Finite Versus Infinite Neural Networks: an Empirical Study

Jaehoon Lee **Samuel S. Schoenholz*** **Jeffrey Pennington*** **Ben Adlam†***

Lechao Xiao* **Roman Novak*** **Jascha Sohl-Dickstein**

Google Brain

{jaehlee, schsam, jpennin, adlam, xlc, romann, jaschasd}@google.com

Abstract

We perform a careful, thorough, and large scale empirical study of the correspondence between wide neural networks and kernel methods. By doing so, we resolve a variety of open questions related to the study of infinitely wide neural networks. Our experimental results include: kernel methods outperform fully-connected finite-width networks, but underperform convolutional finite width networks; neural network Gaussian process (NNGP) kernels frequently outperform neural tangent (NT) kernels; centered and ensembled finite networks have reduced posterior variance and behave more similarly to infinite networks; weight decay and the use of a large learning rate break the correspondence between finite and infinite networks; the NTK parameterization outperforms the standard parameterization for finite width networks; diagonal regularization of kernels acts similarly to early stopping; floating point precision limits kernel performance beyond a critical dataset size; regularized ZCA whitening improves accuracy; finite network performance depends non-monotonically on width in ways not captured by double descent phenomena; equivariance of CNNs is only beneficial for narrow networks far from the kernel regime. Our experiments additionally motivate an improved layer-wise scaling for weight decay which improves generalization in finite-width networks. Finally, we develop improved best practices for using NNGP and NT kernels for prediction, including a novel ensembling technique. Using these best practices we achieve state-of-the-art results on CIFAR-10 classification for kernels corresponding to each architecture class we consider.

1 Introduction

A broad class of both Bayesian [1–17] and gradient descent trained [13–16, 18–29] neural networks converge to Gaussian Processes (GPs) or closely-related kernel methods as their intermediate layers are made infinitely wide The predictions of these infinite width networks are described by the Neural Network Gaussian Process (NNGP) [4, 5] kernel for Bayesian networks, and by the Neural Tangent Kernel (NTK) [18] and weight space linearization [24, 25] for gradient descent trained networks.

This correspondence has been key to recent breakthroughs in our understanding of neural networks [30–39] It has also enabled practical advances in kernel methods [8, 9, 15, 16, 26, 40–42], Bayesian deep learning [43–45], active learning [46], and semi-supervised learning [17]. The NNGP, NTK, and related large width limits [10, 30, 47–59] are unique in giving an exact theoretical description of large scale neural networks. Because of this, we believe they will continue to play a transformative role in deep learning theory.

*SSS, JP, BA, LX and RN contributed equally.

†Work done as a member of the Google AI Residency program (https://g.co/airesidency).

Preprint. Under review.

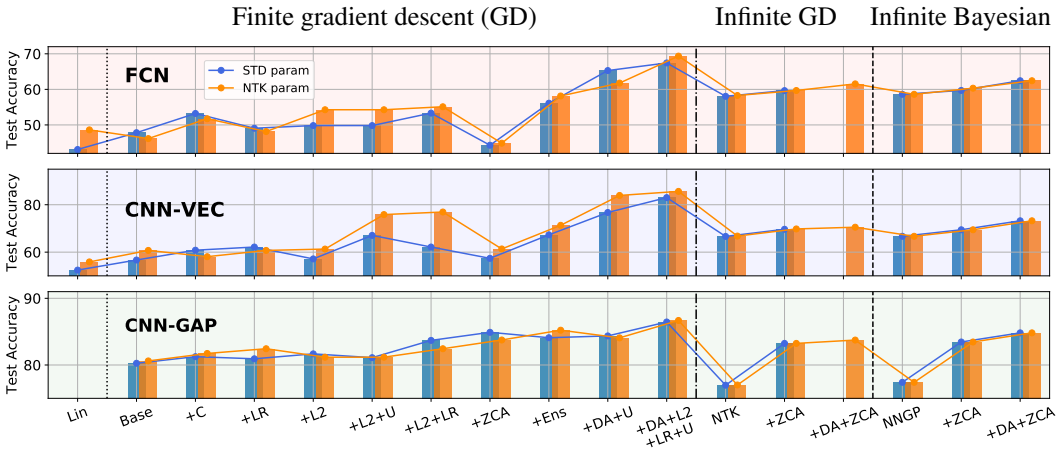

Figure 1: **CIFAR-10 test accuracy for finite and infinite networks and their variations**. Starting from the finite width `base` network of given architecture class described in §2, performance changes from **centering** (`+C`), **large learning rate** (`+LR`), allowing **underfitting** by early stopping (`+U`), input preprocessing with **ZCA regularization** (`+ZCA`), multiple initialization **ensembling** (`+Ens`), and some combinations are shown, for **Standard** and **NTK** parameterizations. The performance of the **linearized** (`lin`) base network is also shown. See Table S1 for precise values for each of these experiments, as well as for additional experimental conditions not shown here.

Infinite networks are a newly active field, and foundational empirical questions remain unanswered. In this work, we perform an extensive and in-depth empirical study of finite and infinite width neural networks. In so doing, we provide quantitative answers to questions about the factors of variation that drive performance in finite networks and kernel methods, uncover surprising new behaviors, and develop best practices that improve the performance of both finite and infinite width networks. We believe our results will both ground and motivate future work in wide networks.

2 Experiment design

To systematically develop a phenomenology of infinite and finite neural networks, we first establish base cases for each architecture where infinite-width kernel methods, linearized weight-space networks, and nonlinear gradient descent based training can be directly compared. In the finite-width settings, the base case uses mini-batch gradient descent at a constant small learning rate [24] with MSE loss (implementation details in §H). In the kernel-learning setting we compute the NNGP and NTK for the entire dataset and do exact inference as described in [60, page 16]. Once this one-to-one comparison has been established, we augment the base setting with a wide range of interventions. We discuss each of these interventions in detail below. Some interventions will approximately preserve the correspondence (for example, data augmentation), while others explicitly break the correspondence in a way that has been hypothesized in the literature to affect performance (for example, large learning rates [39]). We additionally explore linearizing the base model around its initialization, in which case its training dynamics become exactly described by a constant kernel. This differs from the kernel setting described above due to finite width effects.

We use MSE loss to allow for easier comparison to kernel methods, whose predictions can be evaluated in closed form for MSE. See Table S2 and Figure S3 for a comparison of MSE to softmax-cross-entropy loss. Softmax-cross-entropy provides a consistent small benefit over MSE, and will be interesting to consider in future work.

Architectures we work with are built from either Fully-Connected (`FCN`) or Convolutional (`CNN`) layers. In all cases we use ReLU nonlinearities. Except if otherwise stated, we consider `FCN`s with 3-layers and `CNN`s with 8-layers. For convolutional networks we must collapse the spatial dimensions of image-shaped data before the final readout layer. To do this we either: flatten the image into a one-dimensional vector (`VEC`) or apply global average pooling to the spatial dimensions (`GAP`). Finally, we compare two ways of parameterizing the weights and biases of the network: the standard parameterization (STD), which is used in work on finite-width networks, and the NTK parameterization (NTK) which has been used in most infinite-width studies to date (see [27] for the

standard parameterization at infinite width).

Except where noted, for all kernel experiments we optimize over diagonal kernel regularization independently for each experiment. For finite width networks, except where noted we use a small learning rate corresponding to the base case. See §C.1 for details.

The experiments described in this paper are often very compute intensive. For example, to compute the NTK or NNGP for the entirety of CIFAR-10 for `CNN-GAP` architectures one must explicitly evaluate the entries in a 6×10^7-by-6×10^7 kernel matrix. Typically this takes around 1200 GPU hours with double precision, and so we implement our experiments via massively distributed compute infrastructure based on beam [61]. All experiments use the Neural Tangents library [15], built on top of JAX [62].

To be as systematic as possible while also tractable given this large computational requirement, we evaluated every intervention for every architecture and focused on a single dataset, CIFAR-10 [63]. However, to ensure robustness of our results across dataset, we evaluate several key claims on CIFAR-100 and Fashion-MNIST [64].

3 Observed empirical phenomena

3.1 NNGP/NTK can outperform finite networks

A common assumption in the study of infinite networks is that they underperform the corresponding finite network in the large data regime. We carefully examine this assumption, by comparing kernel methods against the base case of a finite width architecture trained with small learning rate and no regularization (§2), and then individually examining the effects of common training practices which break (large LR, L2 regularization) or improve (ensembling) the infinite width correspondence to kernel methods. The results of these experiments are summarized in Figure 1 and Table S1.

First focusing on base finite networks, we observe that infinite `FCN` and `CNN-VEC` outperform their respective finite networks. On the other hand, infinite `CNN-GAP` networks perform worse than their finite-width counterparts in the base case, consistent with observations in Arora et al. [26]. We emphasize that architecture plays a key role in relative performance. For example, infinite-`FCN`s outperform finite-width networks even when combined with various tricks such as high learning rate, L2, and underfitting. Here the performance becomes similar only after ensembling (§3.3).

One interesting observation is that ZCA regularization preprocessing (§3.10) can provide significant improvements to the `CNN-GAP` kernel, closing the gap to within 1-2%.

3.2 NNGP typically outperforms NTK

Recent evaluations of infinite width networks have put significant emphasis on the NTK, without explicit comparison against the respective NNGP models [26, 29, 40, 41]. Combined with the view of NNGPs as "weakly-trained" [24, 26] (i.e. having only the last layer learned), one might expect NTK to be a more effective model class than NNGP. On the contrary, we usually observe that NNGP inference achieves better performance. This can be seen in Table S1 where SOTA performance among fixed kernels is attained with the NNGP across all architectures. In Figure 2 we show that this trend persists across CIFAR-10, CIFAR-100, and Fashion-MNIST (see Figure S5 for similar trends on UCI regression tasks). In addition to producing stronger models, NNGP kernels require about half the memory and compute as the corresponding NTK, and some of the most performant kernels do not have an associated NTK at all [42]. Together these results suggest that when approaching a new problem where the goal is to maximize performance, practitioners should start with the NNGP.

We emphasize that both tuning of the diagonal regularizer (Figure 5) and sufficient numerical precision (§3.7, Figure S1) were crucial to achieving an accurate comparison of these kernels.

3.3 Centering and ensembling finite networks both lead to kernel-like performance

For overparameterized neural networks, some randomness from the initial parameters persists throughout training and the resulting learned functions are themselves random. This excess variance in the network's predictions generically increases the total test error through the variance term of the bias-variance decomposition. For infinite-width kernel systems this variance is eliminated by using

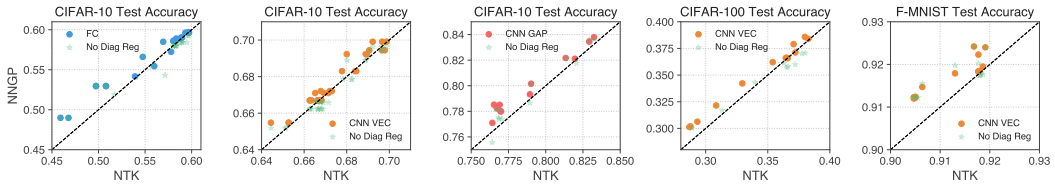

Figure 2: **NNGP often outperforms NTK in image classification tasks when diagonal regularization is carefully tuned.** The performance of the NNGP and NT kernels are plotted against each other for a variety of data pre-processing configurations (§3.10), while regularization (Figure 5) is independently tuned for each.

