# Implicit Bias of Linear RNNs

**Melikasadat Emami** [1] **Mojtaba Sahraee-Ardakan** [1 2] **Parthe Pandit** [1 2] **Sundeep Rangan** [3] **Alyson K. Fletcher** [1 2]

## Abstract

Contemporary wisdom based on empirical studies suggests that standard recurrent neural networks (RNNs) do not perform well on tasks requiring long-term memory. However, RNNs' poor ability to capture long-term dependencies has not been fully understood. This paper provides a rigorous explanation of this property in the special case of linear RNNs. Although this work is limited to linear RNNs, even these systems have traditionally been difficult to analyze due to their non-linear parameterization. Using recently-developed kernel regime analysis, our main result shows that as the number of hidden units goes to infinity, linear RNNs learned from random initializations are functionally equivalent to a certain weighted 1D-convolutional network. Importantly, the weightings in the equivalent model cause an implicit bias to elements with smaller time lags in the convolution, and hence shorter memory. The degree of this bias depends on the variance of the transition matrix at initialization and is related to the classic exploding and vanishing gradients problem. The theory is validated with both synthetic and real data experiments.

## 1. Introduction

Over the past decade, models based on neural networks have become commonplace in virtually all machine learning applications. A key feature about this class of models is that, in principle, they do not require an explicit design of features but instead rely on an implicit learning of "*meaningful*" representations of the data. This makes neural networks attractive from the point of view of machine perception where raw signals are inputs to the model. While neural networks have become ubiquitous in practical applications, several theoretical aspects of them largely remain a mystery. Of significant importance is the lack of understanding of the potential implicit biases that these representations bring to the predictions of the model.

This paper aims to understand the *implicit bias* behaviour of Recurrent Neural Networks (RNN). Several machine learning tasks require dealing with sequential data with possibly varying lengths of input sequences. Example tasks include automatic speech recognition, language translation, and image captioning, among others. A well-known shortcoming of standard RNNs trained using gradient descent is their poor performance at tasks requiring long-term dependence (Bengio et al., 1994). Traditional RNNs suffer from the well-known vanishing / exploding gradient problem where gradients from input time samples at long delays either decay or explode making the learning of long-term effects difficult. In practice, there have been a large number of enhancements to RNNs to overcome these issues – most notably LSTMs (Hochreiter & Schmidhuber, 1997) and GRUs (Cho et al., 2014). However, while these methods have had tremendous practical success, the ability of different RNN architectures to capture or not capture long-term dependencies has been difficult to rigorously analyze.

As a simple starting point, this paper seeks to precisely understand the training and memory of *linear* RNNs. Even such linear models have been difficult to analyze completely due to their non-linear parameterization. Our main result shows rigorously that these models have an implicit bias toward short-term memory, confirming the qualitative empirically observed behavior of these networks. To our knowledge, this paper is the first that provides a such a characterization of an implicit bias in RNNs.

Our results are based on the key observation that linear RNNs are functionally equivalent to a 1D-convolutional model which is feed-forward in nature. Further, due to the Neural Tangent Kernel (NTK) regime based analysis (Jacot et al., 2018), we are able to show that RNNs trained using gradient descent are implicitly biased towards learning tasks with short-term contexts. Our result holds in a certain wide limit regime where the number of hidden units in RNN goes to infinity. The contributions are as follows:

- We explicitly compute the NTK for a linear RNN. This

[1]Department of Electrical and Computer Engineering, University of California, Los Angeles, Los Angeles, USA [2]Department of Statistics, University of California, Los Angeles, Los Angeles, USA [3]Department of Electrical and Computer Engineering, New York University, Brooklyn, New York, USA. Correspondence to: Melikasadat Emami <emami@ucla.edu>.

*Proceedings of the $38^{th}$ International Conference on Machine Learning*, PMLR 139, 2021. 

is challenging due to the weight sharing in RNNs which leads to statistical dependencies across time. We calculate this NTK using a conditioning technique as in (Bayati & Montanari, 2011) to deal with the dependencies. This NTK is also calculated in (Alemohammad et al., 2020) using the Tensor program results of (Yang, 2019a).

- We show that the linear RNN NTK is equivalent to the NTK of a scaled convolutional model with certain scaling coefficients. This means that in the wide limit regime (number of hidden units in RNN $\to \infty$), gradient descent training of a linear RNN with non-linear parameterization is identical to the training of an appropriately scaled convolutional model.

- The above results rigorously show that there is an implicit bias in using the non-linear parameterization associated with a linear RNN. In particular, training linear RNNs with non-linear parameterization using gradient descent is implicitly biased towards short memory.

- We demonstrate the bias-variance trade-off of linear RNNs in experiments on synthetic and real data.

### 1.1. Prior Work

The connection between kernel methods and infinite width neural networks was first introduced in (Neal, 1996). Neural networks in the infinite width limit are equivalent to Gaussian processes at initialization and several papers have investigated the correspondence to kernel methods for a variety of architectures (Lee et al., 2018; Novak et al., 2019; Garriga-Alonso et al., 2019; Yang, 2019a; Daniely et al., 2016; Matthews et al., 2018). In particular, (Daniely et al., 2016) introduced a framework to link a reproducing kernel to the neural network and stochastic gradient descent was shown to learn any function in the corresponding RKHS if the network is sufficiently wide (Daniely, 2017).

A recent line of work has shown that gradient descent on over-parameterized networks can achieve zero training error with parameters very close to their initialization (Allen-Zhu et al., 2018; Du et al., 2018a;b; Li & Liang, 2018; Zou et al., 2018). The analysis of the generalization error in this high dimensional regime led to exact characterizations of the test error for different architectures (Montanari et al., 2019; Hastie et al., 2019; Belkin et al., 2019; Emami et al., 2020; Barbier et al., 2019; Gerbelot et al., 2020). In addition to convergence to a global minimum for an over-parameterized two-layer neural network, (E et al., 2020) also showed that the resulting functions are uniformly close to the ones found with the kernel regime. It was shown by (Jacot et al., 2018) that the behavior of an infinitely wide fully-connected neural network trained by gradient descent is characterized by the so-called Neural Tangent Kernel (NTK) which is essentially the linearization of the network around its initialization. The NTK was later extended for different architectures (Arora et al., 2019; Yang, 2019a;b; Alemohammad et al., 2020).

