Stochastic gradient updates yield deep equilibrium kernels

Anonymous authors
Paper under double-blind review

Abstract

Implicit deep learning allows one to compute with implicitly defined features, for example features that solve optimisation problems. We consider the problem of computing with implicitly defined features in a kernel regime. We call such a kernel a deep equilibrium kernel (DEK). Specialising on a stochastic gradient descent (SGD) update rule applied to features in a latent variable model, we find an exact corresponding deterministic update rule for the DEK in a high dimensional limit. This derived update rule resembles previously introduced infinitely wide neural network kernels. To perform our analysis, we describe an alternative parameterisation of the link function of exponential families, a result that may be of independent interest. This new parameterisation allows us to draw new connections between a statistician's inverse link function and a machine learner's activation function. We describe an interesting property of SGD in this high dimensional limit: even though individual iterates are random vectors, inner products of any two iterates are deterministic, and can converge to a unique fixed point as the number of iterates increases. We find that the DEK empirically outperforms related neural network kernels on a series of benchmarks.

1 Kernel methods, deep learning and implicit deep learning

Kernel methods are a classical paradigm for analysing representational capacity, bias, generalisation performance and practical algorithms for nonparametric prediction (Schölkopf et al., 2002). Many classical nonparametric models can be seen as extensions of parametric models (Saunders, 1998; Rasmussen & Williams, 2006, § 2.2) that allow for increased representational capacity while retaining some statistical model-based properties. Examples of model-based qualities may include the smoothness, stationarity or periodicity of the predictor (Duvenaud, 2014, § 2) or the statistical interpretation of the learning procedure (Sollich, 2002; Rasmussen & Williams,§ 3), which may be understood by examining the kernel or the loss function (Banerjee et al., 2005, Theorem 4).

Despite early successes of kernel methods, when data is plentiful and/or modelling is hard, over-parameterised and under-regularised deep learning is now seen as the dominant paradigm for practical nonparametric-style prediction (OpenAI et al., 2019; Adiwardana et al., 2020; Rombach et al., 2022). Unlike parametric and classical nonparametric approaches, the architecture and loss functions of many explicit neural networks are driven purely from the perspective of representational power or predictive performance (either empirical (Vaswani et al., 2017) or mathematical (Raghu et al., 2017)) rather than model-based qualities.

A fruitful direction is to analyse deep learning predictors through the reductionist lens of kernel methods through sufficiently well-behaved neural networks in certain large parameter count regimes (Neal, 1995). However, to the best of our knowledge, no current theory describes architectural properties of neural networks in the kernel regime such as choice of activation function, depth and skip connections, in terms of model-based properties. It is desirable to motivate predictive deep learning architectures from a more fundamental, statistical model-based perspective (Rudin, 2019; Efron, 2020) in a kernel regime.

Implicit neural networks are an emerging approach to model-based deep learning, where the layers are defined to implicitly satisfy the solution to a given problem. For example, deep declarative networks

(DDNs) (Gould et al., 2021) solve optimisation problems, deep equilibrium models (DEQs) (Bai et al., 2019) solve fixed point (algebraic) problems and neural ODEs solve differential equations (Chen et al., 2018). Such problems are usually computed numerically via the (approximate) fixed point of an iterative procedure. This leads to the view that implicit layers are themselves a composition of infinitely many functions. Owing to the complexity of deep learning algorithms, theory falls short of explaining the empirically demonstrated successes of both implicit and explicit models. To the best of our knowledge, no general notion of an implicit kernel is currently described in the literature.

1.1 Our contribution: an implicit kernel and an update rule in kernel space

Updates in feature space Solutions to optimisation, fixed point or differential equation problems are in practice most often obtained via a possibly stochastic iterative update procedure. Let $X_1 \in \mathbb{X} \subseteq \mathbb{R}^l$ be an input to the problem and $\psi_{X_1} \in \boldsymbol{\psi} \subseteq \mathbb{R}^m$ be the solution to the problem. Note that ψ_{X_1} is the evaluation of an implicit function of X_1. Let $\psi_{X_1}^{(t)}$ be a representation of the solution obtained at iterate t. That is, there exists some possibly stochastic function $g^{(t)}(\,\cdot\,; X) : \boldsymbol{\psi} \to \boldsymbol{\psi}$ such that

$$\psi_{X_1}^{(t+1)} = g^{(t)}\big(\psi_{X_1}^{(t)}; X_1\big), \qquad \text{and} \qquad \psi_{X_2}^{(t+1)} = g^{(t)}\big(\psi_{X_2}^{(t)}; X_2\big). \tag{1}$$

We call $\psi_{X_1}^{(t)}$ and $\psi_{X_2}^{(t)}$ features and $g^{(t)}$ the update rule in feature space. We emphasise that we consider the problem where features are updated, not weight parameters as in some other settings.

Deep equilibrium kernels We find helpful the notion of an implicitly defined kernel, which we call a deep equilibrium kernel (DEK). This allows us to draw parallels between infinitely wide implicit neural networks and implicitly defined kernel machines. We consider three kernel evaluations in terms of the implicit updates in feature space,

$$\underbrace{\overline{\Psi}_{12}^{(t+1)} \triangleq {\psi_{X_1}^{(t+1)}}^{\top} \psi_{X_2}^{(t+1)}}_{\text{finite feature DEK (ffDEK)}}, \qquad \underbrace{\Psi_{12}^{(t+1)} \triangleq \operatorname*{plim}_{m\to\infty} \overline{\Psi}_{12}^{(t+1)}}_{\text{DEK}}, \qquad \text{and} \qquad \underbrace{\Psi_{12} \triangleq \lim_{\tau\to\infty} \Psi_{12}^{(\tau)}}_{\text{limiting DEK } (\ell\text{DEK})}, \tag{2}$$

where defined, where plim denotes convergence in probability. Note the order of the limits. We will similarly write Ψ_{11} and Ψ_{22} to represent evaluations of such DEKs at (X_1, X_1) and (X_2, X_2) respectively. We write $\mathsf{\Psi}^{(t+1)}$ for the corresponding 2×2 PSD matrices containing $\Psi_{11}^{(t+1)}$, $\Psi_{12}^{(t+1)}$ and $\Psi_{22}^{(t+1)}$ (and likewise for the ffDEK and ℓDEK). The dimensionality of features are allowed to grow to infinity, but only after taking the inner product — resulting in a scalar value for examination — avoiding the necessity of describing, analysing and building algorithms involving infinite dimensional feature spaces.

Updates in kernel space Let $\mathbb{S}_+^2 = \{\mathsf{K} \in \mathbb{R}^{2\times 2} \mid \mathsf{K} \succeq 0\}$ denote the space of 2×2 PSD matrices. Our central questions are as follows. Firstly, as $m \to \infty$, does there exist a G such that updates may be performed on 2×2 PSD DEK matrices instead of in $2m$-dimensional feature space? Secondly, can we write a closed form for G? Finally, does this iteration converge? That is, does there exist a closed-form update rule in kernel space $\mathsf{G}(\,\cdot\,; X_1, X_2) : \mathbb{S}_+^2 \to \mathbb{S}_+^2$ such that

$$\mathsf{\Psi}^{(t+1)} = \mathsf{G}(\mathsf{\Psi}^{(t)}; X_1, X_2)\,? \tag{3}$$

$$\text{And does} \qquad \mathsf{\Psi} = \mathsf{G}(\mathsf{\Psi}) = \lim_{\tau\to\infty} \underbrace{\mathsf{G}\circ\ldots\circ\mathsf{G}}_{\tau \text{ compositions}} (\mathsf{\Psi}^{(0)}; X_1, X_2)\,? \tag{4}$$

For convenience, we notationally decompose G into components via a function G satisfying for each $ij \in \{11, 22, 12\}$

$$\mathsf{G}(\mathsf{\Phi}; X_1, X_2) = \begin{pmatrix} G_{11} & G_{12} \\ G_{12} & G_{22} \end{pmatrix}, \qquad \text{where}$$
$$G_{ij} \triangleq G(\Phi_{ii}, \Phi_{jj}, \Phi_{ij}; X_i, X_j) \triangleq \Big(\mathsf{G}(\mathsf{\Phi}; X_1, X_2)\Big)_{ij}. \tag{5}$$

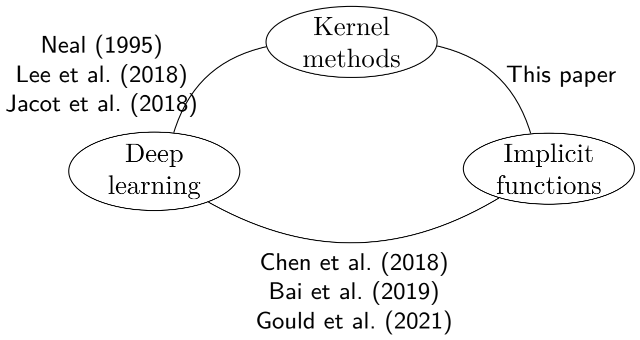

Figure 1: We establish links between kernel methods and implicit functions to design a neural network kernel with corresponding statistical assumptions.

