Neural Networks Classify through the Class-Wise Means of Their Representations

Mohamed El Amine Seddik[1], **Mohamed Tamaazousti**[2]

[1]Huawei Paris Research Center, 92012 Boulogne-Billancourt, France
[2]Université Paris-Saclay, CEA, List, F-91120 Palaiseau, France
mohamed.seddik@huawei.com, mohamed.tamaazousti@cea.fr

Abstract

In this paper, based on an asymptotic analysis of the Softmax layer, we show that when training neural networks for classification tasks, the weight vectors corresponding to each class of the Softmax layer tend to converge to the class-wise means computed at the representation layer (for specific choices of the representation activation). We further show some consequences of our findings to the context of transfer learning, essentially by proposing a simple yet effective initialization procedure that significantly accelerates the learning of the Softmax layer weights as the target domain gets closer to the source one. Experiments are notably performed on the datasets: MNIST, Fashion MNIST, Cifar10, and Cifar100 and using a standard CNN architecture.

Introduction

One of the big trends of the moment is opening up access to the Deep Learning technology to non-experts. Therefore, there is an emergent need to move from manual design of neural networks models by experts to an automatic design for non-experts. This methodological shift is known as AutoML (for Automatic Machine Learning). Technically, this consists in designing automatically the architecture of a neural network as well as its hyperparameters, e.g, the Neural Architecture Search (NAS) approach (Zoph and Le 2017; Zoph et al. 2018; Cai, Zhu, and Han 2019; Liu, Simonyan, and Yang 2019; Real et al. 2019).

Basically, this trend implies to handle more and more specific data domains with generally few data provided by the user for each domain. In this context, the NAS approach, which needs a lot of training data, is no longer relevant. Consequently, the practical solution generally adopted to overcome this problem is to "recycle", as much as possible, the networks already learned, with a minimum of adaptation. This is facilitated by the fact that there are more and more pretrained neural networks available online (open source pretrained models[1]).

To this end, one of the most simple and widespread approach is the transfer of pre-trained representations referred to as Transfer Learning (TL) (Kuzborskij, Orabona, and Caputo 2013; Sharif Razavian et al. 2014; Yosinski et al. 2014; Ginsca et al. 2015; Azizpour et al. 2015; Kornblith, Shlens, and Le 2019; Tamaazousti et al. 2019) in the rest of the document. In principle, TL consists in learning a neural network on a source domain for which we have a large amount of data and then using this network on a target domain (generally with few available data) by updating only the last layer (very often with Softmax activation) of weight associated with the task to be solved (e.g., classification task). Therefore, TL plays a key role in this context by reducing drastically the hyper-parameters space and replacing the neural architecture search by the best available source model/representation search.

Unfortunately, a scale up of Deep Learning based on this TL methodology will be with a huge computational consequences. Indeed, the TL approach implies to duplicate the search for the best hyper-parameters, generally obtained by a very costly grid search, as many times as there are available source models. To the best of our knowledge, there is no clear solution to an *a priori* (i.e., without training the last layer) choice of the best source model from the data associated to the target domain.

To tackle this issue, our **(1)** first contribution consists in analyzing theoretically the behavior of the Softmax layer in neural networks classifiers (see section *Asymptotic analysis of the Softmax classifier weights*). Our **(2)** second contribution consists in showing that Softmax classifies through the class-wise means of the data representations (see section *Expression of the class-weights vectors for near-optimal representations*) for specific choices of the representation activation. This second result is obtained under additional hypotheses on the representations statistics of the data, based on commonly accepted interpretations of the properties expected from the representations of DNNs (Bengio, Courville, and Vincent 2013; Tamaazousti et al. 2019). Our results are experimentally confirmed by showing that when training neural networks for classification tasks, the weight vectors corresponding to each class of the Softmax layer tend to be similar to the class-wise means computed at the representation layer.

Based on this finding, our **(3)** third contribution then consists in addressing the source model selection in the TL context for classification. In addition, this result provides a simple yet effective initialization procedure (through the class-wise means of the representations computed on the target

[1]https://modelzoo.co/

domain) which significantly accelerates TL as the target domain gets closer to the source one. Experiments on real data confirm empirically the relevance of these technical choices (see section *Experiments*). Note that, our second and main contribution might be of independent interest even outside the context of TL. Indeed, our theoretical results also support the remarkable observed performances of the Nearest Class Mean Classifier (Guerriero, Caputo, and Mensink 2018).

Notation: Vectors are represented by boldface lowercase letters and matrices are represented by boldface capital letters. The i-th entry of a vector $\boldsymbol{v}$ is denoted by v_i, and element (i, j) of a matrix $\boldsymbol{M}$ is denoted by $M_{i,j}$. The i-th row and j-th column of $\boldsymbol{M}$ are denoted respectively by $\boldsymbol{M}_{i,:}$ and $\boldsymbol{M}_{:,j}$. Let $[p]$ denote the set of integers $\{1, \dots, p\}$. $\delta_{i,j}$ denotes the Kronecker delta. The notation $\boldsymbol{v}_1 = \boldsymbol{v}_2 + \mathcal{O}_{\|\cdot\|}(u_p)$ means that for any unitary vector $\boldsymbol{u}$, $\boldsymbol{u}^\intercal(\boldsymbol{v}_1 - \boldsymbol{v}_2) = \mathcal{O}(u_p)$. $\mathbf{1}_p$ denotes the vector of size p full of ones. $\|\cdot\|$ denotes the ℓ_2 norm for vectors and the operator norm for matrices. Denote by $\mathrm{Cos}(\boldsymbol{u}, \boldsymbol{v}) \equiv (\boldsymbol{u}/\|\boldsymbol{u}\|)^\intercal (\boldsymbol{v}/\|\boldsymbol{v}\|)$ the cosine similarity between two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$.

