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

Supplementary Material

Table of Contents

1. Proof of Theorems

Assume that the data is mean normalized (mean 0 and identity covariance matrix). In order to prove our main result, we first prove a generic pertubation bound in the following lemma. We analyze hermite polynomials on a simpler setting, a one-hidden-layer network, to get an intuitive feeling of our argument. Consider a one-hidden-layer network with an input layer, a hidden layer, and an output layer, each with multiple units. Let x denote the input vector, and $f_k(x)$ and $f_l(x)$ to be two output units. Denote w_j to be the weight vector between input and the j^{th} hidden unit, and a_{lk} to be the weight connecting l^{th} hidden unit to the k^{th} output unit. The network can be visualized as in Figure 1.

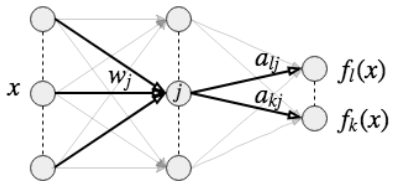


Figure 1: One Hidden Layer Network

In the lemma 1, we bound the distance between two output units $f_k(x)$ and $f_l(x)$. This gives us an upperbound on how any two output units differ as a function of the norm of the (input) data, $\|x\|$.

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 hermite polynomial considered. Then,*

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

Here, C is a constant that depends on the coefficients of the Hermite polynomials and

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

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

Proof.

$$f_l(x) = \sum_j a_{lj} (\sum_i c_i h_i(w_j^T x)) \text{ and}$$
$$f_k(x) = \sum_j a_{kj} (\sum_i c_i h_i(w_j^T x))$$

$$|f_l(x) - f_k(x)|$$
$$= |\sum_j (a_{lj} - a_{kj})(\sum_i c_i h_i(w_j^T x))| \quad (1)$$
$$\leq |\sum_j |(a_{lj} - a_{kj})||(\sum_i c_i h_i(w_j^T x))| \quad (2)$$

h_i is the normalized hermite polynomial ($h_i = \frac{H_i}{\sqrt{i!}}$). $|h_i(z)| = |\frac{H_i(z)}{\sqrt{i!}}| \leq C_1 \frac{|z|^i}{\sqrt{i!}}$, here, C_1 is a function of the coefficients of a given hermite polynomial.

Let's bound $|(\sum_i c_i h_i(w_j^T x))|$,

$$|(\sum_i c_i h_i(w_j^T x))|$$
$$\leq \|c_i\|_p \|h_i\|_q \text{ where } \frac{1}{p} + \frac{1}{q} = 1$$
$$\leq \Big[\sum_i c_i^p\Big]^{1/p} \Big[\sum_i h_i^q\Big]^{1/q}$$
$$\leq d^{1/p} C_2 \left[\sum_{i=0}^{d} \frac{C_1 |w_j^T x|^{iq}}{\sqrt{i!}^q}\right]^{1/q} \forall i, c_i \leq C_2$$
$$\leq d^{1/p} C_2 \Big[\sum_{i=0}^{d} C_1 |w_j^T x|^{iq}\Big]^{1/q}$$
Replace C_1, C_2 with C
$$\leq d^{1/p} C d^{1/q} \max(|w_j^T x|, |w_j^T x|^d)$$
$$\leq Cd \max(|w_j^T x|, |w_j^T x|^d)$$
Using, $J = \text{argmax} \|w_j\|$ and $1/p + 1/q = 1$
$$\leq Cd \max(\|w_J\|_{1/p} \|x\|_{1/q}, \|w_J\|_{1/p}^d \|x\|_{1/q}^d)$$

Substituting the above result back into Equation 2, we get,

$$|f_l(x) - f_k(x)|$$
$$\leq |\sum_j |(a_{lj} - a_{kj})||(\sum_i c_i h_i(w_j^T x))|$$
$$\leq Cd \max(\|w_J\| \|x\|, \|w_J\|^d \|x\|^d) \max_{kl} \sum_j |(a_{lj} - a_{kj})|$$
$$= Cd\alpha\beta$$

Here, $\alpha = \max_{kl} \sum_j |(a_{lj} - a_{kj})|$ and $\beta = \max(\|w_J\| \|x\|, \|w_J\|^d \|x\|^d)$ □

The above lemma can be seen as a perturbation bound since we can simply interpret $f_k = f_l + g_l$ for some other function, (hopefully with nicer properties) g_l. Our main theorem is simply an application of the above lemma to a special case. This allows us to quantify the noise resilience of hermite polynomials. In particular, we show that if the

test data is far from the train data, then hermite polynomials trained deep networks give low confidence predictions. This property allows us to detect outliers during inference, especially in mission critical applications since it allows a human in a loop system [5].

