Deep Network Shrinkage applied to Cross-Spectrum Face Recognition

Christopher Reale[12], Hyungtae Lee[13], Heesung Kwon[1], and Rama Chellappa[2]
[1] U.S. Army Research Laboratory, Adelphi, MD, USA
[2] Center for Automation Research, UMIACS, University of Maryland, College Park, MD, USA
[3] Booz Allen Hamilton Inc., McLean, VA, USA

***Abstract*— In recent years, deep learning has emerged as a dominant methodology in virtually all machine learning problems. While it has been shown to produce state-of-the-art results for a variety of applicatons (including face recognition and heterogeneous face recognition), one aspect of deep networks that has not been extensively researched is how to determine the optimal network structure. This problem is generally solved by ad hoc methods. In this work we address a subproblem of this task: determining the breadth (number of nodes) of each layer. We show how to use group-sparsity-inducing regularization to effectively replace these hyper-parameters with a single hyper-parameter which can be determined by cross-validation. We demonstrate our method by using it to reduce the size of networks on two commonly used NIR face datasets.**

I. Introduction

Virtually all machine learning methods entail choosing a model and using a training algorithm to learn model parameters from data. In most cases, one must set the values of one or more hyper-parameters for the model and/or algorithm. Deep networks are no different. When there are only a few hyper-parameters and training is relatively fast, cross-validation can be used to select optimal values for the hyper-parameters. If cross-validation is too slow, ad hoc methods must be used instead. Deep learning methods fall into the latter category as they have several hyper-parameters for both the model (layer types, nodes per layer) and training algorithm (learning rate, learning policy, weight decay, momentum, etc.) and generally take a significant amount of time to train.

In this work, we provide a method to reduce the complexity of hyper-parameter selection in deep networks. More specifically, we show how to use group lasso regularization [14] to select the appropriate breadth (number of hidden nodes) for each convolutional and fully connected layer of the network. The group lasso penalty encourages parameters to be group sparse (i.e. blocks of the parameters are zeroed). Thus, by incorporating it into the standard backprop [6] stochastic gradient descent and organizing the parameters into blocks based on the hidden node they compute, the optimization will automatically zero out unnecessary hidden nodes. As shown in Figure 2, we can then simply remove the zeroed out nodes to obtain a smaller network.

Choosing the appropriate number of nodes for each layer is important. having too few nodes can overly simplify the model, restricting it from capturing the full complexity of the task at hand. This can cause the network to perform sub-optimally. On the other hand, unnecessary nodes require extra computational power to calculate and necessitate more space in memory to store both the extra model parameters and the extra features for each sample. Additionally, having too many nodes yields an overly complex model that can be more prone to over-fitting. This is of particular importance when fine-tuning with limited data as in the problem we focus on: NIR Heterogeneous Face Recognition.

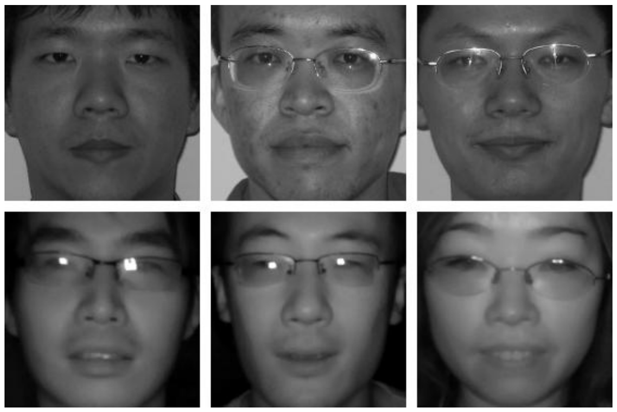
Fig. 1. Aligned and cropped NIR-VIS 2.0 face images. The top row is visible-light and the bottom row is near-infrared.

