Generating Accurate Pseudo-labels in Semi-Supervised Learning and Avoiding Overconfident Predictions via Hermite Polynomial Activations *

Vishnu Suresh Lokhande
lokhande@cs.wisc.edu

Songwong Tasneeyapant
tasneeyapant@wisc.edu

Abhay Venkatesh
abhay.venkatesh@gmail.com

Sathya N. Ravi
sathya@uic.edu

Vikas Singh
vsingh@biostat.wisc.edu

Abstract

Rectified Linear Units (ReLUs) are among the most widely used activation function in a broad variety of tasks in vision. Recent theoretical results suggest that despite their excellent practical performance, in various cases, a substitution with basis expansions (e.g., polynomials) can yield significant benefits from both the optimization and generalization perspective. Unfortunately, the existing results remain limited to networks with a couple of layers, and the practical viability of these results is not yet known. Motivated by some of these results, we explore the use of Hermite polynomial expansions as a substitute for ReLUs in deep networks. While our experiments with supervised learning do not provide a clear verdict, we find that this strategy offers considerable benefits in semi-supervised learning (SSL) / transductive learning settings. We carefully develop this idea and show how the use of Hermite polynomials based activations can yield improvements in pseudo-label accuracies and sizable financial savings (due to concurrent runtime benefits). Further, we show via theoretical analysis, that the networks (with Hermite activations) offer robustness to noise and other attractive mathematical properties.

1. Introduction

Analyzing the optimization or the loss landscape of deep neural networks has emerged as a promising means to understand the behavior and properties of various neural network architectures [4]. One reason is that insights into how the loss function behaves geometrically is closely tied to the types of optimization schemes that may be needed [29], why specific ideas work whereas others do not [23], and how or whether the corresponding model may generalize to unseen data [30]. Notice that we must leverage such "alternative" strategies as a window into these models' behavior because in deep learning, most models of interest are highly non-linear and non-convex. As a result, one finds that extending mature ideas, which work well for analysis in classical settings (e.g., linear models), is quite a bit harder for most deep architectures of interest.

Why study activation functions? While there are many ways we may study the optimization landscape of deep models, a productive line of recent results [6] proposes analyzing the landscape (i.e., the behavior of the loss as a function of the network parameters) via the **activation functions** of the neural network. This makes a lot of sense because the activation function is one critical place where we introduce non-linearity into the network, and their omission significantly simplifies any analysis. Activation functions greatly influence the functional space which a neural network can represent [14], often the first step in a more formal study of the model's behavior. For example, universal finite-sample expressivity of certain architectures has been shown by fixing the activations to be ReLU functions [9]. In other words, if we use ReLU as activations, such an architecture can be shown to represent any function if the model has more parameters than the sample size. The scope of such work is not just theoretical – for instance, the above results were used to derive a much simpler architecture consisting of only residual convolutional layers and ReLU activations. Further, the estimation scheme needed was also much simpler and required no batch normalization, dropout, or max pooling. In summary, the choice of activations enabled understanding the loss landscape and enabled simplifications.

More on activations functions and loss landscape. The authors in [28] showed that conditions that prevent presence of spurious valleys on the loss landscape can be identified, via the use of smooth activations. Independently, [25] demonstrated that the use of quadratic activation functions enables efficient localization of global minima in certain classes of deep models. Closely related to smooth and quadratic activations, the use of polynomial non-linearity as

*Please direct correspondence to Lokhande, Ravi, Singh.

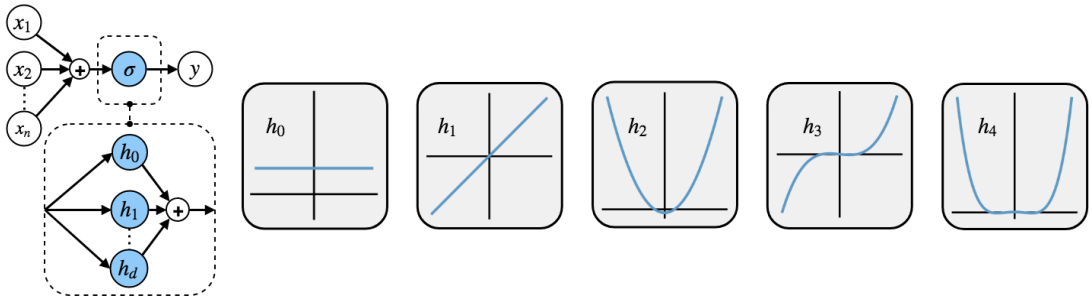


Figure 1: **Hermite Polynomials as Activations** (**leftmost**): Incorporating Hermite Polynomials as an activation function in a single hidden unit one hidden layer network. (**middle**): The functional form of the first 5 hermites are shown in the right.

an activation has also been studied in the last few years. For example, [20] studied deep ReLU networks by using a polynomial non-linearity as an approximation for ReLU: this enabled a much cleaner analysis of the empirical risk landscape. More recently, [14] analyzed the functional space of the networks with the help of polynomial activation based on techniques from algebraic geometry. Earlier, for a one hidden layer network, [6] investigated optimizing the population risk of the loss using stochastic gradient descent and showed that one could avoid spurious local minima by utilizing an orthogonal basis expansion for ReLUs. A key takeaway from this work is that the optimization would behave better if the landscape was nicely behaved — this is enabled via the basis expansion. The foregoing results and analyses, especially the use of **basis expansion**, is interesting and a starting point for the development described here. A common feature of the body of work summarized above is that they rely on networks with polynomial activations (polynomial networks) to analyze the loss landscape. This choice helps make the mathematical exploration easier.

