Infinite width (finite depth) neural networks benefit from multi-task learning unlike shallow Gaussian Processes - an exact quantitative macroscopic characterization

INFINITE WIDTH (FINITE DEPTH) NEURAL NETWORKS BENEFIT FROM MULTI-TASK LEARNING UNLIKE SHALLOW GAUSSIAN PROCESSES – AN EXACT QUANTITATIVE MACROSCOPIC CHARACTERIZATION

JAKOB HEISS, JOSEF TEICHMANN AND HANNA WUTTE

ABSTRACT. We prove in this paper that optimizing wide ReLU neural networks (NNs) with at least one hidden layer using ℓ_2-regularization on the parameters enforces multi-task learning due to representation-learning – also in the limit of width to infinity. This is in contrast to multiple other results in the literature, in which idealized settings are assumed and where wide (ReLU)-NNs loose their ability to benefit from multi-task learning in the infinite width limit. We deduce the ability of multi-task learning from proving an exact quantitative macroscopic characterization of the learned NN in an appropriate function space.

1. INTRODUCTION

One key difference of deep learning models, such as deep *neural networks* (NNs), in contrast to shallow learning models, such as *Gaussian Processes* (GPs)[1], is that deep learning methods are capable of benefiting from multi-task learning. Multi-task learning [5] is highly connected to representation learning, feature learning, metric learning and wiring (see section 4). Deep NNs with multi-dimensional output use the same weights in all hidden layers and only the last layer contains different weights for different outputs (or tasks). In various applications, training with high-dimensional output has outperformed separate training of the individuals tasks [5, 19, 7, 21, 3]. Especially if the available data for some tasks is limited, these tasks greatly benefit from the other tasks.[2]

Due to the success of deep NNs, there is high interest in studying their generalization behavior (macroscopically[3] in function space). In the case of infinite width neural networks [12, 14, 13] have shown in specific settings that deep NNs are equivalent to shallow GPs which cannot benefit from multi-task learning. They

The authors gratefully acknowledge the support from ETH-foundation. We are very thankful for numerous helpful discussions, feedback and code-implementations – especially to Alexis Stockinger.

[1]Within this paper "GPs" always refer to shallow GPs where a prior GP always has a fixed kernel that is not learned from the data. The GPs considered in [12, 14, 13] have completely independent outputs. (In theory it would be possible to define a prior GP with fixed kernel where the outputs are correlated, if one already has a prior belief that the outputs have a positive or negative correlation, but this is not considered true multi-task learning: It does not include representation-learning and with GPs one cannot express any dependence of outputs without an asymmetric prior belief in a certain sign of the correlation of the outputs.)

[2]This paper is still work in progress. We are aware that references might be incomplete. Please do contact us in case.

[3]We refer to the specific regularization structure on function space as macroscopic behavior.

are aware of this problem and [14] suggests to define a specific prior for the weights to circumvent this problem. [17, 24] in contrast suggest to use infinite depth NNs, with a fixed ratio of width and depth to circumvent it.

In this paper we prove that neural networks optimized under ℓ_2-regularization can already benefit from multi-task learning due to representation learning. Furthermore, we give an exact quantitative macroscopic characterization of the (generalization) behaviour in this setting. This can be seen as the infinite width limit on function space of *maximum a posterioris* (MAPs) on parameter space of Gaussian *Bayesian neural networks* (BNNs) in contrast to the MAP of an infinite width limit of a Gaussian BNN. Note that exchanging the order of the MAP-operator, the limit-operator, and when to change from parameter space to function space is crucial (see section 2.1) for this analysis. Moreover, we briefly discuss that infinite width NNs trained with gradient descent without explicit regularization can also benefit from multi-task learning because of implicit regularization when one does *not* use the scaling suggested by *neural tangent kernel* (NTK) theory by Jacot et al. [12]. Seemingly small differences in the setting how NNs are trained can therefore lead to very different generalization behavior.

2. Setting & Notation

We consider fully connected, shallow NNs (i.e., NNs with one hidden layer) with ReLU activation and deep "stacked" NNs that are obtained by concatenating such shallow neural networks. We denote a fully connected, shallow neural network as a "stack". A fully connected, deep, stacked neural network $\mathcal{NN}_\theta$ is then given as a concatenation of stacks and an element-wise activation function $\tilde{\sigma}$ (not necessarily ReLU as for the hidden layers of a stack),

$$\mathcal{NN}_\theta := \ell^{-1} \circ \mathcal{NN}^{\#\text{stacks}}_{\theta_{(\#\text{stacks})}} \circ \tilde{\sigma} \circ \mathcal{NN}^{\#\text{stacks}-1}_{\theta_{(\#\text{stacks}-1)}} \circ \cdots \circ \tilde{\sigma} \circ \mathcal{NN}^{1}_{\theta_{(1)}}, \tag{1}$$

where ℓ^{-1} is a final activation function (e.g. identity or soft-max which corresponds to a the inverse of a link function in classical statistics) and $\tilde{\sigma}$ is any Lipschitz-continous activations function. Throughout the paper, we focus on stacks as in Definition 2.1, see Appendix A for results on different stacks.

Definition 2.1 (Deep Stacked Neural Network)**.** A *deep stacked neural network* is defined as in (1) with stacks $\mathcal{NN}^j_{\theta_{(j)}} : \mathbb{R}^{d_{j-1}} \to \mathbb{R}^{d_j}$ s.t.

$$\forall x \in \mathbb{R}^{d_{j-1}} \quad : \quad \mathcal{NN}^j_{\theta_{(j)}}(x) = \sum_{k=1}^{n_j} w^{(j)}_k \max\left(0, {b_k}^{(j)} + v^{(j)}_k x\right) + c^{(j)}, \tag{2}$$

with[4]

- number of hidden neurons $n_j \in \mathbb{N}$ in the j-th stack, not necessarily equal dimensions $d_{\text{in}} = d_0, \ldots, d_j, \ldots, d_{\#\text{stacks}} = d_{\text{out}} \in \mathbb{N}$ that we call *bottleneck dimensions* and ReLU activation function,
- weights $v^{(j)}_k \in \mathbb{R}^{d_{j-1}}$, $w^{(j)}_k \in \mathbb{R}^{d_j}$, $k = 1, \ldots, n_j$ and
- biases $c^{(j)} \in \mathbb{R}^{d_j}$, $b^{(j)}_k \in \mathbb{R}$, $k = 1, \ldots, n_j$.

