Benefit of Group Sparsity in Deep Learning

Kan Chen Zongyu Dai Hanxiang Pan Yue Sheng

May 7, 2020

1 Introduction

Deep learning [LeCun et al., 2015] has achieved huge empirical success in many domains, like computer vision, natural language processing, etc. In different applications, the goal is always similar. We want to train models which can make accurate predictions. In order to train a "good" model, people usually need a large neural network with thousands or even millions of parameters/neurons. Deep neural networks with tons of neurons are capable of representing any functions, i.e., they have incredible expressive powers. However, it is a folklore of machine learning that overparametrized models tend to overfit. In practice, current evidence shows that the majority of neurons in most deep networks are not necessary to their accuracies (e.g. Frankle and Carbin [2019]). This means we can often use a sub network to achieve very close accuracy as the original network. The problem is, we do not know which sub structure we should adopt in practice.

Sparsity has been one of the biggest driving forces in statistics and machine learning in the last 20 years. In many cases, most of the parameters/weights in a model are exactly zero or very close to zero. So we can add some l_1 penalty to the loss function to force the small weights to be zero. The l_1 norm acts as a convex proxy of the non-convex, non-differentiable l_0 norm. In the realm of linear regression, this results in the famous LASSO estimator [Tibshirani, 1996], which has been widely popularized recently due to the interest in compressive sensing [Candes et al., 2006]. In this project, we will use a simple modification of the Lasso penalty, called the "group Lasso" penalty [Yuan and Lin, 2006], to sparsify neural networks. Group Lasso, as its name suggest, can be used to impose sparsity on a group level, such that all the parameters in a group are either simultaneously set to 0, or none of them are. We will also use an additional variation, called the "sparse group Lasso" [Simon et al., 2013], to remove entire neurons at a time.

We will mainly follow the setup in Scardapane et al. [2017], apply some non-deep learning classification algorithms as well as many deep learning algorithms with or without group sparsity on some real datasets. Our goal is to show the benefits of group sparsity in deep learning: significantly reduce the number of parameters while preserve a good accuracy. We will also include some brief discussion on theoretical properties of group sparsity.

2 Related Work

The idea of using group l_1 regularization in machine learning is not new. There are some works on considering convex loss functions [Jenatton et al., 2011], multi-kernel problems [Bach, 2008], and multi-task problems [Liu et al., 2009]. Zhao et al. [2015] used a group sparse penalty to select groups of features co-occurring in a robotic control task. Zhu et al. [2016] used a group sparse formulation to select informative groups of features in a multi-modal context. Baoyuan Liu et al. [2015] applied a similar formulation to the specific case of convolutional networks. Wen et al. [2016] proposed a Structured Sparsity Learning (SSL) method to regularize the structures (i.e., filters, channels, filter shapes, and layer depth) of deep neural networks. More recently, there are some group sparsified deep learning applications in medical research. For example, Xie et al. [2019] developed GDP (Group lasso regularized Deep learning for cancer Prognosis), a computational tool for survival prediction using both clinical and multi-omics data, to harness the rich information in multi-omics data.

3 Methods

3.1 Learning Tasks

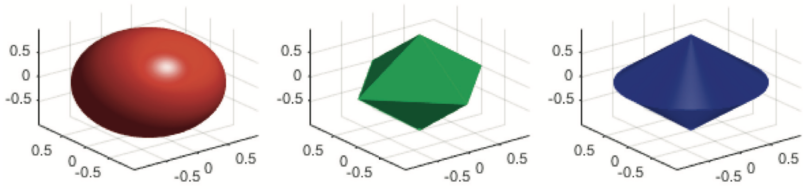
We will consider different classification problems. The first one, the Sensorless drive diagnosis (SDD) dataset, is downloaded from the UCI repository [Dua and Graff, 2017]. In the SDD dataset, the goal is to predict whether a motor has one or more defective components, by using a set of 48 features obtained from the motor's electric drive signals. This dataset has 58508 examples obtained under 11 different operating conditions. In the experiments below, we will use 40956 training samples and 17553 test samples. The second and third datasets are downloaded from the MLData repository. The second dataset is the well-known MNIST dataset, which is composed of 70 thousands 28×28 gray images of the digits 0 to 9 [LeCun and Cortes, 2010]. In the experiments below, we will use 60000 training samples and 10000 test samples. The third dataset is the COVER dataset, the goal of which is to predict the actual cover type of a forest (e.g., ponderosa pine) from a set of 52 features extracted from cartographic data [Blackard and Dean, 1999]. In the experiments below, we will use 406708 training samples and 174304 test samples. The number of classes of these three datasets are $11, 10$, and 7 respectively.

