A Comprehensive Analysis of Deep Regression

Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda, *Member IEEE*, and Radu Horaud

Abstract—Deep learning revolutionized data science, and recently, its popularity has grown exponentially, as did the amount of papers employing deep networks. Vision tasks such as human pose estimation did not escape this methodological change. The large number of deep architectures lead to a plethora of methods that are evaluated under different experimental protocols. Moreover, small changes in the architecture of the network, or in the data pre-processing procedure, together with the stochastic nature of the optimization methods, lead to notably different results, making extremely difficult to sift methods that significantly outperform others. Therefore, when proposing regression algorithms, practitioners proceed by trial-and-error. This situation motivated the current study, in which we perform a systematic evaluation and a statistical analysis of the performance of vanilla deep regression – short for convolutional neural networks with a linear regression top layer –. Up to our knowledge this is the first comprehensive analysis of deep regression techniques. We perform experiments on three vision problems and report confidence intervals for the median performance as well as the statistical significance of the results, if any. Surprisingly, the variability due to different data pre-processing procedures generally eclipses the variability due to modifications in the network architecture.

Index Terms—Deep Learning, Regression, Computer Vision, Convolutional Neural Networks, Statistical Significance, Empirical and Systematic Evaluation, Head-Pose Estimation, Full-Body Pose Estimation, Facial Landmark Detection.

arXiv:1803.08450v1 [cs.CV] 22 Mar 2018

1 Introduction

Regression techniques are widely employed to solve tasks where the goal is to predict continuous values. In computer vision, regression techniques span a large ensemble of applicative scenarios such as: head-pose estimation [1], [2], facial landmark detection [3], [4], human pose estimation [5], [6], age estimation [7], [8], or image registration [9], [10].

For the last decade, deep learning architectures have overwhelmingly outperformed the state-of-the-art in many traditional computer vision tasks such as image classification [11], [12] or object detection [13], [14]. Roughly speaking, these architectures consist of several convolutional layers, generally followed by a few fully-connected layers, and a classification softmax layer with, for instance, a cross-entropy loss; the overall architecture is referred to as convolutional neural network (ConvNet). Besides classification, ConvNets are also used to solve regression problems. In this case, the softmax layer is commonly replaced with a fully connected regression layer with linear or sigmoid activations. We refer to such architectures as *vanilla deep regression*. Following this approach, many deep models were proposed, obtaining state-of-the-art results in classical vision regression problems such as head pose estimation [15], human pose estimation [16], [17] or facial landmark detection [18]. Interestingly, *vanilla deep regression* is often used as the main building block in cascaded approaches, where such regression methods are iteratively used to refine the predictions. Indeed, [16], [17], [18] demonstrate the effectiveness of this strategy. Recently, more complex computer vision methods exploiting regression include multi-stage deep regression using clustering, auto-routing and pseudo-labels for deep fashion landmark detection [19], inverse piece-wise affine probabilistic model for head pose estimation [20] or a head-map estimator from which facial landmarks are extracted [21]. However, the architectures developed in these studies are designed for a specific task, and how to adapt them to a different task is a non-trivial research question.

Regression could also be formulated as classification [22], [23]. In that case, the output space is generally discretized in order to obtain class labels, and a multi-class loss is minimized. However, this approach suffers from important drawbacks. First, there is an inherent trade-off between the complexity of the optimization problem and the accuracy of the method due to the discretization process. Second, in absence of a specific formulation, a confusion between two classes has the same cost independently of the corresponding error in the target space. Last, the discretization method that is employed must be designed specifically for each task, and therefore general-purpose methodologies must be used with care. In this study, we focus on generic vanilla deep regression so as to escape from the definition of a task-dependent trade-off between complexity and accuracy, as well as from a penalization loss that may not take the spatial proximity of the labels into account.

Thanks to the success of deep neural architectures, based on empirical validation, much of the scientific work currently available in computer vision exploits their representation power. Unfortunately, the immense majority of these works provide neither a statistical evaluation of the performance nor a rigorous justification of the methodological choices. The main consequence of this lack of systematic evaluation is that researchers proceed by trial-and-error experimentation because the scientific evidence behind the superiority of newly introduced techniques is neither sufficiently clear nor statistically grounded.

There are a few studies devoted to the extensive and/or

- *All authors are with the PERCEPTION team, INRIA Grenoble Rhône-Alpes and Université Grenoble Alpes. E-mail: firstname.lastname@inria.fr*
- *EU funding via the FP7 ERC Advanced Grant VHIA #340113 is greatly acknowledged.*

systematic evaluation of deep architectures, focusing on different aspects. For instance, seminal papers exploring efficient back-propagation strategies [24], or evaluating ConvNets for visual recognition [25], were already published twenty years ago. Some articles provide general guidance and understanding on appropriate architectural choices [26], [27], [28] or on gradient-based training strategies for deep architectures [29], while others try to delve into the differences in performance of the variants of a specific (recurrent) model [30], or the differences in performance of several models applied to a specific problem [31], [32], [33]. Automatic ways to overcome the problem of choosing the optimal network architecture were also devised [34], [35]. Overall, there is very little guidance on the plethora of design choices and hyper-parameter settings for deep learning architectures, let alone ConvNets for regression problems.

In summary, the analysis of prior work shows that, first, the absence of a systematic evaluation of deep learning advances in regression, and second, an over abundance of papers based on deep learning (for instance, $\approx$ 2000 papers were uploaded on arXiv only in March 2017, in the categories related to machine learning, computer vision and pattern recognition, computation and language, and neural and evolutionary computing). This highlights again the importance of serious comparative empirical studies to discern which are the key blocks in deep regression. Finally, in our opinion, there is a scientifically unjustified lack of statistical tests and confidence intervals in the experimental sections of the vast majority of works published in the field. Both the execution of a single run per benchmarked method and the succinct description of the preprocessing strategies being used limit the reliability of the results and method reproducibility.

Even if some authors devote time and efforts to make their research reproducible [21], many published studies do not describe implementation, practical, or data preprocessing in detail [15], [17], [18], [19]. One possible reason to explain this state of affairs may be the unavailability of systematic comparative procedures that highlight the most important details.

We propose to fill in this gap with a systematic evaluation and a statistical analysis of the performance of vanilla deep regression for computer vision tasks. In order to conduct this study we take inspiration from two recent papers. [32] presents an extensive evaluation of ConvNets on ImageNet for image categorization, including the type of non-linearity, pooling variants, network width, classifier design, image pre-processing, and learning parameters. No statistical analysis accompanies the results reported in [32] such that one understands the differences in performance from a thorough statistical perspective. [30] discusses the differences in performance of several variants of the long short-term memory based recurrent networks for categorical sequence modeling, thus addressing classification of sequential data: each variant is run several times, statistical tests are employed to compare each variant with a baseline architecture, and the performance is discussed over boxplots offering some sort of graphical confidence intervals of the different benchmarked methods.

In this paper we carry out an in-depth analysis of vanilla deep regression methods on three standard computer vision problems: head pose estimation, facial landmark detection and full-body pose estimation. We first focus on the impact of different optimization techniques so as to guarantee that the networks are correctly optimized in the rest of the experiments. With the help of statistical tests we perform a systematic comparison between different network variants (e.g. the fine-tuning depth) and data pre-processing strategies. More precisely, we run each experiment five times and compute 95% confidence intervals for the median performance as well as the Wilcoxon signed-rank test [36]. We claim that the combination of these two indicators is much more robust and reliable than the average performance over a single run as usually done in practice. Therefore, the main contribution is, not only a benchmarking methodology, but also the conclusions that stem out. We compare the vanilla deep regression methods to some state-of-the-art regression algorithms specifically developed for the tasks just mentioned. Finally, we discuss which are the factors with high performance variability and which therefore should be a priority when exploring the use of deep regression in computer vision.

The rest of the manuscript is organized as follows. Section 2 discusses the experimental protocols (data sets and base architectures) used to benchmark the different choices. Section 3 is devoted to find the best optimization strategy for each of the base architectures and data sets. The statistical tests (paired-tests and confidence intervals) are described in detail in Section 4, and used to soundly benchmark the different network variants and data preprocessing strategies, respectively in Sections 5 and 6. We explore how deep regression is positioned with respect to the state-of-the-art in Section 7, before presenting an overall discussion in Section 8. Conclusions are drawn in Section 9.

2 Experimental Protocol

In this section, we describe the protocol adopted to evaluate the influence of different architectures and their variants, training strategies, and data pre-processing methods. We run the experiments using two common base architectures (see Section 2.1), on three standard computer vision problems that are often solved using regression (see Section 2.2). The computational environment is described in 2.3.

2.1 Base Architectures

We choose to perform our study using two architectures that are among the most commonly referred in the recent literature: VGG-16 [37] and ResNet-50 [38]. The VGG-16 model was termed *Model D* in [37]. We prefer VGG-16 to AlexNet [11] because it performs significantly better on ImageNet and has inspired several network architectures for various tasks [19], [39]. ResNet-50 performs even better than VGG-16 with shorter training time, which is nice in general, and in particular for benchmarking.

VGG-16 is composed of 5 blocks containing two or three convolution layers and a max pooling layer. Let CB^i denote the i^{th} convolution block (see [37] for the details on the number of layers and of units per layer, as well as the convolution parameters). Let FC^i denote the i^{th} fully connected layer with a dropout rate of $50\,\%$. Fl denotes a

flatten layer that transforms a 2D feature map into a vector and SM denotes a soft-max layer. With these notations, VGG-16 can be written as $CB^1 - CB^2 - CB^3 - CB^4 - CB^5 - Fl - FC^1 - FC^2 - SM$.

The main novelty of ResNet-50 is the use of identity shortcuts, which augment the network depth and reduce the number of parameters. We remark that all the convolutional blocks of ResNet-50, except for the first one, have identity connections, making them *residual* convolutional blocks. However, since we do not modify the original ResNet-50 structure, we denote them CB as well. In addition, GAP denotes a global average pooling layer. According to this notation, ResNet-50 architecture can be described as follows: $CB^1 - CB^2 - CB^3 - CB^4 - CB^5 - GAP - SM$.