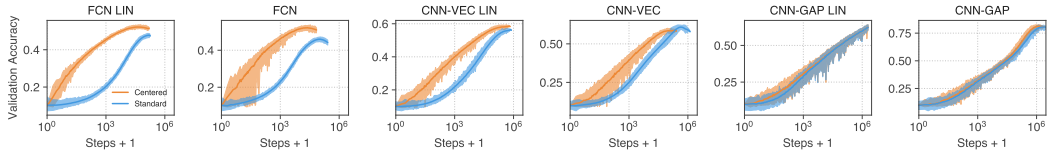

Figure 3: **Centering can accelerate training and improve performance**. Validation accuracy throughout training for several finite width architectures. See Figure S6 for training accuracy.

the mean predictor. For finite-width models, the variance can be large, and test performance can be significantly improved by *ensembling* a collection of models. In Figure 4, we examine the effect of ensembling. For `FCN` networks, ensembling closes the gap with kernel methods, suggesting that `FCN` NNs underperform `FCN` kernels primarily due to variance. For `CNN` models, ensembling also improves test performance, and ensembled `CNN-GAP` models significantly outperform the best kernel methods.

Prediction variance can also be reduced by *centering* the model, i.e. subtracting the model's initial predictions: $f_{\text{centered}}(t) = f(\theta(t)) - f(\theta(0))$. A similar variance reduction technique has been studied in [25, 65–67]. In Figure 3, we observe that centering significantly speeds up training and improves generalization for `FCN` and `CNN-VEC` models, but has little-to-no effect on `CNN-GAP` architectures. We observe that the scale posterior variance of `CNN-GAP`, in the infinite-width kernel, is small relative to the prior variance given more data, consistent with centering and ensembles having small effect.

3.4 Large LRs and L2 regularization drive differences between finite networks and kernels

In practice, L2 regularization (a.k.a. weight decay) or larger learning rates can break the correspondence between kernel methods and finite width neural network training even at large widths.

Lee et al. [24] derives a critical learning rate η_{critical} such that wide network training dynamics are equivalent to linearized training for $\eta < \eta_{\text{critical}}$. Lewkowycz et al. [39] argues that even at large width a learning rate $\eta \in (\eta_{\text{critical}}, c \cdot \eta_{\text{critical}})$ for a constant $c > 1$ forces the network to move away from its initial high curvature minimum and converge to a lower curvature minimum, while Li et al. [68] argues that large initial learning rates enable networks to learn 'hard-to-generalize' patterns.

In Figure 1 (and Table S1), we observe that the effectiveness of a large learning rate (LR) is highly sensitive to both architecture and paramerization: LR improves performance of `FCN` and `CNN-GAP` by about 1% for STD parameterization and about 2% for NTK parameterization. In stark contrast, it has little effect on `CNN-VEC` with NTK parameterization and surprisingly, a huge performance boost on `CNN-VEC` with STD parameterization ($+5\%$).

L2 regularization (Equation S1) regularizes the squared distance between the parameters and the

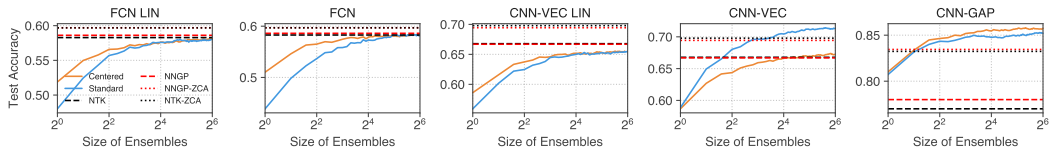

Figure 4: **Ensembling base networks enables them to match the performance of kernel methods, and exceed kernel performance for nonlinear `CNN`s.** See Figure S7 for test MSE.

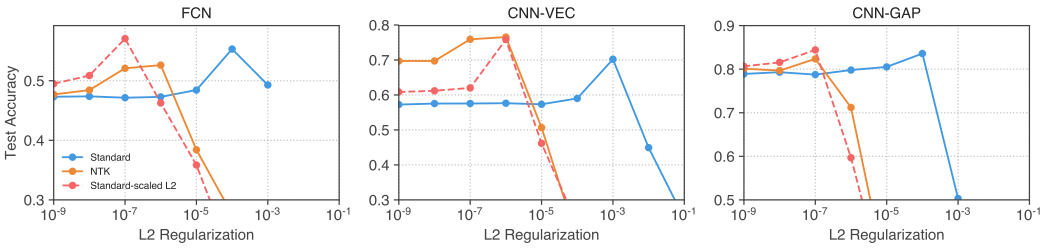

Figure 5: **Layerwise scaling motivated by NTK makes L2 regularization more helpful in standard parameterization networks.** See §3.5 for introduction of the improved regularizer, Figure S9 for further analysis on L2 regularization to initial weights, and Figure S8 for effects on varying widths.

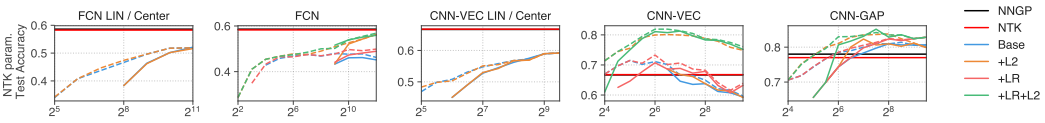

Figure 6: **Finite width networks generally perform better with increasing width, but `CNN-VEC` shows surprising non-monotonic behavior. L2**: non-zero weight decay allowed during training **LR**: large learning rate allowed. Dashed lines are allowing underfitting (**U**). See Figure S10 for plots for the standard parameterization, and §3.11 for discussion of `CNN-VEC` results.

origin and encourages the network to converge to minima with smaller Euclidean norms. Such minima are different from those obtained by NT kernel-ridge regression (i.e. adding a diagonal regularization term to the NT kernel) [32], which essentially penalizes the deviation of the network's parameters from initialization [69]. See Figure S8 for a comparison.

L2 regularization consistently improves (+1-2%) performance for all architectures and parameterizations. Even with a well-tuned L2 regularization, finite width `CNN-VEC` and `FCN` still underperform NNGP/NTK. Combining L2 with early stopping produces a dramatic additional $10\% - 15\%$ performance boost for finite width `CNN-VEC`, outperforming NNGP/NTK. Finally, we note that L2+LR together provide a superlinear performance gain for all cases except `FCN` and `CNN-GAP` with NTK-parameterization. Understanding the nonlinear interactions between L2, LR, and early stopping on finite width networks is an important research question (e.g. see [39, 70] for LR/L2 effect on the training dynamics).

3.5 Improving L2 regularization for networks using the standard parameterization

We find that L2 regularization provides dramatically more benefit (by up to 6%) to finite width networks with the NTK parameterization than to those that use the standard parameterization (see Table S1). There is a bijective mapping between weights in networks with the two parameterizations, which preserves the function computed by both networks: $W^l_{\text{STD}} = W^l_{\text{NTK}}/\sqrt{n^l}$, where W^l is the lth layer weight matrix, and n^l is the width of the preceding activation vector. Motivated by the improved performance of the L2 regularizer in the NTK parameterization, we use this mapping to construct a regularizer for standard parameterization networks that produces the same penalty as vanilla L2 regularization would produce on the equivalent NTK-parameterized network. This modified regularizer is $R^{\text{STD}}_{\text{Layerwise}} = \frac{\lambda}{2}\sum_l n^l \|W^l_{\text{STD}}\|^2$. This can be thought of as a layer-wise regularization constant $\lambda^l = \lambda n^l$. The improved performance of this regularizer is illustrated in Figure 5.

3.6 Performance can be non-monotonic in width beyond double descent

Deep learning practitioners have repeatedly found that increasing the number of parameters in their models leads to improved performance [9, 71–76]. While this behavior is consistent with a Bayesian perspective on generalization [77–79], it seems at odds with classic generalization theory which primarily considers worst-case overfitting [80–86]. This has led to a great deal of work on the interplay of overparameterization and generalization [87–96]. Of particular interest has been the phenomenon of double descent, in which performance increases overall with parameter account, but

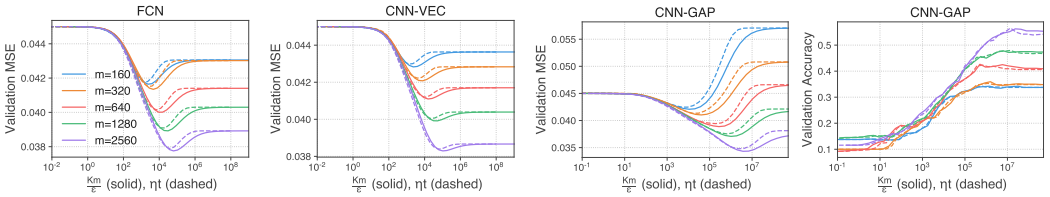

Figure 7: **Diagonal kernel regularization acts similarly to early stopping.** Solid lines corresponds to NTK inference with varying diagonal regularization ε. Dashed lines correspond to predictions after gradient descent evolution to time $\tau = \eta t$ (with $\eta = m/\mathrm{tr}(\mathcal{K})$). Line color indicates varying training set size m. Performing early stopping at time t corresponds closely to regularizing with coefficient $\varepsilon = Km/\eta t$, where $K = 10$ denotes number of output classes.

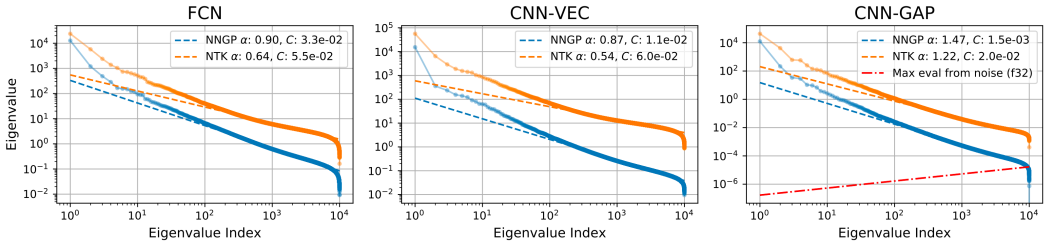

Figure 8: **Tail eigenvalues of infinite network kernels show power-law decay.** The red dashed line shows the predicted scale of noise in the eigenvalues due to floating point precision, for kernel matrices of increasing width. Eigenvalues for CNN-GAP architectures decay fast, and may be overwhelmed by `float32` quantization noise for dataset sizes of $O(10^4)$. For `float64`, quantization noise is not predicted to become significant until a dataset size of $O(10^{10})$ (Figure S1).

drops dramatically when the neural network is roughly critically parameterized [97–99].

Empirically, we find that in most cases (`FCN` and `CNN-GAP` in both parameterizations, `CNN-VEC` with standard parameterization) increasing width leads to monotonic improvements in performance. However, we also find a more complex dependence on width in specific relatively simple settings. For example, in Figure 6 for `CNN-VEC` with NTK parameterization the performance depends non-monotonically on the width, and the optimal width has an intermediate value.[3] This nonmonotonicity is distinct from double-descent-like behavior, as all widths correspond to overparameterized models.

3.7 Diagonal regularization of kernels behaves like early stopping

When performing kernel inference, it is common to add a diagonal regularizer to the training kernel matrix, $\mathcal{K}_{\text{reg}} = \mathcal{K} + \varepsilon \frac{\mathrm{tr}(\mathcal{K})}{m} I$. For linear regression, Ali et al. [101] proved that the inverse of a kernel regularizer is related to early stopping time under gradient flow. With kernels, gradient flow dynamics correspond directly to training of a wide neural network [18, 24].

We experimentally explore the relationship between early stopping, kernel regularization, and generalization in Figure 7. We observe a close relationship between regularization and early stopping, and find that in most cases the best validation performance occurs with early stopping and non-zero ε. While Ali et al. [101] do not consider a $\frac{\mathrm{tr}(\mathcal{K})}{m}$ scaling on the kernel regularizer, we found it useful since experiments become invariant under scale of $\mathcal{K}$.