3.2 A Non-deep Learning Baseline

First, we will establish a non-deep learning baseline for all the tasks. We will use logistic regression with or without regularization. We will use l_1 and l_2 penalty. We train 10 epochs and set batch size to be 32 for all the experiments. The optimizer is Adam, and the learning rates are: 1e-3, 1e-4, 1e-5.

3.3 A Deep Learning Baseline

Then, we will also have a deep learning baseline for all the datasets. Here we use a standard three-layer feedforward neural network. It has two hidden layers, both of them have 32 neurons. We train 20 epochs and set batch size to be 32



(a) ℓ_2 norm (b) ℓ_1 norm (c) $\ell_{2,1}$ norm

Figure 1: Unit sphere for three different regularization penalties

for all the experiments. For all the datasets, we will try the following optimizers: Adam, AdaMax, RMSprop, and SGD. We also try the following learning rates: 5e-3, 1e-3, 3e-4, and 1e-4. In this section, we also try the weight decay (l_2 regularization) with regularization parameter: 1e-2, 1e-3, 1e-4, and 1e-5.

3.4 Deep Learning with Group Sparsity

This is the highlight part of our project. Our goal is to study the effect of different regularization methods on the training of neural networks. We are especially interested in the consequence of imposing group sparsity on neural networks. Before introducing our methods, we need first to define the regularization methods.

The most common choice for regularizing the network, thus avoiding overfitting, is to impose a l_2 norm constraint on the weights:

$$R_2(w) = \|w\|_2^2.$$

In the deep learning literature, this is commonly denoted as "weight decay". Inspired by the LASSO method, we can also try l_1 penalty:

$$R_1(w) = \|w\|_1.$$

Both l_2 regularization and l_1 regularization can somehow prevent overfitting, but they are not very efficient for obtaining compact networks. It is easy to see that a neuron can be removed from the architecture only if all its connections have been removed during training. Thus, we need to find another regularization.

We can borrow the idea from the group-level sparsity [Yuan and Lin, 2006], to force all outgoing connections from a single neuron (corresponding to a group) to be either simultaneously zero, or not. Let us define a new penalty for group lasso:

$$R_{2,1}(w) = \sum_{g \in G} \sqrt{|g|} \|g\|_2,$$

where G is a group of weights/parameters, and $|g|$ is the dimensionality of the vector g. For sparse group lasso, we can use the following penalty:

$$R_{SGL}(w) = R_{2,1}(w) + R_1(w),$$

which is simply a combination of lasso penalty and group lasso penalty. Intuitively, we are able to delete many neurons by using the sparse group lasso

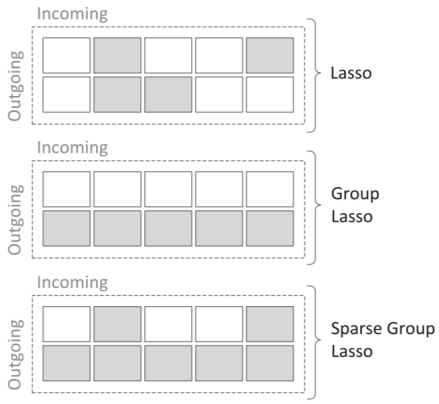


Figure 2: Comparison between lasso, group lasso, and sparse group lasso

penalty. See Figure 1 (which is taken from Scardapane et al. [2017]) for an illustration of l_2, l_1 and group lasso penalty.

