Towards Feature Sparsity in Large Models: Applications of LassoNet in Image Classification

Pratham Soni
prathams@cs.stanford.edu

Jonathan Tseng
jtseng20@stanford.edu

William Zhang
wxyz@cs.stanford.edu

Abstract

Increasing using of Machine Learning has led to a need for model interpretability, including knowing which input features are most important. In this paper, we build on the LassoNet methodology [5] by applying stochastic gradient updates to test on deeper models and larger datasets (CIFAR-10). We also evaluate LassoNet's ability to serve as a feature selector. We conclude that LassoNet as a standalone model has room for improvement in data intensive workloads but is still useful as a feature selector, with robust performance reducing features up to tenfold.

Category: Computer Vision + General Machine Learning

1 Introduction

With the increasing adoption of Machine Learning systems in industry (especially Deep Learning), a principal issue and cause for concern has been a dearth of tools available to interpret model results. While much of interpretability work has focused on model activations, analyzing causal effects at the output level, it also stands to reason that there is value to achieving sparsity in the feature set itself. By pruning out unnecessary features, we can improve interpretability before even considering the deeper layers!

A recently proposed alternative for deeper models is LassoNet, a solution proposed by Lemhadri et al. [5] that uses Proximal Gradient Descent to sacrifice minimal accuracy in exchange for sparse feature selection. LassoNet is a residual architecture that can embed *any* feedforward architecture and claims to efficiently increase input feature sparsity at minimal cost to test accuracy. Its inputs and outputs correspond to whichever training task its embedded architecture needs: for example, a classification task would take inputs x_i and output labels $y_i \in \{1, 2, 3, ...\}$; for regression tasks, LassoNet would map inputs x_i to continuous $y_i \in [a, b]$.

While the original publication showed promising results, LassoNet has so far only been tested on small datasets, like MNIST, with single-layer models using full batch gradient descent. In our study, we extend LassoNet's architecture to enable the use of stochastic gradient descent and test the performance of deeper architectures on the larger CIFAR-10 dataset.

2 Related Works

Past literature has suggested many methods for enforcing feature sparsity, from the traditional lasso regression (L1 norm), to iterative approximation methods like ISTA [12]. These methods have been used to show some applications, such as image super-resolution, a natural extension of sparse representation, and face recognition [11, 10].

There are several distinct categories of feature selectors: filter, wrapper, and embedded methods [1]. Filter methods rank features and pick the ones that maximizes the difference between classes. This method is computationally efficient, and also avoids overfitting since it does not optimize for model performance [1]. However, these methods may choose redundant features and do not give an indication on how many features should be selected [9]. Wrapper methods use the underlying learning algorithms to evaluate subsets of features based on how well they predict the output [1]. The benefit of this method is that it directly optimizes for model performance. However, evaluating every single subset of features scales exponentially with the number of features, making this method computationally inefficient. Similarly to wrapper methods, embedded methods use the learning algorithm but combine feature selection and learning into a single problem [5]. The lasso method [8] is a commonly used embedded method that varies the strength of L_1 regularization to select features. However, a limitation if this method is that it only works on linear models [5].

3 Data

We use CIFAR-10 [4], a commonly used dataset of 60000 color images of size $32 \times 32 \times 3$, with each image labeled with one of 10 classes. The 10 different classes are: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6000 images per class. We normalize the images by subtracting the mean and dividing by standard deviation of the ImageNet dataset [2] before flattening. We split the images into a 80:10:10 train-validation-test split.

4 Methods

4.1 Baselines

We test LassoNet against the following non-NN baselines: KNN, SVM, and PCA with a linear classifier. We argue that KNN is the ideal complement to the goals of LassoNet, since by design, KNN memorizes each element of the training set, making it the least sparse representation possible.

SVM is also relevant as it's designed to search for an L2-sparse set of support vectors. The SVM solves for support vectors that minimize the regularized hinge loss:

$$\frac{1}{N}\sum_{i}\sum_{j\neq y_i}[\max(0, f(x_i;W)_j - f(x_i;W)_{y_i} + \Delta)] + \lambda\sum_{k}\sum_{l} W_{k,l}^2$$

The first term improves classification accuracy, while the second term regularizes the magnitude of the support vectors to prevent infinite margins for linearly separable data.

