Deep Face Model Compression Using Entropy-based Filter Selection

Bingbing Han[1], Zhihong Zhang[1], Chuanyu Xu[1], Beizhan Wang[1] ⋆, Guosheng Hu[2], Lu Bai[3], Qingqi Hong[1], and Edwin R. Hancock[4]

[1] Xiamen University, Xiamen, Fujian, China
[2] Anyvision group, Belfast, UK
[3] Central University of Finance and Economics, Beijing, China
[4] Department of Computer Science, University of York, York, UK

Abstract. The state-of-the-art face recognition systems are built on deep convolutional neural networks (CNNs). However, these CNNs contain millions of parameters, leading to the deployment difficulties on mobile and embedded devices. One solution is to reduce the size of the trained CNNs by model compression. In this work, we propose an entropy-based prune metric to reduce the size of intermediate activations so as to accelerate and compress CNN models both in training and inference stages. First the importance of each filter in each layer is evaluated by our entropy-based method. Then some unimportant filters are removed according to a predefined compressing rate. Finally, we fine-tune the pruned model to improve its discrimination ability. Experiments conducted on LFW face dataset shows the effectiveness of our entropy-based method. We achieve 1.92× compression and 1.88× speed-up on VGG-16 model, 2× compression and 1.74× speed-up on WebFace model, both with only about 1% accuracy decrease evaluated on LFW.

1 Introduction

In the past few years, as the emergence of big training data, Convolutional Neural Networks (CNNs) have attained great success in the field of computer vision, from image classification [1–4], to other applications such as image caption [5], super resolution [6] and others. To extract massive information from big training data, the researchers typically trained a large CNN or a CNN ensemble, which contains millions of parameters. Nevertheless, due to the need for mobile payment and security, it is very important to apply face recognition to mobile devices, computing large CNNs cost too much memory and running time, which prevents them from being widely used. Thus network compression has attracted great attention from researchers. In this paper, we focus on deep model compression in the field of face recognition.

In order to apply large CNNs to mobile devices, a large number of CNN methods have been put forward. The early ideas focused on how to compress

⋆ Corresponding author: wangbz@xmu.edu.cn.

the parameters of fully-connected layers [7], and latter work which compressed convolutional layers [8, 11] for the purpose of accelerating model speed becomes an important research field. However, very little research has been proposed to reduce the size of activation in each layer.

In this work, we apply model compression to face recognition. We propose a framework to remove redundant filters in order to accelerate and compress CNNs at the same time. Our main idea is that, in each layer, we should keep filters that are representative (i.e these filters could extract as much information as original filters) and prune these less representative ones. As a result, we could reduce the number of channels directly without losing the information of feature map, converting a cumbersome network to a much smaller one without or with a little bit performance descend. We propose an entropy-based channel selection metric to evaluate the importance of each filter and prune 'weak' filters in order to keep some important filters that could capture nearly the same information as all the original filters. Then the pruned network is fine-tuned to regain its discrimination ability.

We evaluate our entropy-based channel selection metric for face recognition using two commonly used models: VGG-16 [2] and WebFace [19]. These two models are both trained and finetuned on the CASIA-WebFace dataset [16]. Our entropy-based prune metric achieve 1.92× compression and 1.88× speed-up on VGG-16 model, 2× compression and 1.74× speed-up on WebFace model, both with about 1% accuracy decrease.

Our strategy has the following advantages and contributions:

- We propose a simple yet efficient framework to compress CNN models in both training and inference stage. Our framework can reduce the number of filters so as to compress the size of activation in *each* layer, which catch almost no attention in previous work.
- We use an effective learning strategy to make a balance between training speed and classification accuracy in our pruning framework.
- Our framework does not rely on any specific libraries to gain the compression and acceleration performance, thus could be applied to any popular CNN library, such as caffe [15], tensorflow [17].

2 Related Works

Researchers have revealed that large CNNs suffered from over-parameterization, which causes not only the waste of memory and computation, but also serious over-fitting problem. Denil et al. [9] demonstrated that we can use only a small part of its original parameters to reconstruct a network almost without the accuracy dropping. However, very little work aims at optimizing the number of filters. Most previous works mainly focus on two fields: one focused on how to reconstruct the fully-connected layers, the other one focused on how to prune the weights of convolutional layers.

