MDL Final Report

Eugene Choi | Jeff Cui | Mark Kuang (Alphabetically Ordered)
NYU CSCI-GA 3033 Mathematics of Deep Learning
Spring 2022 Final Project - May 5, 2022

Table of Contents

1 Introduction

In recent years, the success of over-parametrized neural networks **[ZBH+16]** and the provable global convergence of sufficiently wide neural networks with highly non-convex optimization landscapes **[DZPS18]** have challenged the classical notion about the generalization and optimization in machine learning field. This mysterious behavior of the over-parametrized neural network naturally brought about the questions regarding the neural network behaviors at infinite width limit.

Early result by [N96] has shown that an infinitely wide shallow neural network at initialization is equivalent to a function sampled from a Gaussian Process (GP). Several follow-up works [MRH+18] [LXS+19] further developed this idea through the lens of kernel method, as neural networks have a Gaussian distribution described by a kernel in the infinite-width limit. However, these works mostly focused on cases in which the parameter updates were confined to the penultimate layer of the network. Recent work by [JGH18] shows that the network output of infinitely wide neural networks under gradient descent follows the kernel gradient descent with respect to a deterministic limiting neural tangent kernel (NTK). During training, the NTK also stays constant, providing a powerful tool for analyzing the optimization dynamics of neural networks. Building upon the result that the NTK remains asymptotically constant during training, follow-up work by [COB19] shows that infinitely wide neural networks enter the lazy training regime. This can be exactly represented by its first order Taylor approximation with respective to randomly initialized parameters [LXS+19].

Subsequent works on the training dynamics from the mean field theory perspective articulate another notion of "feature learning regime", where the model is not strictly linear [MMM19] [RV18b] [MMN18]. This survey aims to provide a comparison between the neural tangent kernel limit and the mean field limit for the neural network training dynamics. Connection between random feature models and the neural tangent kernel limit are also drawn to further illustrate the limitation of the NTK regime and how mean field differ from the NTK regime.

The layout of the paper is as follows. Section 2-4 covers two important training dynamics (Distributional Dynamics & Residual Dynamics) and their behavior in the infinite-width limit. Section 5 discusses the connection between the random feature models and NTK regime, and shows a separation between neural networks and their linearization. Section 6 surveys empirical results on the difference between the feature learning regime and the lazy regime and provides numerical experiments of our own to further illustrate the differing behavior of neural networks under these two regimes. The code to our numerical experiments can be found in the GitHub Repository Section.

2 Setup

Following the setup from [COB19] [MMM19], consider a shallow neural network of one hidden layer with width N:

$$f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta}) = \frac{\alpha}{N}\sum_{i=1}^{N} a_i \sigma(\boldsymbol{x};\boldsymbol{w}_i) = \frac{\alpha}{N}\sum_{i=1}^{N} \sigma_*(\boldsymbol{x};\boldsymbol{\theta}_i)$$

with $\boldsymbol{x} \in \mathbb{R}^d, \boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_N), \boldsymbol{\theta}_i = (a_i, \boldsymbol{w}_i) \in \mathbb{R}^D, \sigma : \mathbb{R}^d \times \mathbb{R}^{D-1}$ a bounded activation function. For simplicity, assume $\sigma(\boldsymbol{x};\boldsymbol{w}_i) = \sigma(\langle \boldsymbol{x}, \boldsymbol{w}_i \rangle)$.

Consider the data distribution $(\boldsymbol{x}_i, y_i) := (\boldsymbol{x}_i, f(\boldsymbol{x}_i)) \sim \mathcal{P}$, where $\mathcal{P} \in \mathscr{P}(\mathbb{R}^d \times \mathbb{R})$ is the probability distribution and $f : \mathbb{R}^d \to \mathbb{R}$ is the target function [MMM19]. To approximate f with $f_{\alpha,N}$, we minimize the population risk

$$R_{\alpha,N}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}}\Big[\ell\Big(f(\boldsymbol{x}), f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta})\Big)\Big]$$

For simplicity, assume the square loss

$$\ell\Big(f(\boldsymbol{x}), f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta})\Big) := \Big(f(\boldsymbol{x}) - f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta})\Big)^2$$

3 Distributional Dynamics and Residual Dynamics

3. 1 Gradient Flow Dynamics

Consider the gradient flow dynamics of the parameter particles $\{\boldsymbol{\theta}_j\}_{j=1}^N$

$$\frac{d}{dt}\boldsymbol{\theta}_j^t = -\frac{N}{2\alpha^2}\nabla_{\boldsymbol{\theta}_j} R_{\alpha,N}(\boldsymbol{\theta}^t) = \frac{1}{\alpha}\mathbb{E}_{\boldsymbol{x}}\Big[\Big(f(\boldsymbol{x}) - f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta})\Big)\nabla_{\boldsymbol{\theta}}\sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j)\Big] \quad \text{(GFD)}$$

where $\frac{N}{2\alpha^2}$ is a constant term for finite N and α **[MMM19] [S19]**. See the derivation in the Appendix.