Contributions We study an important special case of a DEK where we answer (3) and (4) positively, one in which the features are iteratively updated using SGD applied to a latent variable model. The model is *over-parameterised* (the number of parameters grows much faster than the amount of data), but shallow (the *motivation* for the model more closely resembles exponential family PCA than a deep neural network). The objective (17) to which we apply SGD is an *under-regularised* variant of an expected negative log posterior. Our main result (stated precisely in § 3) is a constructive proof of the existence of G.

Surprisingly, despite the feature space of the DEK having seemingly no direct relation with deep learning predictors, deep learning structures emerge as part of our analysis. Our DEK may be understood as an infinitely wide DEQ whose iterates are computed with stochastic approximation rather than as a deterministic fixed point iteration. Further, the kernel iterates of our DEK resemble previously introduced NNKs and NTKs and may be computed with deterministic fixed point solvers.

Theorem 4 and Corollary 5 (informal). *When the features $\psi^{(t)}_{X_1}$ and $\psi^{(t)}_{X_2}$ are point estimates obtained by SGD applied to objective* (17)*, we construct a deterministic update rule for the DEK,* G*. Repeated applications of* G *converge to a fixed point, the ℓDEK. In other words, for a specific continuous latent variable model, under mild assumptions,* (3) *and* (4) *hold for the kernel update rule* G *on 2×2 PSD matrices.*

We further quantify the degree to which the ℓDEK is an invariant of SGD when treated as an ffDEK (Theorem 8).

2 Background

Our analysis requires combining fixed point theory (optimisation), the exponential family (statistics), and neural network kernels (machine learning). We briefly describe elements of these topics here.

2.1 Fixed points and infinite compositions

Let $f : \mathbb{F} \to \mathbb{F}$ for some set $\mathbb{F}$ equipped with a norm $\| \cdot \|$ and norm-induced metric. A fixed point of f is any $Z^* \in \mathbb{F}$ satisfying $f(Z^*) = Z^*$. Banach's fixed point theorem (BFPT) gives sufficient conditions for the existence and uniqueness of such a fixed point.

Theorem 1 (BFPT)**.** *Let $(\mathbb{F}, \| \cdot \|)$ be a non-empty complete normed space. A mapping $f : \mathbb{F} \to \mathbb{F}$ is called a contraction mapping if there exists some $q \in [0, 1)$ such that $\|f(Z) - f(Z')\| \leq q\|Z - Z'\|$ for every $Z, Z' \in \mathbb{F}$. Every contraction mapping f admits a unique fixed point $Z^* \in \mathbb{F}$. Furthermore, for any initial element $Z^{(1)} \in \mathbb{F}$, the sequence $Z^{(t+1)} = f(Z^{(t)})$ for $t \geq 1$ converges to Z^* as $t \to \infty$.*

It is worth noting that BFPT not only provides a mathematical condition for well-posedness, but also describes an algorithm for approximating fixed points of contraction mappings. We call this algorithm the *naive fixed point solver*, which simply involves applying a τ-fold composition of f to some starting value $Z^{(1)}$, with a linear rate of convergence immediate from the definition of contraction mapping, i.e. $\|Z^* - Z^{(t+1)}\| \leq \frac{q^t}{1-q}\|Z^{(2)} - Z^{(1)}\|$. Other solvers for fixed point problems are available, many of which are approximate Newton methods for root finding (Kelley, 1995).

Deep equilibrium models (DEQs) (Bai et al., 2019) are neural network predictors constructed of parameterised layers that output the solution to fixed point equations $f_{\mathsf{U}}(Z^*) = Z^*$. These layers draw upon earlier works on recurrent backpropagation (Pineda, 1987; Almeida, 1990), leveraging the modern machinery of deep learning architectures, optimisers and heuristics. The unsupervised learning problem for a DEQ is

$$\underbrace{\min_{\mathsf{U}} \sum_{i=1}^{N} L\big(X_i, Z_i^*, \mathsf{U}\big)}_{\text{Empirical risk minimisation for parameters } \mathsf{U}} \qquad \text{subject to} \qquad \underbrace{Z_i^* = f_{\mathsf{U}}(Z_i^*, X_i),}_{\text{Fixed point solution for DEQ predictions } Z_i^*}$$

where L is some loss function, U is a parameter object, and $\{X_i\}_{i=1}^N$ is a collection of input examples. (A supervised setting might also involve a set of output examples). Derivatives $\frac{\partial Z_i^*}{\partial \mathsf{U}}$ of outputs of these layers with respect to their parameters U can be computed without backpropagating through the iterates of the fixed point solver using the implicit function theorem (Bai et al., 2019). This allows first-order stochastic gradient methods that are popular with explicit deep learning architectures to be applied to DEQs.

In general, it is not guaranteed that a function necessarily admits a unique fixed point; various works discuss dealing with multiple fixed points or ensuring or encouraging that exactly or at least one fixed point exists (Winston & Kolter, 2020; Revay et al., 2020; El Ghaoui et al., 2021). Interestingly, if a single DEQ layer involves finding the fixed point of a contraction mapping, by Theorem 1 the output computed by a DEQ has the interpretation of an infinitely deep neural network with shared parameters in each layer. More generally, mappings computed by the naive fixed point solver have interpretations as very deep neural networks with shared parameters in each layer. Since zeros of the gradient of sufficiently well behaved objectives are stationary points of the objectives, DEQ layers share a connection with optimisation-based implicit layers (Gould et al., 2021), as explored in various works (Revay et al., 2020; Xie et al., 2021; Tsuchida et al., 2022; Riccio et al., 2022; Tsuchida & Ong, 2022). Our current investigation concerns a connection more specific than optimisation, since it considers the special case of applying SGD.

2.2 Exponential families

Exponential families The feature mappings that we use to build our kernel are estimates obtained using SGD applied to certain variants of exponential family likelihoods and Gaussian priors. We now define *minimal and regular exponential families in canonical form.* Let h be a probability density (mass) function supported on data space $\mathbb{Y} \subseteq \mathbb{R}$. Let $T : \mathbb{Y} \to \mathbb{R}$ be a function called the *sufficient statistic.* Given some *canonical parameter* η belonging to an open set $\mathbb{H} \subseteq \mathbb{R}$, we may construct a probability density (mass) function by normalising the nonnegative function $h(\cdot)\exp\big(T(\cdot)\eta\big)$. The normalising constant is called the partition function, and its strictly convex and infinitely differentiable logarithm A is called the *log partition function* (Wainwright et al., 2008, Proposition 3.1). We write

$$p(y \mid \eta) = h(y)\exp\big(T(y)\eta - A(\eta)\big), \quad A(\eta) = \log \int_{\mathbb{Y}} h(y)\exp\big(T(y)\eta\big)\, dy$$

for the evaluation of a probability density (mass) function of an exponential family. The log partition function A acts as a cumulant generating function for the conditional distribution of the sufficient statistic T. In particular the expected value of the sufficient statistic (often called the *expectation parameter* (Nielsen & Garcia, 2009)) is the gradient of the log partition function A. That is,

$$\mathbb{E}\big[T(y) \mid \eta\big] = A'(\eta). \tag{6}$$

We consider factorised exponential families in the following sense. Let $y_1, \dots y_d$ be distributed according to the same exponential family and define data vector $Y = (y_1, \dots, y_d)^\top$ and canonical parameter vector $H = (\eta_1, \dots, \eta_d)$. Then the joint distribution of data Y conditioned on canonical parameters H is the product of the individual elements

$$p(Y \mid H) = \prod_{i=1}^{d} p(y_i \mid \eta_i) = \Big(\prod_{r=1}^{d} h(y_r) \Big) \exp \big(T(Y)^\top H - A(H)^\top \mathbf{1} \big), \tag{7}$$

where we write $T(Y) = \big(T(y_1), \dots, T(y_d)\big)^\top$, $A(H) = \big(A(\eta_1), \dots, A(\eta_d)\big)^\top$ and $\mathbf{1} = (1, \dots, 1)^\top$.

Link functions and canonical link functions Exponential families are used in generalised linear models (GLMs) (McCullagh & Nelder, 1989). In GLMs, the conditional expectation (6) of an exponential family is set to be the result of applying an (invertible) *inverse link function* s^{-1} to the result of a linear transformation of features $\phi \in \mathbb{R}^m$ (classically called parameters). That is, for some linear basis $\mathsf{V} \in \mathbb{R}^{d \times m}$ (classically called covariates),

$$A'(H) = \mathbb{E}\big[T(Y) \mid H\big] = s^{-1}(\mathsf{V}\phi). \tag{8}$$

The conditional expectation is then mapped to the canonical parameter H through $H = (A')^{-1} \circ s^{-1}(\mathsf{V}\phi)$, noting that A' is invertible because A is strictly convex. In the case where s^{-1} is chosen to be A', $s \equiv (A')^{-1}$ is called the *canonical link function*, and we observe from (8) that the canonical parameter and conditional expectation satisfy

$$H = \mathsf{V}\phi, \qquad \mathbb{E}[T(Y) \mid \mathsf{V}\phi] = A'(\mathsf{V}\phi) = s^{-1}(\mathsf{V}\phi). \tag{9}$$

There are two main and sometimes conflicting reasons why one might be interested in using a non-canonical link function. The first is computational; if the link function were canonical, for some distributions such as Gamma or exponential one would need a constrained optimisation method over the open set $\mathbb{H}$ instead of $\mathbb{R}$. If s^{-1} were allowed to be non-canonical — that is, we are free to choose s^{-1} different from A' — we could map the conditional expectation to the appropriate constraint set and unconstrained optimisation procedures could be applied. In a Bayesian context, sampling from the posterior over ϕ can be made easier by convenient choices of s. For example, the probit model admits an efficient Gibb's sampler for the posterior (Albert & Chib, 1993). The second, and arguably more important consideration is modelling; we might have reason to suspect that the conditional expectation is constrained. For example, if the observations should have a positive expectation, the power family of link functions might be used (McCullagh & Nelder, 1989, equation 2.9a). Alternative link functions can lead to exploiting particular properties of interest; for example, Wiemann et al. (2021) use the softplus function for positive conditional expectations to exploit its identity-like behaviour at large positive values. In weighing up the possibly conflicting aims of computational convenience and modelling suitability, we highlight the view of Efron & Hastie (2021, page 68); while classical exponential families and link functions may lead to closed-form expressions, modern computer technology allows us more flexible models.