Let us suppose the neural network has $H-1$ hidden layers. The dimension of the input vector is L_1 and the output vector has dimension L_{H+1}. For the k-th layer, assume the input is $h_k \in \mathbb{R}^{L_k}$ and the output is $h_{k+1} \in \mathbb{R}^{L_{k+1}}$. They are related by

$$h_{k+1} = \sigma(W_k h_k + b_k),$$

where $W_k \in \mathbb{R}^{L_{k+1} \times L_k}, b_k \in \mathbb{R}^{L_{k+1}}$ are the weight and bias, and $\sigma(\cdot)$ is the non-linear element-wise activation function. We will typically use ReLU. Then in the implementation, we will divide the parameters into three types of group. The first type is the input groups G_{in}. There are L_1 elements in this group. A single element is the vector of all outgoing connections from a neuron in the first/input layer. More concretely, it is some column of the weight matrix W_1. The second type is the hidden groups G_h. In this case, a single element is the vector of all outgoing connections from a neuron in the hidden layers. It is some column of one of the weight matrices $W_2, \ldots, W_H$. The number of hidden groups is the number of neurons in the internal layers. The third type is the bias groups G_b. Here, we treat a single element of $b_i, i = 1, 2, \ldots, H$ as a group. So the number of bias groups equals to the total number of neurons in the network.

Now, we can define the total set of groups as

$$G = G_{in} \cup G_h \cup G_b.$$

A neuron will be removed if and only if its corresponding input/hidden group and bias group are set to be zero simultaneously. See Figure 2 (also taken from Scardapane et al. [2017]) for a comparison between lasso, group lasso, and sparse group lasso, where the white boxes represent non-zero parameters.

We will compare the impacts of L_1, Group Lasso, and Sparse Group Lasso regularizations on the training of neural networks. The architecture we use in this section is the same as the previous section. We still train 20 epochs and set batch size to be 32 for all the experiments. For all the datasets, we will try the following optimizers: Adam, AdaMax, RMSprop, and SGD. We also try the following learning rates: 5e-3, 1e-3, and 1e-4. We will try both group lasso and sparse group lasso with regularization parameter: 1e-3, 1e-4, and 1e-5.

3.5 Some Theory of Group Sparsity and Group Design

We study the sparse learning problem for classification. Denote $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{w})$ a generic deep neural network taking as input a vector $\boldsymbol{x} \in \mathbb{R}^d$ and return a vector $\boldsymbol{y}$ after propagating it through H hidden layers. The vector $\boldsymbol{w}$ is used as a shorthand for the column-vector concatenation of all adaptable parameters of the network. For training the weights of the network, consider a generic training set of N examples given by $\{(\boldsymbol{x}_1, \boldsymbol{d}_1), (\boldsymbol{x}_2, \boldsymbol{d}_2), \cdots, (\boldsymbol{x}_N, \boldsymbol{d}_N)\}$. The network is trained by minimizing a standard regularized cost function:

$$\boldsymbol{w}^* = \arg_{\boldsymbol{w}} \min\{\frac{1}{N}\sum_{i=1}^{N} L(\boldsymbol{d}_i, f(\boldsymbol{x}_i; \boldsymbol{w})) + \lambda R(\boldsymbol{w})\}$$

where $L(\cdot, \cdot)$ is a proper loss function, $R(\cdot)$ is used to impose regularization, and the scalar coefficient $\lambda \in \mathbb{R}^+$ weights the two terms. Define the support of a sparse vector $\boldsymbol{w} \in \mathbb{R}^p$ as

$$\text{supp}(\boldsymbol{w}) = \{j : w_j \neq 0\}$$

and $\|\boldsymbol{w}\|_0 = |\text{supp}(\boldsymbol{w})|$. A natural method for sparse learning is L_0 regularization:

$$\begin{aligned} &\hat{\boldsymbol{w}}_{L^0} = \arg_{\boldsymbol{w}} \min \frac{1}{N}\sum_{i=1}^{N} L(\boldsymbol{d}_i, f(\boldsymbol{x}_i; \boldsymbol{w})) \\ &\text{s.t. } \|\boldsymbol{w}\|_0 \leq k \end{aligned}$$

