Glassy Dynamics in Deep Recurrent Networks

Quantum Connections, lecture IV
12 June 2024

This work in collaboration with Joanna Tyrcha, Matematiska Institutionen, SU

Motivation and Outline

- Large networks exhibit phase transitions in hyperparameter space
- Knowing the phase diagram is essential for applications (M Tegmark)
- Here: the simplest kind of transition: learnable/unlearnable
    - deep recurrent networks
    - locating the transition
    - critical slowing down of learning near the transition
    - aging

Data

- Expert advice:

  ”Das is doch einmal etwas, aus dem sich etwas lernen lässt.” (Mozart)

  “Study Bach. There you will find everything.” (Brahms)

- Here, we use Bach’s chorales

  413 (?) traditional German hymns arranged for 4 voices (Soprano Alto Tenor Bass)

- Data set from Joshua Bengio’s group, 2012

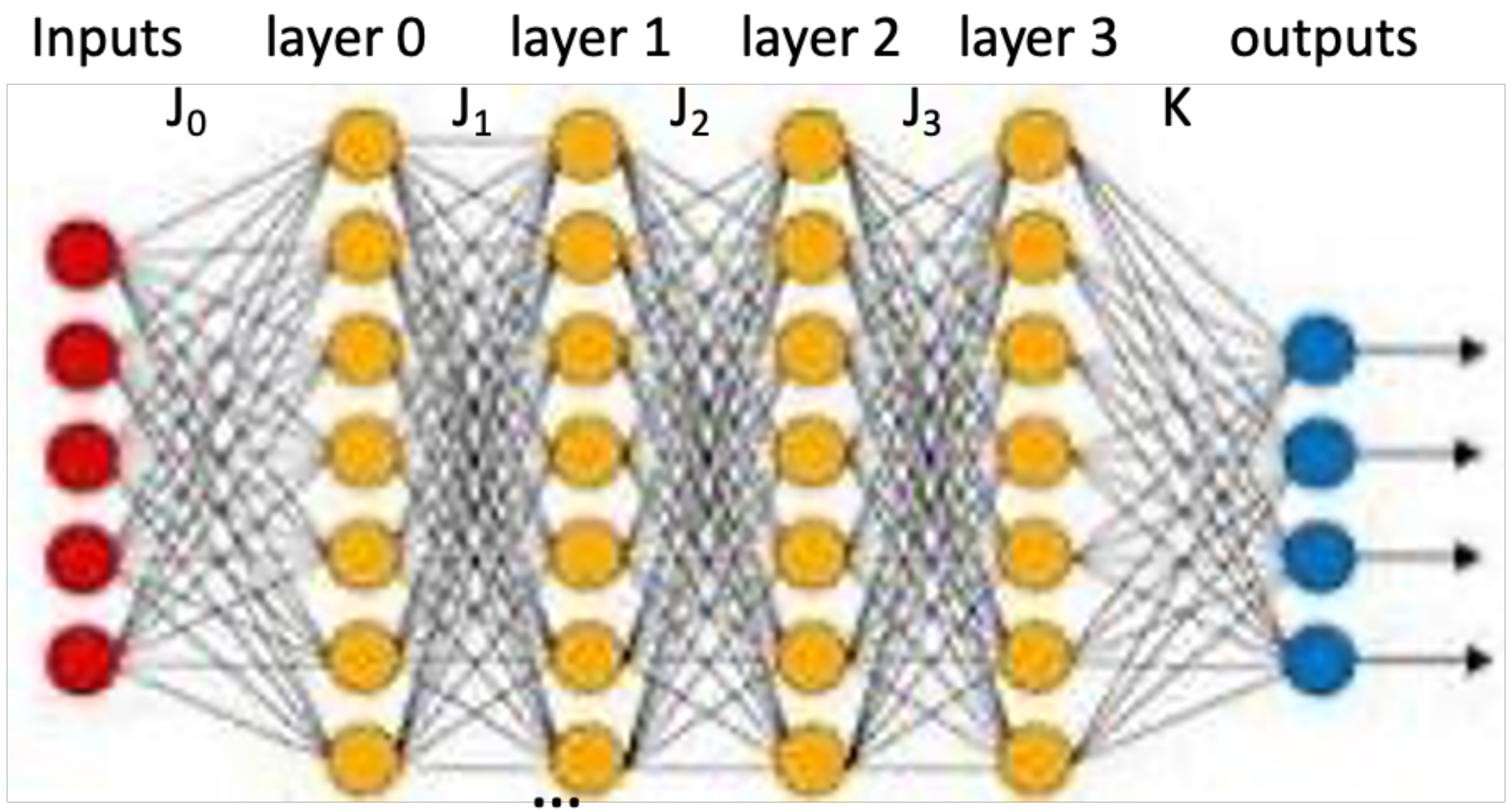
  382 chorales, quantised to quarter notes (crochets), transposed to C major or A minor

  Example: #16 (from St Matthew Passion, also used by Paul Simon in “American Tune”)

- Train networks to predict the next chord

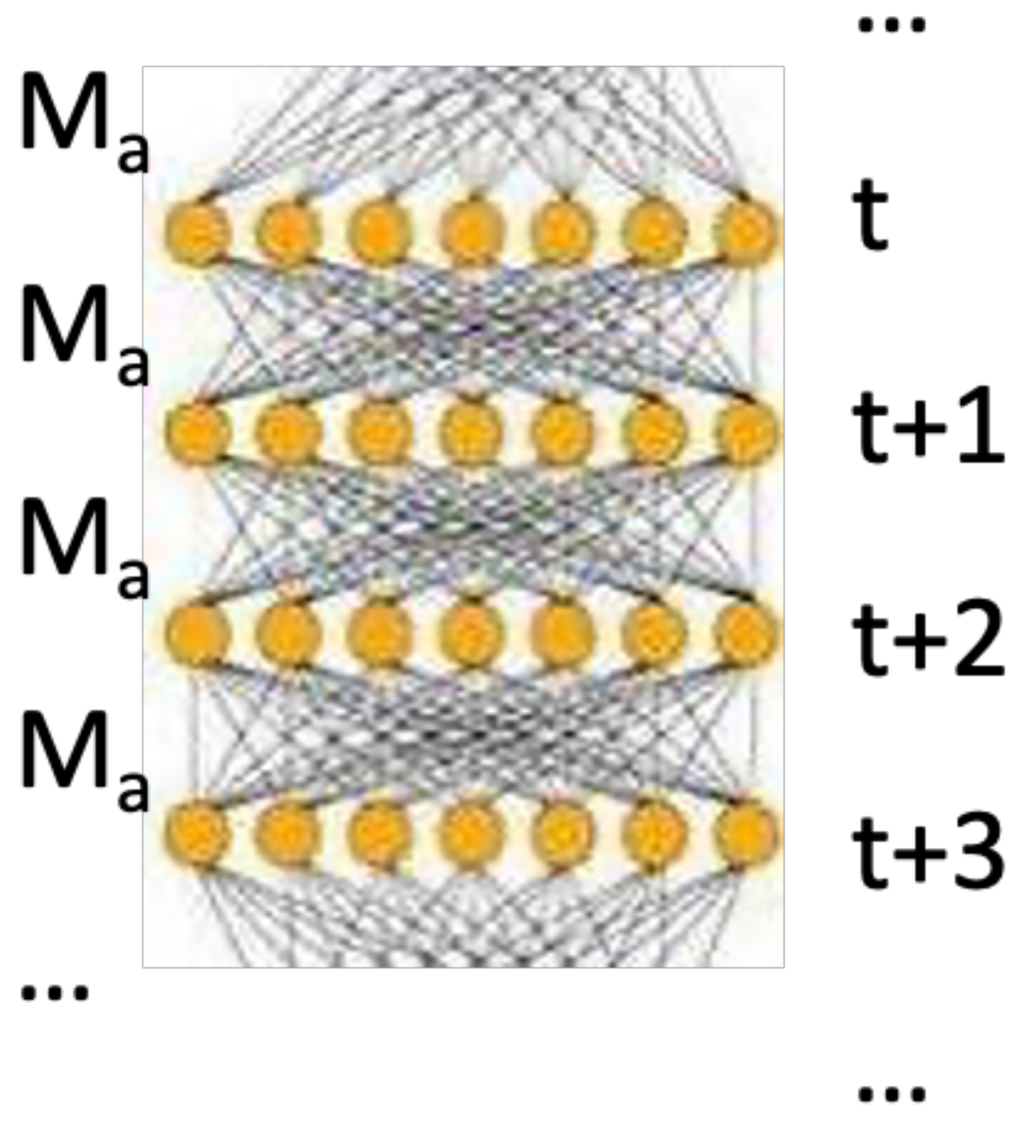
Model: deep recurrent nets

- Layered networks:
  - fully recurrent within each layer
  - feedforward from layer to layer
- Inputs: at each time step, a 54-d vector
  - components represent notes of different pitches
  - value = 1 for the notes that are sung/played, 0 otherwise
- Outputs (final layer):
  - estimates of probabilites of notes at next time step



(full pass at each time step)

(in each layer:)



$$\mu_a(t) = \tanh[\mathrm{J}_a \mu_{a-1}(t) + \mathrm{M}_a \mu_a(t-1)]$$

Figure from B M Mostafa et al, Machine and Deep Learning Approaches in Genome. Alfarama J Basic App Sci **2**, 2021

cost function: negative log likelihood
training: BPTT

Running the trained network

- “Seed” it by forcing it through a starting sequence of notes
- Then use the outputs as the inputs for the next step, repeat …