Point estimation When using a canonical inverse link function $s \equiv A'$, the negative logarithm of the likelihood (7) is strictly convex in H, since A is strictly convex and linear functions are convex. If H is chosen to be $H = \mathsf{V}\phi$, this translates to convexity in ϕ, and maximum likelihood estimates can be computed using first or (more typically, in a classical setting) second order optimisation methods. When s^{-1} is not a canonical inverse link function, convexity does not necessarily hold. Nevertheless, local estimates are practically useful, so pre-implemented link functions and the option to implement custom link functions is available in a number of software frameworks including `R` (R Core Team, 2021, `family`) and `Stata` (Hardin & Hilbe, 2018, `glm`).

2.3 Kernels arising from neural networks

Our main result describes the DEK update rule as a composite function involving evaluations of kernels of a particular form. These are kernels that are constructed from neural network models. In this section, we describe such kernels.

The neural network kernel was first investigated as the covariance function of a certain neural network with random parameters and a single hidden fully connected layer (Neal, 1995). Under mild conditions, as the width of the hidden layer goes to infinity the neural network converges to a Gaussian process. This analysis has since been extended to handle multiple layers (Matthews et al., 2018; Lee et al., 2018), other layer types including convolutional layers (Mairal et al., 2014; Garriga-Alonso et al., 2018; Novak et al., 2019; Yang, 2019a;b), and training under gradient flow via the neural tangent kernel (NTK) (Jacot et al., 2018) . Since our motivation is better described in terms of inner products of the features, we favour the view of the neural network kernel as an inner product in an infinitely wide hidden layer rather than a covariance function of a Gaussian process. We note that connections between Bayesian Gaussian processes and kernel methods exist (Kanagawa et al., 2018) and apply to some but not all infinitely wide neural networks.

Neural network kernel, single hidden layer Let $\mathsf{W}^{(1)} \in \mathbb{R}^{d \times n}$ be the weights of a fully connected hidden layer with activation function ζ defined over the reals. Suppose each entry of $\mathsf{W}^{(1)}$ is i.i.d. with distribution $\mathcal{N}(0,1)$[1]. Given an input feature $\phi_1 \in \mathbb{R}^{n \times 1}$ (we take the convention that vectors are column vectors), the signal in the hidden layer is $h^{(1)} \triangleq \zeta(\mathsf{W}^{(1)}\phi_1)$[2]. Here and throughout the paper the symbol $\triangleq$ means that the object on the left hand side is defined to be the expression on the right hand side. By a strong law of large numbers, a suitably normalised inner product in the hidden layer converges almost surely as $d \to \infty$ to an expectation,

$$\frac{1}{d} {h_1^{(1)}}^\top h_2^{(1)} = \frac{1}{d}\zeta(\mathsf{W}^{(1)}\phi_1)^\top \zeta(\mathsf{W}^{(1)}\phi_2) \overset{a.s.}{\to} \mathbb{E}_W\big[\zeta(W^\top\phi_1)\zeta(W^\top\phi_2)\big],$$

assuming the right hand side is finite, since the inner product is a sum of i.i.d. random variables. Here $W^\top \in \mathbb{R}^{1 \times m}$ is a vector with i.i.d. entries drawn from $\mathcal{N}(0,1)$. We define

$$k_\zeta(\phi_1,\phi_2) \triangleq \mathbb{E}_W\big[\zeta(W^\top\phi_1)\zeta(W^\top\phi_2)\big], \tag{10}$$

and call k_ζ a single hidden layer neural network kernel (NNK) with activation function ζ. The PSD kernel k_ζ uniquely defines an RKHS by the Moore–Aronszajn theorem. Closed-form expressions of k_ζ for different ζ are available (Williams, 1997; Le Roux & Bengio, 2007; Cho & Saul, 2009; Tsuchida et al., 2018; Pearce et al., 2019; Tsuchida, 2020; Meronen et al., 2020; Tsuchida et al., 2021; Han et al., 2022).

Define $(\chi_1,\chi_2)^\top \triangleq \big(W^\top\phi_1, W^\top\phi_2\big)^\top$, which is a zero mean bivariate Gaussian with a covariance matrix $\boldsymbol{\Sigma}^{(1)}$. Note that $k_\zeta(\phi_1,\phi_2)$ depends on the input features ϕ_1 and ϕ_2 only through the covariance matrix $\boldsymbol{\Sigma}^{(1)}$. It is helpful to explicate this dependence structure through a special notation. We have that (10) is equal to

$$\kappa_\zeta\big(\Sigma_{11}^{(1)},\Sigma_{22}^{(1)},\Sigma_{12}^{(1)}\big) \triangleq k_\zeta(\phi_1,\phi_2) = \mathbb{E}_{(\chi_1,\chi_2)^\top\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}^{(1)})}\big[\zeta\big(\chi_1\big)\zeta\big(\chi_2\big)\big], \quad \boldsymbol{\Sigma}^{(1)} \triangleq \begin{pmatrix}\phi_1^\top\phi_1 & \phi_1^\top\phi_2\\ \phi_2^\top\phi_1 & \phi_2^\top\phi_2\end{pmatrix}. \tag{11}$$

With an abuse of terminology, we refer to both k_ζ and κ_ζ as PSD single hidden layer NNKs. For a more detailed description of the NNK, see Appendix C.

Neural network kernel, τ hidden layers One may compose (11) multiple times by applying a sequence of kernels to a 3-dimensional state represented by a 2×2 PSD matrix $\boldsymbol{\Sigma}^{(t)}$, in place of the infinitely wide signals. This 3-dimensional state represents the two squared norms and inner product in each hidden layer. For $t = 1,\dots,\tau$ and $ij \in \{11,22,12\}$,

$$\Sigma_{ij}^{(t+1)} \triangleq \mathbb{E}_{(\chi_i,\chi_j)^\top\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}^{(t)})}\big[\zeta\big(\chi_i\big)\zeta\big(\chi_j\big)\big] = \kappa_\zeta\big(\Sigma_{ii}^{(t)},\Sigma_{jj}^{(t)},\Sigma_{ij}^{(t)}\big), \tag{12}$$

where $\Sigma_{ij}^{(t)}$ denotes the ijth element of $\boldsymbol{\Sigma}^{(t)}$. This iteration appears in deep infinitely wide NNKs (Matthews et al., 2018; Lee et al., 2018). We will refer to this kernel as the τ layer NNK. Evaluations $\Sigma_{12}^{(\tau+1)}$ of the PSD kernel are determined entirely by the activation function ζ, and uniquely define an RKHS.

[1]The effect of non-unit weight variance may be obtained by scaling all inputs ϕ_1 by a hyperparameter. Similarly, arbitrary covariance structures inside rows of $\mathsf{W}^{(1)}$ can be reflected as linear transformations of all inputs ϕ_1.

[2]The effect of zero mean Gaussian biases may be obtained by augmenting inputs with an additional coordinate. The magnitude of this coordinate is equivalent to the quotient of the standard deviation of the weights to the biases.

Neural tangent kernel, τ hidden layers This kernel describes the limiting behaviour of randomly initialised neural networks that are trained under gradient flow (Jacot et al., 2018). The kernel iterations are similar to (12), but also contain components involving the derivative $\dot{\zeta}$ of ζ. Let $\odot$ denote elementwise product. In addition to the iteration (12), let $\Theta^{(1)} = \Sigma^{(1)}$ and define

$$\Theta^{(t+1)} \triangleq \Theta^{(t)} \odot \dot{\Sigma}^{(t+1)} + \Sigma^{(t+1)}, \quad \text{where} \tag{13}$$

$$\dot{\Sigma}_{ij}^{(t+1)} \triangleq \mathbb{E}_{(\chi_i,\chi_j)^\top \sim \mathcal{N}(\mathbf{0},\Sigma^{(t)})}\big[\dot{\zeta}(\chi_i)\dot{\zeta}(\chi_j)\big] = \kappa_{\dot{\zeta}}\big(\Sigma_{ii}^{(t)}, \Sigma_{jj}^{(t)}, \Sigma_{ij}^{(t)}\big)$$

to obtain the evaluation of the PSD NTK in the last iteration $\Theta_{12}^{(\tau+1)}$. Once again, the kernel is determined entirely by the ζ, and uniquely defines an RKHS.