3.8 Floating point precision determines critical dataset size for failure of kernel methods

We observe empirically that kernels become sensitive to `float32` vs. `float64` numerical precision at a critical dataset size. For instance, GAP models suffer `float32` numerical precision errors at a dataset size of $\sim 10^4$. This phenomena can be understood with a simple random noise model (see §D for details). The key insight is that kernels with fast eigenvalue decay suffer from floating point noise. Empirically, the tail eigenvalue of the NNGP/NTK follows a power law (see Figure 8) and measuring

[3] Similar behavior was observed in [100].

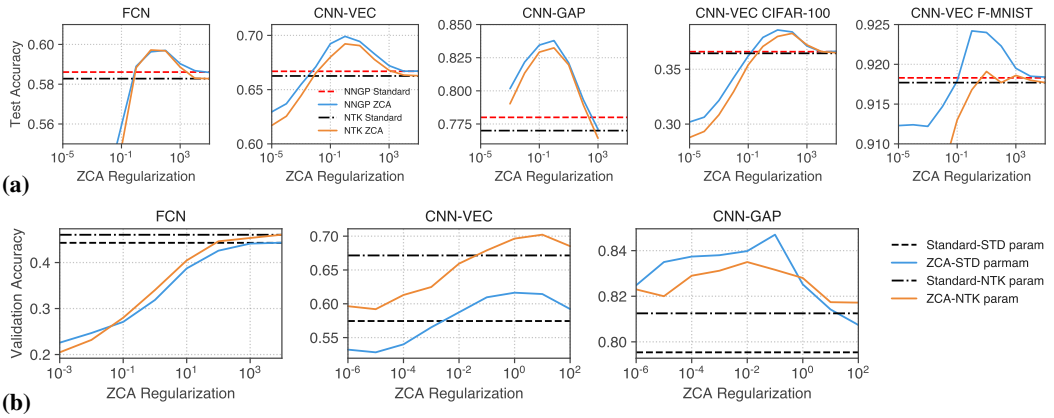

Figure 9: **Regularized ZCA whitening improves image classification performance for both finite and infinite width networks.** All plots show performance as a function of ZCA regularizaiton strength. (**a**) ZCA whitening of inputs to kernel methods on CIFAR-10, Fashion-MNIST, and CIFAR-100. (**b**) ZCA whitening of inputs to finite width networks (training curves in Figure S11).

their decay trend provides good indication of critical dataset size

$$m^* \gtrsim \left(C/(\sqrt{2}\sigma_n)\right)^{\frac{2}{2\alpha-1}} \quad \text{if } \alpha > \tfrac{1}{2} \qquad (\infty \quad \text{otherwise}) \ , \tag{1}$$

where σ_n is the typical noise scale, e.g. `float32` epsilon, and the kernel eigenvalue decay is modeled as $\lambda_i \sim C\, i^{-\alpha}$ as i increases. Beyond this critical dataset size, the smallest eigenvalues in the kernel become dominated by floating point noise.

3.9 Linearized `CNN-GAP` models perform poorly due to poor conditioning

We observe that the linearized `CNN-GAP` converges *extremely* slowly on the training set (Figure S6), leading to poor validation performance (Figure 3). Even after training for more than 10M steps with varying L2 regularization strengths and LRs, the best training accuracy was below 90%, and test accuracy ∼70% – worse than both the corresponding infinite and nonlinear finite width networks.

This is caused by poor conditioning of pooling networks. Xiao et al. [33] (Table 1) show that the conditioning at initialization of a `CNN-GAP` network is worse than that of `FCN` or `CNN-VEC` networks by a factor of the number of pixels (1024 for CIFAR-10). This poor conditioning of the kernel eigenspectrum can be seen in Figure 8. For linearized networks, in addition to slowing training by a factor of 1024, this leads to numerical instability when using `float32`.

3.10 Regularized ZCA whitening improves accuracy

ZCA whitening [102] (see Figure S2 for an illustration) is a data preprocessing technique that was once common [103, 104], but has fallen out of favor. However it was recently shown to dramatically improve accuracy in some kernel methods by Shankar et al. [42], in combination with a small regularization parameter in the denominator (see §F). We investigate the utility of ZCA whitening as a preprocessing step for both finite and infinite width neural networks. We observe that while pure ZCA whitening is detrimental for both kernels and finite networks (consistent with predictions in [105]), with tuning of the regularization parameter it provides performance benefits for both kernel methods and finite network training (Figure 9).

3.11 Equivariance is only beneficial for narrow networks far from the kernel regime

Due to weight sharing between spatial locations, outputs of a convolutional layer are translation-*equivariant* (up to edge effects), i.e. if an input image is translated, the activations are translated in the same spatial direction. However, the vast majority of contemporary `CNN`s utilize weight sharing in conjunction with pooling layers, making the network outputs approximately translation-*invariant* (`CNN-GAP`). The impact of equivariance alone (`CNN-VEC`) on generalization is not well understood – it is a property of internal representations only, and does not translate into meaningful statements about the classifier outputs. Moreover, in the infinite-width limit it is guaranteed to have no impact

Figure 10: **Equivariance is only leveraged in a `CNN` model outside of the kernel regime.** If a `CNN` model is able to utilize equivariance effectively, we expect it to be more robust to crops and translations than an `FCN`. Surprisingly, performance of a wide `CNN-VEC` degrades with the magnitude of the input perturbation as fast as that of an `FCN`, indicating that equivariance is not exploited. In contrast, performance of a narrow model with weight decay (`CNN-VEC+L2+narrow`) falls off much slower. Translation-invariant `CNN-GAP` remains, as expected, the most robust. Details in §3.11, §C.1.

on the outputs [9, 13]. In the finite regime it has been reported both to provide substantial benefits by Novak et al. [9], Lecun [106] and no significant benefits by Bartunov et al. [107].

We conjecture that equivariance can only be leveraged far from the kernel regime. Indeed, as observed in Figure 1 and discussed in §3.4, multiple kernel correspondence-breaking tricks are required for a meaningful boost in performance over NNGP or NTK (which are mathematically guaranteed to not benefit from equivariance), and the boost is largest at a moderate width (Figure 6). Otherwise, even large ensembles of equivariant models (see `CNN-VEC LIN` in Figure 4) perform comparably to their infinite width, equivariance-agnostic counterparts. Accordingly, prior work that managed to extract benefits from equivariant models [9, 106] tuned networks far outside the kernel regime (extremely small size and `+LR+L2+U` respectively). We further confirm this phenomenon in a controlled setting in Figure 10.

3.12 Ensembling kernel predictors enables practical data augmentation with NNGP/NTK

Finite width neural network often are trained with data augmentation (DA) to improve performance. We observe that the `FCN` and `CNN-VEC` architectures (both finite and infinite networks) benefit from DA, and that DA can cause `CNN-VEC` to become competitive with `CNN-GAP` (Table S1). While `CNN-VEC` possess translation equivariance but not invariance (§3.11), we believe it can effectively leverage equivariance to learn invariance from data.

For kernels, expanding a dataset with augmentation is computationally challenging, since kernel computation is quadratic in dataset size, and inference is cubic. Li et al. [40], Shankar et al. [42] incorporated flip augmentation by doubling the training set size. Extending this strategy to more augmentations such as crop or mixup [108], or to broader augmentations strategies like AutoAugment [109] and RandAugment [110], becomes rapidly infeasible.

Here we introduce a straightforward method for ensembling kernel predictors to enable more extensive data augmentation. More sophisticated approximation approaches such as the Nyström method [111] might yield even better performance. The strategy involves constructing a set of augmented batches, performing kernel inference for each of them, and then performing ensembling of the resulting predictions. This is equivalent to replacing the kernel with a block diagonal approximation, where each block corresponds to one of the batches, and the union of all augmented batches is the full augmented dataset. See §E for more details. This method achieves SOTA for a kernel method corresponding to the infinite width limit of each architecture class we studied (Figure 11 and Table 1).

4 Discussion

We performed an in-depth investigation of the phenomenology of finite and infinite width neural networks through a series of controlled interventions. We quantified phenomena having to do with generalization, architecture dependendence, deviations between infinite and finite networks, numerical stability, data augmentation, data preprocessing, ensembling, network topology, and failure modes of linearization. We further developed best practices that improve performance for both finite and infinite networks. We believe our experiments provide firm empirical ground for future studies.

[4]The normalized Gaussian Myrtle kernel used in Shankar et al. [42] does not have a corresponding finite-width neural network, and was additionally tuned on the test set for the case of CIFAR-10.

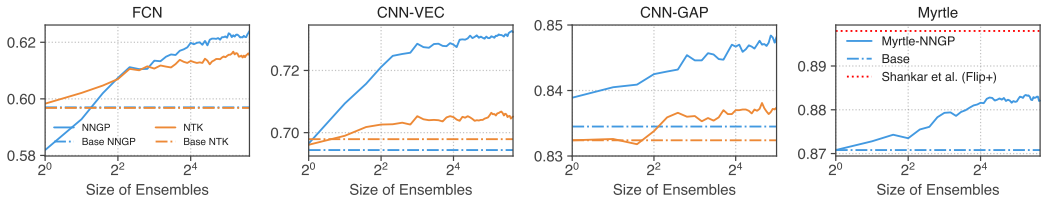

Figure 11: **Ensembling kernel predictors makes predictions from large augmented datasets computationally tractable.** We used standard crop by 4 and flip data augmentation (DA) common for training neural networks for CIFAR-10. We observed that DA ensembling improves accuracy and is much more effective for NNGP compared to NTK. In the last panel, we applied data augmentation by ensemble to the Myrtle architecture studied in Shankar et al. [42]. We observe improvements over our base setting, but do not reach the reported best performance. We believe techniques such as leave-one-out tilt and ZCA augmentation also used in [42] contribute to this difference.

Table 1: **CIFAR-10 test accuracy for kernels of the corresponding architecture type**

Architecture	Method	NTK	NNGP
FC	Novak et al. [9]	-	59.9
	ZCA Reg (this work)	59.7	59.7
	DA Ensemble (this work)	**61.5**	**62.4**
CNN-VEC	Novak et al. [9]	**-**	67.1
	Li et al. [40]	66.6	66.8
	ZCA Reg (this work)	69.8	69.4
	Flip Augmentation, Li et al. [40]	69.9	70.5
	DA Ensemble (this work)	**70.5**	**73.2**
CNN-GAP	Arora et al. [26], Li et al. [40]	77.6	78.5
	ZCA Reg (this work)	83.2	83.5
	Flip Augmentation, Li et al. [40]	79.7	80.0
	DA Ensemble (this work)	**83.7 (32 ens)**	**84.8 (32 ens)**
Myrtle [4]	Myrtle ZCA and Flip Augmentation, Shankar et al. [42]	-	**89.8**

Broader Impact

Developing theoretical understanding of neural networks is crucial both for understanding their biases, and predicting when and how they will fail. Understanding biases in models is of critical importance if we hope to prevent them from perpetuating and exaggerating existing racial, gender, and other social biases [112–115]. Understanding model failure has a direct impact on human safety, as neural networks increasingly do things like drive cars and control the electrical grid [116–118].

We believe that wide neural networks are currently the most promising direction for the development of neural network theory. We further believe that the experiments we present in this paper will provide empirical underpinnings that allow better theory to be developed. We thus believe that this paper will in a small way aid the engineering of safer and more just machine learning models.

Acknowledgments and Disclosure of Funding

We thank Yasaman Bahri and Ethan Dyer for discussions and feedback on the project. We are also grateful to Atish Agarwala, Gamaleldin Elsayed for providing valuable feedbacks on the draft.