II. Related Work

A. Deep Learning Parameter Selection

Despite the ubiquity of deep neural networks and the importance of this problem, until recently it had not been addressed beyond an ad hoc approach. Zhou et al. [21] and Scardapane et al. [17] take similar approaches to ours to prune deep networks trained for image classification tasks. Some methods have attempted to solve the related problem of making deep networks smaller. Liu et al. [12] sparsely decompose the filters of convolutional neural networks in order to reduce the redundancy of their parameters. Yang et al. [18] use kernel methods in place of fully connected layers to greatly reduce the necessary number of parameters. Moczulski et al. [15] introduce a new module consisting of diagonal matrices and the discrete cosine transform to reduce the number of parameters in networks while maintaining good performance. Lebedev and Lempitsky [7] use a method similar to ours to speed up convolution by sparsifying the footprints of convolution filters. Our method can also be used

978-1-5090-4023-0/17 $31.00 © 2017 IEEE
DOI 10.1109/FG.2017.128

shrink/speed up deep networks, though our primary objective is to determine the optimal network structure.

B. NIR Heterogeneous Face Recognition

HFR has been extensively researched over the past decade, with NIR being one of the most researched alternative sensing modalities. Here we go over some work that has focused on this problem. Klare and Jain [5] use kernel similarities to a set of training subjects as features. Zhu et al. [22] use a new feature descriptor and a transductive model for domain-adaptive matching. Yi et al. [19] use restricted Boltzmann machines to reduce the domain difference locally and show how to remove initial PCA coefficients to do the same globally. Jin et al. [2], [3] learn local features to represent images consistenly across domains and discriminatively within each domain. Juefei-Xu et al. [4] take a synthesis approach and use cross spectral joint dictionaries to reconstruct images in one domain from those in another. They can then compare images directly.

Reale et al. [16] were the first to adapt deep convolutional neural networks for HFR. They initially train a network to perform visible face recognition, and then fine-tune two versions of the network (one for each domain) for HFR. We improve upon their method by shrinking the initialization network to have a more optimal numbers of nodes in each layer.

III. Our Method

A. Formulation

Deep networks are trained by using stochastic gradient descent to solve the following optimization problem,

$$\min_{W} \frac{1}{M} \sum_{i=1}^{M} L(x_i, y_i, W) + \lambda R(W),$$

where W represents the model parameters, M is the number of training samples, x_i is a training sample, y_i is label information pertaining to x_i, L is a loss function, R is a regularization function, and λ is a constant referred to as the weight decay. Let W be the set of parameter matrices $\mathbf{W}^{(i)} \in \mathbb{R}^{N_i \times K_i}$ where N_i is the number of hidden nodes in layer i and K_i is number of parameters required per node for layer i. Traditionally, the squared Euclidean norm is used for the regularization function R. In this case, R can be written as follows,

$$R(W) = \sum_{\mathbf{W}^{(i)} \in W} \frac{1}{2} \|\mathbf{W}^{(i)}\|_F^2 = \sum_{\mathbf{W}^{(i)} \in W} \sum_{j=1}^{N_i} \frac{1}{2} \|\mathbf{w}_j^{(i)}\|_2^2,$$

where $\mathbf{w}_j^{(i)} \in \mathbb{R}^{K_i}$ is the jth row of $\mathbf{W}^{(i)}$.

While the squared Euclidean norm generally works well for optimizing a given network, we want to encourage our training algorithm to use fewer nodes than it begins with. To do this, instead of using the Euclidean norm, we propose to use the group lasso penalty (sometimes called the mixed ℓ_1/ℓ_2 norm) as follows,

$$R(W) = \sum_{\mathbf{W}^{(i)} \in W} 2\|\mathbf{W}^{(i)}\|_{\ell_1/\ell_2} = \sum_{\mathbf{W}^{(i)} \in W} \sum_{j=1}^{N_i} 2\|\mathbf{w}_j^{(i)}\|_2.$$

The group lasso penalty encourages the solutions to be group sparse (i.e. entire blocks of parameters are set to zero). In our case we have organized the parameters so that each block corresponds to a single hidden node. Thus, training with the group lasso penalty will cause the parameters entire nodes to be zeroed out, after which they can simply be removed from the network.