[4]One could include an additional bias $c \in \mathbb{R}$ to the last layer too. However in the limit $n \to \infty$ this last-layer bias c does not change the behavior of the trained network-functions $\mathcal{RN}_{w^T}$ or $\mathcal{RN}^{*,\tilde{\lambda}}$. In Figure 8f this last layer bias c was included in the training.

- Weights and biases are collected in $\theta = \left(\theta_{(j)}\right)_{j\in\{1,\dots,\#\text{stacks}\}}$ with

  $\theta_{(j)} := (v^{(j)}, b^{(j)}, w^{(j)}, c^{(j)}) \in \Theta_j := \mathbb{R}^{n_j\times d_{j-1}} \times \mathbb{R}^{n_j} \times \mathbb{R}^{d_j\times n_j} \times \mathbb{R}^{d_j}$.

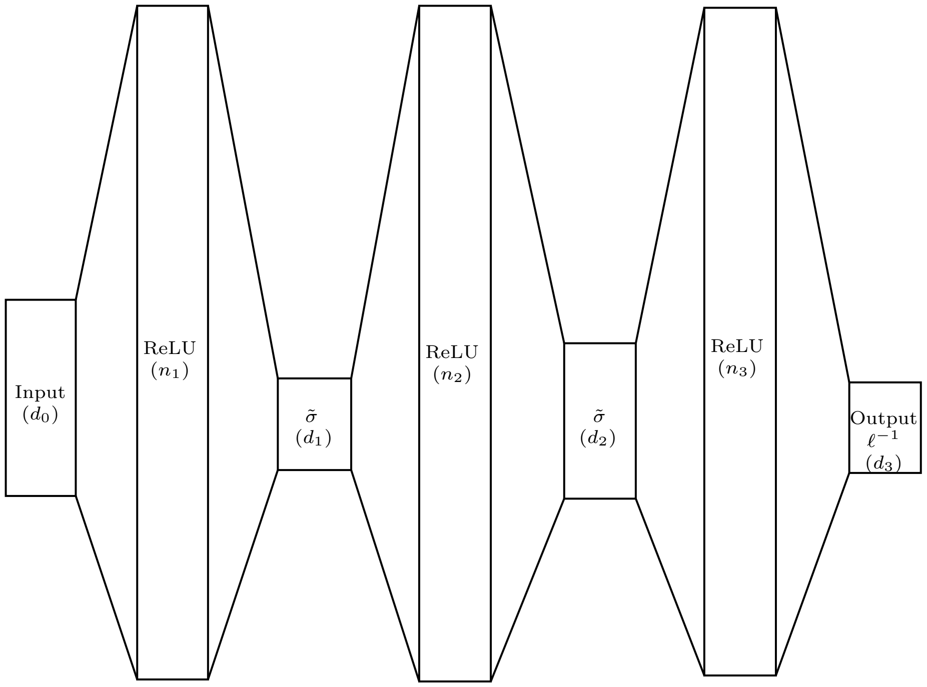


FIGURE 1. Schematic representation of a Deep Stacked NN from Definition 2.1 with #stacks = 3 stacks, where we show the activation functions and the dimensions of the layers of this feed forward NN.

Within this paper, we assume a quadratic training loss

$$L\left(\hat{f}\right) := \sum_{i=1}^{N} \left\| \hat{f}(x_i^{\text{train}}) - y_i^{\text{train}} \right\|_2^2 \tag{3}$$

for simplicity, but actually most of the results would remain true under milder assumptions on $L : W^1 \to \mathbb{R}$ allowing most losses that are typically used in practice and even losses used in applications beyond classical supervised learning. [5]

[5] W^1 denotes the space of weakly differentiable functions. Almost all results here hold analogously for most other supervised learning losses for finite number of training data points (e.g. cross-entropy loss or ℓ_1-loss). For the results, where we let the width of all hidden layers go to infinity (i.e. $d_j \to \infty$ for all $j \in \{1,\dots,\#\text{stacks}-1\}$ and $n_j \to \infty$ for all $j \in \{1,\dots,\#\text{stacks}\}$)it is important tho assume a finite number of training points N. However, we believe that most of the results where we keep the dimension of every second hidden layer d_j finite also hold for more general losses that only have to fulfill similar assumptions as in [9]. These general types of losses can also be used in applications beyond classical supervised learning settings (e.g. [8]). We will formulate precise assumptions on the loss in a future update of this paper.

In this paper we focus on understanding deep stacked ReLU NNs (Definition 2.1) optimized with ℓ_2-regularization (weight decay), i.e.,

$$\mathcal{NN}_{\theta^{*\lambda}} \text{ with } \theta^{*,\lambda} \in \arg\min_{\theta} \left(L\left(\mathcal{NN}_{\theta}\right) + \lambda \left\|\theta\right\|_2^2 \right), \tag{4}$$

for the case of large numbers of n_j for $j = 1 \dots, \#\text{stacks}$. First, we formulate the theory for networks where the widths of every second layer stays finite (i.e. $d_j \in \mathbb{N}$ stay fixed) by letting only the width of the other layers go to infinity (i.e. $n_j \to \infty$). Then, we also derive the theory for the case where the width of every hidden layer goes to infinity (i.e. $n_j, d_j \to \infty$).

2.1. **Connection to BNNs.** [6] The solution $\theta^{*,\lambda}$ of eq. (4) is the MAP on parameter space of a Gaussian BNN, when the ratio of Gaussian i.i.d. data noise variance and the variance of the Gaussian i.i.d. prior of parameters is λ.