Settings

In all the paper we consider the following settings. Let $\mathcal{E}$ denote some data space and $\mathcal{T} = \{(\boldsymbol{d}_i, \ell_i)\}_{i=1}^n$ be a training set containing n data points $\boldsymbol{d}_i$ belonging to one of k different classes $\mathcal{C}_1, \dots, \mathcal{C}_k \subset \mathcal{E}$, where $\ell_i \in [k]$ is the label for data point $\boldsymbol{d}_i$. Let n_j denote the number of training points in class j so that $n = \sum_{j=1}^k n_j$. Let $c : [n] \to [k]$ be the function which returns the class-index of a datum $\boldsymbol{x}_i$ so is defined such that $c(i) = \ell$ if $\boldsymbol{x}_i \in \mathcal{C}_\ell$. We further denote

$$\boldsymbol{x} \equiv \varphi \circ \phi(\boldsymbol{d}\,; \Theta) \in \mathbb{R}^p, \tag{1}$$

the representation of dimension p for $\boldsymbol{d} \in \mathcal{E}$, where $\phi : \mathcal{E} \to \mathbb{R}^p$ is typically implemented by a deep CNN model (with the final layer being a dense layer with a linear activation function) parameterized by Θ, and $\varphi : \mathbb{R} \to \mathbb{R}$ stands for the activation function at the representation layer applied element-wise to $\phi(\boldsymbol{d}\,; \Theta)$. The final class prediction $\tilde{\ell} \in [k]$ is given by a classifier function $\psi : \mathbb{R}^p \to \mathbb{R}^k$, such that $\tilde{\ell} = \arg\max \psi(\boldsymbol{x})$. Generally, ψ is a Softmax classifier with weight matrix $\boldsymbol{W} \in \mathbb{R}^{p \times k}$ and bias vector $\boldsymbol{b} \in \mathbb{R}^k$ so that $\psi(\boldsymbol{x}) = \mathrm{softmax}(\boldsymbol{W}^\intercal \boldsymbol{x} + \boldsymbol{b})$. And the Softmax function is defined for a vector $\boldsymbol{v} \in \mathbb{R}^k$ such that $\mathrm{softmax}(\boldsymbol{v})_i = e^{\boldsymbol{v}_i} / \sum_{j=1}^k e^{\boldsymbol{v}_j}$. We consider in all what follows that the bias vector $\boldsymbol{b}$ is null since its influence is negligible (Kang et al. 2019).

Define the following statistics of the representation layer, for each $\ell \in [k]$

$$\boldsymbol{m}_\ell \equiv \mathbb{E}_{\boldsymbol{d} \in \mathcal{C}_\ell}\left[\boldsymbol{x}\right], \quad \boldsymbol{C}_\ell \equiv \mathbb{E}_{\boldsymbol{d} \in \mathcal{C}_\ell}\left[(\boldsymbol{x} - \boldsymbol{m}_\ell)(\boldsymbol{x} - \boldsymbol{m}_\ell)^\intercal\right], \tag{2}$$

which are the expected class-wise means and covariances of the representations as defined in equation 1. Authors in (Seddik et al. 2020) have shown that the performances of a linear classifier applied on top of the representations $\boldsymbol{x}_i$'s can be quantified through their class-wise means and covariances, namely the quantities $\boldsymbol{m}_\ell$ and $\boldsymbol{C}_\ell$, when the data $\boldsymbol{d}_i$'s are assumed to be generated by a GAN. A result which suggests that the representations behave as Gaussian mixtures for linear classifiers. Let $\boldsymbol{W} = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_k] \in \mathbb{R}^{p \times k}$ where $\boldsymbol{w}_\ell$ are the Softmax classifier weights corresponding to class ℓ. In following, we aim to explicit the relationship between the class-weights $\boldsymbol{w}_\ell$ and the quantities $\boldsymbol{m}_\ell$ and $\boldsymbol{C}_\ell$, when the Softmax classifier is trained on the representations. Namely, assuming a Gaussian Mixture Model on the data representations $\boldsymbol{x}_i$ we give subsequently an implicit expression between $\boldsymbol{w}_\ell$ and the representations statistics (cf. equation 8). And under specific assumptions on $\boldsymbol{m}_\ell$ and $\boldsymbol{C}_\ell$ which we refer to as *the near-optimal representations* assumption, we give a more explicit relationship between $\boldsymbol{w}_\ell$ and the class-wise means $\boldsymbol{m}_\ell$.

Main Results

Preliminaries

Before introducing our results, we give the following technical tool which will be of central interest.

Proposition 1. *Let $\boldsymbol{m} \in \mathbb{R}^p$ a deterministic (mean) vector and $\boldsymbol{C} \in \mathbb{R}^{p \times p}$ a positive semi-definite (covariance) matrix, consider $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{C})$ and let $f : \mathbb{R} \to \mathbb{R}$ be some differentiable function and $\boldsymbol{w} \in \mathbb{R}^p$ a deterministic vector. Then,*

$$\mathbb{E}[f(\boldsymbol{w}^\intercal \boldsymbol{x})\boldsymbol{x}] = \mathbb{E}[f(\boldsymbol{w}^\intercal \boldsymbol{x})]\boldsymbol{m} + \mathbb{E}[f'(\boldsymbol{w}^\intercal \boldsymbol{x})]\boldsymbol{C}\boldsymbol{w}. \tag{3}$$

Proof. See supplementary material. □

Remark 1. *Proposition 1 can be extended beyond the Gaussian assumption when the dimension p is getting large. Indeed, if $\boldsymbol{x} = \boldsymbol{m} + \boldsymbol{C}^{\frac{1}{2}}\boldsymbol{z}$ for some random vector $\boldsymbol{z}$ with random i.i.d. entries with zero mean, unit variance and bounded fourth order moment. As $p \to \infty$, assuming further that $\|\boldsymbol{C}\| = \mathcal{O}(1)$, the result of Proposition 1 can be extended thanks to the central limit theorem under this more general setting and we get $\mathbb{E}[f(\boldsymbol{w}^\intercal \boldsymbol{x})\boldsymbol{x}] = \mathbb{E}[f(\boldsymbol{w}^\intercal \boldsymbol{x})]\boldsymbol{m} + \mathbb{E}[f'(\boldsymbol{w}^\intercal \boldsymbol{x})]\boldsymbol{C}\boldsymbol{w} + \mathcal{O}_{\|\cdot\|}(\|\boldsymbol{w}\|\, p^{-\frac{1}{2}})$.*

Asymptotic Analysis of the Softmax Classifier Weights

Given a set of n labeled representations $(\boldsymbol{x}_1, \boldsymbol{y}^{(1)}), \dots, (\boldsymbol{x}_n, \boldsymbol{y}^{(n)})$ belonging to k classes $\mathcal{C}_1, \dots, \mathcal{C}_k$, where $\boldsymbol{y}^{(i)} \in \mathbb{R}^k$ are classically one-hot encoded vectors such that $y_\ell^{(i)} = 1$ and $y_j^{(i)} = 0$ for $j \neq \ell$ if $\boldsymbol{x}_i \in \mathcal{C}_\ell$. In the following we will consider a generalized expression for the labels given by:

$$y_\ell^{(i)} = \alpha_{c(i)} \frac{|\delta_{\ell, c(i)} - \varepsilon|}{1 + (k-2)\varepsilon}, \tag{4}$$

where $\alpha_{c(i)}$ are hyper-parameters and $\varepsilon > 0$ which will be converging to zero in our analysis, $c(i) \in [k]$ returns the class index of the i-th datum and $\delta_{i,j}$ stands for the Kronecker delta. We will see in the following that a careful choice of these parameters $\alpha_{c(i)}$ can make the Softmax classifier weights independent of the class proportions, which is a desired property for datasets with unbalanced classes also known as the long-tail recognition problem (Kang et al. 2019). In particular, the classical labels are recovered by setting $\alpha_{c(i)} = 1$ and $\varepsilon \to 0$.

