Partial Least Squares: A Deep Space Odyssey

Artur Jordão[§] and William Robson Schwartz
Federal University of Minas Gerais, Brazil
Email: {arturjordao, william}@dcc.ufmg.br

***Abstract*—Modern visual pattern recognition models are based on deep convolutional networks. Such models are computationally expensive, hindering applicability on resource-constrained devices. To handle this problem, we propose three strategies. The first removes unimportant structures (neurons or layers) of convolutional networks, reducing their computational cost. The second inserts structures to design architectures automatically, enabling us to build high-performance networks. The third combines multiple layers of convolutional networks, enhancing data representation at negligible additional cost. These strategies are based on Partial Least Squares (PLS) which, despite promising results, is infeasible on large datasets due to memory constraints. To address this issue, we also propose a discriminative and low-complexity incremental PLS that learns a compact representation of the data using a single sample at a time, thus enabling applicability on large datasets. We assess the effectiveness of our approaches on several convolutional architectures and computer vision tasks, which include image classification, face verification and activity recognition. Our approaches reduce the resource overhead of both convolutional networks and Partial Least Squares, promoting energy- and hardware-friendly models for the academy and industry scenarios. Compared to state-of-the-art methods for the same purpose, we obtain one of the best trade-offs between predictive ability and computational cost.**

I. Introduction

Pattern recognition plays an important role in cognitive tasks such as natural language processing and image understanding. Modern pattern recognition methods have led to a series of breakthroughs, often surpassing human performance [1]. The reason for these remarkable achievements is the improvement in data representation (i.e., features), which allows discovering new abstractions and patterns from data.

In the context of visual pattern recognition, deep convolutional networks have been the focus of intense research due to their state-of-the-art effectiveness in learning discriminative representation. In particular, most efforts have been devoted to the development of architectures for convolutional networks, since large architectures are a major determinant factor for improving their predictive ability (see Figure 1) [2]. In terms of performance, on the other hand, excessively large architectures are computationally expensive, hindering applicability on low-power and internet of things (IoT) devices. Moreover, such architectures are *data-hungry*, meaning that large datasets are needed to provide a better generalization performance [3], hence, the encouragement for large datasets has been growing.

A parallel line of research to obtain discriminative representations is to discover low-dimensional features through dimensionality reduction techniques. Such techniques are capable of yielding discriminative and compact representations from the original (high-dimensional) data [4]. Recent works use dimensionality reduction collaboratively with convolutional networks and produce encouraging results [5], [6]. This combination, however, is unsuitable for large datasets since traditional dimensionality reduction techniques require all the data to be in memory in advance, which is often impractical due to hardware limitations. Additionally, this requirement prevents us from employing dimensionality reduction on streaming applications, where the data are being generated continuously.

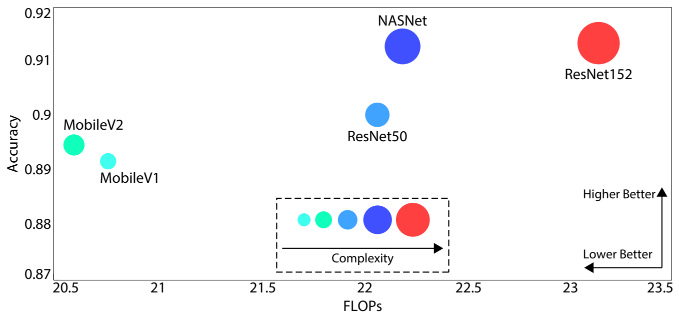

Fig. 1. Comparison of convolutional networks in terms of predictive ability, computational cost, and complexity. Predictive ability is measured by accuracy. Computational cost is measured by Floating Point Operations (FLOPs). Complexity is measured taking into account the number of neurons (width) and layers (depth), and it is represented by the circle size (larger means more complex). The arrows indicate which direction (in both x and y axes) is better. It is evident that there is a strong relationship between predictive ability and network complexity (circle size), in which more complex networks are more accurate. In turn, network complexity incurs computational cost.

Regardless of the mechanism employed to recognize or improve pattern recognition, there is a trade-off between accuracy and complexity, in which more accurate models often incur higher complexity and computational cost, as illustrated in Figure 1. Thereby, discovering accurate and efficient strategies for pattern recognition, which include enhancing the existing ones, have been the focus of intense research.

