Reactivation: Empirical NTK Dynamics Under Task Shifts

Yuzhi Liu[†] *
Zixuan Chen[‡] *
Zirui Zhang[‡] *
Yufei Liu[‡] *
Giulia Lanzillotta[‡]
[†]*Department of Mathematics, ETH Zurich, Switzerland*
[‡]*Department of Computer Science, ETH Zurich, Switzerland*

Abstract

The Neural Tangent Kernel (NTK) offers a powerful tool to study the functional dynamics of neural networks. In the so-called lazy, or kernel regime, the NTK remains static during training and the network function is linear in the static neural tangents feature space. The evolution of the NTK during training is necessary for feature learning, a key driver of deep learning success. The study of the NTK dynamics has led to several critical discoveries in recent years, in generalization and scaling behaviours. However, this body of work has been limited to the single task setting, where the data distribution is assumed constant over time. In this work, we present a comprehensive empirical analysis of NTK dynamics in continual learning, where the data distribution shifts over time. Our findings highlight continual learning as a rich and underutilized testbed for probing the dynamics of neural training. At the same time, they challenge the validity of static-kernel approximations in theoretical treatments of continual learning, even at large scale.

1. Introduction

Continual learning is central to real-world applications where models must learn from a stream of tasks without retraining from scratch or forgetting previous knowledge. While architectural and algorithmic advances have improved performance in such settings, our theoretical understanding of how and why continual learning works remains limited. In particular, the dominant tools for analyzing neural network learning dynamics—developed primarily for stationary, single-task settings—may not capture the behaviors that emerge when data distributions shift over time.

The study of learning dynamics in neural networks seeks to characterize how model parameters, internal representations, and predictions evolve throughout training. This perspective reveals the implicit biases of gradient-based optimization and has helped explain phenomena such as generalization, feature learning, and convergence.

In recent years, powerful tools have emerged for analyzing learning dynamics. The Neural Tangent Kernel (NTK) framework [8], for example, shows that infinitely wide neural networks behave like kernel machines with fixed kernels throughout training. However, practical networks typically operate in finite-width regimes, where the NTK evolves during training, enabling feature learning. Fort et al. [6] and Zhou et al. [16] empirically demonstrated that NTKs in realistic architectures undergo substantial change at early training stages, correlating with stronger representation learning and improved performance. This has led to a conceptual dichotomy between the **lazy**

* Equal contribution. Email {yuzliu, chenzixu, zirzhang}@student.ethz.ch, yufei.liu@inf.ethz.ch

© Y. Liu, Z. Chen, Z. Zhang, Y. Liu & G. Lanzillotta.

(kernel) regime—where internal representations remain fixed—and the **rich (feature learning)** regime—where features evolve to adapt to data structure.

Beyond the lazy/rich distinction, recent studies have explored how NTKs evolve. A key phenomenon is *kernel alignment*, where NTK eigenvectors align with task-relevant directions over time, improving learning efficiency and generalization [1, 13]. In parallel, works on loss landscape geometry reveal *progressive sharpening*—a rise in curvature early in training—linked to changes in the NTK through its connection to the Hessian spectrum [3, 10].

Bringing this lens to continual and sequential learning holds significant promise. Understanding the mechanisms that drive interference and forgetting—potentially from minimal assumptions about task similarity— could illuminate their root causes and guide the design of algorithms that are both more robust and more efficient. Some theoretical works have modeled continual learning in the lazy regime, where NTKs remain static [2, 5, 11]. While analytically convenient, these assumptions overlook the dynamic, evolving nature of representations in realistic networks, leaving a gap between theory and observed behavior.

A deeper challenge is that most existing theory assumes *stationarity*, i.e., that training data is drawn from a fixed distribution. This assumption breaks down in continual learning, where tasks and data distributions shift over time. This raises a central and largely unanswered question:

> *Do existing theories of learning dynamics—particularly those based on the NTK—extend meaningfully to non-stationary environments?*

1.1. Contributions

This work provides a systematic, empirical investigation of Neural Tangent Kernel (NTK) dynamics in the context of continual learning—a setting that challenges the conventional assumption of stationary data distributions. Our contributions are as follows:

1. We evaluate how NTK dynamics of past data respond to changes in network width, learning rate, training duration, and critically, task similarity, across single and multiple task switches.

2. We demonstrate that task transitions consistently trigger abrupt shifts in the NTK of past data, even under a lazy learning regime, revealing a reactivation of feature learning at each task boundary.

3. Through controlled experiments, we distinguish between different types of distributional shifts, showing that the introduction of semantically novel classes leads to significantly greater NTK change.

By systematically characterizing the evolution of NTKs in non-stationary regimes, our experiments provide compelling evidence that challenges the static distribution assumption foundational to the original NTK theory and many previous theoretical works. Indeed, we observe that even in networks with relatively large width, the NTK of past data can undergo abrupt changes when exposed to data from a new distribution—a phenomenon we term *re-activation*. These findings not only reveal critical limitations in existing theoretical frameworks in continual learning but also open up a promising and underexplored direction: developing theories that explicitly account for data distribution shifts. Our work lays the groundwork for advancing continual learning and offers valuable insights into the training dynamics of neural networks.

2. Experiments and Results

2.1. Metrics used and parametrization.

We review the definition and some fundamental ideas related to the Neural Tangent Kernel in Appendix A. Here, we introduce the main metrics used in our experiments. We run image classification experiments on CIFAR and ImageNet across multiple seeds. More training details can be found in Appendix B. In the following sections we present results from experiments involving two sequential tasks, and in Appendix C, we report results on more task sequences. Crucially, all metrics are evaluated solely on the data from the first task throughout the continual experiment.

Kernel Spectral Norm It is equivalent to the max eigenvalue of NTK. We show in Appendix A.1 that the NTK spectral norm controls the convergence rate in certain eigenmodes.

Kernel Distance Following Cortes et al. [4], Fort et al. [6], Kornblith et al. [12], we define the kernel distance based on Centered Kernel Alignment $\mathrm{CKA}(\cdot,\cdot)$ (definition in the Appendix A.2) as $S(\Theta,\Theta') \triangleq 1 - \mathrm{CKA}(\Theta,\Theta')$.