A different line of papers investigated the over-parameterized neural networks from the mean field viewpoint (Mei et al., 2018; Wei et al., 2019; Ma et al., 2019; Dou & Liang, 2020; Sirignano & Spiliopoulos, 2020; Rotskoff & Vanden-Eijnden, 2018). For recurrent neural networks in particular, (Chen et al., 2018) has provided a theory for signal propagation in these networks which could predict their trainability. The authors also give a closed-form initialization to improve the conditioning of input-output Jacobian.

Trainablility of RNNs has also been improved by using orthogonal/unitary weight matrices (Arjovsky et al., 2016; Wisdom et al., 2016; Jing et al., 2017; Emami et al., 2019). It has been shown in (Emami et al., 2019) that for RNNs with ReLU activations, there is no loss in the expressiveness of the model when imposing the unitary constraint.

Note that in an RNN the states are correlated due to weight sharing. Previous work such as (Chen et al., 2018), has simplified the setting by assuming an independence over RNN weights (ignoring the correlation) to show that the pre-activations are Gaussian distributed. In this work, taking into account these dependencies, we use techniques used in (Bayati & Montanari, 2011; Rangan et al., 2019) to characterize the behaviour of RNNs at initialization. Similar techniques have also been explored in (Yang, 2019a).

We should mention that learning the weight matrices of a linear RNN using data is essentially a system identification task. There is a large body of literature in control theory that consider the system identification problem and propose many different methods to find a system that matches the input-output behavior of a given system. These methods include the prediction error method (PEM), subspace methods, empirical transfer function estimate (ETFE), correlation method, spectral analysis method, and sequential Monte Carlo method to name a few. For a more comprehensive list of system identification methods and details see (Ljung, 1999; Ljung & Glad, 1994; Lennart, 1999; Katayama, 2006; Viberg, 1995; Söderström & Stoica, 1989; Pintelon & Schoukens, 2012; Sjöberg et al., 1995). Even though it would be interesting to see how different system identification methods can be incorporated into neural network training pipeline, the vast majority of works currently learn the weights directly by optimizing a loss function via gradient descent or its variants. As such, in this work we solely focus on training of linear RNNs using gradient descent.

## 2. Linear RNN and Convolutional Models

**Linear RNNs** We fix a time period $T$ and consider a linear RNN mapping an input sequence $\mathbf{x} = (x_0, \ldots, x_{T-1})$ to an ouptut sequence $\mathbf{y} = (y_0, \ldots, y_{T-1})$ via the updates

$$h_t = \frac{1}{\sqrt{n}} W h_{t-1} + F x_t,$$
$$y_t = \frac{1}{\sqrt{n}} C h_t, \quad t = 0, \ldots, T-1, \tag{1}$$

with the initial condition $h_{-1} = \mathbf{0}$. We let $n_x$, $n_y$, and $n$ be the dimension at each time of the input, $x_t$, output, $y_t$, and hidden state $h_t$ respectively. Note that a bias term can be added for $h_t$ by extending $x_t$ and $F$. We will let

$$\mathbf{y} = f_{\rm RNN}(\mathbf{x}, \theta_{\rm RNN}) \tag{2}$$

denote the mapping (1) where $\theta_{\rm RNN}$ are the parameters

$$\theta_{\rm RNN} = (W, F, C). \tag{3}$$

The goal is to learn parameters $\theta_{\rm RNN}$ for the system from $N$ training data samples $(\mathbf{x}^i, \mathbf{y}^i)$, $i = 1, \ldots, N$. In this case, each sample $(\mathbf{x}^i, \mathbf{y}^i)$ is a $T$-length input-output pair.

**Wide System Limit** We wish to understand learning of this system in the *wide-system limit* where the number of hidden units $n \to \infty$ while the dimensions $n_x, n_y$ and number of time steps $T$ are fixed. Since the parameterization of the RNN is non-linear, the initialization is critical. For each $n$, we will assume that the parameters $W, F, C$ are initialized with i.i.d. components,

$$W_{ij} \sim \mathcal{N}(0, \nu_W), \quad F_{ij} \sim N(0, \nu_F), \quad C_{ki} \sim N(0, \nu_C), \tag{4}$$

for constants $\nu_W, \nu_F, \nu_C$.

**Stability** In the initialization (4), $\nu_W$ is the variance of the components of the kernel matrix $W$. One critical aspect in selecting $\nu_W$ is the *stability* of the system. A standard result in linear systems theory (see, e.g. (Kailath, 1980)) is that the system (1) is stable if and only if $\frac{1}{\sqrt{n}}\lambda_{max}(W) < 1$ where $\lambda_{\max}(W)$ is the maximum absolute eigenvalue of $W$ (i.e. the spectral radius). Stable $W$ are generally necessary for linear RNNs: Otherwise bounded inputs $x_t$ can result in outputs $y_t$ that grow unbounded with time $t$. Hence, training will be numerically unstable. Now, a classic result in random matrix theory (Bai & Yin, 1986) is that, since the entries of $W$ are i.i.d. Gaussian $\mathcal{N}(0, \nu_W)$,

$$\lim_{n\to\infty} \frac{1}{\sqrt{n}} \lambda_{\max}(W) = \nu_W$$

almost surely. Hence, for stability we need to select $\nu_W < 1$. As we will see below, it is this constraint that will limit the ability of the linear RNN to learn long-term memory.

**Scaled 1D Convolutional Equivalent Systems:** Our main result will draw an equivalence between the learning of linear RNNs and certain types of linear convolutional models. Specifically, consider a *linear convolutional model* of the form,

$$y_t = \sum_{j=0}^{t} L_j x_{t-j}, \tag{5}$$

where $L_j \in \mathbb{R}^{n_y \times n_x}$ are the filter coefficient matrices. In neural network terminology, the model (5) is a simply a linear 1D convolutional network with $n_x$ input channels, $n_y$ output channels and $T$-wide kernels.