Motivation. Modern visual pattern recognition models are predominantly based on convolutional networks since they are capable of learning effective representations from data [7]. According to previous works [2], [8], [9], large (deeper and wider) convolutional networks lead to better results. Figure 1 supports this claim, where larger networks (large circles) have superior predictive ability. In terms of performance, however, such networks suffer from heavy computation and memory overhead, incurring slow inference and hindering

[§]Ph.D. thesis

applicability on low-power and resource-constrained devices. Moreover, since modern networks demand massive computing infrastructure, the financial cost to deploy them can be prohibitive for academic researchers. For example, the estimated cost for training and deploying state-of-the-art networks can surpass hundreds of dollars per hour [10]. Prior research on the climate impact of AI has raised another important issue regarding these networks, which is the quantity of carbon emitted by them based on their energy consumption [10]–[12]. Surprisingly, convolutional networks have a large carbon footprint that can surpass one car in its lifetime [10], [11]. The simplest way to circumvent these problems is to evaluate different trade-offs between accuracy and network complexity, for example, by comparing the performance of ResNet50 (50 layers) with its deeper counterpart ResNet152 (152 layers), see Figure 1. Unfortunately, this process requires significant human engineering due to its trial-and-error essence. Instead, it is possible to transform or automatically design efficient convolutional networks by employing pruning or neural architecture search (NAS), respectively. The former removes unimportant structures (neurons or layers) from the network, reducing its complexity while preserving as much predictive ability as possible. The latter learns to design accurate and efficient architectures automatically. Both strategies, however, are not without their limitations. Existing criteria for identifying and removing structures from convolutional networks are ineffective since the accuracy of the original (unpruned) network is often degraded, as shown in Figure 2 (Left). Besides, many pruning approaches demand a high computational cost, mainly when applied to very deep networks [13]. Regarding the neural architecture search, current strategies analyze a large set of possible candidate architectures and, hence, require vast computational resources and take many days to process even with modern Graphics Processing Units (GPUs) [14]. Motivated by these issues, we propose simple, effective, and efficient mechanisms for eliminating structures of deep networks as well as discovering high-performance architectures automatically (i.e., without involving human engineering). More precisely, our pruning strategies achieve the best trade-offs between accuracy and computational cost compared to state-of-the-art methods, as illustrated in Figure 2 (Left). In the context of NAS, our method discovers competitive and low-cost convolutional networks by exploring one order of magnitude fewer models compared to other approaches, thus designing architectures in a few hours on a single GPU, as shown in Figure 2 (Right).

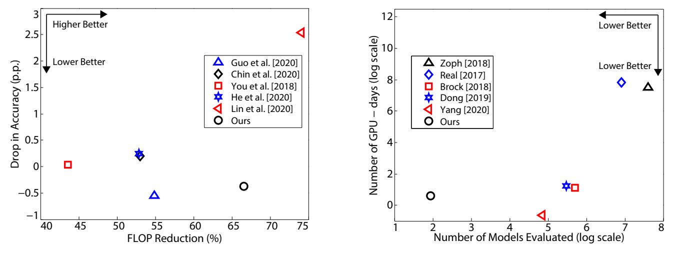

Fig. 2. **Left.** Comparison of existing pruning methods. Compared to state-of-the-art pruning strategies, our pruning method provides a better solution (i.e., it is a non-dominated solution) considering one of the performance metrics: accuracy drop (y-axis) or FLOP reduction (x-axis). In this figure, negative values in the y-axis denote improvement regarding the original, unpruned, network. **Right.** Comparison of existing neural architecture search (NAS) methods. Our NAS method discovers architectures by exploring one order of magnitude fewer models compared to other approaches. In addition, our method is the most resource-efficient as it designs architectures in a few hours on a single GPU. In both figures, the arrows indicate which direction is better.

Besides computational cost concerns, many efforts have been devoted to improve data representation of deep networks. In this context, previous works have demonstrated encouraging results combining features from different levels (layers) of the network. Such works have followed either multi-scale or HyperNet strategies. While the former redesigns network topology to encode features from shallow and deep layers [15], the latter preserves network topology, encouraging application on off-the-shelf networks [16]. Despite the positive results, both strategies increase the computational burden significantly since they insert time-consuming operations at multiple levels of the network. To address this problem, we propose an efficient yet accurate approach to extract different levels of representation across multiple layers of deep networks, thus enhancing data representation at negligible additional cost.

A parallel line of research to improve data representation is to learn compact, but discriminative, representations through dimensionality reduction [4]. In this context, Partial Least Squares (PLS) has presented remarkable results, mainly when compared to other methods such as Principal Component Analysis and Linear Discriminant Analysis [17]–[19]. The promising results of PLS are associated with its characteristics that include being discriminative and robust to *sample size problem* (when the number of samples is smaller than the number of features). Another attractive aspect of PLS is that it can operate as a feature selection method. However, PLS is unsuitable for large datasets (e.g., ImageNet) since all the data need to be available in advance and this could be impractical due to memory constraints. To handle this problem, many works have proposed incremental versions of traditional dimensionality reduction methods [20]–[22], where the idea is to learn compact representations using a single sample at a time. Unfortunately, most incremental PLS fail to keep all its properties and present a high time complexity. To preserve the fundamental characteristics of PLS, we propose a discriminative and low-complexity incremental PLS. Among the advantages of this approach are the preservation of discriminative information, its computational efficiency, and the ability to operate as a feature selection technique.