Kernel Velocity The kernel velocity $v(t) \triangleq S\big(\Theta_t, \Theta_{t+dt}\big)/dt$ quantifies the rate of change of NTKs at time t.

Kernel Alignment The kernel alignment $A(t) \triangleq \mathrm{CKA}(\Theta_t, \mathbf{y}\mathbf{y}^\top)$[4] measures the similarity between the NTK and the target label kernel $\mathbf{y}\mathbf{y}^\top$ at time t, where $\mathbf{y}$ is the label vector.

Parametrization We consider the *Pytorch standard parametrization* in our experiments, which we operate under two regimes: a lazy learning regime, obtained by scaling the learning rate inversely with the width and a feature learning regime, for which we employ the *Kaiming uniform* initialization, which reflects common practice in continual learning. We review in detail the parametrization used in Appendix B.1.

2.2. Task Shifts Reactivate Learning Dynamics

Scaling up neural networks—along with appropriate learning rate rescaling [14]—is known to induce the lazy regime, in which training occurs in a nearly linear function space and the network's internal features remain effectively static. We confirm this behavior on the first task: increasing model width yields more static NTK, as indicated by reduced kernel distance and velocity during training (Figure 1-Column 2; Appendix Figure 12). However, at the moment of task switch, we observe a clear and consistent spike in kernel velocity, signaling a (temporary) departure from the lazy regime. The network briefly enters a dynamic phase of feature adaptation before quickly returning to stability. We refer to this phenomenon as the **re-activation** of feature learning.

This reactivation is accompanied by a sharp drop in the NTK norm of the old task at the onset of the new task, followed by a gradual recovery. This creates a distinctive asymmetric V-shape or "check-mark" trajectory in the NTK norm that we observed consistently across all model widths. The timing of this drop aligns with the spike in velocity, suggesting a rapid reconfiguration of the network's functional representation in response to the new task. Interestingly, we observe the same shape in the feature learning regime, which is closer to common practice. Similar patterns are observed in Kernel Alignment (Figure 1-Column 3), indicating that the NTK rapidly changes direction at the task switch and begins evolving along a new direction. We also find that these patterns persist for multiple task switches, as shown in Figure 10 in the appendix.

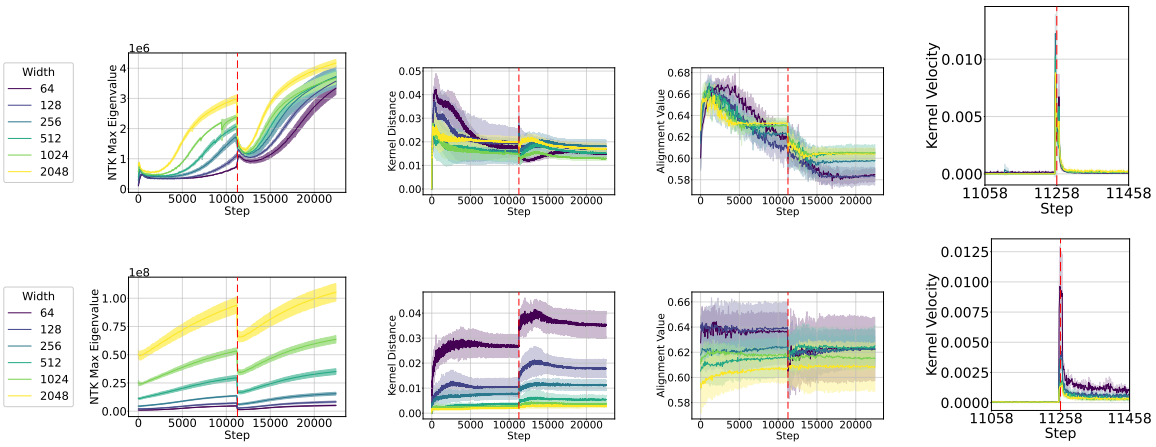


Figure 1: Comparison of NTK Max Eigenvalue, Kernel Distance from initialization, Kernel Alignment and Kernel velocity across different widths with (Row 1) and without (Row 2) feature learning. The measurements are done on the first task before and after the task switch, with 5 classes in each task. One step corresponds to 10 iterations during training.

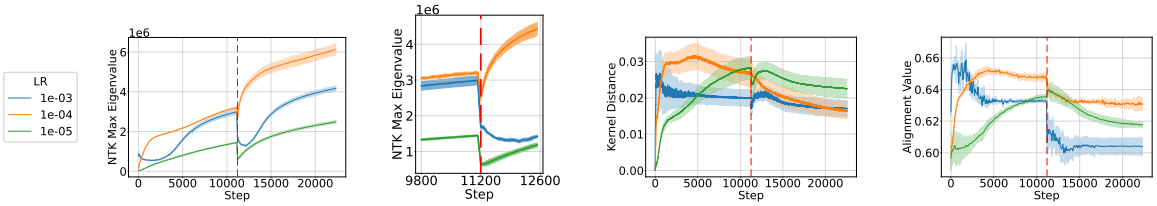


Figure 2: Comparison of NTK Max Eigenvalue, Kernel Distance from initialization, Kernel Alignment and Kernel velocity across different learning rates at fixed width ($N = 2048$).

We find that the behavior at task switch critically depends on the learning rate. Notably, the checkmark shape is more visible for higher learning rates, while for lower learning rates, it is collapsed on the first few steps after the task switch. Moreover, the network recovers from the disruption fastest when using the intermediate learning rate. This difference may stem from the training phase at the task switch: with the smallest learning rate, the network has not yet fully captured general patterns, whereas with the largest learning rate, it has already begun to overfit. Both situations result in a longer recovery time.

Remarks. It is not obvious that the tangent kernel of the first task should evolve according to a structured pattern as observed. Firstly, prior to this work, it was not clear that the NTK would evolve at all under a lazy learning regime, and several theoretical studies made the unsubstantiated assumption that the lazy regime can be maintained under distribution shifts. Secondly, the robustness of these patterns across architectures and hyperparameters points to the existence of a shared underlying mechanism governing task transitions. This mechanism appears to be modulated by hyperparameters such as network width and learning rate, suggesting a unified framework for characterizing adaptation in continual learning.