Both the linear RNN model (1) and the 1D convolutional model (5) define linear mappings of $T$-length input sequences $\mathbf{x}$ to $T$-length output sequences $\mathbf{y}$. To state our equivalence result between these models, we need to introduce a certain *scaled parametrization*: Fix a set of scaling factors $\rho = (\rho_0, \ldots, \rho_{T-1})$ where $\rho_j > 0$ for all $j$. Define the parameters

$$\theta_{\rm conv} = (\theta_0, \ldots, \theta_{T-1}), \tag{6}$$

where $\theta_j \in \mathbb{R}^{n_y \times n_x}$. Given any $\theta$, let the impulse response coefficients be

$$L_j = \sqrt{\rho_j}\theta_j. \tag{7}$$

As we will see below, the effect of the weighting is to favor certain coefficients $L_j$ over others during training: For coefficients $j$ where $\rho_j$ is large, the fitting will tend to select $L_j$ large if needed. This scaling will be fundamental in understanding implicit bias.

Now, given a set of weights $\rho = (\rho_0, \ldots, \rho_{T-1})$, let

$$\mathbf{y} = f_{\rm conv}(\mathbf{x}, \theta_{\rm conv}) := \left\{ \sum_{j=0}^{t} \sqrt{\rho_j}\theta_j x_{t-j} \right\}_{t=0}^{T-1}, \tag{8}$$

denote the mapping of the input $\mathbf{x} = (x_0, \ldots, x_{T-1})$ through the convolutional filter $\theta_{\rm conv}$ with filter coefficients $\rho$ to produce the output sequence $\mathbf{y} = (y_0, \ldots, y_{T-1})$.

It is well-know that the RNN and convolutional models define the same total set of input-output mappings as given by the following standard result:

**Proposition 2.1.** *Consider the linear RNN model* (2) *and the 1D convolutional model* (8)*.*

(a) *Given a linear RNN parameters $\theta_{\rm RNN} = (W, F, C)$, a 1D convolutional filter with coefficient matrices*

$$L_j = \frac{1}{n^{\frac{j+1}{2}}} C W^j F, \quad j = 0, \ldots, T-1, \tag{9}$$

*will have an identical input-output mapping. That is, there exists parameters $\theta_{\rm conv}$ such that*

$$f_{\rm RNN}(x, \theta_{\rm RNN}) = f_{\rm conv}(x, \theta_{\rm conv}), \tag{10}$$

*for all inputs $x$.*

(b) *Conversely, given any $T$ filter coefficients $\{L_j\}_{j=0}^{T-1}$, there exists RNN model with $n \leq Tn_xn_y$ hidden states such that the RNN and 1D convolutional model have identical input-output mappings over $T$-length sequences.*

*Proof.* These are standard results from linear systems theory (Kailath, 1980). In the linear systems theory, the coefficients $L_i$ are together called the *matrix impulse response*. Part (b) follows by finding $C$, $W$ and $F$ to match the equations (9). Details are given in the Appendix B.1. □

**Linear and Non-Linear Parametrizations:** Proposition 2.1 shows that linear RNNs with sufficient width can represent the same input-output mappings as any linear convolution system. The difference between the models is in the parameterizations. The output of the convolutional model is linear in the parameters $\theta_{\text{conv}}$ whereas it is non-linear in $\theta_{\text{RNN}}$. As we will see below, the non-linear parameterization of the RNN results in certain implicit biases.

## 3. NTKs of Linear RNNs and Scaled Convolutional Models

### 3.1. Neural Tangent Kernel Background

To state our first set of results, we briefly review the neural tangent kernel (NTK) theory from (Jacot et al., 2018; Arora et al., 2019). The main definitions and results we need are as follows: Consider the problem of learning a (possibly non-linear) model of the form

$$\widehat{y} = f(x, \theta), \tag{11}$$

where $x \in \mathbb{R}^{m_x}$ is an input, $f(\cdot)$ is a model function differentiable with respect to parameters $\theta$, and $\widehat{y}$ is some prediction of an output $y \in \mathbb{R}^{m_y}$. The problem is to learn the parameters $\theta$ from training data $\{x^i, y^i\}_{i=1}^N$. For sequence problems, we use the convention that each $x_i$ and $y_i$ represent one entire input-output sequence pair. Hence, the dimensions will be $m_x = n_xT$ and $m_y = n_yT$.

Now, given the training data $\{x^i, y^i\}_{i=1}^N$ and an initial parameter estimate $\theta^0$, the *neural tangent kernel (NTK) model* is the linear model

$$\widehat{y} = f^{\text{lin}}(x, \alpha) := f(x, \theta^0) + \sum_{j=1}^{N} K(x^j, x)\alpha_j, \tag{12}$$

where $K(x, x')$ is the so-called NTK,

$$[K(x, x')]_{ij} := \left\langle \frac{\partial f_i(x, \theta^0)}{\partial \theta}, \frac{\partial f_j(x', \theta^0)}{\partial \theta} \right\rangle, \tag{13}$$

where $f_i$ denotes the $i$-th output dimension and $\alpha$ is a vector of dual coefficients,

$$\alpha = (\alpha_1, \ldots, \alpha_N), \quad \alpha_j \in \mathbb{R}^{m_x}. \tag{14}$$

Note that $K(x, x')$ depends implicitly on $\theta^0$. Also, for a fixed initial condition, $\theta^0$, the model (12) is linear in the parameters $\alpha$, and hence potentially easier to analyze than the original non-linear model (11). The key result in NTKs is that, for certain wide neural networks with random initializations, (full-batch) gradient descent training of the non-linear and linear models are *asymptotically identical*. For example, the results in (Lee et al., 2019) and (Alemohammad et al., 2020) provide the following proposition:

**Proposition 3.1.** *Suppose that $f_n(x, \theta)$ is a sequence of recurrent neural networks with $n$ hidden states and non-linear activation function $\sigma(\cdot)$. Let $\{(x^i, y^i)\}_{i=1}^N$ be some fixed training data contained in a compact set. Let $\widehat{\theta}_n^0$ denote a random initial condition generated as (4) and let $\widehat{\theta}_n^\ell$ denote the parameter estimate after $\ell$ steps of (full-batch) gradient descent with some learning rate $\eta$. Let $K_n(x, x')$ denote the NTK of the RNN and $f_n^{\text{lin}}(x, \alpha)$ denote the corresponding linear NTK model (12). Let $\widehat{\alpha}_n^\ell$ denote the parameter estimate obtained with GD with the same learning rate. We further assume that the non-linear activation $\sigma$ satisfies*

$$|\sigma(0)|, \ \ \|\sigma'\|_\infty, \ \ \sup_{x \neq x'} |\sigma(x) - \sigma(x')|/|x - x'| < \infty.$$

*Then, for all $x$ and $x'$,*

$$\lim_n K_n(x, x') = K(x, x') \text{ a.s.} \tag{15}$$

*for some deterministic positive semi-definite matrix $K(x, x')$. Moreover, if $\lambda_{\min}(K) > 0$, then for sufficiently small learning rate $\eta$ and any new sample $x$,*

$$\lim_{n \to \infty} \sup_{\ell \geq 0} \|f_n(x, \widehat{\theta}_n^\ell) - f_n^{\text{lin}}(x, \widehat{\alpha}_n^\ell)\| = 0, \tag{16}$$

*where the convergence is in probability.*

The consequence of this result is that the behavior of certain infinitely-wide neural networks on new samples $x$ is identical to the behavior of the linearized network around its initialization. This essentially means that as $n \to \infty$, the learning dynamics for the original and the linearized networks match during training.