Our learning algorithm can be summarized as follows. We train a given network using group lasso regularization. We then remove any zeroed hidden nodes. Finally, we continue training the altered network using squared Euclidean norm regularization to tweak the network and boost performance.

B. Explanation

The optimization is set up as a balancing act between reducing the loss function and reducing the number of nodes with the weight decay determining how far that balance should skew towards one side or another. The parameters of nodes that do not contribute much to keeping the loss function low have relatively small loss function gradients. Over time, the regularization function gradient dominates for these parameters and they eventually go to zero. On the other hand, the parameters of nodes which greatly contribute to maintaining a small loss have much larger loss function gradients. Over the course of the optimization, the loss function gradients of these parameters dominate and they will not be set to zero.

It is informative to compare the gradients of the squared Euclidean and the group lasso regularization functions to gain a more detailed understanding of our method. In the squared Euclidean case, the gradient for $\mathbf{w}_j^{(i)}$ (the parameters of a given node) is calculated as follows,

$$\nabla \frac{1}{2} \|\mathbf{w}_j^{(i)}\|_2^2 = \mathbf{w}_j^{(i)}.$$

The squared Euclidean gradient effectively reduces each parameter by a fixed percentage (hence the term weight decay). Without the loss function gradients, the optimization will simply step towards $\mathbf{0}$ a fixed fraction of the distance. This has the effect of ensuring no parameter gets too big while also offering very little incentive to shrink already small parameters.

The group lasso gradients are calculated as follows,

$$\nabla 2\|\mathbf{w}_j^{(i)}\|_2 = \frac{\mathbf{w}_j^{(i)}}{\|\mathbf{w}_j^{(i)}\|_2}.$$

As with the squared Euclidean gradient, the group lasso gradient steps towards $\mathbf{0}$. The difference is in the magnitude of that step. The squared Euclidean regularization mandates stepping a fixed percentage of the distance, whereas the group lasso simply steps a fixed amount regardless of the distance to $\mathbf{0}$. In the Euclidean case, as parameters of less