Where is the gap in the literature? Despite the interesting theoretical results summarized above, relatively less is known whether such a strategy or its variants are a good idea for the architectures in computer vision and broadly, in AI, today. We can, in fact, ask a more practically focused question: are there specific tasks in computer vision where such a strategy offers strong empirical advantages? This is precisely the gap this paper is designed to address.

The **main contributions** include: **(a)** We describe mechanisms via which activation functions based on Hermite polynomials can be utilized within deep networks instead of ReLUs, with only minor changes to the architecture. **(b)** We present evidence showing that while these adjustments are not significantly advantageous in supervised learning, our scheme *does* yield sizable advantages in semi-supervised learning speeding up convergence. Therefore, it offers clear benefits in compute time (and cost) needed to attain a certain pseudo-label accuracy, which has direct cost implications. **(c)** We give technical results analyzing the mathematical behavior of such activations, specifically, robustness results showing how the activation mitigates overconfident predictions for (out of distribution) samples.

2. Brief Review of Hermite polynomials

We will use an expansion based on Hermite polynomials as a substitute for ReLU activations. To describe the construction clearly, we briefly review the basic properties.

Hermite polynomials are a class of orthogonal polynomials which are appealing both theoretically as well as in practice, e.g., radio communications [2]. Here, we will use Hermite polynomials [21] defined as,

$$H_i(x) = (-1)^i e^{x^2} \frac{d^i}{dx^i} e^{-x^2}, i > 0; H_0(x) = 1 \quad (1)$$

In particular, we use normalized Hermite polynomials which are given as $h_i = \frac{H_i}{\sqrt{i!}}$. Hermite polynomials are often used in the analysis of algorithms for nonconvex optimization problems [18]. While there are various mathematical properties associated with Hermites, we will now discuss the most important property for our purposes.

Hermite polynomials as bases. Classical results in functional analysis show that the (countably infinite) set $\{H_i\}_{i=0}^{d=\infty}$ defined in (1) can be used as bases to represent smooth functions [22]. Formally, let $L^2(\mathbb{R}, e^{-x^2/2})$ denote the set of integrable functions w.r.t. the Gaussian measure,

$$L^2(\mathbb{R}, e^{-x^2/2}) = \{f : \int_{-\infty}^{+\infty} f(x)^2 e^{-x^2/2} dx < \infty\},$$

It turns out that $L^2(\mathbb{R}, e^{-x^2/2})$ is a Hilbert space with the inner product defined as follows (see [6] for more details)

$$\langle f, g \rangle = \mathbb{E}_{x \sim \mathbb{N}(0,1)}[f(x)g(x)] \quad (2)$$

The normalized Hermite polynomials form an *orthonormal basis* in $L^2(\mathbb{R}, e^{-x^2/2})$ in the sense that $\langle h_i, h_j \rangle = \delta_{ij}$. Here δ_{ij} is the Kronecker delta function and h_i, h_j are any two normalized Hermite polynomials.

Recently, the authors in [6] showed that the lower order terms in the Hermite polynomial series expansion of

ReLU have a different rate of convergence on the optimization landscape than the higher order terms for one hidden layer networks. This leads to a natural **question**: are these properties useful to accelerate the performance of gradient-based methods for deep networks as well?

Hermite Polynomials as Activations. Let $x = (x_1, \ldots, x_n)$ be an input to a neuron in a neural network and y be the output. Let $w = (w_1, w_2, \ldots, w_n)$ be the weights associated with the neuron. Let σ be the non-linear activation applied to $w^T x$. Often we set σ = ReLU. Here, we investigate the scenario where $\sigma(x) = \sum_{i=0}^{d} c_i h_i(x)$, denoted as σ_{hermite}, were h_i's are as defined previously and c_i's are **trainable parameters**. As [6] suggests, we initialized the parameters $c_i's$ associated with hermites to be $c_i = \hat{\sigma}_i$, where $\hat{\sigma}_i = \langle \text{ReLU}, h_i \rangle$. Hermite polynomials as activations on a single layer neural network with a single hidden unit can be seen in Figure 1.

3. Sanity checks: What do we gain or lose?

Replacing ReLUs with other activation functions reviewed in Section 1 has been variously attempted in the literature already, for supervised learning. While improvements have been reported in specific settings, ReLUs continue to be relatively competitive. Therefore, it seems that we should not expect improvements in what are toy supervised learning experiments. However, as we describe below, the experiments yield useful insights regarding settings where Hermites will be particularly useful.

A) Two-layer architectures. We start with the CIFAR10 dataset and a simple two-layer network (more details in the supplement). Basically, we ask if this modification will lead to better or worse performance in accuracy and runtime (over ReLU activation) with all other parameters (e.g., learning rates) fixed. **Positives:** Here, we observed *faster* training loss convergence (over the initial epochs) compared to ReLU in general. **Negatives:** When we assessed the performance as a function of the number of Hermite polynomials d, we see a tradeoff where the speeds first improve with

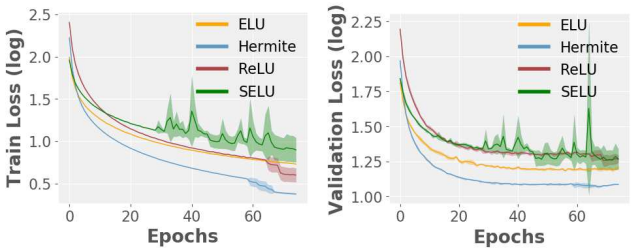


Figure 2: **Other Activation Functions.** We observe an increase in the rate of convergence of the validation loss for Hermites when compared to other activations such as ELU, ReLU and SeLU. A standard auto-encoder model as outlined in [29] is used to test on MNIST dataset.