3.2 Function Dynamics

Consider the function dynamics

$$\begin{aligned}\partial_t f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta}^t) &= \frac{\partial f_{\alpha,N}}{\partial t}\Big|_{(\boldsymbol{x};\boldsymbol{\theta}^t)} = \frac{\partial f_{\alpha,N}}{\partial \boldsymbol{\theta}_j^t}\frac{\partial \boldsymbol{\theta}_j^t}{\partial t}\Big|_{(\boldsymbol{x};\boldsymbol{\theta}^t)} \\ &= \mathbb{E}_{\boldsymbol{x}'}[\mathcal{K}(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta}^t)(f(\boldsymbol{x}') - f_{\alpha,N}(\boldsymbol{x}';\boldsymbol{\theta}^t))],\end{aligned}$$

where the kernel function is defined as

$$\mathcal{K}(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta}^t) := \frac{1}{N}\sum_{j=1}^N \langle \nabla_{\boldsymbol{\theta}}\sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j^t), \nabla_{\boldsymbol{\theta}}\sigma_*(\boldsymbol{x}';\boldsymbol{\theta}_j^t)\rangle.$$

See the Appendix for full derivation. Rewrite $u_t^{\alpha,N}(\boldsymbol{x}) := f(\boldsymbol{x}) - f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta}^t)$, we have

$$\partial_t u_t^{\alpha,N}(\boldsymbol{x}) = -\mathbb{E}_{\boldsymbol{x}'}[\mathcal{K}(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta}^t)u_t^{\alpha,N}(\boldsymbol{x}')] \quad \text{(FD)}.$$

3.3 Empirical Dirac Reparametrization

The gradient flow dynamics (GFD) is usually highly non-convex. So instead of describing the dynamics of individual particle, we can reparametrize GFD using the empirical Dirac measure

$$\mu_t^{\alpha,N}(d\boldsymbol{\theta}) = \frac{1}{N}\sum_{j=1}^N \delta_{\boldsymbol{\theta}_j^t}$$

So we can then rewrite our neural network as [RV18b]

$$f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta}) = f_\alpha(\boldsymbol{x};\mu) = \alpha \int \sigma_*(\boldsymbol{x};\boldsymbol{\theta})\mu(d\boldsymbol{\theta})$$

3.4 Distributional Dynamics

Consider again GFD. This time we can replace $\boldsymbol{\theta}$ with μ. Since the empirical Dirac measure is related to nonlinear Liouville equations [RV18b], we have

$$\begin{aligned}
\partial_t \mu_t^{\alpha,N} &= \frac{1}{\alpha}\nabla_{\boldsymbol{\theta}} \cdot (\mu_t^{\alpha,N}[\nabla_{\boldsymbol{\theta}}\Psi(\boldsymbol{\theta};\mu_t^{\alpha,N})]) \quad \text{(DD)} \\
\Psi_\alpha(\boldsymbol{\theta};\mu) &:= -\mathbb{E}_{\boldsymbol{x}}[(f(\boldsymbol{x}) - f_\alpha(\boldsymbol{x};\mu))\sigma_*(\boldsymbol{x};\boldsymbol{\theta})] \\
\mu_0^{\alpha,N} &= \frac{1}{N}\sum_{j=1}^N \delta_{\boldsymbol{\theta}_j^0}
\end{aligned}$$

where θ_j^0 is initialized independently from the distribution μ_0 [MMM19]. DD is called distributional dynamics. By Theorem 3 in [MMN18], the approximation error between GF and DD is well controlled while the network is overparametrized, i.e. $N \gg D$.

Thus the empirical Dirac is another way of characterizing the individual particle dynamics aggregately. See the figure for illustration.

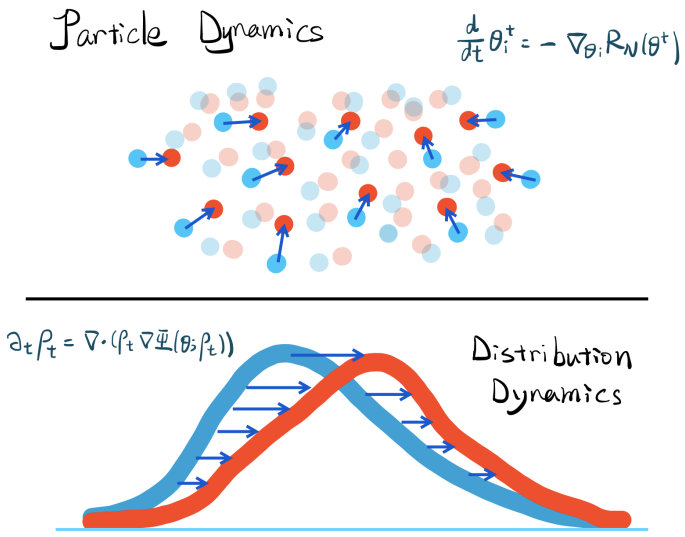


Figure: Particle Dynamics vs. Distribution Dynamics [S19]

Viewing in light of the empirical measure parametrization, as $\mu_t^{\alpha,N} \to \mu := \mu_t^{\alpha,\infty}$, we have the population risk

$$R(\mu) = \lim_{n \to \infty} R(\mu_t^{\alpha, N})$$

becomes a convex function **[MMN18] [RV18a]**. So the distributional dynamics characterization of the parameters avoids the highly non-convex problem in the particle gradient flow dynamics.

Alternatively, the DD can also be regarded as the gradient flow of $R(\mu_t^{\alpha, N})$ in the metric space $(\mathscr{P}(\mathbb{R}^D), \mathcal{W}_2)$, where the 2^{nd} Wasserstein distance $\mathcal{W}_2$ is defined as

$$\mathcal{W}_2(\mu_1, \mu_2) = \left(\inf_{\gamma \in \mathcal{C}(\mu_1, \mu_2)} \int \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2^2 \, \gamma(d\boldsymbol{\theta}_1, d\boldsymbol{\theta}_2) \right)^{1/2}$$

with discrete update formula given by

$$\mu_{t+\tau} \approx \arg\min_{\mu \in \mathscr{P}(\mathbb{R}^D)} \left\{ R(\mu) + \frac{1}{2\xi(t)\tau} \mathcal{W}_2(\mu, \mu_t)^2 \right\}$$

with $\tau \to 0$ **[MMN18] [JKO98] [AGS08] [CMV+03]**.