### 3.2. NTK for the Convolutional Model

Having defined the NTK, we first compute the NTK of the scaled convolutional model (8).

**Theorem 3.2.** *Fix a time period $T$ and consider the convolutional model (8) for a given set of scale factors $\rho =$*

$(\rho_0, \ldots, \rho_{T-1})$. *Then, for any initial condition, and any two input sequences* $\mathbf{x}$ *and* $\mathbf{x}'$*, the NTK for this model is,*

$$K(\mathbf{x}, \mathbf{x}') = \mathcal{T}(\mathbf{x})^\mathsf{T} D(\rho) \mathcal{T}(\mathbf{x}') \otimes I_{n_y}, \tag{17}$$

*where* $\otimes$ *denotes the tensor product and* $\mathcal{T}(\mathbf{x})$ *is the Toeplitz matrix,*

$$\mathcal{T}(\mathbf{x}) := \begin{bmatrix} x_0 & x_1 & \cdots & x_{T-1} \\ 0 & x_0 & \cdots & x_{T-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_0 \end{bmatrix} \in \mathbb{R}^{Tn_x \times T}. \tag{18}$$

*and* $D(\rho)$ *is the diagonal matrix,*

$$D(\rho) := \mathrm{diag}(\rho_0 I_{n_x}, \cdots, \rho_{T-1} I_{n_x}). \tag{19}$$

*Proof Sketch.* Given that $y_t = \sum_{j=0}^{t} \sqrt{\rho_j} \theta_j x_{t-j}$, we consider a perturbation in parameters $\theta$. We show that the NTK of this model can be written as an expectation over a certain Gaussian random variable. We further derive the full kernel as in (17). Details of the proof are provided in Appendix B.2. □

### 3.3. NTK for the RNN Parametrization

We now compare the NTK of the scaled convolutional model to the NTK for the RNN parameterization.

**Theorem 3.3.** *Fix a time period* $T$ *and consider the RNN model* (1) *mapping an input sequence* $\mathbf{x} = (x_0, \ldots, x_{T-1})$ *to an ouptut sequence* $\mathbf{y} = (y_0, \ldots, y_{T-1})$ *with the parameters* $(W, F, C)$*. Assume the parameters are initialized as* (4) *for some constants* $\nu_W, \nu_F, \nu_C > 0$*. In the limit as the number of hidden states* $n \to \infty$*:*

(a) *The impulse response coefficients* $L_j$ *in* (9) *converge in distribution to independent Gaussians, where the components of* $L_j$ *are i.i.d.* $\mathcal{N}(0, \nu_C \nu_F \nu_W^j)$*.*

(b) *Given any input sequences* $\mathbf{x}$ *and* $\mathbf{x}'$*, the NTK converges almost surely to the deterministic limit:*

$$K(\mathbf{x}, \mathbf{x}') = \mathcal{T}(\mathbf{x})^\mathsf{T} D(\rho) \mathcal{T}(\mathbf{x}') \otimes I_{n_y}, \tag{20}$$

*where* $\mathcal{T}(\mathbf{x})$ *and* $D(\rho)$ *are given in* (18) *and* (19) *and*

$$\rho_j = \nu_C (j \nu_F \nu_W^{j-1} + \nu_W^j) + \nu_F \nu_W^j. \tag{21}$$

*Proof Sketch.* Part (a) of the theorem is a special case of a more general lemma, Lemma C.1, that we present in the Appendix. Given (1) and (9), let $q_t$ be the impulse response coefficients from $x_t$ to $h_t$. We show that $(q_0, \ldots, q_t)$ converges to a Gaussian vector $(Q_0, \ldots, Q_t)$ where $Q_i$'s are independent zero mean. We then calculate the variances following Lemma C.1 equations.

In part (b), we consider perturbations $\Delta_W$, $\Delta_F$, and $\Delta_C$ of the parameters of the RNN $W$, $F$, and $C$. We observe that the mapping from $x_t$ to $[h_t\ , \frac{dh_t}{dt}]$ is a linear time-invariant system and its impulse response coefficients can be analyzed in the LSL using Lemma C.1. We can then calculate the NTK as in (20). Details of the proof are provided in Appendix B.3. □

Comparing Theorems 3.2 and 3.3, we see that the NTK for linear RNN is identical to that of an scaled convolution model when the scalings are chosen as (21). From Proposition 3.1, we see that, in the wide limit regime where $n \to \infty$, gradient descent training of the linear RNN with the nonlinear parametrization $\theta_{\mathrm{RNN}} = (W, F, C)$ in (3) is identical to the training of the convolutional model (5) where the linear parameters $L_j$ are initialized as i.i.d. Gaussians and then trained with certain scaling factors (21).

Moreover, the scaling factors have a geometric decay. Recall from Section 2 that, for stability of the linear RNN we require that $\nu_W < 1$. When $\nu_W < 1$ and $j > 1$, the scaling factors (21) can be bounded as

$$\rho_j \le \rho_{\max} \nu_W^{j-1}, \quad \rho_{\max} := \nu_C (T\nu_F + 1) + \nu_F. \tag{22}$$

Consequently, the scale factors decay geometrically with $\nu_W^{j-1}$. This implies that, in the training of the scaled convolutional model, the coefficients with higher delay $j$ will be given lower weight.