where k is the sparsity. Since this optimization problem is generally NP-hard, in practice, one often consider the following L_1 regularization problem, which is the standard convex relaxation of L_0:

$$\hat{\boldsymbol{w}}_{L^0} = \arg_{\boldsymbol{w}} \min\{\frac{1}{N}\sum_{i=1}^{N} L(\boldsymbol{d}_i, f(\boldsymbol{x}_i; \boldsymbol{w})) + \lambda\|\boldsymbol{w}\|_1\}$$

where λ is an appropriately chosen regularization parameter. This method is often referred to as Lasso in the statistical literature. In practical applications, one often knows a group structure on the weighted vector $\boldsymbol{w}$ so that variables in the same group tend to be zeros or nonzeros simultaneously. The purpose of this paper is to show that if such a structure exists, then better results can be obtained.

3.5.1 Strong group sparisity

For simplicity, we shall only consider nonoverlapping groups in this paper, although our analysis can be adapted to handle moderately overlapping groups (i.e., each feature is only covered by a constant number of groups, and the resulting analysis depends on this constant). Assume that $\{1, 2, \cdots, p\} = \bigcup_{j=1}^{m} G_j$ is partitioned into m disjoint groups $G_1, G_2, \cdots, G_m : G_i \cap G_j = \emptyset$ when $i \neq j$. Moreover, we denote $k_j = |G_j|$, and $k_0 = \max_{j\in\{1,\cdots,m\}} k_j$. Given $S \subset \{1, \cdots, m\}$ that denotes a set of groups, we define $G_s = \bigcup_{j\in S} G_j$.
Given a subset of variables $F \subset \{1, c \cdots, p\}$ and a weighted vector $\boldsymbol{w} \in \mathbb{R}^p$, let $\boldsymbol{w}_F$ be the vector in $\mathbb{R}^{|F|}$ which is identical to $\boldsymbol{w}$ in F.

The following method, often referred to as group LASSO, has been proposed to take advantage of the group structure:

$$\hat{\boldsymbol{w}}_{L^0} = \arg_{\boldsymbol{w}} \min\{\frac{1}{N}\sum_{i=1}^{N} L(\boldsymbol{d}_i, f(\boldsymbol{x}_i; \boldsymbol{w})) + \sum_{j=1}^{m} \lambda_j \|\boldsymbol{w}_{G_j}\|_2\}$$

The purpose of developing Strong Group Sparisity is to characterizes the condition of group LASSO outperform than traditional LASSO.

Definition 3.1. *A weighted vector* $\bar{\boldsymbol{w}} \in \mathbb{R}^p$ *is* (g, k) *strongly group-sparse if there exists a set* S *of groups such that*

$$supp(\bar{\boldsymbol{w}}) \subset G_S, \qquad |G_S| \leq k, \qquad |S| \leq g$$

This concept is referred to as strong group-sparsity because k is used to measure the sparisity of $\bar{\boldsymbol{w}}$ instead of $\|\bar{\boldsymbol{w}}\|_0$. If the signal is efficiently covered by the groups, then $k/\|\bar{\boldsymbol{w}}\|_0$ should be small, otherwise, it is large. This concept leads us how to design the group information for the group LASSO or Sparse Group LASSO regularization.

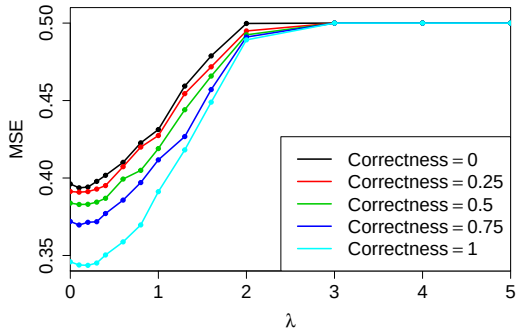


Figure 3: Performance of Group LASSO regularization.Correctness refers to $\|\bar{\boldsymbol{w}}\|_0/k$.