Principal component analysis (PCA) is a method of dimensionality reduction commonly employed in machine learning. While related, PCA does not directly induce feature sparsity, but instead finds a reduced set of principal components, each of which is a linear combination of the input features. The principal components are found by finding the u that maximize

$$u^T\left(\frac{1}{n}\sum_{i=1}^{n} x^{(i)}x^{(i)^T}\right)u.$$

This maximizes the variance of the projections, so by projecting each data point onto only the first few principal components, we can obtain lower dimensional data while keeping much of the data's variance.

4.2 Implementation Details

We implement our models in Pytorch [6] and train with Adam [3], using a learning rate of 1e-3 and a batch size of 64. We perform each run with 100 epochs per λ-iteration with an early stopping criterion of five epochs, hyperparameters determined as described in Section 5.1.

4.3 LassoNet

LassoNet approaches feature sparsity with the goal of finding a network with minimal weights in its first layer, zeroing out unimportant input features. The architecture, seen in 1, is a residual network

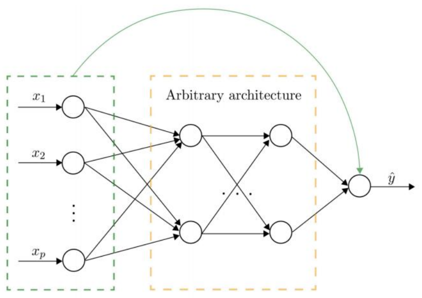

Figure 1: The LassoNet architecture: a single residual connection surrounds an arbitrary feed-forward architecture. By optimizing to make the first hidden layer sparse, features are effectively zeroed out.

$$\mathcal{F} = \{f : f(x) = \theta^T x + f_W(x)\},$$

where f_W is a feedforward network of arbitrary architecture and size with parameters W, and θ represents the residual layer. LassoNet therefore optimizes for the following solution, minimizing the loss of the combined network, while searching for the sparsest possible first layer in W:

$$\arg\min_{\theta,W} L(\theta, W) + \lambda||\theta||_1$$

$$\text{subject to } ||W_j^{(0)}||_\infty \leq M|\theta_j|, j = 1 \ldots d$$

The LassoNet authors note that the selection of M, which controls the sparsifying weight of the linear component of $\mathcal{F}$, is difficult and is hence tuned exhaustively as a hyperparameter [5].

4.4 LassoNet Training

LassoNet searches for a sparse feature set by repeating the following loop:

Algorithm 1: LassoNet training loop

```
Result: Write here the result
while The number of features selected > 0 do
    $\lambda \leftarrow (1+\varepsilon)\lambda$;
    for Each epoch do
        $\theta, W \leftarrow \theta - \alpha\nabla_\theta L, W - \alpha\nabla_W L$;
        Update $\theta, W^{(0)}$ to increase sparsity using proximal gradient descent
    end
end
```

At a high level, LassoNet alternates training the model on the classification task and reducing feature count using “warm starts" to approximate a path from dense to sparse models. By choosing fixed values of λ along a logarithmic scale, and solving for the optimal model at each point, the solution from previous λ can be used as a “warm start" for the next, as the solution to $\lambda(1+\varepsilon)$ is assumed to be close to that of λ.

The original LassoNet publication [5] introduces the *Hierarchical proximal operator*, a computationally efficient method to solve for $\theta, W^{(0)}$ that globally minimize the non-convex constraint. While the exact method is out of the scope of this study, the important takeaway is that the computational cost of the hierarchical operator is negligible compared to the deep learning portion of LassoNet's training loop.

5 Experiments & Results

The original LassoNet implementation only performs full-batch gradient descent, rendering it inoperable for any experiments at scale. We first extend the LassoNet architecture by creating a training pipeline capable of training it with stochastic gradient descent; all further experiments are performed using mini-batches.

5.1 Hyperparameter tuning

We begin by choosing hyperparameters using tests on small, binary classification tasks, such as predicting whether a given MNIST image is one of a 1 or a 4, shown in the below figure. Since the features of images of 1's and 4's are very distinct, it is unsurprising that the vast majority of the features can be reduced away with no change in efficiency.