Both networks are initialized by training on ImageNet for classification, as it is usually done. We then remove the last soft-max layer SM, employed in the context of classification, and we replace it with a fully connected layer with linear activations equal in number to the dimension of the target space. Therefore this last layer is a regression layer, denoted REG, whose output dimension corresponds to the one of the target space of the problem at hand.

2.2 Data Sets

In order to perform empirical comparisons, we choose three challenging problems: head-pose estimation, facial landmark detection and human-body pose estimation. For each of these problems we selected data sets that are widely used, and that favor diversity in output dimension, pre-processing and data-augmentation requirements.

The **Biwi** head-pose data set [40] consists of over $15,000$ RGB-D images corresponding to video recordings of 20 people (16 men and 4 women, some of whom recorded twice) using a Kinect camera. It is one of the most widely used data set for head-pose estimation [15], [20], [41], [42], [43]. During the recordings, the participants freely move their head and the corresponding head orientations lie in the intervals $[-60°, 60°]$ (pitch), $[-75°, 75°]$ (yaw), and $[-20°, 20°]$ (roll). Unfortunately, it would be too time consuming to perform cross-validation in the context of our statistical study. Consequently, we employed the split used in [20]. Importantly, none of the participants appears both in the training and test sets.

We also use the **LFW** and **NET** facial landmark detection (**FLD**) data sets [18] that consist of 5590 and 7876 face images, respectively. We combined both data sets and employed the same data partition as in [18]. Each face/image is labeled with the pixel coordinates of five key-points, namely left and right eyes, nose, and left and right mouth corners.

Finally, we use the **Parse** data set [44] that is a standard data set used for human pose estimation. It is a relatively small data set as it contains only 305 images. Therefore, this data set challenges very deep architectures and requires a data augmentation procedure. Each image is annotated with the pixel coordinates of 14 joints. In addition, given the limited size of **Parse**, we consider all possible image rotations in the interval $[12°, -12°]$ with a step of $0.5°$. This procedure is applied only to the training images.

2.3 Computational Environment

All our experiments were ran on an Nvidia TITAN X (Pascal generation) GPU with Keras 1.1.1 (on the Theano 0.9.0 backend), thus favoring an easy to replicate hardware configuration. This computational environment is used to compare the many possible choices studied in this paper. In total, we summarize the results of more than 600 experimental runs (≈64 days of GPU time). The code and the results of individual runs are available online.[1]

3 Network Optimization

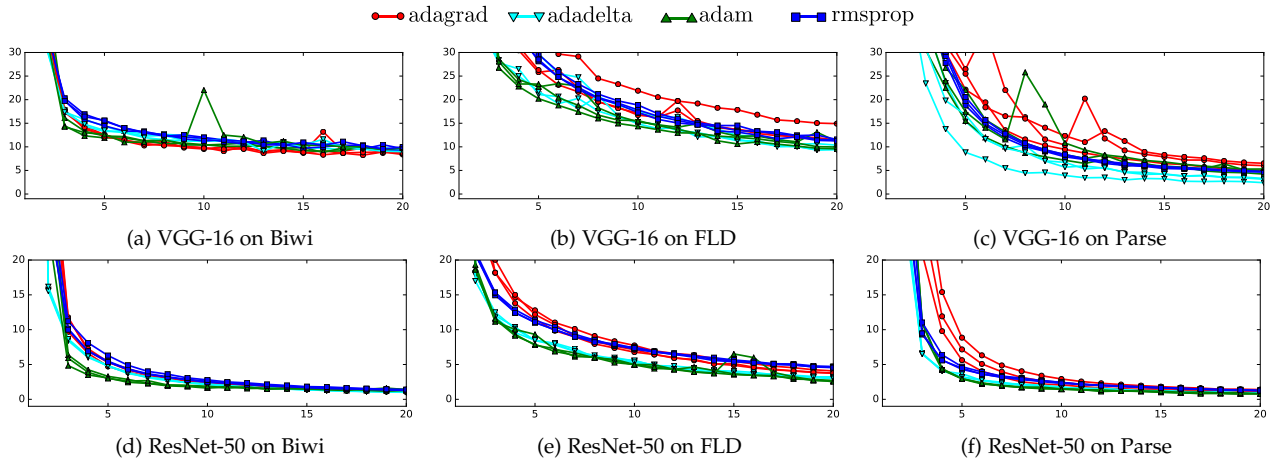
Optimizers for training neural networks are responsible for finding the free parameters θ (usually denoted weights) of a cost function $J(\theta)$ that, typically, includes a performance measure evaluated on the training set and additional regularization terms. Such a cost (also called loss) function lets us quantify the quality of any particular set of weights. In this paper, network optimization and network fine-tuning are used as synonym expressions, since we always start from ImageNet pre-trained weights when fine-tuning on a particular new task. The positive impact of pre-training in deep learning has been extensively studied and demonstrated in the literature [45], [46], [47], [48].

Gradient descent, a first-order iterative optimization algorithm for finding the minimum of a function, is the most common and established method for optimizing neural network loss functions [49]. There are other methods to train neural networks, from derivative-free optimization [50], [51] to second-order methods [49], but their use is much less widespread, so they have been excluded from this paper. Attending to the number of training examples used to evaluate the gradient of the loss function, we can distinguish between batch gradient descent (that employs the entire training set at each iteration), mini-batch gradient descent (that uses several examples in each iteration), and stochastic gradient descent (also called sometimes on-line gradient descent, that employs a single example at each iteration). In this paper, and as is common practice in the deep learning literature, we use a mini-batch gradient descent whose batch size is selected through the preliminary experimentation described in Section 3.2.

Many improvements of the basic gradient descent algorithm have been proposed. In particular, the need to set a learning rate (step size) has been recognized as crucial and problematic: setting this parameter too high can cause the algorithm to diverge; setting it too low makes it slow to converge. In practice, it is generally beneficial to gradually decrease the learning rate over time [49]. Recently, a number of algorithms with adaptive learning rates have been developed, and represent some of the most popular optimization algorithms actively in use.

AdaGrad [52] (Adaptive Gradient) adapts the learning rate of every weight dividing it by the square root of the sum of their historical squared values. Weights with high gradients will have a rapid decrease of their learning rate, while weights with small or infrequent updates will have a relatively small decrease of their learning rate.

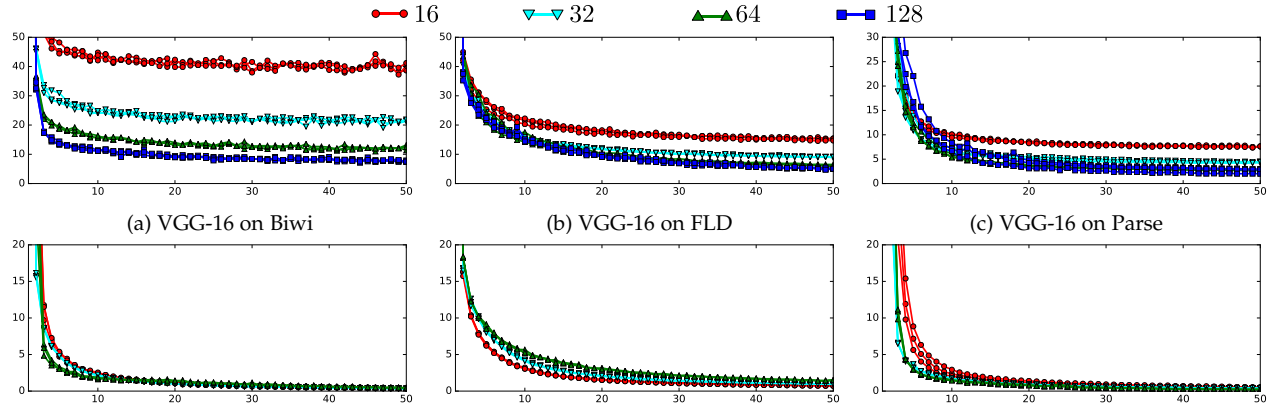
1. https://team.inria.fr/perception/deep-regression/



(a) VGG-16 on Biwi (b) VGG-16 on FLD (c) VGG-16 on Parse

(d) ResNet-50 on Biwi (e) ResNet-50 on FLD (f) ResNet-50 on Parse

Fig. 1: Comparison of the training loss evolution with different optimizers for VGG-16 and ResNet-50.



(a) VGG-16 on Biwi (b) VGG-16 on FLD (c) VGG-16 on Parse

(d) ResNet-50 on Biwi (e) ResNet-50 on FLD (f) ResNet-50 on Parse

Fig. 2: Comparison of the training loss evolution with different batch size on VGG-16 and ResNet-50.

RMSProp [53] (Root Mean Square Propagation) modifies AdaGrad, to avoid lowering the learning rates very aggressively, by changing the gradient accumulation into an exponentially weighted moving average. AdaGrad shrinks the learning rate according to the entire history of the squared gradient, while RMSProp only considers recent gradients for that weight.

AdaDelta [54] is also an extension of Adagrad, similar to RMSProp, that again dynamically adapts over time using only first order information and requires no manual tuning of a learning rate.

Adam [55] (Adaptive Moments) is an update to the RMSProp optimizer where momentum [56] is incorporated, i.e. in addition to store an exponentially decaying average of previous squared gradients (like in RMSProp), Adam also employs an exponentially decaying average of previous gradients (similar to momentum, where such moving average of previous gradients helps to dampen oscillations and to accelerate learning with intertia).

These four adaptive optimizers are evaluated in Section 3.1 and the two exhibiting the highest performance are used in the rest of the paper. We employ the mean square error (MSE) as the loss function to be optimized. As discussed in detail in Section 5, VGG-16 and ResNet-50 are fine-tuned from (and including) the fifth and third convolutional block, respectively.

3.1 Impact of the Network Optimizer

As outlined above, we compare four optimizers: AdaGrad, AdaDelta, Adam and RMSProp. For each optimizer, we train the two networks (VGG-16 and ResNet-50) three times during 50 epochs with the default parameter values given in [57]. In this series of experiments, we choose a batch size of 128 and 64 for VGG-16 and ResNet-50, respectively (see Section 3.2). The evolution of the loss value on the training set is displayed in Figure 1 for each one of the three data sets. Even if the differences between the four optimizers are not extremely noticeable, we can state that the best training performances (in terms of loss value and convergence time) correspond to AdaDelta and Adam.

