Distilling

shuang ao

August 11, 2016

1 Background

Distillation for domain adaptation [4] and *privileged information* [10] are two techniques that enable machines to learn from other machines. Both methods address the problem how to build a student model that can learn from the advanced teacher models. Recently, Lopez *et al.* [6] proposed a framework called *generalized distillation* that unifies both techniques and show that it can be applied in many scenarios.

1.1 Privileged information

In the original work of the privileged information [10], the data can be represented as a collection of the triples:

$$\{(x_1, x_1^*, y_1), (x_2, x_2^*, y_2) \dots (x_n, x_n^*, y_n)\}$$

where each (x_i, y_i) is a feature-label pair, and the novel element x_i is additional information about the example (x_i, y_i) provided by an intelligent teacher, such as to support the learning process. However, in the testing procedure, the learning machine is not able to obtain the privileged information from the teacher. Vapnik?s learning using privileged information is one example of what we call machines-teaching machines: the paradigm where machines learn from other machines, in addition to training data.

1.2 Distillation

Distillation uses a simple (student) machine learns a complex task by imitating the solution of a flexible (teacher) machine. For a c-class scenario, distillation works as follow: consider the data

$$\{x_i, y_i\}_{i=1}^n, \qquad x \in R^d, y \in \Delta^c$$

where Δ^c is a c-dimensional probability vector. In traditional learning, we try to learn a function f_t that:

$$f_t = \arg\min_{f_t \in \mathcal{F}_t} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, \sigma(f_t(x_i))\right) + \Omega(||f_t||) \tag{1}$$

here σ is the softmax operation:

$$\sigma(z)_k = \frac{e^{z_k}}{\sum_{j=1}^{c} e^{z_j}}$$

and ℓ is the cross-entropy loss function:

$$\ell(y, \hat{y}) = -\sum_{i=1}^{c} y_i \log \hat{y}_i$$

In distillation, we try to learn the student machine f_s to imitate the teacher machine f_t. During the learning process, f_s can receive the soft label s_i from the teacher machine f_t for each training example x_i.

$$s_i = \sigma(f_t(x_i)/T) \tag{2}$$

where T is the temperature for distillation. It is worthy to note that in distillation, f_t can either be a single large complex neural network or an ensemble of some complex classifiers. We can see that the soft label s_i, which reveals the class dependency, is more informative than the class label y_i.

The student machine can be learned by:

$$f_s = \arg\min_{f_s \in \mathcal{F}_s} \frac{1}{n} \sum_{i=1}^{n} \left[\lambda \ell \left(y_i, \sigma(f_s(x_i)) \right) + (1 - \lambda) \ell \left(s_i, \sigma(f_s(x_i)) \right) \right] \tag{3}$$

here, $\mathcal{F}_s$ is a simpler function class than $\mathcal{F}_t$ and λ is the imitation parameter to balance the importance between the hard label y_i and the soft label s_i. The temperature $T > 0$ controls how do we want to smooth the probability from the teacher f_t. Higher temperature leads to softer label.

1.3 Generalized distillation

Generalized distillation (GD) tries to unify the two frameworks together while using the soft label as the privileged information. The process of generalized distillation is as follows:

1. Learn teacher f_t using the input-output pairs $\{x_i^*, y_i\}_{i=1}^n$ and Eq. 1.

2. Compute teacher soft labels s_i, using the temperature parameter $T > 0$.

3. Learn the student f_s using the pair $\{(x_i, y_i), (x_i, s_i)\}_{i=1}^n$ and imitation parameter λ.

2 Domain adaptation with distilling SVMs

GD can be used in many scenario such as multi-task learning, semi-supervised learning and reinforcement learning. As generalized distillation only required for the training inputs $\{x_i, y_i\}_{i=1}^n$ and the output s from the teacher function f_t, it can be naturally applied in domain adaptation, called *Generalized Distillation Domain Adaptation* (**GDDA**), where the the source model can be used as the teacher to output the soft labels and the student model is the target model.

Similar to GD, the process of GDDA is as follow:

1. For a N-class classification problem, suppose we have the training data $\{x_i, y_i\}_{i=1}^L$ and $M-1$ source (teacher) models $\{f_j^*\}_{j=1}^{M-1}$.

