EXP002: Empirical NTK Drift and the Onset of the Lazy Regime

Experiment Summary with Figures

Shreyas Kalvankar

November 2025

Goal and Context

The goal of these experiments is to determine when a finite-width neural network enters the *lazy regime*, a regime in which the empirical Neural Tangent Kernel (NTK) remains effectively constant throughout training, and the network dynamics match those of kernel regression with a fixed NTK.

We investigate whether:

1. the empirical NTK becomes approximately constant after some training time;
2. network predictions match kernel regression when using the frozen NTK;
3. the NTK freeze time corresponds to a plateau or slowdown in the training loss.

Our regression target is a Fourier mixture on the unit circle:

$$f^*(\gamma) = \sum_k a_k \cos(k\gamma + \phi_k),$$

evaluated on a dense uniform grid, with random subsampled training points.

We sweep fully-connected ReLU MLPs of widths

$$W \in \{100, 512, 1024, 2048, 10000\}$$

across multiple seeds.

Snapshots at regular intervals include:

- the training–training NTK, $K_{\text{train,train}}(t)$;
- the eval–train NTK, $K_{\text{eval,train}}(t)$;
- the evaluation-grid network predictions $f_{\text{net}}(\gamma, t)$;
- the snapshot training loss $L(t)$;
- metadata for seeds and training steps.

These allow reconstruction of drift curves, kernel regression predictions, loss evolution, and freeze-time detection.

1 Kernel Drift, Eigenvalue Drift, and Loss Plateau

1.1 Kernel Drift Definition

At each snapshot step we compute the empirical NTK on the training set:

$$K(t) := K_{\text{train,train}}(t) \in \mathbb{R}^{M \times M}.$$

To measure how the kernel evolves during training, we track the *normalized NTK drift*:

$$\Delta_K(t) = \frac{\|K(t) - K(0)\|_F}{\|K(0)\|_F}.$$

This quantity measures the deviation of the kernel from its initialization. If the network enters the lazy regime, $\Delta_K(t)$ becomes (almost) constant over time.

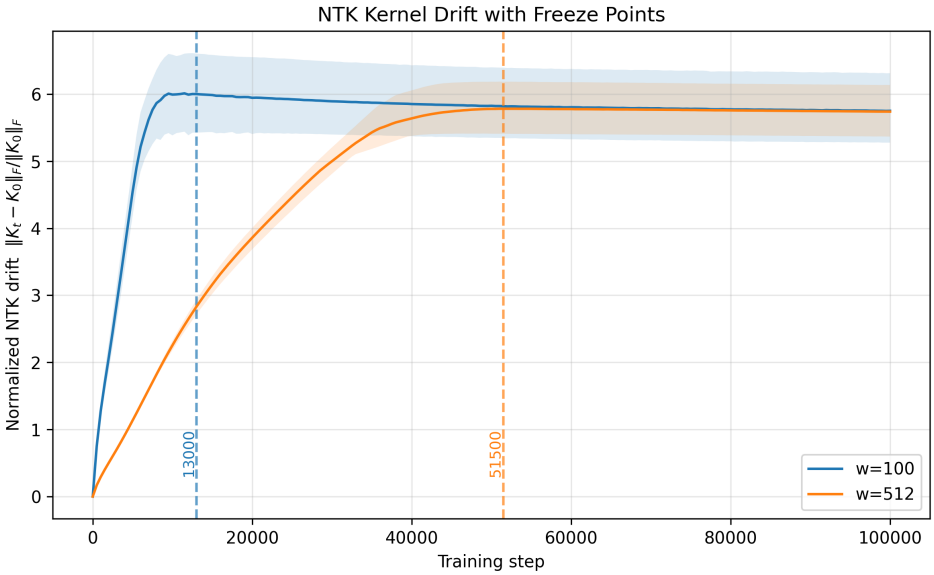

Figure 1: Kernel drift curves for low widths ($W = 100, 512$).

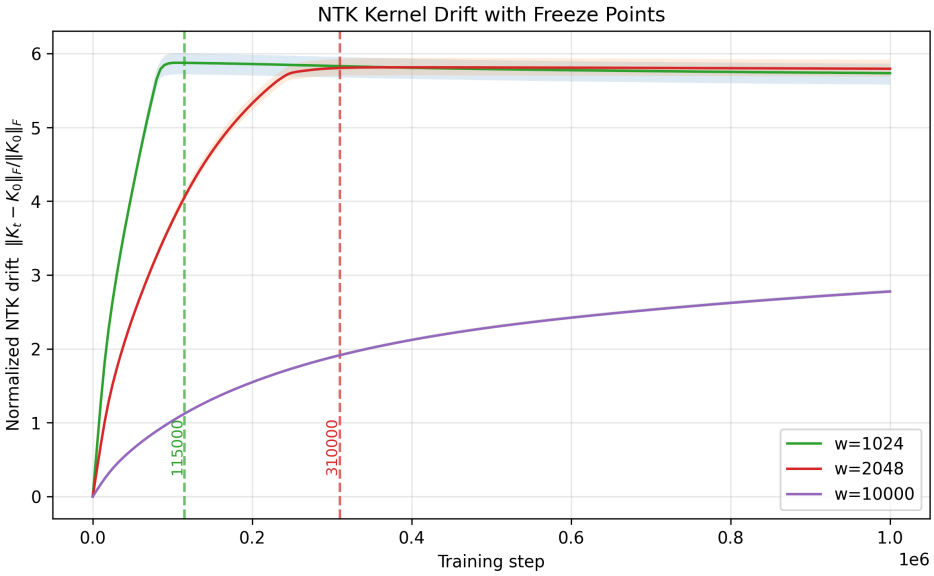

Figure 2: Kernel drift for mid-range widths ($W = 1024, 2048$).