Method	LR	ϵ	Train Loss	Test Loss
Sigmoid-Adam [29]	10^{-3}	10^{-8}	2.97 ± 0.06	7.91 ± 0.14
Sigmoid-Adam [29]	10^{-3}	10^{-3}	1.90 ± 0.08	4.42 ± 0.29
Hermite-Adam (Ours)	10^{-3}	10^{-8}	$\mathbf{1.89 \pm 0.01}$	$\mathbf{3.16 \pm 0.02}$

Table 1: **Deep autoencoders with Hermite Activations give lower test loss** Results from our implementation following directions from [29].

increasing d and then worsen when d is as high as 8, indicating that too many bases, especially for relatively shallow architectures are not ideal.

B) Deep Autoencoders. Encouraged by the shallow network experiments, we moved to a standard benchmark for neural network optimization runtimes: the deep autoencoders setup from [29]. We replicate the experiments reported in [29] for the MNIST dataset: we use 4 Hermite polynomials as an activation instead of the original sigmoid. **Positives:** We see from Table 1 that Hermite activations not only converge faster but also achieve lower test loss. We also evaluated how activation functions such as ReLU, SeLU or ELU perform in this setting. As shown in Figure 2, we find that Hermites still offer runtime improvements here with a comparable validation loss. We find that SeLU does not seem to be as stable as the other activation functions. Therefore, in the remainder of this paper, we study Hermite activations compared to ReLUs as a baseline. Of course, it is quite possible that some other activations may provide slightly better performance compared to ReLUs in specific settings. But the choice above simplifies the presentation of our results and is consistent with the general message of our work: while research on polynomial activations has so far remained largely theoretical, we intend to show that they are easily deployable and offer various practical benefits.

Adjustments needed for deeper architectures? When implemented naively, Hermite activations do **not** work well for deeper architectures directly, which may be a reason that they have not been carefully explored so far. Basically, with no adjustments, we encounter a number of numerical issues

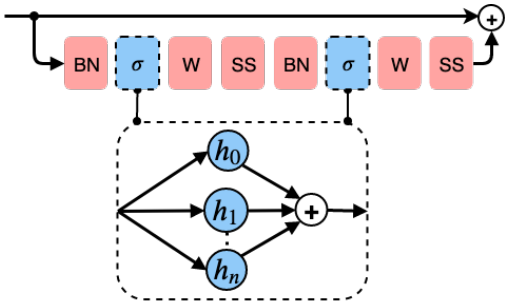
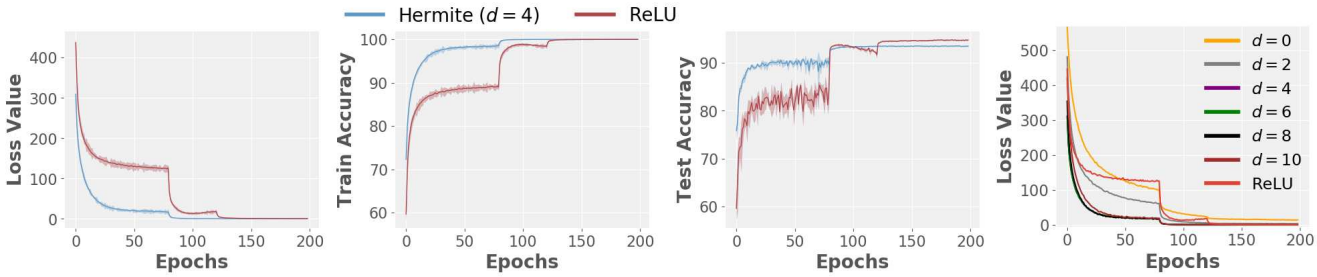


Figure 3: **Hermite Polynomials as Activations in ResNets**. We introduce softsign function to handle the numerical issues from the unbounded nature of Hermite polynomials. W denotes the weight, BN denotes batch normalization, σ is the Hermite activation and SS is the softsign function.



(a) Loss and Accuracy curves

(b) Effects of changing d

Figure 4: **Hermites vs. ReLUs on ResNet18.** (a) Hermites provide faster convergence of train loss and train accuracies than ReLUs. Hermites have faster convergence in test accuracies over the initial epochs but ReLU has the higher test accuracy at the end of training. (b) As we increase the number of hermite polynomials, the speed of loss convergence increases until $d = 6$ and then it starts to reduce. $d \geq 1$ performs better than $d = 0$ where only softsign is used as an activation. The close overlap between $d = 4$ to $d = 10$ suggests that tuning for d is not very expensive.

that are not easy to fix. In fact, [8] explicitly notes that higher-order polynomials tend to make the activations unbounded making the training unstable. Fortunately, a trick mentioned in [1] in the context of quadratic functions, addresses the problem. The solution is to add a softsign function in (3), which has a form similar to tanh however, it approaches its maximal (and minimal) value slower than tanh.