2.3. Task Similarity Controls NTK Dynamics

A task switch introduces a shift in the data distribution, which cause the reactivation of feature learning. However, "distribution shift" is a generic term that applies to many different scenarios. In particular, we consider two specific cases of distribution shift: the introduction of new classes, and the change of the relative frequencies of a set of known classes. All the following experiments are conducted in the feature learning regime. Further details can be found in Appendix B.

In the first case, for each experiment E_i, the network is trained on distribution $\mathcal{D}_0$ in task 1 and $\mathcal{D}_i$ in task 2, where $\mathcal{D}_k$ denotes a uniform mixture over 10 classes $\{k, k+1, \ldots, k+9\}$. Thus the similarity between $\mathcal{D}_0$ and $\mathcal{D}_i$ can be measured as the overlap between the classes:

$$\text{Similarity}(\mathcal{D}_0, \mathcal{D}_i) = \frac{|\mathcal{D}_0 \cap \mathcal{D}_i|}{|\mathcal{D}_0 \cup \mathcal{D}_i|}.$$

By varying $i \in [0, 1]$, we sweep the similarity between 1 (identical tasks) and 0 (no class overlap, a typical benchmark for continual learning).

In the second case, we define two disjoint class subsets $\tilde{\mathcal{D}}_0$ and $\tilde{\mathcal{D}}_1$, and interpolate between them with mixtures: $\tilde{\mathcal{D}}_\alpha = (1-\alpha)\tilde{\mathcal{D}}_0 + \alpha\tilde{\mathcal{D}}_1$. For each experiment $\tilde{E}_\alpha$ the network is trained on $\tilde{\mathcal{D}}_{0.1}$ in task 1 and $\tilde{\mathcal{D}}_\alpha$ in task 2, varying $\alpha \in [0.1, 0.9]$. The similarity metric is linear:

$$\text{Similarity}(\tilde{\mathcal{D}}_\alpha, \tilde{\mathcal{D}}_\beta) = 1 - |\alpha - \beta|.$$

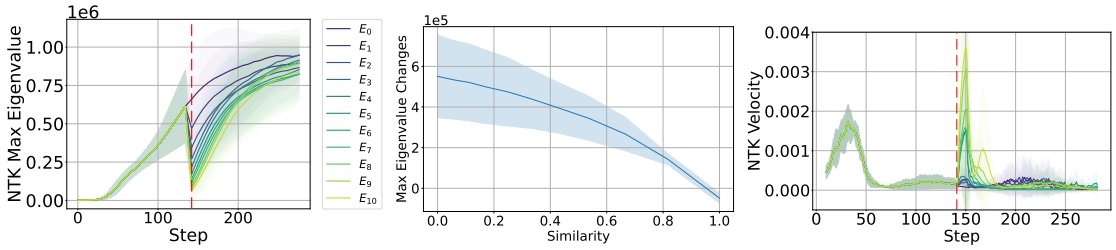


Figure 3: NTK Dynamics of the first task with concept-based distribution shift.

When the new task introduces new concepts (experiments $E_0 \ldots, E_{10}$), there is a direct relationship between the number of new classes and the amount of change in the NTK, as confirmed in the measurement of NTK norm, velocity and kernel distance (Figure 3, Figure 6). We observe the characteristic check-mark shape in all but the E_0 case, where no distributional change occurs (Figure 3-1). The drop in NTK norm becomes progressively smaller as class overlap increases, revealing a clear monotonic relationship between task similarity and the magnitude of NTK disruption.

Again, the NTK norm recovers gradually after the drop, consistent across all levels of similarity (Figure 3-1). A similar trend is observed in the kernel distance (Figure 6), where larger distribution shifts cause more pronounced deviations from the previous NTK state. The trend is neatly ordered by task similarity, suggesting the existence of an underlying law governing the NTK spectral evolution, parametrized by the task similarity. Further, the kernel velocity (Figure 3-3) confirms that most of the feature learning occurs immediately following the task switch, after which the network appears to settle back into a more stable regime within a few epochs.

We also note a diminishing return effect: the introduction of the first few new classes causes disproportionately large changes in NTK, while later additions have more incremental impact—suggesting a sublinear relationship between the number of new concepts and NTK disruption.

The picture is very different if the new task does not introduce new classes, as in the experiments E_α with $\alpha \in [0, 1]$. Figure 4 shows that NTK changes in this case are significantly smaller than in Experiment 1. The NTK norm (Figure 4-1) evolves smoothly without any sharp discontinuity at the task switch. Likewise, the kernel velocity (Figure 4-3) remains low, indicating that feature reactivation does not occur in response to proportion shifts alone. Although some monotonic trends are still visible in the NTK eigenvalues (Figure 4-2), their scale is minor.

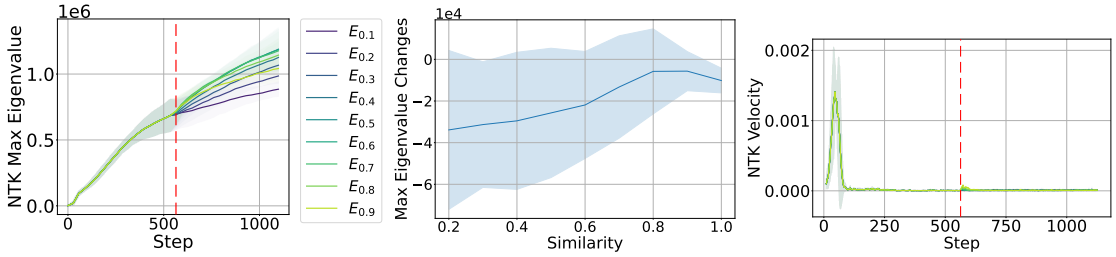


Figure 4: NTK Dynamics of the first task with frequency-based distribution shift.

Together, these results suggest that NTK dynamics encode rich information about task transitions, inviting a deeper theoretical understanding of reactivation phenomena in continual learning.

3. Discussion and Conclusion

We highlight three key observations from our study:

Continual learning disrupts kernel stationarity. Even in the lazy regime, task switches consistently trigger transient reactivation of feature learning. NTK metrics—including norm, velocity, and alignment—show sharp deviations at task transitions, indicating a temporary departure from static representations. This phenomenon is robust across widths and learning rates, challenging the assumption that wide networks behave as fixed-kernel learners in non-stationary settings.