Focusing on the fully-connected layers, some researchers reconstructed layers or modules to substitute bottleneck components. GoogleNet [4] and ResNet [3] are famous examples which use the global average pooling to replace the dense fully-connected layers in order to reduce memory and computation consumption. Recently, SqueezeNet [10] uses a block called “Fire Module” and other strategies to achieve AlexNet [1] level accuracy with only 4.8MB disk size, almost 50× fewer parameters than the original AlexNet size.

Focusing on the convolutional layers, different methods have been explored to reduce number of connections and weights in neural networks. Some researchers approximate the dense parameters matrix with several low-rank matrices in order to compute the matrix-vector with high-speed [7]. Other researchers focus on network pruning, which has widely been studied to reduce the number of connections and prevent over-fitting [12]. Li et al. [13] use the absolute weight to measure the importance of each filter and remove less useful filters, which has the similar idea with ours, but our method is more efficient.

Though the mentioned methods work well in compressing large CNNs, we look for a framework that has optimal number of filters in each layer for specified networks and given tasks.

3 Entropy-Based Model Compression

In this section, we detail the proposed entropy-based channel selection metric which performs important filter selection. Our main idea is to discard some unimportant filters, and to recover its performance via fine-tuning. Finally we describe our efficient learning strategy.

3.1 Framework

Fig. 1 illustrates the framework of the proposed activation pruning method. For a specific layer that we want to prune (i.e. layer k), we just focus on the activation tensor, we use entropy-based channel selection metric to evaluate the importance of each filter, then some less important filters will be removed from the original model, which makes the architecture more compact. Clearly, the corresponding channels of filters in the next layer are removed too. This strategy not only makes the network with fewer parameters which lead to the decrease of running time and memory consumption, but also reduces the size of activations. In addition, some researchers have proved that each neuron is represented by many neurons, each neuron participates in the representation of many concepts [14], so whatever these filters are, they can extract some features from the feature maps of the former layer. Thus, generalization ability of the pruned model will be affected. We fine-tune the whole network after removing the less important filters in order to recovery the performance.

3.2 Entropy-Based Prune Metric

We use a triplet $(I_i, W_i, *)$ to denote the convolution in layer i, where $I_i \in \mathbb{R}^{(c\times h\times w)}$ means the input tensor, $W_i \in \mathbb{R}^{(d\times c\times k\times k)}$ means a set of filter weights,