## 4. Implicit Bias of Linear RNNs

An importance consequence of the geometric decay of the scaling factors $\rho_j$ in (22) is the *implicit bias* of GD training of linear RNNs towards networks with short-term memory. To state this precisely, fix input and output training data $(\mathbf{x}^i, \mathbf{y}^i)$, $i = 1, \ldots, N$. For each $n$, consider the RNN model (2) with $n$ hidden states and parameters $\theta_{\mathrm{RNN}}$ in (3). Assume the parameters are initialized as $\theta^0_{\mathrm{RNN}} = (W^0, F^0, C^0)$ in (4) for some $\nu_W, \nu_F, \nu_C > 0$. Let

$$\theta^\ell_{\mathrm{RNN}} = (W^\ell, F^\ell, C^\ell),$$

denote the parameter after $\ell$ steps of (full batch) gradient descent with some learning rate $\eta$. Let $L^\ell_{\mathrm{RNN}}$ be the resulting impulse response coefficients (9),

$$L^\ell_{\mathrm{RNN},j} = n^{-(j+1)/2} C^\ell (W^\ell)^j F^\ell, \quad j = 0, \ldots, T-1. \tag{23}$$

We then have the following bound.

**Theorem 4.1.** *Under the above assumptions, the norm of the impulse response coefficients of the RNN at the initial iteration* $\ell = 0$ *are given by*

$$\lim_{n \to \infty} \mathbb{E} \| L^0_{\mathrm{RNN},j} \|_F^2 = n_x n_y \nu_C \nu_F \nu_W^j. \tag{24}$$

*Also, there exists constants $B_1$ and $B_2$ such that if the learning rate satisfies $\eta < B_1$, then for all iterations $\ell$,*

$$\limsup_{n\to\infty} \|L^{\ell}_{\mathrm{RNN},j} - L^{0}_{\mathrm{RNN},j}\|_F \leq B_2 \rho_j \eta \ell, \qquad (25)$$

*where the convergence is in probability. Moreover, the constants $B_1$ and $B_2$ can be selected independent of $\nu_W$.*

*Proof Sketch.* We first bound the initial impulse response from the result of Theorem 3.3. Next, we use Theorems 3.2 and 3.3 to construct a scaled convolutional model that has the same NTK and intial conditions as the RNN. Finally, we analyze the convolutional model to obtain the bound in (25). For details of the proof see Appendix B.4. □

To understand the significance of the theorem, observe that in the convolutional model (5), $L_j$ relates the input time samples $x_{t-j}$ to the output $y_t$. The coefficient $L_j$ thus describes the influence of the inputs samples on the output samples $j$ time steps later. Combining (24), (25), and (22), we see that these coefficients decay as

$$L^{\ell}_{\mathrm{RNN},j} = O(\nu_W^{j/2} + \ell \nu_W^{j}).$$

Also, as discussed in Section 2, we need $\nu_W < 1$ for stability. Hence, the magnitude of these coefficients decay geometrically with $\nu_W^{j-1}$. Therefore, for a fixed number of training steps, the effect of the input on the output at a lag of $j$ would be exponentially small in $j$. In this sense, training linear RNNs with the non-linear parameterization $\theta_{\mathrm{RNN}} = (W, F, C)$ is implicitly biased to short memory.

It is useful to compare the performance of an unscaled convolutional model with the linear RNN. The convolutional model can fit any linear time-invariant system with an arbitrary delay. We have seen in Proposition 2.1 that, in principle, the linear RNN can also fit any such system with a sufficient number of hidden states. However, the above theorem shows that unless the number of gradient steps grows exponentially with the desired delay, the parameterization of the linear RNN will strongly bias the solutions to systems with short memory. This restriction will create bias error on systems that have long-term memory. On the other hand, due to the implicit constraint of the linear RNN, the parameterization will reduce the variance error.

## 5. Numerical Experiments

We validate our theoretical results on a number of synthetic and real data experiments.

### 5.1. Synthetic data

In section 3, we showed that the NTK for a linear RNN given in (1) with parameters $\theta_{\mathrm{RNN}}$ in (3) is equivalent to the NTK for its convolutional parameterization with parameters $\theta_{\mathrm{conv}}$. In order to validate this, we compared the training dynamics of a linear RNN, eq (1), with a large number of hidden states, $n$, and a scaled convolutional model (8) with scale coefficients $\rho_j$ defined in (21).

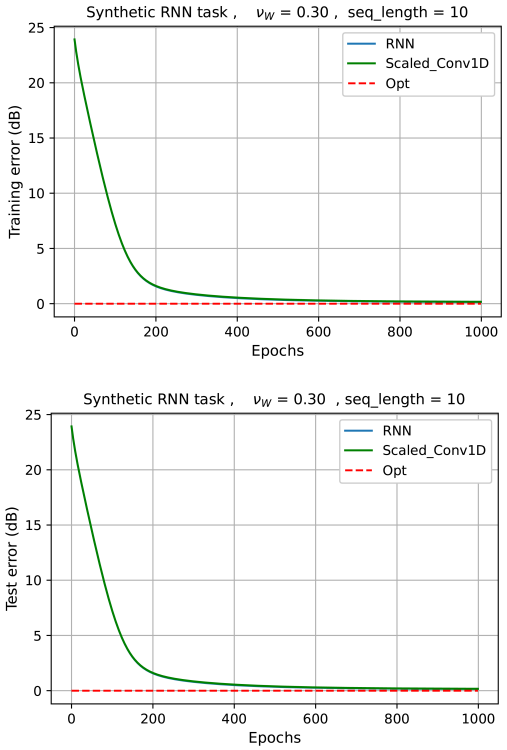



*Figure 1.* Dynamics of an RNN and its equivalent scaled Conv-1D in learning a synthetic task. the data is generated from a synthetic RNN with $n_x = n_y = 1$ and $nh = 4$. Noise is added to the output with SNR= 20 dB The sequence length T=10. Training and test samples are $N_{\mathrm{tr}} = N_{\mathrm{ts}} = 50$. Full batch gradient descent is used with $lr = 10^{-4}$. The dynamics of these models perfectly match.

We generated data from a synthetic (teacher) RNN with random parameters. For the data generation system we used a linear RNN with 4 hidden units and $n_x = n_y = 1$. Matrices $W$, $F$, and $C$ are generated as i.i.d random Gaussian with $\nu_W = 0.3$ and $\nu_F = \nu_C = 1$. We added noise to the output of this system such that the signal-to-noise ratio (SNR) is 20 dB. We generated 50 training sequences and 50 test sequences in total and each sequence has $T = 10$ time steps.