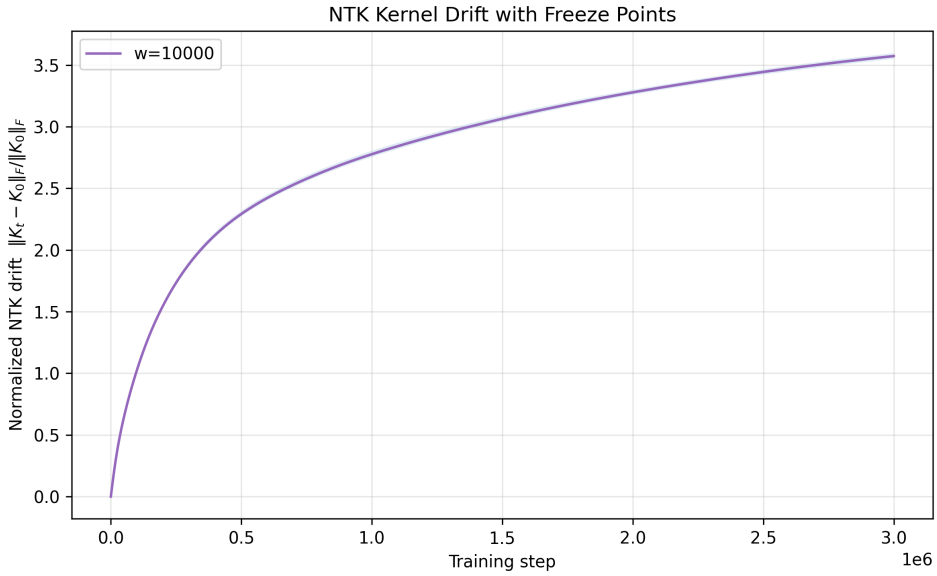

Figure 3: Kernel drift for very wide networks ($W = 10000$).

1.2 Slope-Based Freeze-Time Criterion

Even when the drift $\Delta_K(t)$ grows slowly, the kernel may have effectively stopped evolving. To detect this, we track its *slope*:

$$s(t) = \frac{\Delta_K(t) - \Delta_K(t - \Delta)}{\text{step}(t) - \text{step}(t - \Delta)},$$

with typical snapshot spacing $\Delta = 2$.

We declare the kernel *frozen* at the earliest time t_{freeze} for which the drift slope satisfies

$$|s(t)| < \varepsilon \, |s(0)|,$$

for k consecutive snapshot intervals (typically $k = 3$ and $\varepsilon = 0.01$).

This identifies the point at which the NTK's rate of change becomes negligible.

1.3 Eigenvalue Drift Across Widths

Let $\lambda_j(t)$ denote the eigenvalues of the empirical NTK at time t (sorted in descending order). For each width we track the top five eigenvalues:

$$\lambda_1(t), \ \lambda_2(t), \ \ldots, \ \lambda_5(t).$$

Empirically, we observe that the *asymptotic* values of the top eigenvalues are nearly identical across widths: regardless of the width, the leading eigenvalues of $K(t)$ saturate toward essentially the same plateau. What varies strongly with width is the *timescale* on which this saturation occurs.

- **Narrow networks** reach their asymptotic top eigenvalues quickly. The leading eigenvalues stabilize early in training.
- **Wide networks** converge to the same eigenvalue plateau much more slowly, even though the learning rate is identical for all widths.

This suggests that the empirical NTK of wide networks is less sensitive to parameter updates for the same learning rate; the changes more gradually at large width. Narrow networks, being further from the infinite-width NTK limit, exhibit larger kernel changes per unit parameter movement, which leads to faster spectral evolution.

Thus, the eigenvalue trajectories reveal a width-dependent *spectral stabilization timescale*: small networks rapidly reshape their NTK spectrum, while large networks evolve the spectrum slowly but eventually saturate at similar eigenvalues.