2.4 Notation

Numerical subscripts are used to extract (groups of) indices of a vector or matrix. Parenthesised superscripts indicate a layer or iteration of a naive fixed point solver, both of which turn out to be the same in our constructions. We index objects by iteration by superscript (t), so that $\psi^{(t)}$ represents a feature in the tth iteration. We use ϕ and Φ for arbitrary vectors and inner products that are not necessarily obtained by iterations of SGD. We will use ψ and Ψ for feature mappings and inner products of feature mappings that are obtained by iterations of SGD.

We assume that we are given access to a dataset $\mathsf{X} \in \mathbb{R}^{N \times l}$ of N examples of datapoints $X_i \in \mathbb{X} \subseteq \mathbb{R}^l$. We denote by X_1 and X_2 any two elements of this dataset.

There are two types of function signatures we associate with PSD kernels. The first is for a usual PSD kernel $k : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$, so that an evaluation is written $k(X, X')$ for any two $X, X' \in \mathbb{X}$. We call this form a k-form kernel. The second is for a PSD kernel whose evaluation depends on $\phi_1, \phi_2 \in \boldsymbol{\psi}$ only through evaluations $\Phi_{11} = \langle \phi_1, \phi_1 \rangle, \Phi_{12} = \langle \phi_1, \phi_2 \rangle, \Phi_{22} = \langle \phi_2, \phi_2 \rangle$ of some suitably defined inner product $\langle \cdot, \cdot \rangle : \boldsymbol{\psi} \times \boldsymbol{\psi} \to \boldsymbol{\psi}$. We represent such a kernel through $\kappa : \boldsymbol{\psi}^3 \to \mathbb{R}$ with evaluations $\kappa\big(\langle \phi_1, \phi_1 \rangle, \langle \phi_2, \phi_2 \rangle, \langle \phi_1, \phi_2 \rangle\big)$. An example of this second form is the NNK (11). We call this form a κ-form kernel.

Our notation is summarised in Table 3 in Appendix A.

3 Main results

Our results are most clearly described in terms of an alternative parameterisation of exponential families and link functions, which are perhaps of independent interest. We first describe this alternative parameterisation in § 3.1, before moving onto the setup for our main analysis in § 3.2. We then in § 3.3 provide a special case (Corollary 3) of our main and most general result (Theorem 4) in § 3.4. Finally, quantification of error between the DEK and ffDEK is described in § 3.5. We give examples of our resulting updates in Appendix G.2.

3.1 An alternative view of link functions in exponential families

Instead of computing via the conditional expectation resulting from the application of an inverse link function (8), we follow Tsuchida & Ong (2022) and learn the canonical parameter via a nonlinearity $H = R(\mathsf{V}\phi)$, for some once-differentiable $R : \mathbb{R} \to \mathbb{H}$ called the *canonical nonlinearity*. This means that the conditional likelihood (7) is now

$$p(Y \mid \mathsf{V}, \phi) = \prod_{i=1}^{d} p(y_i \mid \eta_i) = \Big(\prod_{r=1}^{d} h(y_r)\Big) \exp\big(T(Y)^\top R(\mathsf{V}\phi) - A\big(R(\mathsf{V}\phi)\big)^\top \mathbf{1}\big). \tag{14}$$

Such a parameterisation is rich enough to recover the (non-canonical) inverse link function view of the statistician (see Proposition 2). It can therefore be considered to be a change of notation, placing emphasis on the canonical nonlinearity R instead of the inverse link function s^{-1}. In our setting, one

Exponential family	$A(\eta)$	$s^{-1}(a)$	$R(a)$	$\rho(a)$	$\sigma(a)$
Gaussian	$\eta^2/2$	$s^{-1}(a)$	$s^{-1}(a)$	$(s^{-1})'(a)$	$s^{-1}(a)\,(s^{-1})'(a)$
Gaussian	$\eta^2/2$	a	a	1	a
Gaussian	$\eta^2/2$	$\operatorname{erf}(a/\sqrt{2})$	$\operatorname{erf}(a/\sqrt{2})$	$2p(a)$	$\operatorname{erf}(a/\sqrt{2})2p(a)$
Gaussian	$\eta^2/2$	$\operatorname{ReLU}(a)$	$\operatorname{ReLU}(a)$	$\mathrm{u}(a)$	$\operatorname{ReLU}(a)$
Poisson	$\exp(\eta)$	$s^{-1}(a)$	$\log s^{-1}(a)$	$\frac{(s^{-1})'(a)}{s^{-1}(a)}$	$(s^{-1})'(a)$
Poisson	$\exp(\eta)$	$\exp(a)$	a	1	$\exp(a)$
Poisson	$\exp(\eta)$	$\log(1+\exp a)$	$\log\log(1+\exp a)$	$\frac{\exp(a)}{(1+\exp a)\log(1+\exp a)}$	$\exp(a)/\big(1+\exp(a)\big)$
Bernoulli	$\log(1+\exp(\eta))$	$s^{-1}(a)$		$\frac{(s^{-1})'(a)}{s^{-1}(a)\big(1-s^{-1}(a)\big)}$	$\frac{(s^{-1})'(a)}{1-s^{-1}(a)}$
Bernoulli	$\log(1+\exp(\eta))$	$\exp(a)/\big(1+\exp(a)\big)$	a	1	$\exp(a)/\big(1+\exp(a)\big)$
Bernoulli	$\log(1+\exp(\eta))$	$P(a)$	$\log\left(\frac{P(a)}{1-P(a)}\right)$	$\frac{p(a)}{P(a)P(-a)}$	$\frac{p(a)}{P(-a)}$

Table 1: These examples are obtained by plugging the desired log partition function A and inverse link function s^{-1} into expressions (23), (24) and (25). Canonical link functions are shown in blue. General inverse link function settings are shown in red. Here P and p respectively denote the cdf and pdf of the univariate standard Gaussian and erf denotes the error function.

advantage of such a notation is that it avoids more complicated function compositions involving inverses and derivatives. For example, instead of writing $A \circ (A')^{-1} \circ s^{-1}(\mathsf{V}\phi)$ we may write $A \circ R(\mathsf{V}\phi)$. The value of these simple compositions become more evident in Proposition 2.

Nonlinearities and activation functions The derivatives of the log likelihood (14) (a *score function*) play a central role in numerical procedures associated with estimation. In our setting, such derivatives involve terms derived from A and R. These terms are expressed in terms of functions we call *factor activations* $\rho(a) \triangleq R'(a)$ and *chain activations* $\sigma(a) \triangleq (A \circ R)'(a)$. The following identities show how one may map between choices of (A, s) and choices of (A, R), and additionally how these induce activation functions σ and ρ which appear in gradient-based optimisers and our later derivations. Note that we may choose the inverse link function s^{-1} to be non-canonical (not A').

Proposition 2. *Consider a regular and minimal exponential family with log partition function $A : \mathbb{H} \to \mathbb{R}$. Suppose the conditional expectation belongs to a set $\mathbb{A}'$, that is, $A'(\eta) \in \mathbb{A}'$ for all canonical parameters $\eta \in \mathbb{H}$. Let $s^{-1} : \mathbb{B} \to \mathbb{A}'$ be an inverse link function, for some $\mathbb{B} \subseteq \mathbb{R}$. That is, for every $\eta \in \mathbb{H}$ there exists some $a \in \mathbb{B}$ such that $A'(\eta) = s^{-1}(a)$. Then equivalently, $\eta = R(a)$, where $R : \mathbb{B} \to \mathbb{H}$ is defined by $R(a) \triangleq \big((A')^{-1} \circ s^{-1}\big)(a)$. Furthermore,*

$$\rho(a) \triangleq R'(a) = \frac{(s^{-1})'(a)}{A'' \circ (A')^{-1} \circ s^{-1}(a)} \qquad \textit{and} \qquad \sigma(a) \triangleq (A \circ R)'(a) = \frac{s^{-1}(a)\,(s^{-1})'(a)}{A'' \circ (A')^{-1} \circ s^{-1}(a)}.$$

The proof is given in Appendix B. In practice we will take $\mathbb{B} = \mathbb{R}$. We observe that R is the identity if and only if s is a canonical link function (which is to say that $s^{-1}(a) = A'(a)$). For the special case of a Gaussian with known variance, A' is the identity and R is the inverse link function s^{-1}. Further cases are listed in Table 1.

Nonlinear parameterisation framed in terms of R instead of s^{-1} are often used (McCullagh & Nelder, 1989, Chapter 11.4 and references therein), but their general relationship to s^{-1} does not appear to be discussed. As our setting is equivalent to using an arbitrary link function, we inherit the motivation of using a non-identity R from the motivation for using a non-canonical link function. We also inherit the usual difficulties in estimation and sample complexity due to using not necessarily canonical link functions.