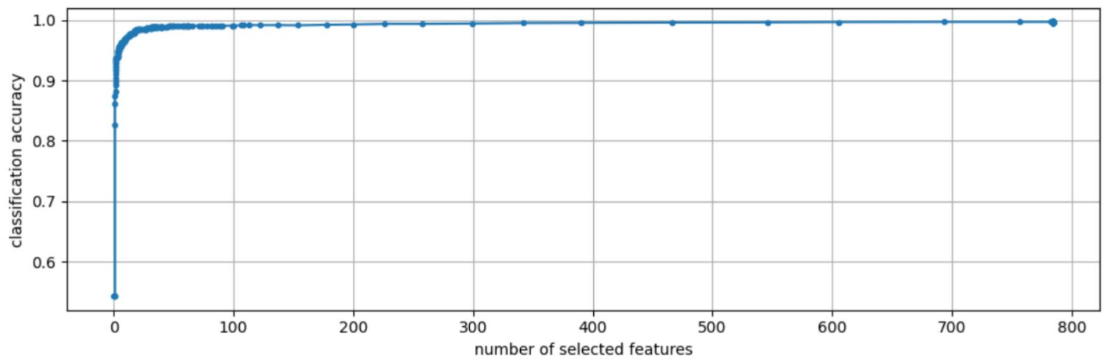

Figure 2: LassoNet results on MNIST plotted against number of selected features.

Starting λ	Epochs
1e-2	12591
1e-1	11167
1e+0	9743
1e+1	7350
1e+2	5130
5e+2	2596

Table 1: Although the original publication claims that much of LassoNet's performance power comes from its "warm starts" design, this comes at a significant training time cost. To enable the testing of LassoNet on much larger datasets, we use the small MNIST dataset to experiment with different starting λ-settings that balance the training time against final performance. Shown are the results of our experiments changing the starting λ hyperparameter.

Higher λ-values cause the network to be initialized with a higher incentive to start eliminating features without as much time to learn the classification task; however, this experiment allowed us to choose an appropriate λ to cut training time in larger experiments by up to 50%. We fix starting λ to 5e+2 for all subsequent experiments and ϵ to 0.3.

5.2 Baselines

We perform KNN with 5-fold cross-validation to determine the optimal k-value, and conclude that a value of $k = 7$ performs best. We similarly cross-validate SVM to determine the optimal L2-regularization strength, and conclude that 3e+4 is the best. Our PCA experiment shows that much of the validation accuracy can be preserved with just 10 principal components, and adding more only marginally increases performance.

Classifier	Features Selected	Accuracy
KNN	3072	30.2%
SVM	3072	35.9%
PCA + Linear	3072	42.7%
LassoNet$_{80,100}$	3072	61.2%
LassoNet$_{80,100}$	1536	25.4%
LassoNet$_{80,100}$	307	18.2%

Table 2: A comparison of the performance of different classifiers. Here, LassoNet$_{M,N...}$ denotes a LassoNet with feedforward architecture using fully-connected layers of size $M, N...$

5.3 LassoNet for Feature Selection

In this section, we evaluate LassoNet's ability to choose optimal features. We compare LassoNet performance against vanilla models trained on the selected feature set as well as performance against randomly selected features at the threshold levels of 1%, 10%, and 100% of the features, choosing the best performing LassoNet model in each category. Results are presented in Figure 3. We note that LassoNet's classification drops significantly with feature count; however, when the vanilla model is retrained on only the selected features, the classification accuracy is significantly higher and somewhat robust against decreasing feature counts, although performance does drop to sub-random levels at extremely low feature counts (1%). LassoNet's feature selection performance with shallow architectures is lacking, but noticeably improves for deeper models. This is strong evidence that LassoNet is an effective feature selector at feature counts up to 10 times smaller than the input.

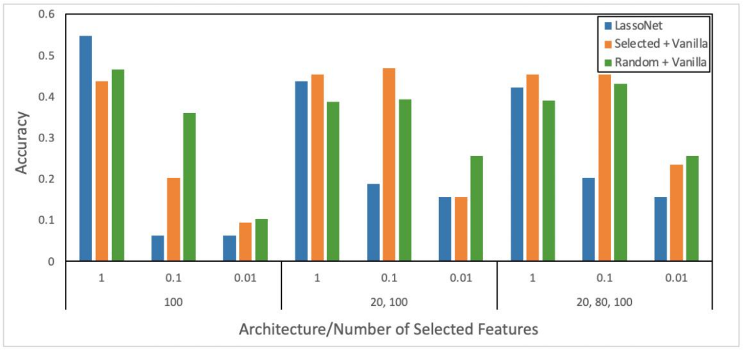

Figure 3: Performance results of LassoNet feature selection. We compare LassoNet accuracies to those of vanilla fully-connected models trained on selected features and random features.

