LIGHTWEIGHT NEURAL NETWORKS FROM PCA & LDA BASED DISTILLED DENSE NEURAL NETWORKS

Mohamed El Amine Seddik[1,2] *Hassane Essafi*[1] *Abdallah Benzine*[1,3] *Mohamed Tamaazousti*[1]

[1]CEA List, France [2]Centralesupélec, France [3]Sorbonne University, CNRS, France

ABSTRACT

This paper presents two methods for building lightweight neural networks with similar accuracy than heavyweight ones with the advantage to be less greedy in memory and computing resources. So it can be implemented in edge and IoT devices. The presented distillation methods are respectively based on Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). The two methods basically rely on the successive dimension reduction of a given dense neural network (teacher) hidden features, and the learning of a *smaller* neural network (student) which solves the initial learning problem along with a mapping problem to the reduced successive features spaces. The presented methods are compared to baselines –learning the student networks from scratch–, and we show that the additional mapping problem significantly improves the performance (accuracy, memory and computing resources) of the student networks.

***Index Terms*—** Teacher-Student Networks, Compression, Distillation, PCA, LDA, Lightweight Networks

1. INTRODUCTION

Neural networks are the most effective machine learning methods nowadays, and since they generally require millions of parameters, their implementation in an IoT environment is quite ineffective. Indeed, IoT requires machine learning models with limited amount of parameters since the edge devices have limited resources (computation capacity, storage, bandwidth,...). This requirement among others have triggered intensive research activities and led to the emergence of new computing paradigms, i.e. edge computing [1] which has emerged as an answer to the need for shifting the computing from cloud to decentralized processing units close to the data sources [2]. The authors in [3] give a survey of works dealing with machine learning at the network edge. AI at the edge [4] is a concrete example of leveraging the computing and storage resources near the places where data is produced. The authors in [5] present a set of approaches proposed for embedding deep learning into the edge computing devices. They also present some applications that can fit with the edge computing paradigm and can take benefit from the edge network. However, the accuracy of deep learning models depends largely on the hyper-parameters of the network in particularly the number and the size of layers.

Nevertheless, big models are resources consuming which can impede the embedding of AI technologies in IoT devices with constrained resources. To tackle this issue some interesting solutions were proposed, model compression was among the first proposed methods. Model compression methods use well proved compression techniques for reducing the storage volume required by neural networks without impacting their performance. For instance the algorithm presented in [6] is composed of three methods applied in pipeline: first pruning (selection of the important weights of the network) method is applied, followed by quantization and Huffman coding (compression without lost). Compression methods are useful for reducing the required storage space of network models but inefficient for reducing the computation power which is one of the main critical aspects of the IoT devices. Recent methods targeting the production of smaller models, with the accuracy near of lager ones, was investigated [7].

In this article, we present two methods for distilling a given large dense neural network into a smaller one. The proposed methods are based on knowledge distillation concept [8], where a large (teacher) pre-trained network is used to train a smaller (student) network. They have the advantage to produce models that consume less storage and computing resources; the knowledge distillation approach is a kind of transfer learning approach which is commonly used in various machine learning problems [9, 10]. In [7] the authors show that the accuracy of the student network model depends highly on the ratio size between the two network models (teacher and student), higher is this ratio (gap size between teacher and student is large), smaller is the accuracy. To alleviate this problem the authors suggest, instead of distilling the student model directly from the teacher model, to use succession of teacher assistants based knowledge distillation approach where the models are distilled step by step until obtaining the final model. The proposed methods in this paper are notably complementary to the methods in [6, 7].

In this work, we have investigated two dense neural networks distillation methods which are respectively based on PCA and LDA. In Section 2 we will present the proposed methods, Section 3 is dedicated to the experiments. We end this paper by a discussion and a conclusion in Section 4.