2. For each of the training example x_i and each source model f_j^*, computer the corresponding N-dimensional soft label y_{ij}^* with some temperature $T > 0$.

3. Learn the target model f_t using the $(M+1)$-tuple $\{x_i, y_i, y_{i1}^*, \ldots, y_{M-1}^*\}_{i=1}^L$ with some imitation parameters $\{\lambda_i\}_{i=1}^M$

There are several advantages for applying generalized distillation for domain adaptation:

1. Compatible with **various of classifiers**. GD only requires the outputs of the teacher model instead of asking for the specific parameters of the teacher model. Therefore, we can treat the teacher model, either a single model (single source) or an ensemble of models (multi-source), as a black-box. This means GDDA is compatible with almost any types of source model.

2. Highly effective in **small data regimes**. The theoretical analysis shows that the teacher is most helpful when working with small datasets, or in the initial stages of online learning. In domain adaptation, we also requires to learn an effective target classifier with just a few training examples.

3. Compatible with **different scenarios**. GD can be used in a semi-supervised scenario. Similarly, GDDA can work in many different transfer learning scenarios besides a supervised setting. For example, in a unsupervised transfer learning scenario, we can set the imitation parameter of the hard label to 0 and learn the target model. In the semi-supervised scenario, we can choose a small value for the imitation parameter of the hard label as it could be less informative than the soft labels.

4. **Better generalization performance**. According to GD, the student model can be a much simple function than the teacher model. This implies that the VC dimension of the student model is smaller than the teacher(s). According to the principle of Empirical Risk Minimization (EMR) [11], in GDDA, as long as the student (target) model achieves the same training error of the teacher (source) model(s), it should have better generalization ability. This means the knowledge of the source model(s) can be effective transferred to the target task.

In GDDA, we have to decide the values of 2 parameters, the temperature T and imitation parameter λ. The temperature T control the smoothness of the soft label and the imitation parameter λ controls how similar the target model is to the source model. It is obvious that the value of imitation parameter can greatly affect the performance of the target model and choosing a proper imitation parameter can greatly improve the transfer efficiency. In the previous work of GD, the imitation parameter is determined by brute force search which greatly reduces its effectiveness. In domain adaptation, it is common that there could be multiple source models to be exploited and it is tedious to search the imitation parameter for each of the source model.

To solve this problem, we propose a novel method that can determined the imitation parameter λ autonomously, called GDDA-SVM. In our GDDA-SVM, instead of using cross-entropy loss, we use Mean Squared Error (MSE) as our loss function for the following two reasons: (1) Some recently work [1] [7] [8] [9] show that MSE is also an efficient measurement for a student model to mimic the behavior of the teacher model. (2) MSE can provide a closed form cross-validation error estimation and help us to choose the proper imitation parameter effectively.

2.1 Distillation with multiple sources

Suppose we have L examples $\{x_i, y_i\}_{i=1}^{L}$ from and N classes where $X \in R^{L\times d}, Y \in R^{L\times N}$. Meanwhile, there are $M-1$ the source (teacher) models providing the soft labels $\{Y_i^*\}_{i=1}^{M-1}$ for each of the L examples. As Combining the hard label Y and soft label Y^*, we have our new label matrix: $S = R^{M\times L\times N}$. To solve this N-class classification problem, we adopt the One-vs-All strategy to build N binary SVMs. To obtain the nth binary SVM, we have to solve the following optimization problem:

$$\begin{aligned} \min \quad & \frac{1}{2}|w_n|^2 + C\sum_{i,j}\lambda_i e_{ijn}^2 \\ s.t. \quad & e_{ijn} = s_{ijn} - w_n x_j \\ & \sum_i \lambda_i = 1 \\ & \lambda \in [0,1]; i \in M; j \in L \end{aligned} \tag{4}$$

To solve this optimization problem, we use KKT theorem [3] and add the dual sets of variables to the Lagrangian of the optimization problem:

$$\mathcal{L} = \frac{1}{2}|w_n|^2 + C\sum_{i,j}\lambda_i e_{ijn}^2 + \sum_{i,j}\alpha_{ij}^{(n)}\left(s_{ij} - wx_j - e_{ij}\right) + \beta^{(n)}\left(\sum_i \lambda_i - 1\right) \tag{5}$$