Task similarity modulates reactivation. The extent of reactivation depends strongly on the semantic relationship between tasks. When new tasks introduce disjoint class content, NTKs shift significantly. In contrast, changes in class proportions without new concepts lead to minimal NTK disruption. This suggests that semantic overlap could be used as a signal to anticipate representation change.

NTK theory is limited in non-stationary settings. Classical NTK analyses rely on stationarity assumptions that fail in continual learning. Our results show that NTKs evolve in response to new data distributions, even at large widths. This motivates a theoretical shift: modeling NTK dynamics under distribution shift is essential for understanding continual adaptation and forgetting.

In conclusion, our work reveals a structured departure from lazy dynamics at task boundaries—a phenomenon we term *reactivation*. Future theory should account for these transitions, and practical algorithms might even benefit from explicitly managing them. We hope our findings motivate further research into the dynamics of neural networks in non-stationary environments.

References

[1] Aristide Baratin, Thomas George, César Laurent, R Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In *International Conference on Artificial Intelligence and Statistics*, pages 2269–2277. PMLR, 2021.

[2] Mehdi Abbana Bennani, Thang Doan, and Masashi Sugiyama. Generalisation guarantees for continual learning with orthogonal gradient descent. *arXiv preprint arXiv:2006.11942*, 2020.

[3] Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jh-rTtvkGeM.

[4] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13(1):795–828, 2012.

[5] Thang Doan, Mehdi Bennani, Bogdan Mazoure, Guillaume Rabusseau, and Pierre Alquier. A theoretical analysis of catastrophic forgetting through the ntk overlap matrix, 2021. URL https://arxiv.org/abs/2010.04003.

[6] Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:5850–5861, 2020.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[8] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf.

[9] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31: 8571–8580, 2018.

[10] Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho*, and Krzysztof Geras*. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1g87C4KwB.

[11] Ryo Karakida and Shotaro Akaho. Learning curves for continual learning in neural networks: Self-knowledge transfer and forgetting, 2022. URL https://arxiv.org/abs/2112.01653.

[12] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR, 2019.

[13] Haozhe Shan and Blake Bordelon. A theory of neural tangent kernel alignment and its influence on training. *arXiv preprint arXiv:2105.14301*, 2021.

[14] Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture, 2020. URL https://arxiv.org/abs/2006.14548.

[15] Greg Yang and Edward J. Hu. Feature Learning in Infinite-Width Neural Networks, July 2022. URL http://arxiv.org/abs/2011.14522. arXiv:2011.14522 [cs].

[16] Zhanpeng Zhou, Yongyi Yang, Jie Ren, Mahito Sugiyama, and Junchi Yan. On the cone effect in the learning dynamics, 2025. URL https://arxiv.org/abs/2503.16316.

Appendix A. The NTK Framework and NTK Spectrum

Consider a dataset $\{\mathbf{x}_i\}_{i=1}^n$ with real targets $\{y_i\}_{i=1}^n$ and loss function ℓ. Let $f_t := f(\theta_t)$ be the neural network at time t, with parameters θ. By gradient descent, in continuous time, the parameters evolve as:

$$\partial_t \theta(t) = -\nabla_\theta \ell = -\frac{1}{n}\sum_{i=1}^{n} \frac{\partial \ell}{\partial f_t(\mathbf{x_i})} \cdot \frac{\partial f_t(\mathbf{x_i})}{\partial \theta_t} \tag{1}$$

Hence, by chain rule the network function space is determined by the *Neural Tangent Kernel* $\Theta_t(\mathbf{x}_i, \mathbf{x}_j) = \nabla_\theta f_t(\mathbf{x}_i)^\top \nabla_\theta f_t(\mathbf{x}_j)$:

$$\partial_t f_t(\mathbf{x}) = \frac{\partial f_t}{\partial \theta_t} \cdot \partial_t \theta_t = -\frac{1}{n}\sum_{i=1}^{n} \Theta_t(\mathbf{x}, \mathbf{x}_i) \frac{\partial \ell}{\partial f_t(\mathbf{x_i})} \tag{2}$$

A.1. Eigenvalues of NTK

Define the function output on dataset as $\mathbf{f} = [f(\mathbf{x_1}), f(\mathbf{x_2}), ..., f(\mathbf{x_n})]^\top$ and define the residuals as $\mathbf{e} = \mathbf{f} - \mathbf{y}$. For squared loss, the evolution of $\mathbf{f}$ and $\mathbf{e}$ at each time step is:

$$\mathbf{e}_{t+1} = (\mathbf{I} - \eta \Theta_t)\mathbf{e}_t, \tag{3}$$

We diagonalize the NTK as $\Theta = Q\Lambda Q^\top$ and project $\mathbf{e}$ onto the eigenbasis Q:

$$\tilde{\mathbf{e}}_{t+1} = (\mathbf{I} - \eta\Lambda)\tilde{\mathbf{e}}_t, \quad \tilde{e}^i_{t+1} = (1 - \eta\lambda_i)\tilde{e}^i_t \quad \text{for each eigenmode } i \tag{4}$$

Thus, the error in each eigenmode decays at a rate determined by the corresponding eigenvalue λ_i, indicating that the NTK governs the learning speed of each mode based on its eigenvalue. An higher NTK norm thus corresponds to faster convergence in some eigenmodes.

A.2. Centered Kernel Alignment

Centered Kernel Alignment (CKA) is a statistical similarity measure used to compare the structure of two kernels, typically in the context of neural network analysis.

The (linear) CKA between A and B is defined as:

$$\mathrm{CKA}(A, B) = \frac{\langle A, B\rangle_F}{\|A\|_F \|B\|_F} = \frac{\mathrm{Tr}(A\top B)}{\sqrt{\mathrm{Tr}(A^\top A)}\sqrt{\mathrm{Tr}(B^\top B)}}$$

Here, $\langle A, B\rangle_F = \mathrm{Tr}(A^\top B)$ denotes the Frobenius inner product, and $\|A\|_F = \sqrt{\mathrm{Tr}(A^\top A)}$ is the Frobenius norm. A high CKA value (close to 1) indicates that the two kernels exhibit similar internal structure.