2. PROPOSED METHODS

2.1. Setting & Notations

Consider a dense neural network, which we refer to as the teacher network (TN), composed of L layers and constructed in the following way, for $\ell \in [L]$:

$$\textbf{(TN)}: \begin{cases} \boldsymbol{h}^{(0)} = \boldsymbol{x} \in \mathbb{R}^{p_0}, \\ \boldsymbol{h}^{(\ell)} = f_\ell\left(\boldsymbol{W}^{(\ell)}\boldsymbol{h}^{(\ell-1)} + \boldsymbol{b}^{(\ell)}\right) \in \mathbb{R}^{p_\ell}, \end{cases} \tag{1}$$

where, $\boldsymbol{x}$ corresponds to the input data features, f_ℓ denotes the ℓ-th layer activation, $\boldsymbol{h}^{(\ell)}$ stands for the features of $\boldsymbol{x}$ extracted at layer ℓ, $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{p_\ell \times p_{\ell-1}}$ and $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{p_\ell}$ are respectively the weight matrix and bias at each layer ℓ. TN is typically of large size, meaning that the hidden features dimensions p_ℓ are relatively large. In the following, we will present two methods that construct a small network size, which we refer to as the student network (SN), based on the TN learned features. The two methods target different learning problems, depending if TN solves a supervised problem or an unsupervised one.

2.2. Neural Nets PCA-based Distillation (Net-PCAD)

Given a set of n training samples $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{p_0 \times n}$ on which TN was initially trained to perform some arbitrary learning task. The Net-PCAD method consists in performing a PCA [11] at each hidden layer of TN, and then training a SN to perform the same learning task as TN along with the task of mapping its hidden features at each layer with the reduced features of TN. Formally, we denote by $\boldsymbol{H}_\ell = [\boldsymbol{h}_1^{(\ell)}, \ldots, \boldsymbol{h}_n^{(\ell)}] \in \mathbb{R}^{p_\ell \times n}$ where $\boldsymbol{h}_i^{(\ell)}$ stands for the features of $\boldsymbol{x}_i$ at layer ℓ. Therefore, a PCA is performed at each layer ℓ in order to reduce the dimension of the hidden features p_ℓ, relying on the top k_ℓ largest eigenvectors of the sample covariance matrix:

$$\boldsymbol{C}_\ell = \frac{1}{n}\sum_{i=1}^{n} \bar{\boldsymbol{h}}_i^{(\ell)} \bar{\boldsymbol{h}}_i^{(\ell)\intercal} \tag{2}$$

where $\bar{\boldsymbol{h}}_i^{(\ell)} = \boldsymbol{h}_i^{(\ell)} - \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{h}_j^{(\ell)}$ are the centred hidden features. We denote by $\boldsymbol{U}_\ell \in \mathbb{R}^{p_\ell \times k_\ell}$ the matrix containing the k_ℓ largest eigenvectors of $\boldsymbol{C}_\ell$. Consequently, the student network (SN) is composed of L layers and has the following structure:

$$\textbf{(SN)}: \begin{cases} \tilde{\boldsymbol{h}}^{(0)} = \boldsymbol{x} \in \mathbb{R}^{p_0}, \\ \tilde{\boldsymbol{h}}^{(\ell)} = f_\ell\left(\tilde{\boldsymbol{W}}^{(\ell)}\tilde{\boldsymbol{h}}^{(\ell-1)} + \tilde{\boldsymbol{b}}^{(\ell)}\right) \in \mathbb{R}^{k_\ell}, \end{cases} \tag{3}$$

with the convention $k_0 = p_0$ and where, $\boldsymbol{x}$ corresponds to the input data features, $\tilde{\boldsymbol{h}}^{(\ell)}$ stands for the features of $\boldsymbol{x}$ extracted at layer ℓ, $\tilde{\boldsymbol{W}}^{(\ell)} \in \mathbb{R}^{k_\ell \times k_{\ell-1}}$ and $\tilde{\boldsymbol{b}}^{(\ell)} \in \mathbb{R}^{k_\ell}$ are respectively the weight matrix and bias at each layer ℓ.

Given the initial learning problem of TN which corresponds to a loss function $\mathcal{L}_{\text{problem}}$, SN is therefore optimized with the following loss function, where the Homoscedastic loss [12] is considered since the optimization problem for SN can be formulated as a multi-task problem.

$$\begin{aligned} \mathcal{L} = {} & e^{-\sigma_{\text{problem}}}\mathcal{L}_{\text{problem}} + \sigma_{\text{problem}} \\ & + \sum_{\ell=1}^{L-1} e^{-\sigma_\ell}\mathcal{L}_{\text{mse}}(\tilde{\boldsymbol{h}}^{(\ell)}, \boldsymbol{U}_\ell^\intercal \boldsymbol{h}^{(\ell)}) + \sigma_\ell \end{aligned} \tag{4}$$

where $\mathcal{L}_{\text{mse}}$ denotes the mean squared error loss function, σ_{problem} and σ_ℓ's are the Homoscedastic loss parameters which are learned during the training of the student network. A full description of the Net-PCAD method is provided as a pseudo-code algorithm in Algorithm 1.