$$\text{Softsign}(x) = \frac{x}{1 + |x|} \tag{3}$$

C) ResNet18. With the above modification in hand, we can use our activation function within Resnet18 using Preactivation Blocks [10, 11]. In the preactivation block, we found that having the second softsign function *after* the weight layer is useful. The slightly modified preactivation block of ResNets is shown in Figure 3. We train ResNets with Hermite activations on CIFAR10 to assess the general behavior of our substitution. We use SGD as an optimizer and follow the data augmentation techniques and the learning rate schedules as in [10, 3]. We perform cross-validation for the hyper-parameters. After training, we obtain the loss curve and the training set accuracies for Hermite activations and ReLUs (see Fig. 4). **Positives:** Figure 4 shows that the loss values and trainset accuracies converge at a much faster rate for Hermites than ReLUs. While the test set accuracy at the *final* epoch is higher for ReLU than Hermites, the test set accuracies for networks using Hermite activations *make much quicker progress in the initial epochs*. These results hold even when varying the learning rates of ReLU networks (see supplement). **Negatives:** We also tune the choice of the number of Hermite polynomials d for $d \in \{0, 2, 4, 6, 8, 10\}$. The setting $d = 0$ is the case where we only use a softsign as an activation without any Hermite activations. Figure 4b shows the results of this experiment. From the plot, we observe a trend similar to the two-layer network above, where the convergence speeds first improves and then reduces as we increase the number of Hermite polynomials. The close overlap of the trend lines between $d = 4$ to $d = 10$ suggest that tuning for d is not very expensive. Hence, in all our experiments we mostly set $d = 4$. The setting $d = 0$ performs worse than when d is at least one, suggesting that the performance benefits are due to Hermite activations with softsign (and not the softsign function on its own).

Interesting take away from experiments? Let us consider the negative results first where we find that a large number of bases is not useful. This makes sense where some flexibility in terms of trainable parameters is useful, but too much flexibility is harmful. Therefore, we simply need to set d to a small (but not too small) constant. On the other hand, the positive results from our simple experiments above suggest that networks with Hermite activations make rapid progress in the early to mid epoch stages – an **early riser** property – and the performance gap becomes small later on. This provides two potential strategies. If desired, we could design a hybrid optimization scheme that exploits this behavior. One difficulty is that initializing a ReLU based network with weights learned for a network with Hermite activations (and retraining) may partly offset the benefits from the quicker progress made in the early epochs. What will be more compelling is to utilize the Hermite activations end to end, but identify scenarios where this "early riser" property is critical and directly influences the final goals or outcomes of the task. It turns out that recent developments in semi-supervised learning satisfy exactly these conditions where leveraging the early riser property is immensely beneficial.

4. Semi-Supervised Learning (SSL)

SSL, Transductive learning and pseudo-labeling. We consider the class of algorithms that solve SSL by generating pseudo-labels for the unlabelled data. These *Pseudo-label (PL) based SSL* methods serve as a way to execute transductive learning [13, 24], where label inference on the **given test dataset** is more important for the user as compared to using the trained model later, on other unseen samples from other data sources. Stand-alone PL based meth-

Algorithm 1

Input: Labeled data (x_i, y_i), unlabeled data (z_i), number of classes k, #-outer (inner) epochs $M_O(M_I)$, loss function $L = L_{CE}(x_i, y_i) + L_{CE}(z_i, y_i) + Reg_E(z_i, y_i)$, learning rates $\eta_w, \eta_P^p, \eta_P^d$. Initial Pseudo-labels for unlabeled data chosen as: $y_i = e_i$ with probability $1/k$ where e_i is the one-hot vector at $i-$th coordinate.

for $O = 0, 1, 2, ..., M_O$ **do**
  Reinitialize the network parameters w^0
  $\Delta P_u = 0$
  for $I = 0, 1, 2, ..., M_I$ **do**
    (Primal) SGD Step on w: $w^{t+1} \leftarrow w^t - \eta_w \nabla L$
    (Primal) SGD Step on ΔP_u: $\Delta P_u \leftarrow \Delta P_u - \eta_P^p \nabla L$
  end for
  (Dual) SGD Step on P_u: $P_u \leftarrow P_u - \eta_P^d \Delta P_u$
end for
Output: Classification model w.

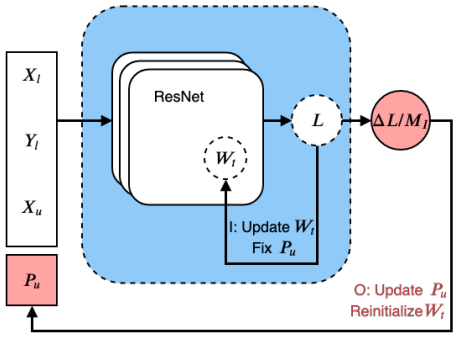


Figure 5: **SaaS framework illustration.** SaaS runs over two loops: inner loop denoted by I and outer loop denoted by O.

ods can be used for generic SSL problems also – where the generation of pseudo-labels are merely a means to an end. The literature suggests that in general they perform competitively, but some specialized SSL methods may provide some marginal benefits. In the other direction, general (non PL based) SSL methods can also be used for obtaining pseudo-labels. But PL based SSL generates pseudo-labels concurrently with (in fact, to facilitate) estimating the parameters of the model. So, if the eventual goal is to obtain labels for unlabeled data at hand, these methods are a good fit – they explicitly assign (potentially) high accurate labels for the unlabeled data at the end of training.