In this paper we study the limit in function space of $\mathcal{NN}_{\theta^{*\lambda}}$ as the number of hidden neurons goes to infinity. We even show that resulting function $\mathcal{NN}_{\theta^{*\lambda}} = \lim_{n\to\infty} \mathcal{NN}_{\theta^{*\lambda}}$ already reaches the limit when $n_j > N$ for all $j \in \{1, \dots, \#\text{stacks}\}$. From this we derive that arbitrarily wide $\mathcal{NN}_{\theta^{*\lambda}}$ can benefit from multi-task learning by representation-learning unlike shallow GPs (even if $n_j \gg N$ and even a single hidden layer is sufficient, i.e. $\#\text{stacks} \geq 1$). This might sound surprising at the first moment, since an insufficient summary of [14] could be vaguely formulated as "Infinitely wide Gaussian BNNs are equivalent to shallow GPs" and the MAP[7] of a GP is not capable of benefiting from multi-task learning or representation learning, because of the fixed data-independent kernel. The solution to this paradox is that exchanging the order of the MAP-operator[8] and the $\lim_{n\to\infty}$-operator[9] and if the MAP is calculated on parameter space or function space vastly changes the behavior of the obtained function: Neal [14] shows that the prior of a very wide BNN is similar to a GP. But Neal [14] never claims that the MAP of a very wide BNN is close to the MAP of the corresponding GP. We show that that if one actually calculates the MAP $\mathcal{NN}_{\theta^{*\lambda}} = \lim_{n\to\infty} \mathcal{NN}_{\theta^{*\lambda}}$ of a sufficiently wide BNN as given in eq. (4) is typically not close at all to the MAP of the corresponding GP. We think this result is important since the gradient descend-based algorithms typically used in practice aim to approximate eq. (4).

3. Main Theorem: Characterizing the learned function $\lim_{n\to\infty} \mathcal{NN}_{\theta^{*\lambda}}$

The goal of this section is to formulate $\lim_{n\to\infty} \mathcal{NN}_{\theta^{*\lambda}}$ as the solution of an optimization problem of the form

$$f^{*,\lambda} \in \arg\min_{f} \left(L\left(f\right) + \lambda P(f) \right) \tag{5}$$

to better understand how the solution of (4) behaves in function space in the case of many hidden neurons.

[6]In section 2.1 we actually assume for multiple statements that L is the squared loss as given in eq. (3), while outside section 2.1 most results hold for more general losses too.

[7]The MAP of a GP is always interpreted in a Cameron-Martin sense [4] in this paper which is equivalent to taking point-wisely the MAP of the point-wise marginals.

[8]The MAP-operator maps a prior and the observed data to the max a posteriori (MAP).

[9]We use $\lim_{n\to\infty}$ as a short notation for $\lim_{(n_1,\dots,n_{\#\text{stacks}})\to(\infty,\dots,\infty)}$.

The form of the P-functional depends on the architecture of the network:

$$P(f) = \inf_{\substack{(h_1,\dots h_{\#\text{stacks}}),\text{ s.t.}\\ f=\ell^{-1}\circ h_{\#\text{stacks}}\circ\cdots\circ\tilde{\sigma}\circ h_1}} \left(P_1(h_1) + P_2(h_2) + \cdots + P_{\#\text{stacks}}(h_{\#\text{stacks}})\right), \tag{6}$$

where P_j is given for each stack as[10]

$$P_j(h_j) := \min_{\substack{\varphi\in\mathcal{T},\ c\in\mathbb{R}^{d_j}\text{ s.t.}\\ h_j=\int_{S^{d_j-1-1}}\varphi_s(\langle s,\cdot\rangle)\,ds+c}} \left(\int_{S^{d_j-1-1}}\int_{\mathbb{R}} \frac{\left\|\varphi_s(r)''\right\|_2}{g(r)}\,dr\,ds + \|c\|_2^2\right), \tag{7}$$

where

$$\mathcal{T} := \left\{\varphi\in\mathcal{C}^2(\mathbb{R},\mathbb{R}^{d_j})^{S^{d_j-1-1}} \middle| \forall s\in S^{d-1}: \lim_{r\to-\infty}\varphi_s(r)=0 \text{ and } \lim_{r\to-\infty}\frac{\partial}{\partial r}\varphi_s(r)=0\right\}$$

handles a boundary condition, S^{d-1} denotes the $(d-1)$-dimensional unit sphere and we have a weighing function $g(r) = \frac{1}{\sqrt{r^2+1}}$.

Neither the solutions $\mathcal{NN}_{\theta^{*\lambda}}$ to eq. (4) nor the solutions $f^{*,\lambda}$ to eq. (5) have to be unique. But even in cases where they are not unique the set of solutions to eq. (4) converges to the set of solutions to eq. (5) as $n\to\infty$.

Theorem 3.1. *Using the definitions from Sections 2 and 3, it holds that for a sufficiently large*[11] *number of neurons $n > N$ every solution $\mathcal{NN}_{\theta^{*\lambda}}$ to eq. (4) is a solution to eq. (5) too, i.e.,*

$$\mathcal{NN}_{\theta^{*\lambda}} \in \operatorname*{arg\,min}_{f\in W^1}\left(L(f) + \lambda P(f)\right).$$

[10]As we need to apply P_j on functions $h_j \notin \mathcal{C}^2$, we always use the symbol P_j for its lower semi-continuous continuation, i.e., $P_j : W^1 \to \mathbb{R}\cup\{\infty\}, f \mapsto P_j(f) := \lim_{\varepsilon\to 0+} \inf_{\tilde{f}\in C^2:\|\tilde{f}-f\|_{W^{1,1}}<\varepsilon} P_j(\tilde{f})$, where we use the symbol "P_j" on the right-hand-side of this equation for the functional on the space of twice continuously differentiable functions $\mathcal{C}^2$ and on the left-hand-side we use the same symbol for its continuation to the space of once weakly differential functions W^1. Alternatively one could replace $\mathcal{C}^2$ by W^1 in the definition of $\mathcal{T}$ and interpret the formula $\int_{\mathbb{R}} \frac{\left\|\varphi_s(r)''\right\|_2}{g(r)}\,dr$ as a symbol for $I_g(\varphi_s(r))$, where $I_g : W^1 \to \mathbb{R}\cup\{\infty\}, f\mapsto I_g(f) := \lim_{\varepsilon\to 0+}\inf_{\tilde{f}\in C^2:\|\tilde{f}-f\|_{W^{1,1}}<\varepsilon} \tilde{I}_g(\tilde{f})$ is the lower semi-continuous continuation of $\tilde{I}_g : C^2 \to \mathbb{R}\cup\{\infty\}, f\mapsto \tilde{I}_g(f) := \int_{\mathbb{R}}\frac{\left\|f(r)''\right\|_2}{g(r)}\,dr$. (This is actually a continuation, because of [2].) This results for example in $\int_{\mathbb{R}} \frac{\left\|w_k \max\left(0,\left\langle v_k, \frac{v_k}{\|v_k\|_2} r\right\rangle + b_k\right)''\right\|_2}{g(r)}\,dr = \frac{\|v_k\|_2\|w_k\|_2}{g\left(\frac{-b_k}{\|v_k\|_2}\right)}$. Equation (5) could be formulated more precisely as $f^{*,\lambda}\in\operatorname{arg\,min}_{f\in W^1}\left(L(f)+\lambda P(f)\right)$.