We acknowledge the Python community [119] for developing the core set of tools that enabled this work, including NumPy [120], SciPy [121], Matplotlib [122], Pandas [123], Jupyter [124], JAX [125], Neural Tangents [15], Apache Beam [61], Tensorflow datasets [126] and Google Colaboratory [127].

References

[1] Radford M. Neal. Priors for infinite networks (tech. rep. no. crg-tr-94-1). *University of Toronto*, 1994.

[2] Christopher KI Williams. Computing with infinite networks. In *Advances in neural information processing systems*, pages 295–301, 1997.

[3] Tamir Hazan and Tommi Jaakkola. Steps toward deep kernel methods from infinite neural networks. *arXiv preprint arXiv:1508.05133*, 2015.

[4] Jaehoon Lee, Yasaman Bahri, Roman Novak, Sam Schoenholz, Jeffrey Pennington, and Jascha Sohl-dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

[5] Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

[6] Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 9 2018.

[7] Anastasia Borovykh. A gaussian process perspective on convolutional neural networks. *arXiv preprint arXiv:1810.10798*, 2018.

[8] Adrià Garriga-Alonso, Laurence Aitchison, and Carl Edward Rasmussen. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2019.

[9] Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019.

[10] Ge Yang and Samuel Schoenholz. Mean field residual networks: On the edge of chaos. In *Advances in Neural Information Processing Systems*. 2017.

[11] Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. A mean field theory of batch normalization. In *International Conference on Learning Representations*, 2019.

[12] Arnu Pretorius, Herman Kamper, and Steve Kroon. On the expected behaviour of noise regularised deep neural networks as gaussian processes. *arXiv preprint arXiv:1910.05563*, 2019.

[13] Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

[14] Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. In *Advances in Neural Information Processing Systems*, 2019.

[15] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020. URL https://github.com/google/neural-tangents.

[16] Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak. Infinite attention: NNGP and NTK for deep attention networks. In *International Conference on Machine Learning*, 2020.

[17] Jilin Hu, Jianbing Shen, Bin Yang, and Ling Shao. Infinitely wide graph convolutional networks: semi-supervised learning via gaussian processes. *arXiv preprint arXiv:2002.12168*, 2020.

[18] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, 2018.

[19] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pages 8157–8166, 2018.

[20] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, 2018.

[21] Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, 2019.

[22] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

[23] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Stochastic gradient descent optimizes over-parameterized deep relu networks. *Machine Learning*, 2019.

[24] Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems*, 2019.

[25] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, pages 2937–2947, 2019.

[26] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pages 8141–8150. Curran Associates, Inc., 2019.

[27] Jascha Sohl-Dickstein, Roman Novak, Samuel S Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization. *arXiv preprint arXiv:2001.07301*, 2020.

[28] Wei Huang, Weitao Du, and Richard Yi Da Xu. On the neural tangent kernel of deep networks with orthogonal initialization. *arXiv preprint arXiv:2004.05867*, 2020.

[29] Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Advances in Neural Information Processing Systems*. 2019.

[30] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, 2018.

[31] Guillermo Valle-Perez, Chico Q. Camargo, and Ard A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rye4g3AqFm.

[32] Colin Wei, Jason D Lee, Qiang Liu, and Tengyu Ma. Regularization matters: Generalization and optimization of neural nets vs their induced kernel. In *Advances in Neural Information Processing Systems*, pages 9709–9721, 2019.

[33] Lechao Xiao, Jeffrey Pennington, and Samuel S Schoenholz. Disentangling trainability and generalization in deep learning. In *International Conference on Machine Learning*, 2020.

[34] Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. In *Advances in Neural Information Processing Systems 32*. 2019.

[35] Oded Ben-David and Zohar Ringel. The role of a layer in deep neural networks: a gaussian process perspective. *arXiv preprint arXiv:1902.02354*, 2019.

[36] Greg Yang and Hadi Salman. A fine-grained spectral perspective on neural networks. *arXiv preprint arXiv:1907.10599*, 2019.

[37] Sebastian W Ober and Laurence Aitchison. Global inducing point variational posteriors for bayesian neural networks and deep gaussian processes. *arXiv preprint arXiv:2005.08140*, 2020.

[38] Wei Hu, Lechao Xiao, and Jeffrey Pennington. Provable benefit of orthogonal initialization in optimizing deep linear networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgqN1SYvr.

[39] Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

[40] Zhiyuan Li, Ruosong Wang, Dingli Yu, Simon S Du, Wei Hu, Ruslan Salakhutdinov, and Sanjeev Arora. Enhanced convolutional neural tangent kernels. *arXiv preprint arXiv:1911.00809*, 2019.

[41] Sanjeev Arora, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkl8sJBYvH.

[42] Vaishaal Shankar, Alex Chengyu Fang, Wenshuo Guo, Sara Fridovich-Keil, Ludwig Schmidt, Jonathan Ragan-Kelley, and Benjamin Recht. Neural kernels without tangents. In *International Conference on Machine Learning*, 2020.

[43] Ziyu Wang, Tongzheng Ren, Jun Zhu, and Bo Zhang. Function space particle optimization for bayesian neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BkgtDsCcKQ.

[44] Zezhou Cheng, Matheus Gadelha, Subhransu Maji, and Daniel Sheldon. A bayesian perspective on the deep image prior. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[45] Eduardo D C Carvalho, Ronald Clark, Andrea Nicastro, and Paul H. J. Kelly. Scalable uncertainty for computer vision with functional variational inference. In *CVPR*, 2020.

[46] Evgenii Tsymbalov, Sergei Makarychev, Alexander Shapeev, and Maxim Panov. Deeper connections between neural networks and gaussian processes speed-up active learning. In *International Joint Conference on Artificial Intelligence, IJCAI-19*, 2019.

[47] Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Advances In Neural Information Processing Systems*, 2009.

[48] Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances In Neural Information Processing Systems*, 2016.

[49] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances In Neural Information Processing Systems*, 2016.

[50] Minmin Chen, Jeffrey Pennington, and Samuel Schoenholz. Dynamical isometry and a mean field theory of RNNs: Gating enables signal propagation in recurrent neural networks. In *International Conference on Machine Learning*, 2018.

[51] Ping Li and Phan-Minh Nguyen. On random deep weight-tied autoencoders: Exact asymptotic analysis, phase transitions, and implications to training. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJx54i05tX.

[52] Amit Daniely. SGD learns the conjugate kernel class of the network. In *Advances in Neural Information Processing Systems*, 2017.

[53] Arnu Pretorius, Elan van Biljon, Steve Kroon, and Herman Kamper. Critical initialisation for deep signal propagation in noisy rectifier neural networks. In *Advances in Neural Information Processing Systems*. 2018.

[54] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the selection of initialization and activation function for deep neural networks. *arXiv preprint arXiv:1805.08266*, 2018.

[55] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal statistics of fisher information in deep neural networks: Mean field approach. *arXiv preprint arXiv:1806.01316*, 2018.

[56] Yaniv Blumenfeld, Dar Gilboa, and Daniel Soudry. A mean field theory of quantized deep networks: The quantization-depth trade-off. *arXiv preprint arXiv:1906.00771*, 2019.

[57] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. Mean-field behaviour of neural tangent kernel for deep neural networks, 2019.

[58] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *International Conference on Learning Representations*, 2017.

[59] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in neural information processing systems*, 2017.

[60] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[61] The Apache Software Foundation. Apache beam. URL https://beam.apache.org/.

[62] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[63] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[64] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[65] Yaoyu Zhang, Zhi-Qin John Xu, Tao Luo, and Zheng Ma. A type of generalization error induced by initialization in deep neural networks. *arXiv preprint arXiv:1905.07777*, 2019.

[66] Wei Hu, Zhiyuan Li, and Dingli Yu. Simple and effective regularization methods for training on noisily labeled data with generalization guarantee. In *International Conference on Learning Representations*, 2020.

[67] Yu Bai and Jason D. Lee. Beyond linearization: On quadratic and higher-order approximation of wide neural networks. In *International Conference on Learning Representations*, 2020.

[68] Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems*, 2019.

[69] Wei Hu, Zhiyuan Li, and Dingli Yu. Understanding generalization of deep neural networks trained with noisy labels. *arXiv preprint arXiv:1905.11368*, 2019.

[70] Aitor Lewkowycz and Guy Gur-Ari. On the training dynamics of deep networks with l_2 regularization. *arXiv preprint arXiv:2006.08643*, 2020.

[71] Steve Lawrence, C Lee Giles, and Ah Chung Tsoi. What size neural network gives optimal generalization? convergence properties of backpropagation. Technical report, 1998.

[72] Peter L Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE transactions on Information Theory*, 1998.

[73] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *Proceeding of the international Conference on Learning Representations workshop track*, abs/1412.6614, 2015.

[74] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[75] Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018.

[76] Daniel S. Park, Jascha Sohl-Dickstein, Quoc V. Le, and Samuel L. Smith. The effect of network width on stochastic gradient descent and generalization: an empirical study. In *International Conference on Machine Learning*, 2019.

[77] David JC MacKay. Probable networks and plausible predictions—a review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems*, 6(3):469–505, 1995.

[78] Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017.

[79] Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

[80] David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and computation*, 1992.

[81] Eric B. Baum and David Haussler. What size net gives valid generalization? In *Advances in Neural Information Processing Systems*. 1989.

[82] Vladimir Vapnik. Statistical learning theory. 1998.

[83] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[84] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

[85] Sayan Mukherjee, Partha Niyogi, Tomaso Poggio, and Ryan Rifkin. Statistical learning: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. Technical report, 2004.

[86] Tomaso Poggio, Ryan Rifkin, Sayan Mukherjee, and Partha Niyogi. General conditions for predictivity in learning theory. *Nature*, 428(6981):419–422, 2004.

[87] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

[88] Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.

[89] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

[90] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2019.

[91] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*. 2018.

[92] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pages 6155–6166, 2019.

[93] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Limitations of lazy training of two-layers neural network. In *Advances in Neural Information Processing Systems*, pages 9108–9118, 2019.

[94] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Linearized two-layers neural networks in high dimension. *arXiv preprint arXiv:1904.12191*, 2019.

[95] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*, 2019.

[96] Alon Brutzkus and Amir Globerson. Why do larger models generalize better? A theoretical perspective via the XOR problem. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[97] M Opper, W Kinzel, J Kleinz, and R Nehl. On the ability of the optimal perceptron to generalise. *Journal of Physics A: Mathematical and General*, 23(11):L581, 1990.

[98] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116 (32):15849–15854, 2019.

[99] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019.

[100] Anders Andreassen and Ethan Dyer. Asymptotics of wide convolutional neural networks. *preprint*, 2020.

[101] Alnur Ali, J Zico Kolter, and Ryan J Tibshirani. A continuous-time view of early stopping for least squares regression. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1370–1378, 2019.

[102] Anthony J Bell and Terrence J Sejnowski. The "independent components" of natural scenes are edge filters. *Vision research*, 37(23):3327–3338, 1997.

[103] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *International Conference on Machine Learning*, 2018.

[104] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016.

[105] Neha Wadia, Daniel Duckworth, Samuel Schoenholz, Ethan Dyer, and Jascha Sohl-Dickstein. Whitening and second order optimization both destroy information about the dataset, and can make generalization impossible. *preprint*, 2020.

[106] Yann Lecun. Generalization and network design strategies. In *Connectionism in perspective*. Elsevier, 1989.

[107] Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Advances in Neural Information Processing Systems*, pages 9368–9378, 2018.

[108] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

[109] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019.

[110] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019.

[111] Christopher KI Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.

[112] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323, 2016.

[113] Solon Barocas and Andrew D Selbst. Big data's disparate impact. *Calif. L. Rev.*, 104:671, 2016.