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. Observe that,

$$\frac{e^{f_k(x)}}{\sum_{l=1}^{K} e^{f_l(x)}} = \frac{1}{\sum_{l=1}^{K} e^{f_l(x) - f_l(x)}}$$

Using the fact that $|x| \geq x$ and $-|x| \leq x$, we observe,

$$\begin{aligned} \frac{1}{\sum_{l=1}^{K} e^{|f_l(x) - f_k(x)|}} &\leq \frac{1}{\sum_{l=1}^{K} e^{f_l(x) - f_k(x)}} \\ &\leq \frac{1}{\sum_{l=1}^{K} e^{-|f_l(x) - f_k(x)|}} \end{aligned} \tag{3}$$

Let's bound $|f_l(x) - f_k(x)|$,

$$\begin{aligned} &|f_l(x) - f_k(x)| \\ &= |\sum_j a_{kj}(\sum_i c_i h_i(w_j^T x)) - \sum_j a_{lj}(\sum_i c_i h_i(w_j^T x))| \\ &= |\sum_j (a_{kj} - a_{lj})(\sum_i c_i h_i(w_j^T x))| \\ &= |\sum_j (a_{kj} - a_{lj})(\sum_i \frac{c_i}{(-1)^n} h_i(w_j^T x)(-1)^n)| \end{aligned}$$

From the properties of hermite polynomials,

$$\begin{aligned} e^{xt - t^2/2} &= \sum_{i=0}^{\infty} h_i(x) t^n \\ e^{-x - 1/2} &= \sum_{i=0}^{\infty} h_i(x)(-1)^n \text{ ,when } t = -1 \end{aligned} \tag{4}$$

Choose $c_i = (-1)^i$, then

$$\begin{aligned} \max_j \sum_i (c_i)^i h_i(w_j^T x)(-1)^n &= \max_j e^{-w_j^T x - 1/2} \\ &\leq e^{-\min_j w_j^T x} \end{aligned}$$

Thus,

$$\begin{aligned} &|f_l(x) - f_k(x)| \\ &\quad = |\sum_j (a_{kj} - a_{lj})(\sum_i h_i(w_j^T x)(-1)^n)| \\ &\quad \leq e^{-\min_j w_j^T x} |\sum_j (a_{kj} - a_{lj})| \\ &\quad \leq e^{-\min_j w_j^T x} \alpha \text{ where } \alpha = \max_{kl} |\sum_j (a_{kj} - a_{lj})| \end{aligned}$$

Let $e^{-\min_j w_j^T x} \alpha \leq log(1 + K\epsilon)$, then,

$$\begin{aligned} \implies & \frac{1}{\sum_{l=1}^{K} e^{|f_l(x) - f_k(x)|}} \leq \frac{1}{\sum_{l=1}^{K} e^{f_l(x) - f_k(x)}} \\ & \qquad \leq \frac{1}{\sum_{l=1}^{K} e^{-|f_l(x) - f_k(x)|}} \\ \implies & \frac{1}{K(1 + K\epsilon)} \leq \frac{1}{\sum_{l=1}^{K} e^{f_l(x) - f_k(x)}} \leq \frac{1 + K\epsilon}{K} \\ \implies & \frac{1}{K} - \epsilon \leq \frac{1}{\sum_{l=1}^{K} e^{f_l(x) - f_k(x)}} \leq \frac{1}{K} + \epsilon \end{aligned}$$

Let $J = \text{argmin}_j (w_j^T x)$. The condition, $e^{-w_J^T x} \alpha \leq log(1 + K\epsilon)$,

$$\begin{aligned} &\implies w_J^T x \geq \log \frac{\alpha}{\log(1 + K\epsilon)} \\ &\implies \|w_J\| \|x\| \geq w_J^T x \geq \log \frac{\alpha}{\log(1 + K\epsilon)} \\ &\implies \|x\| \geq \frac{1}{\|w_J\|} \log \frac{\alpha}{\log(1 + K\epsilon)} \end{aligned}$$

□

We interpret the above theorem in the following way: whenever $\|x_{test}\|$ is large, the infinity norm of the function/prediction is approximately $1/K$ where K is the number of classes. This means that the predictions are the same as that of random chance – low confidence. The only caveat of the above theorem is that it requires the all of (infinite) hermite polynomials for the result to hold. However, **all** of our experiments indicate that such a property holds true empirically, as well.