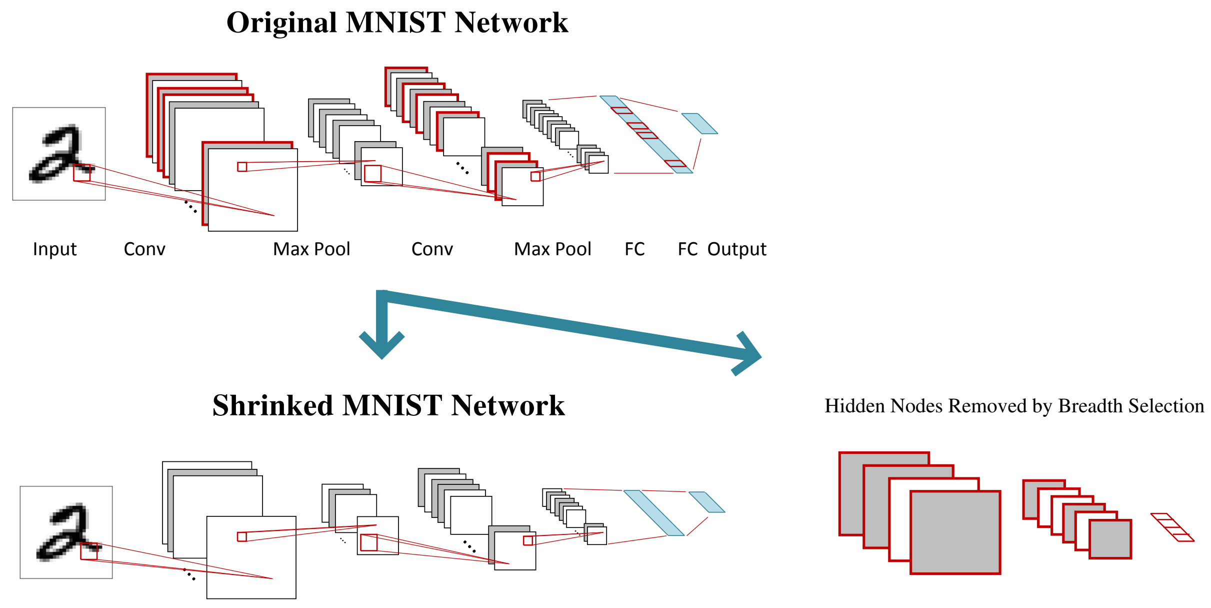

Fig. 2. A visualization of our method.

important nodes shrink, the regularization affects them less and so they stop shrinking before they get to **0**. On the other hand, group lasso always effects all filters the same amount, so less important filters will continue shrinking until they are zeroed out. A side-effect of the the group lasso penalty is that nodes with large parameters are not reined in as much as they would be in the Euclidean case. This leads to some filters with larger parameters. To counteract this, we use both the group lasso and squared euclidean norms for regularization.

C. *Gradient Existence*

The gradient of the group lasso penalty does not exist when the norm of a block is zero. This can be problematic as the goal of our method is to ensure some blocks have that exact property. We found it easiest to avoid this problem by adding a small constant ϵ to all norm calculations as follows,

$$\nabla 2\|\mathbf{w}_j^{(i)}\|_2 \approx \frac{\mathbf{w}_j^{(i)}}{\|\mathbf{w}_j^{(i)}\|_2 + \epsilon}.$$

For small ϵ, this leaves the gradient virtually unchanged most of the time (when the norm is not close to zero) and ensures it is defined at $\mathbf{w}_j^{(i)} = \mathbf{0}$.

IV. Experiments

As a sanity check, we first test our method on the MNIST handwritten digit classification dataset. We then use it to shrink a visible face recognition network, which in turn we use as an initialization network for Near-infrared Heterogeneous Face Recognition.

A. *MNIST*

The MNIST handwritten digit recognition dataset [8] consists of 70,000 (60,000 for training, 10,000 for testing) 28x28 pixel images labeled based on the numerical digit (0-9) they depict. We use the network structure and solver parameters from the mnist example in the Caffe deep learning framework. The network structure is shown in Table I.

We run experiments with a range of ten different weight decays (log space between .0001 and .004). We take results of these experiments and produce plots showing the trade-off between classification rate and resource usage (shown in Figures 3 and 4).

Figure 3 shows that the number of parameters can be reduced by about two thirds with very little change in performance. Figure 4 shows a slightly different picture, where the performance tails off a bit more quickly with less computational power (measured in multiplication operations required to classify a single image). This is because there are significantly more unnecessary nodes in the fully connected layer of the original network compared to the convolution layers and fully connected layers use more parameters but require less computation than convolution layers. The breakdown of the extent to which each layer shrinks is shown in Figure 5 where a larger fraction of convolution nodes are retained compared to fully connected nodes. The main take-away from this experiment is that if a network has too many nodes in a layer, Group Lasso regularization can automatically identify and shrink the layer without negatively affecting performance.