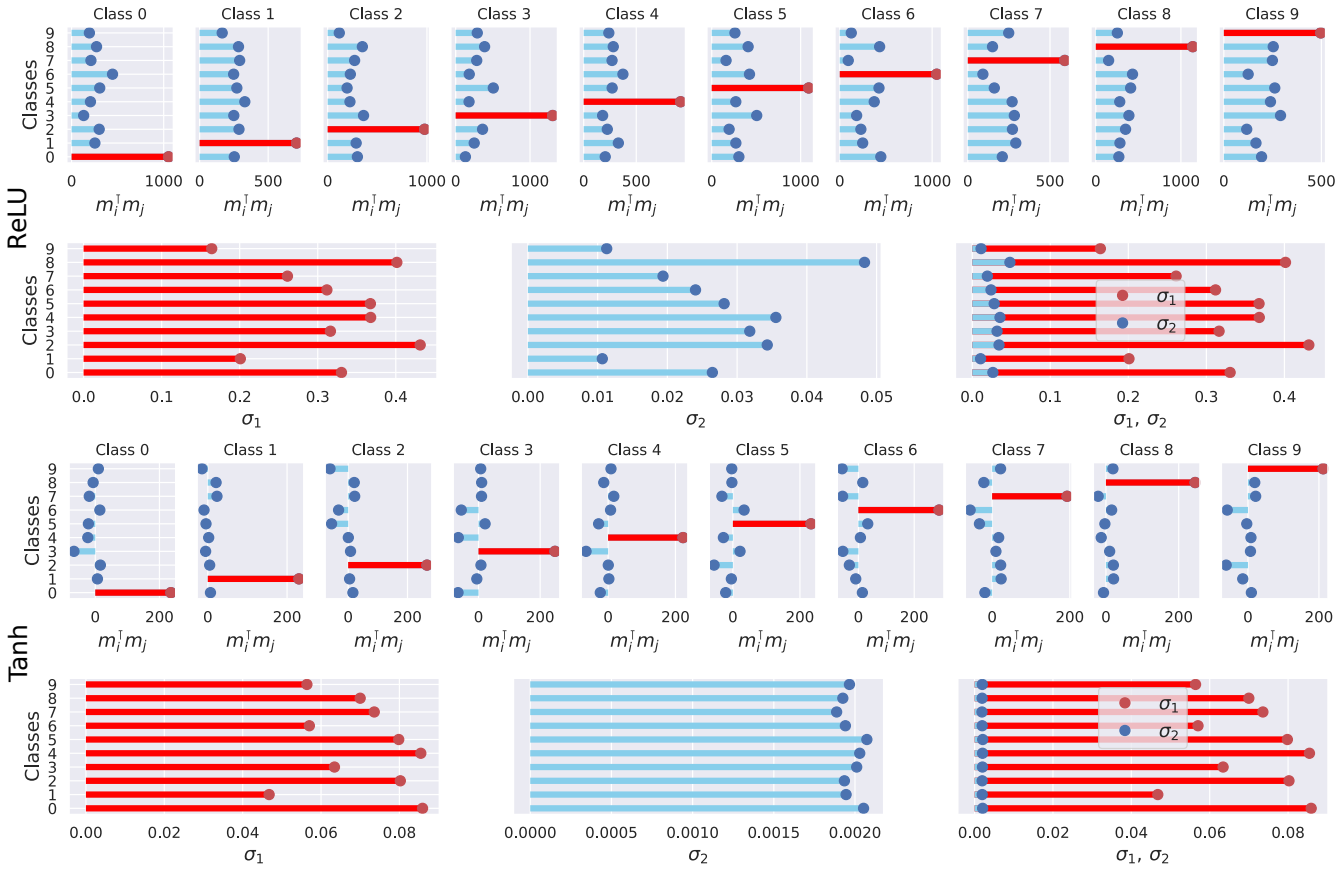

Figure 1: Empirical values of the parameters μ_1, μ_2, $\sigma^2_{1,j}$ and $\sigma^2_{2,j}$ as in Assumption 2. The top figures correspond to a ReLU activation in the representation layer and the bottom figures for a tanh activation. As we can see, Assumption 2 is verified when using the tanh activation. Indeed, the means satisfy $\mu_1 \approx \|\boldsymbol{m}_i\|^2 \gg \boldsymbol{m}_i^\intercal \boldsymbol{m}_j \approx \mu_2$ for $i \neq j$ and the covariance scalars satisfy $\sigma^2_{1,j} \gg \sigma^2_{2,j}$. While these conditions are not satisfied when using the ReLU activation (for which $\mu_1 \approx \mu_2$ and $\sigma_1^{\text{ReLU}} \approx 10 \times \sigma_1^{\tanh}$). The CNN model has been trained on the MNIST dataset, and when using both activations (tanh or ReLU) we get the same test accuracy 99.6%.