2. Computational Benefits on AWS p2.xlarge

In the paper, we have seen the cost benefits of using Hermite Polynomials on AWS p3.16xlarge instance. In this section, we will examine the performance on AWS p2.xlarge instance as indicated in Table 1. Clearly, as AWS p2.xlarge costs less, the benefits achieved when using hermite polynomials is more significant in the compute time.

		Time per Epoch (sec)	Total Time (hours)	Cost ($)	Time Savings (hours)
SVHN	Hermite	666.8	2.22	2.0	2.6
	ReLU	435.3	4.84	4.35	
CIFAR-10	Hermite	454.2	3.53	3.18	$\geq$ 8.5
	ReLU	320.7	$\geq$ 12	$\geq$ 10.8	
SmallNORB	Hermite	164.8	1.74	1.57	$\geq$ 1.33
	ReLU	81.8	$\geq$ 3.07	$\geq$ 2.76	
MNIST	Hermite	345.4	4.41	3.97	-2
	ReLU	180	2.4	2.16	

Table 1: **Hermite-SaaS on AWSp2.xlarge instance.** Hermite-SaaS saves compute time when compared to ReLU-SaaS

3. Higher number of active units with Hermite Polynomials

There is consensus that ReLUs suffer from the "dying ReLU" problem. Even ignoring pathological cases, it is not uncommon to find that as much as 40% of the activations could "die". A direct consequence of this behavior is that the dead neurons will stop responding to variations in error/input. This makes ReLUs not very suitable for Recurrent Network based architectures such as LSTMs [2]. On the other hand [3] shows that having a certain number of active units, the number determined based on the network architecture is a necessary condition for successful training. We demonstrate here that we meet the necessary condition with a large margin.

Experiment We investigated the number of active units present in a Hermite Network and a ReLU network at the start and the end of training using preactResNet18 model. We determined that the percentage of active units in Hermites are $100\%/99.9\%$ (EPOCH 0/ EPOCH 200) whereas in ReLU are $51\%/45\%$ (EPOCH 0 / EPOCH 200). We also plot the number of active units at the end of training across the layers in Figure 2. It can be seen in Figure 2 that Hermites have twice the number of Active Units than ReLU.

4. Early Riser Property Preserved: ResNets

Setup We used CIFAR10 data in all the experiments in here with random crop and random horizontal flip for data augmentation schemes. With a batch size of 128 for training and SGD as an optimizer, we trained ResNets for 200 epochs. We began with some initial learning rate and then reduce it by 10 at 82^{nd} and 123^{rd} epoch – this is standard practice. We repeated every experiment 4 times in order to control the sources of variance.

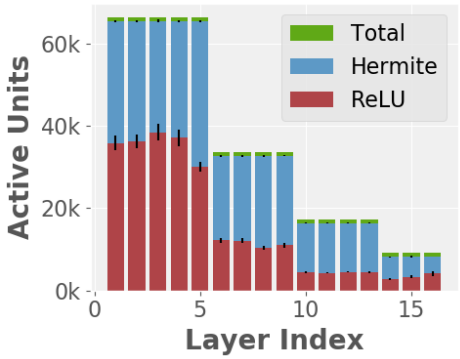


Figure 2: **Hermite networks have more active units.** This figure displays the number of active units present in different layers of ResNet18 network at the end of training. It can be seen that Hermite Nets have close to 100% active units while in ReLU nets half the number of neurons are dead at the end of training.

Experiment The experimental results for ResNet18 and ResNet50 models can be found in Figure 3, Figure 3a and Figure 3b respectively. The results for ResNet150 can be found in Table 2. We observe that the early riser property is preserved accross different learning rates for these ResNet models. There is a small increase in the number of parameters proportional to the number of layers in the network.