TABLE II

FACE RECOGNITION NETWORK LAYER DETAILS

	Name	Type	Filter Size	Stride	Output Size	Params
Section 1	conv11	Convolution	$3 \times 3 \times 32$	1	$100 \times 100 \times 32$	288
	relu11	ReLU			$100 \times 100 \times 32$	0
	conv12	Convolution	$3 \times 3 \times 64$	1	$100 \times 100 \times 64$	18.4K
	relu12	ReLU			$100 \times 100 \times 64$	0
	pool1	Max Pooling	2×2	2	$50 \times 50 \times 64$	0
Section 2	conv21	Convolution	$3 \times 3 \times 64$	1	$50 \times 50 \times 64$	36.7K
	relu21	ReLU			$50 \times 50 \times 64$	0
	conv22	Convolution	$3 \times 3 \times 128$	1	$50 \times 50 \times 128$	73.7K
	relu22	ReLU			$50 \times 50 \times 128$	0
	pool2	Max Pooling	2×2	2	$25 \times 25 \times 128$	0
Section 3	conv31	Convolution	$3 \times 3 \times 96$	1	$25 \times 25 \times 128$	111K
	relu31	ReLU			$25 \times 25 \times 128$	0
	conv32	Convolution	$3 \times 3 \times 192$	1	$25 \times 25 \times 192$	166K
	relu32	ReLU			$25 \times 25 \times 192$	0
	pool3	Max Pooling	2×2	2	$13 \times 13 \times 192$	0
Section 4	conv41	Convolution	$3 \times 3 \times 128$	1	$13 \times 13 \times 128$	221K
	relu41	ReLU			$13 \times 13 \times 128$	0
	conv42	Convolution	$3 \times 3 \times 256$	1	$13 \times 13 \times 256$	295K
	relu42	ReLU			$13 \times 13 \times 256$	0
	pool4	Max Pooling	2×2	2	$7 \times 7 \times 256$	0
Section 5	conv51	Convolution	$3 \times 3 \times 160$	1	$7 \times 7 \times 160$	369K
	relu51	ReLU			$7 \times 7 \times 160$	0
	conv52	Convolution	$3 \times 3 \times 320$	1	$7 \times 7 \times 320$	461K
	relu52	ReLU			$7 \times 7 \times 320$	0
	pool5	Avg Pooling	7×7	1	$1 \times 1 \times 320$	0
	fc6	Fully Connected			10575	3.38M
	cost	Softmax			10575	0

TABLE III

FACE RECOGNITION LAYER-WISE SHRINKAGE

Weight Decay	Nodes Remaining After Shrinkage									
	conv11	conv12	conv21	conv22	conv31	conv32	conv41	conv42	conv51	conv52
0 (Initial Network)	32	64	64	128	96	192	128	256	160	320
0.0005	17	34	60	128	94	192	106	256	132	320
0.0010	8	17	35	115	94	192	106	256	123	320
0.0015	7	14	22	71	92	192	105	255	119	320
0.0020	7	13	19	51	83	192	105	233	116	320

TABLE I

CAFFE MNIST NETWORK

Layer Type	Dimensions	Stride
Conv	5×5×20	1
Max Pool	2×2	2
Conv	5×5×50	1
Max Pool	2×2	2
Fully Connected	500	-
ReLU	-	-
Fully Connected	10	-
Softmax	10	-

B. *Face Recognition Initialization*

While our ultimate goal is to improve the performance of deep heterogeneous face recognition [16], we accomplish this by first shrinking the initialization network provided to the algorithm. This removes unnecessary filters, making the network less prone to overfitting.

As in [16], we first train the deep convolutional neural network shown in TABLE II to perform face recognition on the CASIA WebFace Database [20] for 200,000 iterations using stochastic gradient descent. The CASIA WebFace dataset is a visible face dataset containing 494,414 images of 10,575 subjects. We then further train it for another 250,000 iterations with an added group lasso weight decay. We do this for five weight decay values (including zero). All images are preprocessed by subtracting the mean face image.

Before using this network for Cross-Spectrum Face recognition, we first examine the properties of the trained and shrunk networks. TABLE III shows which layers shrank and which did not. It is interesting that the layers that compute lower-level features (i.e. towards the bottom of the network) seem to be more susceptible to being shrunk. We believe due to the fact that high-level features are significantly more complicated than low-level features. High-level features must be able to represent a large variation of appearances (e.g. different types of eyes, noses, mouths, facial hair, etc.) whereas low-level features represent a relatively smaller range (e.g. edges, corners, simple-textures). Thus there are many more possible appearances for higher-level features to recognize. This is despite the fact that the lower level layers have fewer nodes to begin with.