In the light of the results described above, AdaGrad and RMSProp do not seem to be the optimizers to be used when training vanilla deep regression. While Adam and AdaDelta perform well, a comparative study prior to the selection of a particular optimizer is strongly encouraged since the choice may depend on the architecture and the problem at hand.

3.2 Impact of the Batch Size

In this case, we test the previously selected optimizers (AdaDelta for VGG-16 and Adam for ResNet-50) with batch sizes of 16, 32, and 64. A batch size of 128 is tested on VGG-16 but not in ResNet-50 due to GPU memory limitations. We assess how the batch size impacts the optimization

performance: Figure 2 shows the loss values obtained when training with different batch sizes.

First, we remark that the impact of the batch size in VGG-16 is more important than in ResNet-50. The latter is more robust to the batch size, yielding comparable results no matter the batch size employed. Second, in general, we could conclude that using a larger batch size is a good heuristic towards good optimization (because in VGG-16 shows to be decisive, and in ResNet-50 does not harm performance). However, the maximal batch size that can be used is constrained by the GPU memory being used. In our case, with an Nvidia TITAN X with 12 GB of memory, we could not train the ResNet-50 with a batch size of 128. As a consequence, we choose 128 and 64 as batch sizes for VGG-16 and ResNet-50, respectively, and all subsequent experiments are performed using these batch sizes. As it is commonly found in the literature, the larger batch size the better (specially for VGG-16 according to our experiments). Importantly, when used for regression, the performance of ResNet-50 seems to be quite independent of the batch size.

4 Statistical Analysis of the Results

Deep learning methods are generally based on stochastic optimization techniques, in which different sources of randomness, i.e. weight initialization, optimization and regularization procedures, have an impact on the results. In the analysis of optimization techniques and of batch sizes presented in the previous section we already observed some stochastic effects. While these effects did not forbid us to make reasonable optimization choices, other architecture design choices may be in close competition. In order to appropriately referee such competitions, one should draw conclusions based on rigorous statistical tests, rather than based on the average performance of a single training trial. In this section we describe the statistical procedures implemented to analyze the results obtained after several training trials. We use two statistical tools widely used in many scientific domains.

Generally speaking, statistical tests measure the probability of obtaining experimental results D if hypothesis H is correct, thus computing $P(D|H)$. The null hypothesis (H_0) refers to a general or default statement of a scientific experiment. It is presumed to be true until statistical evidence nullifies it for an alternative hypothesis (H_1). H_0 assumes that any kind of difference or significance observed in the data is due to chance. In this paper, H_0 is that none of the configurations under comparison in a particular experiment is any better, in terms of median performance, than other configurations. The estimated probability of rejecting H_0 when it is true is called *p-value*. If the p-value is less than the chosen level of significance α then the null hypothesis is rejected. Therefore, α indicates how extreme observed results must be in order to reject H_0. For instance, if the p-value is less than the predetermined significance level (usually 0.05, 0.01, or 0.001, indicated with one, two, or three asterisks, respectively), then the probability of the observed results under H_0 is less than the significance level. In other words, the observed result is highly unlikely to be the result of random chance. Importantly, the p-value only provides an index of the evidence against the null hypothesis, i.e. it is mainly intended to establish whether further research into a phenomenon could be justified. We consider it as one bit of evidence to either support or challenge accepting the null hypothesis, rather than as conclusive evidence of significance [58], [59], [60], and a statistically insignificant outcome should be interpreted as "absence of evidence, not evidence of absence" [61].

Statistical tests can be categorized into two classes: parametric and non-parametric. Parametric tests are based on assumptions (like normality or homoscedasticity) that are commonly violated when analyzing the performance of stochastic algorithms [62]. In our case, the visual inspection of the error measurements as well as the application of normality tests (in particular, the Lilliefors test) indicates a lack of normality in the data, leading to the use of non-parametric statistical tests.

Statistical tests can perform two kinds of analysis: pairwise comparisons and multiple comparisons. Pairwise statistical procedures perform comparisons between two algorithms, obtaining in each application a p-value independent from another one. Therefore, in order to carry out a comparison which involves more than two algorithms, multiple comparison tests should be used. If we try to draw a conclusion involving more than one pairwise comparison, we will obtain an accumulated error. In statistical terms, we are losing control on the Family-Wise Error Rate (FWER), defined as the probability of making one or more false discoveries (type I errors) among all the hypotheses when performing multiple pairwise tests. Examples of post-hoc procedures, used to control the FWER, are Bonferroni-Dunn [63], Holm [64], Hochberg [65], Hommel [66], Holland [67], Rom [68], or Nemenyi [69]. Following the recommendation of Derrac et al. [62], in this paper we use the Holm procedure to control the FWER.

Summarizing, once established that non-parametric statistics should be used, we decided to follow standard and well-consolidated statistical approaches: when pairwise comparisons have to be made, the Wilcoxon signed-rank test [36] is applied; when multiple comparisons have to be made (i.e. more than two methods are compared, thus increasing the number of pairwise comparisons), the FWER is controlled by applying the Bonferroni-Holm procedure (also called the Holm method) to multiple Wilcoxon signed-rank tests. Finally, the 95% confidence interval for the median of the MAE is reported.

4.1 Wilcoxon Signed-rank Test

The Wilcoxon signed-rank test [36] is a non-parametric statistical hypothesis test used to compare two related samples to assess the null hypothesis that the median difference between pairs of observations is zero. It can be used as an alternative to the paired Student's t-test, t-test for matched pairs,[2] when the population cannot be assumed to be normally distributed. We use Wilcoxon signed-rank test to evaluate which method is the best (i.e. the most recommendable configuration according with our results) and the worst (i.e. the less recommendable configuration according with our results). The statistical significance is displayed

2. Two data samples are matched/paired if they come from repeated observations of the same subject.

TABLE 1: Network baseline specification.

Network	BN	FT	DO	LR	PL
VGG-16	**BN**	$\mathbf{CB^5}$	$\mathbf{10{-}DO}$	$\rho(\mathbf{FC^2})$	$\mathbf{FC^2}$
ResNet-50	~~**BN**~~	$\mathbf{CB^3}$	-	$\rho(\mathbf{GAP})$	**GAP**

on each table using asterisks, as commonly employed in the scientific literature: * represents a p-value smaller than 0.05 but larger or equal than 0.01, ** represents a p-value smaller than 0.01 but larger or equal than 0.001, and *** represents a p-value smaller than 0.001. When more than one configuration has asterisks it implies that these configurations are significantly better than the others but there are no statistically significant differences between them. The worst performing configurations are displayed using circles and following the same criterion.

4.2 Confidence Intervals for the Median

Importantly, with a sufficiently large sample, statistical significance tests may detect a trivial effect, or they may fail to detect a meaningful or obvious effect due to small sample size. In other words, very small differences, even if statistically significant, can be practically meaningless. Therefore, since we consider that reporting only the significant p-value for an analysis is not enough to fully understand the results, we decided to introduce confidence intervals as a mean to quantify the magnitude of each parameter of interest.

Confidence intervals consist of a range of values (interval) that act as good estimates of the unknown population parameter. Most commonly, the 95% confidence interval is used. A confidence interval of 95% does not mean that for a given realized interval there is a 95% probability that the population parameter lies within it (i.e. a 95% probability that the interval covers the population parameter), but that there is a 95% probability that the calculated confidence interval from some future experiment encompasses the true value of the population parameter. The 95% probability relates to the reliability of the estimation procedure, not to a specific calculated interval. If the true value of the parameter lies outside the 95% confidence interval, then a sampling event that has occurred with a probability of 5% (or less) of happening by chance.

We can estimate confidence intervals for medians and other quantiles using the binomial distribution. The 95% confidence interval for the q-th quantile can be found by applying the binomial distribution [70]. The number of observations less than the q quantile will be an observation from a binomial distribution with parameters n and q, and hence has mean nq and standard deviation $\sqrt{(nq(1-q)}$. We calculate j and k such that: $j = nq - 1.96\sqrt{nq(1-q)}$ and $k = nq + 1.96\sqrt{nq(1-q)}$. We round j and k up to the next integer. Then the 95% confidence interval is between the j^{th} and k^{th} observations in the ordered data.

5 Network Variants

The statistical tests described above are used to compare the performance of each choice on the three data sets for the two base architectures. Due to the amount of time necessary to train deep neural architectures, we cannot compare all possible combinations, and therefore we must evaluate one choice at a time (e.g. the use of batch normalization). In order to avoid over-fiting, we use holdout as model validation tecnique, and test the generalization ability with an independent data set. In detail, we use 20% of training data for validation (26% in the case of FLD, because the validation set is explicitly provided in [18]). We use early stopping with a patience equal to four epochs (an epoch being a complete pass through the entire training set). In other words, the network is trained until the loss on the validation set does not decrease during four consecutive epochs. The two baseline networks are trained with the optimization settings chosen in Section 3. In this section, we evaluate the performance of different network *variants*, on the three problems.

Since ConvNets have a high number of parameters, they are prone to over-fitting. Two common regularization strategies are typically used [71]:

Batch Normalization (BN) was introduced to lead to fast and reliable network convergence [72]. In the case of VGG-16 (resp. ResNet-50), we cannot add (remove) a batch normalization layer deeply in the network since the pre-trained weights of the layers after this batch normalization layer were obtained without it. Consequently, in the case of VGG-16 we can add a batch normalization layer either right before REG (hence after the activation of FC^2, denoted by $\mathbf{BN}$), or before the activation of FC^2 (denoted by $\mathbf{BNB}$), or we do not use batch normalization ~~$\mathbf{BN}$~~. In ResNet-50 we consider only $\mathbf{BN}$ and ~~$\mathbf{BN}$~~, since the batch normalization layer before the activation of the last convolutional layer is there by default. Intuitively, VGG-16 will benefit from the configuration $\mathbf{BN}$, but not ResNet-50. This is due to the fact that the original VGG-16 does not exploit batch normalization, while ResNet-50 does. Using $\mathbf{BN}$ in ResNet-50 would mean finishing by convolutional layer, batch normalization, activation, GAP, batch normalization and REG. A priori we do no expect gains when using ResNet-50 with $\mathbf{BN}$ (and this is why the ResNet-50 baselines do not use BN), but we include this comparison for completeness.