To find the saddle point,

$$\begin{aligned} \frac{\partial L}{\partial w_n} &= w_n - \sum_j \alpha_{ij}^{(n)} x_j = 0 \rightarrow w_n = \sum_j \alpha_{ij}^{(n)} x_j \\ \frac{\partial L}{\partial e_{ijn}} &= 2C\lambda_i e_{ijn} - \alpha_{ij}^{(n)} = 0 \rightarrow \alpha_{ij}^{(n)} = 2C\lambda_i e_{ijn} \end{aligned} \tag{6}$$

For each example x_j and its constraint of label s_{ijn}, we have $e_{ijn} + w_n x_j = s_{ijn}$. Replacing w_n and e_{ijn}, we have:

$$\lambda_i x_j \sum_k \alpha_{ik}^{(n)} x_k + \frac{\alpha_{ij}^{(n)}}{2C} = \lambda_i s_{ijn} \tag{7}$$

Summing over each constraint of example x_j, we have:

$$\underbrace{\sum_i \lambda_i}_{=1} x_j \sum_k \alpha_{ik}^{(n)} x_k + \sum_i \frac{\alpha_{ij}^{(n)}}{2C} = \sum_i \lambda_i s_{ijn} \tag{8}$$

Let $\eta_{jn} = \sum_i \alpha_{ij}^{(n)}$, we have:

$$\sum_j \eta_{jn} x_j x_i + \frac{\eta_{in}}{2C} = \sum_i \lambda_i s_{ijn} \tag{9}$$

This implies that solving the optimization problem (4) is equivalent to solve a standard LS-SVM whose the target is encoded as $\sum_i \lambda_i s_{ijn}$, i.e. the weighted label matrix S.

Here we use Ω to denote the matrix $\Omega = [K + \frac{\mathbf{I}}{2C}]$ where K is the kernel matrix $K = \{x_i x_j | i, j \in 1 \ldots L\}$. To simplify our notation, let $\eta'_n = M^{-1} S_n$ where S_n is the matrix $S_n = \{s_{ijn} | i \in M; j \in L\}$ and Ω^{-1} is the inverse of matrix Ω.

$$\eta_{jn} = \sum_i \lambda_i \eta'_{ijn}$$

The Leave-one-out estimation of the example x_j for the nth binary SVM can be written as [2]:

$$\begin{aligned} \sum_i \lambda_i s_{ijn} - \hat{y}_{jn} &= \frac{\eta_{jn}}{\Omega_{jj}^{-1}} = \frac{\sum_i \lambda_i \eta'_{ijn}}{\Omega_{jj}^{-1}} \\ \hat{y}_{jn} &= \sum_i \lambda_i \left(s_{ijn} - \frac{\eta'_{ijn}}{\Omega_{jj}^{-1}} \right) \end{aligned} \tag{10}$$

where Ω_{jj}^{-1} is the jth diagonal element of Ω^{-1}. Now, we have found an efficient way to estimate the output of each binary student model for example x_j. In the following part, we will introduce how to find the optimal λ_i for each of the source models.

2.2 Cross-entropy loss for imitation parameter estimation

From the previous part, we have already found a effective way to estimate the output of the SVM. The optimal imitation parameters, can be found by solving the following optimization problem:

$$\min \quad L_c(\lambda) = \frac{1}{2} \sum_i^M ||\lambda_i||^2 + \frac{1}{L} \sum_{j,n} \ell\left(y_{in}, \hat{y}_{jn}(\lambda)\right) \tag{11}$$

Here we use the L2 regularization term to guarantee that the estimated λ can perform well on the testing data. For the loss $\ell(\cdot)$, We choose the cross-entropy (CE) as the loss function. Compared to MSE, cross-entropy pay less attention to a single incorrect predicted example which reduced the affect of the outlier example in the training data. Moreover, cross-entropy works better with unlabeled data. Assume that we use one-hot strategy to encode the hard labels of the labeled examples while encoding the unlabeled examples with all 0s. When we use cross-entropy, no loss with generate from the side of the hard labels for the unlabeled examples. The target model only simulate the outputs from the source models. Therefore, Let:

$$\mu_{ijn} = s_{ijn} - \frac{\eta'_{ijn}}{\Omega^{-1}_{jj}} \tag{12}$$

$$P_{jn} = \frac{e^{\hat{y}_{jn}}}{\sum_h e^{\hat{y}_{jh}}} \tag{13}$$

$$\frac{\partial \ell(\lambda)}{\partial \lambda_i} = \sum_n \mu_{ijn} \left(P_{jn} - y_{jn}\right) \tag{14}$$