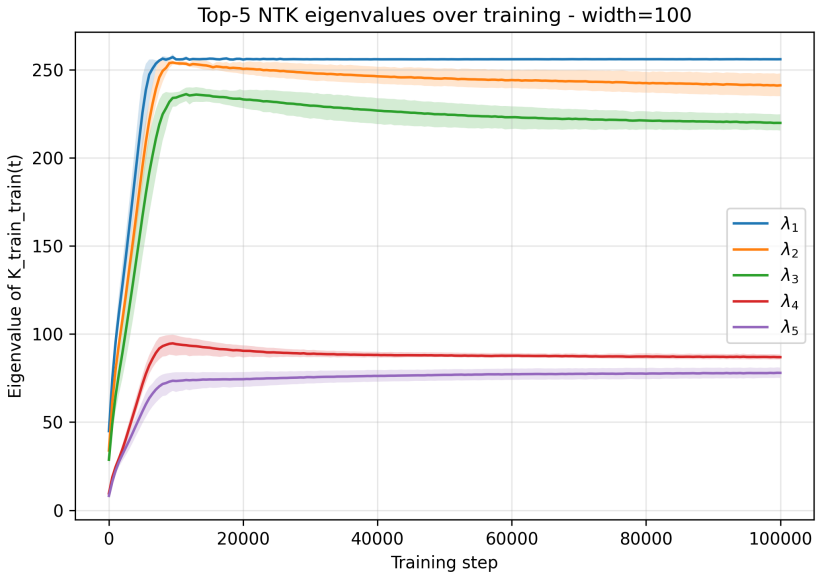

Figure 4: Top eigenvalues over training, width $W = 100$.

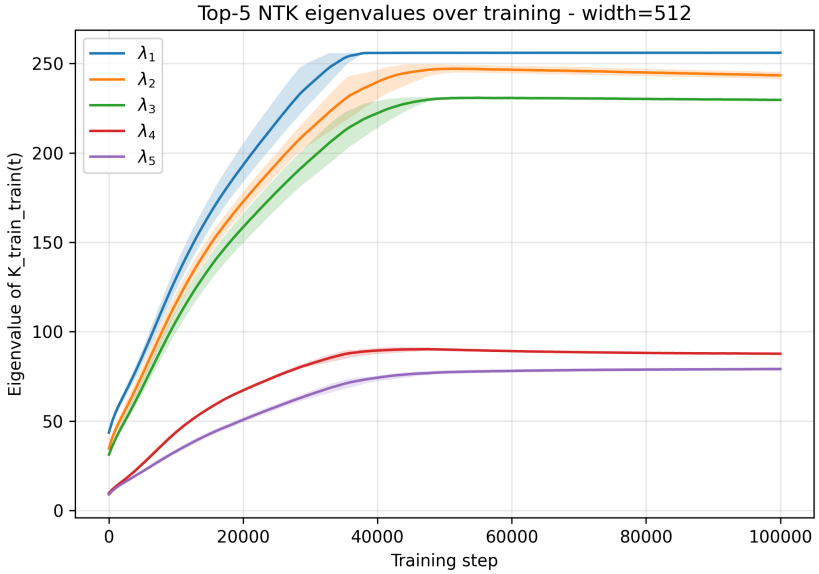

Figure 5: Top eigenvalues over training, width $W = 512$.

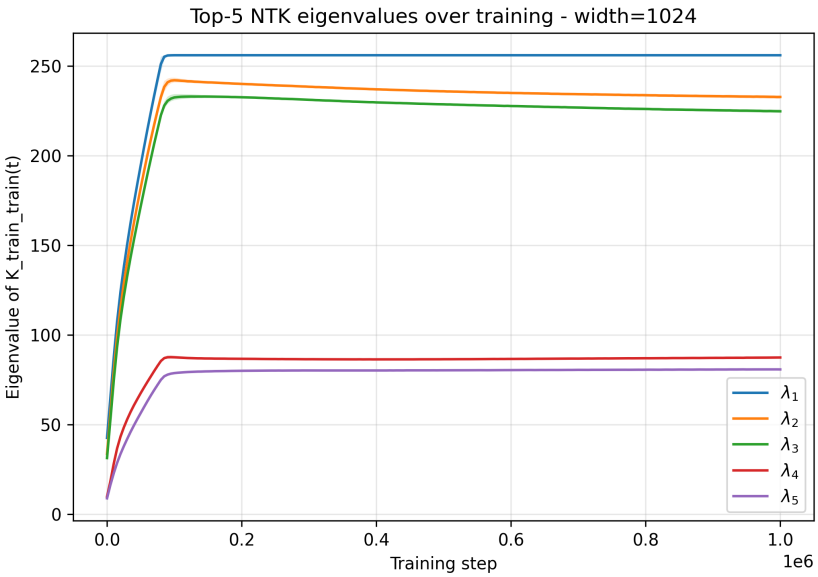

Figure 6: Top eigenvalues over training, width $W = 1024$.