[114] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[115] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. http://www.fairmlbook.org.

[116] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[117] Cynthia Rudin, David Waltz, Roger N Anderson, Albert Boulanger, Ansaf Salleb-Aouissi, Maggie Chow, Haimonti Dutta, Philip N Gross, Bert Huang, Steve Ierome, et al. Machine learning for the new york city power grid. *IEEE transactions on pattern analysis and machine intelligence*, 2011.

[118] Mete Ozay, Inaki Esnaola, Fatos Tunay Yarman Vural, Sanjeev R Kulkarni, and H Vincent Poor. Machine learning methods for attack detection in the smart grid. *IEEE transactions on neural networks and learning systems*, 2015.

[119] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[120] S. van der Walt, S. C. Colbert, and G. Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering*, 13(2):22–30, 2011.

[121] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[122] J. D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[123] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi: 10.25080/Majora-92bf1922-00a.

[124] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter Notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[125] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[126] TensorFlow Datasets, a collection of ready-to-use datasets. https://www.tensorflow.org/datasets.

[127] Google Research. Google colab. URL https://colab.research.google.com/.

[128] Robert T Clemen. Combining forecasts: A review and annotated bibliography. *International journal of forecasting*, 5(4):559–583, 1989.

[129] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[130] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[131] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[132] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[133] David W Opitz and Jude W Shavlik. Generating accurate and diverse members of a neural-network ensemble. In *Advances in neural information processing systems*, pages 535–541, 1996.

[134] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[135] Lior Rokach. Ensemble-based classifiers. *Artificial intelligence review*, 33(1-2):1–39, 2010.

[136] Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1gFvANKDS.

[137] Christopher J Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 2019.

[138] Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

[139] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

[140] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016.

Supplementary Material

A Glossary

We use the following abbreviations in this work:

- **L2**: L2 reguarization a.k.a. weight decay;
- **LR**: using large learning rate;
- **U**: allowing underfitting;
- **DA**: using data augmentation;
- **C**: centering the network so that the logits are always zero at initialization;
- **Ens**: neural network ensembling logits over multiple initialization;
- **ZCA**: zero-phase component analysis regularization preprocessing;
- **FCN**: fully-connected neural network.;
- **CNN-VEC**: convolutional neural network with a vectorized readout layer;
- **CNN-GAP**: convolutional neural network with a global average pooling readout layer;
- **NNGP**: neural network Gaussian process;
- **NTK**: neural tangent kernel.

B Main table

Table S1: **CIFAR-10 classification accuracy for nonlinear and linearized finite neural networks, as well as for NTK and NNGP kernel methods**. Starting from `Base` network of given architecture class described in §2, performance change of **centering** (`+C`), **large learning rate** (`+LR`), allowing **underfitting** by early stopping (`+U`), input preprocessing with **ZCA regularization** (`+ZCA`), multiple initialization **ensembling** (`+Ens`), and some combinations are shown, for **Standard** and **NTK** parameterization. See also Figure 1.

	Param	Base	+C	+LR	+L2	+L2 +U	+L2 +LR	+L2 +LR +U	+ZCA	Best w/o DA	+Ens	+Ens +C	+DA +U	+DA +L2 +LR +U
FCN	STD	47.82	53.22	49.07	49.82	49.82	55.32	55.32	44.29	55.90	58.11	58.25	65.29	67.43
	NTK	46.16	51.74	48.14	54.27	54.27	55.11	55.44	44.86	55.44	58.14	58.31	61.87	69.35
CNN-VEC	STD	56.68	60.82	62.16	57.15	67.07	62.16	68.99	57.39	68.99	67.30	65.65	76.73	83.01
	NTK	60.73	58.09	60.73	61.30	75.85	76.93	77.47	61.35	77.47	71.32	67.23	83.92	85.63
CNN-GAP	STD	80.26	81.25	80.93	81.67	81.10	83.69	83.01	84.90	84.22	84.15	84.62	84.36	86.45
	NTK	80.61	81.73	82.44	81.17	81.17	82.44	82.43	83.75	83.92	85.22	85.75	84.07	86.68

	Param	Lin Base	+C	+L2	+L2 +U	+Ens	+Ens +C	NTK	+ZCA	+DA +ZCA	NNGP	+ZCA	+DA +ZCA
FCN	STD	43.09	51.48	44.16	50.77	57.85	57.99	58.05	59.65	-	58.61	59.70	62.40
	NTK	48.61	52.12	51.77	51.77	58.04	58.16	58.28	59.68	61.54			
CNN-VEC	STD	52.43	60.61	58.41	58.41	64.58	64.67	66.64	69.65	-	66.69	69.44	73.23
	NTK	55.88	58.94	58.52	58.50	65.45	65.54	66.78	69.79	70.52			
CNN-GAP	STD	>70.00* (Train accuracy 86.22 after 14M steps)						76.97	83.24	-	78.0	83.45	84.82
	NTK	>68.59* (Train accuracy 79.90 after 14M steps)						77.00	83.24	83.74			

C Experimental details

For all experiments, we use Neural Tangents (NT) library [15] built on top of JAX [125]. First we describe experimental settings that is mostly common and then describe specific details and hyperparameters for each experiments.

Finite width neural networks We train finite width networks with Mean Squared Error (MSE) loss

$$\mathcal{L} = \frac{1}{2|\mathcal{D}|K} \sum_{(x_i, y_i) \in \mathcal{D}} \|f(x_i) - y_i\|^2 ,$$

where K is the number of classes and $\|\cdot\|$ is the L^2 norm in $\mathbb{R}^K$. For the experiments with `+L2`, we add L2 regularization to the loss

$$R_{\text{L2}} = \frac{\lambda}{2} \sum_l \|W^l\|^2 \,, \tag{S1}$$

and tune λ using grid-search optimizing for the validation accuracy.

We optimize the loss using mini-batch SGD with constant learning rate. We use batch-size of 100 for `FCN` and 40 for both `CNN-VEC` and `CNN-GAP` (see §H for further details on this choice). Learning rate is parameterized with learning rate factor c with respect to the critical learning rate

$$\eta = c\, \eta_{\text{critical}} \,. \tag{S2}$$

In practice, we compute empirical NTK $\hat{\Theta}(x, x') = \sum_j \partial_j f(x) \partial_j f(x')$ on 16 random points in the training set to estimate η_{critical} [24] by maximum eigenvalue of $\hat{\Theta}(x, x)$. This is readily available in NT library [15] using `nt.monte_carlo_kernel_fn` and `nt.predict.max_learning_rate`. Base case considered without large learning rate indicates $c \leq 1$, and large learning rate (`+LR`) runs are allowing $c > 1$. Note that for linearized networks η_{critical} is strict upper-bound for the learning rates and no $c > 1$ is allowed [24, 36, 39].

Training steps are chosen to be large enough, such that learning rate factor $c \leq 1$ can reach above 99% accuracy on $5k$ random subset of training data for 5 logarithmic spaced measurements. For different learning rates, physical time $t = \eta \times$ (# of steps) roughly determines learning dynamics and small learning rate trials need larger number of steps. Achieving termination criteria was possible for all of the trials except for linearized `CNN-GAP` and data augmented training of `FCN`, `CNN-VEC`. In these cases, we report best achieved performance without fitting the training set.

NNGP / NTK For inference, except for data augmentation ensembles for which default zero regularization was chosen, we grid search over diagonal regularization in the range `numpy.logspace(-7, 2, 14)` and 0. Diagonal regularization is parameterized as

$$\mathcal{K}_{\text{reg}} = \mathcal{K} + \varepsilon \frac{\text{tr}(\mathcal{K})}{m} I$$

where $\mathcal{K}$ is either NNGP or NTK for the training set. We work with this parameterization since ε is invariant to scale of $\mathcal{K}$.

Dataset For all our experiments (unless specified) we use train/valid/test split of 45k/5k/10k for CIFAR-10/100 and 50k/10k/10k for Fashion-MNIST. For all our experiments, inputs are standardized with per channel mean and standard deviation. ZCA regularized whitening is applied as described in §F. Output is encoded as mean subtracted one-hot-encoding for the MSE loss, e.g. for a label in class c, $-0.1 \cdot \mathbf{1} + \mathbf{e_c}$. For the softmax-cross-entropy loss in §G, we use standard one-hot-encoded output.

For data augmentation, we use widely-used augmentation for CIFAR-10; horizontal flips with 50% probability and random crops by 4-pixels with zero-padding.

Details of architecture choice: We only consider ReLU activation (with the exception of Myrtle-kernel which use scaled Gaussian activation [42]) and choose critical initialization weight variance of $\sigma_w^2 = 2$ with small bias variance $\sigma_b^2 = 0.01$. For convolution layers, we exclusively consider 3×3 filters with stride 1 and `SAME` (zero) padding so that image size does not change under convolution operation.

C.1 Hyperparameter configurations for all experiments

We used grid-search for tuning hyperparameters and use accuracy on validation set for deciding on hyperparameter configuration or measurement steps (for underfitting / early stopping). All reported numbers unless specified is test set performance.

Figure 1, Table S1: We grid-search over L2 regularization strength $\lambda \in \{0\} \cup \{10^{-k} | k \text{ from -9 to -3}\}$ and learning rate factor $c \in \{2^k | k \text{ from -2 to 5}\}$. For linearized networks same search space is used except that $c > 1$ configuration is infeasible and training diverges. For non-linear, centered runs $c \in \{2^k | k \text{ from 0 to 4}\}$ is used. Network ensembles uses base configuration with $\lambda = 0$, $c = 1$ with 64 different initialization seed. Kernel ensemble is over 50 predictors for `FCN` and `CNN-VEC` and 32 predictors for `CNN-GAP`. Finite networks trained with data-augmentation has different learning rate factor range of $c \in \{1, 4, 8\}$.

Figure 2: Each datapoint corresponds to either standard preprocessed or ZCA regularization preprocessed (as described in §3.10) with regularization strength was varied in $\{10^{-k}|k \in [-6, -5, ..., 4, 5]\}$ for `FCN` and `CNN-VEC`, $\{10^{-k}|k \in [-3, -2, ..., 2, 3]\}$ for `CNN-GAP`.

Figure 3, Figure 4, Figure S6, Figure S7: Learning rate factors are $c = 1$ for non-linear networks and $c = 0.5$ for linearized networks. While we show NTK parameterized runs, we also observe similar trends for STD parameterized networks. Shaded regions show range of minimum and maximum performance across 64 different seeds. Solid line indicates the mean performance.

Figure 5 While `FCN` is the base configuration, `CNN-VEC` is a narrow network with 64 channels per layer since moderate width benefits from L2 more for the NTK parameterization Figure S10. For `CNN-GAP` 128 channel networks is used. All networks with different L2 strategy are trained with `+LR` ($c > 1$).

Figure 6, Figure S8, Figure S10: $\lambda \in \{0, 10^{-9}, 10^{-7}, 10^{-5}, 10^{-3}\}$ and $c \in \{2^k|k \text{ from } -2 \text{ to } 5\}$.

Figure 7: We use 640 subset of validation set for evaluation. `CNN-GAP` is a variation of the base model with 3 convolution layers with $\sigma_b^2 = 0.1$ while `FCN` and `CNN-VEC` is the base model. Training evolution is computed using analytic time-evolution described in Lee et al. [24] and implemented in NT library via `nt.predict.gradient_descent_mse` with 0 diagonal regularization.

Figure 9: Kernel experiments details are same as in Figure 2. Finite networks are base configuration with $c = 1$ and $\lambda = 0$.