Objectives. From a practical perspective, our goal is to promote mechanisms capable of reducing the financial cost, carbon emission and computational cost of convolutional networks (see Figure 3). More specifically, we pretend to provide strategies for (i) accelerating convolutional networks, (ii) discovering high-performance convolutional architectures automatically and (iii) efficiently improving data represen-

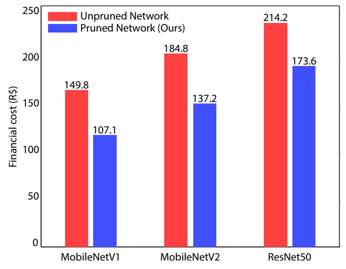

Fig. 3. Financial cost (Brazilian real) and carbon emission for training different convolutional networks. Values above the bars indicate CO_2 in kgCO_2eq (lower is better), which indicates the global warming potential of various greenhouse gases as a single number. Our strategies (blue bars) provide significantly more efficient convolutional networks.

tation of convolutional networks. Additionally, we target to provide a memory-friendly version of PLS. From a theoretical perspective, our goal is to demonstrate the potential of PLS as a tool for determining the importance of structures composing a convolutional network. Besides, we intend to show that it is possible to preserve underlying properties of PLS in its incremental version through simple algebraic decomposition.

Contributions. The contributions of this thesis are simple, effective and efficient strategies for improving computational cost and predictive ability of convolutional networks. Specifically, we reduce more than half of computation, memory usage and inference time, which enables modern convolutional networks suitable to low-power systems. Furthermore, we decrease the financial cost of deploying convolutional networks, which is significant progress in making them more accessible to academic researchers, as shown in Figure 3. Regarding the climate impact of AI, our work enables that modern networks emit around 91% less CO_2. This result is an important step towards green AI. Last but not the least, we expand the applicability of a powerful dimensionality reduction technique, PLS, to large datasets and streaming applications. Particularly, all our contributions are beneficial for academics, researchers, and students with limited computational budgets. To promote reproducibility, we release the source code at: https://arturjordao.github.io/PLSDeepSpaceOdyssey/.

II. Preliminaries

Partial Least Squares. Partial Least Squares is a dimensionality reduction method that yields a set of discriminative latent variables taking into account the relationship between independent (X) and dependent (Y) variables [23], [24].

The idea behind Partial Least Squares (PLS) is to find a projection matrix $W(w_1, w_2, ..., w_c)$ that projects the high dimensional space onto a low c-dimensional space (latent space), where $c \ll m$. In essence, W can be interpreted as a weight matrix that assigns importance for each feature of X. To find W, PLS aims at maximizing the covariance (Cov for short) between the independent and dependent variables. Formally, PLS constructs W such that

$$w_i := maximize(\mathrm{Cov}(Xw, Y)), \ \text{s.t}\|w\| = 1, \qquad (1)$$

where w_i denotes the ith component of the c-dimensional space. The exact solution to Equation 1 is given by

$$w_i := \frac{X^TY}{\|X^TY\|}, \qquad (2)$$

with X and Y normalized (transformed into Z-scores).

From Equation 2, it is possible to compute all c components (c is a parameter) using Nonlinear Iterative Partial Least Squares (NIPALS) [24]. It is worth mentioning that to estimate the other components (w_i, $i > 1$), NIPALS removes the contribution of the previous component on X and Y, named deflation step (see Equation 6 for details). Such a step ensures orthogonality between all the components $w_1, w_2, ...w_c$. We refer interested readers to Algorithms 1 and 2 in the thesis for more details.

Variable Importance in Projection. Besides being more flexible and, often, attaining superior performance than traditional dimensionality reduction techniques [18], [19], another interesting aspect of PLS is that it can operate as a feature selection method. For this purpose, after computing the projection matrix W, we need to employ Variable Importance in Projection (VIP) that estimates the importance of each feature (from the high-dimensional space) w.r.t its contribution to yield the low dimensional space. Due to space constraints, we refer the reader to Section 2.2.1 in the thesis for additional details.

III. Pruning Approaches

Problem Definition. Let $\mathcal{F}$ be a convolutional network with L layers, where the number of neurons in each layer $f_i \in \{1, 2, ..., L\}$ is defined by $|f_i|$. Define $\mathcal{F}'$ a network without some structures of $\mathcal{F}$ such that $|f'_i|_{i=1}^{L'} \leq |f_i|_{i=1}^{L}$ (pruning filters) or $L' < L$ (pruning layers). Thus, $\mathcal{F}'$ is an efficient and lower-complexity version of $\mathcal{F}$. Our target is to identify and remove structures from $\mathcal{F}$ that preserve as much accuracy as possible, which means yielding $\mathcal{F}'$ such that its accuracy is close (ideally superior) to $\mathcal{F}$.