Algorithm 1: Net-PCAD description.

Input: A trained teacher network **TN**, a data matrix $\boldsymbol{X}$ and the learning problem loss $\mathcal{L}_{\text{problem}}$.
Output: Trained student network **SN**.
for $\ell \leftarrow 1$ ***to*** $L-1$ **do**
  1. Extract the representations $\boldsymbol{H}_\ell$ of $\boldsymbol{X}$ from **TN**;
  2. Compute $\boldsymbol{U}_\ell$ through a PCA on $\boldsymbol{H}_\ell$;
end
Train the student network **SN** with $\mathcal{L}$ as in equation 4;

2.3. Neural Nets LDA-based Distillation (Net-LDAD)

If the initial learning problem of the TN is a supervised classification problem, one can take advantage of the fact that the data belong to K different classes $\{\mathcal{C}_j\}_{j=1}^K$ and therefore project the hidden features of the TN in structured low-dimensional spaces. Linear Discriminant Analysis (LDA) is a dimension reduction technique that specifically reduces the dimension of data relying on their classes structure [13]. LDA is closely related to PCA but differs from the latter by the fact that it explicitly attempts to model the difference between the classes of the data, while PCA does not take into account any difference in class labels. Therefore, the idea behind Net-LDAD is to exploit the labels information layer-wise in the training of the student network from the teacher network. Specifically, we compute at each layer ℓ of the TN the within class scatter matrix as:

$$\boldsymbol{S}_w^{(\ell)} = \sum_{j=1}^{K}\sum_{\boldsymbol{x}\in\mathcal{C}_j} (\boldsymbol{h}_{\boldsymbol{x}}^{(\ell)} - \boldsymbol{m}_j^{(\ell)})(\boldsymbol{h}_{\boldsymbol{x}}^{(\ell)} - \boldsymbol{m}_j^{(\ell)})^\intercal \tag{5}$$

where $\boldsymbol{h}_{\boldsymbol{x}}^{(\ell)}$ is the representation of $\boldsymbol{x}$ at layer ℓ of the TN and $\boldsymbol{m}_j^{(\ell)} = \frac{1}{|\mathcal{C}_j|}\sum_{\boldsymbol{x}\in\mathcal{C}_j} \boldsymbol{h}_{\boldsymbol{x}}^{(\ell)}$. And the between class scatter matrix at each layer ℓ is given by:

$$\boldsymbol{S}_b^{(\ell)} = \sum_{j=1}^{K} |\mathcal{C}_j|(\boldsymbol{m}_j^{(\ell)} - \boldsymbol{m}^{(\ell)})(\boldsymbol{m}_j^{(\ell)} - \boldsymbol{m}^{(\ell)})^\intercal \tag{6}$$

Algorithm 2: Net-LDAD description.

Input: A trained teacher network **TN**, a data matrix $\boldsymbol{X}$ and the learning problem loss $\mathcal{L}_{\text{problem}}$.
Output: Trained student network **SN**.
for $\ell \leftarrow 1$ ***to*** $L-1$ **do**
  1. Extract the representations $\boldsymbol{H}_\ell$ of $\boldsymbol{X}$ from **TN**;
  2. Compute $\boldsymbol{V}_\ell$ through a LDA on $\boldsymbol{H}_\ell$;
end
Train the student network **SN** with $\mathcal{L}$ as in equation 7;

where $\boldsymbol{m}^{(\ell)} = \frac{1}{n}\sum_{\boldsymbol{x}\in\boldsymbol{X}} \boldsymbol{h}_{\boldsymbol{x}}^{(\ell)}$. Therefore, the projection matrix of LDA at each layer ℓ is computed as the k_ℓ largest eigenvectors of $(\boldsymbol{S}_w^{(\ell)})^{-1}\boldsymbol{S}_b^{(\ell)}$. We denote by $\boldsymbol{V}_\ell \in \mathbb{R}^{p_\ell \times k_\ell}$ such a projection matrix. Similarly to the PCA case, the student network is therefore trained to minimize the following objective:

$$\mathcal{L} = e^{-\sigma_{\text{classification}}}\mathcal{L}_{\text{classification}} + \sigma_{\text{classification}} + \sum_{\ell=1}^{L-1} e^{-\sigma_\ell}\mathcal{L}_{\text{mse}}(\tilde{\boldsymbol{h}}^{(\ell)}, \boldsymbol{V}_\ell^\intercal \boldsymbol{h}^{(\ell)}) + \sigma_\ell \tag{7}$$

where $\mathcal{L}_{\text{classification}}$ is typically a categorical cross entropy loss function since the initial learning problem of the TN is supposed to be a classification problem. A full description of the Net-LDAD method is provided as a pseudo-code algorithm in Algorithm 2.

3. EXPERIMENTS

In this section, we present experiments which highlight the effectiveness of the proposed methods to train student networks that are smaller in size w.r.t. a given large size dense teacher network. In particular, we consider three teacher networks composed of $L = 4$ dense layers which are trained to perform a classification problem respectively on the datasets MNIST [14], Fashion-MNIST [15] and CIFAR10 [16]. Therefore, we train the corresponding student networks by successively reducing their hidden dimensions using the presented methods Net-PCAD and Net-LDAD. Note that, for simplicity, we reduce all the hidden dimensions to a constant value k and we vary k in all our experiments.

Layer	Teacher	Student
Dense 1	$p_0 \times 1024$	$p_0 \times k$
Dense 2	1024×512	$k \times k$
Dense 3	512×256	$k \times k$
Dense 4	256×10	$k \times 10$

Table 1. Architectures of the teacher and student networks. The dimensions of the weight matrix at each dense layer are shown for both networks.

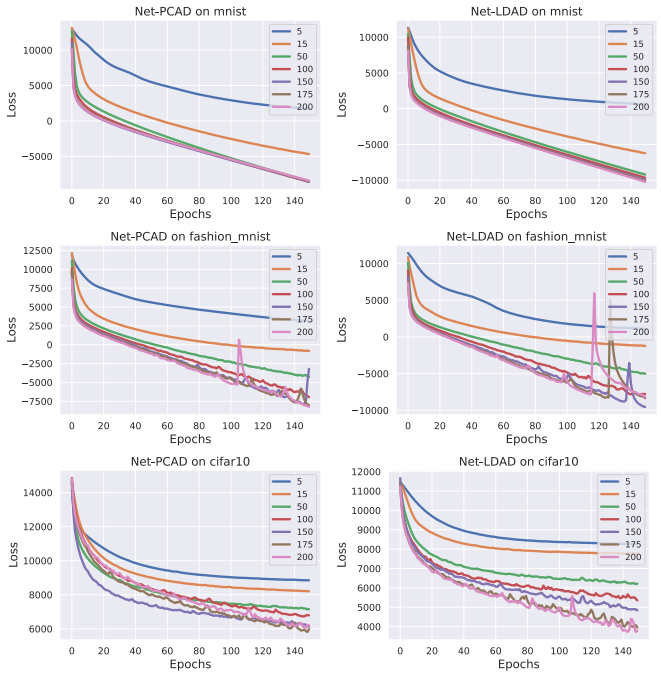

Fig. 1. Training loss of the student network when trained using Net-PCAD **(left)** and Net-LDAD **(right)** in terms of the training epochs, for different values of k, and across three different datasets.

Table 1 presents the considered architectures for the teacher and student networks.

Figure 1 depicts the training Homoscedastic loss of the student networks for different values of k and across the different considered datasets using our methods Net-PCAD and Net-LDAD. Note from this figure that both methods yield generally to a stable learning of the student networks, and the classification problem gets much easier as k increases. However, note that a careful choice of k must be made in order to get a smooth learning loss (e.g., see $k = 200$ on the Fashion-MNIST dataset).

Figure 2 depicts the learned Homoscedastic parameters once the student networks have been trained for different values of k. We can observe from this figure (at least for the datasets MNIST and Fashion-MNIST), that the weight $e^{-\sigma_{\text{classification}}}$ is much larger than the weights on the hidden features for the Net-PCAD method, while all weights have the same order of magnitude for the Net-LDAD method. This can be interpreted by the fact that LDA finds layer wise a low dimensional space where data can be classified and therefore "helps" the classification learning problem. This is not the case regarding the curves for the CIFAR10 dataset, since it is a "hard" classification problem given the architecture of the teacher network (TN gets 45% accuracy).