3.5 Residual Dynamics

Given the empirical Dirac formulation, we can reparametrize the FD as:

$$\partial_t u_t^{\alpha, N}(\boldsymbol{x}) = -\mathbb{E}_{\boldsymbol{x}'}[\mathcal{K}_{\mu_t^{\alpha, N}}(\boldsymbol{x}, \boldsymbol{x}') u_t^{\alpha, N}(\boldsymbol{x}')] \quad (\mathrm{RD}).$$

where

$$\mathcal{K}_{\mu_t^{\alpha, N}}(\boldsymbol{x}, \boldsymbol{x}') := \int \langle \nabla_{\boldsymbol{\theta}} \sigma_*(\boldsymbol{x}; \boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} \sigma_*(\boldsymbol{x}'; \boldsymbol{\theta}) \rangle \mu_t^{\alpha, N}(d\boldsymbol{\theta})$$

4 Mean Field Limit and NTK Limit

4.1 The Coupled Dynamics

Bring the DD and RD together, we have the following coupled dynamics

$$\begin{aligned} \partial_t \mu_t^{\alpha, N} &= \frac{1}{\alpha} \nabla_{\boldsymbol{\theta}} \cdot (\mu_t^{\alpha, N} [\nabla_{\boldsymbol{\theta}} \Psi_\alpha(\boldsymbol{\theta}; \mu_t^{\alpha, N})]) \quad (\mathrm{DD}) \\ \partial_t u_t^{\alpha, N}(\boldsymbol{x}) &= -\mathbb{E}_{\boldsymbol{x}'}[\mathcal{K}_{\mu_t^{\alpha, N}}(\boldsymbol{x}, \boldsymbol{x}') u_t^{\alpha, N}(\boldsymbol{x}')] \quad (\mathrm{RD}) \end{aligned}$$

The coupled dynamics in the pre-limit setup via the empirical Dirac measure reparametrization describe the evolution of the training dynamics through time under finite scaling.

4.2 Mean Field Limit

Let $\alpha = \mathcal{O}(1)$, as $N \to \infty$, we have the mean field limit of the distributional dynamics

$$\partial_t \mu_t^\alpha = \frac{1}{\alpha} \nabla \cdot (\mu_t^\alpha [\nabla_{\boldsymbol{\theta}} \Psi_\alpha(\boldsymbol{\theta}; \mu_t^\alpha)]) \quad \text{(DD-MF Limit)}$$

with the following convergence guarantee **[MMM19]**

$$\lim_{N \to \infty} \mathcal{W}_2(\mu_t^{\alpha, N}, \mu_t^\alpha) \to 0.$$

Now consider the mean field limit of the residual dynamics **[MMM19]**

$$\partial_t u_t^\alpha(\boldsymbol{x}) = -\mathbb{E}_{\boldsymbol{x}'}[\mathcal{K}_{\mu_t^\alpha}(\boldsymbol{x}, \boldsymbol{x}') u_t^\alpha(\boldsymbol{x}')] \quad \text{(RD-MF Limit)}$$

Notice that the RD-MF Limit is dependent on the kernel $\mathcal{K}_{\mu_t^\alpha}(\boldsymbol{x}, \boldsymbol{x}')$ which varies along time. So we are in the feature learning regime under the mean field limit, in comparison to the lazy regime in the neural tangent kernel limit in the next section.

4.3 Neural Tangent Kernel Limit

Let $\alpha = \mathcal{O}(N^{1/2})$. As $N \to \infty$, we have the neural tangent kernel limit of the residual dynamics

$$\partial_t u_t^*(\boldsymbol{x}) = -\mathbb{E}_{\boldsymbol{x}'}[\mathcal{K}_{\mu_0}(\boldsymbol{x}, \boldsymbol{x}') u_t^*(\boldsymbol{x}')] \quad \text{(RD-NTK Limit)}$$

where $\mathcal{K}_{\mu_0}(\boldsymbol{x}, \boldsymbol{x}')$ is a fixed kernel at initialization. Since the kernel is fixed, RD-MF Limit converges to RD-NTK Limit for $\alpha = \mathcal{O}(N^{1/2})$ and we enter the lazy regime with the explicit solution for the residual **[MMM19]** **[COB19]**

$$u_t^* = e^{-\mathcal{K}_{\mu_0} t} u_0^*.$$

Here it is shown that the first-order linear approximation of the neural network $\hat{f}(\boldsymbol{x}) = f(\boldsymbol{x}_0) + Df(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0)$ is a good approximation of $f(\boldsymbol{x})$ such that $\sup |R(f(\boldsymbol{x})) - R(\hat{f}(\boldsymbol{x}))|$ is bounded at all time step as $\alpha = \mathcal{O}(N^{1/2})$ and $N \to \infty$ **[COB19]** **[MMM19]**.

4.4 Summary of the Two Limits

For $\alpha = \mathcal{O}(1)$, $N \to \infty$, this is mean field limit **[S19]**(p. 82) **[GSJW20]**

$$\lim_{N \to \infty} f_{\alpha, N}(\boldsymbol{x}; \boldsymbol{\theta}) \approx \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} \sigma_*(\boldsymbol{x}; \boldsymbol{\theta}_i)$$

For $\alpha = N^{1/2}$, $N \to \infty$, this is neural tangent kernel limit **[S19]** (p. 82) **[GSJW20]**

$$\lim_{N\to\infty} f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta}) \approx \lim_{N\to\infty} \frac{1}{\sqrt{N}} \sum_{i=1}^{N} \sigma_*(\boldsymbol{x};\boldsymbol{\theta}_i)$$

So we get back the original NTK formulation **[JGH18]**.

5 Separation between Neural Nets and Their Linearization

For many years, numerous studies have tried to find connections between the kernel method and deep neural network, as both methods provide a powerful framework for describing a wide range of functions. **[CS09]** proposed `arc-cosine kernel`, a framework that enabled the kernel-based methods to leverage the compositional nature of the neural networks. More recent work **[LXS+19]** **[JGH18]** used kernel methods as a lens through which we can get a deeper understanding of the neural network and its training dynamics.