Dropout (DO) is a widely used method to avoid over-fitting [71]. Dropout is not employed in ResNet-50, and thus we perform experiments only on VGG-16. We compare different settings: no dropout (denoted by $\mathbf{00{-}DO}$), dropout in FC^1 but not in FC^2 ($\mathbf{10{-}DO}$), dropout in FC^2 but not in FC^1 ($\mathbf{01{-}DO}$) and dropout in both ($\mathbf{11{-}DO}$).

Other approaches consist in choosing a network architecture that is less prone to over-fitting, for instance by chaging the number of parameters that are learned (fine-tuned). We compare three different strategies:

Fine-tuning depth. FT denotes the deepest block being trained, i.e. only the last layers are modified. Concerning VGG-16, we compare $\mathbf{CB^3}$, $\mathbf{CB^4}$, $\mathbf{CB^5}$ and $\mathbf{FC^1}$. Concerning ResNet-50, we compare $\mathbf{CB^2}$, $\mathbf{CB^3}$, $\mathbf{CB^4}$, $\mathbf{CB^5}$.

Regression layer. RL denotes the layer after which the BN and REG layers are added. $\rho(LR)$ denotes the model where the regression is performed on the output activations of the layer LR, meaning that on top of that layer we directly add batch normalization and linear regression layers. For VGG-16, we compare $\rho(\mathbf{CB^5})$, $\rho(\mathbf{FC^1})$ and $\rho(\mathbf{FC^2})$. For ResNet-50 we compare $\rho(\mathbf{CB^5})$ and $\rho(\mathbf{GAP})$.

Pooling layer. PL denotes the pooling strategy used.

TABLE 2: Impact of the batch normalization (BN) layer on VGG-16 and ResNet-50.

Data Set	BN	VGG-16				ResNet-50			
		MAE test	MSE train	MSE valid	MSE test	MAE test	MSE train	MSE valid	MSE test
Biwi	~~**BN**~~	[5.04 5.23]°°	[2.52 2.56]	[20.68 21.45]	[35.68 37.81]	[3.60 3.71]***	[1.25 1.27]	[21.49 22.25]	[17.15 18.14]
	BN	[3.66 3.79]***	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]	[4.59 4.69]	[2.18 2.22]	[22.93 23.63]	[28.56 30.07]
	BNB	[4.63 4.76]	[8.11 8.29]	[16.69 17.32]	[30.49 32.57]	–	–	–	–
FLD	~~**BN**~~	[3.67 3.90]	[21.19 21.45]	[22.26 22.76]	[19.70 22.25]	[1.96 2.05]	[5.21 5.25]	[6.85 6.95]	[5.79 6.39]
	BN	[2.61 2.76]***	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]	[1.92 2.01]	[4.81 4.86]	[6.83 6.94]	[5.70 6.34]
	BNB	[15.53 16.65]°°°	[300.3 304.5]	[300.4 307.7]	[326.9 369.4]	–	–	–	–
Parse	~~**BN**~~	[6.54 7.17]	[9.50 9.68]	[56.74 57.96]	[69.12 84.83]	[4.86 5.68]***	[0.64 0.64]	[29.40 30.21]	[43.58 55.71]
	BN	[4.90 5.59]***	[2.38 2.41]	[27.74 28.48]	[41.35 52.13]	[5.72 6.25]	[0.89 0.89]	[36.24 37.22]	[55.57 69.71]
	BNB	[11.20 12.68]°°°	[152.2 154.0]	[142.9 146.2]	[184.9 238.1]	–	–	–	–

TABLE 3: Impact of the dropout (DO) layer on VGG-16.

Dataset	DO	MAE test	MSE train	MSE valid	MSE test
Biwi	**00**	[4.47 4.60]°°°	[6.34 6.45]	[14.42 14.91]	[28.80 30.98]
	01	[3.56 3.67]	[3.35 3.40]	[12.00 12.40]	[17.52 18.54]
	10	[3.66 3.79]	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]
	11	[3.37 3.48]***	[3.46 3.53]	[11.66 12.04]	[15.39 16.38]
FLD	**00**	[2.55 2.70]	[8.60 8.71]	[10.53 10.74]	[10.16 11.23]
	01	[2.26 2.38]***	[7.81 7.90]	[9.19 9.38]	[7.87 8.68]
	10	[2.61 2.76]	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]
	11	[2.27 2.42]***	[7.59 7.68]	[9.10 9.29]	[7.94 8.92]
Parse	**00**	[4.83 5.44]	[1.39 1.41]	[27.28 28.09]	[40.38 51.11]
	01	[4.87 5.52]	[2.87 2.90]	[27.89 28.64]	[42.92 50.80]
	10	[4.90 5.59]	[2.38 2.41]	[27.74 28.48]	[41.35 52.13]
	11	[4.91 5.59]	[3.25 3.28]	[29.50 30.21]	[43.46 54.62]

TABLE 4: Impact of the finetunig depth (FT) on VGG-16.

Data	FT.	MAE test	MSE train	MSE valid	MSE test
Biwi	$\mathbf{FC^1}$	[5.13 5.27]	[4.12 4.20]	[29.18 30.44]	[37.10 39.03]
	$\mathbf{CB^5}$	[3.66 3.79]***	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]
	$\mathbf{CB^4}$	[4.88 5.03]	[4.32 4.40]	[15.96 16.54]	[30.95 32.71]
	$\mathbf{CB^3}$	[5.33 5.46]°°°	[9.30 9.51]	[33.48 34.52]	[38.66 40.34]
FLD	$\mathbf{FC^1}$	[3.32 3.47]	[4.00 4.04]	[16.02 16.29]	[16.56 18.25]
	$\mathbf{CB^5}$	[2.61 2.76]***	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]
	$\mathbf{CB^4}$	[2.85 3.10]	[6.31 6.40]	[7.79 7.97]	[12.52 14.45]
	$\mathbf{CB^3}$	[3.48 3.74]°°°	[10.83 10.98]	[11.80 12.05]	[18.52 21.96]
Parse	$\mathbf{FC^1}$	[5.50 6.31]°	[1.31 1.33]	[37.29 38.33]	[49.61 63.18]
	$\mathbf{CB^5}$	[4.90 5.59]***	[2.38 2.41]	[27.74 28.48]	[41.35 52.13]
	$\mathbf{CB^4}$	[4.92 5.87]	[2.42 2.46]	[29.91 30.61]	[43.04 57.09]
	$\mathbf{CB^3}$	[5.38 6.13]	[2.90 2.95]	[37.49 38.26]	[49.21 65.75]

TABLE 5: Impact of the fine tuning depth (FT) on ResNet-50.

Data	FT	MAE test	MSE train	MSE valid	MSE test
Biwi	$\mathbf{CB^5}$	[8.69 8.90]°°°	[32.57 33.17]	[140 145.1]	[102.5 107.8]
	$\mathbf{CB^4}$	[3.40 3.51]***	[0.87 0.89]	[17.85 18.43]	[15.98 16.86]
	$\mathbf{CB^3}$	[3.60 3.71]	[1.25 1.27]	[21.49 22.25]	[17.15 18.14]
	$\mathbf{CB^2}$	[4.17 4.30]	[1.41 1.43]	[26.31 27.18]	[24.19 25.34]
FLD	$\mathbf{CB^5}$	[8.79 9.30]°°°	[114.6 116.2]	[120.2 123.1]	[113.1 127.1]
	$\mathbf{CB^4}$	[2.21 2.31]	[4.48 4.52]	[8.10 8.24]	[7.36 8.01]
	$\mathbf{CB^3}$	[1.96 2.05]***	[5.21 5.25]	[6.85 6.95]	[5.79 6.39]
	$\mathbf{CB^2}$	[2.04 2.14]	[5.10 5.16]	[7.00 7.13]	[6.28 6.89]
Parse	$\mathbf{CB^5}$	[8.27 9.28]°°°	[54.22 54.99]	[77.32 78.88]	[102.5 132.5]
	$\mathbf{CB^4}$	[5.07 5.86]	[0.90 0.91]	[31.74 32.54]	[44.85 56.78]
	$\mathbf{CB^3}$	[4.86 5.68]***	[0.64 0.64]	[29.40 30.21]	[43.58 55.71]
	$\mathbf{CB^2}$	[5.02 5.84]	[0.84 0.85]	[30.48 31.29]	[45.65 61.04]

Often, pooling is used to convert convolutional maps into vectors. In the case of VGG-16, the 2D feature maps of CB^5 are converted via a flattening layer Fl that does not reduce the dimension. Alternatively, we could replace Fl by global (max or average) pooling, denoted by GMP and GAP respectively. ResNet-50 already uses GAP and hence we can only compare to GMP. The models are denoted $\mathbf{GAP}$, $\mathbf{GMP}$, for both, and $\mathbf{FC^2}$ for VGG-16.

The settings corresponding to our baselines are detailed in Table 1. These baselines are chosen in the light of preliminary experiments (not reported) and without important changes with respect to the original design. The background color is gray, as it will be for these two configurations in the remainder of the paper. We now discuss the results obtained using the previously discussed variants.

5.1 Batch Normalization

Table 2 shows the results obtained with the various choices of batch normalization. In the case of VGG-16, we observe that the impact of adding a batch normalization layer after the activations (i.e. $\mathbf{BN}$) is significant and beneficial compared to the other two options. In the case of FLD, we notice that the problem with ~~$\mathbf{BN}$~~ and $\mathbf{BNB}$ occurs at training since the final training MSE score is much higher than the one obtained with $\mathbf{BN}$. Interestingly, on the Biwi data set we observe that the training MSE is better with ~~$\mathbf{BN}$~~ but $\mathbf{BN}$ performs better on the validation and the test sets. We conclude that in the case of Biwi, $\mathbf{BN}$ does not help the optimization, but increases the generalization ability. The high beneficial impact observed in VGG-16 justifies that we use it in the baseline network. The exact opposite trend is observed when running the same experiments with ResNet-50. Indeed, $\mathbf{BN}$ neither improves the optimization (with the exception of FLD, which appears to be quite insensitive to BN using ResNet-50), nor the generalization ability. As discussed, this is expected because ResNet-50 uses already batch normalization (before the activation).