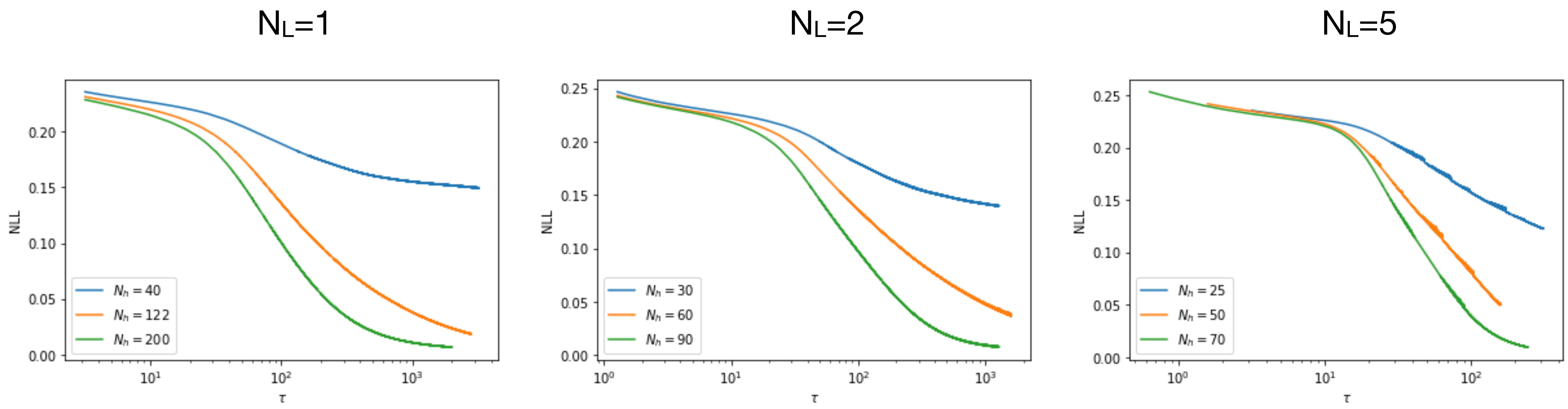
Performance: general features

- Can train networks with $N_L = 1 - 10$ hidden layers, $N_h = 25 - 300$ units/layer to the NLL (negative log likelihood /unit /timestep) on the training set (20-300 chorales) to generate recognisable music.

- Require an NLL $\lesssim 0.02$ bits or less for reasonable results.

- Generalization error not informative about performance quality.

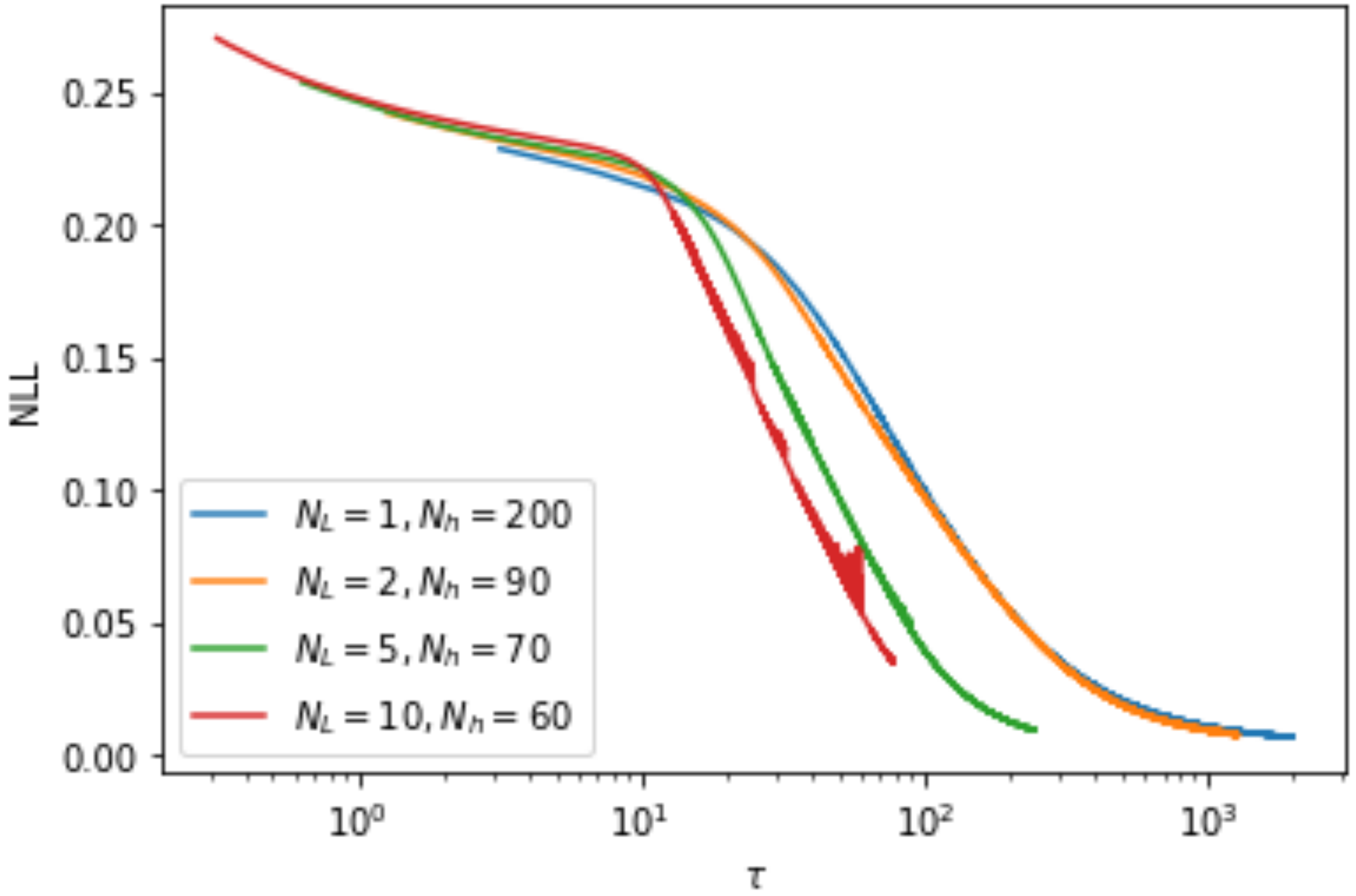
Learning: general features



τ: learning time measured in units of inverse learning rate

Interesting region: 0.01 < NLL < 0.1 (power law decay NLL ~ $1/\tau^{\gamma}$)

Deeper is faster*



* as measured in τ ; but learning rate has to be reduced with increasing depth

The Learnability Transition

- Similar to the capacity problem for a perceptron with N input components (Cover, Gardner & Derrida): What is the maximum number p of input patterns that can be correctly classified into 1 of 2 possible classes?

- Gardner-Derrida analysis (simplest case): $p < 2N$.
  For $p < 2N$, replica symmetry holds; for $p > 2N$ solution requires full (Parisi) replica symmetry breaking. I.e., $p > 2N$ is in the same class as the Sherrington-Kirkpatrick (SK) spin glass.

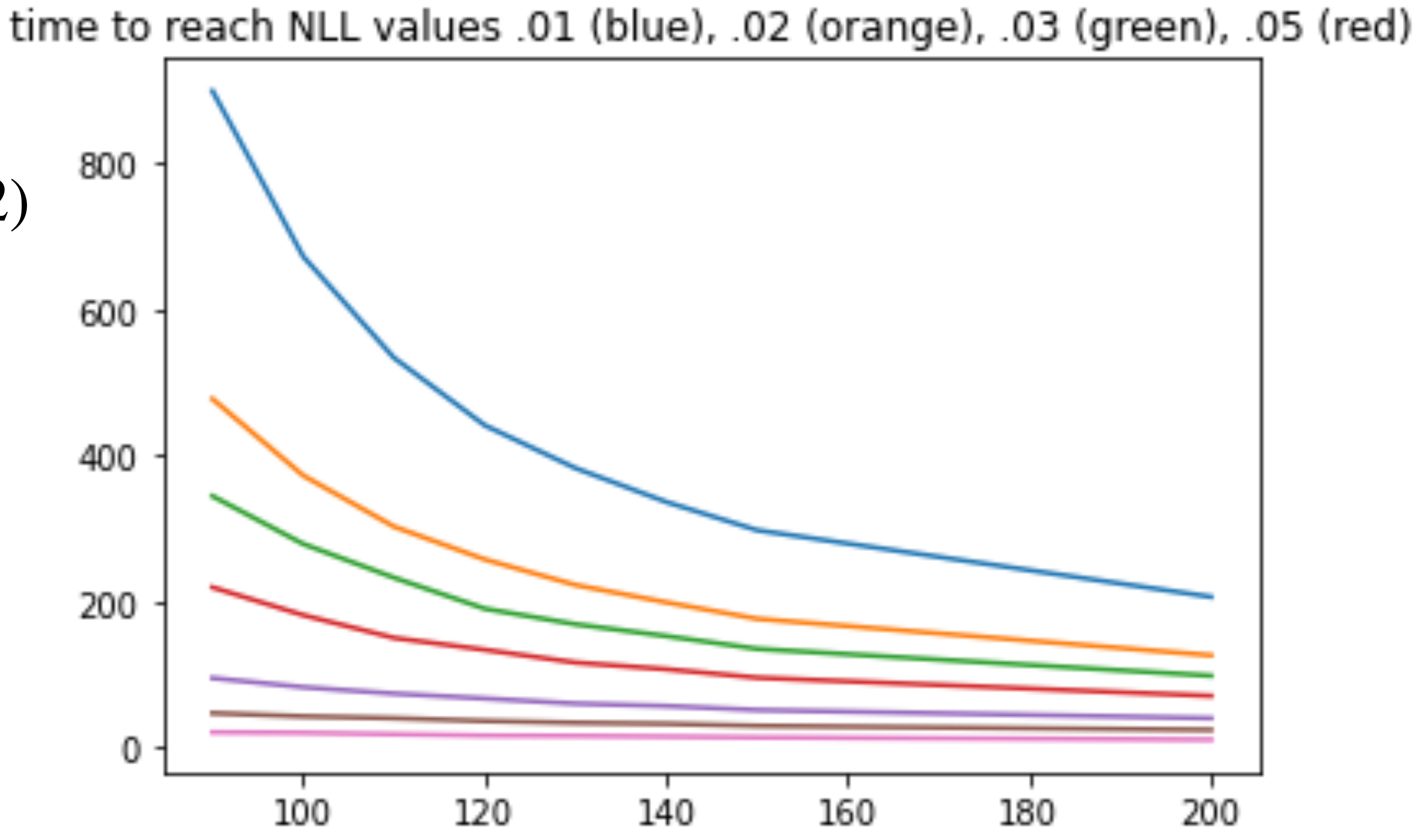
- Is the learnability transition here like that in the perceptron and the SK model?