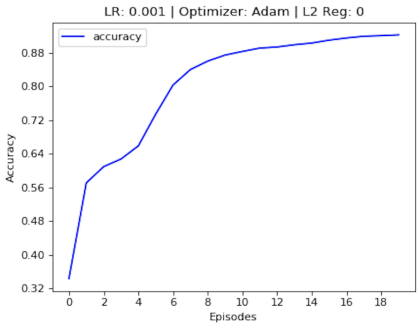
Here in Figure 3, we show the effect of strong group-sparsity on linear regression. Due to the time constraint, we did not try it on neural networks. We hope to study its effect on deep neural networks in the future.

4 Analysis

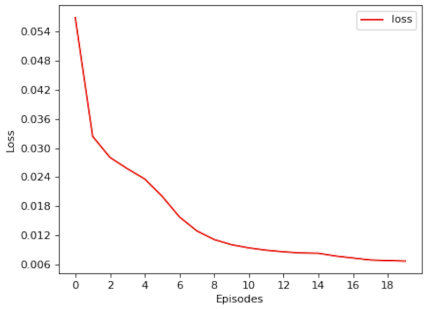
In this section, we will report and analyze the result of the experiments on the above three data sets. We will use test accuracy as the main performance metric. However, for the experiments related to group sparsity, we will also take the number of active neurons into consideration.

4.1 SDD Data Set

The best parameters for logistic regression are: learning rate is 1e-3, l_2 penalty with regularization parameter 1e-3. The optimal test accuracy is 0.6281.

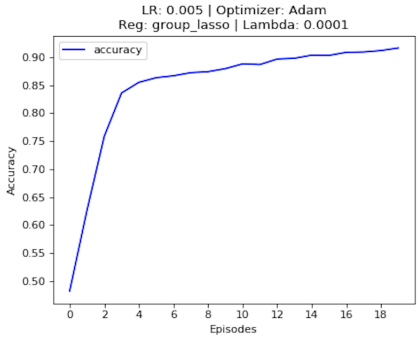


(a) Test accuracy

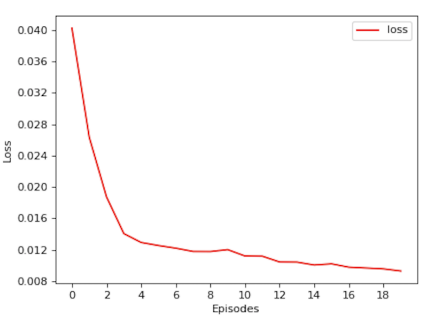


(b) Loss

Figure 4: Loss and test accuracy curves of best deep learning baseline for SDD data set



(a) Test accuracy

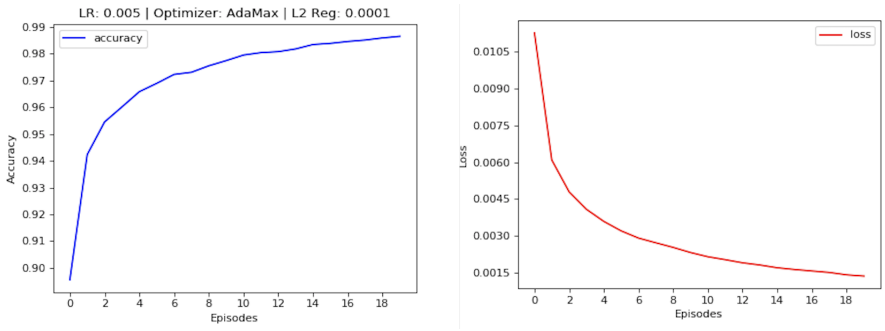


(b) Loss

Figure 5: Loss and test accuracy curves of the most sparse (and accurate) deep learning model for SDD data set

For the deep learning baseline, we find that the optimal hyperparameter combination: optimizer is Adam, learning rate is 1e-3, and there is no weight decay. The optimal test accuracy is 0.92725. See Figure 4 for the loss and accuracy curves.