We know from the original NTK paper that in the lazy regime, a neural net can be approximated by its linearization around the initial weights; however, **[COB20]** remarks that it is unlikely that lazy regime explains the successes of neural nets. For a two-layer neural net,

$$f(x) = \sum_{i=1}^{N} a_i \sigma(\langle w_i, x\rangle)$$

first-order Taylor expansion at initialization $\theta_0 = (a_0, w_0)$ gives

$$\begin{aligned} & f(x) - f_0(x) \\ \approx & \sum_{i=1}^{N} (a_i - a_{0,i})\sigma(\langle w_{0,i}, x\rangle) + \sum_{i=1}^{N} a_{0,i}\langle w_i - w_{0,i}, x\rangle \sigma'(\langle w_{0,i}, x\rangle) \end{aligned}$$

[GMMM19] studies the lower bound of approximation error of neural nets in the lazy regime by bounding each term above. For initial weights drawn iid. random, the first term corresponds to random feature regression; they denote the second term the neural tangent (which usually means the entire linearization), rewritten as below:

$$\mathcal{F}_{RF} := \left\{ f(x) = \sum_{i=1}^{N} a_i \sigma(\langle w_i, x\rangle) : a_i \in \mathbb{R} \right\}$$

$$\mathcal{F}_{NT} := \left\{ f(x) = \sum_{i=1}^{N} \langle a_i, x\rangle \sigma'(\langle w_i, x\rangle) : a_i \in \mathbb{R}^d \right\}$$

In the lazy regime, as $N \to \infty$, with initial weights $w \underset{iid.}{\sim} \nu$, the random features model corresponds to regression with the kernel

$$k(x, x') = \int \sigma(\langle w, x \rangle)\, \sigma(\langle w, x' \rangle)\, \nu(dw)$$

and the neural tangent model class (as defined above) corresponds to the kernel

$$k(x, x') = \langle x, x' \rangle \int \sigma'(w^T x) \sigma'(w^T x') \nu(dw)$$

and the finite width cases can be viewed as random approximations of kernel regression with different kernels.

The setup is to fit a target function f_*, with data $(x_i, y_i)_i$ drawn such that

$$\begin{aligned} x_i &\sim \texttt{Unif}(\mathbb{S}^{d-1}(\sqrt{d})), \|x_i\|_2^2 = d \\ y_i &= f_{*,l}(x_i) \end{aligned}$$

where d is the input dimension. It is shown that in this setting, the approximation error by the linearized model is arbitrarily close to that of a degree-$(l+1)$ polynomial regression. Take sample size $n \to \infty$, limit model width $N \sim d^l$:

- $\mathcal{F}_{RF}$ has approximation error $\sim l$-th degree polynomial regression;
- $\mathcal{F}_{NT}$ has approximation error $\sim (l+1)$-th degree polynomial regression.

When the target function is a single neuron $f_*(x) = \sigma(\langle w_*, x \rangle)$, with model width polynomial in d ($N \sim d^l$), the linearized neural tangent has a generalization error bounded away from 0, assuming the activation σ is not polynomial; in contrast, a single neuron target can be learned with a single neuron model.

[GMMM20] develops a more general scenario from the single neuron setup. They hypothesize that neural networks perform better than kernel methods when the data occupies a much lower-dimensional subspace of the input dimension, and considers a target function of the form

$$f_*(x) = \varphi(U^\top x)$$

where $U \in \mathbb{R}^{d \times d_0}$ is a low-dimensional projection matrix with $d_0 \ll d$; denote $U^\perp$ the projection onto the subspace orthogonal to U, and $\varphi : \mathbb{R}^{d_0} \mapsto \mathbb{R}$ a smooth function that maps the low-dimensional projection to a label. Input x is modeled with a signal component Uz_0 and a noise component $U^\perp z_1$:

$$\begin{aligned} x &= Uz_0 + U^\perp z_1 \\ z_0 &\sim \texttt{Unif}(\mathbb{S}^{d_0 - 1}(r\sqrt{d_0})) \\ z_1 &\sim \texttt{Unif}(\mathbb{S}^{d - d_0 - 1}(\sqrt{d_0})) \end{aligned}$$

with the ball radius r modeling the signal-to-noise ratio. Setting $d_0 = 1$ recovers the single neuron case. They show that when $d_0 \ll d$, kernel ridge regression performance degrades much faster than neural nets when the input is perturbed by noise ($r \to 1^+$). Image classification problems satisfy this setup: input image can be well-represented by its low-frequency components, and the class label can still be predicted with this

approximation. By adding high-frequency noise to the *Fashion MNIST* dataset inputs, they observe that kernel ridge regression models degrade faster than neural nets.

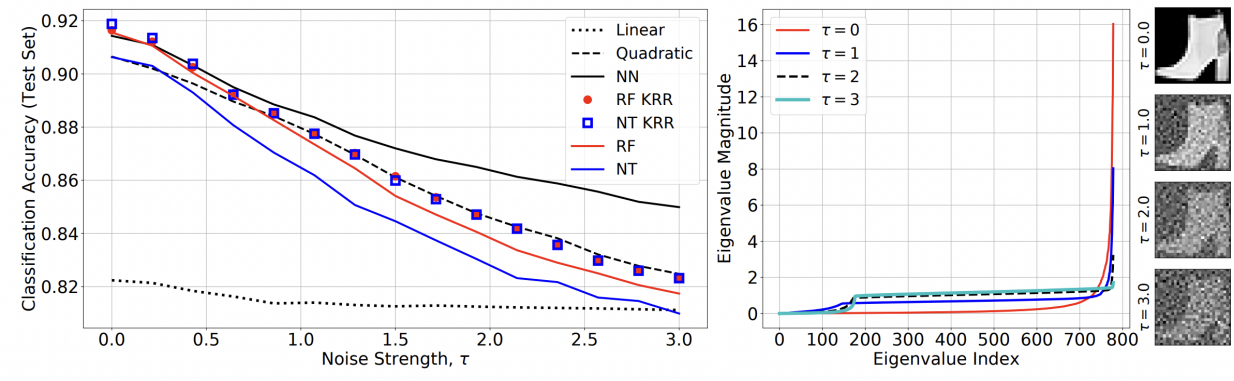


Figure: neural nets are more robust to high-frequency noise than kernel methods [GMMM20]

6 Comparisons between the feature learning regime and the lazy regime

To illustrate the relationship between the lazy and the feature training regimes studied by **[GSJW20]**, we re-illustrate some notations the authors use differently from **[MMM19]**:

- $f(w, x)$ is a deep network defined as:

$$f(w, x) = h^{-1/2} W^L z^L$$

  (Note: the network is defined with the scaling $h^{-1/2}$ instead of h^{-1}.)