To ensure that the shrinkage does not remove too many

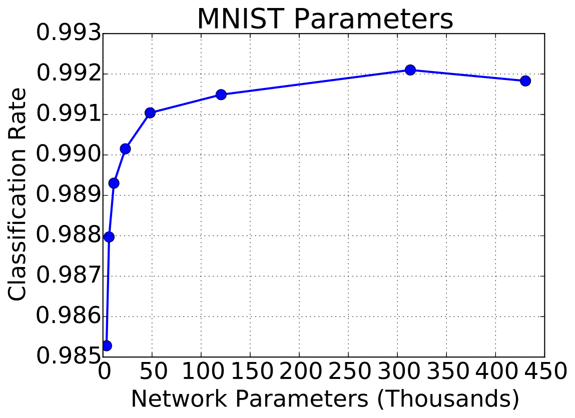

Fig. 3. Trade-off between the number of network parameters and classification rate on the MNIST dataset. Note the number of parameters in the original network is 430,500.

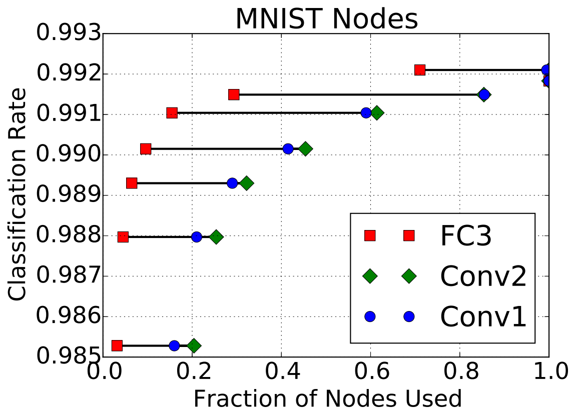

Fig. 5. Trade-off between the number of nodes in each layer and classification rate on the MNIST dataset. Black lines connect data points collected from the same network. Note that the initial number of nodes for the FC3, Conv2, and Conv1 layers are 500, 50, and 20 respectively.

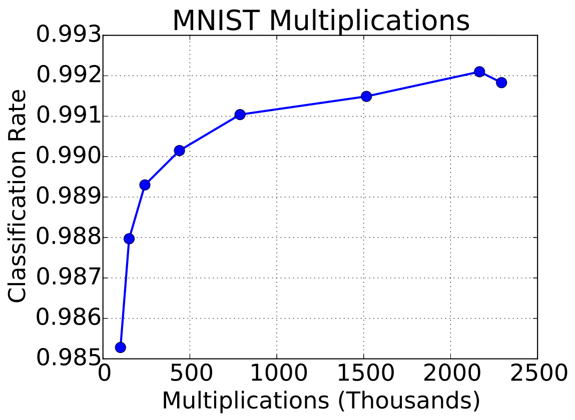

Fig. 4. Trade-off between the number of multiplications per image and classification rate on the MNIST dataset. Note the number of multiplications required in the original network is 2,293,000.

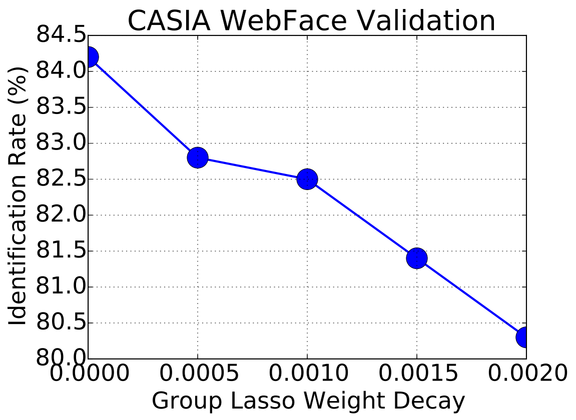

Fig. 6. Trade-off between the recognition rate of the initial network on the validation set and weight decay on the CASIA WebFace dataset.

nodes from the networks, we examine their performance on a visible face recognition task. Before training the networks, we removed a 10,000 image validation set from the CASIA WebFace dataset. Figure 6 shows the ability of the networks to classify the validation set images for varying weight decays. It is clear that even the smallest non-zero weight decay value (.0005) removes enough nodes to hamper recognition. This is most likely due to the large number of subjects in the dataset (10,575), which necessitates fine-grained classification and thus a complicated network. As we will show in Section IV-C, this is not an issue for HFR because the fine-grained classification problem is dwarfed by the cross-domain gap. Also, having a smaller network helps to reduce overfitting in the data-constrained HFR scenario.