Algorithm 1 GDDA-SVM

```
Input: Input examples $X = \{x_1, ..., x_L\}$, number of classes $N$, size of sources $M$, 3D label matrix,
    $S = [Y_1, Y_2, ..., Y_M]$ with size $L \times M \times N$, temperature $T$, optimization iteration $iter$
Output: Target model $f_t = Wx$
    Compute $\Omega = [K + \frac{\mathbf{I}}{2C}]$
    Compute imitation parameter $\lambda$ with Algorithm 2
    Compute new label $Y_{new} = \sum_i \lambda_i Y_i$
    Compute $\eta = \Omega^{-1} Y_{new}$
    Compute $w_n = \sum_j \eta_{jn} x_j$
```

Algorithm 2 λ Optimization

```
Input: Input examples $X$, number of classes $N$, size of sources $M$, 3D label matrix $S$, temperature $T$,
    optimization iteration $iter$, Kernel matrix $\Omega$
Output: Imitation parameter $\lambda$
    Initialize $\lambda = \frac{1}{M}$,
    Let $S_n$ be the label matrix of $S$ for class $n$
    for Each label $S_n$ do
        Compute $\eta'_n = \Omega^{-1} S_n$
    end for
    Compute $\mu$ using (12)
    for $it \in iter$ do
        Compute $\hat{y}_{jn}$ and $P_{jn}$ with (10) and (14)
        $\Delta_\lambda \leftarrow 0$
        for each $x_j$ in $X$ do
            $\Delta_\lambda = \Delta_\lambda + \sum_n \mu_{ijn} (P_{jn} - y_{jn})$
        end for
        $\Delta_\lambda = \Delta_\lambda / L$
        $\lambda = \lambda - \frac{1}{itr}(\Delta_\lambda + \lambda)$
    end for
```

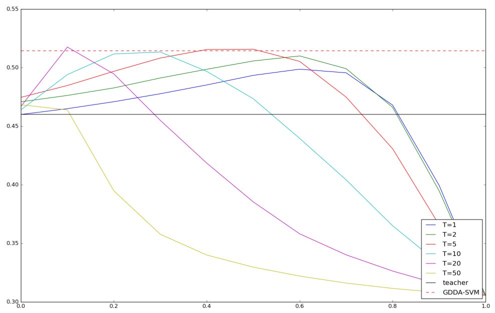
3 Experiments

3.1 Single Source for Office datasets

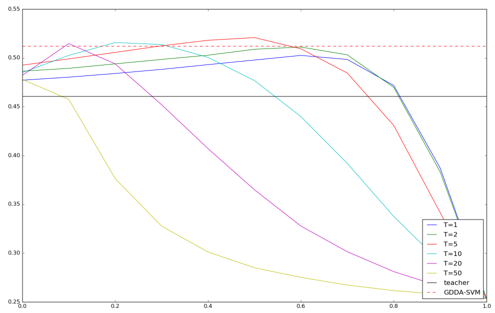
There are 3 subsets in offices datasets, webcam (795), amazon (2817) and dslr (498), sharing 31 classes. We perform 2 adaptation experiments, transferring from webcam to amazon and from dslr to amazon.

We use the features extracted from Alexnet [5] FC7 as the input features for both source and target domain. The teacher (source) model is trained with MLP on the whole source dataset and student (target) model is trained with GDDA-SVM. We also use brute force to search the imitation parameter λ in range $[0, 0.1, ..., 1]$ and show the results. Here our experiments run in a semi-supervised scenario where there are some unlabeled data in the target training set in each experiment. We also show the performance of the teacher model on the target task, denoted as "teacher". To avoid the randomness, we perform each experiments 10 times and report the results. The experiment results are shown in Figure 1 and 2 respectively.

