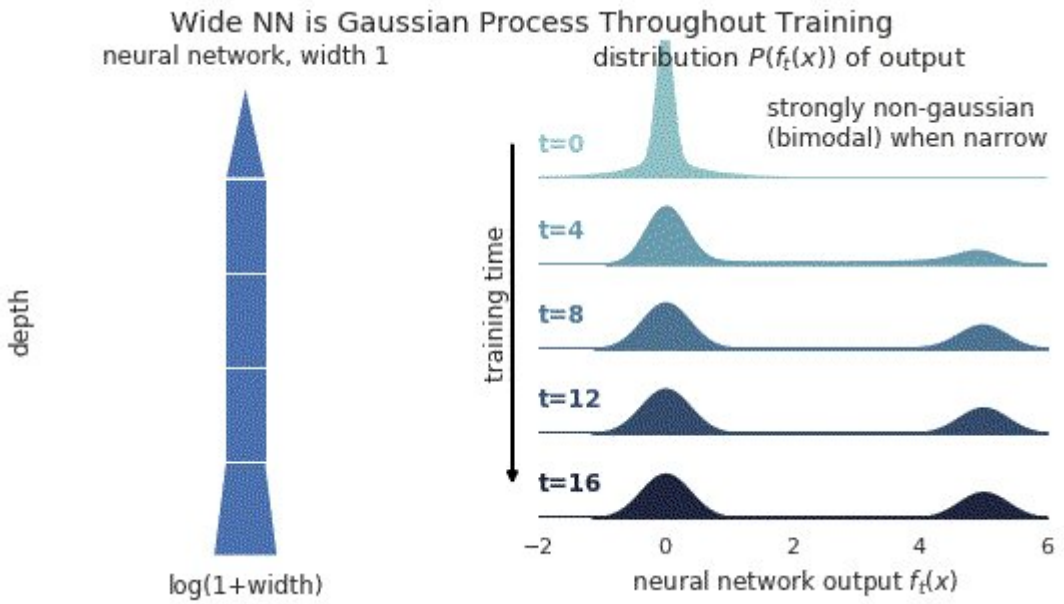
Twitter Thread by Greg Yang

Greg Yang
@TheGregYang

1/ A ∞-wide NN of *any architecture* is a Gaussian process (GP) at init. The NN in fact evolves linearly in function space under SGD, so is a GP at *any time* during training. https://t.co/v1b6kndqCk With Tensor Programs, we can calculate this time-evolving GP w/o trainin any NN



2/ In this gif, narrow relu networks have high probability of initializing near the 0 function (because of relu) and getting stuck. This causes the function distribution to become multi-modal over time. However, for wide relu networks this is not an issue.

3/ This time-evolving GP depends on two kernels: the kernel describing the GP at init, and the kernel describing the linear evolution of this GP. The former is the NNGP kernel, and the latter is the Neural Tangent Kernel (NTK).

4/ Once we have these two kernels, we can derive the GP mean and covariance at any time t via straightforward linear algebra.

NN distributed like $\mathcal{N}(\mu_t, \Sigma_t)$ at time t in training, where

$$
\begin{aligned}
\mu_t(a) &= \Theta_{aX}\Theta^{(t)}Y \\
\Sigma_t(a,b) &= K_{ab} - 2\Theta_{aX}\Theta^{(t)}K_{Xb} \\
&\quad - 2\Theta_{bX}\Theta^{(t)}K_{Xa} \\
&\quad + \Theta_{aX}\Theta^{(t)}K\Theta^{(t)}\Theta_{Xb} \\
\text{where} \quad & \Theta^{(t)} = \Theta^{-1}(I - e^{-\eta\Theta t}) \\
& \Theta = \text{NTK},\ K = \text{NNGP} \\
& X = \text{train set},\ Y = \text{labels}
\end{aligned}
$$

5/ So it remains to calculate the NNGP kernel and NT kernel for any given architecture. The first is described in https://t.co/cFWfNC5ALC and in this thread
https://t.co/6RO7VZDQNZ

6/ The NTK for any architecture is calculated in https://t.co/v1b6kndqCk and in this thread
https://t.co/OOoOMdPOsR