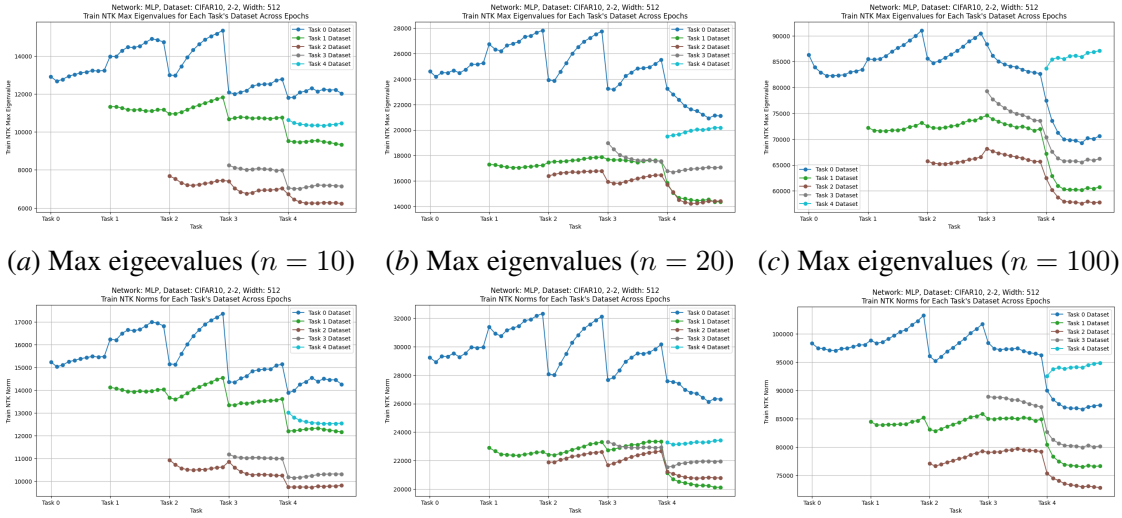
A.3. Computation of NTK

Neural Tangent Kernel (NTK) matrices are computed based on Yang [14]. The gradients of the trained models are cleared and a single sample is fed through the network while recording the gradient. This process is repeated for multiple samples and the NTK matrix is computed by taking the inner product between all the gradients.

We empirically evaluated the impact of sample size on the computation of the empirical NTK as a sanity check, alongside ablation studies. Specifically, we tested three different sample sizes (10,

20, and 100). The results (Fig. 5) indicate that while the NTK norm increases linearly with sample size, the overall trend of the NTK norm remains consistent across all sample sizes.

For all our main experiments, we use a batch size of 32 random samples to compute the empirical NTK. This choice is sufficient to reliably capture the trend of the NTK throughout the training process.



(a) Max eigeevalues ($n = 10$) (b) Max eigenvalues ($n = 20$) (c) Max eigenvalues ($n = 100$)

(d) NTK Norm ($n = 10$) (e) NTK Norm ($n = 20$) (f) NTK Norm ($n = 100$)

Figure 5: NTK dynamics computed with different sample sizes (n).

Appendix B. Experiment Details

B.1. Network parametrization.

The NTK parametrization induces lazy learning dynamics as the width of the network increases to infinity [9, 15]. For a network of width N, the weights are initialized according to $W_{ij}^{(l)} \sim \mathcal{N}(0, 1)$ and rescaled by defining the training parameters to be $\omega_{ij}^{l} = \frac{1}{\sqrt{N}} W_{ij}^{(l)}$. To mimic the NTK parametrization, we adopt the Kaiming Normal initialization and scale the learning rate with respect to width with $1/N$. The learning rate re-scaling has been shown to be essential in order to produce a valid limit as the width increases, and avoid the blow-up in the network dynamics [15]. Kaiming Normal initialization draws weights from a zero-mean normal distribution with $W_{ij} \sim \mathcal{N}(0, 2/n_{in})$, where n_{in} is the number of input connections to the neuron, proportional to N in our settings [7]. Therefore, Kaiming Normal initialization with scaled learning rate $1/N$ is very similar to NTK parametrization with only factor-wise difference.

Unless stated otherwise, all the experiments are performed under feature learning regime, with Kaiming uniform initialization.

B.2. Task Shifts Experiments

To analyze learning dynamics in a continual learning setting, we train a simple Convolutional Neural Network (CNN) consisting of three convolutional layers, three pooling layers, and a fully connected

layer with ReLU activation functions for image classification on CIFAR and ImageNet. All experiments use the SGD optimizer and cross-entropy loss.

The classification problem on CIFAR-10 is split into two tasks, each containing five classes. We explore various settings, including CNN widths (64, 128, 256, 512, 1024, 2048), learning rates (1e-3, 1e-4, 1e-5), and training epochs (10, 20, 40, 80, 160). Here, the width of the CNN refers to the number of channels in the convolutional layers.

B.3. Task Similarity Experiments

We train a simple CNN using the same configuration described in Section B.2. Each experiment consists of two sequential tasks with varying degrees of similarity, where no data from the first task is reused in the second. The model with width 32 and 64 is trained for 20 epochs per task using stochastic gradient descent (SGD) with a constant learning rate of 1e-3.

Experiment 1: Gradual Shift with New Class Introduced. We define a family of input distributions $\mathcal{D}_i = \{i, i+1, \ldots, i+9\}$, where each $\mathcal{D}_i$ is a uniform mixture over 10 consecutive CIFAR-100 classes. In experiment E_i, we construct a two-task continual learning scenario: **Task 1** trains on $\mathcal{D}_0$ and **Task 2** trains on $\mathcal{D}_i$ for $i = 0, \ldots, 10$. The similarity between $\mathcal{D}_0$ and $\mathcal{D}_i$ is defined as:

$$\text{Similarity}(\mathcal{D}_0, \mathcal{D}_i) = \frac{|\mathcal{D}_0 \cap \mathcal{D}_i|}{|\mathcal{D}_0 \cup \mathcal{D}_i|}.$$

Experiment 2: Gradual Shift within Fixed Class Support. Define two disjoint sets of classes on CIFAR 10: $\tilde{\mathcal{D}}_0 = \{0, 1, 2, 3, 4\}$ and $\tilde{\mathcal{D}}_1 = \{5, 6, 7, 8, 9\}$, and construct a family of mixed distributions:

$$\tilde{\mathcal{D}}_\alpha = (1-\alpha)\tilde{\mathcal{D}}_0 + \alpha\tilde{\mathcal{D}}_1, \quad \alpha \in \{0.1, 0.2, \ldots, 0.9\}.$$

Here for each experiment $\tilde{E}_\alpha$, **Task 1** is fixed to learn from $\tilde{\mathcal{D}}_{0.1}$, and **Task 2** learns from $\tilde{\mathcal{D}}_\alpha$.