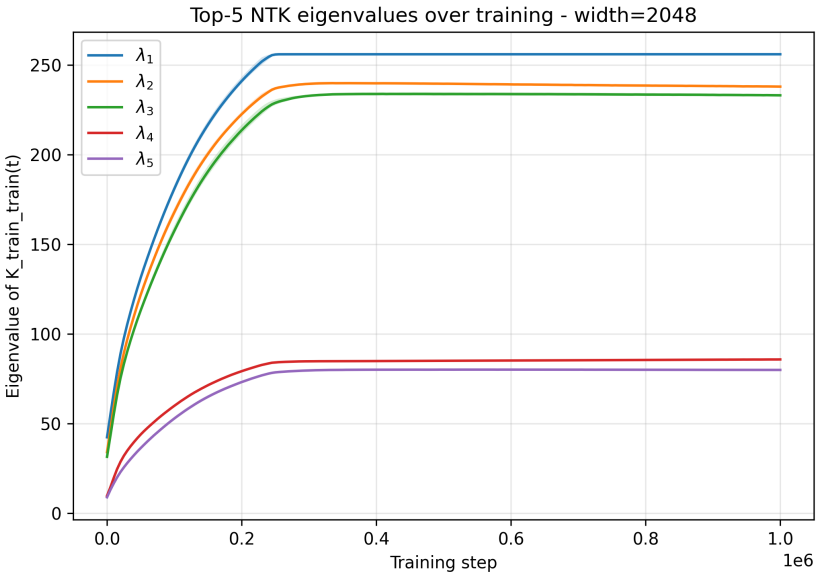

Figure 7: Top eigenvalues over training, width $W = 2048$.

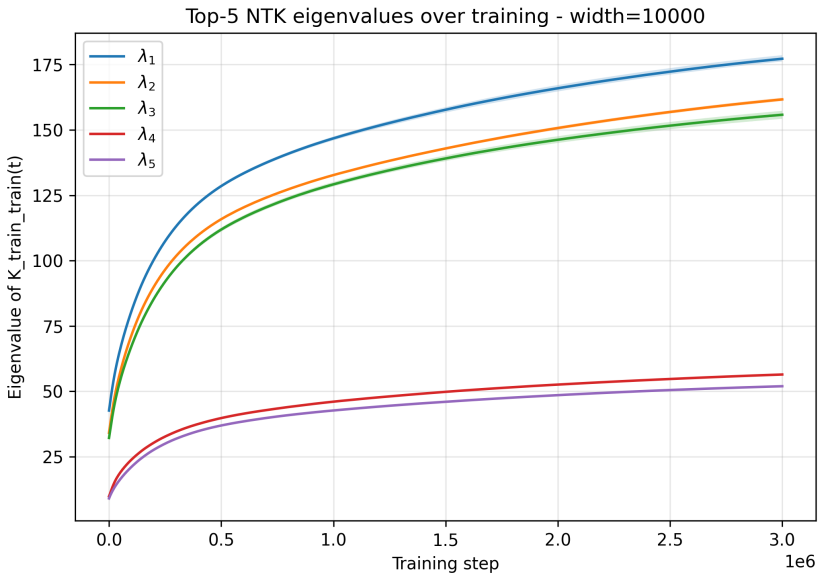

Figure 8: Top eigenvalues over training, width $W = 10000$.

1.4 Loss Plateau and Kernel Drift

We record the loss:

$$L(t) = \frac{1}{M} \sum_{i=1}^{M} \big(f_{\text{net}}(x_i, t) - y_i\big)^2.$$

To test whether loss stabilization corresponds to NTK freezing, we plot $L(t)$ alongside drift $\Delta_K(t)$ and slope $s(t)$.

The loss curve seems to have plateaued earlier than the time t_{freeze} at which the kernel drift slope becomes small, particularly for large widths.

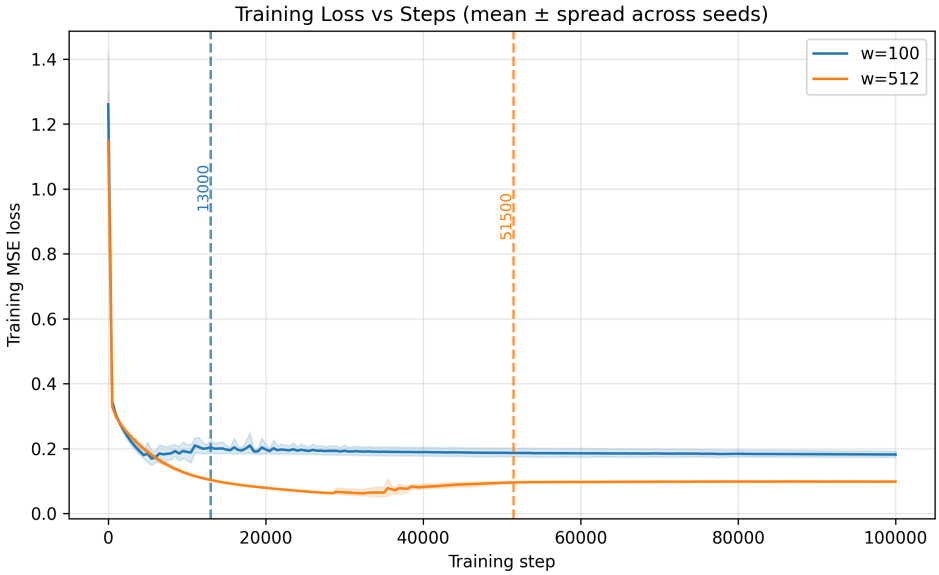

Figure 9: Loss curves for low widths.