Recall that the choice of (A, s) should be informed by both modelling and numerical convenience (sampling, optimisation) considerations. Motivated by neural network kernels, we find a different set of (A, s) pairs convenient to work with compared with the generalised linear model setting. Convenience here translates to being able to compute certain Gaussian integrals of the form (11) in closed form. For example, we find it easy to work with a Gaussian with non-negative conditional expectation, parameterised by $A(\eta) = \eta^2/2$ and $s^{-1}(a) = \operatorname{ReLU}(a)$, where ReLU is the popular rectified linear unit (Efron

& Hastie, 2021, page 362). Another convenient setting is a Gaussian likelihood and probit inverse link function (in contrast with the often seen Bernoulli and probit inverse link function). Our theory holds for general (A, s) pairs, but its practical efficiency is contingent upon the existence of efficient numerical routines for computing the integral (11). Such numerical routines in the absence of closed-forms are available in other works (Zandieh et al., 2021; Han et al., 2022), but we do not study their application here.

3.2 Setup

Stochastic gradient descent We apply SGD (Wright & Recht, 2022, Chapter 5) to a minimisation objective $\mathcal{L}(\phi; X) = \mathbb{E}_{\mathsf{V}} L\big(\phi; X, \mathsf{V}\big)$ with decision variable ϕ, input X and random object V. Given two inputs X_1 and X_2, the $t+1$th iterates are

$$\psi_{X_1}^{(t+1)} = \psi_{X_1}^{(t)} - \alpha^{(t)} \frac{\partial}{\partial \psi_{X_1}^{(t)}} L\big(\psi_{X_1}^{(t)}; X_1, \mathsf{V}^{(t)}\big) \quad \text{and} \quad \psi_{X_2}^{(t+1)} = \psi_{X_2}^{(t)} - \alpha^{(t)} \frac{\partial}{\partial \psi_{X_2}^{(t)}} L\big(\psi_{X_2}^{(t)}; X_2, \mathsf{V}^{(t)}\big) \tag{15}$$

with initial features $\psi_{X_1}^{(0)}$ and $\psi_{X_2}^{(0)}$, a sequence $\{\alpha^{(t)}\}_t$ of step sizes, and a sequence of $\{\mathsf{V}^{(t)}\}_t$ of iid samples of V. We use the features $\psi_{X_1}^{(t)}$ and $\psi_{X_2}^{(t)}$ in (15) to define the ffDEK, DEK and ℓDEK via (2). We stress that we are updating features, not weight parameters.

Continuous latent variable model We work with a data generating process which is a slight nonlinear generalisation (Tsuchida & Ong, 2022) of exponential family PCA (Collins et al., 2001), allowing for nonlinear R (or equivalently, non-canonical link functions as in Proposition 2). This model describes data Y as being drawn from an exponential family distribution with a canonical parameter that is a function of a latent ϕ.

More concretely, let $X \in \mathbb{X} \subseteq \mathbb{R}^l$ be an input and suppose that data $Y = \Gamma(X)$ follows a factorised exponential family (7) for some realisation of a random mapping $\Gamma : \mathbb{X} \to \mathbb{Y}^d$. Let $R : \mathbb{R} \to \mathbb{H}$ be a once-differentiable function. Choose the canonical parameter $H = R(\mathsf{V}\phi)$ to be the composition of R and a linear transformation V of a latent input variable $\phi \in \Psi = \mathbb{R}^m$. Place an i.i.d. $\mathcal{N}(0, 1)$ prior over each entry of $\mathsf{V} \in \mathbb{R}^{d \times m}$, independent of Γ. Place an i.i.d. $\mathcal{N}(0, \lambda^{-1}/m)$ prior over ϕ. This results in a pre-nonlinearity parameter $\mathsf{V}\phi$ having components with variance which stays in d and m. For some constant C not depending on ϕ, we have

$$-\log p\big(\phi \mid \Gamma(X), \mathsf{V}\big) = -\Bigg(\underbrace{\log p\big(\Gamma(X) \mid R(\mathsf{V}\phi)\big)}_{\text{Log likelihood}} - \underbrace{m\frac{\lambda}{2}\|\phi\|^2}_{\text{-Log prior}} \Bigg) + C.$$

As discussed in Proposition 2, the derivative of the negative log-posterior with log-partition function A, which appears in our later optimisation procedure, induces two functions $\rho(a) \triangleq R'(a)$ and $\sigma(a) \triangleq (A \circ R)'(a)$ which we call factor activations and chain activations respectively.

Objective function The expected negative log posterior

$$\overline{\mathcal{L}}(\phi; X) \triangleq \mathbb{E}_{\mathsf{V}} \overline{L}\big(\phi; X, \mathsf{V}\big), \quad \text{where} \quad \overline{L}\big(\phi; X, \mathsf{V}\big) \triangleq \frac{1}{d}\Big(-\log p\big(\Gamma(X) \mid R(\mathsf{V}\phi)\big) + m\frac{\lambda}{2}\|\phi\|^2 \Big), \tag{16}$$

is a commonly used minimisation objective to find point estimates of ϕ. See Appendix H.1 for a discussion on this objective. The division by d is introduced to account for the natural numerical scaling of the likelihood term, which is a sum of d parts. Following recent deep learning trends, we consider an *over-parameterised* and *under-regularised* variant

$$\mathcal{L}(\phi; X) \triangleq \mathbb{E}_{\mathsf{V}} L\big(\phi; X, \mathsf{V}\big), \quad \text{where} \quad L\big(\phi; X, \mathsf{V}\big) \triangleq \frac{1}{d}\Big(-\log p\big(\Gamma(X) \mid R(\mathsf{V}\phi)\big) + \sqrt{md}\frac{\lambda}{2}\|\phi\|^2 \Big), \tag{17}$$

where $d < m$. This expected negative log posterior may be obtained by choosing an overly broad i.i.d. prior $\mathcal{N}\big(0, \lambda^{-1}/\sqrt{md}\big)$ over ϕ. We will take d to be a well-behaved function of m such that $d \to \infty$ as $m \to \infty$ (see Assumption 1).

Assumptions We now describe two assumptions common to both settings. Our first assumption says that the dimensionality m of the feature space should become larger much faster than the dimensionality d of the exponential family.

Assumption 1. *Consider any limit path $(m, d) \to (\infty, \infty)$ such that $\lim_{m\to\infty} \frac{d}{m} = 0$.*

Our second assumption describes how the step size α should depend on m, d, and the SGD iteration t. Recall the nomenclature "fixed" and "decreasing" as qualifiers for step-size, which describe a dependency on t (but not d). Recall that λ is the regularisation parameter.

Assumption 2 (a). $\lim_{m\to\infty} \alpha^{(t)}\lambda\sqrt{\frac{m}{d}} = 1$.

Assumption 2 (a) allows for fixed or decreasing step sizes, such as $\frac{1}{\lambda}\frac{\sqrt{d}}{\sqrt{m+r(t)}}$ for increasing but finite r.

We will find that in our setup, the DEK is a composite function involving NNK building blocks.

3.3 Error function inverse link and Gaussian likelihood to match a random mapping

We will find that the update rule G of the DEK is a composite function involving NNK building blocks. In order to clearly highlight the role of these NNK building blocks, we first present a special case of our more general Theorem 4. This provides a clear link between the statistical likelihood model and closed form expressions for the DEK update rule.

We choose an exponential family, Canonical nonlinearity and random mapping Γ. This particular setup leads to closed-form expressions for the NNKs involved in the update rule. As an activation function, we choose the error function $\operatorname{erf}(z) = 2/\sqrt{\pi}\int_0^z e^{-v^2}\,dv$ (closely related to the Probit function), and rely on a closed-form NNK derived in Williams (1997),

$$\kappa_{\operatorname{erf}(\cdot/\sqrt{2})}(\Sigma_{11}, \Sigma_{22}, \Sigma_{12}) = \frac{2}{\pi}\sin^{-1}\frac{\Sigma_{12}}{\sqrt{(1+\Sigma_{11})(1+\Sigma_{22})}}. \tag{18}$$

We show here the statistical modelling choices and their corresponding effect on the DEK update rule. In this special case, our main result (Theorem 4) implies Corollary 3, as proven in Appendix G.1. Recall from § 2.2, that the designer needs to choose a log partition function A, and a canonical nonlinearity R. We map input $X \in \mathbb{R}^l$ to data $Y \in \mathbb{R}^d$ through a random mapping $Y = \operatorname{erf}(\mathsf{W}X/\sqrt{2}) + Q$, where $\mathsf{W} \in \mathbb{R}^{d\times l}$ and $Q \in \mathbb{R}^d$ contain i.i.d. standard Gaussian elements. The distribution of Y given $\operatorname{erf}(\mathsf{W}X/\sqrt{2})$ is conditionally Gaussian, with conditional expectation $\operatorname{erf}(\mathsf{W}X/\sqrt{2})$ having elements between -1 and 1. We therefore choose a matching inverse link function, to represent the conditional expectation as a function of features ψ_X. The inverse link function is $s^{-1}(a) = \operatorname{erf}(a/\sqrt{2})$. The log partition function is $A(\eta) = \eta^2/2$ and the sufficient statistic is $T(y) = y$. Since the likelihood is Gaussian, the canonical nonlinearity R and inverse link function s^{-1} are the same, as shown in the first row of Table 1. In this particular case, the activations ρ and σ are shown in the third row of Table 1.