Proposed Method. The first step in our pruning method is to represent the structures that compose the network as feature vectors. For this purpose, we present the training data to the network and interpret the feature maps of each structure as a feature vector (or a set of features). The intuition for using the feature map as a feature is that we are able to measure its relationship with the class label on the latent space (PLS criterion). In this way, a filter or layer associated with a feature with low relationship might be removed.

After executing the previous steps, we have created a high-dimensional feature space, representing all structures of the network at once. Then, we measure the structure importance score to remove the ones with low importance. To this end, we project the high dimensional space onto a latent space using PLS and employ the VIP technique to estimate the contribution of each feature in generating the latent space. Recall that, following the modeling performed in the first step of our method, each feature corresponds to a filter or layer.

Given the importance of all structures, we can remove $p\%$ of these structures associated with low scores. Finally, we

TABLE I

Drop in accuracy using different criteria for determining structure importance. Negative values denote improvement regarding the original (unpruned) network. The best results are in bold. The arrows indicate which direction is better.

	Filter		Layer	
Structure Importance Criterion	CIFAR-10 Acc. Drop$\downarrow$	ImageNet 224×224 Acc. Drop$\downarrow$	CIFAR-10 Acc. Drop $\downarrow$	ImageNet 224×224 Acc. Drop $\downarrow$
infFS [26]	-0.69	-0.50	-0.68	-2.03
ilFS [27]	0.65	-0.36	-0.46	$\mathbf{-2.11}$
infFS$_U$ [28]	0.48	-0.33	-0.50	-2.03
KL [13]	-0.59	-0.41	-0.32	-2.06
HRank [29]	-0.84	-0.47	-0.73	-2.03
ABS [30]	-0.62	-0.42	-0.54	$\mathbf{-2.11}$
PLS+VIP	$\mathbf{-0.89}$	$\mathbf{-0.58}$	$\mathbf{-0.84}$	-1.92

perform some stages of fine-tuning in $\mathcal{F}'$ to compensate for the structures that have been removed.

Results. We compare the proposed criterion (PLS+VIP) for assigning structure importance with other criteria and state-of-the-art feature selection techniques. Table I shows the results obtained by different pruning criteria on the CIFAR-10 and ImageNet datasets. Compared to state-of-the-art pruning criteria [13], [29], [30] and feature selection techniques [26]–[28], PLS+VIP obtained the lowest drop in accuracy.

Even though PLS+VIP underperforms some criteria for assigning layer importance, it is worth mentioning that PLS+VIP is computationally more attractive. For example, the feature selection techniques (infFS, ilFS, infsFSU) require an adjacency matrix representing all pairs of features, consuming substantial computational resources. The approach by Lin et al. [29] (HRank) is time-consuming since the feature map rank is estimated using SVD. Finally, KL-divergence (KL) [13] is one of the most computationally expensive criteria since it requires a forward prediction for each structure of the network.

IV. Neural Architecture Search

Problem Definition. Let $\mathcal{F}$ be a convolutional network composed of S stages. Each stage $s_i \in S$ consists of b_i modules, which in turn define the depth of stage s_i. Following the structure of modern architectures, the layers within a stage operate on the same input/output resolution (i.e., their feature maps have the same dimension). In previous works, including NAS, b is the same for all stages or defined empirically. For example, ResNet39 has six residual blocks in each of its stages (i.e., $b_{i \in \{1,\dots,S\}} = 6$). Our target is to design architectures by learning the number of modules b_i for each stage s_i.

Proposed Method. The first step in our NAS is to define a module type. We consider two types of modules: residual blocks from ResNet [7] or cells from NASNet [14]. The next step in our method is to measure the importance score for each stage $s_i \in S$. For this purpose, given a stage s_i of a network, we present the training samples to the network and extract the feature maps from the last layer of this stage.

Let X_i be the features of s_i estimated following the procedure above. The next step is to compute the importance of these features. By estimating the importance of X_i we are estimating the importance of the stage s_i. Such importance is estimated by presenting X_i to PLS followed by VIP.

Once we are able to estimate the score α_i for each stage s_i, the next step is to build a candidate architecture by adjusting the depth of each stage based on its importance. To this end, we first create a network $\mathcal{F}$ with S stages, each one containing the same number of modules, e.g., by employing $S = 3$ and $|b_i|_{i=1}^{S} = 6$. Then, we create a temporary architecture T by increasing the depth of s_i to $b_i + \delta$, where δ is the growth step, i.e., the number of modules that can be inserted in a stage in a single iteration. Afterwards, we compute the importance scores $\alpha_{\mathcal{F},i}$ and $\alpha_{T,i}$, for each stage s_i of the initial and temporary architectures, respectively. Finally, we update b_i to $b_i + \delta$ if $\alpha_{T,i} > \alpha_{\mathcal{F},i}$ and create a candidate architecture $\hat{\mathcal{F}}$ using the updated b_i. The idea behind this incremental process is to measure if increasing depth will improve the representation learned by the candidate architecture.