C. NIR Heterogeneous Face Recognition

We test our algorithm on two Near-infrared Heterogeneous Face Recognition (HFR) datasets. The CASIA NIR-VIS 2.0 database [11] is the largest publicly available NIR HFR dataset. It contains 17,580 total images of 725 subjects. The CASIA HFB dataset [10] is an older and thus more widely used (though not as challenging) NIR HFR dataset. It has 5,098 total images from 200 subjects. Both datasets contains two views: View1 for algorithm development and parameter tuning, and View2 for performance reporting. View1 contains separate testing (probe/gallery) and training sets with different subjects. View2 has 10 sub-experiments each of which has the same setup as View1 with slightly different numbers of images.

After training our initialization networks, we follow the rest of the algorithm from [16] and perform heterogeneous face recognition on two datasets. Performance on View 1 of the datasets is shown in Figures 7 and 8. For both datasets, it is clear that for the optimal weight decay (.0005), there is a negligible effect on the data set performance. As the weight decay increases and the initialization networks get smaller, the performance eventually decreases.

After optimizing hyper-parameters on View 1, we then evaluate our approach on View 2 of both datasets. TABLE IV shows that we achieve the best performance to date on the NIRVIS 2.0 dataset. We outperform the previous state-of-the-art by 2.8 percent. TABLE V shows that, although we do not achieve state-of-the-art results, we do improve upon the previous deep learning approach of [16].

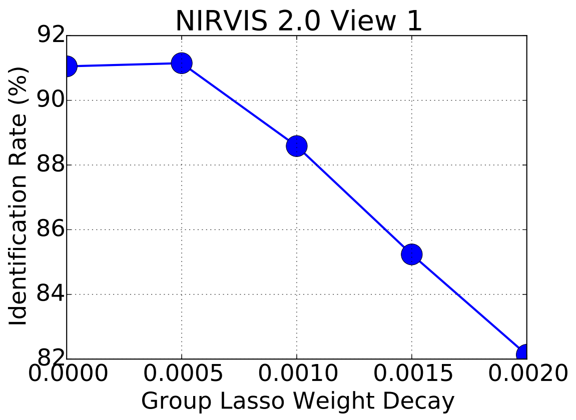

Fig. 7. Trade-off between the recognition rate and initialization network weight decay on View 1 of the NIR Vis 2.0 dataset.

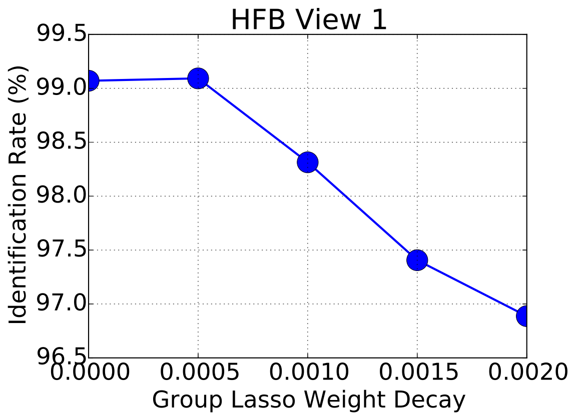

Fig. 8. Trade-off between the recognition rate and initialization network weight decay on View 1 of the HFB dataset.