Given the training and test data, we train (i) a (student) linear RNN with $n = 1000$ hidden units, $\nu_W = 0.3$, and $\nu_F = \nu_C = 1$; and (ii) a 1D-convolutional model with scale coefficients $\rho_j$ calculated in (21) using $\nu_W = 0.3$, $\nu_F = \nu_C = 1$. We used mean-squared error as the loss function for both models and applied full-batch gradient descent with learning rate $lr = 10^{-4}$. Fig. 1 shows the

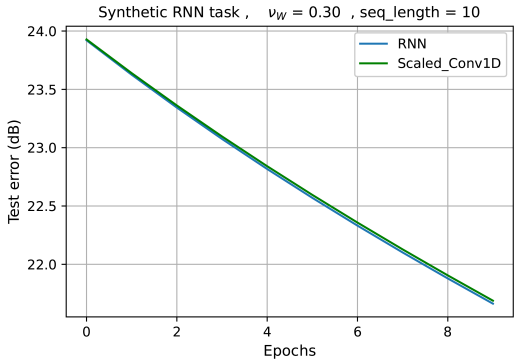



*Figure 2.* The first 10 epochs in Fig. 1. Note that to be theoretically accurate, we need $lr \to 0$ to be in the kernel regime.

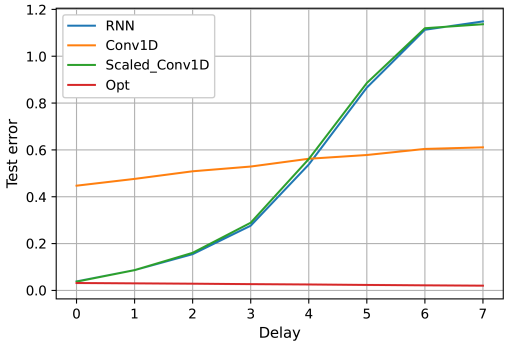



*Figure 3.* Test performance with respect to delay. For this task we have $n_h = 1000$, $N_{\text{tr}} = N_{\text{ts}} = 10$, $n_x = 15$, $n_y = 1$, and $T = 20$. The delay is added manually by shifting i.e. $y_t = x_{t-delay}$ and the output SNR = 20 dB.

identical dynamics of training for both models. Fig. 2 shows a zoomed-in version of the dynamics for training error. We see, as the theory predicts, the RNN and scaled convolution 1D model have an identical performance.

To evaluate the performance of these models for a task with long-term dependencies, we created a dataset where we manually added different delay steps $\tau$ to the output of a true linear RNN system i.e. $y_t = x_{t-\tau}$. We have chosen longer ($T = 20$) true sequences for this task. We then learned this data using the aforementioned linear RNN and scaled 1D convolutional models. We also trained an unscaled 1D convolutional model with this data to compare performances. With unscaled convolutional model, we exactly learn the impulse response coefficients $L_j$ defined in (5) during training.

Fig. 3 shows the test error with respect to delay steps for all three models. Observe that the performance of the scaled convolutional and the linear RNN models match during training. Due to the bias of these models against the delay, the test error increases as we increase the delay steps in our system. On the other hand, the performance of the unscaled convolutional model stays almost the same with increasing delay, slightly changing at larger delays as there is less data to track. We thus see the effect of implicit bias: When the true system does not have high delay, the implicit bias of the RNN and scaled convolutional model against delay helps. As the true delay increases, the implicit bias causes bias error not present in the unscaled convolutional model.

Our theoretical results hold in the asymptotic regime where the number of hidden units of the RNN goes to infinity. To test the applicability of our results to real-world RNNs that have a finite number of hidden units, we run an experiment where we change the number of hidden units and test how closely the RNN training follows the kernel training, i.e. the equivalent scaled convolutional model. In this experiment, the true RNN that generates the synthetic data has 20 hidden units and we train different RNNs with 10, 40, and 200 hidden units to learn the synthetic data. The results are shown in Figure 4. For each $n_h$, the RNN (solid lines) along with the equivalent scaled convolutional model (dashed line) are trained. Our theory claims that when the learning rate of gradient descent goes to zero and the number of hidden units goes to infinity, the training error of these two models should be exactly the same over the course of the optimization, i.e. the solid curves and dashed curves should match. We see that it is indeed the case. As we increase $n_h$ the two curves become closer and for $n_h = 200$ they almost perfectly match. This suggests that our results should be applicable in practice to RNNs of moderates size with more that 200 hidden units.

### 5.2. Real data

We also validated our theory using spikes rate data from the macaque primary somatosensory cortex (S1) (Benjamin et al., 2018). Somatosensory cortex is a part of the brain responsible for receiving sensations of touch, pain, etc from the entire body. The data is recorded during a two-dimensional reaching task. In this task, a macaque was rewarded for positioning a cursor on a series of randomly generated targets on the screen using a handle. The data is from a single recording of 51 minutes and includes 52 neurons. The mean and median firing rates are 9.3 and 6.3 spikes/sec.Similar to the previous experiments, we also trained an unscaled 1D convolutional model with this data and compared the performances with the linear RNN and the scaled convolutional models.

We compared the performances on two sets of experiments. We first used only the 4.5 minutes of the total recorded data. The purpose of this experiment is to compare the performances in limited data circumstances. With this limited data, we expect the scaled convolutional model (and thus

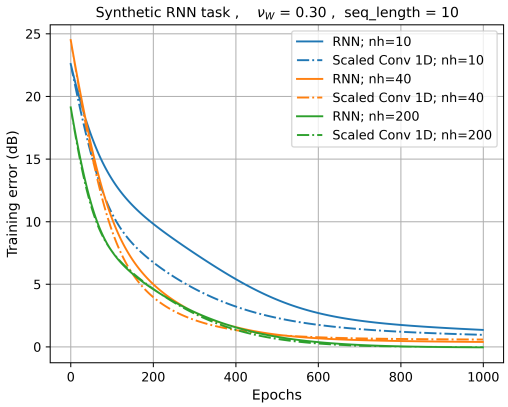



*Figure 4.* Training error over the course of training for RNNs with different number of hidden units. Solid curves show the error of RNN whereas the dashed curves show the error of equivalent scaled convolutional model for each epoch. Our theoretical results show that in the asymptotic regime of $n_h \to \infty$ with inifinitesimal learning rate, the solid curves and dashed curves should exactly match. Here, we see that as we increase $n_h$ the two curves get closer and even for $n_h = 200$ our theoretical prediction almost perfectly matches what we observe in practice.