Dynamical perspective

- At the SG transition, the Almeida-Thouless (AT) instability is reached — diverging fluctuations of the pair correlations:
- $\langle S_i S_j \rangle = 0 \qquad \frac{1}{N}\sum_{ij} \langle S_i S_j \rangle^2 \propto \frac{1}{T^2 - T_g^2}$
- This is reflected in the dynamics: critical slowing down. $\tau \propto 1/(T - T_g)$ (Sompolinsky-Zippelius).
- Do we see this here?

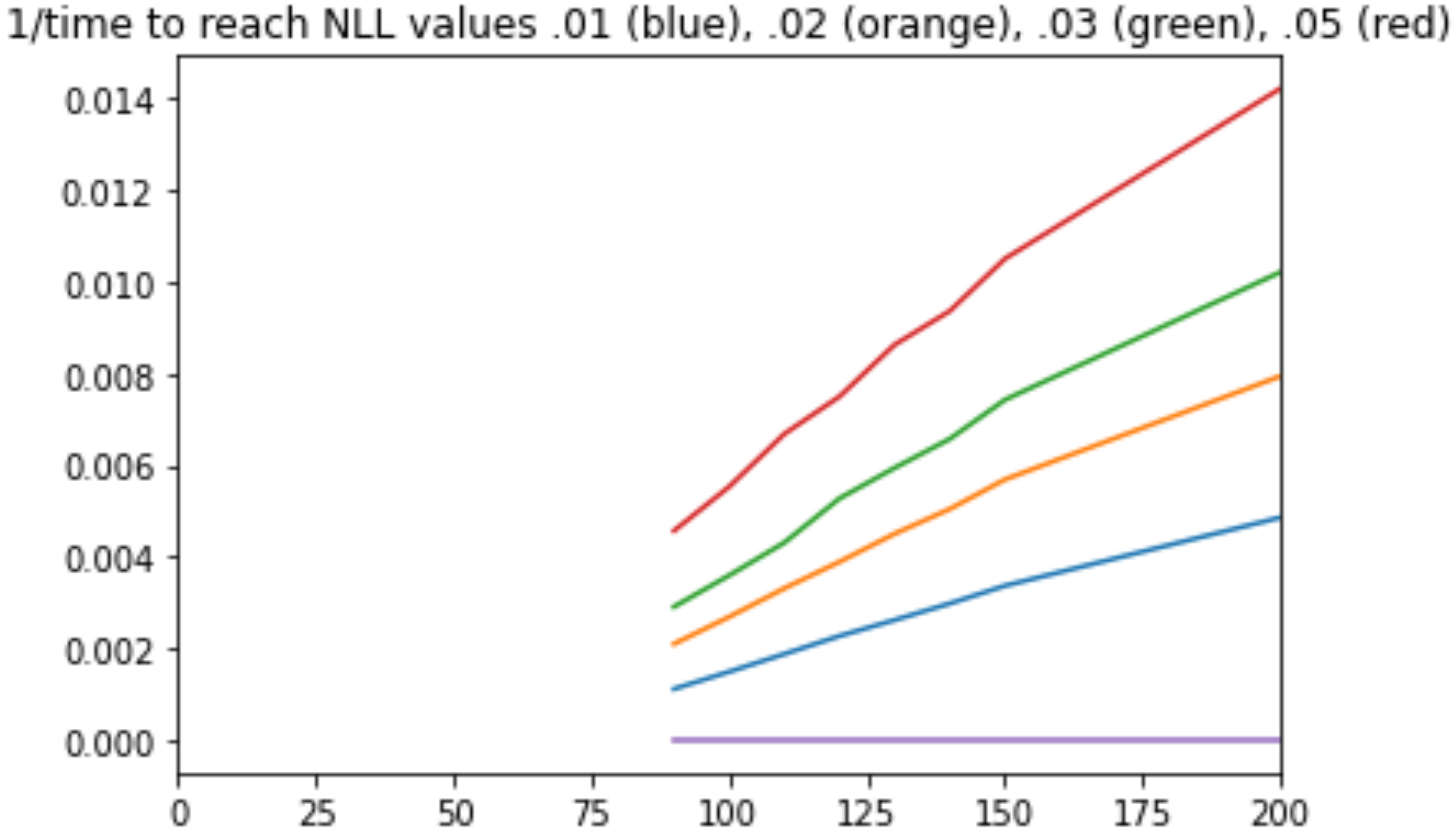
Learning times

$(N_l = 2)$



N_h

Inverse learning times

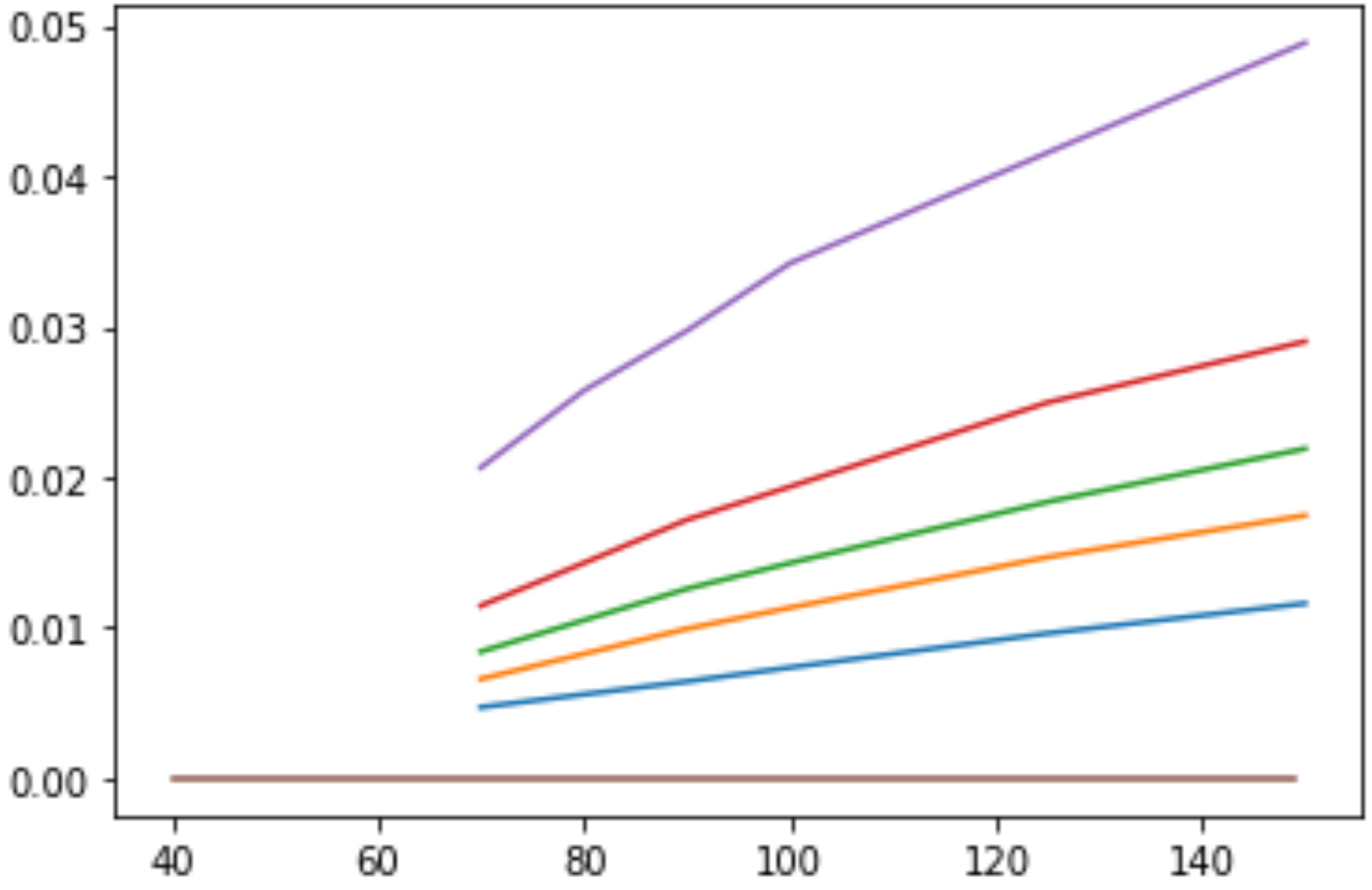


N_h

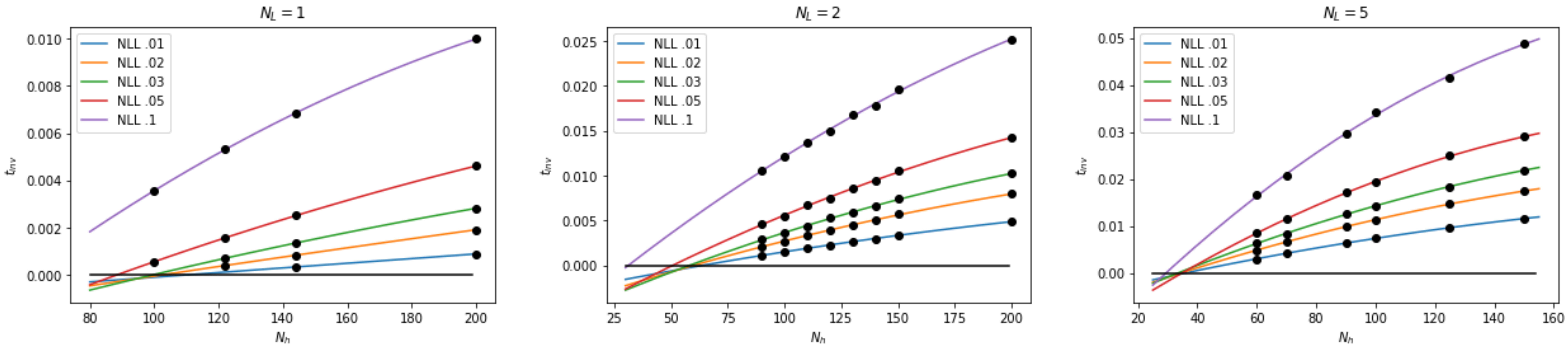
Deeper network:

