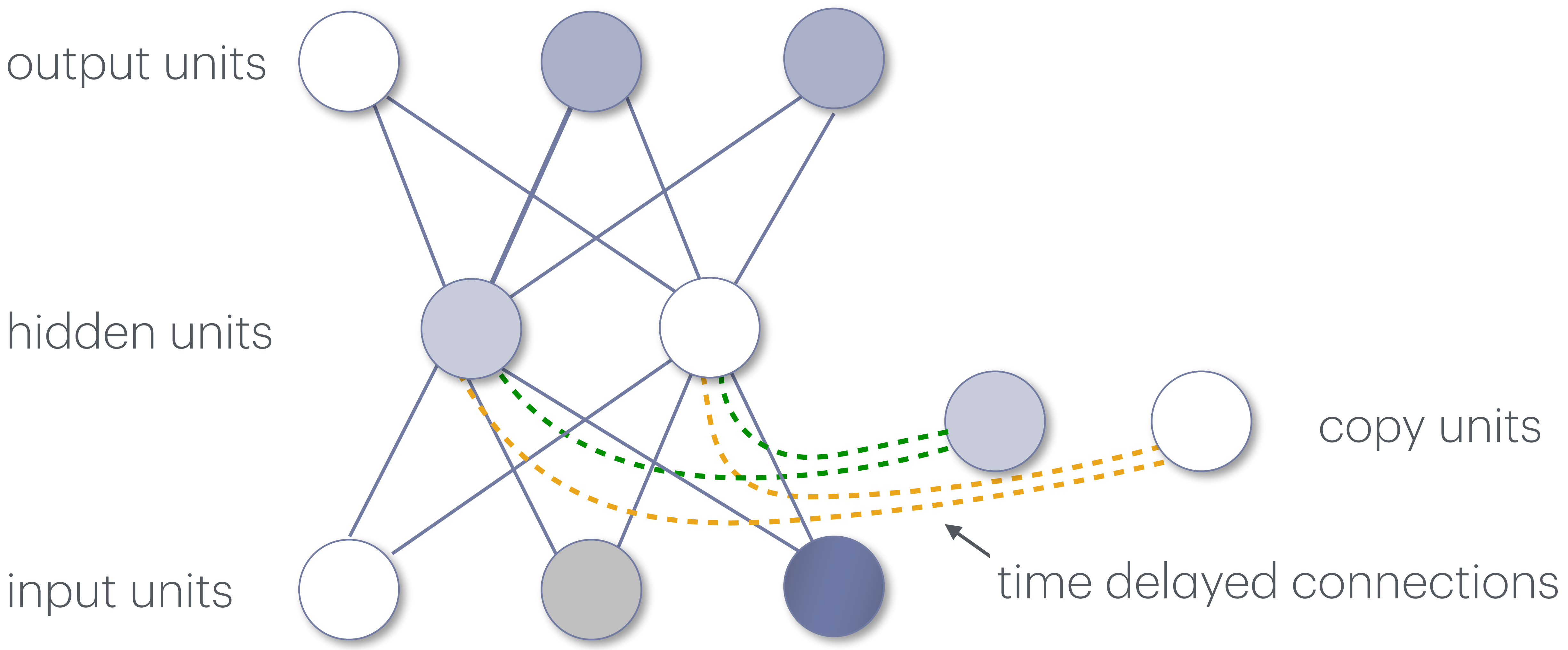
A simplistic account of GRUs

with roughly the same conceptual function (but without showing the implementation)