- h^{ℓ} is the number of hidden neurons at ℓ^{th} hidden layer.
- w_0 is the parameter at initialization.

We try to minimize the function $F(w, x)$ defined as

$$F(w, x) \equiv \alpha[f(w, x) - f(w_0, x)]$$

using

$$\mathcal{L}(w) = \frac{1}{\alpha^2 |\mathcal{D}_{train}|} \sum_{(x,y) \in \mathcal{D}_{train}} \ell(\alpha\big(f(w, x) - f(w_0, x)\big), y).$$

The penultimate layer weight scales as $\alpha h^{-1/2}$ at initialization, and varying α, h allows us to explore both the lazy and the feature-training regimes as follows:

- If α scale is constant ($\alpha = \mathcal{O}(1)$), then we enter the lazy-training regime. In this case, a frozen kernel controls the training dynamics.
- If α scales with respect to h ($\alpha = \mathcal{O}(h^{-1/2})$), then we enter the feature-training regime. In this case, the features are learned and the tangent kernel evolves during training.

With these in mind, the authors present three interesting points about the NTK limit and the mean-field limit:

1. **Delineation of the training regimes through test performance:**

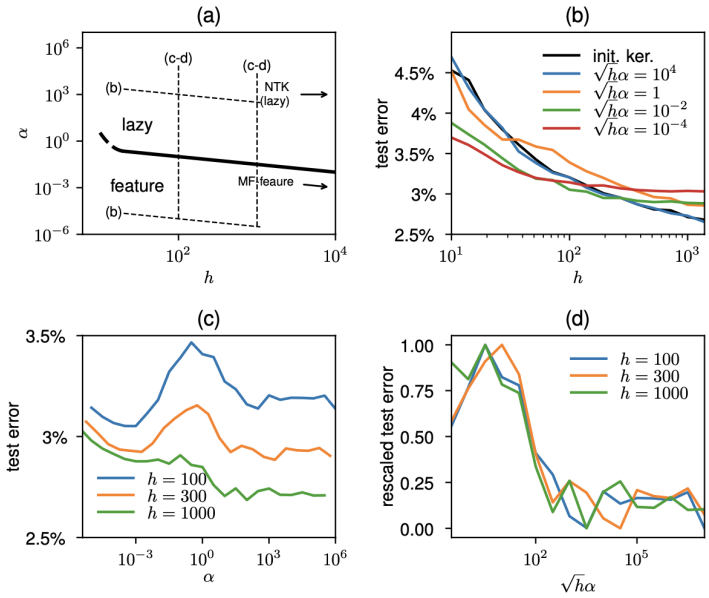


Fig. 3 [GSJW20] the four subplots illustrate the existence of lazy & feature training regimes. Higher h results in a better predictive performance (b). Here, $\sqrt{h}\alpha = 10^4, 1$ refer to lazy regime and $10^{-2}, 10^{-4}$ refer to feature training regime.

- There is a value of α for which we leave the lazy-training regime and enter a nonlinear regime that we call feature training.
- Figure (a) shows the solid black line in the middle that marks the moment we either enter the lazy-training regime or the non-linear feature-training regime. The same delineating point that separates the two regimes can also be found in figure (c), where the test error spikes up for different h.