$N_L = 5$:

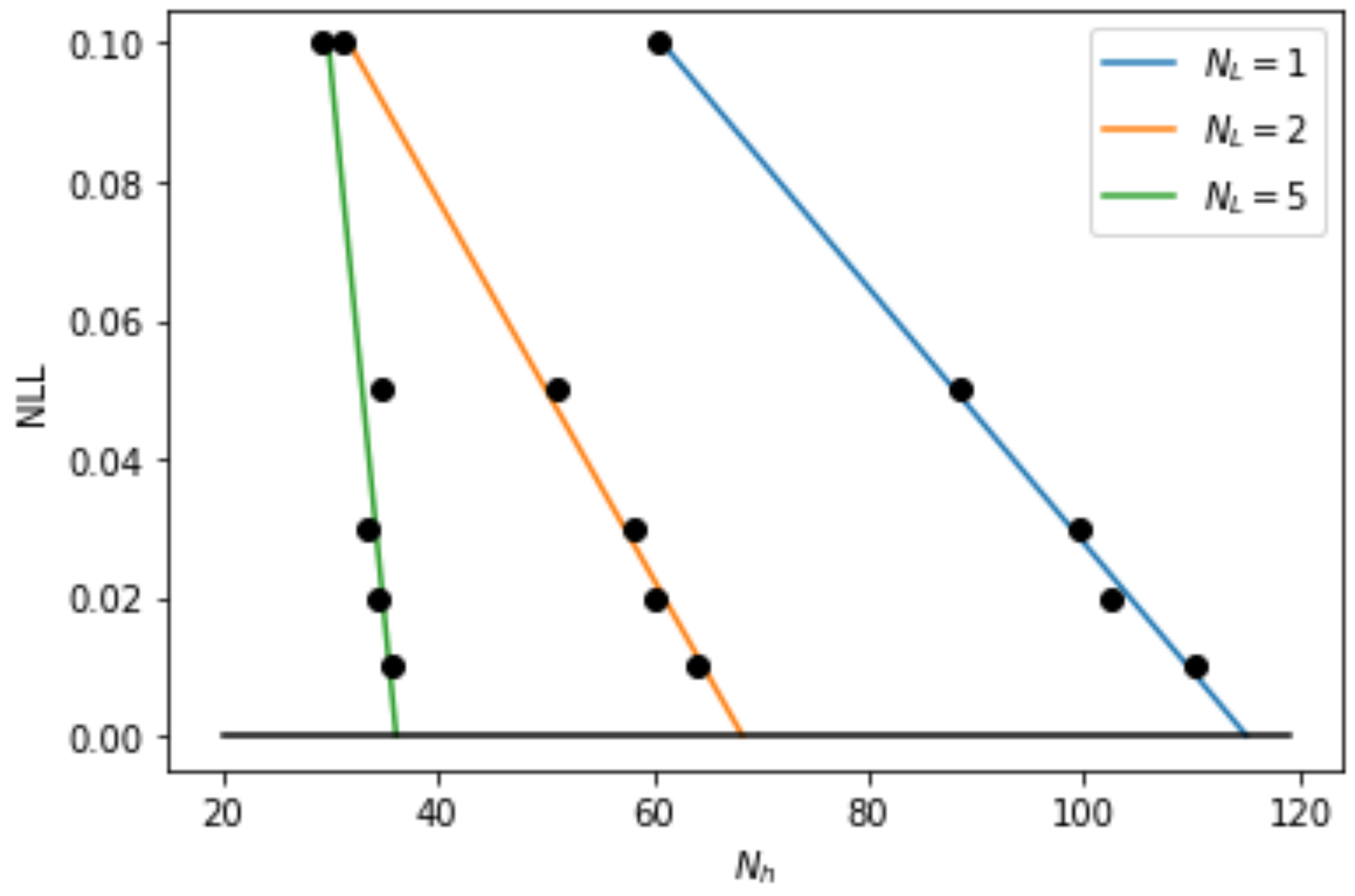
inverse time to reach NLL .01 (blue), .02 (orange), .03 (green), .05 (red), .1 (violet)



Locating the transition



NLL vs critical N_h



Glassy dynamics

In a spin glass phase, dynamics show aging

(longer time since quench $\implies$ slower dynamics)

measure of aging:

weights spread away from values at t_w, mean square displacement at t: $D(t_\mathrm{w}, t_\mathrm{w} + t) = \langle [J_{ij}(t_w + t) - J_{ij}(t_w)]^2 \rangle_{ij}$

effective noise (temperature) in stochastic gradient descent:

$$\Gamma(t) = \frac{1}{N_\mathrm{batch}} \left\langle \mathrm{var}_\mu \left(\frac{\partial E_\mu(t)}{\partial w_{ij}} \right) \right\rangle_{ij}$$

where $E_\mu(t)$ is the cost function evaluated on training example μ at time t.

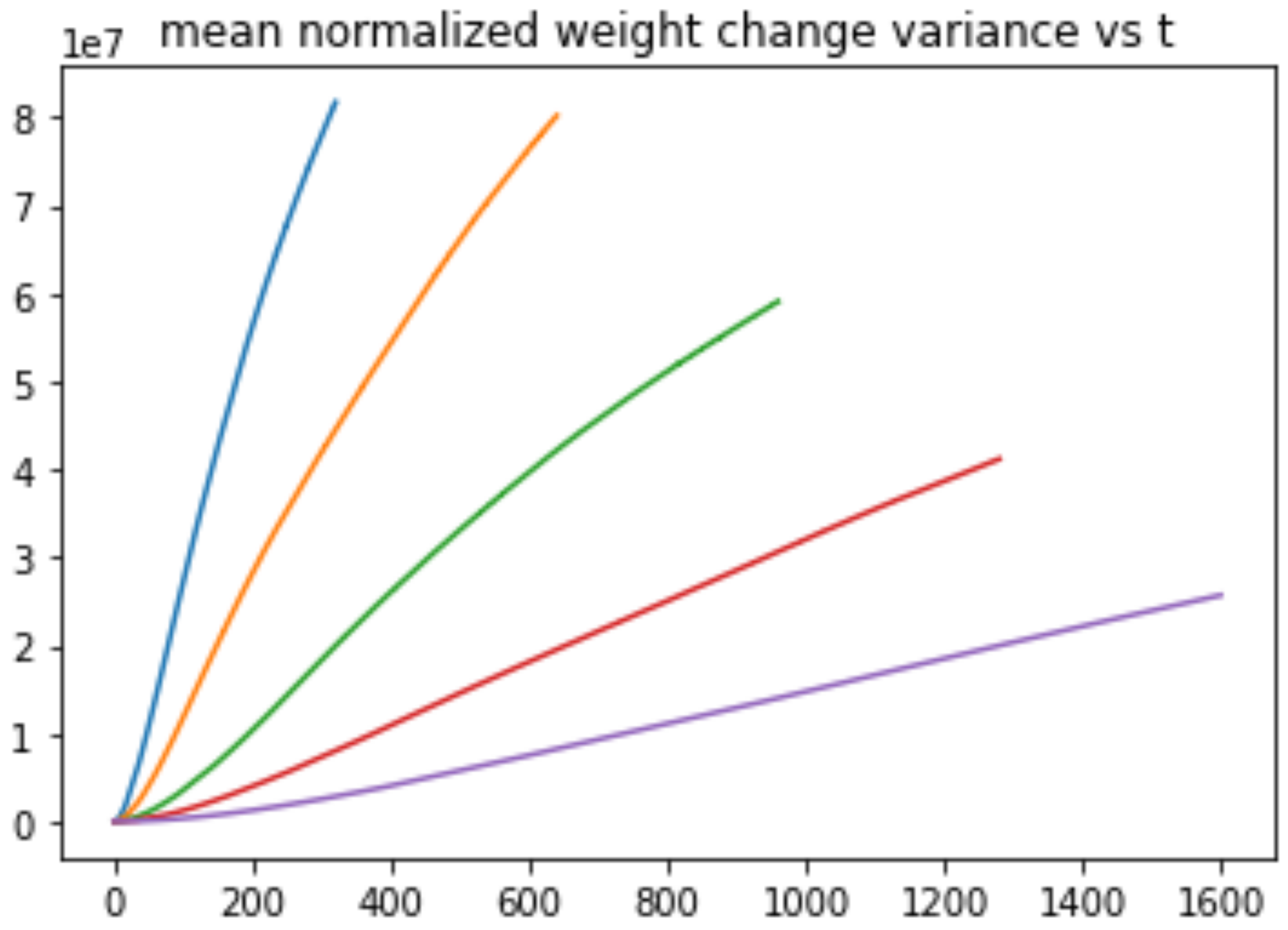
At each step t' between t_w and $t_\mathrm{w} + t$, accumulate the growth of D relative to $\Gamma(t')$:

$$\Delta(t_\mathrm{w}, t_\mathrm{w} + t) = \int_{t_\mathrm{w}}^{t_\mathrm{w}+t} dt' \frac{\partial D(t_\mathrm{w}, t_\mathrm{w} + t')/\partial t'}{\Gamma(t')}$$