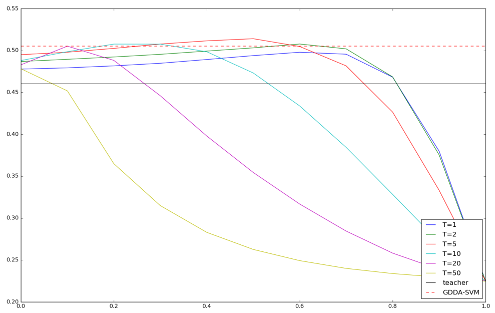
Observation: The performance of the target model can exceed the performance of the teacher in a "one-short" learning scenario. This means the target model can quickly learn the knowledge of the teacher. Increasing the size of the unlabeled example doesn't necessarily increase the performance. The performance of the model increases at the beginning but decreases then. May due to the performance of the difference between the source model and target model.



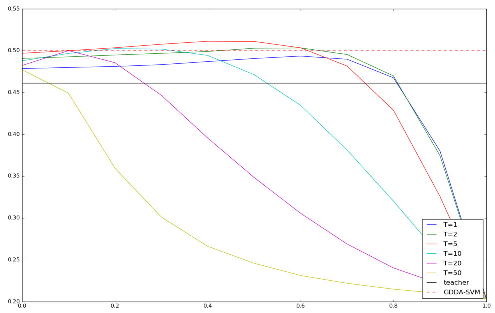
(a) 5 unlabeled



(b) 10 unlabeled



(c) 15 unlabeled

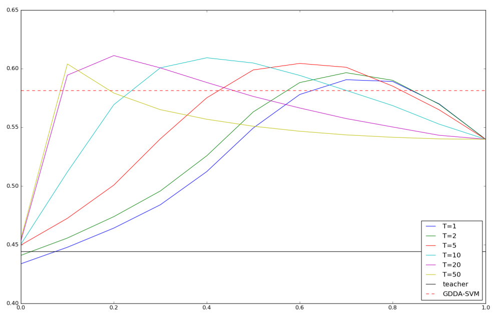


(d) 20 unlabeled

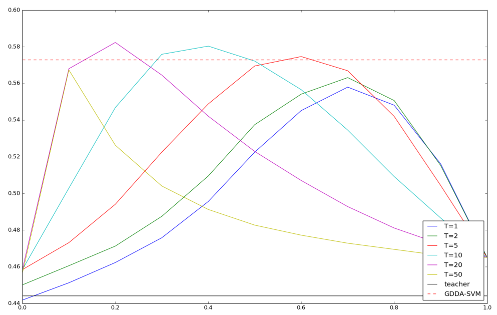
Figure 1: DSLR $\rightarrow$ Amazon. Semi-supervised adaptation with one labeled instance per class.

3.2 Multi-Source for Office datasets

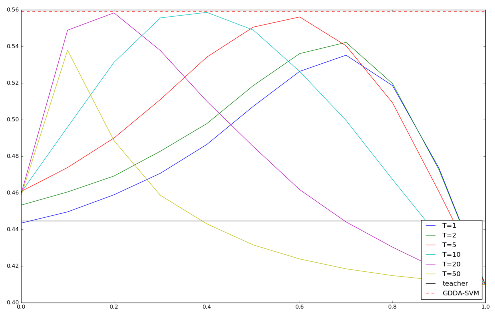
We transfer DSLR and Webcam to Amazon datasets with different settings similar to the experiment above. The experiment results are shown in Figure 3. In our experiment we have two baselines the source



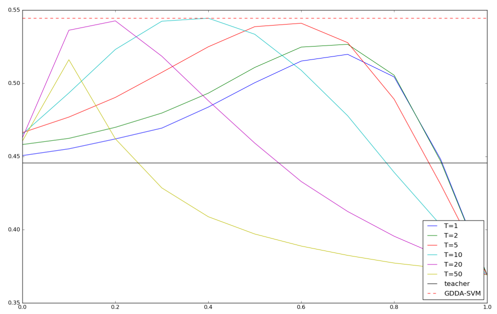
(a) 5 unlabeled



(b) 10 unlabeled



(c) 15 unlabeled



(d) 20 unlabeled

Figure 2: Webcam → Amazon. Semi-supervised adaptation with one labeled instance per class.