The Softmax classification task consists in optimizing the categorical-cross entropy loss function

$$\mathcal{L}(\boldsymbol{w}_1, \dots, \boldsymbol{w}_k) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{\ell=1}^{k} y_\ell^{(i)} \log p_\ell^{(i)} \tag{5}$$

where

$$p_\ell^{(i)} = \frac{e^{\boldsymbol{w}_\ell^\intercal \boldsymbol{x}_i}}{\sum_{j=1}^{k} e^{\boldsymbol{w}_j^\intercal \boldsymbol{x}_i}} \tag{6}$$

The gradient of which with respect to each class-weight vector $\boldsymbol{w}_\ell$ is given by

$$\begin{aligned} \nabla_{\boldsymbol{w}_\ell} \mathcal{L} &= \frac{1}{n} \sum_{i=1}^{n} \alpha_{c(i)} \left(\frac{e^{\boldsymbol{w}_\ell^\intercal \boldsymbol{x}_i}}{\sum_{j=1}^{k} e^{\boldsymbol{w}_j^\intercal \boldsymbol{x}_i}} - \frac{|\delta_{\ell,c(i)} - \varepsilon|}{1 + (k-2)\varepsilon} \right) \boldsymbol{x}_i \\ &\equiv \frac{1}{n} \sum_{i=1}^{n} \alpha_{c(i)} f_{\ell,i}(\boldsymbol{w}_\ell^\intercal \boldsymbol{x}_i) \boldsymbol{x}_i. \end{aligned} \tag{7}$$

Expression of the class-weights vectors under a GMM model In order to characterize the behavior of the weight vectors $\boldsymbol{w}_\ell$, we need to assume a statistical prior on the data representations. Motivated by the result of (Seddik et al. 2020) which shows that deep learning representations tend to behave as Gaussian mixtures when their dimension grow, we make the assumption that the $\boldsymbol{x}_i$ follow a Gaussian Mixture Model (GMM) as follows.

Assumption 1 (Statistical model on the representations)**.** *We assume that the data representations are independent random vectors which follow a GMM with means $\boldsymbol{m}_\ell$ and covariances $\boldsymbol{C}_\ell$ for each class $\mathcal{C}_\ell$, further we denote by $\pi_\ell \equiv \lim_n \frac{n_\ell}{n} \in (0,1)$ where n_ℓ stands for the cardinality of the class $\mathcal{C}_\ell$. Formally, if $\boldsymbol{x}_i \in \mathcal{C}_\ell$ then $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{m}_\ell, \boldsymbol{C}_\ell)$. We further assume that both $p, n \to \infty$.*

Under assumption 1, the loss function in equation 5 is a sum of i.i.d. random variables and therefore converges to its expectation for sufficiently large n thanks to the central limit theorem. Therefore, the class-weights vectors $\boldsymbol{w}_\ell$ con-

centrate around some deterministic vectors $\bar{\boldsymbol{w}}_\ell$ which are defined as $\mathbb{E}[\nabla_{\bar{\boldsymbol{w}}_\ell}\mathcal{L}] = 0$ since the expectation and the gradient are linear operators. The following proposition provides an implicit expression for the asymptotic class-weights vectors $\bar{\boldsymbol{w}}_\ell$.

Proposition 2 (Asymptotic class-weight vectors)**.** *Let $\bar{\boldsymbol{w}}_1, \dots, \bar{\boldsymbol{w}}_k$ be the deterministic vectors satisfying $\mathbb{E}[\nabla_{\bar{\boldsymbol{w}}_\ell}\mathcal{L}] = 0$. Thus, under Assumption 1, each $\bar{\boldsymbol{w}}_\ell$ satisfies the implicit equation*

$$\bar{\boldsymbol{w}}_\ell = \left(\sum_{j=1}^{k} \alpha_j \pi_j \mathbb{E}_j[f'_{\ell,i}(\bar{\boldsymbol{w}}_\ell^\intercal \boldsymbol{x}_i)]\boldsymbol{C}_j\right)^{-1} \left(\sum_{j=1}^{k} \alpha_j \pi_j \mathbb{E}_j[f_{\ell,i}(\bar{\boldsymbol{w}}_\ell^\intercal \boldsymbol{x}_i)]\boldsymbol{m}_j\right) \tag{8}$$

where the notation $\mathbb{E}_j[g(\boldsymbol{x}_i)] \equiv \mathbb{E}[g(\boldsymbol{x}_i) \mid \boldsymbol{x}_i \in \mathcal{C}_j]$ for some $g : \mathbb{R} \to \mathbb{R}$.

Proof. The result is straightforward thanks to Proposition 1. Indeed,

$$\mathbb{E}[\nabla_{\boldsymbol{w}_\ell}\mathcal{L}] = \sum_{j=1}^{k} \frac{n_j}{n} \alpha_j \mathbb{E}_j[f_{\ell,i}(\boldsymbol{w}_\ell^\intercal \boldsymbol{x}_i)]\boldsymbol{m}_j + \sum_{j=1}^{k} \frac{n_j}{n} \alpha_j \mathbb{E}_j[f'_{\ell,i}(\boldsymbol{w}_\ell^\intercal \boldsymbol{x}_i)]\boldsymbol{C}_j \boldsymbol{w}_\ell.$$

Finally, letting $\mathbb{E}[\nabla_{\boldsymbol{w}_\ell}\mathcal{L}] = 0$ completes the proof. □

Remark 2 (Relaxation of the Gaussian assumption)**.** *Note, thanks to Remark 1, that the Gaussian assumption can be relaxed and one may assume that $\boldsymbol{x}_i = \boldsymbol{m}_\ell + \boldsymbol{C}_\ell^{\frac{1}{2}} \boldsymbol{z}_i$ with $\boldsymbol{z}_i$ having i.i.d. entries with zero mean, unit variance an bounded fourth order moment. In which case, we have*

$$\bar{\boldsymbol{w}}_\ell = \left(\sum_{j=1}^{k} \alpha_j \pi_j \mathbb{E}_j[f'_{\ell,i}(\bar{\boldsymbol{w}}_\ell^\intercal \boldsymbol{x}_i)]\boldsymbol{C}_j\right)^{-1} \left(\sum_{j=1}^{k} \alpha_j \pi_j \mathbb{E}_j[f_{\ell,i}(\bar{\boldsymbol{w}}_\ell^\intercal \boldsymbol{x}_i)]\boldsymbol{m}_j\right) + \mathcal{O}_{\|\cdot\|}\left(\|\bar{\boldsymbol{w}}_\ell\| \, p^{-\frac{1}{2}}\right).$$

Looking carefully at the expression in equation 8, we see that in order to make the weight vectors $\bar{\boldsymbol{w}}_\ell$ independent of the class proportions π_j which is a desirable property for the long-tail recognition problem (Kang et al. 2019), a natural choice of the parameters α_j's in the expression of the generalized labels in equation 4 is such that $\alpha_j = (k\,\pi_j)^{-1}$. We hence have the following corollary to Proposition 2.