The process above composes one iteration of our method, where at the end of each iteration one architecture is discovered. The input for the next iteration is the architecture designed with the values of b_i updated. Thus, given k iterations, our method creates only $2k+1$ architectures, which is an order of magnitude fewer than state-of-the-art NAS approaches.

Results. We compare our method with state-of-the-art NAS approaches, Table II. According to this table, our method is the more cost-effective NAS approach in terms of the number of evaluated models and amount of GPUs required. In particular, our method designs competitive architectures by evaluating a significantly smaller number of models, enabling our approach to run in a few hours on a single GPU. Compared to approaches that also evaluate a small number of models [31]–[33], our method achieves the best trade-off between accuracy and number of GPUs. In summary, our method built more parameter-efficient architectures even without considering the computational cost in the searching process.

Following a recent trend [10]–[12], we also measure the carbon emission for training architectures. Our candidate architectures emit notably less carbon, even taking into account shallow versions of ResNet. Compared to ResNet110, our final architecture trained from scratch emits 41% less CO$_2$. This occurs because our architectures are computationally more

TABLE II
COMPARISON WITH STATE-OF-THE-ART NAS APPROACHES. RESULTS TAKEN FROM PREVIOUS WORKS. THE BEST VALUES ARE SHOWN IN BOLD. '–' INDICATES THE METRIC IS NOT REPORTED BY THE ORIGINAL PAPER. THE ARROWS INDICATE WHICH DIRECTION IS BETTER.

Model	Evaluated↓ Models	GPUs↓	Param.↓ (Million)	Accuracy↑ CIFAR-10
Zoph et al. [14]	20,000	800	2.5	94.51
Baker et al. [34]	1,500	10	11.1	93.08
Real et al. [35]	1,000	250	5.4	94.60
Brock et al. [36]	300	**1**	4.6	94.47
Dong and Yang [37]	240	**1**	2.5	96.25
Yang et al. [38]	128	1	3.6	97.38
Jin et al. [33]	≈60	**1**	–	88.56
Elsken et al. [31]	40	5	19.7	94.80
Kandasamy et al. [32]	**10**	4	–	91.31
Chen et al. [39]	–	**1**	10.5	**97.75**
Li et al. [40]	–	1	3.90	96.21
Ours (Res. modules)	11	**1**	**1.7**	94.27
Ours (Cell modules)	11	**1**	2.3	94.74
Ours (Ensemble)	–	–	7.27	95.68

efficient, leading to a considerably faster training stage.

V. HyperNet Approach

Problem Definition. Let X_i be an output (feature map) of a specific layer $f_i \in \{1, 2..., L\}$ from a network $\mathcal{F}$ of L layers. Define $\mathbb{O}$ a set of feature maps X_i such that $|\mathbb{O}| > 1$. We assume that $\mathbb{O}$ provides better data representation than using a single X_i. Our target is to efficiently yield $\mathbb{O}$, which means combining multiple X_i in an efficient yet accurate way.

Proposed Method. The first step is our Latent HyperNet (LHN) is to define a set of layers, $l \subset L$, to be combined. This is a typical step in HyperNet approaches and it is necessary because some early layers contain simple patterns (i.e., edges), which do not contribute to the classification but increase computational cost [41], [42]. Therefore, setting the layers to be combined is more appropriate than using all of them.

Once we have set the layers l, we use the feature maps X_i of each layer $f_i \in l$ to learn a PLS model. Such feature maps are high dimensional, which reinforces the employment of PLS as it is proper for these scenarios. Following this modeling, each $f_i \in l$ will have a PLS model associated with it.

After executing the above steps, we project the feature maps X_i on its respective PLS model yielding compact representations of X_i, which in turn are concatenated in $\mathbb{O}$. In summary, before inserting X_i into $\mathbb{O}$ we reduce its dimensionality using PLS. According to this definition, our LHN neither modifies the design nor the learned weights of the network, enabling it to be easily adaptable to any network.

Results. Table III shows the improvements in accuracy achieved by the HyperNets approaches using multiple layers from VGG16 and ResNet20. According to this table, on CIFAR-10, the approach by Kong et al. [41] was not able to improve the accuracy compared to the original network. In contrast, our LHN obtained a marginal improvement. On the ImageNet dataset, the approach by Kong et al. [41] improved the accuracy of VGG16 and ResNet20 in 0.01 p.p. and 3.60 p.p., respectively. On the other hand, LHN improved the accuracy of VGG16 and ResNet20 in 0.66 p.p. and 2.65 p.p..