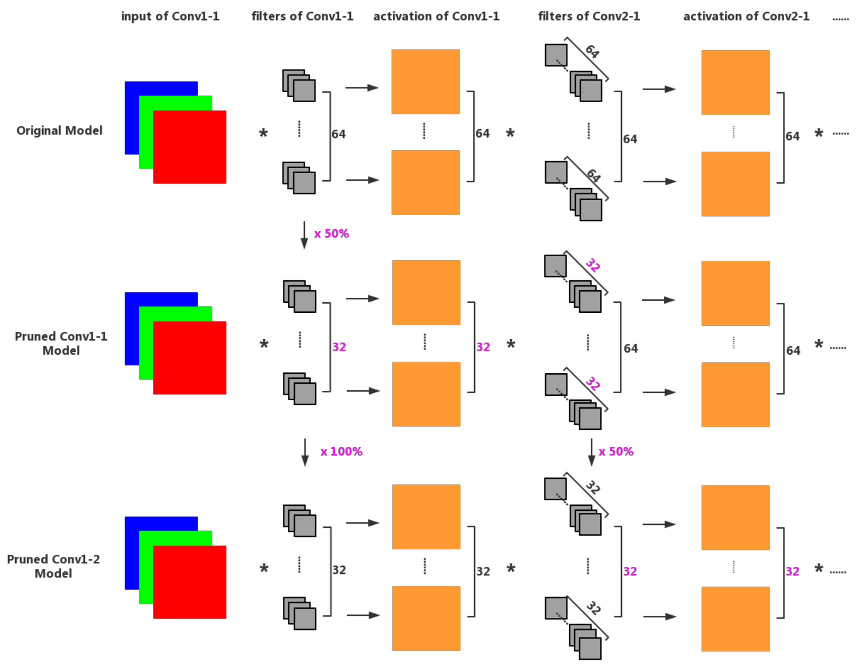

Fig. 1. Illustration of the framework of the iterative activation pruning approach on VGG-16 network. We predefine the compressing rate as 0.5, meaning that 50% filters in each layer will be removed. First, we compute the entropy of the activation of Conv1-1 and discard the unimportant filters. Then we fine-tune the pruned model with few iterations to recover the discrimination of pruned Conv1-1 model. Next we prune the Conv1-2 based on the former model in the same way. After the last layer being pruned, the final model will be fine-tuned carefully. $*$ is the convolution operator.

$*$ means the convolution operation, and d, c , $k \times k$ are the number of filters, the number of output channels on the upper layer and the size of the filters respectively. Our goal is to remove some unimportant filters, i.e. reduce d.

From the structure of networks, we know that each filter corresponds to a single channel of its activation tensor, so the ability of extracting features of each filter is closely related to its activation channel. In this paper, we propose an entropy-based metric to evaluate the importance of filters. Entropy is a commonly used metric to measure the disorder or uncertainty in information theory. A large entropy value means the system contains more information. In our filter pruning method, if a channel of activation tensor contains less information, its corresponding filter is less important, thus could be removed.

Specifically, we first calculate the mean value of each channel, converting a $c \times h \times k$ tensor into a $1 \times c$ vector. In this way, each channel of a $I_{(i+1)}$ (activation of layer i is also the input of layer $i+1$) has a corresponding value for one image. In order to calculate the entropy, more output values need to be collected, which can be obtained using an evaluation set. In practice, the evaluation set can simply be the original training set, or a subset of it. Finally, we get a matrix $M \in \mathbb{R}^{(n \times c)}$, where n is the number of images in the evaluation set, and c is the channel number. For each channel j, we compute its final score H_j according to the entropy of its output$M_{(:,j)}$:

$$H_j = E_p log\frac{1}{p(x)} = -\sum_{x \in M_{(:,j)}} p(x) log p(x)$$

Another important issue is to decide the pruning boundary. One possible method is to denote a specific value, all channels with score below this value are removed from the network. However the specific value is a hyperparameter, which is hard to be specified. Another more practical method is to predefine a compression rate, all the filters are sorted in the descending order corresponding to the entropy score, and only the top k filters are preserved. Of course, the corresponding channels are removed too.

3.3 Network Trimming

Our network trimmimg consists of three main steps. First the network is trained under conventional process and the number of filters in each layer is set empirically. Next, we run the network on a large validation dataset to obtain the entropy score of each filter, only the top k filters are preserved according to the compression rate. Of course, the connections to and from the removed filter are removed accordingly. After the filter pruning, the trimmed network is initialized using the weights before trimmimg. The trimmed network reduces some level of performance. So, at last we re-train the network to enhance the performance of the trimmed one.

4 Experiments

We use the standard caffe library [15] to train our deep models. The two CNN architectures we used are VGG-16 [2] model and WebFace model [19]. These two architectures achieve great succuss in the filed of face recognition. Our models are trained using the CASIA-WebFace dataset [16] consisting of 419922 face images of 10575 identities. The training images are horizontally flipped for data argumentation. The finetuning process during model compression is conducted on CASIA-WebFace dataset as well. The face recognition performance is evaluated on the famous LFW (Labeled Faces in the Wild) database [21]. LFW contains 5,749 subjects and 13,233 images. We follow the standard unrestricted protocol and conduct 10-fold cross validation. The face recognition rate is evaluated by mean accuracy. The sample images of LFW are shown in Fig. 2 All the experiments are conducted on a computer equipped with Nvidia Tesla K80 GPU.

Fig. 2. A column in the green box indicates the same person and the red box indicates different people.