[11]$n > N$ is a short notation for $\forall j\in\{1,\dots,\#\text{stacks}\}: n_j > N$. Note that for a more general loss L that does not only depend the values of f on finitely many training data points as described in footnote 5, even for arbitrarily large $n \gg N$, it can happen that $\mathcal{NN}_{\theta^{*\lambda}}$ is not exactly in the set of solutions of eq. (5). Nonetheless, in this case we will prove in a future version of this paper that $\mathcal{NN}_{\theta^{*\lambda}}$ gets arbitrarily close to the set of solutions of eq. (5) as $n\to\infty$ similarly to eq. (8) in the other direction.

Furthermore, it holds that for every compact $K \subset \mathbb{R}^{d_{in}}$, $\forall \epsilon \in \mathbb{R}_{>0}$:[12]

$$\forall f^{*,\lambda} \in \operatorname*{arg\,min}_{f \in W^1} \left(L(f) + \lambda P(f) \right) : \exists \tilde{n} \in \mathbb{N}^{\#stacks} : \forall n > \tilde{n} :$$
$$\exists \theta^{*,\lambda} \in \operatorname*{arg\,min}_{\theta} \left(L(\mathcal{NN}_\theta) + \lambda \|\theta\|_2^2 \right) : \left\| f^{*,\lambda} - \mathcal{NN}_{\theta^{*\lambda}} \right\|_{W^{1,\infty}(K)} < \epsilon \tag{8}$$

Proof. We will add a detailed proof in a future version of the paper. Similar theorems for shallow NNs with one-dimensional output have already been presented in concurrent and previous work, such as [6, 15, 16, 20, 23, 9, 11]. □

To some extent one can also understand the limit $d_j \to \infty$ for $j \in \{1, \dots, \#\text{stacks} - 1\}$ if one wants to let the width of all hidden layers go to infinity.

Corollary 3.2. *Let $\tilde{\sigma}$ be ReLU or linear, let $n > N$ and $d_j > N$, then there exists $\theta^{*,\lambda} \in \arg\min_\theta \left(L(\mathcal{NN}_\theta) + \lambda \|\theta\|_2^2 \right)$ such that the function $\mathcal{NN}_{\theta^{*\lambda}}$ can also be represented by a network with $d_j = N+1$ and $\mathcal{NN}_{\theta^{*\lambda}} \in \arg\min_{f \in W^1} \left(L(f) + \lambda P(f) \right)$, where $d_j = N+1$ can be used in the definition of P.*

Proof. The proof follows from Theorem 3.1 and [18]. □

Corollary 3.2 implies that one could formulate a functional P with $d_j = N + 1 \; \forall j \in \{1, \dots, \#\text{stacks}\}$ such that $\arg\min_f \left(L(f) + \lambda P(f) \right)$ can at least cover some of the solutions $\mathcal{NN}_{\theta^{*\lambda}}$ of wide NNs, where all layers can be arbitrarily much wider than N (e.g. $n \to \infty$ and $d_j \to \infty \; \forall j \in \{1, \dots, \#\text{stacks}\}$).

Remark 3.3 (No regularization on biases). If one does not regularize the biases but only the weights, one obtains

$$P_j(h_j) := \min_{\substack{\varphi \in \mathcal{T},\; c \in \mathbb{R}^{d_j} \text{ s.t.} \\ h_j = \int_{S^{d_{j-1}-1}} \varphi_s(\langle s, \cdot \rangle)\, ds + c}} \left(\int_{S^{d_{j-1}-1}} \int_{\mathbb{R}} \left\| \varphi_s(r)'' \right\|_2 dr\, ds \right), \text{where} \tag{9}$$

one can either keep the definition of $\mathcal{T}$ or redefine it as

$$\mathcal{T} := \left\{ \varphi \in \mathcal{C}^2(\mathbb{R}, \mathbb{R}^{d_j})^{S^{d_{j-1}-1}} \middle| \forall s \in S^{d-1} : \lim_{r \to -\infty} \frac{\partial}{\partial r} \varphi_s(r) = 0 \right\}.$$

In this case, if $d_j = 1$, [16] provides a simpler reformulation of P_j from eq. (9), i.e,

$$P_j(h_j) := \min_{\substack{\varphi \in \mathcal{C}^2(\mathbb{R}, \mathbb{R}^{d_j})^{S^{d_{j-1}-1}},\; c \in \mathbb{R}^{d_j} \text{ s.t.} \\ h_j = \int_{S^{d_{j-1}-1}} \varphi_s(\langle s, \cdot \rangle)\, ds + c}} \left(\int_{S^{d_{j-1}-1}} \max \left(\int_{\mathbb{R}} \left| \varphi_s(r)'' \right| dr, \left| \lim_{r \to -\infty} \varphi_s(r)' + \lim_{r \to +\infty} \varphi_s(r)' \right| \right) ds \right), \tag{10}$$

(where the spheres could also be replaced by half-spheres without changing the functional P_j). Equation (10) can be particularly intuitively interpreted as a generalized additive model (GAM), where first, instead of only using finitely many directions $(e_1, \dots, e_{d_j})$ all possible directions s are used, second, instead of the typical smoothing spline regularization $\int_{\mathbb{R}} \left\| \varphi_s(r)'' \right\|_2^2 dr$ a L_1-regularization $\int_{\mathbb{R}} \left\| \varphi_s(r)'' \right\|_2 dr$ is applied and third, the first derivative additionally gets regularized. Equation (7) behaves qualitatively similar, but also the zeroth derivative gets slightly regularized

[12] Theorem 3.1 holds for every fixed number $\#\text{stacks} \in \mathbb{N}$ and for every fixed Lipschitz-continous activation function $\tilde{\sigma}$ in-between the stacks (the hidden layers within each stack always have ReLU-activation functions. Moreover, $n > \tilde{n}$ is understood component-wise. On the first sight, it looks as if n does not appear after "$\forall n > \tilde{n}$", but recall that when we write $\mathcal{NN}_\theta$ this always refers to a network with n_j neurons in the corresponding layers.

and the second derivative gets penalized more strongly far away from zero than close to zero. We don't think that the regularization eq. (7) is more desirable than eq. (10), but even if one does not regularize the biases, the obtained functions will in practice have some qualitative aspects of eq. (7), since gradient descent initialized close to zero implicitly regularizes the bias too.