Corollary 3. *Suppose input X is mapped to data Y by $Y = \operatorname{erf}(\mathsf{W}X/\sqrt{2}) + Q$, where* erf *is the error function and $\mathsf{W} \in \mathbb{R}^{d\times l}$ and $Q \in \mathbb{R}^d$ contain i.i.d. standard Gaussian elements. Choose the log partition function $A(\eta) = \eta^2/2$. Choose the canonical nonlinearity $R(a) = \operatorname{erf}(a/\sqrt{2})$, or equivalently, choose the inverse link function to be $s^{-1}(a) = \operatorname{erf}(a/\sqrt{2})$. This implies that $\rho(a) = 2p(a)$ and $\sigma(a) = 2p(a)\operatorname{erf}(a/\sqrt{2})$, where p is the pdf of the standard Gaussian. Then κ_ρ and κ_σ are given by*

$$\kappa_\rho(\Phi_{11}, \Phi_{22}, \Phi_{12}) = \frac{2}{\pi\sqrt{(1+\Phi_{11})(1+\Phi_{22}) - \Phi_{12}^2}},$$
$$\kappa_\sigma(\Phi_{11}, \Phi_{22}, \Phi_{12}) = \kappa_\rho(\Phi_{11}, \Phi_{22}, \Phi_{12})\kappa_{\operatorname{erf}(\cdot/\sqrt{2})}(F_{11}, F_{22}, F_{12}), \quad \textit{where} \quad \mathsf{F} = \big(\Phi^{-1} + \mathsf{I}\big)^{-1}.$$

Let $C_{ij} = \kappa_{\operatorname{erf}(\cdot/\sqrt{2})}(X_i^\top X_i, X_j^\top X_j, X_i^\top X_j)$. Suppose Assumptions 1 and 2 (a) hold. Then applying SGD to objective (17), the update rule G (3) exists and can be decomposed into G (5) satisfying

$$G(\Phi_{ii}, \Phi_{jj}, \Phi_{ij}; X_i, X_j) = \frac{1}{\lambda^2}\Bigg(C_{ij}\kappa_\rho\big(\Phi_{ii}, \Phi_{jj}, \Phi_{ij}\big) + \kappa_\sigma\big(\Phi_{ii}, \Phi_{jj}, \Phi_{ij}\big)\Bigg).$$

Note that in this case the component G of the update rule G can be computed entirely in closed form. The 2×2 matrix $\mathsf{F} = (\Phi + \mathsf{I})^{-1}$ has a simple closed-form in terms of Φ (see Appendix G.1). Recall that the decomposition of G into G says that, by plugging (5) into (3), for each $ij \in \{11, 22, 12\}$,

$$\Psi_{ij}^{(\tau+1)} = G\big(\Psi_{ii}^{(\tau)}, \Psi_{jj}^{(\tau)}, \Psi_{ij}^{(\tau)}; X_i, X_j\big). \qquad \text{That is,} \quad \Psi^{(\tau+1)} = \mathsf{G}(\Psi^{(\tau)}; X_1, X_2).$$

3.4 General case

We now consider the general setting, allowing for arbitrary (A, s) pairs and random mappings Γ. In order to analyse this generalised setting, we require one additional definition (19) and two additional assumptions 3 and 4.

In the most general setting, the DEK includes some non-symmetric (hence not PSD and not a kernel) cross terms. Given two activations ζ_1 and ζ_2,

$$\kappa_{\zeta_1,\zeta_2}\big(\Sigma_{11}, \Sigma_{22}, \Sigma_{12}\big) \triangleq \mathbb{E}_{(\chi_1,\chi_2)^\top \sim \mathcal{N}(\mathbf{0},\mathbf{\Sigma})}\big[\zeta_1\big(\chi_1\big)\zeta_2\big(\chi_2\big)\big]. \tag{19}$$

The third assumption says that if the inner products were empirical estimates of an expectation, the resulting expectation is real valued and finite. Recall that T is the sufficient statistic of the exponential family, Γ is the random mapping from input space to data space, and $\sigma(a) = (A \circ R)'(a)$

Assumption 3. *The expectation $K(a) = \mathbb{E}_Z\Big[\Big(T\big(\Gamma(X)\big) \odot \rho\big(aZ\big) - \sigma\big(aZ\big)\Big)^2\Big]$ is finite for all $X \in \mathbb{X}$ and $a \in \mathbb{R}$, where Z is a standard Gaussian random variable.*

The fourth assumption describes the properties of the random mapping $\Gamma : \mathbb{X} \to \mathbb{Y}^d$ as $d \to \infty$. In order to understand what happens to the solutions found by SGD as d becomes large, we need the inputs which are passed through Γ to be well-behaved. It suffices that a kernel and average defined by Γ converges. We call the limiting kernel c the *explicit kernel*, which contrasts with our implicitly defined DEK. We give examples in Appendix F.

Assumption 4. *The PSD kernel c defined by $c(X_1, X_2) \triangleq \lim_{m\to\infty} \frac{1}{d} T\big(\Gamma(X_1)\big)^\top T\big(\Gamma(X_2)\big) = \mathbb{E}T\big(\Gamma(X_1)\big)^\top T\big(\Gamma(X_2)\big)$ is finite. Similarly, the mean function defined by $\mu(X_1) \triangleq \lim_{m\to\infty} \frac{1}{d} T\big(\Gamma(X_1)\big)^\top 1 = \mathbb{E}T\big(\Gamma(X_1)\big)^\top 1$ is finite.*

Our main result is a constructive proof for the existence of an update rule G, as posed in (3).

Theorem 4. *Suppose Assumptions 1, 2 (a), 3, and 4 hold. Let $C_{ij} = c(X_i, X_j)$ and $\mu_i = \mu(X_i)$ be as defined in Assumption 4. Then applying SGD to objective (17), the update rule G (3) exists and can be decomposed into G (5) satisfying*

$$\begin{aligned} &G(\Phi_{ii}, \Phi_{jj}, \Phi_{ij}; X_i, X_j) \\ &= \frac{1}{\lambda^2}\Bigg(C_{ij}\kappa_\rho\big(\Phi_{ii}, \Phi_{jj}, \Phi_{ij}\big) - \kappa_{\sigma,\rho}\big(\Phi_{ii}, \Phi_{jj}, \Phi_{ij}\big)\mu_i - \kappa_{\rho,\sigma}\big(\Phi_{ii}, \Phi_{jj}, \Phi_{ij}\big)\mu_j + \kappa_\sigma\big(\Phi_{ii}, \Phi_{jj}, \Phi_{ij}\big)\Bigg). \end{aligned}$$

Here κ_σ, κ_ρ, $\kappa_{\sigma,\rho}$ and $\kappa_{\rho,\sigma}$ are as defined by (11), (19) and Proposition 2.

The proof is given in Appendix E. Recall again that the decomposition of G into G says that, by plugging (5) into (3), for each $ij \in \{11, 22, 12\}$,

$$\Psi_{ij}^{(\tau+1)} = G\big(\Psi_{ii}^{(\tau)}, \Psi_{jj}^{(\tau)}, \Psi_{ij}^{(\tau)}; X_i, X_j\big). \qquad \text{That is,} \quad \Psi^{(\tau+1)} = \mathsf{G}(\Psi^{(\tau)}; X_1, X_2).$$

Note the cross terms involving $\kappa_{\sigma,\rho}$ and μ which were not present in the special case of Corollary 3. These cross-terms arise from random mappings Γ with an average element that is non-zero. In the case of Corollary 3, these cross-terms cancel out.

Theorem 4 implies a fixed point condition by Theorem 1, providing a positive answer for (4).

Corollary 5. *Suppose the same setting as Theorem 4. If* G *is a contraction mapping, then the DEK converges to a unique fixed point as* $t \to \infty$*. That is, for each* $ij \in \{11, 22, 12\}$,

$$\Psi_{ij} = \frac{1}{\lambda^2}\Bigg(C_{ij}\kappa_\rho\big(\Psi_{ii}, \Psi_{jj}, \Psi_{ij}\big) - \kappa_{\sigma,\rho}\big(\Psi_{ii}, \Psi_{jj}, \Psi_{ij}\big)\mu_i - \kappa_{\rho,\sigma}\big(\Psi_{ii}, \Psi_{jj}, \Psi_{ij}\big)\mu_j + \kappa_\sigma\big(\Psi_{ii}, \Psi_{jj}, \Psi_{ij}\big)\Bigg). \tag{20}$$

Whether G is a contraction can be determined by a derivative test and an identity given in Theorem 20, as we demonstrate in § G.2. Note that even if a unique fixed point does not exist (which may be the case if G is not a contraction), one may still compute with finite-t iterates of SGD via Theorem 4.