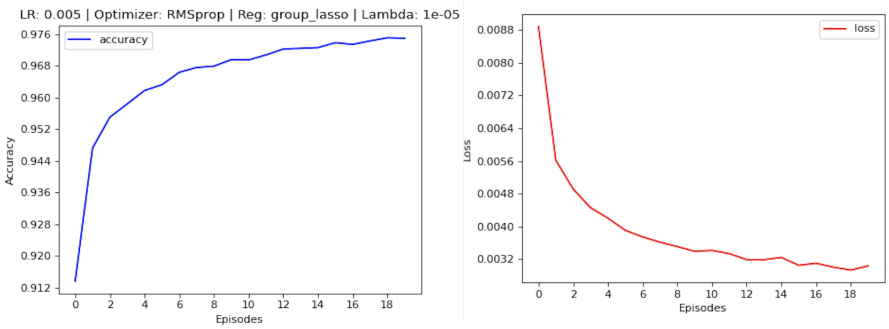
Now we move to investigate the effect of group sparsity. Here, we are not merely looking for models with best test accuracy. What we want is a model with significantly less active neurons, but still has a very high test accuracy. For SDD data set, we find the following setting is pretty favorable: optimizer is Adam, learning rate is 5e-3, the penalty is group lasso with regularization parameter 1e-4. The test accuracy is 0.9372, which is even higher than the deep learning baseline. See Figure 5 for the loss and accuracy curves. Of course, the biggest benefit is here only 1970 out of 2987 neurons, i.e. 66% are active. This confirms our hypothesis and suggests that we can use group sparsity to reduce the size of neural networks.



(a) Test accuracy

(b) Loss

Figure 6: Loss and test accuracy curves of best deep learning baseline for MNIST data set



(a) Test accuracy

(b) Loss

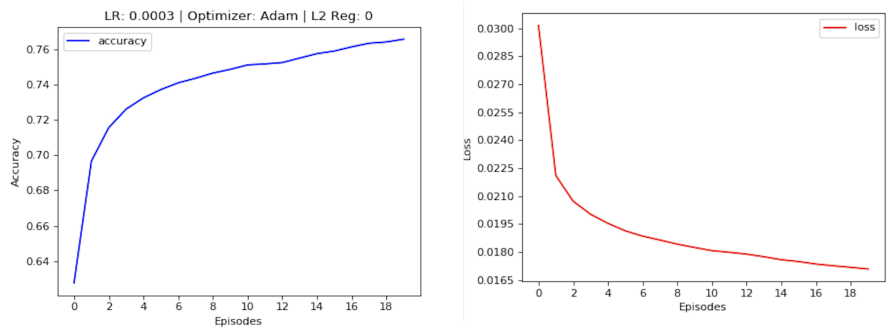
Figure 7: Loss and test accuracy curves of the most sparse (and accurate) deep learning model for MNIST data set

4.2 MNIST Data Set

The best parameters for logistic regression are: learning rate is 1e-4, l_2 penalty with regularization parameter 1e-1. The optimal test accuracy is 0.8953.

For the deep learning baseline, we find that the optimal hyperparameter combination: optimizer is AdaMax, learning rate is 5e-3, and the weight decay parameter is 1e-4. The optimal test accuracy is 0.9723. See Figure 6 for the loss and accuracy curves.