Remark 3.4 (Random hidden layers [11]). Comparing eq. (7) to [11], where the first-layer weights and biases v and b are not trained but chosen randomly, one can see that the main difference is that the integrand $\frac{\left\|\varphi_s(r)''\right\|_2}{g(r)}$ replaced the integrand $\frac{\left\|\varphi_s(r)''\right\|_2^2}{g(r)}$, i.e., the integrand in eq. (7) takes the square root of the numerator and also the weighting function g does not depend on the distribution of v and b anymore (since v and b are trainable now too). If one still sampled $v^{(j)}$ and $b^{(j)}$ randomly without training them, one could plug in [11, $P_{\circ}^{g}$] for P_j in eq. (6). See [11] for some intuition on how P-functionals relate to classical statistical models (GAMs).

4. Discussion (Deep Learning vs. Shallow GPs)

Already a single hidden layer is sufficient to read off representation learning and multi-task learning from the P-functional of the previous section: In the case of #stacks $= 1$, $P = P_1$ from eq. (7) or (10) (depending on the regularization of the biases). The square-root in the definition of the Euclidean norm $\left\|\cdot\right\|_2$ that appears in eqs. (7), (9) and (10) is responsible for the representation learning and multi-task learning: If for example multiple outputs f_k almost only change[13] in certain directions $s \in S^{d_{\text{in}}-1}$, then $\left\|\varphi_s(r)''\right\|_2$ is much stronger for these directions s than for others.

Thus, the marginal costs for second derivative of any other $\tilde{k}$th component of $\varphi_s(r)$ are much smaller for these directions s than for other directions, because of the strict concavity of the square root-function. So the $\tilde{k}$th task can learn from the other task which directions tend to be more important than other directions and thus prefer functions $f_{\tilde{k}}$ which mainly change in the directions where the other tasks also change a lot. (Note that the architecture is still universal and thus is also able to learn functions f where different components f_k change in very different directions *if* there is enough data evidence to do so.)

Already for one-dimensional input, the square-root can lead to multi-task learning: If some outputs f_k have stronger second derivative $|f_k''(x)|$ or even kinks at some positions x, other outputs $f_{\tilde{k}}$ will also prefer to have stronger second derivative or even kinks at these positions x. If one samples the first layer weights and biases randomly and only train the second layer, the main difference in P is that $\left\|\cdot\right\|_2$ would be replaced by $\left\|\cdot\right\|_2^2$ and squaring the Euclidean norm cancels the square root that connects the outputs to each other.

[13]We say that f_k "changes" a lot in a direction s, when changing the input x in the direction of s the output $f_k(x)$ changes a lot, possibly very non-linearly with very strong second derivative in this direction s. We say that f_k does almost not change in other directions when changing the input in other directions has little influence on the output, i.e. the output in these other directions is mostly linear and very flat (i.e. it has low first derivative and very low second derivative in these directions).

Without the square-root, learning a separate function f_k for each task would result exactly in the same functions f_k as training them all together[14], since

$$\int \left\|f^{''}(x)\right\|_2^2 dx = \int \sum_{k=1}^{d_{\text{out}}} \left(f_k^{''}(x)\right)^2 dx = \sum_{k=1}^{d_{\text{out}}} \int \left(f_k^{''}(x)\right)^2 dx.$$

On the contrary, with the square-root this is obviously not the case, since

$$\int \left\|f^{''}(x)\right\|_2 dx = \int \sqrt[2]{\sum_{k=1}^{d_{\text{out}}} \left(f_k^{''}(x)\right)^2} dx$$

is in general not equal to $\sum_{k=1}^{d_{\text{out}}} \int \left|f_k^{''}(x)\right| dx$.

Thus one can see that already a single hidden layer is sufficient to get the effect of multi-task learning for $\lim_{n\to\infty} \mathcal{NN}_{\theta^{*\lambda}}$ when $\mathcal{NN}_{\theta^{*\lambda}}$ are trained with ℓ_2-regularization. Note that for exactly the same NN-architecture the corresponding limits discussed in [12, 14, 17, 24, 13] result in a GP-regression with absolutely no multi-task learning benefits, where learning the tasks separately would result in exactly the same functions as jointly learning them.

Increasing the number of stacks $\#\text{stacks} > 1$ further strengthens the representation learning and multi-task learning effects, because not only does the square-root in each P_j enforce multi-task learning for each h_j, but all the functions h_j for $j \in \{1, \dots, \#\text{stacks} - 1\}$ are shared by the all the tasks. So, $H := \tilde{\sigma} \circ h_{\#\text{stacks}-1} \circ \cdots \circ \tilde{\sigma} \circ h_1$ has to be learned to transform inputs x into a vector representation $H(x)$ that allows jointly for all functions $(\ell \circ f)_k$ to be nicely representable as $h_{\#\text{stacks},k} \circ H$ such that $P_{\#\text{stacks}}(h_j)$ is not too large. In Appendix B we visualize this multi-task learning benefits on a simple example.

In a future version of this paper, we want to discuss in more detail the connections between multi-task learning, representation learning, feature learning, metric learning, wiring, transfer learning, and [10, Desiderata D4]. In future work we also want to write down how these results on different limits of infinitely wide NNs can shed light on an open aspect of a paradox regarding the mysterious benefits of "cold posteriors"[15] in the context of Bayesian inference. We also want to discuss gradient descend and *implicit* regularization in a future version of this paper.

References

[1] Laurence Aitchison. A statistical theory of cold posteriors in deep neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Rd138pWXMvG. 8

[2] Luigi Ambrosio, Maria Colombo, and Simone Di Marino. Sobolev spaces in metric measure spaces: reflexivity and lower semicontinuity of slope, 2012. URL https://arxiv.org/abs/1212.3779. 5

[3] Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q. Tran, Dara Bahri, Jianmo Ni, Jai Gupta, Kai Hui, Sebastian Ruder, and Donald Metzler. Ext5: Towards

[14] We assume that the final activation function ℓ^{-1} is the identity whenever we say that different tasks do not influence each other. For a soft-max activation function the different outputs obviously influence each other even without representation-learning.