Corollary 1. *Recalling Proposition 2, setting the weight labels α_j's such that $\alpha_j = (k\,\pi_j)^{-1}$. The class-weight vectors become asymptotically independent of the class proportions π_j. Specifically,*

$$\bar{\boldsymbol{w}}_\ell = \left(\sum_{j=1}^{k} \mathbb{E}_j[f'_{\ell,i}(\bar{\boldsymbol{w}}_\ell^\intercal \boldsymbol{x}_i)]\boldsymbol{C}_j\right)^{-1} \left(\sum_{j=1}^{k} \mathbb{E}_j[f_{\ell,i}(\bar{\boldsymbol{w}}_\ell^\intercal \boldsymbol{x}_i)]\boldsymbol{m}_j\right). \tag{9}$$

Corollary 1 provides a general implicit expression for the class-weight vectors in function of the representations statistics namely the means $\boldsymbol{m}_j$ and the covariances $\boldsymbol{C}_j$. In the following, we will provide an explicit expression for the class-weights $\boldsymbol{w}_\ell$ under specific assumptions on the representations statistics which we refer to as the case of *near-optimal representations*.

Expression of the class-weights vectors for near-optimal representations

Assumption 2 (Near-optimal representations)**.** *Let $\epsilon > 0$, as $p \to \infty$*

- *A1) $\boldsymbol{m}_i^\intercal \boldsymbol{m}_j = \delta_{i,j}\,\mu_1 + (1 - \delta_{i,j})\,\mu_2$ with $\mu_1 = \mathcal{O}(1)$ and $\mu_2 = \mathcal{O}(p^{-\epsilon})$.*
- *A2) $\boldsymbol{C}_j = \sigma_{1,j}^2 \boldsymbol{I}_p + \sigma_{2,j}^2 \left(\mathbf{1}_p \mathbf{1}_p^\intercal - \boldsymbol{I}_p\right)$ with $\sigma_{1,j}^2 = \mathcal{O}\left(p^{-\epsilon}\right)$ and $\sigma_{2,j}^2 = \mathcal{O}(p^{-1-3\epsilon})$.*

An optimal representation for a given dataset is typically a representation which perfectly separates the different classes by maximizing the between-class variance and minimizing the within-class variance while having a large number of degrees of freedom (Bengio, Courville, and Vincent 2013; Tamaazousti et al. 2019). Assumption 2 makes a formal definition for what we call a *near-optimal representation* which satisfies for a sufficiently large number of degrees of freedom (dimension p of the representation) two main properties: *(A1)* which ensures that the between-class means are asymptotically orthogonal; $\boldsymbol{m}_i^\intercal \boldsymbol{m}_j \to \delta_{i,j}\mu_1$ (maximize the between-class variance); and *(A2)* which makes the within-class covariances asymptotically isotropic; $\boldsymbol{C}_j \cong \sigma_{1,j}^2 \boldsymbol{I}_p$ at the first order (which models the fact that an optimal representation should have independent features) and with low variance; $\sigma_{1,j}^2 \to 0$ (minimize the within-class variance). The term *near-optimal* is used in the sense that these properties hold as the dimension p grows. We will see in the experiments part that when training jointly the representation and the classification parts of a neural network classifier (for a specific choice of the representation activation function), its representations tend to satisfy the near-optimal representations assumption. Under this assumption, we have the following proposition which gives an explicit relationship between the class-weights vectors $\bar{\boldsymbol{w}}_\ell$ and the class-wise means.

Proposition 3 (Expression of the weights for near-optimal representations)**.** *Under Assumption 2, for sufficiently large p and letting $\varepsilon \to 0$ in the expression of the generalized labels in equation 4, the class-weight vectors are asymptotically proportional to the centred class-wise means as follows: Let $\kappa > 0$ and $\gamma_\ell =$*