Figure 10: Evaluated networks uses NTK parameterization with $c = 1$. `CNN-VEC+L2+narrow` uses 128 channels instead of 512 of the base `CNN-VEC` and `CNN-GAP` networks, and trained with L2 regularization strength $\lambda = 10^{-7}$. *Crop* transformation uses zero-padding while *Translate* transformation uses circular boundary condition after shifting images. Each transformation is applied to the test set inputs where shift direction is chosen randomly. Each points correspond to average accuracy over 20 random seeds. `FCN` had 2048 hidden units.

Figure 11, Table 1: For all data augmentation ensembles, first instance is taken to be from non-augmented training set. Further details on kernel ensemble is described in §E. For all kernels, inputs are preprocessed with optimal ZCA regularization observed in Figure 9 (10 for `FCN`, 1 for `CNN-VEC`, `CNN-GAP` and `Myrtle`.). We ensemble over 50 different augmented draws for `FCN` and `CNN-VEC`, whereas for `CNN-GAP`, we ensemble over 32 draws of augmented training set.

Figure S3, Table S2: Details for MSE trials are same as Figure 1 and Table S1. Trials with softmax-cross-entropy loss was tuned with same hyperparameter range as MSE except that learning rate factor range was $c \in \{1, 4, 8\}$.

Figure S4: We present result with NTK parameterized networks with $\lambda = 0$. `FCN` network is width 1024 with $\eta = 10.0$ for MSE loss and $\eta = 2.0$ for softmax-cross-entropy loss. `CNN-GAP` uses 256 channels with $\eta = 5.0$ for MSE loss and $\eta = 0.2$ for softmax-cross-entropy loss. Random seed was fixed to be the same across all runs for comparison.

Figure S9: NTK pamareterization with $c = 4$ was used for both L2 to zero or initialization. Random seed was fixed to be the same across all runs for comparison.

D Noise model

In this section, we provide details on noise model discussed in §3.8. Consider a random $m \times m$ Hermitian matrix N with entries order of σ_n which is considered as noise perturbation to the kernel matrix

$$\tilde{K} = K + N\,. \tag{S3}$$

Eigenvalues of this random matrix N follow Wigner's semi-circle law, and the smallest eigenvalue is given by $\lambda_{\min}(N) \approx -\sqrt{2m}\sigma_n$. When the smallest eigenvalue of K is smaller (in order) than $|\lambda_{\min}(N)|$, one needs to add diagonal regularizer larger than the order of $|\lambda_{\min}(N)|$ to ensure positive definiteness. For estimates, let us use machine precision[5] $\epsilon_{32} \approx 10^{-7}$ and $\epsilon_{64} \approx 2 \times 10^{-16}$ which we use as proxy values for σ_n. Note that noise scale is relative to elements in K which is assume to be $O(1)$. Naively scaling K by multiplicative constant will also scale σ_n.

[5] `np.finfo(np.float32).eps`, `np.finfo(np.float64).eps`

Empirically one can model tail i^{th} eigenvalues of infinite width kernel matrix of size $m \times m$ as

$$\lambda_i \approx C\frac{m}{i^\alpha}\,. \tag{S4}$$

Note that we are considering $O(1)$ entries for K and typical eigenvalues scale linearly with dataset size m. For a given dataset size, the power law observed is α and C is dataset-size independent constant. Thus the smallest eigenvalue is order $\lambda_{\text{min}}(K) \sim Cm^{1-\alpha}$.

In the noise model, we can apply Weyl's inequality which says

$$\lambda_{\text{min}}(K) - \sqrt{2m}\sigma_n \leq \lambda_{\text{min}}(\tilde{K}) \leq \lambda_{\text{min}}(K) + \sqrt{2m}\sigma_n\,. \tag{S5}$$

Consider the worst-case where negative eigenvalue noise affecting the kernel's smallest eigenvalue. In that case perturbed matrices minimum eigenvalue could become negative, breaking positive semi-definiteness(PSD) of the kernel.

This model allows to predict critical dataset size (m^*) over which PSD can be broken under specified noise scale and kernel eigenvalue decay. With condition that perturbed smallest eigenvalue becomes negative

$$Cm^{1-\alpha} \lesssim \sqrt{2m}\sigma_n\,, \tag{S6}$$

we obtain

$$m^* \gtrsim \begin{cases} \left(\frac{C}{\sqrt{2}\sigma_n}\right)^{\frac{2}{2\alpha-1}} & \text{if } \alpha > \frac{1}{2} \\ \infty & \text{else} \end{cases} \tag{S7}$$

When PSD is broken, one way to preserve PSD is to add diagonal regularizer (§3.7). For CIFAR-10 with $m = 50k$, typical negative eigenvalue from `float32` noise is around 4×10^{-5} and 7×10^{-14} with `float64` noise scale, considering $\sqrt{2m}\sigma_n$. Note that Arora et al. [26] regularized kernel with regularization strength 5×10^{-5} which is on par with typical negative eigenvalue introduced due to `float32` noise. Of course, this only applies if kernel eigenvalue decay is sufficiently fast that full dataset size is above m^*.

We observe that `FCN` and `CNN-VEC` kernels with small α would not suffer from increasing dataset-size under `float32` precision. On the other-hand, worse conditioning of `CNN-GAP` not only affects the training time (§3.9) but also required precision. One could add sufficiently large diagonal regularization to mitigate effect from the noise at the expense of losing information and generalization strength included in eigen-directions with small eigenvalues.

E Data augmentation via kernel ensembling

We start considering general ensemble averaging of predictors. Consider a sequence of training sets $\{\mathcal{D}_i\}$ each consisting of m input-output pairs $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ from a data-generating distribution. For a learning algorithm, which we use NNGP/NTK inference for this study, will give prediction $\mu(x^*, \mathcal{D}_i)$ of unseen test point x^*. It is possible to obtain better predictor by averaging output of different predictors

$$\hat{\mu}(x^*) = \frac{1}{E}\sum_i^E \mu(x^*, \mathcal{D}_i)\,, \tag{S8}$$

where E denotes the cardinality of $\{\mathcal{D}_i\}$. This ensemble averaging is simple type of committee machine which has long history [128, 129]. While more sophisticated ensembling method exists (e.g. [130–135]), we strive for simplicity and considered naive averaging. One alternative we considered is generalizing average by

$$\hat{\mu}_w(x^*) = \frac{1}{E}\sum_i^E w_i\,\mu(x^*, \mathcal{D}_i)\,, \tag{S9}$$

were w_i in general is set of weights satisfying $\sum_i w_i = 1$. We can utilize posterior variance σ_i^2 from NNGP or NTK with MSE loss via Inverse-variance weighting (IVW) where weights are given as

$$w_i = \frac{\sigma_i^{-2}}{\sum_j \sigma_j^{-2}}\,. \tag{S10}$$

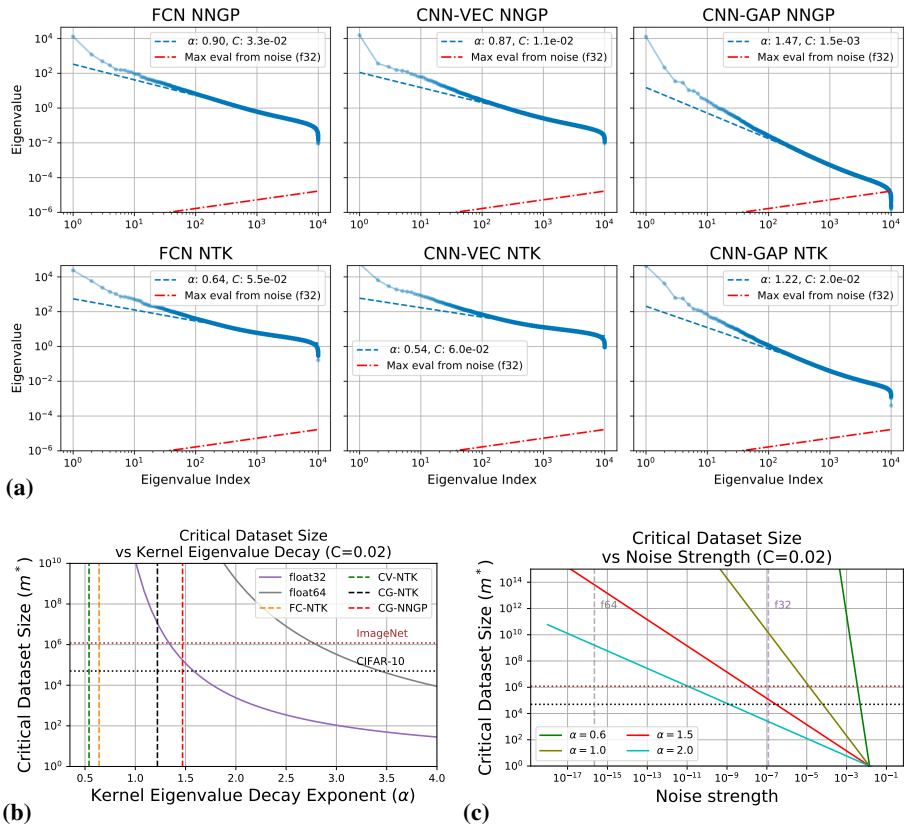

Figure S1: **The CNN-GAP architecture has poor kernel conditioning (a)** Eigenvalue spectrum of infinite network kernels on 10k datapoints. Dashed lines are noise eigenvalue scale from `float32` precision. Eigenvalue for CNN-GAP's NNGP decays fast and negative eigenvalue may occur when dataset size is $O(10^4)$ in `float32` but is well-behaved with higher precision. **(b-c)** Critical dataset size as function of eigenvalue decay exponent α or noise strength σ_n given by Equation 1.

In simple bagging setting [131], we observe small improvements with IVW over naive averaging. This indicates posterior variance for different draw of $\{\mathcal{D}_i\}$ was quite similar.

Application to data augmentation (DA) is simple as we consider process of generating $\{\mathcal{D}_i\}$ from a (stochastic) data augmentation transformation $\mathcal{T}$. We consider action of $\mathcal{T}(x, y) = T(x, y)$ be stochastic (e.g. T is a random crop operator) with probability p augmentation transformation (which itself could be stochastic) and probability $(1 - p)$ of $T = \text{Id}$. Considering $\mathcal{D}_0$ as clean un-augmented training set, we can imagine dataset generating process $\mathcal{D}_i \sim \mathcal{T}(\mathcal{D}_0)$, where we overloaded definition of $\mathcal{T}$ on training-set to be data generating distribution.

For experiments in §3.12, we took T to be standard augmentation strategy of horizontal flip and random crop by 4-pixels with augmentation fraction $p = 0.5$ (see Figure S12 for effect of augmentation fraction on kernel ensemble). In this framework, it is trivial to generalize the DA transformation to be quite general (e.g. learned augmentation strategy studied by Cubuk et al. [109, 110]).