[15] Aitchison [1] only partially solves the paradox explained in [22].

extreme multi-task scaling for transfer learning, 11 2021. URL https://arxiv.org/abs/2111.10952. 1

[4] R. H. Cameron and W. T. Martin. Transformations of weiner integrals under translations. *Annals of Mathematics*, 45(2):386–396, 1944. ISSN 0003486X. URL http://www.jstor.org/stable/1969276. 4

[5] Rich Caruana, Lorien Pratt, and Sebastian Thrun. Multitask learning. *Machine Learning 1997 28:1*, 28:41–75, 1997. ISSN 1573-0565. doi: 10.1023/A:1007379606734. URL https://link.springer.com/article/10.1023/A:1007379606734. 1

[6] Lénaïc Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 1305–1338. PMLR, 09–12 Jul 2020. URL https://proceedings.mlr.press/v125/chizat20a.html. 6

[7] Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning, 2021. URL https://arxiv.org/abs/2109.04617. Summarizing blog post: https://ai.googleblog.com/2021/10/deciding-which-tasks-should-train.html. 1

[8] Matteo Gambara and Josef Teichmann. Consistent recalibration models and deep calibration. *arXiv: Computational Finance*, 2020. URL https://arxiv.org/abs/2006.09455. 3

[9] Jakob Heiss, Josef Teichmann, and Hanna Wutte. How implicit regularization of Neural Networks affects the learned function – Part I, November 2019. URL https://arxiv.org/abs/1911.02903. 3, 6

[10] Jakob Heiss, Jakob Weissteiner, Hanna Wutte, Sven Seuken, and Josef Teichmann. NOMU: Neural optimization-based model uncertainty, 2021. URL https://arxiv.org/abs/2102.13640. 8

[11] Jakob Heiss, Josef Teichmann, and Hanna Wutte. How (Implicit) Regularization of ReLU Neural Networks Characterizes the Learned Function Part II: the multi-D Case of Two Layers with Random First Layer, unpublished work in progress, will be published soon. 6, 7

[12] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018. URL https://arxiv.org/abs/1806.07572v3. 1, 2, 8

[13] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes, 2018. URL https://arxiv.org/abs/1711.00165. 1, 8

[14] Radford M. Neal. *Bayesian Learning for Neural Networks*, volume 118. Springer New York, 1996. ISBN 978-1-4612-0745-0. doi: 10.1007/978-1-4612-0745-0. URL http://link.springer.com/10.1007/978-1-4612-0745-0. 1, 2, 4, 8

[15] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In Search of the Real Inductive Bias: On the Role of Implicit Regularization in Deep Learning. *arXiv e-prints*, art. arXiv:1412.6614, December 2014. URL https://arxiv.org/abs/1412.6614v4. 6

[16] Greg Ongie, Rebecca Willett, Daniel Soudry, and Nathan Srebro. A function space view of bounded norm infinite width relu nets: The multivariate case. *arXiv preprint arXiv:1910.01635*, 2019. URL https://arxiv.org/pdf/1910.01635.pdf. 6

[17] Daniel A. Roberts, Sho Yaida, and Boris Hanin. The principles of deep learning theory, 2021. URL https://arxiv.org/abs/2106.10165. 2, 8

[18] Saharon Rosset, Grzegorz Swirszcz, Nathan Srebro, and Ji Zhu. ℓ_1 regularization in infinite dimensional feature spaces. In Nader H. Bshouty and Claudio Gentile, editors, *Learning Theory*, pages 544–558, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-72927-3. URL https://link.springer.com/chapter/10.1007/978-3-540-72927-3_39. 6

[19] Sebastian Ruder. An overview of multi-task learning in deep neural networks, 2017. URL https://arxiv.org/pdf/1706.05098.pdf. 1

[20] Pedro Savarese, Itay Evron, Daniel Soudry, and Nathan Srebro. How do infinite width bounded norm networks look in function space? *arXiv preprint arXiv:1902.05040*, 2019. URL https://arxiv.org/abs/1902.05040. 6

[21] Chau Tran, Shruti Bhosale, James Cross, Philipp Koehn, Sergey Edunov, and Angela Fan. Facebook ai wmt21 news translation task submission, 2021. URL https://arxiv.org/abs/2108.03265. Summarizing blog post: https://ai.facebook.com/blog/the-first-ever-multilingual-model-to-win-wmt-beating-out-bilingual-models. 1

[22] Florian Wenzel, Kevin Roth, Bastiaan S. Veeling, Jakub Swiatkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really?, 2020. URL https://arxiv.org/abs/2002.02405. 8

[23] Francis Williams, Matthew Trager, Daniele Panozzo, Claudio Silva, Denis Zorin, and Joan Bruna. Gradient dynamics of shallow univariate relu networks. In *Advances in Neural Information Processing Systems*, pages 8378–8387, 2019. URL http://papers.nips.cc/paper/9046-gradient-dynamics-of-shallow-univariate-relu-networks.pdf. 6

[24] Sho Yaida. Advancing ai theory with a first-principles understanding of deep neural networks, June 2021. URL https://ai.facebook.com/blog/advancing-ai-theory-with-a-first-principles-understanding-of-deep-neural-networks/. 2, 8

Appendix A. Network Architectures and their P-functionals

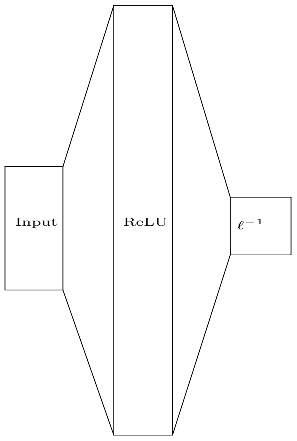


Figure 2. Schematic representation of a Shallow Neural Network

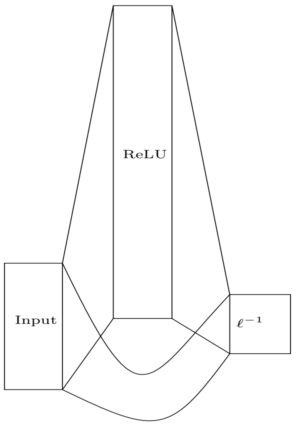


Figure 3. Schematic representation of a Shallow Neural Network with a skip connection