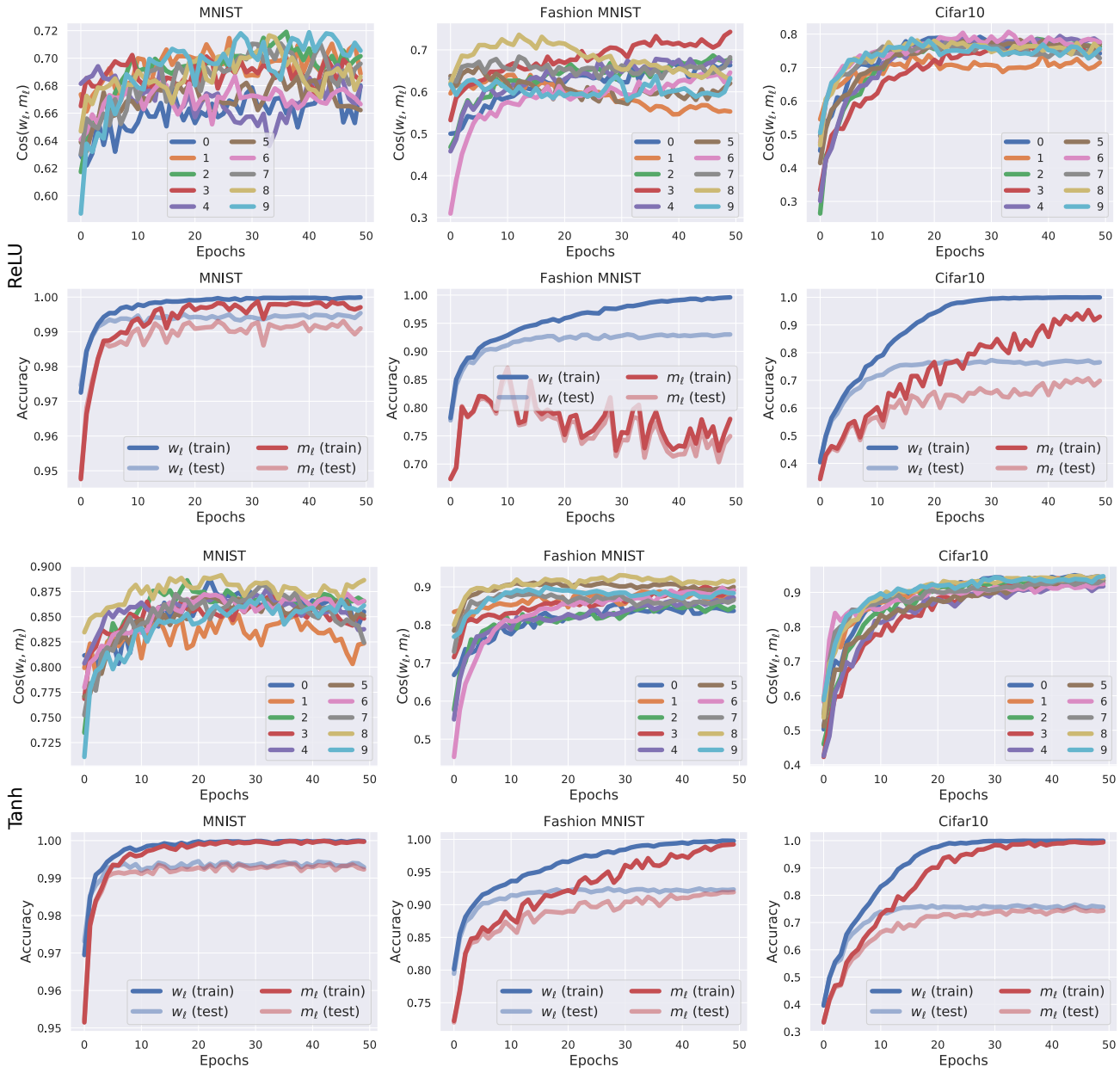

Figure 2: The first and third rows show, respectively for ReLU and $\tanh$ activation at the representation, the cosine similarity during the training between the learned weights $\boldsymbol{w}_\ell$ and the centred class-wise means $\bar{\boldsymbol{m}}_\ell$ for different classes and across different datasets (columns). The second and fourth rows show the train and test accuracy during training, when considering the learned weights (in blue) and the centred class-wise means (in red). As we can see, the cosine similarity between $\boldsymbol{w}_\ell$ and $\bar{\boldsymbol{m}}_\ell$ for $\tanh$ is higher than ReLU since Assumption 2 holds when using $\tanh$ (see Figure 1).

$$\left(\sum_{j=1}^{k} \sigma_{1,j}^2 \mathbb{E}_j[-f'_{\ell,i}(\bar{\boldsymbol{w}}_\ell^\intercal \boldsymbol{x}_i)]\right)^{-1} \geq \frac{4}{k\,\sigma_{1,max}^2}$$

$$\bar{\boldsymbol{w}}_\ell = \frac{\gamma_\ell\, k\, e^{-\kappa\,\mu_1}}{1+(k-1)\,e^{-\kappa\,\mu_1}} \left(\boldsymbol{m}_\ell - \frac{1}{k}\sum_{j=1}^{k} \boldsymbol{m}_j\right) + \mathcal{O}_{\|\cdot\|}\,(1)$$

Proof. See supplementary material. □

Proposition 3 provides an explicit link between $\bar{\boldsymbol{w}}_\ell$ and the class-wise means of the representations when the latter satisfy the near-optimal assumption. Note that since $\gamma_\ell \geq \frac{4}{k\,\sigma_{1,\max}^2} = \mathcal{O}(p^\epsilon)$, the $\bar{\boldsymbol{w}}_\ell$'s are equal at the first order to the (scaled) centred class-wise means, as a consequence, the cosine similarity between them is asymptotically high as in the following corollary.

Corollary 2. *Denote the centred class-wise means by*

$\bar{\boldsymbol{m}}_\ell \equiv \boldsymbol{m}_\ell - k^{-1}\sum_{j=1}^k \boldsymbol{m}_j$ *and the scaling factor* $\varrho_\ell = \frac{\gamma_\ell\, k\, e^{-\kappa\,\mu_1}}{1+(k-1)\,e^{-\kappa\,\mu_1}}$, *therefore* $\mathrm{Cos}(\varrho_\ell^{-1}\bar{\boldsymbol{w}}_\ell, \bar{\boldsymbol{m}}_\ell) = 1 - \mathcal{O}(p^{-\epsilon})$.

We now turn to the experiments to show the application of this result to actual neural network architecture applied to the classification of image datasets and present the consequences to TL.

Experiments

In all the experiments we consider the following settings.

Datasets. All the experiments performed on four datasets which are: MNIST (LeCun 1998), Fashion MNIST (Xiao, Rasul, and Vollgraf 2017), Cifar10 (Krizhevsky and Hinton 2010) and Cifar100 (Krizhevsky, Nair, and Hinton 2009). Each contains about $n = 60000$ training images and 10000 test images belonging to $k = 10$ classes, except Cifar100 which contains $n = 50000$ training images belonging to $k = 100$ classes. All the images are normalized in $[0, 1]$.

Architecture. We consider a CNN architecture for the representation $\phi(\boldsymbol{d}\,;\Theta)$. The dimension of the representation is taken to be $p = 512$ and the activation function φ is either ReLU or $\tanh$, and other choices of the activation are considered in the experiments part of the supplementary material. The classification layer is a dense layer with Softmax activation and without biases since their role is negligible (Kang et al. 2019). We use the Keras (Géron 2019) deep learning framework for the implementation. More details about the architecture are provided in the supplementary material.

Training parameters. Four CNN networks having the above architecture are trained on the four considered datasets for 50 epochs (except Cifar100 with 100 epochs) and using a batch-size of 1000 images, using the Adam optimizer (Kingma and Ba 2014). $\boldsymbol{m}_\ell$ and $\boldsymbol{C}_\ell$ are estimated by their empirical estimates.

Near-optimal Representations in Practice

Under a specific choice of the activation function φ at the representation layer, it turns out that Assumption 2 can hold once the network has been trained. Figure 1 depicts the empirical values of the parameters μ_1, μ_2, $\sigma_{1,j}^2$ and $\sigma_{2,j}^2$ for two choices of the activation function at the representation layer. Here we considered the MNIST dataset and for each class, the means are estimated through their empirical estimates and the parameters $\sigma_{1,j}^2$, $\sigma_{2,j}^2$ are estimated respectively by the average of the diagonal and off-diagonal entries of the sample covariance matrix corresponding to class j.

As we can see in Figure 1, when considering the ReLU activation function at the representation layer, the parameters $\mu_1 \approx \|\boldsymbol{m}_i\|^2$ and $\mu_2 \approx \boldsymbol{m}_i^\intercal \boldsymbol{m}_j$ for $i \neq j$ are of the same order which is not consistent with Assumption 2. Besides, when considering the $\tanh$ activation function, $\mu_1 \gg \mu_2$ and $\sigma_{1,j}^2 \gg \sigma_{2,j}^2$ which agrees with the near-optimal representation assumption. Therefore, when using the ReLU activation function the classes-separability information is shared between the class-wise means and covariances, while in the case of $\tanh$ all the information is encoded in the class-wise means. As a consequence, we will see in the next section that, when considering the $\tanh$ activation, the weights vectors of the Softmax classifier get aligned with the centred class-wise means as suggested by Corollary 2 under the near-optimal representations assumption.