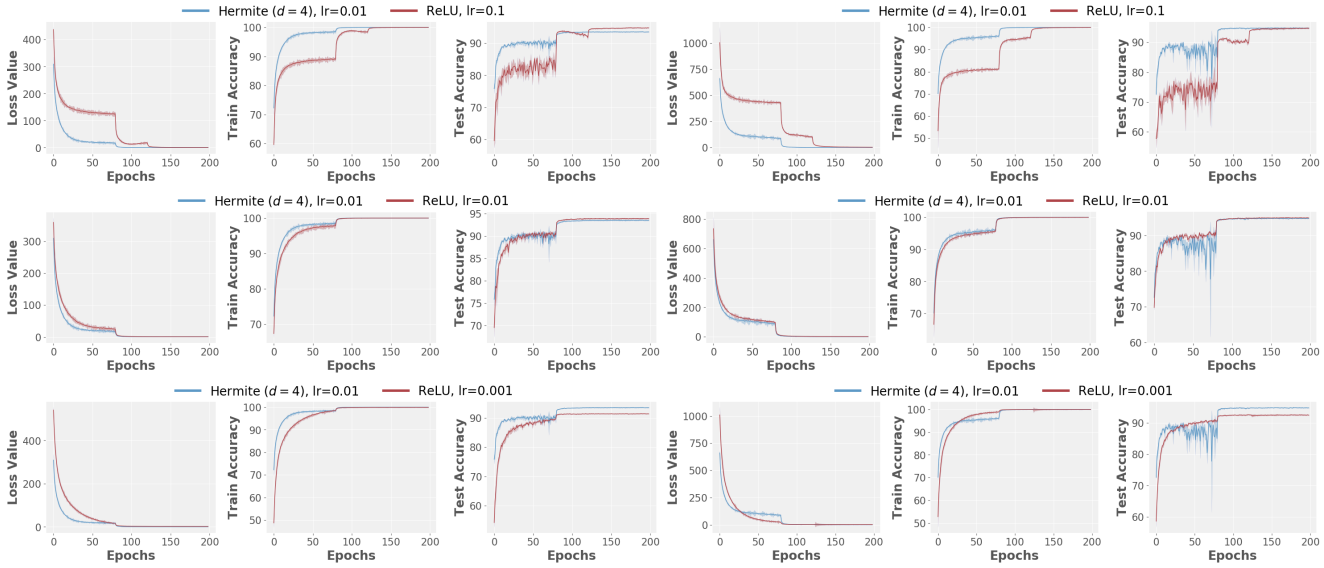
Dataset CIFAR10	Number of Trainable Parameters	Best Test Accuracy	Epochs to reach 90% Test Accuracy
Hermite	58,145,574	95.48%	30
ReLU	58,144,842	94.5%	80

Table 2: **Hermite Polynomials in ResNet152**. We observe small increase in the number of parameters. Test accuracy for the hermite model converges in less than half the number of epochs.

5. Early Riser Property Preserved: DenseNets

Setup All the experiments here were conducted on CIFAR10 dataset with random crop and random horizontal flip schemes for data augmentation. We used the basic block architecture [1] with 40 layers, a growth rate of 12, and a compression rate of 1.0 in the transition stage.

Experiment Figure 4 shows the experimental results of using Hermite Polynomials in a DenseNet. We observe that the early riser property is preserved only in the training loss and the training accuracy curves.



(a) ResNet18 (b) ResNet50

Figure 3: **Early riser property preserved in ResNets** (a) ResNet18 and (b) ResNet50. We observe that a test accuracy of 90% is achieved in approximately half the number of epochs in Hermite-ResNets over ReLU-ResNets on CIFAR10 dataset over different learning rates. The closest that Hermite gets to ReLU is the case when both their learning rates are 0.01 in ResNet50. In this case, we observe that 90% testset accuracy achieved in 20 epochs for Hermite-ResNet50 and 30 epochs for ReLU-ResNet50.

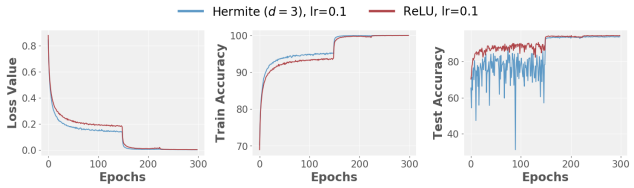


Figure 4: **Early Riser Property in DenseNets.** We observe that the training loss and the training accuracy converge at a faster rate when using Hermite Polynomials in DenseNets.