Improved deep classifiers by assigning PL. It is well known that poor (random) labels are harder to fit (train) [30], or equivalently, high quality (pseudo) labels accelerates the training procedure. In other words, highly accurate labels $\rightarrow$ fast training. Interestingly, [5] showed that the converse statement, i.e., fast training $\rightarrow$ accuracy of labels, is *also* true using empirical evidence, during training. SaaS (**Speed as a Supervisor**) is used to *estimate* the pseudo-labels using "speed" as a surrogate. That is, for a classifier

Dataset	# Labeled.	# Unlabeled.	Augmnt.	M_I/ M_O
SVHN	1K	72,257	A + N	5 / 75
CIFAR-10	4K	46,000	A + N	10 / 135
SmallNORB	1K	23,300	None	10 / 135
MNIST	1K	59,000	N	1 / 75

Table 2: **SSL experimental details.** A and N denote the data augmentation techniques, affine transformation and normalization respectively. M_I and M_O denote the inner and outer epochs respectively.

such as ResNet, SaaS takes advantage of the fact that the *loss decreases at a faster rate for correct labels* (compared to random labels) during training. Furthermore, the *rate of loss reduction decreases as the percentage of incorrect labels in the dataset increases*. **The SaaS framework.** There are two phases in SaaS. In the primal phase (inner loop), SaaS seeks to find a set of pseudo-labels that decreases the loss function over a *small number of epochs*, the most. In the dual phase (outer loop), the "pseudo-label" optimization is carried out by computing a posterior distribution over the unlabeled data. Specifically, the algorithm consists of two loops: (i) in the outer loop, we optimize over the posterior distribution of the pseudo-labels, and (ii) in the inner loop, we retrain the network with fixed pseudo-labels. After every inner loop, we reinitialize the network, and compute the rate of change of the loss value with which the posterior can be computed. A flowchart is shown in Figure 5. We can easily use a ResNet/DenseNet model in the primal phase. There are two different loss functions that are optimized during training: the cross-entropy loss (L_{CE}) over the labeled data, the unlabeled data and an entropy regularizer (Reg_E) on the pseudo-labels. A pseudocode is provided in Algorithm 1.

Pseudo-labels with Hermites. Recall that networks with Hermite activations manifest the "early riser" property. This turns out to be ideal in the setting described above and next, we show how this property can be exploited for transductive/semi-supervised learning. Intuitively, the early riser property implies that the training loss decreases at a **much faster rate** in the initial epochs. This is expected, since the optimization landscape, when we use Hermites are, by definition, *smoother* than ReLUs, since *all* the neurons are *always* active during training with probability 1.

Setting up. For our experiments, we used a ResNet-18 architecture (with a preactivation block) [11] architecture to run SaaS [5] with **one crucial difference**: ReLU activations were replaced with Hermite activations. All other hyperparameters needed are provided in Table 2. We will call this version, **Hermite-SaaS**. To make sure that our findings are broadly applicable, we conducted experiments with four datasets commonly used in the semi-supervised learning literature: SVHN, CIFAR10, SmallNORB, and MNIST.

4.1. Faster Convergence.

SaaS tries to identify labels on which the training loss decreases at a faster rate. Our experiments show that the use of

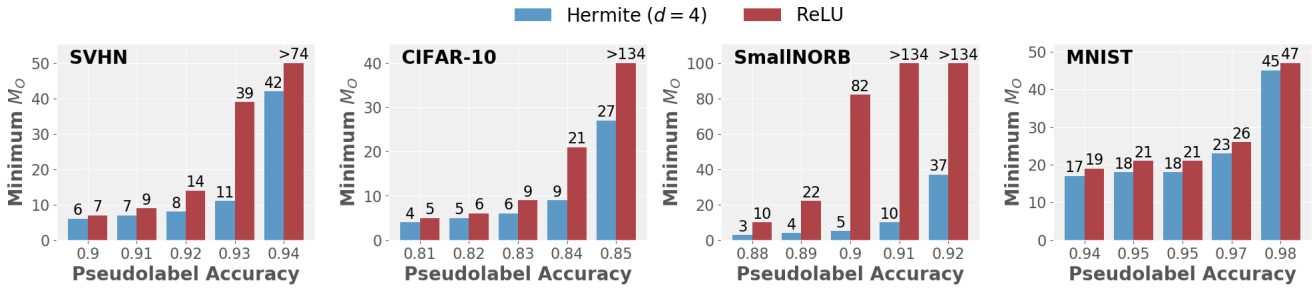


Figure 6: **Hermite-SaaS trains faster.** We plot the number of outer epochs M_O vs. the pseudo-label accuracy across 4 datasets. We consistently observe that the minimum number of outer epochs M_O to reach a given value of pseudo-label accuracy is always lower for Hermite-SaaS than ReLU-SaaS.

Hermites provides models on which the loss function can be optimized easily, thus stabilizing the two phase procedure. Figure 7 shows the result of our experiment on CIFAR10 dataset. Notice that the periodic jumps in the loss value is expected. This is because the big jumps correspond to the termination of an inner epoch, where the pseudo-labels are updated and weights are reinitialized. From Figure 7, we observe that Hermite-SaaS provides a smoother landscape during training, accelerating the training process overall.