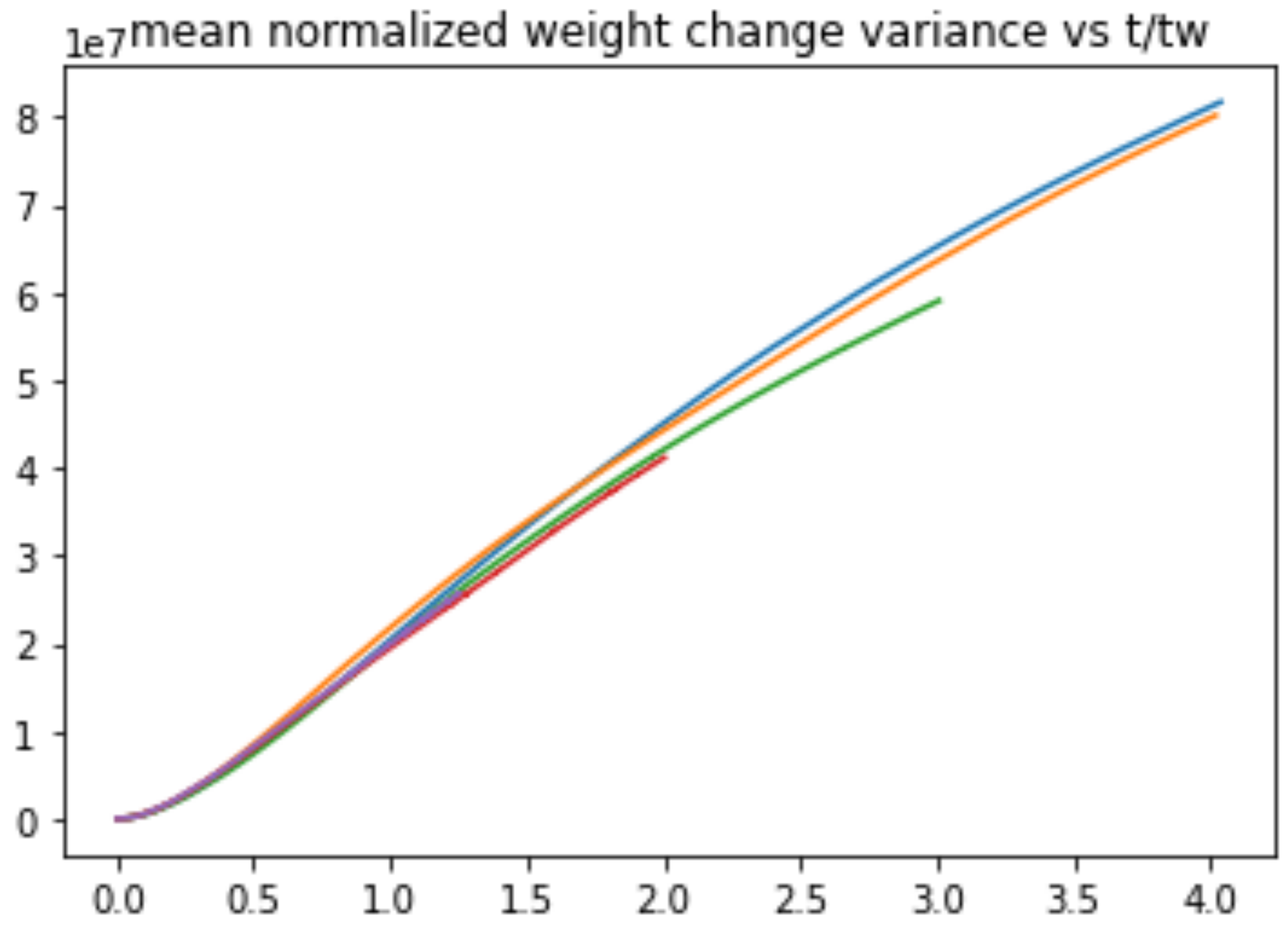
(studied for non-recurrent nets far from transition by Bati-Jesi et al, ICML 2018)

Here, $N_L = 2$, $N_h = 30$:

Δ vs t:



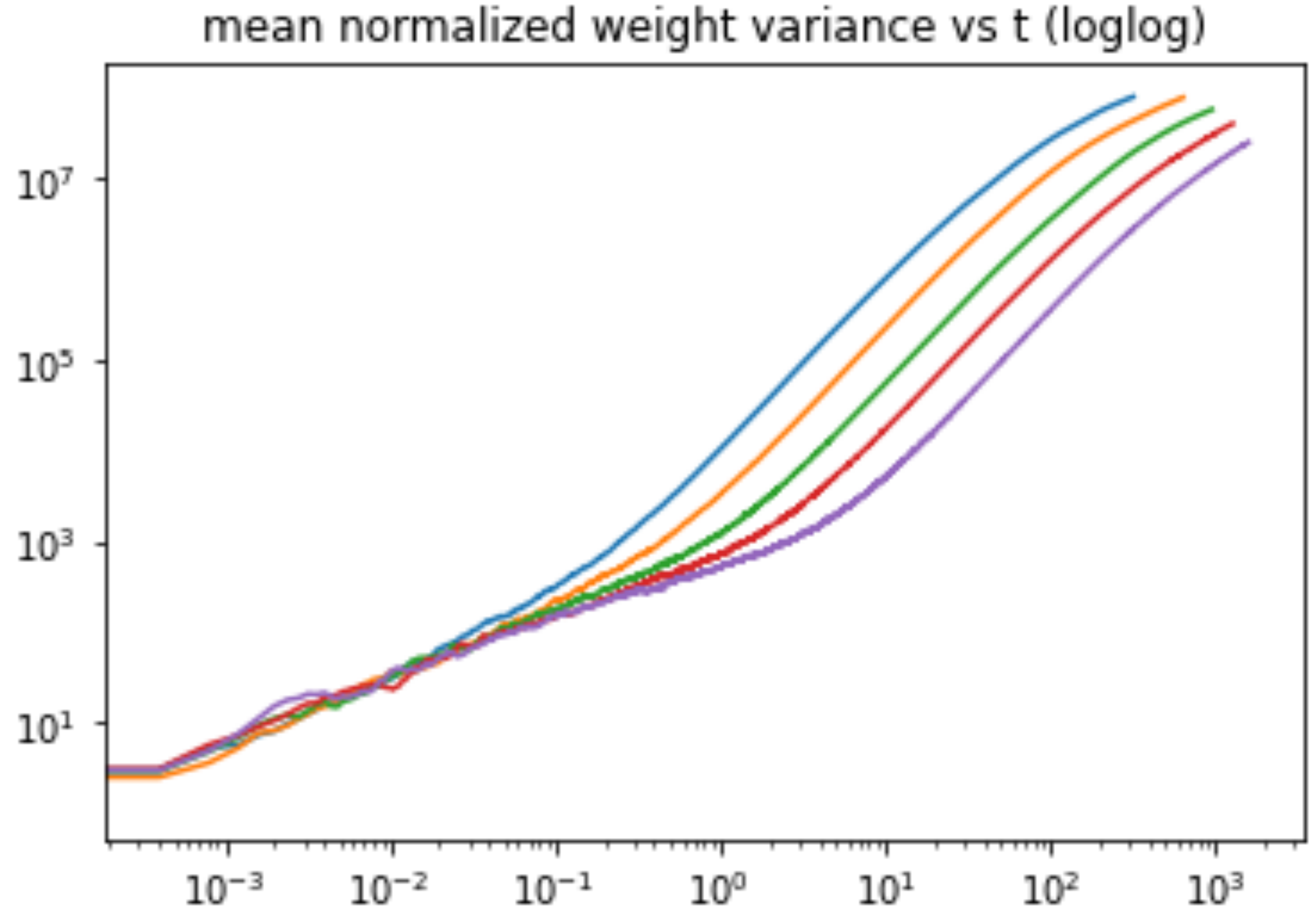
Δ vs t/t_w:



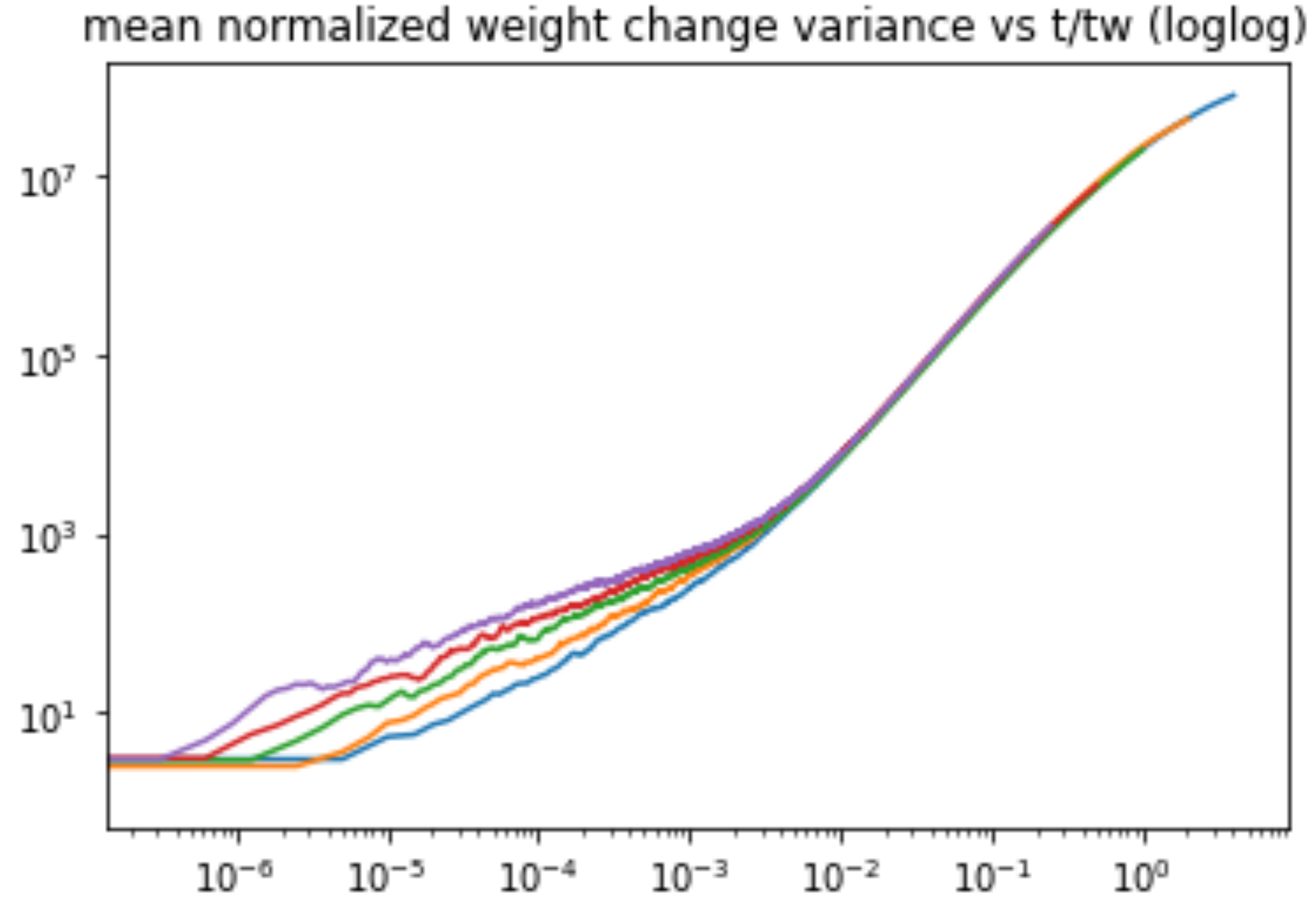
t_w:
80: blue
160: orange
320: green
640: red
1280: violet