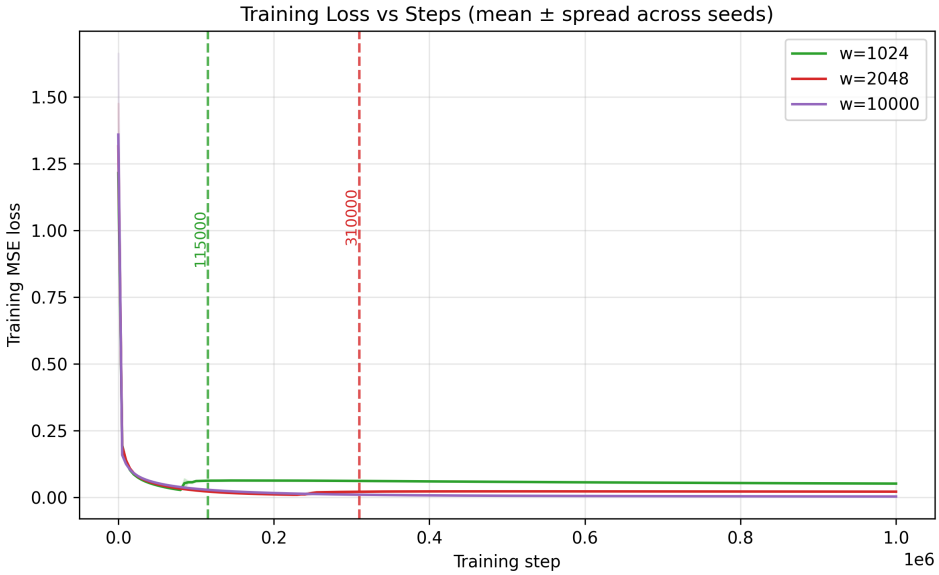

Figure 10: Loss curves for mid-range widths.

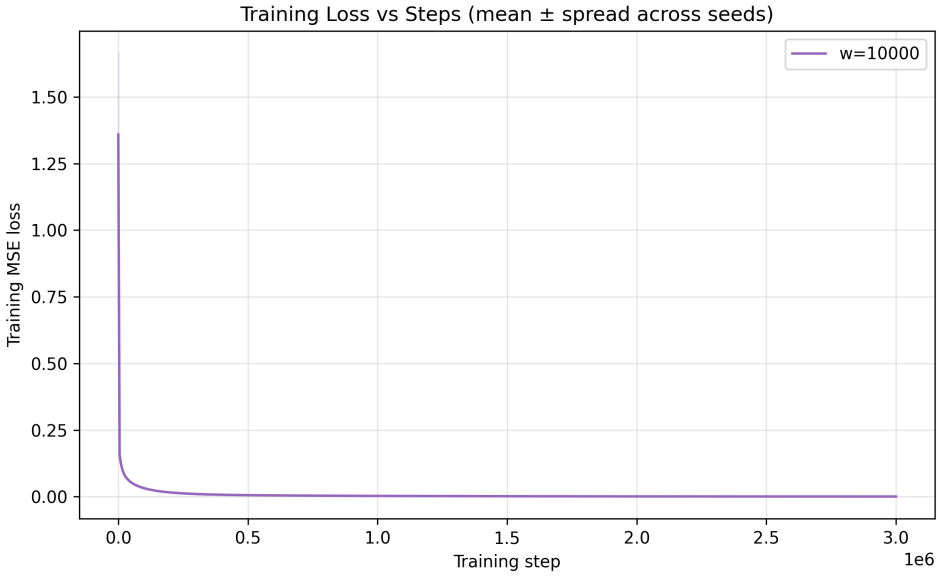

Figure 11: Loss curves for high widths.

2 Kernel Regression Behavior

2.1 Kernel Regression Solution at Freeze Time

Once a freeze time t^* is detected, we construct the kernel regression solution using the frozen empirical NTK:

$$\alpha(t^*) = \left(K_{\text{train,train}}(t^*) + \lambda I\right)^{-1} y_{\text{train}},$$

and the corresponding evaluation-grid predictions:

$$f_{\text{KR}}(\gamma; t^*) = K_{\text{eval,train}}(t^*)\, \alpha(t^*).$$

This yields the function predicted by exact kernel regression at the freeze-time NTK.