Remark 6. *We may compute iterates of SGD in the limit via the update rule for any* τ *to obtain* $\Psi_{ij}^{(\tau)}$*, which is the naive fixed point algorithm applied to* (20)*. Alternatively, we may compute the ℓDEK by solving* (20) *for each* $ij \in \{11, 22, 12\}$ *using any other fixed point solver.*

Notable special cases Some further examples arising from special choices of A, R and Γ (inducing corresponding ρ, σ, c and μ) are discussed in Appendix G.2. We find that the linear Gaussian ($A(\eta) = \eta^2/2, R(a) = a$) results in a DEK that is a scale multiple of c (Appendix G.2.1). We can recover an NNK with activation σ when A and R are allowed to be general and C and μ are set to zero (Appendix G.2.2). The setting we found useful for our experiments (§ 4) is a nonlinearly parameterised Gaussian ($A(\eta) = \eta^2/2$ and $R(a) = \mathrm{ReLU}(a)$) with a first-order arc-cosine kernel for c and a corresponding mean function μ. This setting admits a closed-form update rule for G. See Appendices G.2.4 and G.2.5 for details.

3.5 Sensitivity

Assumption 2 (a) suffices for our limiting result to hold. In order to quantify the distance between the limit and finite-dimensional kernels, we require the stronger Assumption 2 (b), which in particular requires a fixed step-size. Both variants 2 (a) and 2 (b) result in a limiting step size of 0, under Assumption 1.

Assumption 2 (b). *We have a fixed step-size* $\alpha^{(t)} = \frac{1}{\lambda}\sqrt{\frac{d}{m}}$.

Finally, in order to quantify the rate at which the infinite-dimensional, infinite-iteration DEK converges with respect to the dimension, we need the feature mapping to be well-behaved. Lipschitzness and boundedness allows concentration inequalities to be applied.

Assumption 5. *Suppose* ρ *is bounded or Lipschitz. Suppose* σ *is bounded or Lipschitz.*

We may quantify the degree to which the infinite-dimensional, infinite-iteration DEK is an invariant of SGD. We define specific values of finite dimensional ϕ and ϕ' using the infinite dimensional kernel fixed point Ψ_{11}, Ψ_{22} and Ψ_{12}. This definition will serve as a good approximation of an invariant.

Definition 7. *Define*

$$r_1 = \sqrt{\Psi_{11}}\,\big(1, 0, \dots 0\big)^\top \in \mathbb{R}^m, \qquad r_2 = \sqrt{\Psi_{22}}\,\big(\cos\omega, \sin\omega, 0, \dots 0\big)^\top \in \mathbb{R}^m,$$

where $\cos\omega = \frac{\Psi_{12}}{\sqrt{\Psi_{11}\Psi_{22}}}$*. Then* $r_1^\top r_2 = \Psi_{12}$, $r_1^\top r_1 = \Psi_{11}$*, and* $r_2^\top r_2 = \Psi_{22}$.

We bound the residual of the finite dimensional kernel evaluated at an initial guess that is the solution of the infinite (m, d) system. When this bound is small, intuitively speaking, the limiting solution is "almost" an invariant of the finite-dimensional system.

Theorem 8. *Suppose Assumptions 1, 2 (b), 3, 4 and 5 hold. Let initial guesses be* $\psi_{X_1}^{(0)} = r_1$ *and* $\psi_{X_2}^{(0)} = r_2$ *as in Definition 7. Then there exist constants* $Q_2, Q_3, c_2, c_3 > 0$ *such that for all* $\delta > 0$, $\epsilon_2 > 0$ *and* ε_2,

$$\mathbb{P}\Big(\big|\overline{\Psi}_{12}^{(1)} - \Psi_{12}\big| \le \varepsilon_1 + \varepsilon_2\Big) \ge 1 - \delta_1 - \delta_2,$$

where

$$\varepsilon_1 = \frac{K+\epsilon_2}{\lambda^2}(2\epsilon_1+\epsilon_1^2), \quad \delta_1 = 2\exp\Big(-c_2 d M_2\Big) + \exp\big(-m\delta^2/2\big) \quad \textit{and} \quad \delta_2 = 2\exp\big(-dc_3 M_3\big)$$

and $\epsilon_1 = \sqrt{\frac{d}{m}} + \delta$, $M_2 = \min\left\{\frac{\epsilon_2^2}{Q_2^2}, \frac{\epsilon_2}{Q_2}\right\}$ *and* $M_3 = \min\left\{\frac{\varepsilon_2^2}{Q_3^2}, \frac{\varepsilon_2}{Q_3}\right\}$ *and* $c_3 > 0$ *is some absolute constant.*

The proof is given in Appendix E.

4 Experiments

Recall that the ffDEK is defined for finite SGD iteration τ as an inner product of finite m-dimensional features. The DEK is defined for finite SGD iteration τ as a limit as $m \to \infty$ of an inner product of m-dimensional features. The ℓDEK is defined as a limit as SGD iteration $\tau \to \infty$ of the DEK. Although the DEK and ℓDEK are defined in terms of infinite dimensional features, evaluations of the DEK and ℓDEK are scalar values and can be used to form matrices with a finite number of rows and columns. These matrices can be used in downstream kernel algorithms to build predictive algorithms.

4.1 Measuring finite-width effects

We empirically measure the similarity of (finite-τ, finite-d) ffDEK matrices and (infinite-τ, infinite-d) ℓDEK matrices using the centered variant (Cortes et al., 2012) of kernel alignment (Cristianini et al., 2001), abbreviated CKA, as d increases. We vary d between 5 and 500 in steps of 5 and choose $m = d^{3/2}$. For control, we also measure the CKA between the (finite-τ, finite-d) ffDEK and the squared exponential kernel (SEK). See Figure 2, and Appendix J.1 for full details on the experimental setup. As expected (Theorem 4), the CKA between the DEK matrices becomes larger as d and m increase, but not between the SEK and finite DEK.

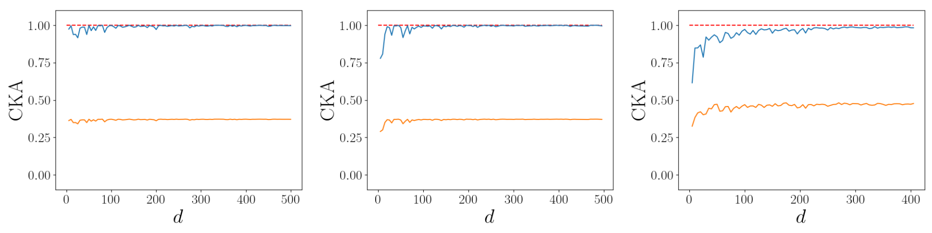

Figure 2: CKA between kernel matrices consisting of entries Ψ_{ij} and $k_d^{(t)}(X_i, X_j)$ (Blue) and squared exponential kernel for control and $k_d^{(t)}(X_i, X_j)$ (Orange) for three choices of A and R. (Left) Gaussian exponential family, $A(\eta) = \eta^2/2$ and $R(\eta) = \eta$. (Middle) Bernoulli exponential family, $A(\eta) = \log(1 + \exp \eta)$ and $R(\eta) = \eta$. (Right) Rectified Gaussian exponential family, $A(\eta) = \eta^2$ and $R(\eta) = \text{ReLU}(\eta)$.

4.2 Inference using the DEK

We use the DEK for kernel ridge regression (Saunders, 1998) (KRR) (*cf* Gaussian process regression (Rasmussen & Williams, 2006)) on a suite of benchmarks. For each dataset, we first partition the data into an $80-20$ train-test split. Using the training set, we perform 5-fold cross-validation for hyperparameter selection using the default settings of sci-kit learn's `GridSearchCV`, which performs model selection based on the coefficient of determination. The hyperparameter grid we search over is described in Table 4, Appendix J.2. We then compute the RMSE on the held out test set using all training data. We repeat this procedure for 100 different random shuffles of dataset, and find the sample average and standard deviation RMSE over the random shuffles. The results are reported in Table 2. The input X is preprocessed by subtracting the sample average and dividing by the sample standard deviation of each feature. Additionally, the target data y is mean-centered and scaled by the sample standard deviation. The reported RMSE is after conversion of y back to original units.

Data	DEK or ℓDEK	NNK	NTK	SEK	
yacht	$\mathbf{0.65 \pm 0.21}$	2.13 ± 0.57	2.75 ± 0.58	3.62 ± 0.67	0
diabetes	$\mathbf{54.51 \pm 3.29}$	$\mathbf{54.58 \pm 3.30}$	$\mathbf{55.05 \pm 3.38}$	$\mathbf{54.75 \pm 3.32}$	70
energy1	$\mathbf{1.00 \pm 0.11}$	$\mathbf{1.01 \pm 0.11}$	1.67 ± 0.14	$\mathbf{1.08 \pm 0.13}$	78
energy2	$\mathbf{1.58 \pm 0.15}$	$\mathbf{1.58 \pm 0.15}$	2.10 ± 0.18	$\mathbf{1.58 \pm 0.16}$	68
concrete	$\mathbf{4.94 \pm 0.47}$	$\mathbf{4.97 \pm 0.47}$	$\mathbf{5.05 \pm 0.48}$	5.65 ± 0.39	60
wine	$\mathbf{0.57 \pm 0.02}$	0.61 ± 0.02	$\mathbf{0.54 \pm 0.02}$	0.62 ± 0.02	1

Table 2: RMSE of KRR models (± one standard deviation over 100 random seeds). We use the DEK described in § G.2.5, which outperforms other kernels according to the sample average of the RMSE, although often the difference in performance is small compared with the standard deviation over 100 seeds. The final column is the number of times the best DEK found using `GridSearchCV` was an NNK.