$N_L = 2$, $N_h = 30$ log-log plots:

log Δ vs log t:



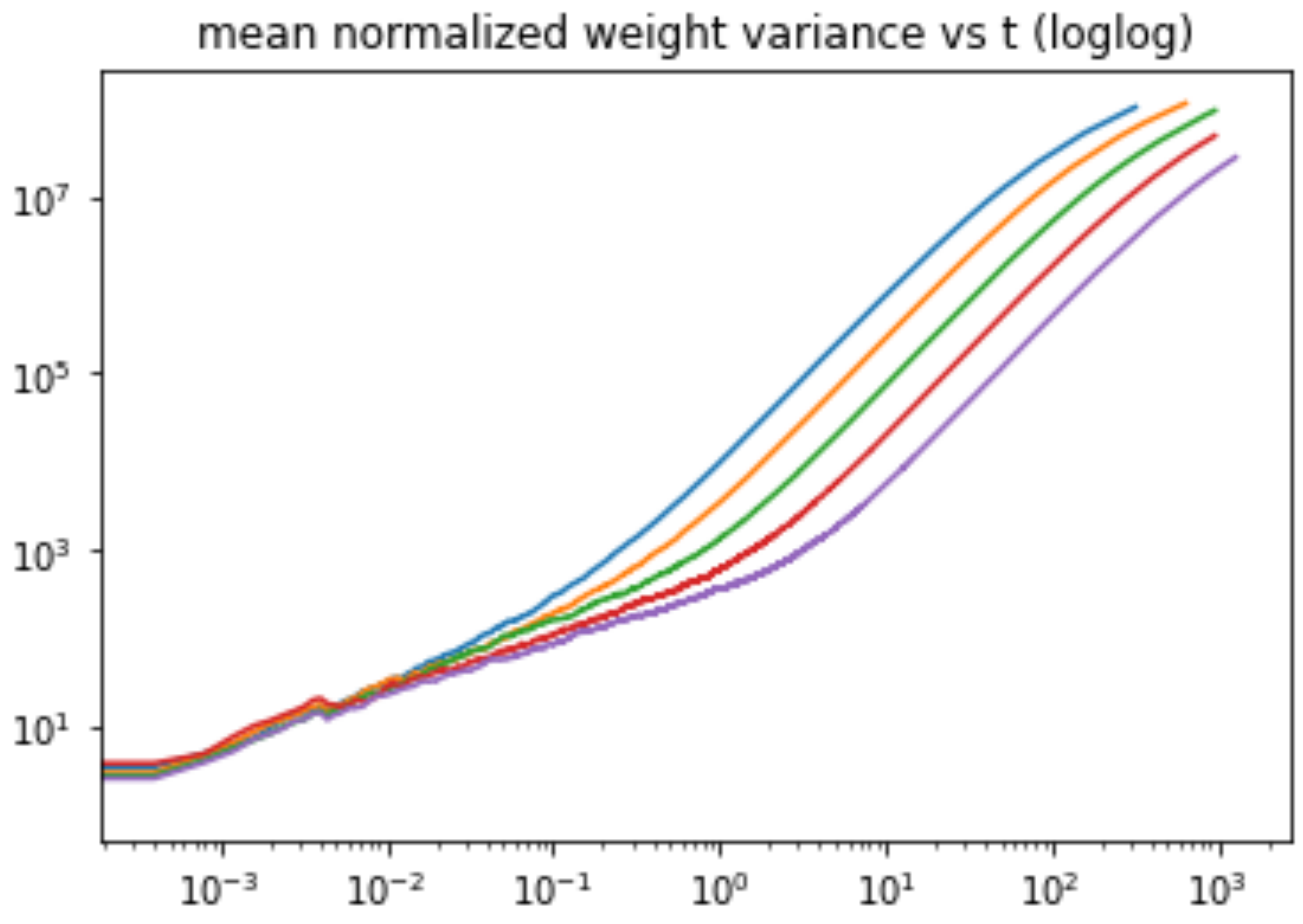
log Δ vs log t/t_w:

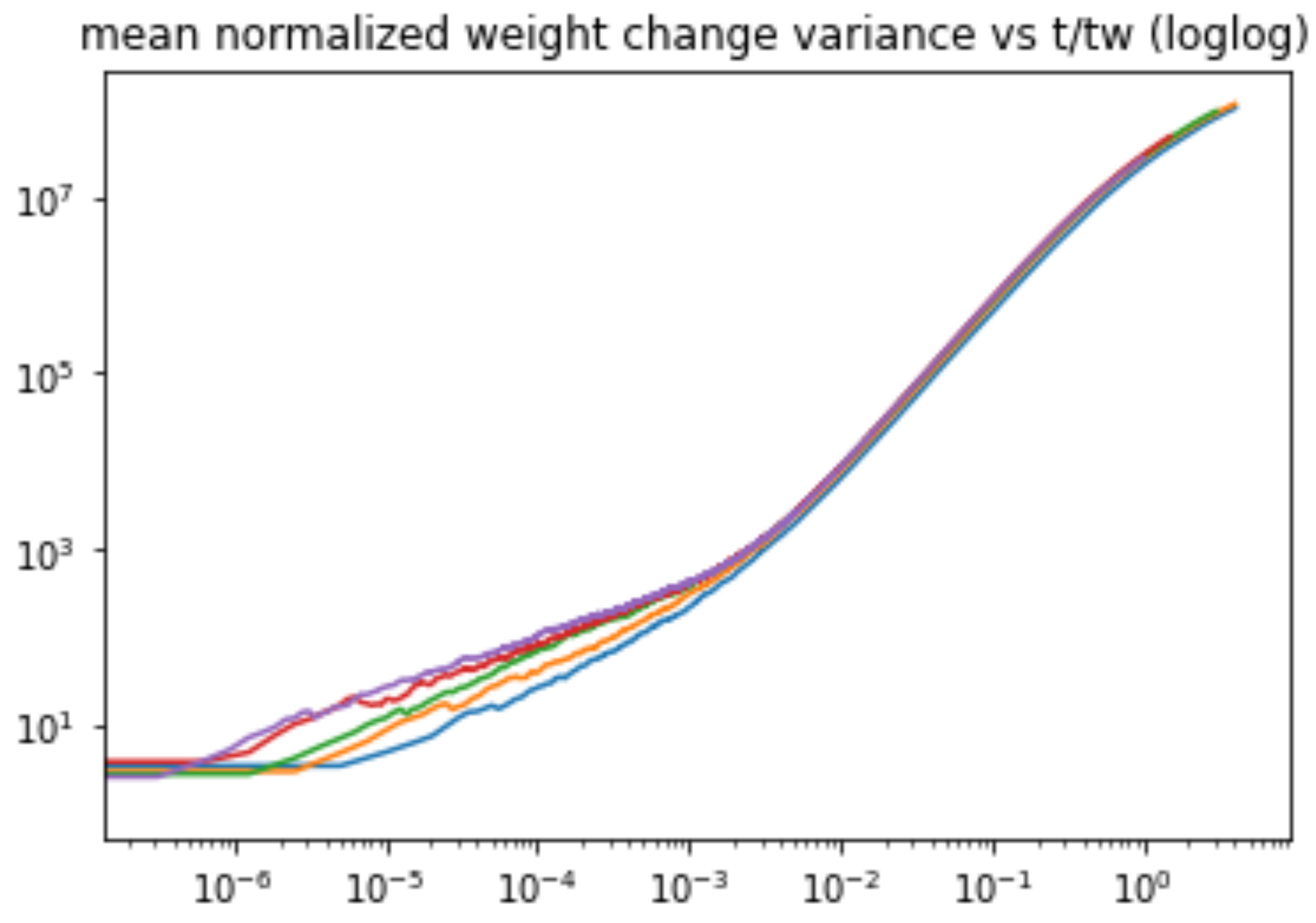


t_w:
80: blue
160: orange
320: green
640: red
1280: violet

Collapse to function of t/t_w: like ``weak ergodicity breaking” seen in p-spin glass models