4.1 VGG-16 Network

The original VGG-16 model contains 13 convolution layers, 2 fully-connected layers and one softmax layer. As our training data CASIA-WebFace database is much smaller than ImageNet [20] which is used to train the original VGG-16 network, in this work, we remove the two fully-connected layers to avoid overfitting. At the same time, the last convolutional layer is followed by a global average pooling layer whose kernel size is of 14×14. This type of global pooling can replace a fully connected layer but has much fewer parameters. It has widely been used to design an efficient network such as google Inception network [4].

Before training, all the training images are aligned and then resized to 224×224. We achieve 97.55% face recognition rate on LFW using the full (not compressed) VGG-16 model. The next step is to prune the model based on our entropy-based prune metric. In this work, we set the compression rate as 0.5, meaning that half of the parameters of convolutional layers should be removed.

We empirically find that the first few convolution layers have much more redundant information than the deeper layers which capture the semantic information. Therefore, we only prune the first 10 layers (Conv1-1 to Conv4-3 in Table 1). During finetuning/pruning, we set the original learning rate to 0.01 for each layer. The learning rate is reduced to 0.1 of the previous one when the loss stops decreasing. The number of iterations and the learning rate changing are detailed in Table 2.

In Table 1, the parameters before and after compression of VGG-16 network are detailed. The convolutional kernel weights are greatly reduced, leading to a more efficient inference. The computations measured by FLOPs are also dramatically reduced. Interestingly, our method halves the intermediate activation size which can greatly save the memory.

Table 1. The performance of our method to reduce Parameters and FLOPs (FLoating-point OPerations) on the VGG-16 model. The activation size is the sum of convolutional, relu, pooling layers' output and the input data when batch size is set to 1.

Layer	Parameters		FLOPs		Intermediate Activation Size	
	Original	Pruned	Original	Pruned	Original	Pruned
Conv1-1	1.73K	0.86K	86.7M	43.35M	12.25MB	6.125MB
Conv1-2	36.86K	9.22K	1.85B	462.42M	12.25	6.125MB
Conv2-1	73.73K	18.43K	0.92B	231.21M	6.13MB	3.06MB
Conv2-2	147.46K	36.86K	1.85B	462.42M	6.13MB	3.06MB
Conv3-1	294.91K	73.73K	0.92B	231.21M	3.06MB	1.53MB
Conv3-2	589.82K	147.46K	1.85B	462.42M	3.06MB	1.53MB
Conv3-3	589.82K	147.46K	1.85B	462.42M	3.06MB	1.53MB
Conv4-1	1.18M	294.92K	0.92B	231.21M	1.53MB	0.77MB
Conv4-2	2.36M	589.82K	1.85B	462.42M	1.53MB	0.77MB
Conv4-3	2.36M	589.82K	1.85B	462.42M	1.53MB	0.77MB
Conv5-1	2.36M	1.18M	462.42M	231.21M	392KB	392KB
Conv5-2	2.36M	2.36M	462.42M	462.42M	392KB	392KB
Conv5-3	2.36M	2.36M	462.42M	462.42M	392KB	392KB
Total	14.71M	7.66M	15.35B	4.67B	109.89MB	56.33MB

Table 2. The number of iterations on finetuning/pruning each layer. For {10000, 20000} of Conv1-2, it means the learning rate changes to 0.1 of previous one at 10000th iteration, and the training stops at 20000th iteration.

Layer	Conv1-1	Conv1-2	Conv2-1	Conv2-2	Conv3-1
	-	10000	-	10000	-
Iterations	10000	20000	10000	20000	10000
Layer	Conv3-2	Conv3-3	Conv4-1	Conv4-2	Conv4-3
	-	10000	-	-	40000/80000
Iterations	10000	20000	10000	10000	12000

In Table 3, we report our compression results. We can see that our model gains 1.88 × speed up measured by the averaged running time of 1000 images during inference. Also, we achieve 1.92× compression results in terms of the number of model parameters. Compared with original VGG-16 model, our method reduces the FLOPS computations by 3.29×. However, the face recognition rate only drops by around 1%.