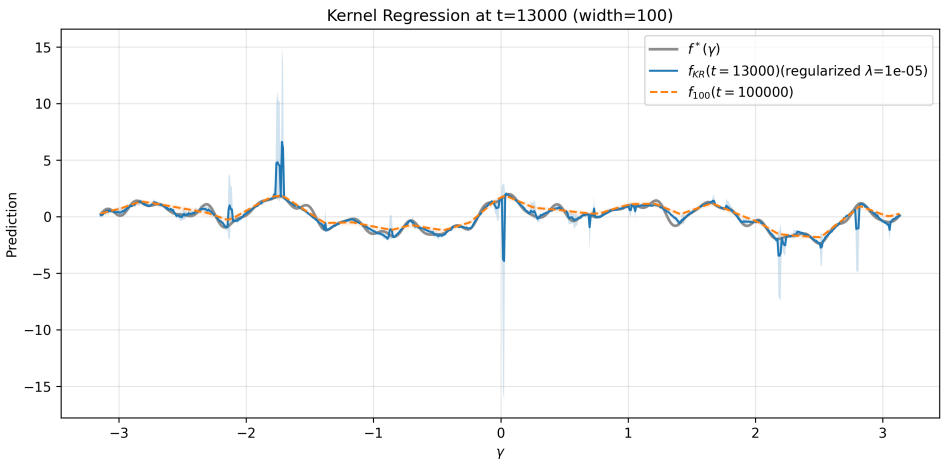

Figure 12: Kernel regression prediction using frozen NTK (width 100).

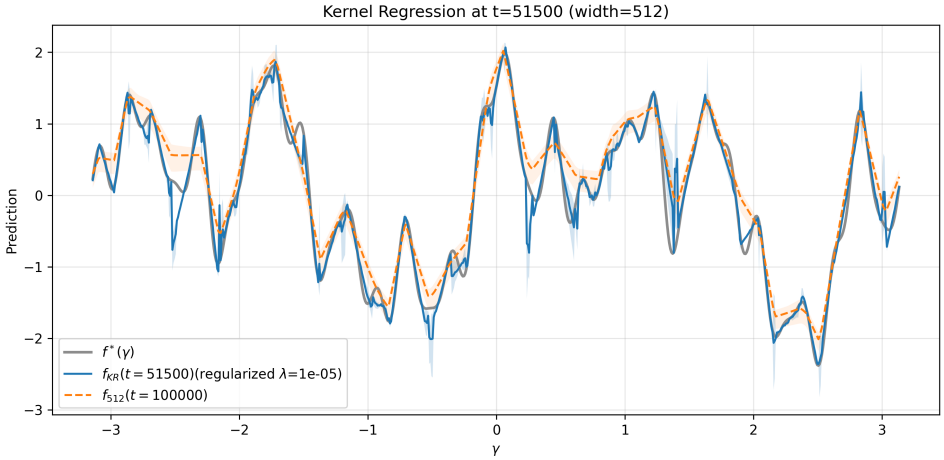

Figure 13: Kernel regression prediction using frozen NTK (width 512).

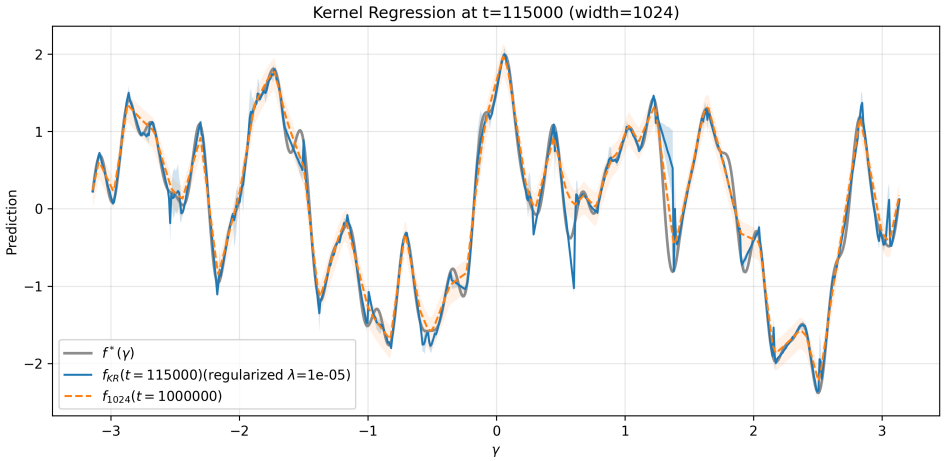

Figure 14: Kernel regression prediction using frozen NTK (width 1024).

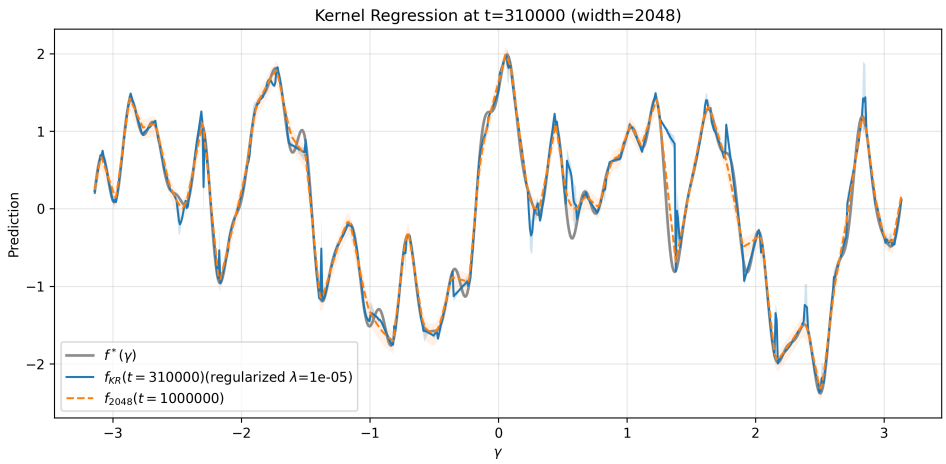

Figure 15: Kernel regression prediction using frozen NTK (width 2048).