2. **Output variance and the ensembling effect:**

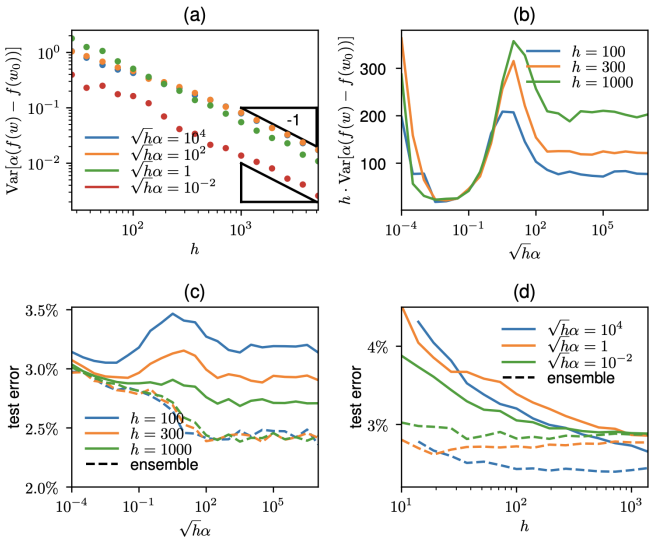


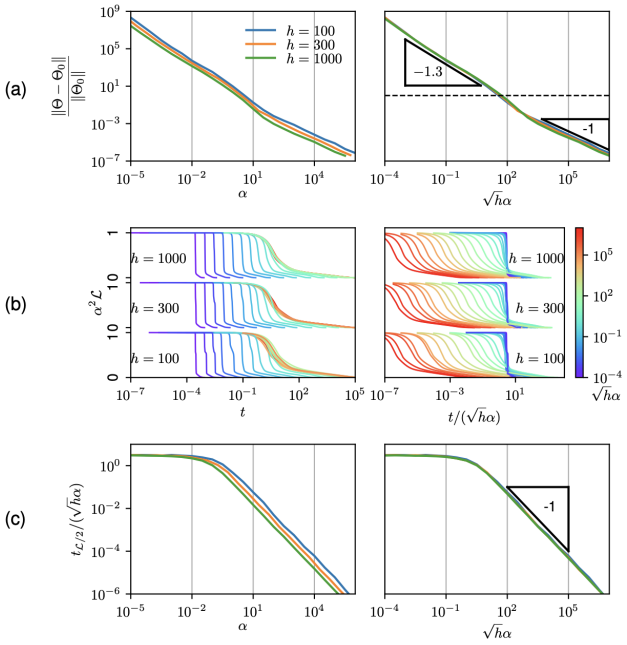
Fig. 4 [GSJW20]- the four subplots show the fluctuations (the variance) of the network output with varying degrees of α and h (high $\alpha \to$ lazy & low $\alpha \to$ feature training).

- Fig. 4(a) shows $\mathrm{Var}(\alpha(f - f_0)) \sim h^{-1}$, or the fluctuations in both lazy and feature-training regimes decreases as the $h \to \infty$ and their limiting behaviors get more pronounced. That is, the fluctuations stemming from the initialization decreases as the h got larger.
    - (Significance) this explains why larger width lead to a better performance.
- A natural question that follows is: how does ensembling, a popular variance reduction technique, compare with the result about increasing h above?

$$\bar{F}(x) \equiv \frac{\alpha}{20} \sum_{i=1}^{20} \big(f(w_i, x) - f(w_{i,0}, x)\big)$$

- The authors present that the ensemble averaging alleviates fluctuations in both regimes.

3. **The difference in the training dynamics of the two regimes:**



- From (a), the fact that the curves denoting the relative variation of the feature-learning kernel at the end of training collapse by changing the variable from α to $\sqrt{h}\alpha$ means that $\sqrt{h}\alpha$ is the parameter that controls feature training. Most importantly, the feature learning follows below with $a \approx 1.3$:

$$\frac{\|\Theta(w) - \Theta(w_0)\|}{\|\Theta(w_0)\|} \sim \left(\sqrt{h}\alpha\right)^{-a}$$

- In the NTK regime, there is no change as the kernel is frozen.

The paper concludes with the following result:

As the width (h) and the output scale (α) are varied, both the feature-training regime (in which the tangent kernel evolves during training) and lazy-training (in which a frozen kernel controls the dynamics) regime depend on the value of $\sqrt{h}\alpha$, *the parameter that controls training.*

Additionally, the authors present a point about the depth and generalization capability of kernel learning:

- Depth has almost no effect on the scaling of the test error with $\sqrt{h}\alpha$. But it decreases the performance of the individual network.
- Kernel learning with the NTK obtained using a trained network leads to nearly the same generalization error as regular inference with the trained network.

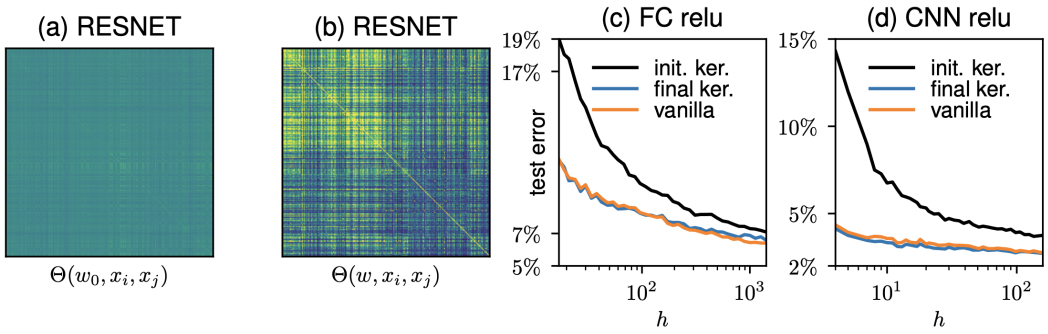


Figure 8: Gram matrix of the test set at initialization $\Theta(w_0, x_\mu, x_\nu)$ *(a)* and at the end of training $\Theta(w, x_\mu, x_\nu)$ *(b)*, for a wide-resnet 28x10 architecture (Zagoruyko and Komodakis, 2016) ($L = 25$ hidden layers) trained on a binary version of CIFAR10. The first half of the indices $\mu = 1 \dots n/2$ has label $y = 1$ and the other half has label $y = -1$. The kernel inflates during learning in a way that depends on the two classes. See Appendix A for a description of the architecture. *(c-d)* Test error *v.s.* the width h for the regular dynamics, the dynamics with the frozen kernel at initialization and the dynamics with the frozen kernel of the end of training. The training performance is captured by the kernel.

Kernel learning with the NTK using a trained network results in approximately same generalization error as that of the regular NN results.

From these insights, we conducted a simple experiment using the NTK on the MNIST dataset using the MLP and the CNN architectures with varying degrees of width size.

To classify with the kernel, we followed the classification method following [ADH+19], which uses:

$$f^*(\boldsymbol{x}) = \big(\ker(\boldsymbol{x}, \boldsymbol{x}_1), \ker(\boldsymbol{x}, \boldsymbol{x}_2), \dots, \ker(\boldsymbol{x}, \boldsymbol{x}_n)\big) \cdot \big(\boldsymbol{H}^*\big)^{-1} \boldsymbol{Y}$$

To construct $\boldsymbol{H}^*$, we used 30 samples per digit from $\mathcal{D}_{train}$ (total 300 samples). Using this, we tried a different number of hidden layers (or a number of channels in the case of CNN) to see how the varying degree of width affects the generalization capability of the architectures. We used the [HZ21] library to compute the NTK. We tested the kernel induced by different architectures approximating different regimes on an MNIST test-set of $10,000$ samples.