TABLE III
IMPROVEMENTS IN ACCURACY ACHIEVED BY THE HYPERNETS. NEGATIVE VALUES DENOTE A **DECREASE** IN ACCURACY REGARDING THE ORIGINAL NETWORK. THE BETTER METHOD IS SHOWN IN BOLD. THE ARROWS INDICATE WHICH DIRECTION IS BETTER.

		CIFAR-10↑	ImageNet ↑
VGG16	HyperNet [41]	−0.22	0.01
	LHN (Ours)	**0.05**	**0.66**
ResNet20	HyperNet [41]	**-0.02**	**3.60**
	LHN (Ours)	−0.13	2.65

VI. Incremental Partial Least Squares

Problem Definition. Let $W(w_1, w_2, ..., w_c)$ be a projection matrix that projects the high dimensional space onto a low c-dimensional space. Considering that W was obtained by PLS, which means that each component w_i maximizes the covariance between Xw_i and Y, where X and Y represent all the data samples and their respective labels. Our target is to find W using a single sample $x \in X$, and its respective label y, at a time while maintaining the property of maximizing the covariance across all c-components.

Proposed Method. Our incremental Partial Least Squares approach focuses on ensuring that, as in traditional PLS, the relationship between independent and dependent variables (Equation 2) be kept on all the components. To this end, our method works as follows. First, we center the data to the mean of the training samples X. In incremental approaches the mean is unknown since we cannot assume that all the data are known a priori [20], [43]. To face this problem, we centralize the current sample using an approximate centralization process [43], which consists of estimating an incremental mean using the nth sample. According to Weng et al. [43], we can compute the incremental mean μ_n w.r.t. the nth data sample as

$$\mu_n := \frac{n-1}{n}\mu_{(n-1)} + \frac{1}{n}x_n. \tag{3}$$

Once we have centralized the sample, the next step in our method is to compute the component w_i following Equation 2. As we mentioned, X and its respective Y are unknown or are not in memory in advance, which prevents us from employing Equation 2 directly. However, as suggested by Zeng and Li [20], we employ the following decomposition:

$$X^TY = \sum_{k=1}^{n-1} x_k^T y_k + x_n^T y_n. \tag{4}$$

By replacing X^TY in Equation 2 by Equation 4, it is possible to calculate the ith component of PLS considering a single sample at a time. In other words, Equation 4 enables to compute w_i incrementally.

To compute the higher-order components (w_i, $i > 1$), we employ a *deflation* process, which consists of subtracting the contribution of the current component on the sample before estimating the next component [44], [45]. Following the NIPALS algorithm, the deflation process works as follows

$$t := Xw_i,\ p := X^T t,\ q := Y^T t, \tag{5}$$

$$X := X - tp^T,\ Y := Y - tq^T, \tag{6}$$

where t denotes the projected samples onto the current component w_i, and p and q represent the scores of this projection. It should be noted that while t works in an incremental scheme, p and q cannot be computed since X and Y are neither known nor are in memory in advance. Fortunately, in light of Equation 4, we can decompose p and q as

$$p = \sum_{k=1}^{n-1} x_k^T t_k + x_n^T t_n,\ q = \sum_{k=1}^{n-1} y_k^T t_k + y_n^T t_n. \tag{7}$$

By embedding Equation 7 on the deflation process, we can remove the contribution of the current component and repeat the process to compute a single component w_i. Observe that Equation 6 can be computed sample-by-sample working, therefore, in an incremental scheme. At this stage, we obtain all the requirements to find c components incrementally.

Results. We compare the proposed CIPLS with other incremental dimensionality reduction methods. Table IV summarizes the results. On the LFW dataset, CIPLS outperformed SGDPLS and IPLS in 1.18 and 1.48 p.p., respectively. Similarly, on the YTF dataset, CIPLS outperformed SGDPLS and IPLS in 0.88 and 1.88 p.p., in this order. In particular, on these datasets, the results of CIPLS were statistically superior to IPLS and SGDPLS. On ImageNet, the difference in accuracy compared to IPLS was of 0.07 and 1.35 p.p., for the 32×32 and 224×224 versions, respectively.

To demonstrate the efficiency of CIPLS, we compare its time complexity to compute the projection matrix and prediction time with the incremental methods evaluated. Table V shows the results. Overall, our method presents a low time complexity for estimating the projection matrix. In addition, the time for estimating the projection matrix of our method was equivalent to CCIPCA, which is the fastest incremental dimensionality reduction. Therefore, CIPLS is the fastest among the compared incremental PLS methods.

TABLE IV
COMPARISON OF EXISTING INCREMENTAL METHODS IN TERMS OF ACCURACY. THE SYMBOL '–' DENOTES THAT IT WAS NOT POSSIBLE TO EXECUTE THE METHOD DUE TO MEMORY CONSTRAINTS OR CONVERGENCE PROBLEMS. PLS DENOTES THE USE OF THE TRADITIONAL PLS. THE CLOSER TO THE ACCURACY OF PLS, THE BETTER.