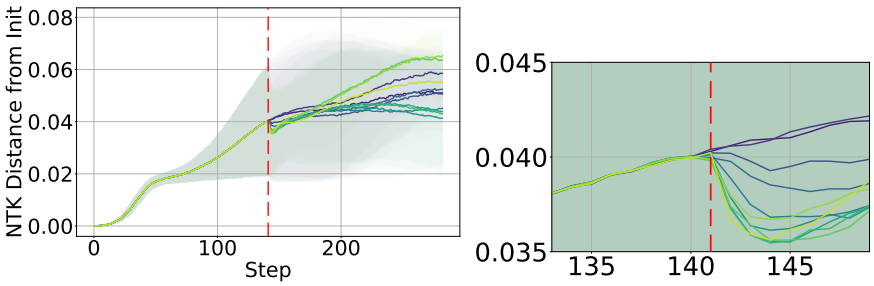


Figure 6: Kernel distance in Experiment 1 and a zoomed-in view around the task switch.

Appendix C. Complementary Results

C.1. Comprehensive Metrics evaluation for CIFAR10 (Two Tasks)

In Figures 7, 8, and 9, we show various metrics calculated during the CIFAR10 experiment as described in B.2.

C.2. Extension to Multiple Task Switches

To investigate whether the patterns persist during different task switches, we also perform experiments on 5 sequential tasks with 2 classes in each task on CIFAR-10 shown in Figure 10.

C.3. Experiments on ImageNet100

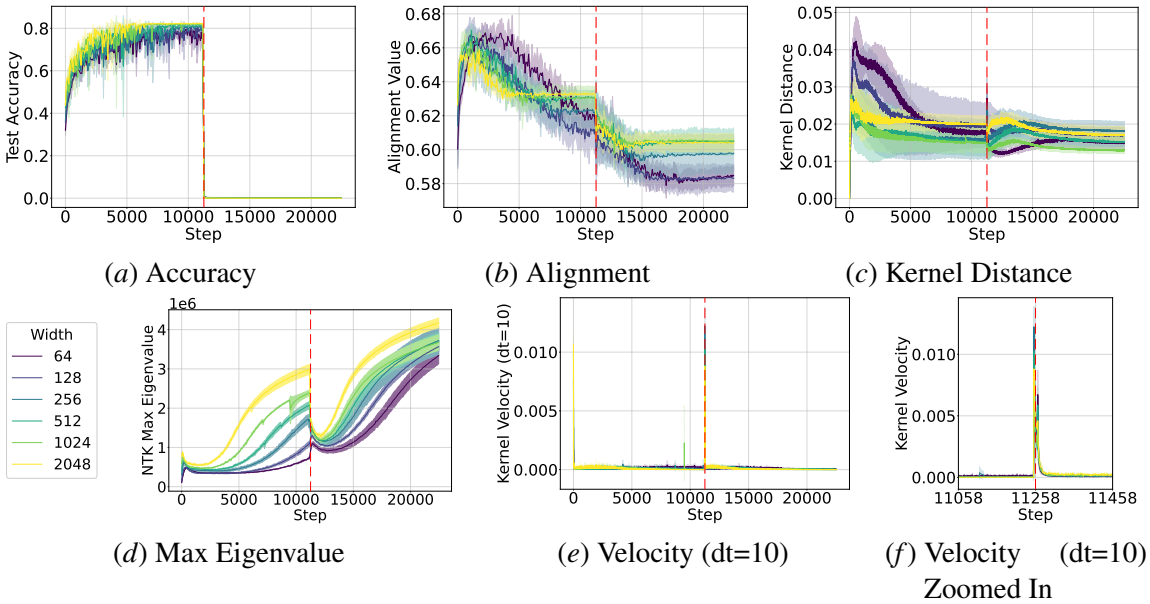
To further support our conclusions, we analyzed the evolution of the NTK spectrum on a larger dataset, ImageNet100. In Figure 11, we compare the effects of varying network width and the number of epochs per task on the NTK spectrum. Similarly to the experiments on CIFAR 10, our results show a significant change in NTK dynamics upon the arrival of new tasks. This change does get smoother as the network width expands, however it does not disapear. Also, no matter whether the old task is trained to full convergence (evaluated by the change in NTK), the re-activation upon task switch persists.

All experiments use a learning rate of 1×10^{-3} and SGD as the optimizer. Comparisons are made relative to the base setting: width 250 and 10 epochs per task.

C.4. Additional lazy regime results

The addition results presented in Figure 12 demonstrate that as the network width increases exponentially from 64 to 2048, the magnitude of changes in test accuracy (a), alignment (b), kernel distance (c), and the maximum eigenvalue of the NTK (d) decreases during task transitions in continual learning.