5.2 Dropout Ratio

Table 3 shows the results obtained when comparing different dropout strategies with VGG-16 (ResNet-50 does not have fully connected layers and thus no dropout is used). In the light of these results, it is encouraged to use dropout in both fully connected layers. However, for Parse, this does not seem to have any impact, and on FLD the use of dropout in the first fully connected layer seems to have very mild impact, at least in terms of MAE performance on the test set. Since on the Biwi data set the $\mathbf{00-DO}$ strategy is significantly worse than the other strategies, and not especially competitive in the other two data sets, we would suggest not to use this strategy. Globally, $\mathbf{11-DO}$ is the safest option (best for Biwi/FLD, equivalent for Parse).

5.3 Fine Tuning Depth

Tables 4 and 5 show results obtained with various fine-tuning depth values, as described in Section 5, for both VGG-16 and ResNet-50. In the case of VGG-16, we observe

TABLE 6: Impact of the regressed layer (RL) for VGG-16.

Data	RL	MAE test	MSE train	MSE valid	MSE test
Biwi	$\rho(\mathbf{FC^2})$	[3.66 3.79]***	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]
	$\rho(\mathbf{FC^1})$	[5.17 5.31]°°°	[9.49 9.66]	[17.84 18.40]	[36.32 38.53]
	$\rho(\mathbf{CB^5})$	[4.64 4.75]	[5.38 5.47]	[16.96 17.49]	[28.84 29.85]
FLD	$\rho(\mathbf{FC^2})$	[2.61 2.76]***	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]
	$\rho(\mathbf{FC^1})$	[3.51 3.68]	[9.87 10.00]	[12.98 13.23]	[18.22 19.97]
	$\rho(\mathbf{CB^5})$	[3.61 3.82]°°	[16.55 16.74]	[18.49 18.84]	[19.00 21.38]
Parse	$\rho(\mathbf{FC^2})$	[4.90 5.59]**	[2.38 2.41]	[27.74 28.48]	[41.35 52.13]
	$\rho(\mathbf{FC^1})$	[4.99 5.66]	[2.61 2.64]	[30.13 30.74]	[43.25 55.67]
	$\rho(\mathbf{CB^5})$	[5.58 6.14]°°°	[2.83 2.85]	[34.46 35.12]	[52.62 61.34]

TABLE 7: Impact of the regressed layer (RL) for ResNet-50.

Data	RL	MAE test	MSE train	MSE valid	MSE test
Biwi	$\rho(\mathbf{GAP})$	[3.60 3.71]***	[1.25 1.27]	[21.49 22.25]	[17.15 18.14]
	$\rho(\mathbf{CB^5})$	[3.61 3.73]	[0.71 0.72]	[15.42 15.92]	[17.60 18.55]
FLD	$\rho(\mathbf{GAP})$	[1.96 2.05]	[5.21 5.25]	[6.85 6.95]	[5.79 6.39]
	$\rho(\mathbf{CB^5})$	[1.61 1.70]***	[1.77 1.79]	[4.81 4.90]	[3.98 4.36]
Parse	$\rho(\mathbf{GAP})$	[4.86 5.68]***	[0.64 0.64]	[29.30 30.09]	[43.58 55.71]
	$\rho(\mathbf{CB^5})$	[5.56 6.24]	[0.48 0.48]	[36.76 37.61]	[54.69 67.21]

a behavior of $\mathbf{CB^5}$ similar to the one observed for batch normalization. Indeed, $\mathbf{CB^5}$ may not be the best choice in terms of optimization ($\mathbf{FC^1}$ exhibits smaller training MSEs) but it is the best one in terms of generalization ability (for MSE validation and test as well as MAE test). For VGG-16, this result is statistically significant for all three data sets. In addition, the results shown in Table 4 also discourage to use $\mathbf{CB^3}$. It is more difficult to conclude in the case of ResNet-50. While for two of the data sets the recommendation is to choose the baseline (i.e. fine tune from CB^3), ResNet-50 on Biwi selects the model $\mathbf{CB^4}$ by a solid margin in a statistically significant manner. Therefore, we suggest than, when using ResNet-50 for regression, one still runs an ablation study varying the number of layers that are tuned. In this ablation study, the option $\mathbf{CB^5}$ should not necessarily be included, since for all data sets this option is significantly worse than the other ones.

5.4 Regression Layer

Tables 6 and 7 show the results obtained when varying the regression layer, for VGG-16 and ResNet-50 respectively. In the case of VGG-16 we observe a strongly consistent behavior, meaning that the best method in terms of optimization performance is also the method that best generalizes (in terms of MSE validation and test as well as MAE test). Regressing from the second fully connected layer is a good choice when using VGG-16, whereas the other choices may be strongly discouraged depending on the data set. The results on ResNet-50 are a bit less conclusive. The results obtained are all statistically significant but different depending on the data set. Indeed, experiments on Biwi and Parse point to use $\mathbf{GAP}$, while results on FLD point to use $\mathbf{CB^5}$. However, we can observe that the confidence intervals of $\rho(\mathbf{GAP})$ and $\rho(\mathbf{CB^5})$ with ResNet-50 on Biwi are not that different. This means that the difference between the two models is small while consistent over the test set images.

5.5 Pooling Layer

Table 8 shows the results obtained for different pooling strategies on VGG-16 and ResNet-50. Regarding VGG-16, we observe that except for FLD, the best option is to keep the flatten layer, and therefore discard the use of pooling layers ($\mathbf{GMP}$ or $\mathbf{GAP}$). The case of FLD is different, since the results would suggest that it is significantly better to use either $\mathbf{GAP}$ or $\mathbf{GMP}$. Similar conclusions can be drawn from the results obtained with ResNet-50 in the sense that the standard configuration ($\mathbf{GAP}$) is the optimal one for Biwi and Parse, but not for FLD. In the case of FLD, the results point to choose $\mathbf{GMP}$.

5.6 Discussion on Network Variants

This section summarized the results obtained with different network variants. Firstly, evaluating the use of batch normalization is mandatory, but it would appear that conclusions are constant through different data sets, hence there is no need to run extensive experimentation to select the best option. Second, regarding the number of layers to be fine tuned, we recommend to exploit the $\mathbf{CB^5}$ model for VGG-16 unless strong reasons would support a different choice. For ResNet-50 the choice would be between $\mathbf{CB^3}$ and $\mathbf{CB^4}$ (but definitely not $\mathbf{CB^5}$). Third, regression should be done from FC^2 in VGG-16 (we do not recommend any of the tested alternatives) and either from GAP or from CB^5 in ResNet-50. Fourth, in terms of dropout, as it is classically observed in the literature, one should use dropout in VGG-16, especially in the second layer (this experiment does not apply to ResNet-50). Finally, the pooling strategies should all be tested, since their optimality depends upon the network and data set in a statistically significant manner.

Very importantly, in case of limited resources (e.g. computing time), taking the suboptimal choice may not lead to a crucial difference with respect to the perfect combination of parameters. However, one must avoid the cases in which the method is proven to be significantly worse than the other, because in those cases performances (in train, validation and test) have proven to be evidently different.

6 Data Pre-Processing

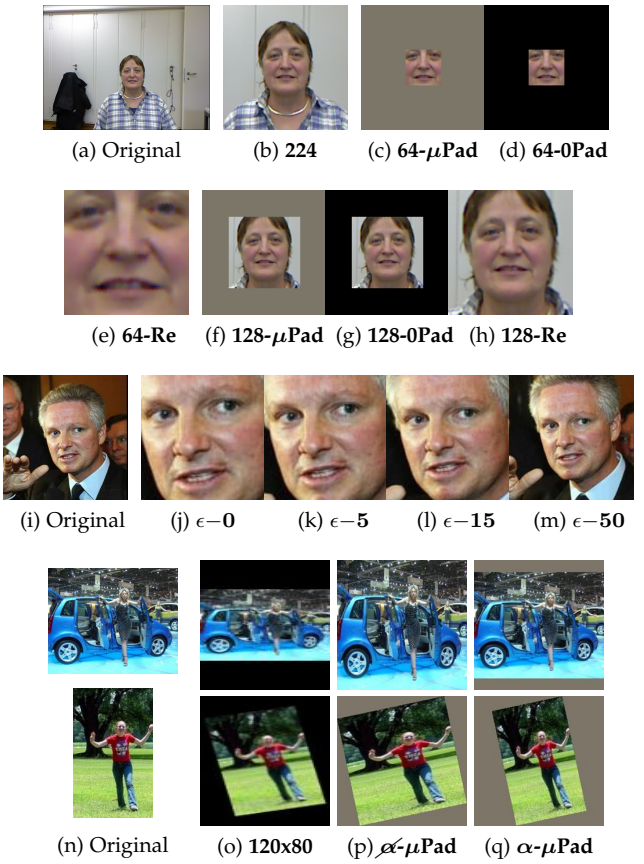
In this section we discuss the different data pre-processing techniques that we consider in our benchmark. Because VGG-16 has two fully connected layers, we are constrained to use the pre-defined input size of 224×224 pixels. Hence, we systematically resize the images to this size. For the sake of a fair comparison, the very same input images are given to both networks. Firstly, we evaluate the impact of mirroring the training images (not used for test). Table 9 reports the results when evaluating the impact of mirroring. The conclusion is unanimous: mirroring is statistically better for all configurations. In addition, in most of the cases the confidence intervals are disjoint meaning that with high probability the output obtained when training with mirroring will have lower error than training without mirroring.

Since the three data sets used in our study have different characteristics, the pre-processing steps differ from data set to data set. Importantly, in all cases the pre-processing baseline technique is devised from the common usage of the data sets in the recent literature. Below we specify the baseline pre-processing as well as other tested pre-processing alternatives for each data set.

Biwi. The baseline for the Biwi data set is inspired from [20], [73], where the authors crop a 64×64 and a

TABLE 8: Impact of the pooling layer (PL) on VGG-16 and ResNet-50.