The fact that we observe a difference in the behavior between different choices of the activation function at the representation layer may be associated to the symmetry properties of the considered activation function. Indeed, $\tanh$ is an odd function while ReLU does not have any symmetries, a deeper analysis of the interplay of the non-linearity and the data statistics has been done in (Liao and Couillet 2018) which provides insights in the case of random features maps. More examples as in Figure 1 using different activation functions and datasets are provided in the supplementary material.

Note however that the choice of the activation function at the representation layer does not affect much the performances of the model. Indeed, in the case of the MNIST dataset in Figure 1, the model has the same test accuracy of 99.6% for both choices of the activation function (ReLU or $\tanh$). The same property hold for the datasets Fashion MNIST, Cifar10 and Cifar100 as provided in the supplementary material.

Similarity Between $\bar{\boldsymbol{w}}_\ell$ and $\bar{\boldsymbol{m}}_\ell$

Corollary 2 states that the cosine similarity between the learned $\boldsymbol{w}_\ell$ and $\bar{\boldsymbol{m}}_\ell$ is, at a scaling factor, asymptotically high when the representations statistics $\boldsymbol{m}_\ell$ and $\boldsymbol{C}_\ell$ satisfy the near-optimal representations assumption 2. As we discussed in the previous section, this assumption can be verified when the CNN model is trained for a specific choice of the representation activation φ. Figure 2 shows the empirically computed cosine similarities $\mathrm{Cos}(\boldsymbol{w}_\ell, \bar{\boldsymbol{m}}_\ell)$ during training for both activations ReLU and $\tanh$. As we can notice, $\mathrm{Cos}(\boldsymbol{w}_\ell, \bar{\boldsymbol{m}}_\ell)$ is relatively high when considering the $\tanh$ activation since it yields to near-optimal representations as assumed in 2.

We also provide the curves of accuracy during training of the CNN model (in blue) along with the accuracy when replacing the weight matrix $\boldsymbol{W}$ of the Softmax layer by the normalized matrix $\bar{\boldsymbol{M}}$ containing the centred class-wise means $\bar{\boldsymbol{m}}_\ell$ (in red). Precisely, $\bar{\boldsymbol{M}} = \boldsymbol{M}\boldsymbol{P}/\|\boldsymbol{M}\boldsymbol{P}\|$ with $\boldsymbol{M} = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k]$ and $\boldsymbol{P}$ stands for the centring matrix of dimension k and is given by $\boldsymbol{P} = \boldsymbol{I}_k - \frac{1}{k}\mathbf{1}_k\mathbf{1}_k^\intercal$. The normalization by $\|\boldsymbol{M}\boldsymbol{P}\|$ is considered since the scaling factor ϱ_ℓ depends implicitly on $\boldsymbol{w}_\ell$, but still one can use spectral normalization (Miyato et al. 2018) as we considered here by the term $\|\boldsymbol{M}\boldsymbol{P}\|$.

We can see in Figure 2 that, for $\varphi = \tanh$, there is a matching between the train and test accuracy when using the default weights $\boldsymbol{W}$ and when replacing them by $\bar{\boldsymbol{M}}$ at the end of the training phase. Therefore, this result suggests that the weight matrix $\boldsymbol{W}$ encode the class-wise means of the representations, and hence the neural network classifier make decisions through the $\bar{\boldsymbol{m}}_\ell$'s. The similarity between $\bar{\boldsymbol{w}}_\ell$ and $\bar{\boldsymbol{m}}_\ell$ notably supports the remarkable observed performances of the Nearest Class Mean Classifier (Guerriero, Caputo, and Mensink 2018), which relies on classifying through the euclidean distances to the class-wise means.