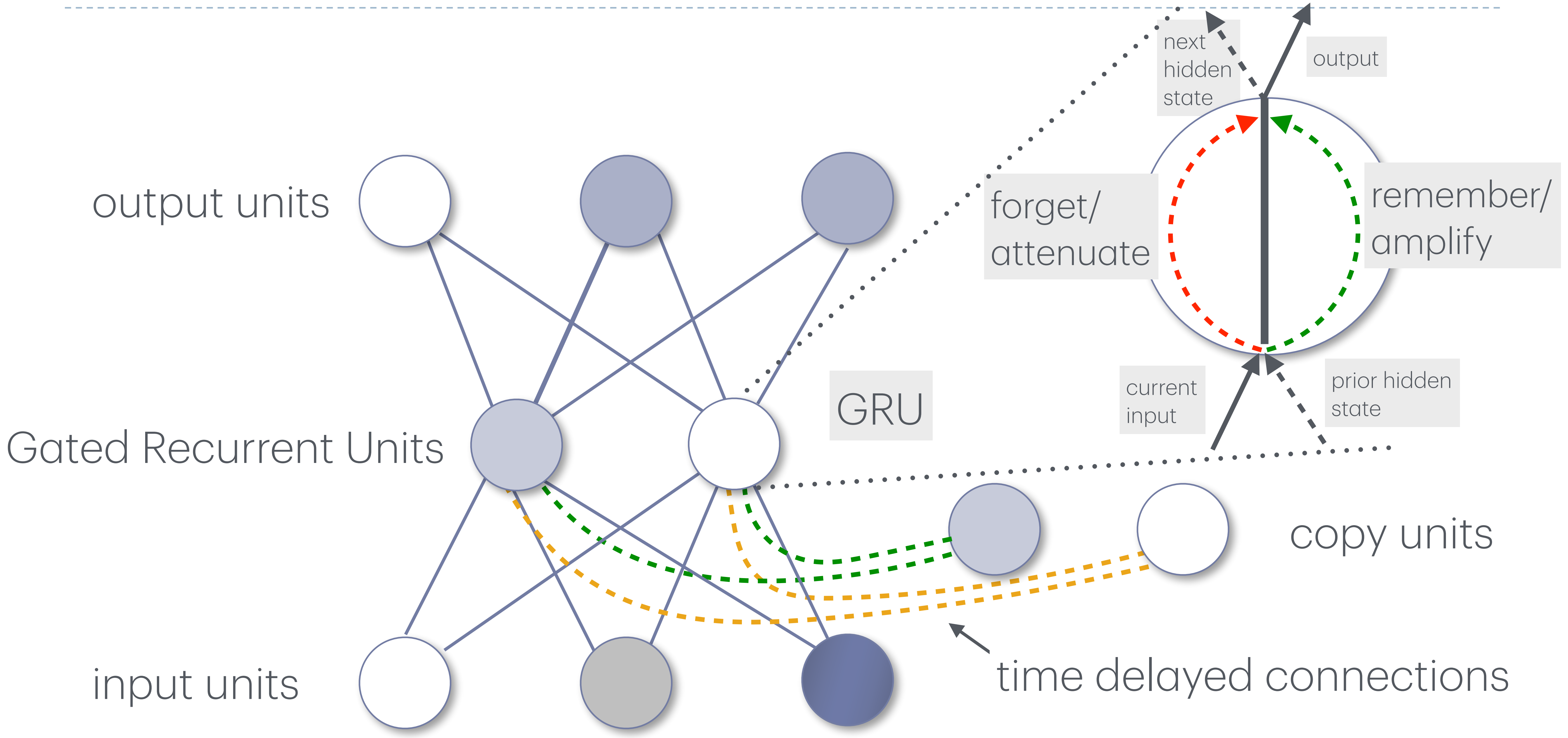
simple feedforward network



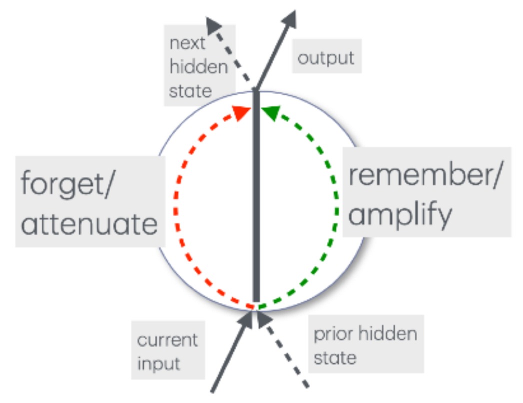
simple recurrent network



simple recurrent network with GRUs

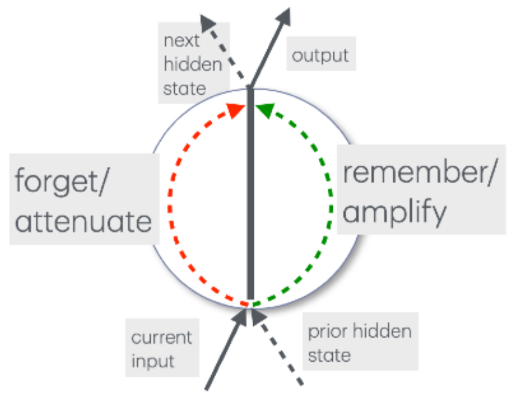


What GRUs do



The GRU can essentially pass through the prior hidden state more or less “as is” (“updating”), and combine that with the current input, or it can do a “reset” and essentially not pass through any of the prior hidden state. Of course, whether it updates or resets is just down to weights, that can vary between e.g. 0 and 1. So all options in-between are possible too (e.g. amplifying to a degree - which means the context is more heavily weighted relative to the input, or attenuating to a degree, which means it’s less heavily weighted). For reasons I don’t quite understand, but am willing to leave to the engineers and mathematicians, the up-regulating or down-regulating (to use that terminology) of the prior hidden state (which is just a vector) are calculated separately (the red and the green lines in the diagram), and the up-regulated and down-regulated vectors are then combined. So it’s like taking the hidden vector at the previous time-step, multiplying it by a multiplication factor that reflects how much it should be down-regulated, multiplying another copy by a factor that reflects how much it should be up-regulated, and then adding them together. Why it’s not just a single regulation (up or down) beats me! But I don’t really care… all I need to know is that some GRUs will learn to amplify (up-regulate) the context (the prior hidden state) and others will learn to attenuate (down-regulate) the context.

Why GRUs are helpful



Some GRUs will become sensitive to long distance dependencies, allowing the network to correctly predict that in “the boy fell down and hurt his…” the pronoun “his” inherits characteristics of “the boy”. Other GRUs will become sensitive to short distance dependencies, allowing the network to correctly predict that in “the …” the thing after “the” will be a noun or an adjective (or other things with lesser probabilities)

How do they learn the appropriate amplification/ attenuation weights?

Imagine that the layer of GRUs is actually a layer of weights (the red, black, and green shown inside the GRU). One can just use back propagation through those weights using pretty much the standard backdrop algorithm (establishing how much error each weight is contributing, and adjusting accordingly). Over many iterations, the weights will converge on something useful. Think of it as “intelligent trial-and-error”!!