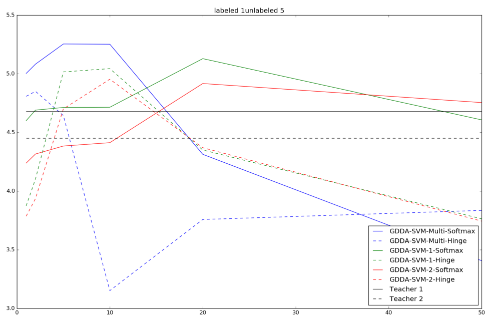
models from DSLR and Webcam (denoted as teacher 1 and teacher 2). We also show the performance of GDDA-SVM with each of the single source model (denoted as GDDA-SVM1 and GDDA-SVM2). The performance of the 2-source GDDA-SVM is denoted as GDDA-SVM-Multi.

Observation: The GDDA-SVM-Multi can exploit the knowledge from both source models effectively and outperform any single source model as well as the source models themselves.

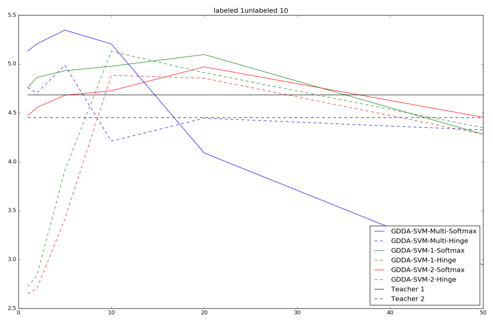
4 Future works

From current experiments we can see that GDDA can be a good framework for domain adaptation in single source scenario. We show that GDDA-SVM can fully exploit the knowledge from the source model. There are still a lot of work to be done within the GDDA-SVM framework:

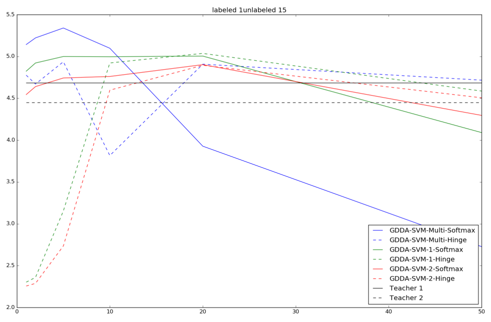
1. Add experiments on heterogeneous data.(Ask the advice from Wei Yin)
2. Add theoretical analysis on how the size of unlabeled data affect the performance of the target model. (Why increase the size of the unlabeled data can improve the performance at the beginning?)
3. Can we aggregate GDDA-SVM framework into the deep neural network to train a deep GDDA network? Currently, GDDAA-SVM tries to estimate the imitation parameter by solving a convex problem. How to effectively estimate the imitation parameter when it is integrated into a GDDA network.



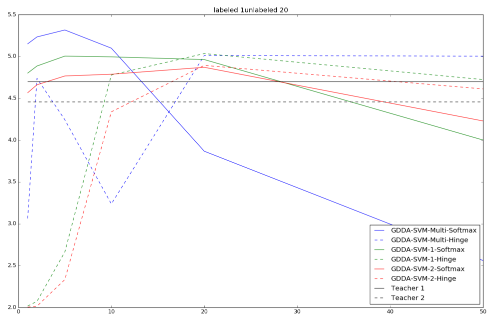
(a) 5 unlabeled



(b) 10 unlabeled



(c) 15 unlabeled



(d) 20 unlabeled

Figure 3: Webcam → Amazon. Semi-supervised adaptation with one labeled instance per class.

References

[1] J. Ba and R. Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[2] G. C. Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 1661–1668. IEEE, 2006.

[3] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[4] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[6] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

[7] P. Luo, Z. Zhu, Z. Liu, X. Wang, and X. Tang. Face model compression by distilling knowledge from neurons. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[8] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[9] G. Urban, K. J. Geras, S. E. Kahou, O. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson. Do deep convolutional nets really need to be deep (or even convolutional)? *arXiv preprint arXiv:1603.05691*, 2016.

[10] V. Vapnik and R. Izmailov. Learning using privileged information: Similarity control and knowledge transfer. *Journal of Machine Learning Research*, 16:2023–2049, 2015.

[11] V. N. Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.