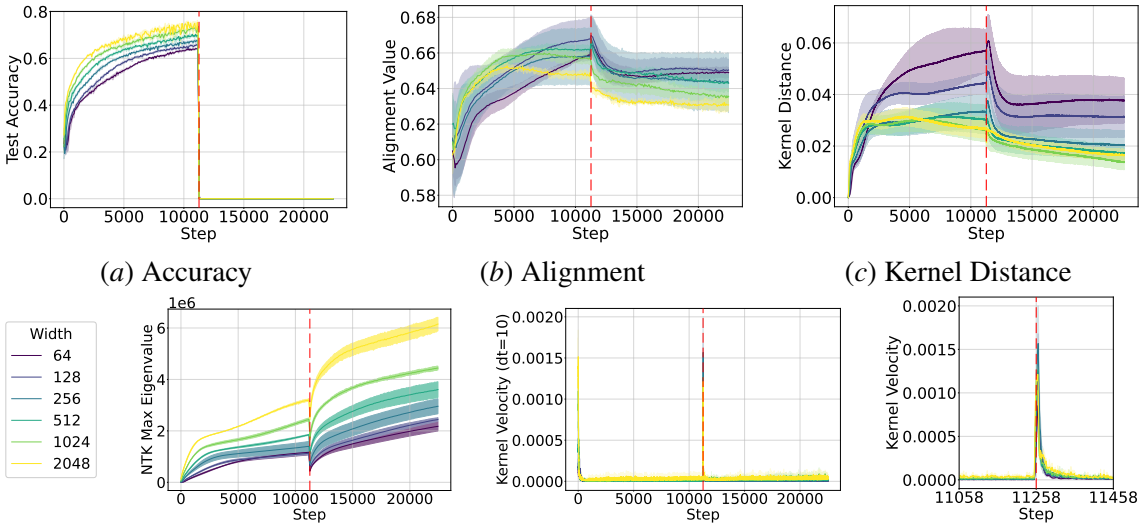
C.5. Figures



(*a*) Accuracy (*b*) Alignment (*c*) Kernel Distance

(*d*) Max Eigenvalue (*e*) Velocity (dt=10) (*f*) Velocity (dt=10) Zoomed In

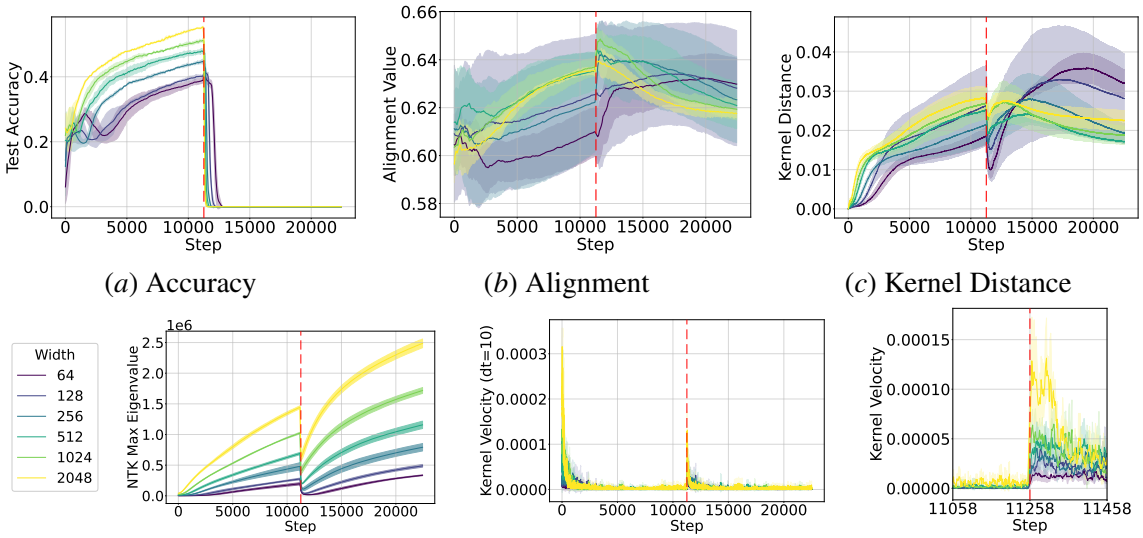
Figure 7: Comparison of different metrics across network widths for CNN trained on CIFAR10 with learning rate 0.001. The number of epochs per task is set to 160. (a) Test accuracy, (b) Alignment, (c) Kernel distance, (d) Maximum eigenvalue of NTK, (e) Kernel velocity with dt=10, (f) Kernel velocity (zoom in).



(*a*) Accuracy (*b*) Alignment (*c*) Kernel Distance

(*d*) Max Eigenvalue (*e*) Velocity (dt=10) (*f*) Velocity (dt=10) Zoomed In

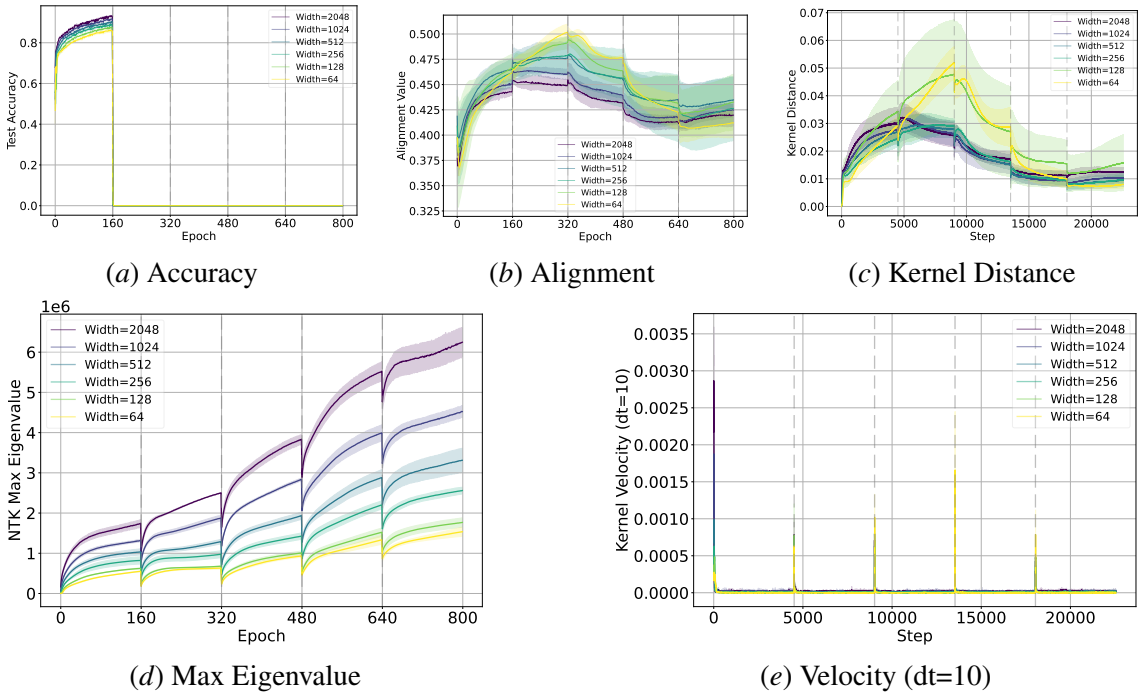
Figure 8: Comparison of different metrics across network widths for CNN trained on CIFAR10 with learning rate 0.0001. The number of epochs per task is set to 160. (a) Test accuracy, (b) Alignment, (c) Kernel distance, (d) Maximum eigenvalue of NTK, (e) Kernel velocity with dt=10, (f) Kernel velocity (zoom in).