	LFW	YTF	ImageNet 32×32	ImageNet 224×224
CCIPCA	89.87	81.48	40.30	52.58
SGDPLS	90.60	83.22	–	–
IPLS	90.30	82.22	43.24	65.74
CIPLS (Ours)	91.78	84.10	43.31	67.09
PLS	92.47	85.96	–	–

TABLE V
COMPARISON OF THE METHODS IN TERMS OF TIME COMPLEXITY FOR ESTIMATING THE PROJECTION MATRIX AND AVERAGE (AVG.) PREDICTION TIME (THE LOWER THE BETTER). m, n DENOTE DIMENSIONALITY OF THE ORIGINAL DATA AND NUMBER OF SAMPLES, WHILE c, L AND T DENOTE NUMBER OF PLS COMPONENTS, NUMBER OF PCA COMPONENTS AND CONVERGENCE STEPS, RESPECTIVELY.

	Time Complexity	Avg. Time (seconds)
CCIPCA [43]	$O(nLm)$	0.003
SGDPLS [46]	$O(Tcm)$	0.01
IPLS [20]	$O(nLm + c^2m)$	0.006
CIPLS (Ours)	$O(ncm)$	0.002

VII. PUBLICATIONS AND AWARDS

Journals

1) Jordao, A., Yamada, F., and Schwartz, W. R. Deep Network Compression based on Partial Least Squares. Neurocomputing, 2020.
2) Jordao, A., Lie, M., and Schwartz, W. R. Discriminative Layer Pruning for Convolutional Neural Networks. Journal of Selected Topics in Signal Processing, 2020.

Conferences

1) Jordao, A., Kloss, R. B., and Schwartz, W. R. Latent hypernet: Exploring the layers of Convolutional Neural Networks. International Joint Conference on Neural Networks (IJCNN), 2018.
2) Jordao, A., Kloss, R., Yamada, F., and Schwartz, W. R. Pruning Deep Neural Networks using Partial Least Squares. British Machine Vision Conference (BMVC) Workshops: Embedded AI for Real-Time Machine Vision, 2019.
3) Jordao, A., Yamada, F., Lie, M., and Schwartz, W. R. Stage-Wise Neural Architecture Search. International Conference on Pattern Recognition (ICPR), 2020.
4) Jordao, A., Lie, M., de Melo, V. H. C., and Schwartz, W. R. Covariance-free partial least squares: An Incremental Dimensionality Reduction Method. Winter Conference on Applications of Computer Vision (WACV), 2021.

Awards

1) Finalist of the XXXIV Concurso de Teses e Dissertações (CTD) 2021 – XLI Congresso da Sociedade Brasileira de Computação (CSBC), ranking among the top 6 (out of 46) best theses.
2) Award nomination to CAPES Thesis Award and UFMG Thesis Grand Prize.

Acknowledgments. The authors would like to thank Ricardo Kloss, Maiko Lie, Fernando Yamada, and Victor de Melo for their valuable contributions to this thesis. The authors would like to thank the Brazilian National Research Council – CNPq (Grants 438629/2018-3, 309953/2019-7 and 140082/2017-4), the Minas Gerais Research Foundation FAPEMIG (Grants APQ-00567-14 and PPM-00540-17) and the Coordination for the Improvement of Higher Education Personnel – CAPES (DeepEyes Project).

References

[1] A. P. Badia, B. P. S. K. P. S. A. V. Z. Guoand, and C. Blundell, "Agent57: Outperforming the atari human benchmark," in *International Conference on International Conference on Machine Learning (ICML)*, 2020.

[2] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning (ICML)*, 2019.

[3] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, "Big transfer (bit): General visual representation learning," in *European Conference on Computer Vision (ECCV)*, 2020.

[4] Y. Li, M. Yang, and Z. Zhang, "A survey of multi-view representation learning," *Transactions on Knowledge and Data Engineering*, vol. 31, no. 10, pp. 1863–1883, 2019.

[5] M. A. Diniz and W. R. Schwartz, "Face attributes as cues for deep face recognition understanding," in *International Conference on Automatic Face and Gesture Recognition(FG)*, 2020.

[6] X. Suau, L. Zappella, and N. Apostoloff, "Filter distillation for network compression," in *Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] J. S. Rosenfeld, A. Rosenfeld, Y. Belinkov, and N. Shavit, "A constructive prediction of the generalization error across scales," in *International Conference on Learning Representations (ICLR)*, 2020.

[9] K. Han, Y. Wang, Q. Zhang, W. Zhang, C. XU, and T. Zhang, "Model rubiks cube: Twisting resolution, depth and width for tinynets," in *Neural Information Processing Systems (NeurIPS)*, 2020.

[10] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Conference of the Association for Computational Linguistics*, 2019.

[11] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the carbon emissions of machine learning," in *Neural Information Processing Systems (NeurIPS)*, 2019.