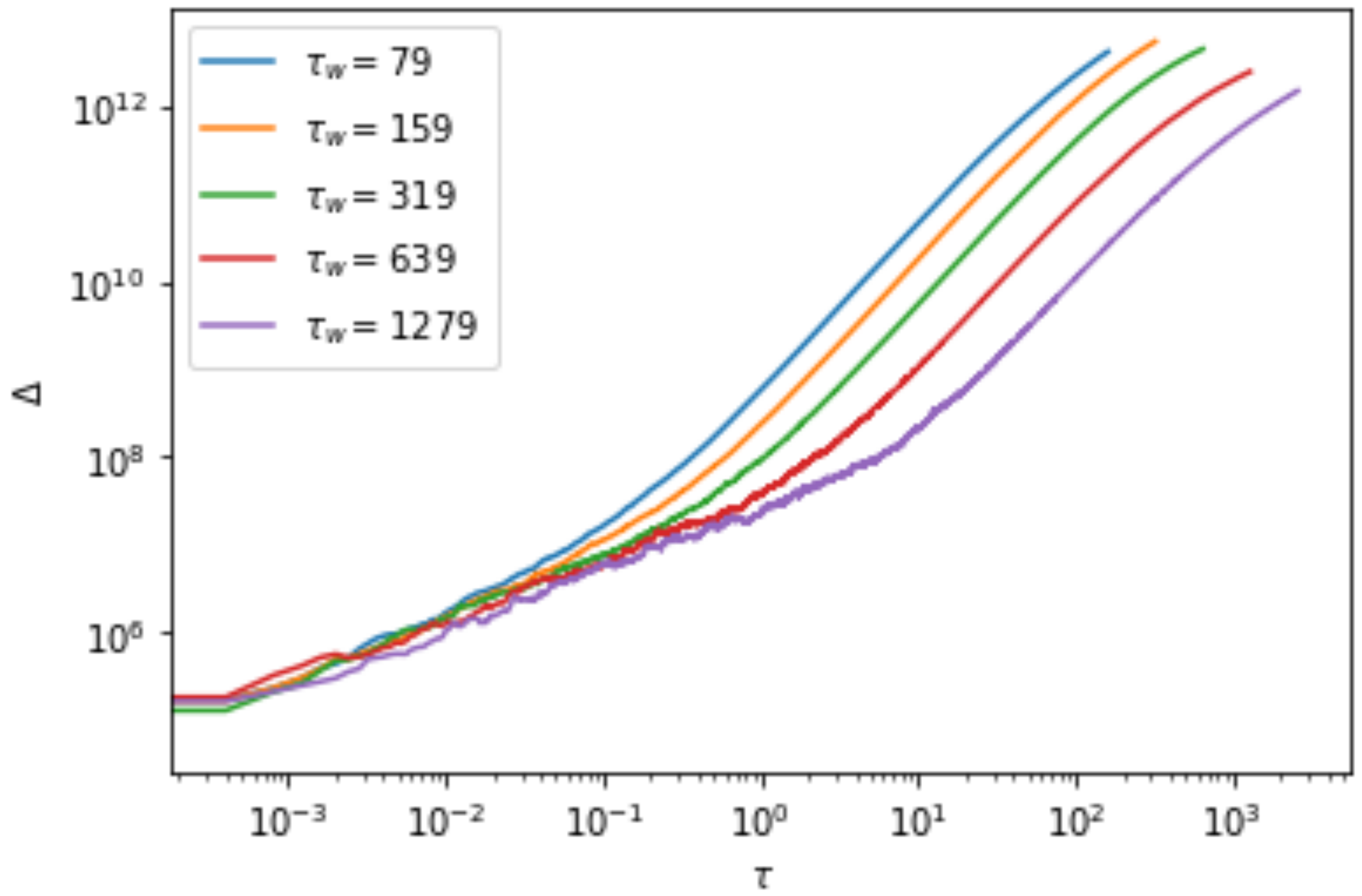
60 hidden units / layer:

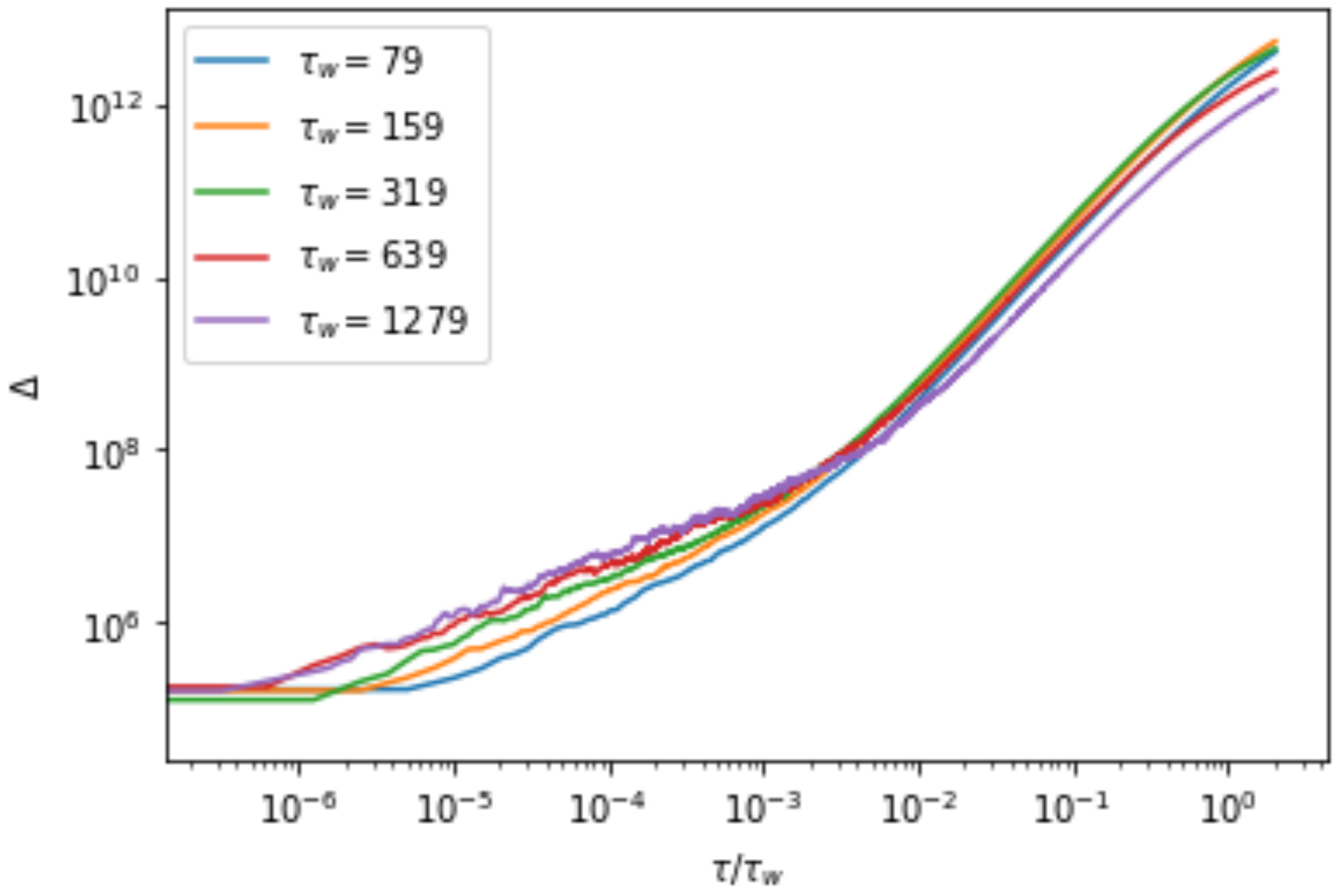




t_w:
80: blue
160: orange
320: green
640: red
1280: violet

$N_L = 2$, $N_h = 90$:





Scaling with t/t_w breaks down

What we've learned

1. Learning undergoes critical slowing down $\tau \propto 1/(N_h - N_h^c)$ like that in spin glasses

2. Weight correlations show aging behaviour (function of τ/τ_{w}), independent of τ_{w} (but for $N_h > N_h^c$ (overparametrized case), aging stops at very long times.)

Things we still want to know:

1. The $C(\tau/\tau_\mathrm{w})$ we find is not of the power-law form found by Cugliandolo & Kurchan.
   Why?
2. We have data only up to $\tau/\tau_\mathrm{w} = 2$.
   What is the asymptotic $\tau/\tau_\mathrm{w} \to \infty$ behaviour?
3. In Cugliandolo-Kurchan theory, there is no aging at or above the transition at N_h^c.
   But we find it, at least if τ_w is not too long, even above N_h^c ("para-aging?") Why?
4. Everything here was for NLL cost function and stochastic gradient descent.
   What would happen for other cost functions and learning algorithms?
5. We used a layered recurrent architecture with orthogonal matrices.
   What about other architectures?
6. Further exploration of the phase diagram -
   Other phases?
7. Your suggestions?

A quantum connection

Every step of the operation of our networks involves a orthogonal transformation (and could also be done with general unitary matrices).

Quantum computation is also done with unitary transformations.

So how would one make a quantum network do our problem?

Quantum computing

Quantum computing is done with quantum gates
Quantum gates perform unitary transformations on 1 or 2 qubits (spinors)

Classical computers also do simple operations at the bit level, but we never think about what is happening to the individual transistors. We have (several levels up) compilers, etc. But in quantum computing we think (for now) at the level of qubits and gates.

However, qubits are much richer objects than bits, so we can do some interesting things already using manipulations at the 1- and 2-qubit level.

qubits and gates:

A general qubit is just a spinor $\alpha|1\rangle + \beta|0\rangle, \quad |\alpha|^2 + |\beta|^2 = 1$
A physical qubit can be rotated by applying a magnetic field.

Simplest kind of gate (1-qubit) just rotates it around a specific axis by a specific angle. (Angle $\propto$ time the field is applied.)

2-qubit gate entangles the input qubits (makes an output qubit which is a linear combination of them). Coefficients in the linear combination can be controlled (using a 1-qubit rotation gate as part of the gate).

So how do we make a quantum recurrent network?