(*a*) Accuracy (*b*) Alignment (*c*) Kernel Distance

(*d*) Max Eigenvalue (*e*) Velocity (dt=10) (*f*) Velocity (dt=10) Zoomed In

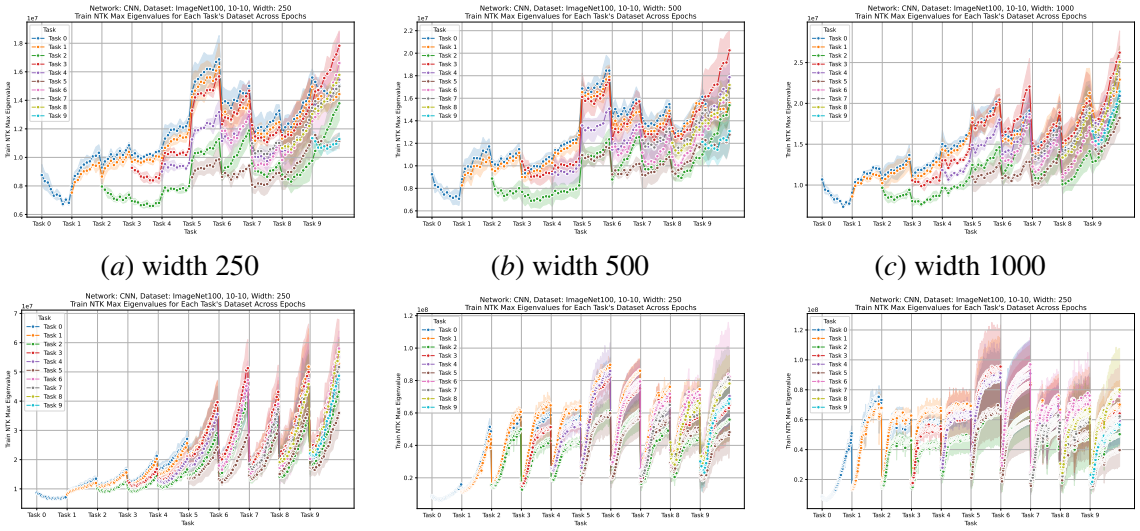
Figure 9: Comparison of different metrics across network widths for CNN trained on CIFAR10 with learning rate 0.00001. The number of epochs per task is set to 160. (a) Test accuracy, (b) Alignment, (c) Kernel distance, (d) Maximum eigenvalue of NTK, (e) Kernel velocity with dt=10, (f) Kernel velocity (zoom in).



(*a*) Accuracy (*b*) Alignment (*c*) Kernel Distance

(*d*) Max Eigenvalue (*e*) Velocity (dt=10)

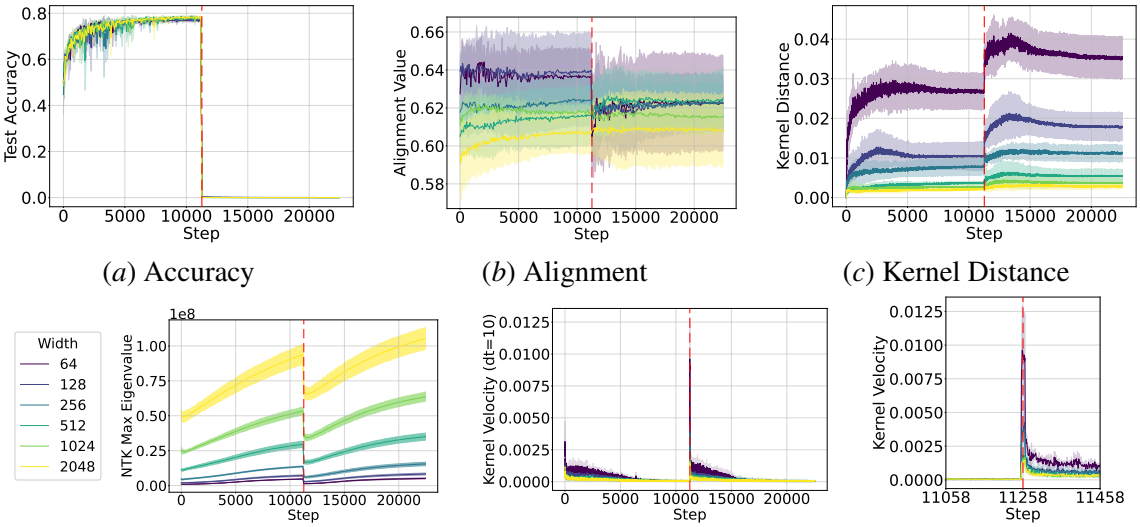
Figure 10: Comparison of different metrics across network widths for CNN trained on CIFAR10 during multiple task switches. The number of epochs per task is set to 160. (a) Test accuracy, (b) Alignment, (c) Kernel distance, (d) Maximum eigenvalue of NTK, (e) Kernel velocity with dt=10.



(*a*) width 250 (*b*) width 500 (*c*) width 1000

(*d*) 10 epochs (*e*) 20 epochs (*f*) 100 epochs

Figure 11: The effect of width and number of epochs per task on NTK spectrum on ImageNet100. The first row, left to right width 250, 500 and 1000 respectively, the second row, left to right epoch number per task 20, 50 and 100 respectively.



(*a*) Accuracy (*b*) Alignment (*c*) Kernel Distance

(*d*) Max Eigenvalue (*e*) Velocity (dt=10) (*f*) Velocity (dt=10) Zoomed In

Figure 12: Comparison of different metrics across network widths for CNN trained on CIFAR10 with NTK parametrization. The number of epochs per task is set to 160. (a) Test accuracy, (b) Alignment, (c) Kernel distance, (d) Maximum eigenvalue of NTK, (e) Kernel velocity with dt=10, (f) Kernel velocity (zoom in).