Data Set	PL	VGG-16				ResNet-50			
		MAE test	MSE train	MSE valid	MSE test	MAE test	MSE train	MSE valid	MSE test
Biwi	**GMP**	[3.97 4.08]	[4.03 4.11]	[18.25 18.99]	[21.19 22.38]	[3.64 3.75]	[1.48 1.50]	[20.62 21.23]	[17.80 18.88]
	GAP	[3.99 4.09]	[2.75 2.80]	[20.71 21.40]	[21.03 22.07]	[3.60 3.71]***	[1.25 1.27]	[21.49 22.25]	[17.15 18.14]
	FC2	[3.66 3.79]***	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]	–	–	–	–
FLD	**GMP**	[2.50 2.64]***	[3.02 3.06]	[7.65 7.81]	[9.56 11.07]	[1.75 1.82]***	[2.17 2.19]	[5.21 5.30]	[4.67 5.22]
	GAP	[2.53 2.67]**	[5.18 5.23]	[9.06 9.22]	[9.69 10.64]	[1.96 2.05]	[5.21 5.25]	[6.85 6.95]	[5.79 6.39]
	FC2	[2.61 2.76]	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]	–	–	–	–
Parse	**GMP**	[5.55 6.06]	[1.17 1.18]	[30.00 30.67]	[52.17 61.86]	[5.74 6.51]	[0.89 0.90]	[36.39 37.17]	[54.63 69.10]
	GAP	[5.61 6.07]	[1.48 1.50]	[31.46 32.15]	[51.33 60.57]	[4.86 5.68]***	[0.64 0.64]	[29.30 30.09]	[43.58 55.71]
	FC2	[4.90 5.59]***	[2.38 2.41]	[27.74 28.48]	[41.35 52.13]	–	–	–	–



(a) Original (b) **224** (c) **64-μPad** (d) **64-0Pad**

(e) **64-Re** (f) **128-μPad** (g) **128-0Pad** (h) **128-Re**

(i) Original (j) $\epsilon-\mathbf{0}$ (k) $\epsilon-\mathbf{5}$ (l) $\epsilon-\mathbf{15}$ (m) $\epsilon-\mathbf{50}$

(n) Original (o) **120x80** (p) **α-μPad** (q) **α-μPad**

Fig. 3: Pre-processed examples for the Biwi (first and second rows), FLD (third row) and Parse (fourth row) data sets.

224×224 window respectively, centered on the face. We investigate the use of three window sizes: 64×64, 128×128 and 224×224. The latter is referred to as **224**. When cropping windows smaller than 224×224 pixels, we investigate three possibilities: resize (denoted by **64-Re** and **128-Re**), padding with the mean value of ImageNet (**64-μPad** and **128-μPad**) and padding with zeros (**64-0Pad** and **128-0Pad**). Examples of this pre-processing steps are shown in Figures 3a-3h.

FLD. In the case of the facial landmark data set, original images and face bounding boxes are provided. Similarly to the Biwi data set, the issue of the amount of context information is investigated. [18] proposes to expand the bounding boxes and then to adopt a cascade strategy to refine the input regions of the networks. As we want to keep our processing as general as possible, we adopt the following procedure: the face bounding box is expanded by $\epsilon\%$ in each direction. We compare four different expanding ratios: 0%, 5%, 15% and 50%. These are denoted with **ϵ-0**, **ϵ-5**, **ϵ-15**, and **ϵ-50** respectively, see Figure 3i to 3m.

Parse. When using this data set in [17], the images were resized to 120×80 pixels. In our case, we resize the images to fit into a rectangle of this size and pad to a squared image with zeros, followed by resizing to 224×224 pixels. This strategy is referred to as **120x80**. We also consider directly resizing into 224×224 images, hence without keeping the aspect ratio, and padding with the mean value of ImageNet (**α-μPad**), or keeping the aspect ratio (**α-μPad**). Examples are shown in Figures 3n-3q. The two last strategies are also employed using zero-padding (**α-0Pad** and **α-0Pad**).

Table 10 reports the results obtained by the different pre-processing techniques for each data set for both VGG-16 and ResNet-50. The point locations are represented by their pixel Cartesian coordinates in the case of FLD and Parse. When we evaluate a data pre-processing strategy, the images are geometrically modified and the same transformation is applied to the annotations. Consequently, the errors cannot be directly compared between two different pre-processing strategies. For instance, in the last row of Figure 3 a 5-pixel error for the right elbow location may be acceptable in the case of **α-0Pad** but may correspond to a confusion with a shoulder location in the case of **α-0Pad**. In order to compare the pre-processing strategies, we transform all the errors into a common space. We choose common spaces such that the aspect ratio of the original images is kept. For Parse, the errors are compared in the original image space (i.e. before any resize or crop operation). In all the experiments of Section 5, the errors are reported in the space of **120x80**. This choice is justified by the fact that the MSE reported are the exact loss values used to optimize and proceed to early stopping. The drawback of this choice is that the errors on Parse obtained in Tables 2-9 are not directly comparable with those of Table 10. In the case of FLD, **ϵ-0** space corresponds to original detections. However, as the transformations between spaces are only linear scalings, the comparison can be performed in any space without biasing the results. Therefore, we chose to report the errors in the space corresponding to **ϵ-5** to allow direct comparison with Tables 2 to 9. In the case of Biwi, as the head angle is independent of the pre-processing strategy, no transformation is required to compare the strategies.

Regarding the experiments on Biwi, we notice that the best strategy for VGG-16 is **224**, whereas for ResNet-50 is **64-Re**. Having said that, the differences with the second best (in terms of the confidence interval) are below 1 degree

TABLE 9: Impact of the mirroring (Mirr.) on VGG16 and ResNet50.

Data Set	Mirr.	VGG-16				ResNet-50			
		MAE test	MSE train	MSE valid	MSE test	MAE test	MSE train	MSE valid	MSE test
Biwi	Yes	[3.66 3.79]***	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]	[3.60 3.71]***	[1.25 1.27]	[21.49 22.25]	[17.15 18.14]
	No	[5.49 5.67]	[9.09 9.36]	[24.22 25.61]	[42.55 45.73]	[4.46 4.57]	[1.39 1.42]	[19.70 20.83]	[27.34 28.60]
FLD	Yes	[2.61 2.76]***	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]	[1.96 2.05]***	[5.21 5.25]	[6.85 6.95]	[5.79 6.39]
	No	[3.06 3.24]	[14.13 14.38]	[15.29 15.73]	[14.28 15.68]	[2.05 2.13]	[4.41 4.46]	[7.04 7.20]	[6.43 7.15]
Parse	Yes	[4.90 5.59]***	[2.38 2.41]	[27.74 28.48]	[41.35 52.13]	[4.86 5.68]***	[0.64 0.64]	[29.40 30.21]	[43.58 55.71]
	No	[5.08 5.76]	[2.31 2.36]	[28.98 30.14]	[45.15 57.08]	[5.88 6.62]	[1.14 1.16]	[42.99 44.38]	[59.30 77.70]

TABLE 10: Impact of the data pre-processing on VGG-16 and ResNet-50.

Data Set &		VGG-16				ResNet-50			
	Pre-processing	MAE test	MSE train	MSE valid	MSE test	MAE test	MSE train	MSE valid	MSE test
Biwi	**128-μPad**	[3.93 4.02]	[5.89 6.00]	[15.34 15.85]	[21.13 21.94]	[3.47 3.55]	[1.05 1.06]	[18.71 19.35]	[16.41 17.13]
	128-Re	[3.87 3.97]	[4.98 5.06]	[13.23 13.75]	[19.80 20.79]	[3.18 3.25]	[0.97 0.98]	[16.37 16.91]	[13.63 14.25]
	128-0Pad	[3.97 4.05]	[7.83 7.98]	[15.55 16.09]	[21.51 22.31]	[3.20 3.28]	[1.52 1.55]	[17.94 18.52]	[14.45 15.03]
	64-μPad	[4.48 4.63]	[6.09 6.22]	[21.52 22.31]	[27.91 29.77]	[3.34 3.47]	[1.45 1.47]	[21.18 21.99]	[15.55 16.73]
	64-Re	[4.06 4.21]	[6.79 6.92]	[15.05 15.59]	[22.72 24.32]	[2.98 3.07]**	[0.99 1.00]	[13.18 13.66]	[12.20 12.90]
	64-0Pad	[4.80 5.02]°°°	[12.15 12.40]	[19.35 20.05]	[32.72 35.33]	[3.29 3.42]	[3.34 3.40]	[19.21 20.01]	[14.91 16.02]
	224	[3.66 3.79]***	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]	[3.60 3.71]°°°	[1.25 1.27]	[21.49 22.25]	[17.15 18.14]
FLD	**ϵ-0**	[2.25 2.36]***	[7.61 7.69]	[8.94 9.11]	[7.69 8.57]	[1.63 1.73]***	[2.41 2.44]	[5.07 5.16]	[4.23 4.63]
	ϵ-5	[2.61 2.76]	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]	[1.96 2.05]	[5.21 5.25]	[6.85 6.95]	[5.79 6.39]
	ϵ-15	[2.35 2.48]	[8.42 8.50]	[9.88 10.07]	[8.37 9.36]	[2.00 2.08]	[4.48 4.53]	[6.59 6.70]	[5.95 6.43]
	ϵ-50	[2.96 3.17]°°°	[11.68 11.82]	[13.33 13.60]	[13.32 15.31]	[2.59 2.72]°°°	[7.12 7.18]	[9.92 10.07]	[10.11 11.20]
Parse	**α-μPad**	[9.99 11.52]°°°	[8.47 8.62]	[114.5 117.5]	[161.4 226.1]	[11.61 13.63]°°°	[4.29 4.35]	[180.7 185.8]	[235.3 311.5]
	α-0Pad	[9.72 11.13]	[9.77 9.95]	[116.1 119.2]	[165.3 212.9]	[10.90 12.54]	[3.34 3.39]	[158.5 163.7]	[207.9 279.6]
	$\not\alpha$-μPad	[8.54 9.56]***	[6.22 6.35]	[106.9 109.4]	[125.5 158.7]	[9.30 11.28]***	[2.83 2.87]	[140.3 144.5]	[160.8 218.3]
	$\not\alpha$-0Pad	[8.39 9.33]***	[8.23 8.39]	[111.9 115.3]	[120.9 152.9]	[9.15 10.82]***	[3.35 3.39]	[136.6 140.7]	[151.7 194.1]
	120x80	[9.55 11.34]	[12.05 12.23]	[130.6 134.3]	[161.0 215.8]	[9.14 10.57]***	[3.40 3.45]	[136.5 140.4]	[155.4 204.7]

in MAE. Interestingly, we observe that in both cases two strategies are significantly worse: **64-0Pad** for VGG-16 and **224** for ResNet-50. The fact that the same strategy **224** is significantly the best for VGG-16 and significantly the worst for ResNet-50 demonstrates that a serious ablation study of pre-processing strategies is required on Biwi. The experiments on the FLD data set are clearly more conclusive than the ones on Biwi. Indeed, for both architectures the **ϵ-0** strategy is significantly better and the **ϵ-50** is significantly worse. The differences range from approxiamtively 0.1 pixel (with respect to the second best strategy) to almost one pixel (with respect to the worst strategy). Regarding the experiments based on Parse, we obtain a statistical tie. Indeed, **$\not\alpha$-μPad** and **$\not\alpha$-0Pad** with VGG-16 are better than the rest, but without statistical differences between them. A similar behavior is found in ResNet-50 including **120x80**.