Preparing the input

Richer possibilities than classical bits:

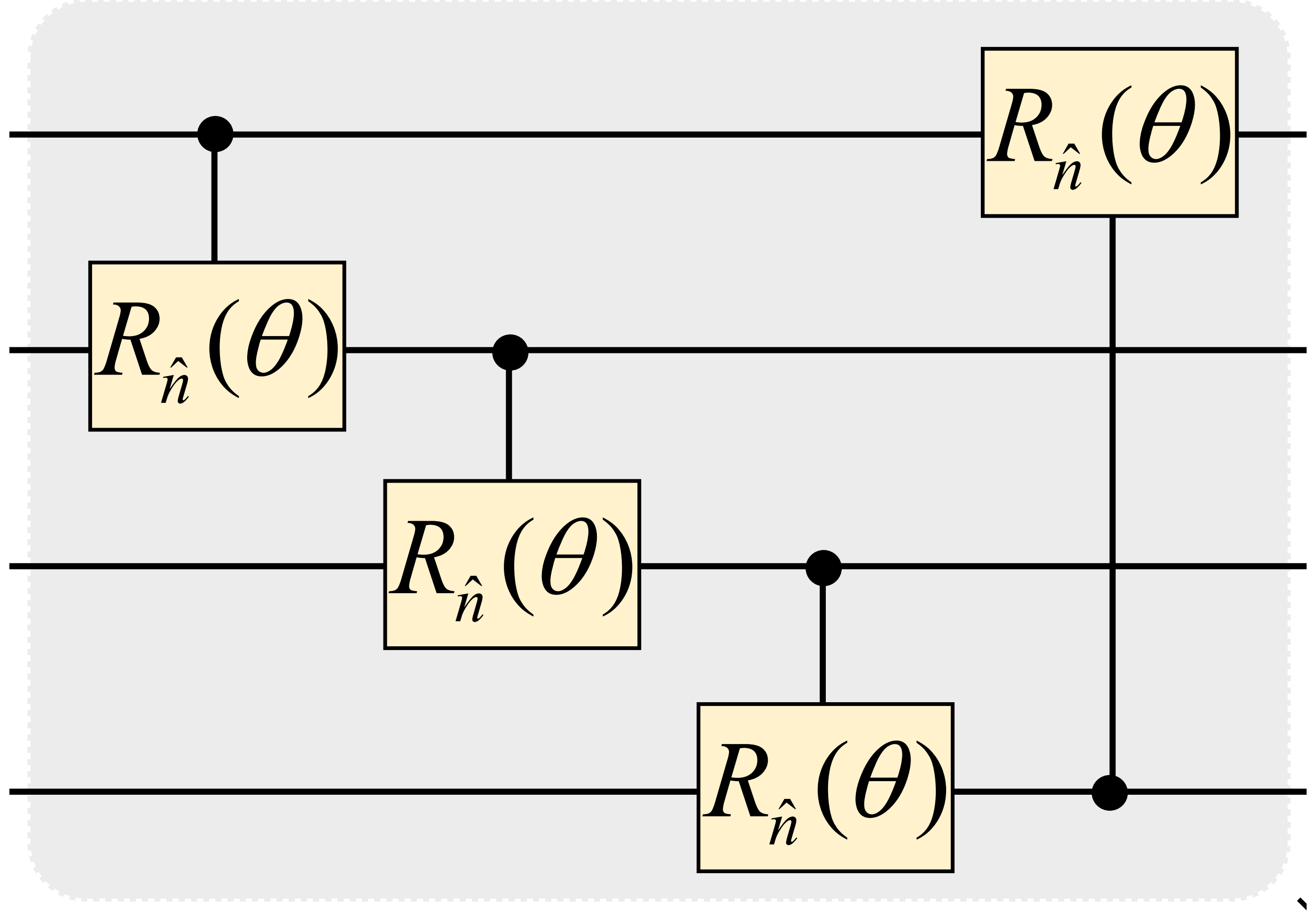
In our problem inputs are combinations of 4 musical notes (SATB).
So, for example, we could (with good enough hardware) encode all 12 notes in 1 octave by rotation an initially spin-up qubit by multiples of $\pi/6$ around (say) the y axis (like hours on a clock). Then we would only need 4 qubits (1 for Soprano, 1 for alto, etc.).

In general, many ways to exploit the different rotation possibilities.

First processing layer:

In the classical network we first calculate $\sum_j J_{ij} x_j$

In the quantum network, we have to entangle all the input qubits (and differently for each receiving unit i). To do this with 2-qubit gates only, do it in steps:

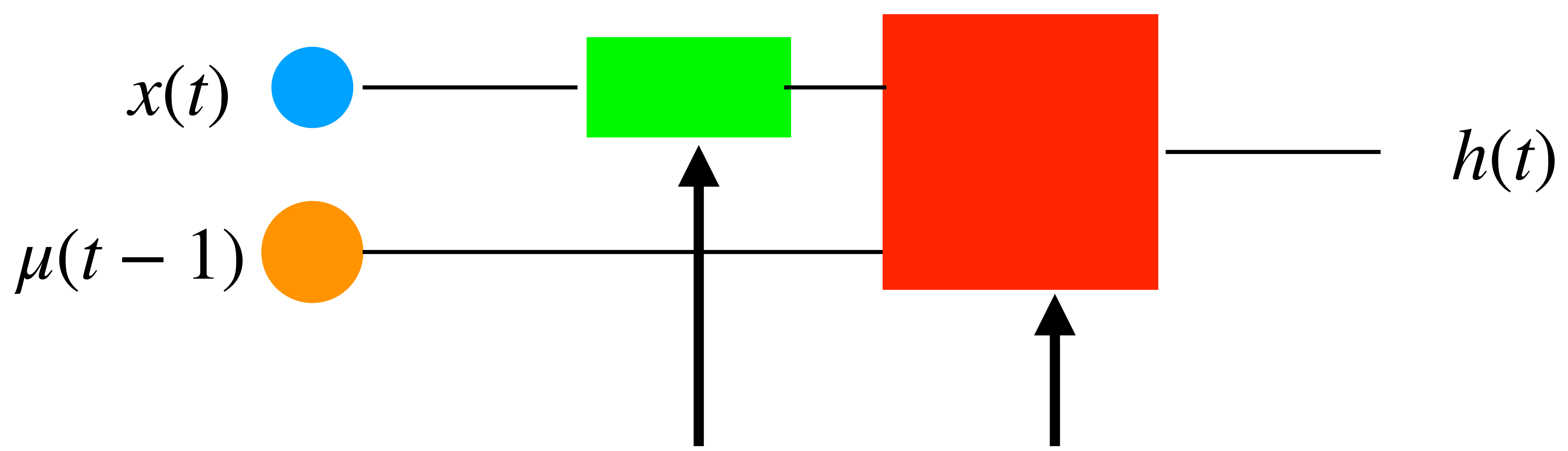


From Li et al, arXiv 2302.03244

The angles θ play the role here of the J_{ij}

Recurrent layer

Now we have to entangle these outputs with the hidden-unit qubits
(here: just 1 hidden layer)



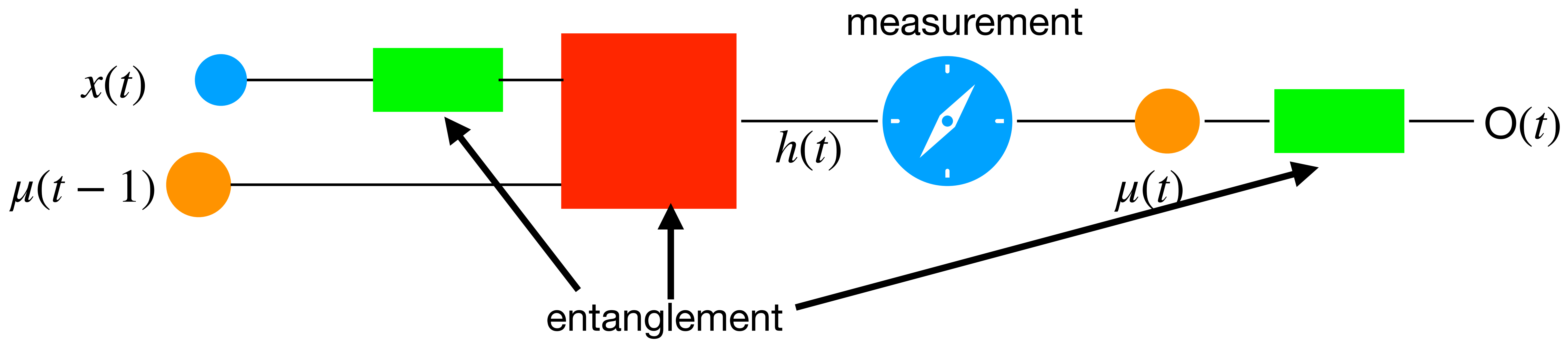
(modified from Li et al)

entanglement like on preceding slide

But where is the nonlinearity?

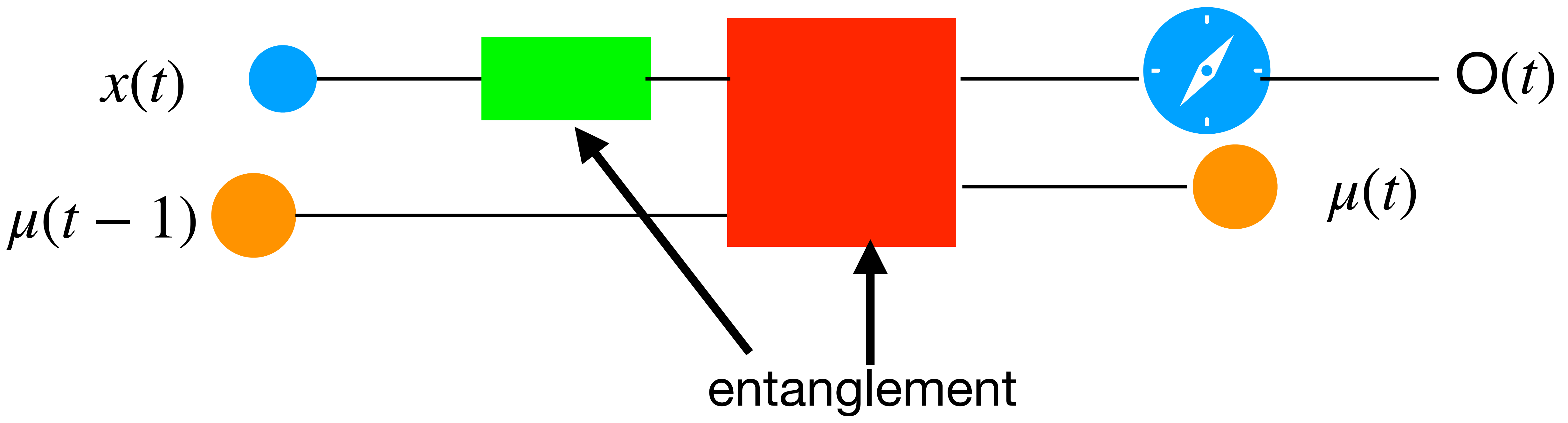
Recurrent layer output:

Need to make a measurement!
(or many to average over, to estimate output qubits better)



Alternative scheme

(Li et al)



(Linear except for output measurement)