Since the DEK is a strict generalisation of the NNK, we expect the DEK to strictly out-perform the NNK. We find that `GridSearchCV` sometimes picks out settings that correspond with an NNK, but often does not. The number of times `GridSearchCV` collapses the DEK to the NNK is indicated in the last column of Table 2. Our results are consistent with the previously established observation that "NNKs frequently outperform NTKs" (Lee et al., 2020). More interestingly, we find that for each dataset, the DEK performs as well or better than every other kernel, including the NNK.

5 Conclusion

We introduced the DEK, a kernel analogue of implicit neural network models. The DEK is defined as the limiting inner product between two features computed using a feature update procedure as the dimensionality of the features goes to infinity.

We considered the problem of whether a deterministic update procedure for the DEK exists (3), and whether this update rule converges (4). We focused on the special case where the features are latent variables in an exponential family PCA model (with not necessarily canonical link function) learnt using SGD. Leveraging the connection between infinitely wide explicit neural networks and kernel methods, we showed how in such a setting an explicit update rule can be computed. The update rule is a composition of functions involving NNK building blocks.

The DEK has a number of interesting properties. The DEK is able to recover instances of the NNK, and also resembles the NTK. Importantly, unlike the NNK and NTK, the deep layer structure of the DEK is motivated entirely from an optimisation perspective. The activation functions (and thus kernels) involved in the computation of the DEK can be related back to statistical modelling assumptions on the data through the exponential family. In particular, the activation functions share a connection to the log partition function and inverse link function of the exponential family. On a series of benchmarks, the DEK performs as well as or outperforms the NNK, NTK and SEK.

Our work admits several natural extensions. The matrix V which represents a linear transformation or fully connected layer may be constrained to resemble a convolutional layer, and we expect a convolutional variant of the DEK to be tractable (Novak et al., 2019). Since our construction is naturally probabilistic, the Laplace approximation about the MAP may yield a tractable means of obtaining principled uncertainty estimates for kernel methods beyond the regular Gaussian process framework. Since the DEK satisfies a fixed point equation, implicit differentiation may be used to compute derivatives of the DEK with respect to its hyperparameters, mirroring the neural network counterpart (Bai et al., 2019).

We hope that our optimisation view and deterministic kernel update rule stimulates new research in both deep learning and kernel methods.

References

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*, 2020.

James H Albert and Siddhartha Chib. Bayesian analysis of binary and polychotomous response data. *Journal of the American statistical Association*, 88(422):669–679, 1993.

Luís B. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. 1990.

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. *nternational Conference on Learning Representations*, 2020.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, Joydeep Ghosh, and John Lafferty. Clustering with bregman divergences. *Journal of machine learning research*, 6(10), 2005.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32, 2019.

Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, pp. 342–350. 2009.

Michael Collins, Sanjoy Dasgupta, and Robert E Schapire. A generalization of principal components analysis to the exponential family. *Advances in neural information processing systems*, 14, 2001.

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

Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz Kandola. On kernel-target alignment. *Advances in neural information processing systems*, 14, 2001.

Ricky Der and Daniel Lee. Beyond gaussian processes: On the distributions of infinite networks. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

Bradley Efron. Prediction, estimation, and attribution. *International Statistical Review*, 88:S28–S59, 2020.

Bradley Efron and Trevor Hastie. *Computer Age Statistical Inference, Student Edition: Algorithms, Evidence, and Data Science*, volume 6. Cambridge University Press, 2021.

Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Tsai. Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3):930–958, 2021.

Stefano Favaro, Sandra Fortini, and Stefano Peluchetti. Deep stable neural networks: large-width asymptotics and convergence rates. *arXiv preprint arXiv:2108.02316*, 2021.

Stefano Favaro, Sandra Fortini, and Stefano Peluchetti. Neural tangent kernel analysis of shallow alpha-stable relu neural networks. *arXiv preprint arXiv:2206.08065*, 2022.

Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2018.

Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):3988–4004, 2021.

Insu Han, Amir Zandieh, Jaehoon Lee, Roman Novak, Lechao Xiao, and Amin Karbasi. Fast neural kernel embeddings for general activations. *Advances in neural information processing systems*, 2022.

James W Hardin and Joseph W M Hilbe. *Generalized Linear Models and Extensions.* College Station Tex: StataCorp Press, 4th edition, 2018.

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

Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations.* Society for Industrial and Applied Mathematics, 1995. doi: 10.1137/1.9781611970944.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. 2014.

Nicolas Le Roux and Yoshua Bengio. Continuous neural networks. In *Artificial Intelligence and Statistics*, pp. 404–411, 2007.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33:15156–15172, 2020.

David JC MacKay. *Introduction to Gaussian processes.* 1998.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. *Advances in neural information processing systems*, 27, 2014.

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

P. McCullagh and J.A. Nelder. *Generalized Linear Models, Second Edition.* Monographs on Statistics and Applied Probability Series. Chapman & Hall, 1989.

Lassi Meronen, Christabella Irwanto, and Arno Solin. Stationary activations for uncertainty calibration in deep learning. *Advances in Neural Information Processing Systems*, 33:2338–2350, 2020.

Shakir Mohamed, Zoubin Ghahramani, and Katherine A Heller. Bayesian exponential family pca. *Advances in neural information processing systems*, 21, 2008.

Radford M Neal. *Bayesian learning for neural networks.* PhD thesis, University of Toronto, 1995.

Frank Nielsen and Vincent Garcia. Statistical exponential families: A digest with flash cards. *arXiv preprint arXiv:0911.4863*, 2009.

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 *The International Conference on Learning Representations*, 2019.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Jozefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Ponde de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. 2019.

Tim Pearce, Russell Tsuchida, Mohamed Zaki, Alexandra Brintrup, and Andy Neely. Expressive priors in Bayesian neural networks: Kernel combinations and periodic functions. In *Uncertainty in Artificial Intelligence*, 2019.

Stefano Peluchetti, Stefano Favaro, and Sandra Fortini. Stable behaviour of infinitely wide deep neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 1137–1146. PMLR, 2020.

Fernando Pineda. Generalization of back propagation to recurrent and higher order neural networks. In *Neural information processing systems*, 1987.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL `https://www.R-project.org/`.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pp. 2847–2854. PMLR, 2017.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006.

Max Revay, Ruigang Wang, and Ian R Manchester. Lipschitz bounded equilibrium networks. *arXiv preprint arXiv:2010.01732*, 2020.

Danilo Riccio, Matthias J Ehrhardt, and Martin Benning. Regularization of inverse problems: Deep equilibrium models versus bilevel learning. *arXiv preprint arXiv:2206.13193*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

C Saunders. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.

Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2002.

William Fleetwood Sheppard. On the application of the theory of error to cases of normal distribution and normal correlation. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, (192):140, 1899.

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.

Peter Sollich. Bayesian methods for support vector machines: Evidence and predictive class probabilities. *Machine learning*, 46(1):21–52, 2002.

Russell Tsuchida. *Results on infinitely wide multi-layer perceptrons.* PhD thesis, The University of Queensland, 2020.

Russell Tsuchida and Cheng Soon Ong. Deep equilibrium models as estimators for continuous latent variables. In *arXiv preprint*, 2022.

Russell Tsuchida, Fred Roosta, and Marcus Gallagher. Invariance of weight distributions in rectified MLPs. In *International Conference on Machine Learning*, pp. 5002–5011, 2018.

Russell Tsuchida, Tim Pearce, Chris van der Heide, Fred Roosta, and Marcus Gallagher. Avoiding kernel fixed points: Computing with elu and gelu infinite networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9967–9977, 2021.

Russell Tsuchida, Suk Yee Yong, Mohammad Ali Armin, Lars Petersson, and Cheng Soon Ong. Declarative nets that are equilibrium models. In *International Conference on Learning Representations*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Martin J. Wainwright. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint.* Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019. doi: 10.1017/9781108627771.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Paul FV Wiemann, Thomas Kneib, and Julien Hambuckers. Using the softplus function to construct alternative link functions in generalized linear models and beyond. *arXiv preprint arXiv:2111.14207*, 2021.

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

Ezra Winston and J Zico Kolter. Monotone operator equilibrium networks. *Advances in Neural Information Processing Systems*, 33:10718–10728, 2020.

Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, pp. 3635–3673. PMLR, 2020.

Stephen J. Wright and Benjamin Recht. *Optimization for Data Analysis.* Cambridge University Press, 2022. doi: 10.1017/9781009004282.

Xingyu Xie, Qiuhao Wang, Zenan Ling, Xia Li, Yisen Wang, Guangcan Liu, and Zhouchen Lin. Optimization induced equilibrium networks. *arXiv preprint arXiv:2105.13228*, 2021.

Greg Yang. Wide feedforward or recurrent neural networks of any architecture are Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 9947–9960. 2019a.

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

Amir Zandieh, Insu Han, Haim Avron, Neta Shoham, Chaewon Kim, and Jinwoo Shin. Scaling neural tangent kernels via sketching and random features. *Advances in Neural Information Processing Systems*, 34:1062–1073, 2021.