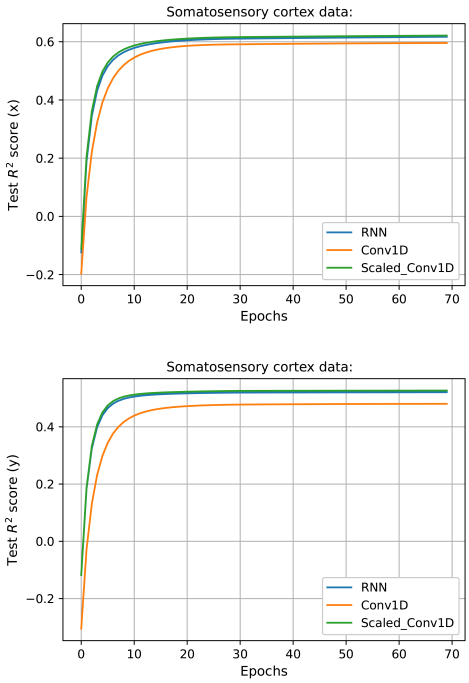
the RNN) to perform better than the unscaled model due to the implicit bias of the towards short-term memory and the fact that the effective number of parameters is smaller in the scaled model which leads to a smaller variance. In the second experiment, we trained our models using all the available data ($\approx 51$ mins). In this case, the scaled model (and the RNN) performs worse because of the increased bias error. In our experiments setup, the linear RNN has $n = 1000$ hidden states and the sequence length $T = 15$. Also, $\nu_W = 0.3$ and $\nu_F = \nu_C = 1$.

Fig. 5 shows the $R^2$ scores for x and y directions of all three models for this task. Observe that, the dynamics of the linear RNN and scaled convolutional model are identical during training using either the entire recording or a part of it. For the case of limited data, as discussed earlier, we observe the implicit bias of the RNN and scaled convolutional model in the figures (a). This bias leads to better performance of these two models compared to the unscaled model. Using the total available data, the unscaled convolutional model performs better because of the increased bias error in the other two models (figures in (b)). Table 1 shows the test $R^2$-score of the final trained models for all three cases.

## 6. Conclusion

In this work, we focus on the special class of linear RNNs and observe a functional equivalence between linear RNNs and 1D convolutional models. Using the kernel regime framework, we show that the training of a linear RNN is

(a) Limited data




(b) Entire data

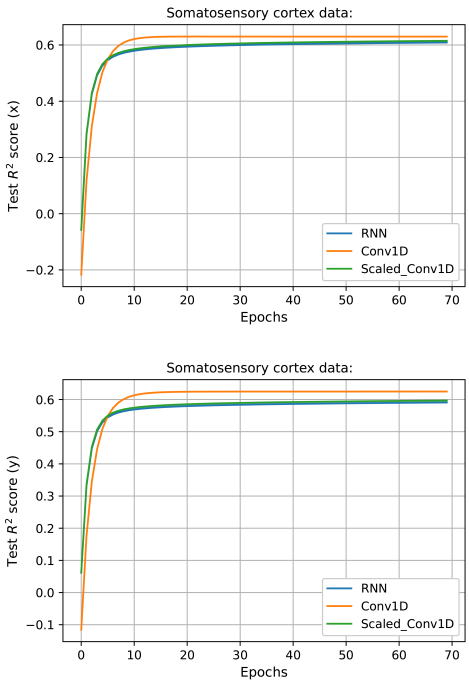



*Figure 5.* $R^2$ score for the two dimensional reaching task described in section 5.2 . The data is recorded from the primary somatosensory cortex of macaques. (a) Limited data: The models are trained on 4.5 minutes of recorded data. (b) Entire data: the whole recording ($\approx 51$ mins) is used to compare the performances. For both cases we used mini-batch (batch size = 128) gradient descent with lr = $10^{-4}$.

| | RNN | Scaled Conv-1D | Conv-1D |
|---|---|---|---|
| $R_x^2$ | 0.6462 | 0.6442 | 0.6565 |
| $R_y^2$ | 0.5911 | 0.5860 | 0.6027 |
| $R_x^2$ (limited data) | 0.6043 | 0.6046 | 0.5856 |
| $R_y^2$ (limited data) | 0.4257 | 0.4234 | 0.3918 |

*Table 1.* $R^2$-score on test data for x and y directions in the two dimensional reaching task described in section 5.2

identical to the training of a certain scaled convolutional model. We further provide an analysis for an inductive bias in linear RNNs towards short-term memory. We show that this bias is driven by the variances of RNN parameters at random initialization. Our theory is validated by both synthetic and real data experiments.

## Acknowledgements

The work of M. Emami, M. Sahraee-Ardakan, P. Pandit, and A. K. Fletcher was supported in part by the National Science Foundation under Grants 1254204 and 1738286, and the Office of Naval Research under Grant N00014-15-1-2677. S. Rangan was supported in part by the National Science Foundation under Grants 1116589, 1302336, and 1547332, NIST, the industrial affiliates of NYU WIRELESS, and the SRC.

## References

Alemohammad, S., Wang, Z., Balestriero, R., and Baraniuk, R. The recurrent neural tangent kernel. *arXiv preprint arXiv:2006.10246*, 2020.

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018.

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pp. 1120–1128, 2016.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., and Wang, R. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pp. 8139–8148, 2019.

Bai, Z. D. and Yin, Y. Q. Limiting behavior of the norm of products of random matrices and two problems of geman-hwang. *Probability theory and related fields*, 73 (4):555–569, 1986.

Barbier, J., Krzakala, F., Macris, N., Miolane, L., and Zdeborová, L. Optimal errors and phase transitions in high-dimensional generalized linear models. *Proc. National Academy of Sciences*, 116(12):5451–5460, March 2019. ISSN 1091-6490. doi: 10.1073/pnas.1802705116. URL http://dx.doi.org/10.1073/pnas.1802705116.

Bayati, M. and Montanari, A. The dynamics of message passing on dense graphs, with applications to compressed sensing. *IEEE Transactions on Information Theory*, 57 (2):764–785, 2011.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proc. National Academy of Sciences*, 116(32):15849–15854, 2019.