In terms of the test accuracy, we note from Figure 3 that learning the students networks with our methods improves

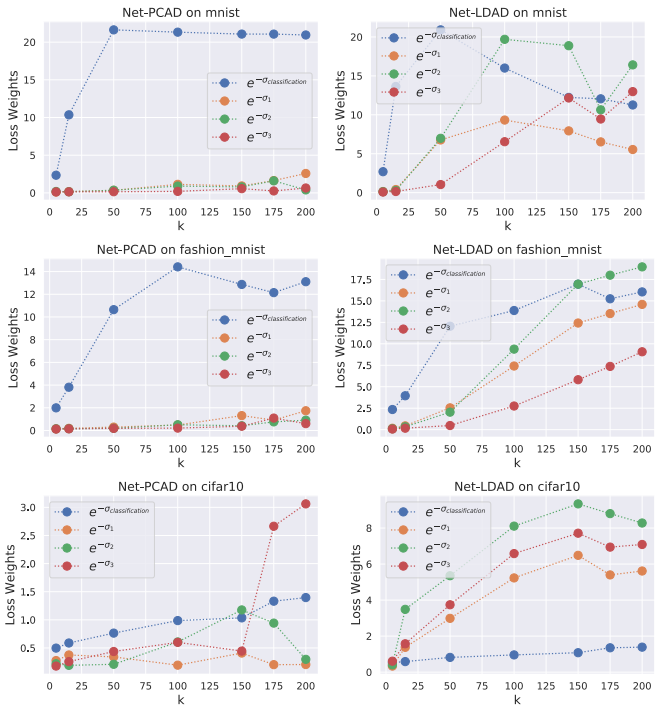

Fig. 2. The learned Homoscedastic loss parameters using Net-PCAD **(left)** and Net-LDAD **(right)** in terms of k and across three different datasets. The weights corresponding to the reduced features mapping loss are of the same order of magnitude for Net-LDAD as the classification loss, which is a consequence of the fact that LDA is classes dependent.

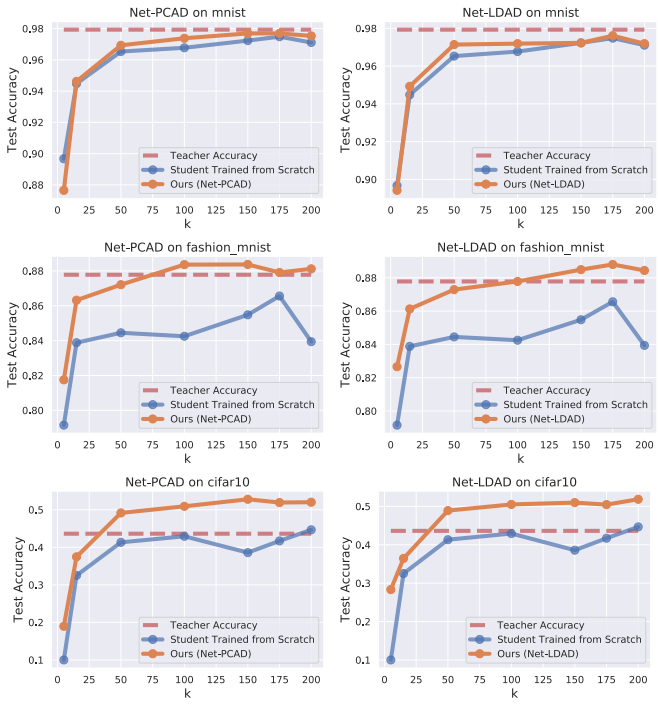

Fig. 3. Test accuracy of the student network in orange trained using Net-PCAD **(left)** and Net-LDAD **(right)**, and test accuracy of the student network in blue trained from scratch, in terms of k and across three different datasets. The test accuracy of the teacher corresponds to the dashed red lines.

largely their generalization capacities compared with learning them from scratch, and one can see that they even surpass the teacher network on the CIFAR10 dataset. As a summary, Net-PCAD and Net-LDAD yield to better generalization performances of the student networks as the learning problem gets harder in the sense of the teacher network test accuracy, knowing that classifying MNIST is an easy problem (TN gets 98%), Fashion-MNIST medium (TN gets 88%) while classifying CIFAR10 is a harder learning problem (TN gets 45%).