- NTK performances (accuracy) on MNIST test set:

Model	h/ch	N	Lazy	Feature	Regular Inf.
MLP (S)	$1,024$	$2,913,290$	0.7705	0.9609	0.9859
MLP (M)	$2,048$	$10,020,874$	0.7851	0.9618	0.9863
MLP (L)	$4,096$	$36,818,954$	0.7885	0.9509	0.9861
MLP (XL)	$8,192$	$140,746,762$	0.7966	(GPU limit)	(GPU limit)
CNN (S)	32	$173,706$	0.5983	0.7774	0.9903
CNN (M):	64	$384,266$	0.6299	0.8187	0.9905
CNN (L):	128	$915,978$	0.6433	0.7911	0.9885

Model	h/ch	N	Lazy	Feature	Regular Inf.
CNN (XL)	256	$2,421,770$	0.6522	0.7335	0.9855

Our observations mainly agrees with the reports from **[COB19]**:

- In **[COB19]**, based on teacher-student numerical experiments, the authors argued that the NTK limit is unlikely to explain the success of neural networks.

Our results agree with those on the paper for CNN, which states that feature training tends to perform better.

We also find that feature-learning works better for the fully-connected networks, which disagrees with the point on the paper that lazy training tends to outperform feature learning. We believe that this is probably due to the discrepancy in the implementation detail. Mainly, we opted for gradient descent, whereas the authors implemented gradient flow.

7 Conclusion

In this report, we have compared the distributional dynamics and residual dynamics respectively in the mean field limit and the neural tangent kernel limit.

From the empirical results, we investigated the relationship between the lazy and feature-training regimes further in-depth by differentiating them by their performance, output variance, and training dynamics. We additionally tried validating the results observed in the papers through a simple experiment. Still, our results did not exactly match those presented in the papers due to a few implementation details.

Recent papers also established a separation in approximation error between the active and lazy regimes in certain scenarios. While these scenarios (single neuron target, or activation after affine on noisy data) seem constructed to favor neural nets, they show that there exists a separation between neural nets and kernel methods.

References

[ADH+19] Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., & Wang, R. (2019). On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems*, *32*.

[AGS08] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. Gradient flows: in metric spaces and in the space of probability measures. Springer Science & Business Media, 2008.

[CMV+03] José A Carrillo, Robert J McCann, Cédric Villani, et al. Kinetic equilibration rates for granular media and related equations: entropy dissipation and mass transportation estimates. *Revista Matematica Iberoamericana*, 19(3):971–1018, 2003.

[COB19] Chizat, Lenaic, Edouard Oyallon, and Francis Bach. "On lazy training in differentiable programming." *Advances in Neural Information Processing Systems* 32 (2019).

[CS09] Cho, Y., & Saul, L. (2009). Kernel methods for deep learning. *Advances in neural information processing systems*, *22*.

[DZPS18] Du, S. S., Zhai, X., Poczos, B., & Singh, A. (2018). Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*.

[GMMM19] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Linearized two-layers neural networks in high dimension. *arXiv preprint* arXiv:1904.12191, 2019.

[GMMM20] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. When do neural networks outperform kernel methods? *Advances in Neural Information Processing Systems* 33 (2020): 14820-14830.

[GSJW20] Geiger, M., Spigler, S., Jacot, A., & Wyart, M. (2020). Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2020*(11), 113301.

[HZ21] He, H., Zou, R. (2021). functorch: JAX-like composable function transforms for PyTorch. https://github.com/pytorch/functorch.

[JGH18] Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, *31*.

[JKO98] Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the Fokker–Planck equation. SIAM journal on mathematical analysis, 29(1):1–17, 1998.

[LXS+19] Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., & Pennington, J. (2019). Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, *32*.

[MMM19] Mei, S., Misiakiewicz, T., & Montanari, A. (2019, June). Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. In *Conference on Learning Theory* (pp. 2388-2464). PMLR.

[MMN18] Mei, S., Montanari, A., & Nguyen, P. M. (2018). A mean field view of the landscape of two-layer neural networks. Proceedings of the National Academy of Sciences, 115(33), E7665-E7671.

[MRH+18] Matthews, A. G. D. G., Rowland, M., Hron, J., Turner, R. E., & Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*.

[N96] Neal, R. M. (1996). Priors for infinite networks. In *Bayesian Learning for Neural Networks* (pp. 29-53). Springer, New York, NY.

[RV18a] Rotskoff, G., & Vanden-Eijnden, E. (2018). Parameters as interacting particles: long time convergence and asymptotic error scaling of neural networks. *Advances in neural information processing systems*, *31*.

[RV18b] Rotskoff, G. M., & Vanden-Eijnden, E. (2018). Trainability and accuracy of neural networks: An interacting particle system approach. *arXiv preprint arXiv:1805.00915*.

[S19] Mei, S. Mean field theory and tangent kernel theory of neural networks. https://stats385.github.io/assets/lectures/MF_dynamics_Stanford.pdf

[ZBH+16] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization (2016). *arXiv preprint arXiv:1611.03530*.

Appendix

GitHub Repository

The code to the numerical experiments can be found at here.