6. SSL Results on CIFAR10-1K

In the paper, we report the cost benefits on CIFAR10 dataset with 4000 labelled items. Here, we report benefits on Hermite Polynomials on CIFAR10 dataset with 1000 labelled item, another popular experimental setting for CIFAR10 dataset. We find that Hermite-SaaS works equally well for CIFAR10-1K. We achieve a max pseudolabel accuracy of 77.4% compared to 75.6% in ReLU-SaaS. Hermite-SaaS saves cost $>$ \$280 on AWS p3.16xlarge and compute time by 13+ hours on AWS p2.xlarge, compared to ReLU-SaaS when run with $(20/135)$ inner/outer epoch config.

7. Noise injection experiments on Hermites

Setup We repeat the SaaS experiments by injecting 10% and 30% label noise to CIFAR10 dataset. We utilize the method proposed by [4], let's call it *Tanaka* et al., as an optional post-processing step after obtaining pseudo-labels via SaaS. Basically, *Tanaka* et al. performs a "correction" to minimize the influence of noise once the pseudolabels have been estimated. The authors in [4] propose an alternating optimization algorithm where the weights of the network and the labels of the data are updated alternatively. We conduct experiments with 10% and 30% label noise levels on the CIFAR10 dataset. After estimating the pseudolabels and/or using the scheme in [4], we trained a model in a supervised manner using Resnet18.

Experiment Our results summarized in Table 3 show that Hermite-SaaS based models obtain a similar or higher test set accuracy. We also observe that our model converges faster compared to a ReLU-SaaS model. Our experimental results also indicate that post processing techniques (such as [4]) may not always be usefuls to improve the generalization performance of models.

8. Experiments on Shallow Nets

Setup We used a 3-layer network where the hidden layers have 256 nodes each. We used CIFAR10 dataset for this experiment with normalized pixel values and no other data augmentation schemes. The architecture we worked with is thus [3072, 256, 256, 10]. We ran two experiments, one with ReLU activation function and the other with Hermite activations ($d = 5$ polynomials). The loss function was cross-entropy, SGD was the optimization algorithm and we added batch normalization. Learning rate was chosen as

10% Noise		A_{Best}	N_{Best}
SaaS	Hermite	85	224
	ReLU	84	328
SaaS + *Tanaka* et al.	Hermite	85	244
	ReLU	84	284

30% Noise		A_{Best}	N_{Best}
SaaS	Hermite	∼80	95
	ReLU	∼80	≥600
SaaS + *Tanaka* et al.	Hermite	∼80	61
	ReLU	∼80	299

Table 3: **SaaS experiments on Noisy Labelled dataset.** We report the best accuracy (A_{Best}), and number of epochs (N_{Best}) to reach this accuracy. *Tanaka* et al. stands for noisy label processing method proposed in [4].

0.1 and the batch size as 128. Figure 5 shows the loss function and the training accuracy curves that we obtained. It was observed that, when ReLU is just replaced with Hermite activations while maintaining everything else the same, the loss function for hermites converges at a faster rate than ReLU atleast for the earlier epochs.

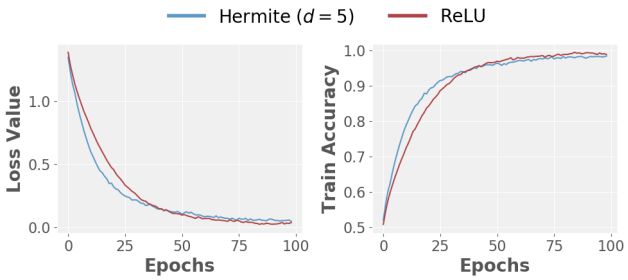


Figure 5: **Experiments on Shallow Nets.** The training loss and training accuracy reduce at a faster rate with Hermite-Polynomials

9. Cloud Services Details

We utilize AWS EC2 On-Demand instances in our research. Specifically, we use p3.16xlarge instance which costs $24.48 per hour and p2.xlarge instance which costs $0.9 per hour.

References

[1] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 4

[2] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015. 4

[3] Yeonjong Shin and George Em Karniadakis. Trainability and data-dependent initialization of over-parameterized relu neural networks. *arXiv preprint arXiv:1907.09696*, 2019. 4

[4] 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. 5, 6

[5] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 3