The results on data pre-processing and mirroring behave quite differently. While mirroring results are consistent over data sets and networks, and therefore we recommend to systematically use mirroring for training, the results on data pre-processing require more detailed discussion. Indeed, one must be extremely careful when choosing the pre-processing on a data set. In the three cases we can see how the performance can present strong variations depending on the pre-processing used. Even more dangerous, when comparing the two architectures, the conclusions of which is best can change depending on the pre-processing. More generally, the superiority of a newly developed method with respect to the state-of-the-art may strongly depend on the data pre-processing. Ideally, standard ways to pre-process data so as to avoid unsupported conclusions should be established. In the case of Biwi, fixing one pre-processing strategy would bias the decision either towards ResNet-50 (**64-Re** or **64-0Pad**), or towards VGG-16 (**224**). But the fairest comparison would be ResNet-50 with **64-Re** against VGG-16 with **224**. This indicates a clear interest on discussing different pre-processing strategies when presenting a new data set/architecture.

7 Positioning of Vanilla Deep Regression

The aim of this section is to position the vanilla deep regression methods based on VGG-16 and ResNet-50 with respect to the state-of-the-art. Importantly, the point here is not to outperform all previous methods that are specifically designed for each of the studied tasks, but rather to understand how far or how close a vanilla deep regression method is from the state-of-the-art. Another question is whether or not a "correctly fine-tuned base network" (meaning having chosen the optimal network variant and pre-processing strategy) compares to some of the methods in the literature.

In order to compare to the state-of-the-art, we employ the metric proper to each problem. Importantly, since the statistical tests are run on the per-image error, and these intermediate results are unavailable for state-of-the-art methods, we cannot run the same experimental evaluation as in previous sections. Furthermore, given that the experimental protocols of many papers are not fully detailed (we take the numbers directly from the references), we are uncertain that all row of Tables 11, 12 and 13, are equally significant. In other words, the differences exhibited between the various state-of-the-art methods are to be taken with extreme care, since there is not other metric than the average performance (of probably one single run with a particular network design and data pre-processing strategy).

Since in the previous section we observed that the pre-processing strategy is crucial for the performance, we report

TABLE 11: Comparison of different methods on the FLD data set. Failure rate are given in percentage. We report the median and best run behavior of the worse and best data pre-processing strategies for each baseline network.

	Left Eye	Right Eye	Nose	Left Mouth	Right Mouth	Average
VGG-16 ϵ**-50**	11.65/7.23	9.64/6.83	16.47/13.65	10.04/10.04	12.45/8.43	12.05/9.24
ResNet-50 ϵ**-50**	10.44/4.02	6.83/4.02	5.22/7.63	6.43/6.02	6.83/5.22	7.15/5.38
VGG-16 ϵ**-0**	1.20/0.80	0.40/0.00	3.21/2.41	3.61/2.81	3.61/2.41	2.41/1.69
ResNet-50 ϵ**-0**	0.80/1.20	0.00/0.00	0.00/0.40	2.01/0.80	1.20/1.20	0.80/0.72
Sun et al. [18]	0.67	0.33	0.00	1.16	0.67	0.57

TABLE 12: Comparison of different methods on the Parse database. Strict PCP scores are reported. We report the median and best run behavior of the worse and best data pre-processing strategies for each baseline network.

	Head	Torso	Upper Legs	Lower Legs	Upper Arms	Lower Arms	Full Body
VGG-16 α-μ**Pad**	68.85/70.49	85.25/83.61	77.05/79.51	63.11/63.93	45.90/45.08	40.16/42.62	60.66/61.64
ResNet-50 α-μ**Pad**	57.4/65.6	68.9/73.8	71.3/74.6	55.7/62.3	39.3/45.1	36.1/41.0	53.1/58.5
VGG-16 α**-0Pad**	77.05/68.85	90.16/88.52	80.33/81.15	63.11/67.21	50.82/52.46	44.26/50.00	64.43/65.90
ResNet-50 $\mathbf{120 \times 80}$	67.2/68.9	78.7/80.3	77.9/77.9	65.6/63.9	45.1/50.0	46.7/45.9	61.6/62.1
Andriluka et al [74]	72.7	86.3	66.3	60.0	54.0	35.6	59.2
Yang & Ramanan [75]	82.4	82.9	68.8	60.5	63.4	42.4	63.6
Pishchulin et al. [76]	77.6	90.7	80.0	70.0	59.3	37.1	66.1
Johnson et al. [77]	76.8	87.6	74.7	67.1	67.3	45.8	67.4
OuYang et al. [78]	89.3	89.3	78.0	72.0	67.8	47.8	71.0
Belagiannis et al. [17]	91.7	98.1	84.2	79.3	66.1	41.5	73.2

TABLE 13: Comparison of different methods on the Biwi head-pose data set (MAE in degrees). The superscript† denotes the use of extra training data (see [15] for details). We report the median and best run of the worst and best data pre-processing strategies for each baseline network.

	Pitch	Yaw	Roll	Mean
VGG-16 **64-0Pad**	10.11/4.75	4.70/3.82	4.16/4.18	6.33/4.25
ResNet-50 **224**	4.73/4.53	2.96/3.49	4.91/4.10	4.20/4.04
VGG-16 **224**	4.51/4.02	4.68/3.74	3.22/3.28	4.14/3.68
ResNet-50 **64-Re**	5.98/5.22	2.39/2.37	3.93/4.04	4.10/3.88
Liu et al. [15]	6.1	6.0	5.7	5.94
Liu et al. [15]†	4.5	4.3	2.4	3.73
Mukherjee et al. [42]	5.18	5.67	–	5.43
Drouard et al. [73]	5.43	4.24	4.13	4.60
Lathuiliere et al. [20]	4.68	3.12	3.07	3.62

the results of the two baseline architectures (Table 1) combined with the best and worst data pre-processing strategy for each architecture and each problem. For each of these four combinations of base architecture and pre-processing strategy, we run five network trainings and report the performance of the "best" and "average" runs (selected from the overall performance of the problem at hand, i.e., last column of the tables below). Consequently, it is possible that for a particular sub-task the "average" run has better performance than the "best" run.

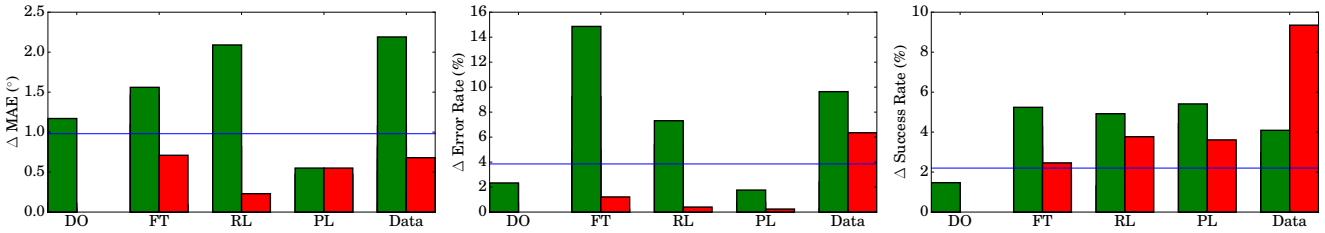
Table 11 reports the failure percentage of different methods on the FLD dataset where errors larger than 5% of the bounding box width are counted as failures. First of all, a correctly fine-tuned simple regressor is in competition with the state-of-the-art. Even if in average the best vanilla deep regression methods do not outperform the state of the art, they obtain quite decent results (e.g. the average run with the optimal strategy for ResNet-50 is only 0.23% worse than the state of the art).

Regarding the Parse dataset, Table 12 reports the results of six published methods. Performance is measured using the strict PCP (Percentage of Correctly estimated Parts) score. According to strict PCP, a limb is considered as correctely estimated if the distances between the estimated and true joint locations are smaller than 50% of the limb length. Vanilla deep regression does not outperform the state of the art. However, it is important to notice that regarding the Head, Torso, Upper Legs and Lower Legs, the best vanilla deep regression variant outperforms half of existing methods and in the case of Lower Arms, it outperforms the state of the art. On an average, vanilla deep regression competes well with half of the methods in the literature. We believe this is a very interesting result from two perspectives. On the one side, vanilla deep regression methods compete with the state-of-the-art in different tasks as long as they are properly fine tuned. On the other side, methods specifically designed for a problem may often outperform standard baselines, and therefore it is worth pursuing research in most applications. In other words, standard baselines may offer a good starting point, but developing problem-specific methods is worth to push the performances.

Finally, Table 13 reports the results on the Biwi dataset using MAE. Correctly fine-tuned deep regression is clearly competing well with the state of the art (even with a method using extra training data). Overall, the best deep regression is very close to the state-of-the-art on the Biwi dataset.