Table 3. The summarization of the performance of our compression method.

Model	Accuracy (LFW)	FLOPS	Compression	Speed-up
Original VGG-16	97.55%	1×	1×	1×
Pruned VGG-16	96.50%	3.29×	1.92×	1.88×

4.2 WebFace Network

The WebFace [19] is another popular face recognition CNN architecture. Compared with VGG-16, WebFace is more compact. We conduct our compression method on WebFace to verify the effectiveness of our method on smaller architecture. The architecture of WebFace is detailed in Fig. 3. Clearly, WebFace also stackes several Conv-Pool-Relu unites. Before fully-connected layers, a 7×7 global pooling is used. Following the original work [19], the input images are aligned and cropped to the size of 100×100. The training and finetuning strategies are similar to those shown in Section 4.1.

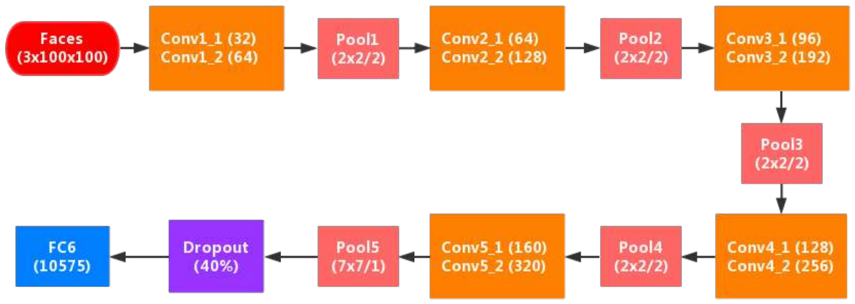
Faces (3x100x100)
Conv1_1 (32) Conv1_2 (64)
Pool1 (2x2/2)
Conv2_1 (64) Conv2_2 (128)
Pool2 (2x2/2)
Conv3_1 (96) Conv3_2 (192)
Pool3 (2x2/2)
Conv4_1 (128) Conv4_2 (256)
Pool4 (2x2/2)
Conv5_1 (160) Conv5_2 (320)
Pool5 (7x7/1)
Dropout (40%)
FC6 (10575)

Fig. 3. The framework of Webface.

In Table 4, we report our compression results using WebFace network. Unlike VGG-16 network, we evaluate the compression results on different compression rates: 0.9 (Pruned-90 in Table 4), 0.75 and 0.5. The results in terms of face recognition rates (LFW), FLOPS, compression and running time are detailed. For the largest compression rate 0.5 (Pruned-50), the accuracy drops only 1.05%, but the FLOPS of Pruned-50 is only 28.07% of the original WebFace, the model size is only half, and the running time is only 57.60% compared with the original model. Another 2 models (Pruned-75, Pruned-90) also achieve great performance. In the real application, it is always a trade-off between accuracy and model size/computation. Therefore, Table 4 is an important reference for researchers and engineers to look for such a trade-off.

Table 4. WebFace compression results on different compression rates.

Model	Accuracy (LFW)	FLOPS		Compression		Speed-up	
		Nums	Percent	Nums	Percent	Nums	Percent
Original WebFace	96.92%	770M	100%	1.75M	100%	5.92ms	100%
Pruned-90	97.28%	619M	80.37%	1.53M	87.53%	5.24ms	88.51%
Pruned-75	96.62%	448M	58.18%	1.26M	71.71%	4.60ms	77.70%
Pruned-50	95.87%	216M	28.07%	0.88M	50%	3.16ms	57.60%

4.3 Effectiveness Analysis

To demonstrate the effectiveness of our entropy-based metric, we compare our method with random selection which prunes the CNN model parameters randomly. We prune 3 different layers of WebFace with these two methods. All the experimental setups are kept the same, except for channel selection method.