V. Conclusion

We have presented a method that uses the group lasso penalty to remove unnecessary hidden nodes from a deep neural network. It works by enforcing group sparsity in the network parameters to zero out hidden nodes which do not provide much contribution. The method can be used to make a network smaller without sacrificing performance and to help determine the optimal number of hidden nodes for each layer. Finally, we used our method to shrink the initialization network of a NIR HFR algorithm and improved upon the state-of-the-art for that problem.

TABLE IV
Performance on View2 of CASIA NIR-VIS 2.0 Face Database

NIR-VIS 2.0	Rank 1	Std. Dev.	FAR=.001
CDFL[2]	71.5	1.4	55.1
LMCFL[3]	75.7	2.5	55.9
[4]	78.5	1.67	85.8
C-CBFD[13]	81.8	2.3	47.3
[19]	86.2	0.98	81.29
[16]	87.1	0.88	74.5
Our Method	**89.7**	1.06	77.0

TABLE V
Performance on View2 of CASIA HFB Face Database

HFB	Rank 1	FAR=.001
P-RS [5]	87.8	95.8
C-DFD[9]	92.2	65.5
THFM [22]	99.28	98.42
[19]	99.38	92.25
[16]	97.58	85.0
Our Method	98.34	87.5

VI. Acknowledgements

All plots in this work were generated with Matplotlib [1]

References

[1] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.
[2] Y. Jin, J. Lu, and Q. Ruan. Coupled discriminative feature learning for heterogeneous face recognition. *Information Forensics and Security, IEEE Transactions on*, 10(3):640–652, 2015.
[3] Y. Jin, J. Lu, and Q. Ruan. Large margin coupled feature learning for cross-modal face recognition. In *Biometrics (ICB), 2015 International Conference on*, pages 286–292, May 2015.
[4] F. Juefei-Xu, D. Pal, and M. Savvides. Nir-vis heterogeneous face recognition via cross-spectral joint dictionary learning and reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 141–150, 2015.
[5] B. F. Klare and A. K. Jain. Heterogeneous face recognition using kernel prototype similarities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(6):1410–1422, 2013.
[6] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*. Citeseer, 1990.
[7] V. Lebedev and V. Lempitsky. Fast convnets using group-wise brain damage. *arXiv preprint arXiv:1506.02515*, 2015.
[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
[9] Z. Lei, M. Pietikainen, and S. Li. Learning discriminant face descriptor. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(2):289–302, Feb 2014.
[10] S. Z. Li, Z. Lei, and M. Ao. The HFB face database for heterogeneous face biometrics research. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 1–8. IEEE, 2009.
[11] S. Z. Li, D. Yi, Z. Lei, and S. Liao. The casia nir-vis 2.0 face database. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, pages 348–353. IEEE, 2013.

[12] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814, 2015.

[13] J. Lu, V. Liong, X. Zhou, and J. Zhou. Learning compact binary face descriptor for face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(10):2041–2056, Oct 2015.

[14] L. Meier, S. Van De Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

[15] M. Moczulski, M. Denil, J. Appleyard, and N. de Freitas. Acdc: A structured efficient linear layer. *arXiv preprint arXiv:1511.05946*, 2015.

[16] C. Reale, N. M. Nasrabadi, H. Kwon, and R. Chellappa. Seeing the forest from the trees: A holistic approach to near-infrared heterogeneous face recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2016.

[17] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini. Group sparse regularization for deep neural networks. *arXiv preprint arXiv:1607.00485*, 2016.

[18] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, and Z. Wang. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1476–1483, 2015.

[19] D. Yi, Z. Lei, and S. Li. Shared representation learning for heterogenous face recognition. In *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, volume 1, pages 1–7, May 2015.

[20] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

[21] H. Zhou, J. M. Alvarez, and F. Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, pages 662–677. Springer, 2016.

[22] J.-Y. Zhu, W.-S. Zheng, J.-H. Lai, and S. Li. Matching nir face to vis face using transduction. *Information Forensics and Security, IEEE Transactions on*, 9(3):501–514, March 2014.