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Benjamin, A. S., Fernandes, H. L., Tomlinson, T., Ramkumar, P., VerSteeg, C., Chowdhury, R. H., Miller, L. E., and Kording, K. P. Modern machine learning as a benchmark for fitting neural responses. *Frontiers in computational neuroscience*, 12:56, 2018.

Chen, M., Pennington, J., and Schoenholz, S. S. Dynamical isometry and a mean field theory of rnns: Gating enables signal propagation in recurrent neural networks. *arXiv preprint arXiv:1806.05394*, 2018.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder–decoder approaches. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014. doi: 10.3115/v1/w14-4012. URL http://dx.doi.org/10.3115/v1/W14-4012.

Daniely, A. Sgd learns the conjugate kernel class of the network. In *Advances in Neural Information Processing Systems*, pp. 2422–2430, 2017.

Daniely, A., Frostig, R., and Singer, Y. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances In Neural Information Processing Systems*, pp. 2253–2261, 2016.

Dou, X. and Liang, T. Training neural networks as learning data-adaptive kernels: Provable representation and approximation benefits. *Journal of the American Statistical Association*, pp. 1–14, 2020.

Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*, 2018a.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018b.

E, W., Ma, C., and Wu, L. A comparative analysis of optimization and generalization properties of two-layer neural network and random feature models under gradient descent dynamics. *Science China Mathematics*, Jan 2020. ISSN 1869-1862. doi: 10.1007/s11425-019-1628-5. URL http://dx.doi.org/10.1007/s11425-019-1628-5.

Emami, M., Ardakan, M. S., Rangan, S., and Fletcher, A. K. Input-output equivalence of unitary and contractive rnns. In *Advances in Neural Information Processing Systems*, pp. 15342–15352, 2019.

Emami, M., Sahraee-Ardakan, M., Pandit, P., Rangan, S., and Fletcher, A. K. Generalization error of generalized linear models in high dimensions. *arXiv preprint arXiv:2005.00180*, 2020.

Garriga-Alonso, A., Rasmussen, C. E., and Aitchison, L. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bklfsi0cKm.

Gerbelot, C., Abbara, A., and Krzakala, F. Asymptotic errors for convex penalized linear regression beyond gaussian matrices. *arXiv preprint arXiv:2002.04372*, 2020.

Givens, C. R., Shortt, R. M., et al. A class of wasserstein metrics for probability distributions. *The Michigan Mathematical Journal*, 31(2):231–240, 1984.

Hastie, T., Montanari, A., Rosset, S., and Tibshirani, R. J. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pp. 8571–8580, 2018.

Jing, L., Shen, Y., Dubcek, T., Peurifoy, J., Skirlo, S., LeCun, Y., Tegmark, M., and Soljačić, M. Tunable efficient unitary neural networks (eunn) and their application to rnns. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1733–1741. JMLR. org, 2017.

Kailath, T. *Linear systems*, volume 156. Prentice-Hall Englewood Cliffs, NJ, 1980.

Katayama, T. *Subspace methods for system identification*. Springer Science & Business Media, 2006.

Lee, J., Sohl-dickstein, J., Pennington, J., Novak, R., Schoenholz, S., and Bahri, Y. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1EA-M-0Z.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*, pp. 8570–8581, 2019.

Lennart, L. System identification: theory for the user. *PTR Prentice Hall, Upper Saddle River, NJ*, pp. 1–14, 1999.

Li, Y. and Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pp. 8157–8166, 2018.

Ljung, L. System identification. *Wiley encyclopedia of electrical and electronics engineering*, pp. 1–19, 1999.

Ljung, L. and Glad, T. *Modeling of Dynamic Systems*. Prentice-Hall information and system sciences series. PTR Prentice Hall, 1994. ISBN 9780135970973. URL https://books.google.com/books?id=zO9qQgAACAAJ.

Ma, C., Wu, L., et al. A comparative analysis of the optimization and generalization property of two-layer neural network and random feature models under gradient descent dynamics. *arXiv preprint arXiv:1904.04326*, 2019.

Matthews, A. G. d. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

Mei, S., Montanari, A., and Nguyen, P.-M. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33): E7665–E7671, 2018.

Montanari, A., Ruan, F., Sohn, Y., and Yan, J. The generalization error of max-margin linear classifiers: High-dimensional asymptotics in the overparametrized regime. *arXiv preprint arXiv:1911.01544*, 2019.

Neal, R. M. *Bayesian Learning for Neural Networks*. Springer New York, 1996. doi: 10.1007/978-1-4612-0745-0. URL https://doi.org/10.1007%2F978-1-4612-0745-0.

Novak, R., Xiao, L., Bahri, Y., Lee, J., Yang, G., Abolafia, D. A., Pennington, J., and Sohl-dickstein, J.

Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1g30j0qF7.

Pintelon, R. and Schoukens, J. *System identification: a frequency domain approach*. John Wiley & Sons, 2012.

Rangan, S., Schniter, P., and Fletcher, A. K. Vector approximate message passing. *IEEE Transactions on Information Theory*, 65(10):6664–6684, 2019.

Rotskoff, G. M. and Vanden-Eijnden, E. Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *arXiv preprint arXiv:1805.00915*, 2018.

Sirignano, J. and Spiliopoulos, K. Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*, 130(3):1820–1852, 2020.

Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Deylon, B., Glorennec, P.-Y., Hjalmarsson, H., and Juditsky, A. *Nonlinear black-box modeling in system identification: a unified overview*. Linköping University, 1995.

Söderström, T. and Stoica, P. *System identification*. Prentice-Hall International, 1989.

Viberg, M. Subspace-based methods for the identification of linear time-invariant systems. *Automatica*, 31(12): 1835–1851, 1995.

Villani, C. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Wei, C., Lee, J. D., Liu, Q., and Ma, T. Regularization matters: Generalization and optimization of neural nets vs their induced kernel. In *Advances in Neural Information Processing Systems*, pp. 9709–9721, 2019.

Wisdom, S., Powers, T., Hershey, J., Le Roux, J., and Atlas, L. Full-capacity unitary recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 4880–4888, 2016.

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

Yang, G. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 9951–9960, 2019b.

Zou, D., Cao, Y., Zhou, D., and Gu, Q. Stochastic gradient descent optimizes over-parameterized deep relu networks. *arXiv preprint arXiv:1811.08888*, 2018.