4.2. Computational Benefits.

When the accuracy of pseudo-labels is assessed against the actual labels for SVHN dataset, we obtain an error rate of **5.79** over the pseudo-label error rate of 6.22 in [5]. This indicates that the quality of pseudo-labels can be significantly improved using Hermite-SaaS. Moreover, we also find that the number of epochs needed to reach a specific pseudo-label accuracy is also significantly lower. Table 3 shows two common metrics used in classification: (i) "max gap in accuracy" (Max Δ) measures the maximum difference in pseudo-label accuracy over epochs during training; and (ii) "pseudo-label accuracy" (Max PL ACC) measures the maximum pseudo-label accuracy attained during the course of training. Both these measures form a proxy to assess the the quality of pseudo-labels. In addition, Figure 6 shows that the number of epochs needed to reach a specific pseudo-label accuracy is **significantly lower** when using Hermite-SaaS. This behavior for Hermite-SaaS holds over all the four datasets.

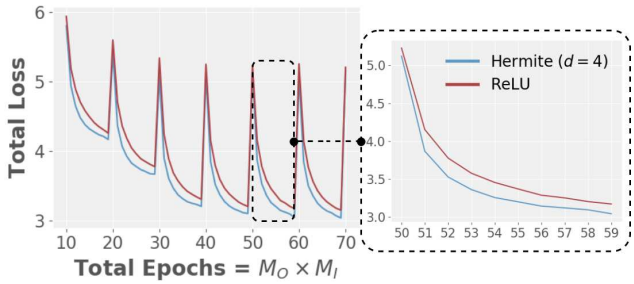


Figure 7: **Convergence of loss functions in SaaS.** Larger spikes correspond to the end of inner loop and are due to weight reinitialization. Right plot shows that Hermites accelerate the training, ensuring high-quality pseudo-labels.

4.3. Financial Savings.

ReLU takes less time per iteration since in our Hermite-SaaS procedure, we must perform a few more matrix-vector multiplications. However, our experimental results indicate that the lower *per iteration* time of ReLU is negligible as compared to the total number of iterations. To provide a better context for the amount of savings possible using Hermite-SaaS, we performed an experiment to assess financial savings. We use AWS p3.16x large cluster for training. We calculate the cost (to train a network) by running the algorithm to reach a minimum level of pseudo-label accuracy, using the default pricing model given by AWS. We can clearly see from Table 3, that we get significant cost savings if we use Hermite-SaaS. Note that we found cases where ReLU-SaaS could not reach the pseudo-label accuracy that is achieved by Hermite-SaaS: in these cases, we report a conservative lower bound for cost savings.

		Max Δ	Max PL ACC	Time/ Epoch (sec)	Total Time (hrs)	Cost ($)	Saved ($)
SVHN	H	6.1%	94.2%	470	5.6	137	71
	R		93.3%	409	8.5	208	
CIFAR10	H	3.4%	85.5%	348	2.7	66	≥ 213
	R		84.4%	304	≥ 11	≥ 280	
Small NORB	H	5.2%	92.6%	47	0.5	12	≥ 13
	R		90.4%	27	≥ 1	≥ 25	
MNIST	H	4.5%	98.2%	94	1.2	29	-11
	R		98.2%	55	0.3	18	

Table 3: **Cost effective and accurate pseudo-labels are generated by Hermite-SaaS.** Expenses when training Hermite-SaaS (H) and ReLU-SaaS (R) on AWS and the performance metrics on pseudo-labels generated. We observe that although Hermite-SaaS takes more time per epoch than ReLU-SaaS, the overall gains are better for Hermite-SaaS.

Method	SVHN 1000
VAT+EntMin [17]	3.86
MT [27]	3.95
TSSDL [24]	3.80
TE [15]	4.42
ReLU-SaaS	3.82
Hermite-SaaS	**3.57 ± 0.04**

Method (SSL with PL)	CIFAR10 500	1000	2000	4000
TSSDL [24]	-	21.13	14.65	10.90
Label Prop. [15]	32.4	22.02	15.66	12.69
ReLU-SaaS	-	-	-	10.94
Hermite-SaaS	**29.25**	**20.77**	**14.28**	**10.65**

Table 4: **Hermite-SaaS generalizes better.** (**left**): Test-set accuracies on SVHN dataset in comparison to the baselines provided in [5]. (**right**):Test-set accuracies on CIFAR10 dataset in comparison to other pseudo-label based SSL methods.

4.4. Better Generalization.

From the work of [30], we know that more accurate labels provide better generalization. With more accurate pseudo-labels from Hermite-SaaS, we expect a standard supervised classifier, trained using the pseudo-labels, to provide better accuracy on an unseen test set. We indeed observe this behavior in our experiments. The left subtable in Table 4 shows the performance of Hermite-SaaS on SVHN datast against popular SSL methods reported in [5]. We also compare against known PL based SSL methods for CIFAR10 dataset in right sub-table of Table 4. We use ResNet18 with a preactivation block for the supervised training phase, although other networks can be used.

4.5. Noise Tolerance.

SSL approaches, especially PL methods, suffer from confirmation bias [27] which is when the network starts to predict incorrect labels with higher confidence and resists change in the course of training. A recent result describes how ReLU networks [12] tend to provide high confidence predictions even when the test data is different from the training data: this can be used in adversarial settings [19]. In other words, [27] and [12] taken together hint at the possibility that ReLUs may, at least, partly contribute to the confirmation bias in PL based SSL methods. While evaluating this question comprehensively is an interesting topic on its own, we present some interesting preliminary results. We can assess the question partly by asking whether Hermites-SaaS is more tolerant to noise than ReLU-SaaS: approached by artificially injecting noise into the labels. In this section, we first evaluate this issue on the theoretical side (for one hidden-layer Hermite networks) where we find that Hermite networks do not give (false) high confidence predictions when the test data are different from the train data. We later demonstrate noise tolerance of Hermite-SaaS.