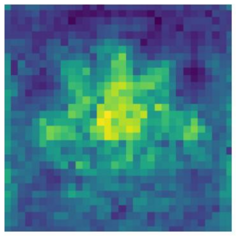

Figure 4: A side effect of the feature-selection process in LassoNet is that the λ iteration at which a feature is eliminated may be used as a measure of its relative importance, i.e. later elimination implies greater selective importance. Figure 4 shows the relative importance of CIFAR-10 features; brighter is more important.

6 Discussion

LassoNet is shown to be an effective feature selector for larger datasets, which, in certain applications, can be powerful for feature interpretability, and is universally useful for dimensionality reduction, and subsequently, performance optimization. Ultimately, the randomly selected feature sets do not significantly underperform compared to the LassoNet-selected feature sets. We hypothesize that this may be due to the nature of the CIFAR-10 dataset: if the features are highly non-prunable, any attempt at feature selection, no matter how well-designed, will result in near-random reductions of features. Since CIFAR-10 data is largely close-cropped photography of many different subjects of varying shapes and forms, it is intuitive that the data would be difficult to prune. It is also worth noting that due to the stochastic nature of our experimentation, these results are noisy and constrained by limited time resources. As LassoNet is an embedded method of feature selection, a feasible weakness is that it is easier to overfit to the training set. However, our use of a large train set mitigates this, as there are not enough parameters to overfit.

7 Conclusion & Future Work

Overall, we have pushed forward the ability to use LassoNet's methodology on more complex data. Further, we establish that LassoNet is suitable as a feature selector for further fine-tuning. One limitation of this work is that while the vanilla models embedded in LassoNet are not large, the LassoNet training regime is iterative and fine-tune the model hundreds of times, which can take days on datasets the size of CIFAR-10. Additionally, since part of our study focused on surveying the performance of LassoNet under different conditions, this meant systematically retraining on many combinations of hyperparameters and architectures. Our study balanced this time consumption by increasing the stochasticity of our training, lowering the iterations per epoch, and limiting the maximum number of epochs per λ-iteration. A future, greater-scope study of LassoNet's capacities may utilize more compute to achieve a more precise view of performance as well as exploring other architecture choices like CNNs.

8 Acknowledgments

This project has been assisted by the gracious mentorship of Ismael Lemhadri and Dr. Rob Tibshirani.

9 Contributions

William conducted baseline experiments with PCA, and initial LassoNet experiments on the MNIST dataset. Jonathan worked on baseline experiments with KNN, SVM, and LassoNet experiments with multilayer networks. Pratham created the dataloaders, implemented a stochastic version of LassoNet, and did finetuning experiments.

References

[1] Girish Chandrashekar and Ferat Sahin. "A survey on feature selection methods". In: *Computers & Electrical Engineering* 40.1 (2014), pp. 16–28.

[2] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: `10.1109/CVPR.2009.5206848`.

[3] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[4] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[5] Ismael Lemhadri et al. *LassoNet: A Neural Network with Feature Sparsity*. 2021. arXiv: `1907.12207 [stat.ML]`.

[6] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[7] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. ISSN: 00359246. URL: `http://www.jstor.org/stable/2346178`.

[8] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[9] Zenglin Xu et al. "Discriminative semi-supervised feature selection via manifold regularization". In: *IEEE Transactions on Neural networks* 21.7 (2010), pp. 1033–1047.

[10] Jian Yang et al. "Sparse Representation Classifier Steered Discriminative Projection With Applications to Face Recognition". In: *IEEE Transactions on Neural Networks and Learning Systems* 24.7 (2013), pp. 1023–1035. DOI: `10.1109/TNNLS.2013.2249088`.

[11] Jianchao Yang et al. "Image Super-Resolution Via Sparse Representation". In: *IEEE Transactions on Image Processing* 19.11 (2010), pp. 2861–2873. DOI: `10.1109/TIP.2010.2050625`.

[12] Zheng Zhang et al. "A Survey of Sparse Representation: Algorithms and Applications". In: *IEEE Access* 3 (2015), pp. 490–530. DOI: `10.1109/ACCESS.2015.2430359`.