2.2 Comparison to Gradient-Descent Predictions

In this section we examine how the network's predictions evolve *after* the NTK has effectively frozen.

For each width, we plot:

1. the network prediction at the NTK freeze time t^*:

$$f_{\text{net}}(\gamma, t^*),$$

2. the network prediction at the end of training:

$$f_{\text{net}}(\gamma, t_{\text{final}}).$$

This comparison shows whether the network continues to change significantly in function space after the NTK has stopped evolving.

If the NTK has truly entered the lazy regime, we expect the function to move very little after t^*.

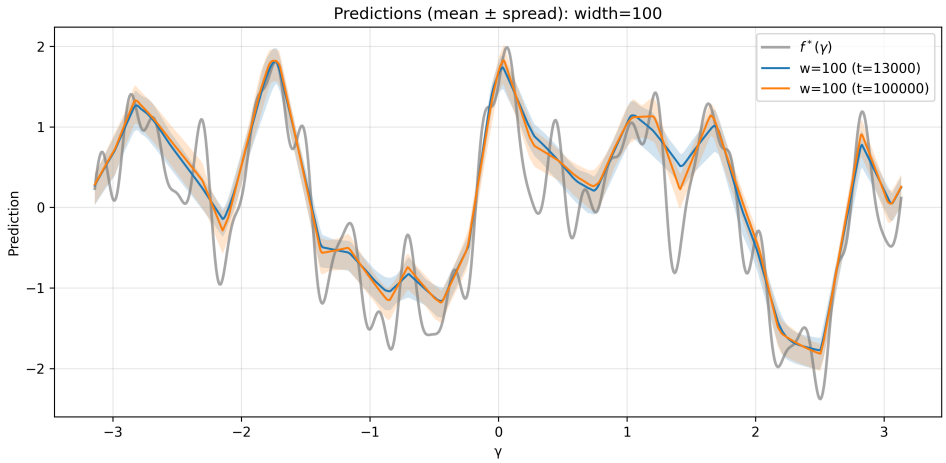

Figure 16: Network predictions at freeze time vs. final time (width 100).

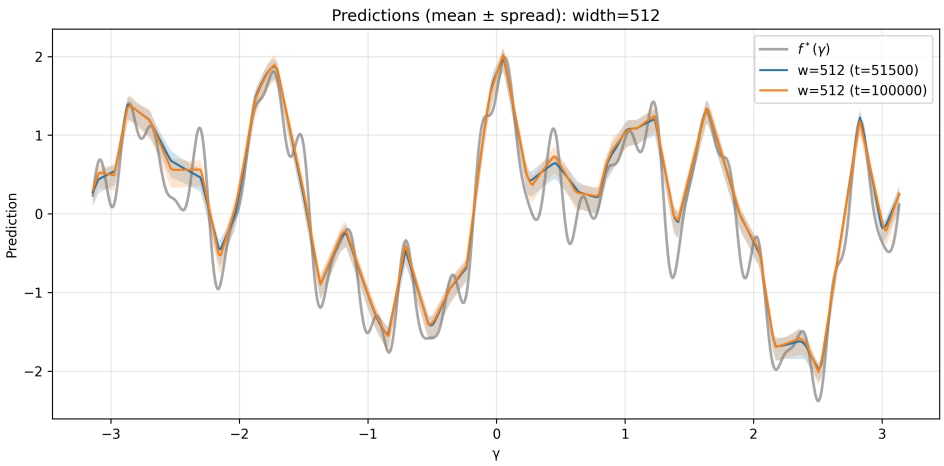

Figure 17: Network predictions at freeze time vs. final time (width 512).

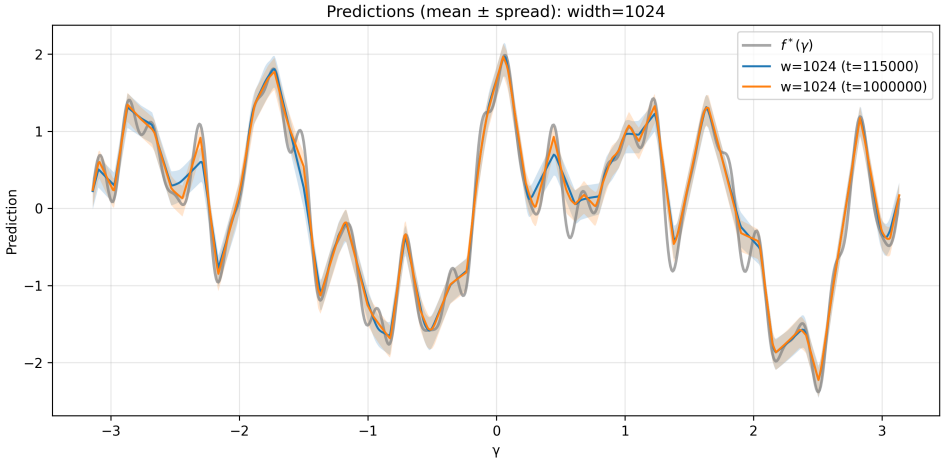

Figure 18: Network predictions at freeze time vs. final time (width 1024).