Accurate confidence of predictions in one-hidden layer hermite networks. In order to prove our main result, we first prove a generic pertubation bound in the following lemma. Consider a 2 layer network with an input layer, hidden layer and an output layer, each with multiple units. Denote $f_k(x)$ and $f_l(x)$ to be two output units with x denoting the input vector. Let w_j be the weight vector between input and the j^{th} and a_{lk} be the weight connecting l^{th} hidden unit to the k^{th} output unit.

Lemma 1. *Consider the output unit of the network, $f_k(x) = \sum_j a_{kj} \sum_{i=0}^{d} c_i h_i(w_j^T x)$, where $c_i's$ are the Hermite coefficients and d is the maximum degree of the herimite polynomial considered. Then,*

$$|f_l(x) - f_k(x)| \leq Cd\alpha\beta$$

Here, C is a constant proportional to the ℓ_∞ norms of c_i's;

$$\alpha = \max_{lk} \sum_j |a_{lj} - a_{kj}| \ ; \beta = \max(\|w\|_p^d \|x\|_q^d, \|w\|_p \|x\|_q),$$

such that $1/p + 1/q = 1$.

Now, we use the perturbation bound from Lemma 1 to show the following result (proof in the supplement) that characterizes the behavior of a network if the test example is "far" from examples seen during training.

Theorem 2. *Let $f_k(x) = \sum_j a_{kj} \sum_{i=0}^{\infty} c_i h_i(w_j^T x)$, be a one-hidden layer network with the sum of infinite series of Hermite polynomials as an activation function. Here, $k = 1, 2, ..., K$ are the different classes. Define $w_J = \min w_j^T x$. Let the data x be mean normalized. If $\epsilon > 0$, the Hermite coefficients $c_i = (-1)^i$ and*

$$\|x\| \geq \frac{1}{\|w_J\|} \log \frac{\alpha}{\log(1 + K\epsilon)}$$

then, we have that the predictions are approximately (uniformly) random. That is,

$$\frac{1}{K} - \epsilon \leq \frac{e^{f_k(x)}}{\sum_{l=1}^{K} e^{f_l(x)}} \leq \frac{1}{K} + \epsilon \ \forall \ k \in \{1, 2..., K\}$$

Proof. Sketch. Note that the form of coefficients is important for this theorem. In particular, we use the exponential generating functions expansion of Hermites and exploit the form of normalization due to the softmax layer to provide a lower bound for the confidence of prediction for an arbitrary class. For this event to occur with high probability, we show that the test data has to be at least a certain distance far away from training data. □

As the data is mean normalized, any test example x with high $\|x\|$ implies that it is far from the training data. For large $\|x_{\text{test}}\|$, this result shows that the predictions are fairly random – a desirable property – clearly, we should not predict confidently if we have not seen such an example earlier.

Hermite-SaaS is more noise tolerant. To further assess if Hermite-SaaS reduces confirmation bias, we experiment by injecting noise in the labels provided for SSL training.

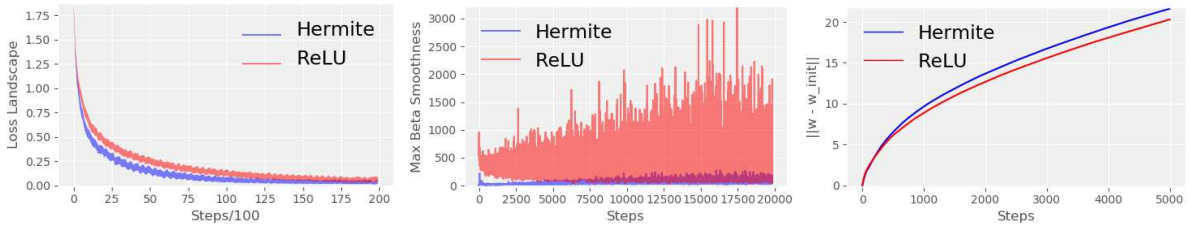


Figure 8: **Hermite generates smoother landscape than ReLU.** (**left**): Magnitude of loss and its variation is lower for Hermites. (**middle**): Gradients are more stable on Hermite loss landscape: lower maximum beta smoothness. (**right**): Hermite networks have a higher effective learning rate .

In particular, we chose the label for a subset of images, uniformly at random. We conduct experiments with 30% label noise levels on CIFAR10. After estimating the pseudo-labels we trained a model in a supervised manner using Resnet18. Our results show that Hermite-SaaS based models obtain similar or a higher test set accuracy of about 80%. This is encouraging, but we also observe faster convergence **(95 epochs)** compared to a ReLU-SaaS model **(at least 600 epochs)**. In other words, Hermite activations based training yields models/estimators with low variance suggesting that they may behave well in the presence of outliers. We also investigate how Hermite activations behave in presence of a standard robust learning method with noisy labels, specifically [26]. We observe that the Hermites version boosts the performance of [26] both in terms of accuracy and rate of convergence (see supplement).