F ZCA whitening

Consider m (flattened) d-dimensional training set inputs X (a $d \times m$ matrix) with data covariance

$$\Sigma_X = \frac{1}{d} X X^T . \tag{S11}$$

The goal of whitening is to find a whitening transformation W, a $d \times d$ matrix, such that the features of transformed input

$$Y = WX \tag{S12}$$

are uncorrelated, e.g. $\Sigma_Y \equiv \frac{1}{d} Y Y^T = I$. Note that Σ_X is constructed only from training set while W is applied to both training set and test set inputs. Whitening transformation can be efficiently

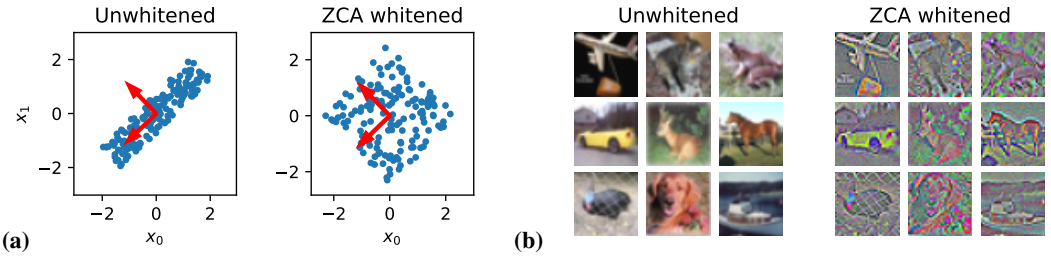

Figure S2: **Illustration of ZCA whitening.** Whitening is a linear transformation of a dataset that removes correlations between feature dimensions, setting all non-zero eigenvalues of the covariance matrix to 1. ZCA whitening is a specific choice of the linear transformation that rescales the data in the directions given by the eigenvectors of the covariance matrix, but without additional rotations or flips. *(a)* A toy 2d dataset before and after ZCA whitening. Red arrows indicate the eigenvectors of the covariance matrix of the unwhitened data. *(b)* ZCA whitening of CIFAR-10 images preserves spatial and chromatic structure, while equalizing the variance across all feature directions. Figure reproduced with permission from Wadia et al. [105]. See also §3.10.

computed by eigen-decomposition[6]

$$\Sigma_X = UDU^T \tag{S13}$$

where D is diagonal matrix with eigenvalues, and U contains eigenvector of Σ_X as its columns.

With this ZCA whitening transformation is obtained by following whitening matrix

$$W_{\text{ZCA}} = U\sqrt{\left(D + \epsilon\frac{tr(D)}{d}I_d\right)^{-1}}U^T\,. \tag{S14}$$

Here, we introduced trivial reparameterization of conventional regularizer such that regularization strength ϵ is input scale invariant. It is easy to check $\epsilon \to 0$ corresponds to whitening with $\Sigma_Y = I$. In §3.10, we study the benefit of taking non-zero regularization strength for both kernels and finite networks. We denote transformation with non-zero regularizer, ZCA regularization preprocessing. ZCA transformation preserves spatial and chromatic structure of original image as illustrated in Figure F. Therefore image inputs are reshaped to have the same shape as original image.

In practice, we standardize both training and test set per (RGB channel) features of the training set before and after the ZCA whitening. This ensures transformed inputs are mean zero and variance of order 1.

G MSE vs Softmax-cross-entropy loss training of neural networks

Our focus was mainly on fininte networks trained with MSE loss for simple comparison with kernel methods that gives closed form solution. Here we present comparison of MSE vs softmax-cross-entropy trained networks. See Table S2 and Figure S3.

H Comment on batch size

Correspondence between NTK and gradient descent training is direct in the full batch gradient descent (GD) setup (see [136] for extensions to mini-batch SGD setting). Therefore base comparison between finite networks and kernels is the full batch setting. While it is possible to train our base models with GD, for full CIFAR-10 large emprical study becomes impractical. In practice, we use mini-batch SGD with batch-size 100 for FCN and 40 for CNNs.

We studied batch size effect of training dynamics in Figure S4 and found that for these batch-size choices does not affecting training dynamics compared to much larger batch size. Shallue et al. [137], McCandlish et al. [138] observed that universally for wide variety of deep learning models there are batch size beyond which one could gain training speed benefit in number of steps. We observe that maximal useful batch-size in workloads we study is quite small.

[6]For PSD matricies, it is numerically more reliable to obtain via SVD.

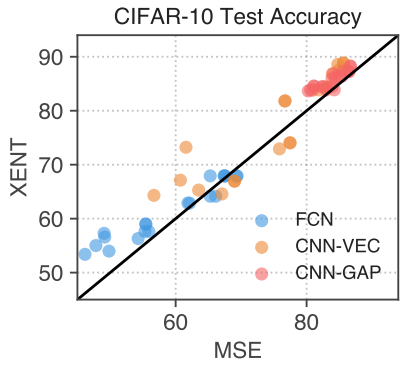

Figure S3: **MSE trained networks are competitive while there is a clear benefit to using Cross-entropy loss**

Table S2: Effects of MSE vs softmax-cross-entropy loss on base networks with various interventions

Architecture	Type	Param	Base	+LR+U	+L2+U	+L2+LR+U	Best
FCN	MSE	STD	47.82	49.07	49.82	55.32	55.90
		NTK	46.16	49.17	54.27	55.44	55.44
	XENT	STD	55.01	57.28	53.98	57.64	57.64
		NTK	53.39	56.59	56.31	58.99	58.99
	MSE+DA	STD	65.29	66.11	65.28	67.43	67.43
		NTK	61.87	62.12	67.58	69.35	69.35
	XENT+DA	STD	64.15	64.15	67.93	67.93	67.93
		NTK	62.88	62.88	67.90	67.90	67.90
CNN-VEC	MSE	STD	56.68	63.51	67.07	68.99	68.99
		NTK	60.73	61.58	75.85	77.47	77.47
	XENT	STD	64.31	65.30	64.57	66.95	66.95
		NTK	67.13	73.23	72.93	74.05	74.05
	MSE+DA	STD	76.73	81.84	76.66	83.01	83.01
		NTK	83.92	84.76	84.87	85.63	85.63
	XENT+DA	STD	81.84	83.86	81.78	84.37	84.37
		NTK	86.83	88.59	87.49	88.83	88.83
CNN-GAP	MSE	STD	80.26	80.93	81.10	83.01	84.22
		NTK	80.61	82.44	81.17	82.43	83.92
	XENT	STD	83.66	83.80	84.59	83.87	83.87
		NTK	83.87	84.40	84.51	84.51	84.51
	MSE+DA	STD	84.36	83.88	84.89	86.45	86.45
		NTK	84.07	85.54	85.39	86.68	86.68
	XENT+DA	STD	86.04	86.01	86.42	87.26	87.26
		NTK	86.87	87.31	86.39	88.26	88.26

I Addtional tables and plots

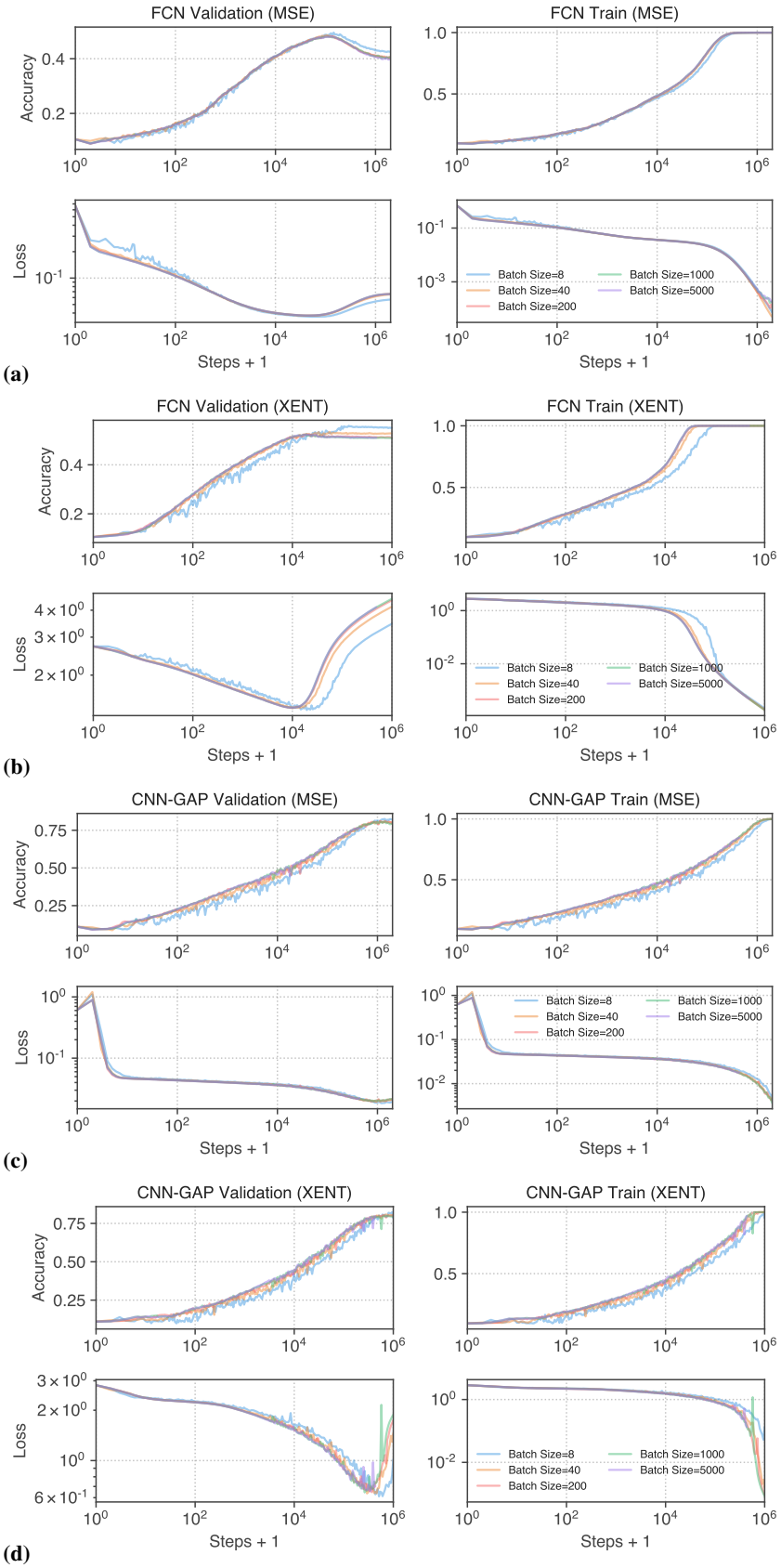

Figure S4: **Batch size does not affect training dynamics for moderately large batch size.**

Table S3: **CIFAR-10 classification mean squared error(MSE) for nonlinear and linearized finite neural networks, as well as for NTK and NNGP kernel methods**. Starting from `Base` network of given architecture class described in §2, performance change of **centering** (+C), **large learning rate** (+LR), allowing **underfitting** by early stopping (+U), input preprocessing with **ZCA regularization** (+ZCA), multiple initialization **ensembling** (+Ens), and some combinations are shown, for **Standard** and **NTK** parameterization. See also Table S1 and Figure 1 for accuracy comparison.