[12] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[13] J.-H. Luo and J. Wu, "Neural network pruning with residual-connections and limited-data," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[14] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[15] L. Yang, Y. Han, X. Chen, S. Song, J. Dai, and G. Huang, "Resolution adaptive networks for efficient inference," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[16] V. Sindagi and V. M. Patel, "Multi-level bottom-top and top-bottom feature fusion for crowd counting," in *International Conference on Computer Vision (ICCV)*, 2019.

[17] A. Sharma and D. W. Jacobs, "Bypassing synthesis: PLS for face recognition with pose, low-resolution and sketch," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[18] R. Hasegawa and K. Hotta, "Plsnet: A simple network using partial least squares regression for image classification," in *International Conference on Pattern Recognition (ICPR)*, 2016.

[19] R. B. Kloss, A. Jordão, and W. R. Schwartz, "Boosted projection: An ensemble of transformation models," in *Iberoamerican Congress on Pattern Recognition (CIARP)*, 2017.

[20] X. Zeng and G. Li, "Incremental partial least squares analysis of big streaming data," *Pattern Recognition*, vol. 47, pp. 3726–3735, 2014.

[21] A. E. Stott, S. Kanna, D. P. Mandic, and W. T. Pike, "An online NIPALS algorithm for partial least squares," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2017.

[22] S. Alakkari and J. Dingliana, "An acceleration scheme for mini-batch, streaming PCA," in *British Machine Vision Conference (BMVC)*, 2019.

[23] P. Geladi and B. Kowalski, "Partial least-squares regression: a tutorial," *Analytica Chimica Acta*, vol. 185, pp. 1–17, 1986.

[24] H. Abdi, "Partial least squares regression and projection on latent structure regression (pls regression)," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 1, pp. 97–106, 2010.

[25] T. Mehmood, K. H. Liland, L. Snipen, and S. Saebo, "A review of variable selection methods in partial least squares regression," *Chemometrics and Intelligent Laboratory Systems*, 2012.

[26] G. Roffo, S. Melzi, and M. Cristani, "Infinite feature selection," in *International Conference on Computer Vision (ICCV)*, 2015.

[27] G. Roffo, S. Melzi, U. Castellani, and A. Vinciarelli, "Infinite latent feature selection: A probabilistic latent graph-based ranking approach," in *International Conference on Computer Vision (ICCV)*, 2017.

[28] G. Roffo, S. Melzi, U. Castellani, A. Vinciarelli, and M. Cristani, "Infinite feature selection: a graph-based feature filtering approach," *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2020.

[29] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao, "Hrank: Filter pruning using high-rank feature map," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[30] C. M. J. Tan and M. Motani, "Dropnet: Reducing neural network complexity via iterative pruning," in *International Conference on International Conference on Machine Learning (ICML)*, 2020.

[31] T. Elsken, J. H. Metzen, and F. Hutter, "Simple and efficient architecture search for convolutional neural networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[32] K. Kandasamy, W. Neiswanger, J. Schneider, B. Póczos, and E. P. Xing, "Neural architecture search with bayesian optimisation and optimal transport," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[33] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *International Conference on Knowledge Discovery & Data Mining (SIGKDD)*, 2019.

[34] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2017.

[35] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on International Conference on Machine Learning (ICML)*, 2017.

[36] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "SMASH: one-shot model architecture search through hypernetworks," in *International Conference for Learning Representations(ICLR)*, 2018.

[37] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[38] Z. Yang, Y. Wang, X. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, and C. Xu, "CARS: continuous evolution for efficient neural architecture search," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[39] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *International Conference on Computer Vision (ICCV)*, 2019.

[40] Z. Li, T. Xi, J. Deng, G. Zhang, S. Wen, and R. He, "GP-NAS: gaussian process based neural architecture search," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[41] T. Kong, A. Yao, Y. Chen, and F. Sun, "Hypernet: Towards accurate region proposal generation and joint object detection," in *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[42] B. Hariharan, P. A. Arbeláez, R. B. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[43] J. Weng, Y. Zhang, and W. Hwang, "Candid covariance-free incremental principal component analysis," *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 25, no. 8, pp. 1034–1040, 2003.

[44] A. L. Andrew and R. C. E. Tan, "Computation of derivatives of repeated eigenvalues and the corresponding eigenvectors of symmetric matrix pencils," *SIAM Journal on Matrix Analysis and Applications*, vol. 20, no. 1, pp. 78–100, 1998.

[45] L. W. Mackey, "Deflation methods for sparse PCA," in *Neural Information Processing Systems (NIPS)*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2008.

[46] R. Arora, P. Mianjy, and T. V. Marinov, "Stochastic optimization for multiview representation learning using partial least squares," in *International Conference on International Conference on Machine Learning (ICML)*, 2016.