5. Why Hermites provide faster convergence?

We discussed how the noise tolerance properties of Hermites help with faster convergence of Hermite-SaaS. Here, we show how incorporating Hermite activations within a network makes the loss landscape smoother relative to ReLUs. Smoother landscapes implying faster convergence is not controversial [7]. One difference between ReLUs and Hermites is the **nonsmooth** behavior: for ReLU networks, standard first order methods require $O(1/\epsilon^2)$ (versus $O(1/\epsilon)$ iterations for Hermite nets) to find a local minima. This is the reason why is it not enough to just replace Hermites with (a sufficiently large number of) ReLUs even though ReLU networks are universal approximators. We provide three pieces of empirical evidences to support our claim that Hermites provide smoother landscape.

(a) Lower Loss Values. We examine the loss landscape, along the SGD trajectory, following the directions outlined in [23]. The authors there showed that `BN` generates smoother landscapes: in Fig. 8, we show that `BN+Hermites` generate even smoother landscapes implying much faster training. In Fig. 8 (left), we plot training loss $L(w - \eta \nabla L)$ for different η values for ResNet18 architecture. Hermite network generates a better loss landscape (lower magnitude) than ReLU network. **(b) Maximum Beta smoothness.** In Fig. 8 (middle), we show the maximum difference in the ℓ_2 norm of gradients over the distance moved in that direction. Hermite networks have a lower variation of gradient norm change than ReLU networks, indicating that the gradients on Hermite Landscape are more stable implying faster convergence. **(c) Higher effective learning rate.** In Fig. 8 (right), we plot deviation of weights from initialization and observe an increase in this metric with Hermites.

6. Conclusion

In this paper, we studied the viability and potential benefits of using a finite Hermite polynomial bases as activation functions, as a substitute for ReLUs. The lower order Hermite polynomials have nice mathematical properties from the optimization point of view, although little is known in terms of their practical applicability to networks with more than a few layers. We observed from our experiments that simply replacing ReLU with an expansion in terms of Hermite polynomials can yield significant computational benefits, and we demonstrate the utility of this idea in a computationally intensive semi-supervised learning task. Under the assumption that the training is being performed on the cloud (published pricing structure), we show sizable financial savings are possible. On the mathematical side, we also showed that Hermite based networks have nice noise stability properties that appears to be an interesting topic to investigate, from the robustness or adversarial angles. Furthermore, since Hermite-nets avoid over-confident predictions on newer test samples, it would be interesting to investigate the benefits of using Hermite-nets to solve (variants of) meta learning problems.

Acknowledgments

Research supported by NIH R01 AG062336, NSF CAREER award RI#1252725 and American Family Insurance. We thank one of the reviewers for pointing out a promising connection to meta-learning that will be pursued in follow-up work. We are grateful to Glenn Fung for discussions and pointing out a nice paper on smooth SVMs by Yuh-Jye Lee and Olvi Mangasarian [16] which dealt with smoothing the "plus" function. Rest in peace, Olvi.

References

[1] James Bergstra, Guillaume Desjardins, Pascal Lamblin, and Yoshua Bengio. Quadratic polynomials learn better image features. *Technical report, 1337*, 2009. 4

[2] John P Boyd. Asymptotic coefficients of hermite function series. *Journal of Computational Physics*, 54(3):382–410, 1984. 2

[3] François Chollet et al. Keras. https://keras.io, 2015. 4

[4] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015. 1

[5] Safa Cicek, Alhussein Fawzi, and Stefano Soatto. Saas: Speed as a supervisor for semi-supervised learning. In *The European Conference on Computer Vision (ECCV)*, September 2018. 5, 6, 7

[6] Rong Ge, Jason D Lee, and Tengyu Ma. Learning one-hidden-layer neural networks with landscape design. *arXiv preprint arXiv:1711.00501*, 2017. 1, 2, 3

[7] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013. 8

[8] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011. 4

[9] Moritz Hardt and Tengyu Ma. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016. 1

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 4, 5

[12] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. *arXiv preprint arXiv:1812.05720*, 2018. 7

[13] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5070–5079, 2019. 4

[14] Joe Kileel, Matthew Trager, and Joan Bruna. On the expressive power of deep polynomial neural networks. *arXiv preprint arXiv:1905.12207*, 2019. 1, 2

[15] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016. 7

[16] Yuh-Jye Lee and Olvi L Mangasarian. Ssvm: A smooth support vector machine for classification. *Computational optimization and Applications*, 20(1):5–22, 2001. 8

[17] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018. 7

[18] Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, 2005. 2

[19] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015. 7

[20] Tomaso Poggio and Qianli Liao. *Theory II: Landscape of the empirical risk in deep learning*. PhD thesis, Center for Brains, Minds and Machines (CBMM), arXiv, 2017. 2

[21] AI Rasiah, R Togneri, and Y Attikiouzel. Modelling 1-d signals using hermite basis functions. *IEE Proceedings-Vision, Image and Signal Processing*, 144(6):345–354, 1997. 2

[22] Walter Rudin. *Real and complex analysis*. Tata McGraw-Hill Education, 2006. 2

[23] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018. 1, 8

[24] Weiwei Shi, Yihong Gong, Chris Ding, Zhiheng MaXiaoyu Tao, and Nanning Zheng. Transductive semi-supervised deep learning using min-max features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 299–315, 2018. 4, 7

[25] Mahdi Soltanolkotabi, Adel Javanmard, and Jason D Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769, 2018. 1

[26] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018. 8

[27] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017. 7

[28] Luca Venturi, Afonso S Bandeira, and Joan Bruna. Spurious valleys in two-layer neural network optimization landscapes. *arXiv preprint arXiv:1802.06384*, 2018. 1

[29] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 9793–9803, 2018. 1, 3

[30] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016. 1, 5, 7