		Student		
Dataset	Teacher	$k = 50$	100	200
MNIST	2.23s	0.38s	0.45s	0.65s
	98%	97%	97.5%	97.8%
FASHION	2.23s	0.38s	0.45s	0.65s
	88%	87.5%	88.5%	88.5%
CIFAR10	4.63s	0.75s	0.92s	1.35s
	45%	50%	50.1%	50.3%

Table 2. Forward execution time in seconds (and corresponding test accuracies in %) of the teacher network and the student network for different values of k, the forward pass is applied (on a i7-7700HQ CPU @ 2.80GHz) to the train set of the respective datasets using a batch size of 50000 images.

Table 2 summarizes the performances of the learned student networks using the Net-PCAD method[1], in terms of the forward execution time and test accuracy. As we can notice, Net-PCAD yields to an accurate speedup of inference time (depending on the choice of k) while not degrading the accuracy of the learned student networks w.r.t. the teacher network and even surpassing the teacher network's accuracy in the case of hard classification problems (see CIFAR10).

4. DISCUSSION AND CONCLUSION

The article presented two methods to distillate a given teacher network (TN) into a student network (SN). Our methods improve the performance of SN compared to learning SN from scratch and even surpasses TN performances when the learning problem gets hard, therefore the resulting learned SN is suited to be implemented in an edge IoT device which requires limited resources. Note that the presented methods need to setup an hyper-parameter k_ℓ. For the lack of space, this will be addressed in an extended version of the article.

[1]Net-LDAD gets similar results.

5. REFERENCES

[1] Mahadev Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[2] Ali Alnoman, Shree Krishna Sharma, Waleed Ejaz, and Alagan Anpalagan, "Emerging edge computing technologies for distributed iot systems," *IEEE Network*, vol. 33, no. 6, pp. 140–147, 2019.

[3] MG Murshed, Christopher Murphy, Daqing Hou, Nazar Khan, Ganesh Ananthanarayanan, and Faraz Hussain, "Machine learning at the network edge: A survey," *arXiv preprint arXiv:1908.00080*, 2019.

[4] Lauri Lovén, Teemu Leppänen, Ella Peltonen, Juha Partala, Erkki Harjula, Pawani Porambage, Mika Ylianttila, and Jukka Riekki, "Edge ai: A vision for distributed, edge-native artificial intelligence in future 6g networks," *The 1st 6G Wireless Summit*, pp. 1–2, 2019.

[5] Sahar Voghoei, Navid Hashemi Tonekaboni, Jason G Wallace, and Hamid R Arabnia, "Deep learning at the edge," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2018, pp. 895–901.

[6] Song Han, Huizi Mao, and William J Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[7] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, and Hassan Ghasemzadeh, "Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher," *arXiv preprint arXiv:1902.03393*, 2019.

[8] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4133–4141.

[9] Youssef Tamaazousti, Hervé Le Borgne, Céline Hudelot, Mohamed El Amine Seddik, and Mohamed Tamaazousti, "Learning more universal representations for transfer-learning," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[10] Sara Meftah, Youssef Tamaazousti, Nasredine Semmar, Hassane Essafi, and Fatiha Sadat, "Joint learning of pre-trained and random units for domain adaptation in part-of-speech tagging," *arXiv preprint arXiv:1904.03595*, 2019.

[11] Mohamed El Amine Seddik, Mohamed Tamaazousti, and Romain Couillet, "A kernel random matrix-based approach for sparse PCA," in *International Conference on Learning Representations*, 2019.

[12] Alex Kendall, Yarin Gal, and Roberto Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7482–7491.

[13] Alaa Tharwat, Tarek Gaber, Abdelhameed Ibrahim, and Aboul Ella Hassanien, "Linear discriminant analysis: A detailed tutorial," *AI Commun.*, vol. 30, pp. 169–190, 2017.

[14] LeCun Yann, Cortes Corinna, and J Christopher, "The mnist database of handwritten digits," *URL http://yhann. lecun. com/exdb/mnist*, 1998.

[15] Han Xiao, Kashif Rasul, and Roland Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[16] Alex Krizhevsky and Geoff Hinton, "Convolutional deep belief networks on cifar-10," *Unpublished manuscript*, vol. 40, no. 7, pp. 1–9, 2010.