Derivation of the Gradient Flow Dynamics

Consider the gradient flow dynamics of the parameter particles $\{\boldsymbol{\theta}_j\}_{j=1}^N$ with the constant term $\frac{N}{2\alpha^2}$ for finite N and α **[MMM19] [S19]**

$$\frac{d}{dt}\boldsymbol{\theta}_j^t = -\frac{N}{2\alpha^2}\nabla_{\boldsymbol{\theta}_j} R_{\alpha,N}(\boldsymbol{\theta}^t)$$

Notice that the gradient of the population risk with respect to $\boldsymbol{\theta}_j$ is given by

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_j} R_{\alpha,N}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}_j} \mathbb{E}_{\boldsymbol{x}}\Big[\Big(f(\boldsymbol{x}) - f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta})\Big)^2\Big] \\
&= \nabla_{\boldsymbol{\theta}_j} \mathbb{E}_{\boldsymbol{x}}\Big[\Big(f(\boldsymbol{x}) - \frac{\alpha}{N}\sum_{j=1}^N \sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j)\Big)^2\Big] \\
&= \nabla_{\boldsymbol{\theta}_j} \int_{\boldsymbol{x}\sim\mathcal{P}_{\boldsymbol{x}}} \Big(f(\boldsymbol{x}) - \frac{\alpha}{N}\sum_{j=1}^N \sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j)\Big)^2 \cdot p(\boldsymbol{x}) d\boldsymbol{x} \\
&= \int_{\boldsymbol{x}\sim\mathcal{P}_{\boldsymbol{x}}} \nabla_{\boldsymbol{\theta}_j} \Big(f(\boldsymbol{x}) - \frac{\alpha}{N}\sum_{j=1}^N \sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j)\Big)^2 \cdot p(\boldsymbol{x}) d\boldsymbol{x} \\
&= \int_{\boldsymbol{x}\sim\mathcal{P}_{\boldsymbol{x}}} 2\Big(f(\boldsymbol{x}) - \frac{\alpha}{N}\sum_{j=1}^N \sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j)\Big) \cdot \Big(-\frac{\alpha}{N}\Big) \cdot \nabla_{\boldsymbol{\theta}_j}\sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j) \cdot p(\boldsymbol{x}) d\boldsymbol{x} \\
&= -\frac{2\alpha}{N} \int_{\boldsymbol{x}\sim\mathcal{P}_{\boldsymbol{x}}} \Big(f(\boldsymbol{x}) - \frac{\alpha}{N}\sum_{j=1}^N \sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j)\Big) \cdot \nabla_{\boldsymbol{\theta}_j}\sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j) \cdot p(\boldsymbol{x}) d\boldsymbol{x} \\
&= -\frac{2\alpha}{N} \mathbb{E}_{\boldsymbol{x}}\Big[\Big(f(\boldsymbol{x}) - f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta})\Big) \cdot \nabla_{\boldsymbol{\theta}_j}\sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j)\Big]
\end{aligned}$$

So we get

$$\begin{aligned}
\frac{d}{dt}\boldsymbol{\theta}_j^t &= -\frac{N}{2\alpha^2}\nabla_{\boldsymbol{\theta}_j} R_{\alpha,N}(\boldsymbol{\theta}^t) \\
&= -\frac{N}{2\alpha^2} \cdot \Big(-\frac{2\alpha}{N}\mathbb{E}_{\boldsymbol{x}}\Big[\Big(f(\boldsymbol{x}) - f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta})\Big) \cdot \nabla_{\boldsymbol{\theta}_j}\sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j)\Big]\Big) \\
&= \frac{1}{\alpha}\mathbb{E}_{\boldsymbol{x}}\Big[\Big(f(\boldsymbol{x}) - f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta})\Big)\nabla_{\boldsymbol{\theta}}\sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j)\Big] \quad \text{(GFD)}. \ \square
\end{aligned}$$

Derivation of the Function Dynamics

The function dynamics is given by

$$
\begin{aligned}
\partial_t f_{\alpha,N}(\boldsymbol{x};\boldsymbol{\theta}^t) &= \frac{\partial f_{\alpha,N}}{\partial t}\Big|_{(\boldsymbol{x};\boldsymbol{\theta}^t)} = \frac{\partial f_{\alpha,N}}{\partial \boldsymbol{\theta}_j^t}\frac{\partial \boldsymbol{\theta}_j^t}{\partial t}\Big|_{(\boldsymbol{x};\boldsymbol{\theta}^t)} \\
&= \frac{\alpha}{N}\sum_{j=1}^{N}\Big\langle \nabla_{\boldsymbol{\theta}}\sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j^t), \frac{\partial \boldsymbol{\theta}_j^t}{\partial t}\Big|_{(\boldsymbol{x};\boldsymbol{\theta}^t)}\Big\rangle \\
&= \frac{\alpha}{N}\sum_{j=1}^{N}\Big\langle \nabla_{\boldsymbol{\theta}}\sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j^t), \frac{1}{\alpha}\mathbb{E}_{\boldsymbol{x}'}\Big[\Big(f(\boldsymbol{x}') - f_{\alpha,N}(\boldsymbol{x}';\boldsymbol{\theta}^t)\Big)\nabla_{\boldsymbol{\theta}}\sigma_*(\boldsymbol{x}';\boldsymbol{\theta}_j^t)\Big]\Big\rangle \\
&= \mathbb{E}_{\boldsymbol{x}'}\Big[\Big(f(\boldsymbol{x}') - f_{\alpha,N}(\boldsymbol{x}';\boldsymbol{\theta}^t)\Big)\frac{1}{N}\sum_{j=1}^{N}\langle \nabla_{\boldsymbol{\theta}}\sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j^t), \nabla_{\boldsymbol{\theta}}\sigma_*(\boldsymbol{x}';\boldsymbol{\theta}_j^t)\rangle\Big] \\
&= \mathbb{E}_{\boldsymbol{x}'}\Big[\mathcal{K}(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta}^t)\Big(f(\boldsymbol{x}') - f_{\alpha,N}(\boldsymbol{x}';\boldsymbol{\theta}^t)\Big)\Big]
\end{aligned}
$$

with the kernel function defined as

$$
\mathcal{K}(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta}^t) := \frac{1}{N}\sum_{j=1}^{N}\langle \nabla_{\boldsymbol{\theta}}\sigma_*(\boldsymbol{x};\boldsymbol{\theta}_j^t), \nabla_{\boldsymbol{\theta}}\sigma_*(\boldsymbol{x}';\boldsymbol{\theta}_j^t)\rangle. \square
$$