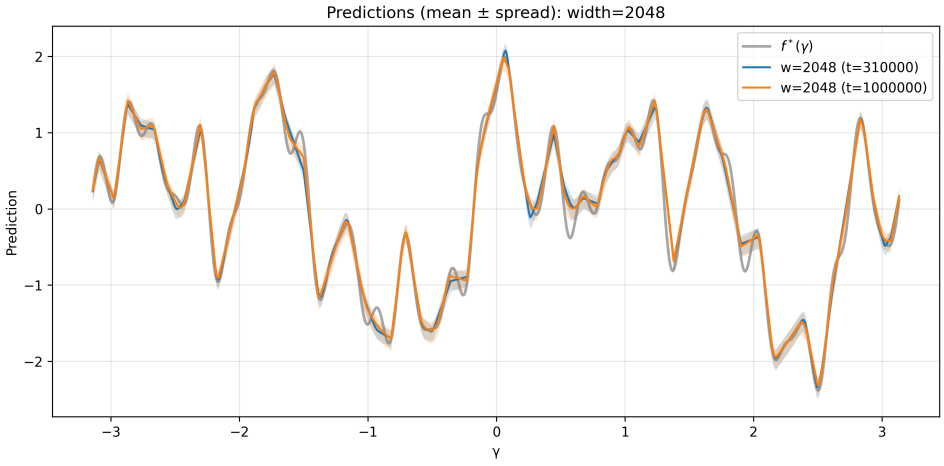

Figure 19: Network predictions at freeze time vs. final time (width 2048).

3 Summary

- **Freeze time vs. width:** Wider networks freeze *later*. Narrow networks reach a small drift slope very early, while the widest networks ($W = 10000$) do not satisfy the freeze criterion even after several million steps.
- **Eigenvalue stabilization timescales:** All widths saturate to similar top eigenvalue values, but the rate of saturation differs strongly: narrow networks converge to their eigenvalue plateau quickly, whereas wide networks saturate far more slowly.
- **Kernel regression vs. target accuracy:** Kernel regression behaves differently across widths:
  - For **narrow networks**, the kernel-regression prediction computed at the freeze time is *closer to the target function* than the final trained network. In this regime, the finite-width network drifts away from the KR solution as training progresses.
  - For **wide networks**, the *final trained network* is closer to the target than the kernel regression solution. Kernel regression underperforms relative to gradient descent at large width.

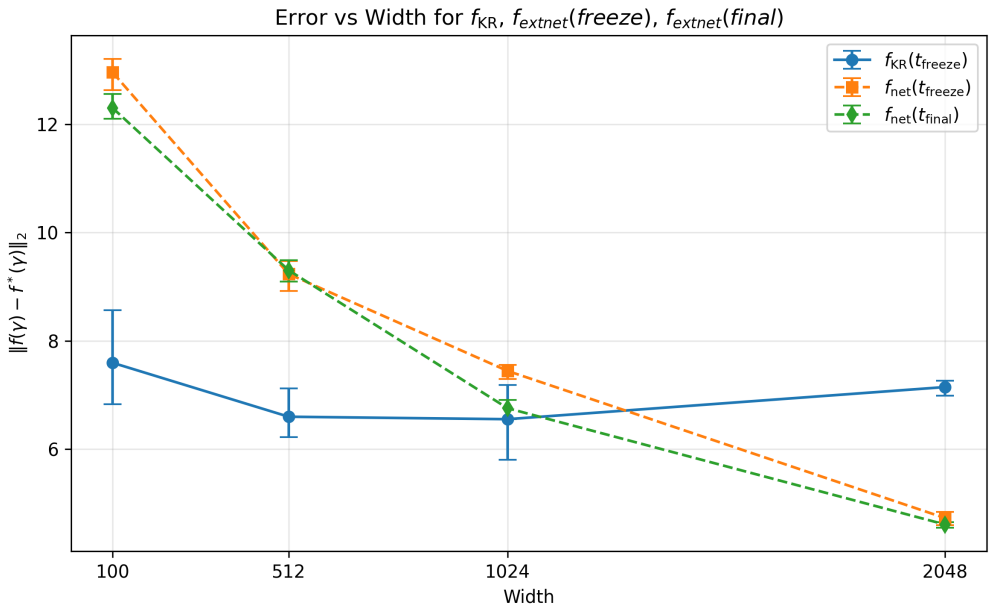

Figure 20: Kernel regression error and final network error versus width. Narrow networks show lower kernel-regression error than final-network error, while for wide networks the situation reverses.