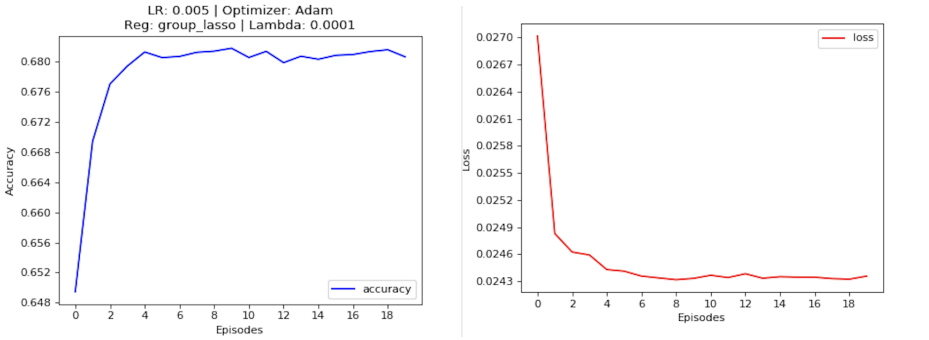
In term of the group sparsity, we find the following setting is pretty favorable: optimizer is RMSprop, learning rate is 5e-3, the penalty is group lasso with regularization parameter 1e-5. The test accuracy is 0.9574, which is only slightly less than the deep learning baseline. See Figure 7 for the loss and accuracy curves. There are 26506 neurons in total, and 16733 neurons (63%) are active. The experiment on MNIST data set also shows that group sparsity regularization can be very efficient for model compression.



(a) Test accuracy

(b) Loss

Figure 8: Loss and test accuracy curves of best deep learning baseline for COVER data set



(a) Test accuracy

(b) Loss

Figure 9: Loss and test accuracy curves of the most sparse (and accurate) deep learning model for COVER data set

4.3 COVER Data Set

The best parameters for logistic regression are: learning rate is 1e-4, l_2 penalty with regularization parameter 1e-1. The optimal test accuracy is 0.6671.

For the deep learning baseline, we find that the optimal hyperparameter combination: optimizer is Adam, learning rate is 3e-4, and no weight decay. The optimal test accuracy is 0.77266. See Figure 8 for the loss and accuracy curves.

In term of the group sparsity, two settings are worth noting: optimizer is Adam, learning rate is 5e-3, the penalty is group lasso with regularization parameter 1e-4 or 1e-5. The test accuracies are 0.6831 and 0.6741, which are still very good comparing to the deep learning baseline. The total number of neurons is 3047, the active neurons are 693 (23%) and 608 (20%). See Figure 9 for the loss and accuracy curves of the case with regularization parameter 1e-4. The experiment on cover data is more exciting. It tells us that, we can use a small sub-network (20% neurons) of the original one to achieve 87% accuracy. This is remarkable and can be potentially very useful in real applications.

5 Discussion

In this project, we mainly focus on the benefit of group sparsity in deep learning. Through our extensive experiments, we conclude that group sparsity can be very efficient in compressing neuron networks, while still have relatively high accuracy. This is an important technique, which may have huge impact in practice.

On the other hand, although we can see the empirical success of group sparsity in deep learning, there are not many theoretical guarantees. The theory of lasso and its variants (e.g. group lasso, sparse group lasso) has been well studied in many statistics and machine learning literature [Yuan and Lin, 2006, Bach, 2008]. More work need to be done to understand the role of group sparsity in deep learning. This can be an interesting future research direction.

Another direction could be to systematically compare different neuron network compression/pruning methods, either empirically or theoretically.

References

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521: 436–444, 2015.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

E. J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241: 81 – 89, 2017.

Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research*, 12(84):2777–2824, 2011.

Francis Bach. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

Jun Liu, Shuiwang Ji, and Jieping Ye. Multi-task feature learning via efficient l2, 1-norm minimization. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, page 339–348, 2009.

L. Zhao, Q. Hu, and W. Wang. Heterogeneous feature selection with multi-modal deep neural networks and sparse group lasso. *IEEE Transactions on Multimedia*, 17(11):1936–1948, 2015.

Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, page 3697–3703, 2016.

Baoyuan Liu, Min Wang, H. Foroosh, M. Tappen, and M. Penksy. Sparse convolutional neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 806–814, 2015.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems 29*, pages 2074–2082. Curran Associates Inc., 2016.

Gangcai Xie, Chengliang Dong, Yinfei Kong, Jiang F. Zhong, Mingyao Li, and Kai Wang. Group lasso regularized deep learning for cancer prognosis from multi-omics and clinical features. *Genes*, 10(3):240, 2019.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Jock A. Blackard and Denis J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24(3):131 – 151, 1999.