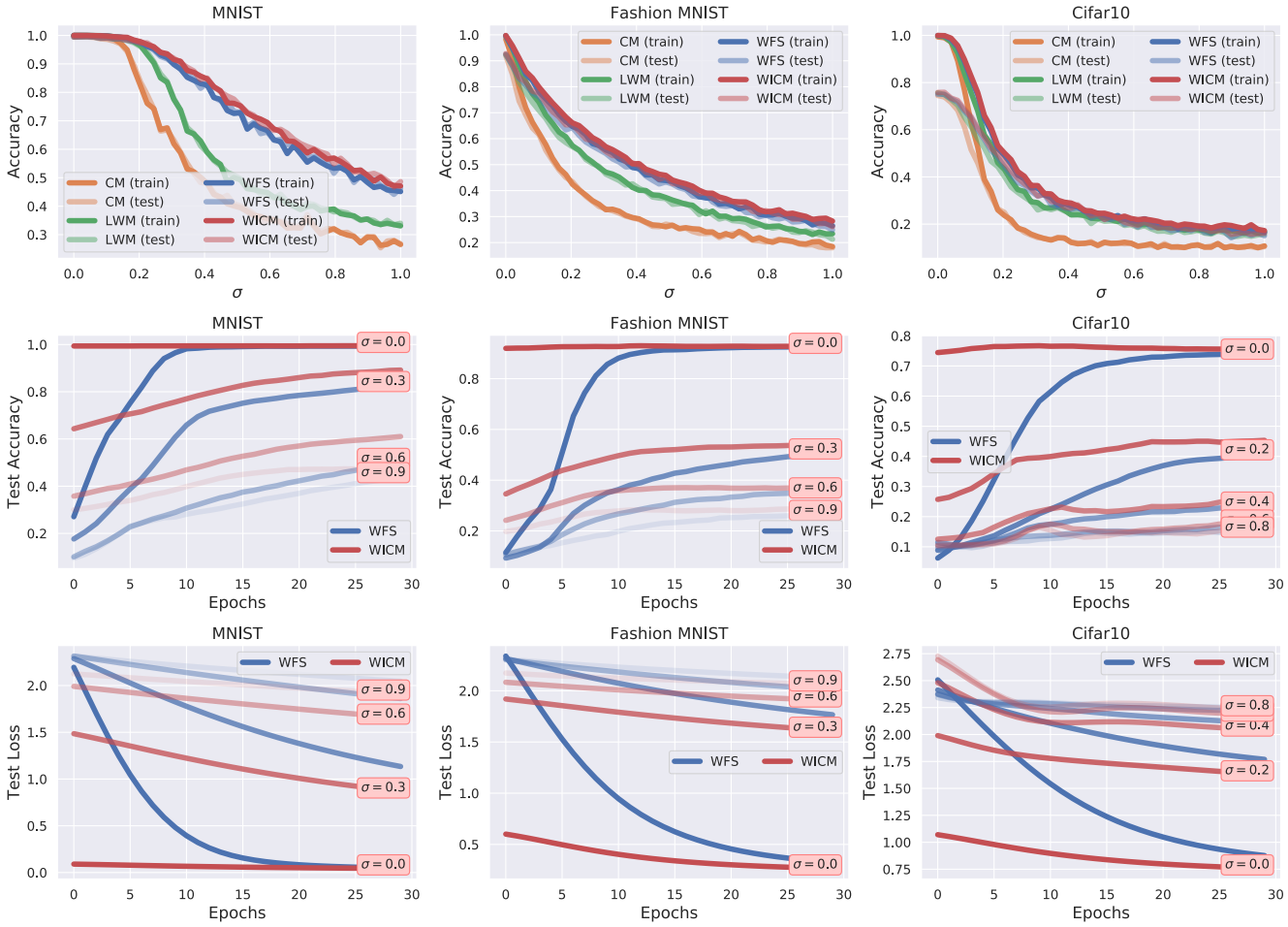

Figure 3: Training of the Softmax layer on target data. The target data ($n = 5000$ sampled at random) being noisy versions of the considered datasets with different Gaussian noise variance σ^2. Accuracy (Top) and dynamics (Bottom) of different baselines in terms of the added noise variance σ^2. The different methods are: CM (Centred Means) the weights are replaced with the scaled centred means; LWM (Learned Weights of Means) a learned weighted sum of means; WFS (Weights Learned From Scratch); WICM (Weights Initialized with Centred Means).

Note that the performance of the CNN model for $\varphi = \mathrm{ReLU}$ is the same as when considering $\varphi = \tanh$, but the latter has the advantage to be more interpretable through Corollary 2.

Consequences for Transfer Learning

As a consequence of our finding in the previous section, in a deep transfer learning scheme (Bengio, Courville, and Vincent 2013; Tamaazousti et al. 2019), one can initialize the weights of the Softmax layer with the scaled centred class-wise means $\bar{\boldsymbol{M}}$ of the target data representations. To stress this point, we first train a network on the used datasets considered as source domain and the target domain is considered as noisy versions of the input data ($n = 5000$ randomly samples) by adding Gaussian noise with variance σ^2. We particularly compare three methods which are: CM (Centred Means) the weights are replaced with the scaled centred means $\bar{\boldsymbol{M}}$ without further training; LWM (Learned Weights of Means) a learned weighted sum of means $\boldsymbol{MW}$ with $\boldsymbol{W}$ a $k \times k$ weight matrix to be trained; WFS (Weights Learned From Scratch); WICM (Weights Initialized with Centred Means).

Figure 3 depicts the accuracy on the target domain (along the different datasets) in terms of the noise variance σ^2. We particularly notice from this figure that when the weights of the Softmax layer are initialized with the class-wise means of the representations (WICM), we get a slightly better accuracy than initializing them randomly (WFS). Moreover, internalizing with the class-wise means yield to faster convergence as the target domain gets closer to the source one, i.e., as $\sigma \to 0$. Besides, the source model selection can be performed by comparing the accuracies, of different representations, of the Softmax layer with the weights being the centred class-wise means of the representations (CM method in Figure 3), thereby not requiring the training of the Softmax layer multiple times.

Conclusion

We have analyzed throughout the paper the behavior of the Softmax layer in neural networks classifiers. We have notably shown that the weight vectors corresponding to each class of the Softmax layer tend to converge to the class-wise means computed at the representation layer, a result which unfolds from the near-optimal representations assumption. Our findings suggested three main procedures for efficient transfer learning: (i) use of symmetric representation activations to ensure the near-optimal representations assumption; (ii) we provided a systematic approach to perform source model selection without training the Softmax layer; (iii) and finally we provided an initialization procedure which accelerates the training of the Softmax layer as the target domain gets closer to the source domain.

References

Azizpour, H.; Razavian, A. S.; Sullivan, J.; Maki, A.; and Carlsson, S. 2015. Factors of transferability for a generic convnet representation. *IEEE transactions on pattern analysis and machine intelligence*, 38(9): 1790–1802.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828.

Cai, H.; Zhu, L.; and Han, S. 2019. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*.

Géron, A. 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

Ginsca, A. L.; Popescu, A.; Le Borgne, H.; Ballas, N.; Vo, P.; and Kanellos, I. 2015. Large-scale image mining with flickr groups. In *International Conference on Multimedia Modeling*, 318–334. Springer.

Guerriero, S.; Caputo, B.; and Mensink, T. 2018. Deep nearest class mean classifiers. In *International Conference on Learning Representations, Worskhop Track*.

Kang, B.; Xie, S.; Rohrbach, M.; Yan, Z.; Gordo, A.; Feng, J.; and Kalantidis, Y. 2019. Decoupling Representation and Classifier for Long-Tailed Recognition. arXiv:1910.09217.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kornblith, S.; Shlens, J.; and Le, Q. V. 2019. Do better imagenet models transfer better? In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2661–2671.

Krizhevsky, A.; and Hinton, G. 2010. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7): 1–9.

Krizhevsky, A.; Nair, V.; and Hinton, G. 2009. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/-cifar. html*, 6.

Kuzborskij, I.; Orabona, F.; and Caputo, B. 2013. From n to n+ 1: Multiclass transfer incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3358–3365.

LeCun, Y. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Liao, Z.; and Couillet, R. 2018. On the spectrum of random features maps of high dimensional data. *arXiv preprint arXiv:1805.11916*.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. Darts: Differentiable architecture search. *ICLR*.

Miyato, T.; Kataoka, T.; Koyama, M.; and Yoshida, Y. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, 4780–4789.

Seddik, M. E. A.; Louart, C.; Tamaazousti, M.; and Couillet, R. 2020. Random matrix theory proves that deep learning representations of gan-data behave as gaussian mixtures. In *International Conference on Machine Learning*, 8573–8582. PMLR.

Sharif Razavian, A.; Azizpour, H.; Sullivan, J.; and Carlsson, S. 2014. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 806–813.

Tamaazousti, Y.; Le Borgne, H.; Hudelot, C.; Seddik, M. E. A.; and Tamaazousti, M. 2019. Learning more universal representations for transfer-learning. *IEEE transactions on pattern analysis and machine intelligence*, 42(9): 2212–2224.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, 3320–3328.

Zoph, B.; and Le, Q. V. 2017. Neural architecture search with reinforcement learning. *ICLR*.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8697–8710.