	Param	Base	+C	+LR	+L2	+L2 +U	+L2 +LR	+L2 +LR +U	+ZCA	Best w/o DA	+Ens	+Ens +C	+DA +U	+DA +L2 +LR +U
FCN	STD	0.0443	0.0363	0.0406	0.0411	0.0355	0.0337	0.0329	0.0483	0.0319	0.0301	0.0304	0.0267	0.0242
	NTK	0.0465	0.0371	0.0423	0.0338	0.0336	0.0308	0.0308	0.0484	0.0308	0.0300	0.0302	0.0281	0.0225
CNN-VEC	STD	0.0381	0.0330	0.0340	0.0377	0.0279	0.0340	0.0265	0.0383	0.0265	0.0278	0.0287	0.0228	0.0183
	NTK	0.0355	0.0353	0.0355	0.0355	0.0231	0.0246	0.0227	0.0361	0.0227	0.0254	0.0278	0.0164	0.0143
CNN-GAP	STD	0.0209	0.0201	0.0207	0.0201	0.0201	0.0179	0.0177	0.0190	0.0159	0.0172	0.0165	0.0185	0.0149
	NTK	0.0209	0.0201	0.0195	0.0205	0.0181	0.0175	0.0170	0.0194	0.0161	0.0163	0.0157	0.0186	0.0145

	Param	Lin Base	+C	+L2	+L2 +U	+Ens	+Ens +C	NTK	+ZCA	+DA +ZCA	NNGP	+ZCA	+DA +ZCA
FCN	STD	0.0524	0.0371	0.0508	0.0350	0.0309	0.0305	0.0306	0.0302	-	0.0309	0.0308	0.0297
	NTK	0.0399	0.0366	0.0370	0.0368	0.0305	0.0304	0.0305	0.0302	0.0298			
CNN-VEC	STD	0.0436	0.0322	0.0351	0.0351	0.0293	0.0291	0.0287	0.0277	-	0.0286	0.0281	0.0256
	NTK	0.0362	0.0337	0.0342	0.0339	0.0286	0.0286	0.0283	0.0274	0.0273			
CNN-GAP	STD	< 0.0272* (Train accuracy 86.22 after 14M steps)						0.0233	0.0200	-	0.0231	0.0204	0.0191
	NTK	< 0.0276* (Train accuracy 79.90 after 14M steps)						0.0232	0.0200	0.0195			

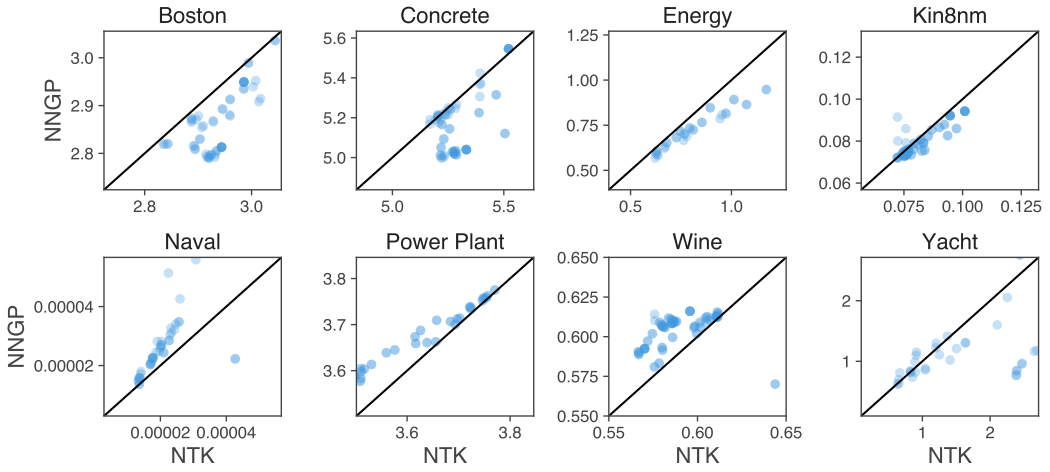

Figure S5: **On UCI dataset NNGP often outperforms NTK on RMSE.** We evaluate predictive performance of FC NNGP and NTK on UCI regression dataset in the standard 20-fold splits first utilized in [139, 140]. We plot average RMSE across the splits. Different scatter points are varying hyperparameter settings of (depth, weight variance, bias variance). In the tabular data setting, dominance of NNGP is not as prominent across varying dataset as in image classification domain.

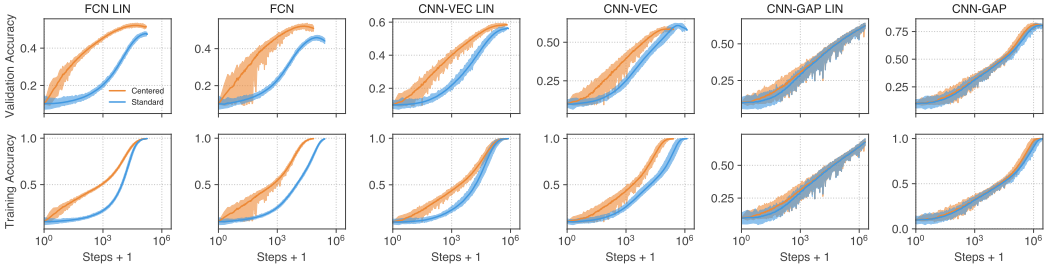

Figure S6: **Centering can accelerate training**. Validation (top) and training (bottom) accuracy throughout training for several finite width architectures. See also §3.3 and Figure 3.

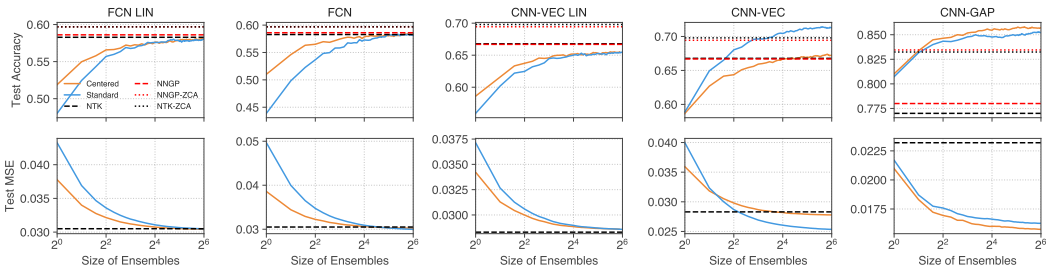

Figure S7: **Ensembling base networks causes them to match kernel performance, or exceed it for nonlinear CNNs.** See also §3.3 and Figure 4.

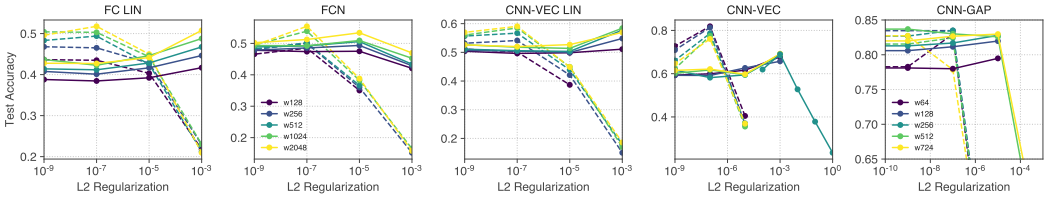

Figure S8: **Performance of nonlinear and linearized networks as a function of L2 regularization for a variety of widths.** Dashed lines are NTK parameterized networks while solid lines are networks with standard parameterization. We omit linearized `CNN-GAP` plots as they did not converge even with extensive compute budget. L2 regularization is more helpful in networks with an NTK parameterization than a standard parameterization

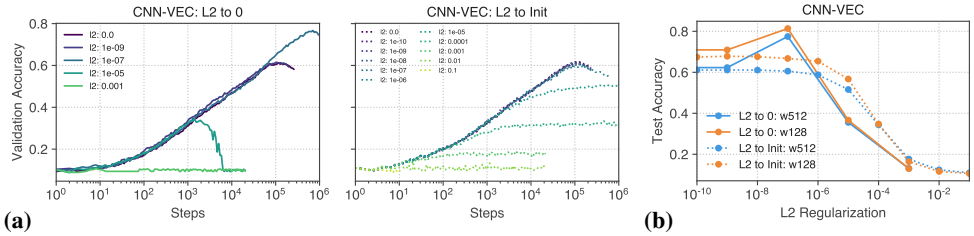

Figure S9: **L2 regularization to initial weights does not provide performance benefit. (a)** Comparing training curves of L2 regularization to either 0 or initial weights. **(b)** Peak performance of after L2 regularization to either 0 or initial weights. Increasing L2 regularization to initial weights do not provide performance benefits, instead performance remains flat until model's capacity deteriorates.

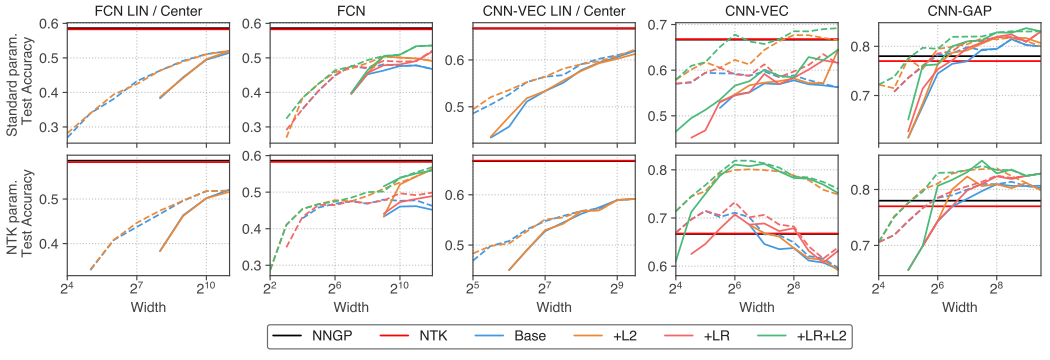

Figure S10: **Finite width networks generally perform better with increasing width, but `CNN-VEC` shows surprising non-monotonic behavior.** See also §3.6 and Figure 6 **L2**: non-zero weight decay allowed during training **LR**: large learning rate allowed. Dashed lines are allowing underfitting (**U**).

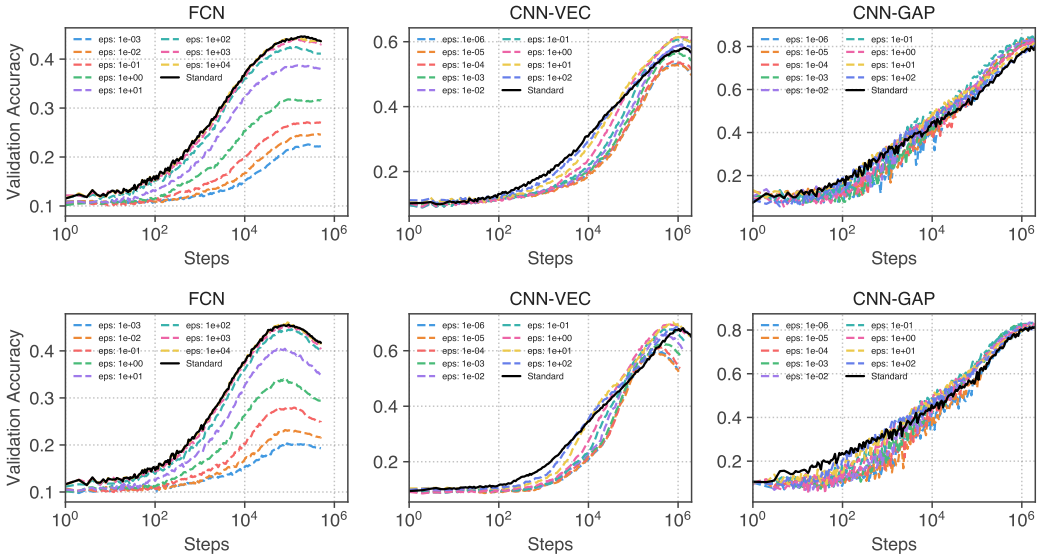

Figure S11: **ZCA regularization helps finite network training.** (**upper**) Standard parameterization, (**lower**) NTK parameterization. See also §3.10 and Figure 9.

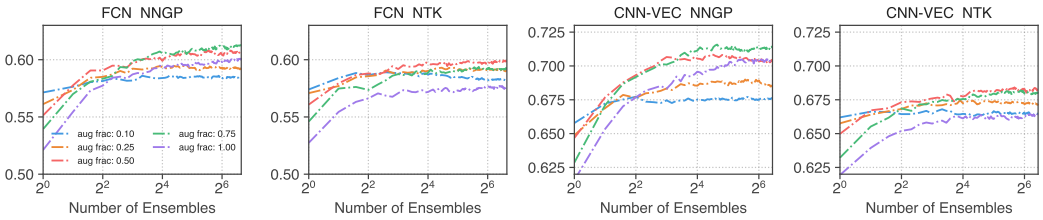

Figure S12: **Data augmentation ensemble for infinite network kernels with varying augmentation fraction.** See also §3.12.