The results are summarized in Table 5. The 3 layers of WebFace we choose to prune are: Conv1-1, Conv3-1, and Conv4-2 (shown in Fig. 3). Each layer is pruned independently with 0.5 compression rate, and fine-tuned 10000 iterations. As can been seen, our entropy-based method consistently works better than random selection method on all the 3 selected layers, showing the effectiveness of our compression method. The advantage of our method is larger on deeper layers, e.g. 96.48% vs 96.01% (Conv1-1), 95.97% vs 95.02% (Conv4-2). In addition, the finetuing is very important for performance improvement for both our method and random selection.

Table 5. Comparison of entropy-based channel selection metric and random selection. 0 iteration means finetuning is not conducted after pruning.

Model	Conv1-1		Conv3-1		Conv4-2	
	0 Iteration	10K Iterations	0 Iteration	10K Iterations	0 Iteration	10K Iterations
Entropy	94.48%	96.48%	93.00%	96.90%	92.30%	95.97%
Random	94.27%	96.01%	92.55%	96.32%	92.27%	95.02%

5 Conclusion

In this paper, we propose a network pruning strategy to trim redundant filters based on the entropy-based channel selection metric. Our method can remove unimportant filters that provide little contribution to the final performance without damaging results of our network. In addition, our strategy does not rely on any dedicated library, thus can widely be used in various applications with current deep learning libraries.

Acknowledgment

This work is supported by National Natural Science Foundation of China (Grant No.61402389, 61503422 and 61502402), the Fundamental Research Funds for the Central Universities in China (no. 20720160073) and the Natural Science Foundation of Fujian Province of China(Grant No. 2015J05129).

References

1. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenetclassication with deep convolutional neural networks. In Advances in Neural Information Processing Systems (NIPS),pages 1097C1105, 2012. 1, 3, 4
2. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations (ICLR) , 2015. 1, 2, 4, 6
3. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) , pages 770C778, 2016. 1, 2, 3, 4, 5, 7
4. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) , pages 1C9, 2015. 3, 4, 6
5. Fang H, Gupta S, Iandola F, et al. From Captions to Visual Concepts and Back. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1473-1482
6. Chao Dong, Chen Change Loy, Kaiming He, Xiaoou Tang. Learning a Deep Convolutional Network for Image Super-Resolution. In Proceedings of European Conference on Computer Vision (ECCV), 2014
7. V. Sindhwani, T. Sainath, and S. Kumar. Structured transforms for small footprint deep learning. In Advances in Neural Information Processing Systems (NIPS) , pages 3088C3096, 2015. 1, 2
8. J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. Quantized convolutional neural networks for mobile devices. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) , pages 4820C4828, 2016. 1, 2, 3, 6
9. M. Denil, B. Shakibi, L. Dinh, and N. de Freitas. Predicting parameters in deep learning. In Advances in Neural Information Processing Systems (NIPS) , pages 2148C2156, 2013. 2
10. F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size. In arXiv preprint arXiv:1602.07360 , 2016. 3

11. Jian-Hao Luo, Jianxin Wu. An Entropy-based Pruning Method for CNN Compression. arXiv:1706.05791, 2017. 6
12. S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efcient neural network. In Advances in Neural Information Processing Systems (NIPS) , pages 1135C 1143, 2015. 2, 3, 6
13. H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf.Pruning lters for efcient ConvNets. In arXiv preprintarXiv:1608.08710, 2016. 2, 5, 7
14. Y. Bengio, A. Courville, and P. Vincent. Representationlearning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI),35(8):1798C1828, 2013. 3
15. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Learning distributed representations of concepts. In ACM International Conference on Multimedia (ACM MM) , pages 675C678, 2014. 5
16. CASIA-WebFace dataset:
`http://www.cbsr.ia.ac.cn/english/CASIA-WebFace-Database.html`
17. Abadi M, Agarwal A, Barham P, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems[J]. arXiv preprint arXiv:1603.04467, 2016.
18. Min Lin, Qiang Chen, Shuicheng Yan. Network in Network. CoRR, abs/1312.4400, 2013.
19. Dong Yi, Zhen Lei, Shengcai Liao, Stan Z. Li. Learning Face Representation from Scratch. arXiv:1411.7923.
20. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09, 2009.
21. LFW:
`http://vis-www.cs.umass.edu/lfw/`