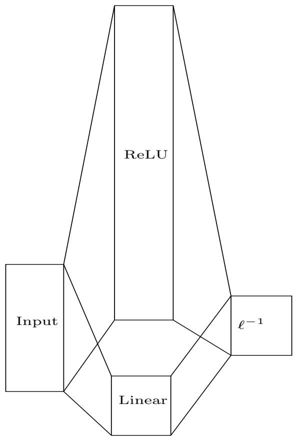


Figure 4. Schematic representation of One Stack

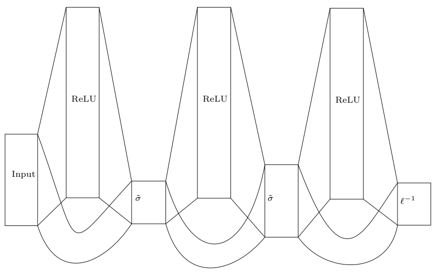


Figure 5. Schematic representation of a simplified three-stacked network

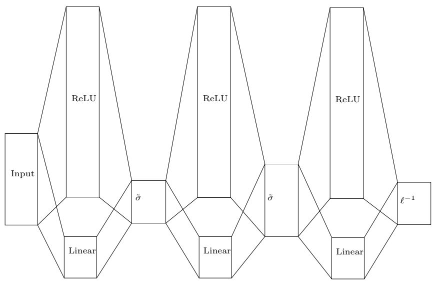


Figure 6. Schematic representation of a three-stacked network which will be studied in more detail in the following work

In a future version of this paper we want to give P-functionals for these modifications of the architecture, e.g. for architectures of the type of Figures 3 and 5, one would get

(11)
$$P_j(h_j) := \min_{\substack{\varphi\in\mathcal{T},\ c\in\mathbb{R}^{d_j}, A\in\mathbb{R}^{d_{j-1}\times d_j} \text{ s.t.}\\ h_j=\int_{S^{d_{j-1}-1}}\varphi_s(\langle s,\cdot\rangle)\,ds+c+A(\cdot)}} \left(\int_{S^{d_{j-1}-1}}\int_{\mathbb{R}} \frac{\left\|\varphi_s(r)''\right\|_2}{g(r)}\,dr\,ds + \|c\|_2^2 + \|A\|_2^2\right),$$

where $\|A\|_2$ is the Frobenius norm of A and

$$\mathcal{T} := \left\{\varphi\in\mathcal{C}^2(\mathbb{R},\mathbb{R}^{d_j})^{S^{d_{j-1}-1}} \,\middle|\, \forall s\in S^{d-1}: \lim_{r\to-\infty}\varphi_s(r)=0 \text{ and } \lim_{r\to-\infty}\frac{\partial}{\partial r}\varphi_s(r)=0.\right\}$$

For an architecture of the type of Figures 4 and 6 one gets:

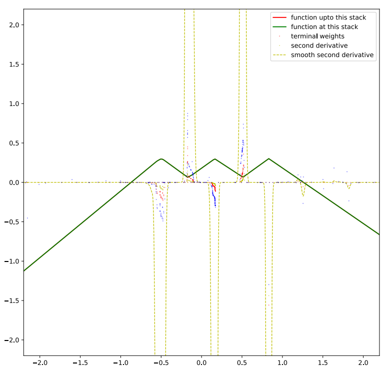
(12)
$$P_j(h_j) := \min_{\substack{\varphi\in\mathcal{T},\ c\in\mathbb{R}^{d_j}, A\in\mathbb{R}^{d_{j-1}\times d_j} \text{ s.t.}\\ h_j=\int_{S^{d_{j-1}-1}}\varphi_s(\langle s,\cdot\rangle)\,ds+c+A(\cdot)}} \left(\int_{S^{d_{j-1}-1}}\int_{\mathbb{R}} \frac{\left\|\varphi_s(r)''\right\|_2}{g(r)}\,dr\,ds + \|c\|_2^2 + \|A\|_{\text{Schatten1}}\right),$$

where $\|A\|_{\text{Schatten1}}$ is the Schatten 1-norm (or Schatten–von-Neumann 1-norm) of A, i.e., the sum of the absolut values of the singualar values of A.

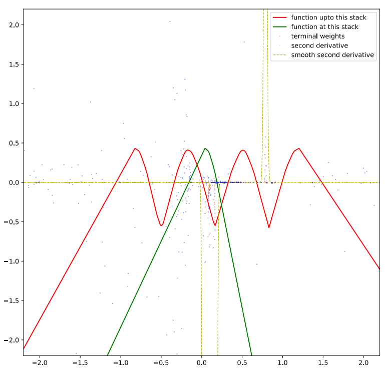
Appendix B. Visualizing multi-task-learning

In this section we present a simple example, where the true function f has $d_{\text{out}} = 7$ outputs which are all periodic with the same periodicity and $d_{\text{in}} = 1$ input. The knowledge that the outputs are periodic is not given to the network a priori and we hope that the network with $\#\text{stacks} = 3, n \gg N, d_1 = d_2 = 1, \tilde{\sigma} = id, \ell^{-1} = id$ (witht he architecture from Figure 6) is able to find a periodic representation H by itself, because this would be helpful for all 7 outputs. In the plots we show $\mathcal{NN}^j_{\theta_{(j)}}$ instead of h_j, since we have obtained θ from actually training a NN with gradient descent (that can get stuck in local minima) for a a finite time instead of calculating the perfect solution. In the following figure, we visualize what each stack has learned:

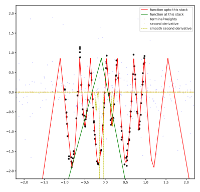
ETH Zürich, D-Math, Rämistrasse 101, CH-8092 Zürich, Switzerland
Email address: jakob.heiss@math.ethz.ch, jteichma@math.ethz.ch, hanna.wutte@math.ethz.ch



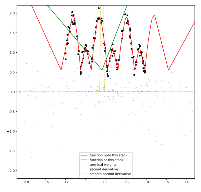
(A) First stack: $\mathcal{NN}^1_{\theta_{(1)}}$ in green and its (distributional) second derivative $\sum_{i=1}^{n_1} v_i^{(1)} w_i^{(1)} \delta_{\xi_i^{(1)}}$ (visualized by yellow dots $(\xi_i^{(1)}, v_i^{(1)} w_i^{(1)})$), where $\xi_i^{(1)} = \frac{-b_i^{(1)}}{v_i^{(1)}}$ and a smoothed version of it (yellow line). The smooth second derivative was obtained from a convolution using a Gaussian kernel. Moreover, the values of the terminal layer's weights w_k at the respective kink positions ξ_k are given (red dots).



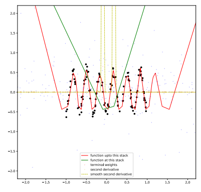
(B) Second stack: $\mathcal{NN}^2_{\theta_{(2)}}$ in green and $H := \mathcal{NN}^2_{\theta_{(2)}} \circ \mathcal{NN}^1_{\theta_{(1)}}$ in red and the (distributional) second derivative $h_2^{''} = \sum_{i=1}^{n_2} v_i^{(2)} w_i^{(2)} \delta_{\xi_i^{(2)}}$ (visualized by yellow dots $(\xi_i^{(2)}, v_i^{(2)} w_i^{(2)})$), where $\xi_i^{(2)} = \frac{-b_i^{(2)}}{v_i^{(2)}}$ and a smoothed version of it (yellow line). The smooth second derivative was obtained from a convolution using a Gaussian kernel. Moreover, the values of the terminal layer's weights w_k at the respective kink positions ξ_k are given (red dots).



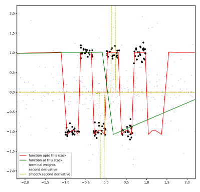
(A) First component of third stack: $\left(\mathcal{NN}^3_{\theta_{(3)}}\right)_1$ in green and $\hat{f}_1 := \left(\mathcal{NN}_\theta\right)_1 := \left(\mathcal{NN}^3_{\theta_{(3)}}\right)_1 \circ \mathcal{NN}^2_{\theta_{(2)}} \circ \mathcal{NN}^1_{\theta_{(1)}}$ in red and the training data points $(x_i^{\text{train}}, y_{i,1}^{\text{train}})$ for $i \in \{1, \dots, N\}$ as black dots and the (distributional) second derivative ${\mathcal{NN}^3_{\theta_{(3)}}}_1''$ is visualized as in the previous plots.



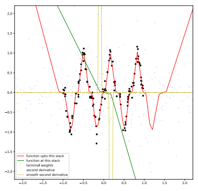
(B) Second component of third stack: $\left(\mathcal{NN}^3_{\theta_{(3)}}\right)_2$ in green and $\hat{f}_2 := \left(\mathcal{NN}_\theta\right)_2 := \left(\mathcal{NN}^3_{\theta_{(3)}}\right)_2 \circ \mathcal{NN}^2_{\theta_{(2)}} \circ \mathcal{NN}^1_{\theta_{(1)}}$ in red and the training data points $(x_i^{\text{train}}, y_{i,2}^{\text{train}})$ for $i \in \{1, \dots, N\}$ as black dots and the (distributional) second derivative ${\mathcal{NN}^3_{\theta_{(3)}}}_2''$ is visualized as in the previous plots.



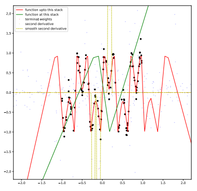
(C) Third component of third stack: $\left(\mathcal{NN}^3_{\theta_{(3)}}\right)_3$ in green and $\hat{f}_3 := \left(\mathcal{NN}_\theta\right)_3 := \left(\mathcal{NN}^3_{\theta_{(3)}}\right)_3 \circ \mathcal{NN}^2_{\theta_{(2)}} \circ \mathcal{NN}^1_{\theta_{(1)}}$ in red and the training data points $(x_i^{\text{train}}, y_{i,3}^{\text{train}})$ for $i \in \{1, \dots, N\}$ as black dots and the (distributional) second derivative ${\mathcal{NN}^3_{\theta_{(3)}}}_3''$ is visualized as in the previous plots.



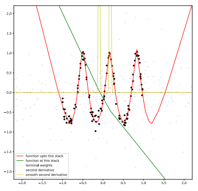
(D) Fourth component of third stack: $\left(\mathcal{NN}^3_{\theta_{(3)}}\right)_4$ in green and $\hat{f}_4 := \left(\mathcal{NN}_\theta\right)_4 := \left(\mathcal{NN}^3_{\theta_{(3)}}\right)_4 \circ \mathcal{NN}^2_{\theta_{(2)}} \circ \mathcal{NN}^1_{\theta_{(1)}}$ in red and the training data points $(x_i^{\text{train}}, y_{i,4}^{\text{train}})$ for $i \in \{1, \dots, N\}$ as black dots and the (distributional) second derivative ${\mathcal{NN}^3_{\theta_{(3)}}}_4''$ is visualized as in the previous plots.



(E) Fifth component of third stack: $\left(\mathcal{NN}^3_{\theta_{(3)}}\right)_5$ in green and $\hat{f}_5 := \left(\mathcal{NN}_\theta\right)_5 := \left(\mathcal{NN}^3_{\theta_{(3)}}\right)_5 \circ \mathcal{NN}^2_{\theta_{(2)}} \circ \mathcal{NN}^1_{\theta_{(1)}}$ in red and the training data points $(x_i^{\text{train}}, y_{i,5}^{\text{train}})$ for $i \in \{1, \dots, N\}$ as black dots and the (distributional) second derivative ${\mathcal{NN}^3_{\theta_{(3)}}}_5''$ is visualized as in the previous plots.



(F) Sixth component of third stack: $\left(\mathcal{NN}^3_{\theta_{(3)}}\right)_6$ in green and $\hat{f}_6 := \left(\mathcal{NN}_\theta\right)_6 := \left(\mathcal{NN}^3_{\theta_{(3)}}\right)_6 \circ \mathcal{NN}^2_{\theta_{(2)}} \circ \mathcal{NN}^1_{\theta_{(1)}}$ in red and the training data points $(x_i^{\text{train}}, y_{i,6}^{\text{train}})$ for $i \in \{1, \dots, N\}$ as black dots and the (distributional) second derivative ${\mathcal{NN}^3_{\theta_{(3)}}}_6''$ is visualized as in the previous plots.



(G) Seventh component of third stack: $\left(\mathcal{NN}^3_{\theta_{(3)}}\right)_7$ in green and $\hat{f}_7 := \left(\mathcal{NN}_\theta\right)_7 := \left(\mathcal{NN}^3_{\theta_{(3)}}\right)_7 \circ \mathcal{NN}^2_{\theta_{(2)}} \circ \mathcal{NN}^1_{\theta_{(1)}}$ in red and the training data points $(x_i^{\text{train}}, y_{i,7}^{\text{train}})$ for $i \in \{1, \dots, N\}$ as black dots and the (distributional) second derivative ${\mathcal{NN}^3_{\theta_{(3)}}}_7''$ is visualized as in the previous plots.