8 Overall Discussion

Figure 4 displays the difference, in terms of problem-specific metrics (see Section 7), between the best and worst median runs per configuration. More precisely, we compute the metric for each one of the 5 runs and, from there, compute the median for each configuration. Finally, we display the difference between the best and worst medians. The main



(a) Biwi (b) FLD (c) Parse

Fig. 4: Performance improvement between the best and worst median runs for each configuration on (a) Biwi, (b) FLD and (c) Parse for four network variants and data pre-processing (Data), for VGG-16 (green) and ResNet-50 (red). Horizontal blue lines indicate the improvement obtained by recent approaches when they outperform existing methods. Importantly, all the methods compared in this figure obtained statistically significant differences with $p < 0.001$, except for DO on Parse.

goal of this figure is to visualize the impact in performance of each network variant and the data-preprocessing per network and problem. We exclude those unequivocal cases where one particular configuration offers a clear and systematic improvement or deterioration with respect to other configurations. These are: batch normalization (see Table 2), $\mathbf{CB^5}$ when fine-tunning ResNet-50 (see Table 5) and mirroring (see Table 9). The results of VGG-16 are shown in green while those of ResNet-50 are shown in red.

The first element to consider is the improvement that can be achieved when the network variants are properly selected (up to 2° MAE in Biwi, up to 14% error rate in FLD, and up to 6% success rate in Parse). Although, in Figure 4, the margins of improvement may seem small in some cases, we are dealing with performance increases that would be sufficient to clearly outperform prior results. In order to give a visual reference in this regard, for each problem, we add a horizontal line that indicates the improvement obtained by recent approaches when they outperform preceding methods. Such lines are drawn by computing the difference in performance between the best performing method in Tables 13, 11 and 12, for Biwi, FLD and Parse, respectively, and the second best performing method in the corresponding original papers [20], [17], and [18]. We do not suggest that, in a particular task, the precise adjustment of, for instance FT, will provide results superior to the state of the art. We only give a reference of the magnitude of the potential performance improvement (or deterioration) between different variants of the same network, and claim that this potential improvement is not only significant but also far from being negligible.

The aspect that attracts the most immediate attention is that the margin of improvement in VGG-16 is generally larger than in ResNet-50, as shown by the gap between the top of the green and red columns in Figure 4 (with the exceptions of PL in Biwi and data-preprocessing in Parse). Regarding the network variants, we conclude that VGG-16 obtains highly variable results while, on the contrary, the behavior of ResNet-50 is generally more stable. From a practical point of view, a substantial improvement can be expected from the careful configuration of the VGG-16 variants. With respect to the data pre-processing, the improvement may also be quite high, and which network is subject to larger improvement is highly dependent on the problem at hand.

Generally speaking, the most critical factors seem to be FT, RL and data pre-processing. The only exception is VGG-16 in Parse, where PL offers a larger improvement than data pre-processing. As a consequence, it is preferable to invest time trying different FT and RL strategies than different variants of PL or DO, which appear to be elements with least variability between the best and worst configurations.

Finally, we observe that the impact of proper data pre-processing is crucial for the performance of vanilla deep regression. Moreover, the margin of improvement highly depends both on the problem at hand and on the architecture. In addition, given that for some datasets the optimal data pre-processing strategy depends on the network, we argue that the pre-processing can have huge impact on the results and we draw two conclusions. First, at practical level, we strongly recommend to compare different data pre-processing strategies in order to improve the performance. Second, at a scientific level, our analysis illustrate that the employed data pre-processing procedures must be carefully detailed when it comes to reporting results. Different pre-processing techniques must be evaluated and carefully explained in order to obtain reliable conclusions and reproducible results. To be sure of the potential improvement, one must provide the outcome of statistical tests and, if possible, the confidence interval for the median performance.

9 Conclusion

This manuscript presents the first comprehensive study on deep regression for computer vision. Motivated by a lack of systematic means of comparing two deep regression techniques and by the proliferation of deep architectures (also for regression in computer vision), we propose an experimental protocol to soundly compare different methods. In paticular, we are interested in vanilla deep regression methods –short for convolutional neural networks with a linear regression layer– and their variants (e.g. modifying the number of fine-tuned layers or the data pre-processing strategy). Each of them is run five times and 95% confidence interval for the median performance as well as statistical tests are reported. These measures are used to discern between variations due to stochastic effects from systematic improvements. The conducted extensive experimental evaluation allows us to extract some conclusions on what to

do, what not to do and what to test when employing vanilla deep regression methods for computer vision tasks.

In addition, we have shown that correctly fine-tuned deep regression networks compete with problem-specific methods in the literature that are entirely devoted to solve only one task. Importantly, the overall set of experiments clearly points out the need of a proper data-preprocessing strategy to obtain results that are meaningful and competitive with the state-of-the-art. This behavior should encourage the community to use very clean experimental protocols that include transparent and detailed descriptions of the different pre-processing strategies used. Ideally, papers should evaluate the impact of different pre-processing strategies when presenting a new dataset or a new method. We believe this would help the community understand deep regression for computer vision and to discern systematic behaviors from those due to chance.

REFERENCES

[1] G. Fanelli, J. Gall, and L. Van Gool, "Real time head pose estimation with random regression forests," in *CVPR*, 2011, pp. 617–624.

[2] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *CVPR*, 2012, pp. 2879–2886.

[3] X. P. Burgos-Artizzu, P. Perona, and P. Dollár, "Robust face landmark estimation under occlusion," in *ICCV*, 2013, pp. 1513–1520.

[4] M. Dantone, J. Gall, G. Fanelli, and L. Van Gool, "Real-time facial feature detection using conditional regression forests," in *CVPR*, 2012, pp. 2578–2585.

[5] A. Agarwal and B. Triggs, "3d human pose from silhouettes by relevance vector regression," in *CVPR*, vol. 2, 2004, pp. II–II.

[6] M. Sun, P. Kohli, and J. Shotton, "Conditional regression forests for human pose estimation," in *CVPR*, 2012, pp. 3394–3401.

[7] S. Yan, H. Wang, X. Tang, and T. S. Huang, "Learning auto-structured regressor from uncertain nonnegative labels," in *CVPR*, 2007, pp. 1–8.

[8] G. Guo, Y. Fu, C. R. Dyer, and T. S. Huang, "Image-based human age estimation by manifold learning and locally adjusted robust regression," *IEEE TIP*, vol. 17, no. 7, pp. 1178–1188, 2008.

[9] C.-R. Chou, B. Frederick, G. Mageras, S. Chang, and S. Pizer, "2d/3d image registration using regression learning," *CVIU*, vol. 117, no. 9, pp. 1095–1106, 2013.

[10] M. Niethammer, Y. Huang, and F.-X. Vialard, "Geodesic regression for image time-series," in *MICCAI*, 2011, pp. 655–662.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *NIPS*, 2012.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *CVPR*, 2014.

[14] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *ICLR*, 2014.

[15] X. Liu, W. Liang, Y. Wang, S. Li, and M. Pei, "3D head pose estimation with convolutional neural network trained on synthetic images," in *ICIP*, 2016, pp. 1289–1293.

[16] A. Toshev and C. Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks," in *CVPR*, 2014.

[17] V. Belagiannis, C. Rupprecht, G. Carneiro, and N. Navab, "Robust optimization for deep regression," in *ICCV*, 2015.

[18] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *CVPR*, 2013.

[19] Z. Liu, S. Yan, P. Luo, X. Wang, and X. Tang, "Fashion Landmark Detection in the Wild," in *ECCV*, 2016.

[20] S. Lathuilière, R. Juge, P. Mesejo, R. Muñoz Salinas, and R. Horaud, "Deep Mixture of Linear Inverse Regressions Applied to Head-Pose Estimation," in *CVPR*, 2017, pp. 7149–7157.

[21] A. Bulat and G. Tzimiropoulos, "How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks)," in *ICCV*, 2017.

[22] R. Rothe, R. Timofte, and L. Van Gool, "Deep expectation of real and apparent age from a single image without facial landmarks," *IJCV*, 2016.

[23] G. Rogez, P. Weinzaepfel, and C. Schmid, "Lcr-net: Localization-classification-regression for human pose," in *CVPR*, July 2017.

[24] Y. LeCun, L. Bottou, G. B. Orr, and K. Müller, "Effiicient backprop," in *Neural Networks: Tricks of the Trade*, 1998, pp. 9–50.

[25] C. Nebauer, "Evaluation of convolutional neural networks for visual recognition," *IEEE TNN*, vol. 9, no. 4, pp. 685–696, 1998.

[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *CVPR*, 2016, pp. 2818–2826.

[27] L. N. Smith and N. Topin, "Deep Convolutional Neural Network Design Patterns," *CoRR*, vol. abs/1611.00847, 2016.

[28] V. K. Ithapu, S. N. Ravi, and V. Singh, "On architectural choices in deep learning: From network structure to gradient convergence and parameter estimation," *CoRR*, vol. abs/1702.08670, 2017.

[29] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.

[30] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE TNNLS*, vol. 28, no. 10, pp. 2222–2232, 2017.

[31] V. Chandrasekhar, J. Lin, O. Morère, H. Goh, and A. Veillard, "A practical guide to CNNs and Fisher Vectors for image instance retrieval," *Signal Processing*, vol. 128, pp. 426–439, 2016.

[32] D. Mishkin, N. Sergievskiy, and J. Matas, "Systematic evaluation of convolution neural network advances on the imagenet," *CVIU*, vol. 161, pp. 11–19, 2017.

[33] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *BMVC*, 2014.

[34] S. Saxena and J. Verbeek, "Convolutional neural fabrics," in *NIPS*, 2016, pp. 4053–4061.

[35] L. Xie and A. L. Yuille, "Genetic CNN," in *CVPR*, 2017.

[36] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, pp. 80–83, 1945.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[39] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE TPAMI*, vol. 39, pp. 1137–1149, 2015.

[40] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Gool, "Random Forests for Real Time 3D Face Analysis," *IJCV*, vol. 101, no. 3, pp. 437–458, 2013.

[41] B. Wang, W. Liang, Y. Wang, and Y. Liang, "Head pose estimation with combined 2D SIFT and 3D HOG features," in *ICIG*, 2013.

[42] S. Mukherjee and N. Robertson, "Deep Head Pose: Gaze-Direction Estimation in Multimodal Video," *IEEE TMM*, vol. 17, no. 11, pp. 2094–2107, 2015.

[43] V. Drouard, R. Horaud, A. Deleforge, S. Ba, and G. Evangelidis, "Robust head-pose estimation based on partially-latent mixture of linear regressions," *IEEE TIP*, vol. 26, no. 3, pp. 1428–1440, 2017.

[44] D. Ramanan, "Learning to parse images of articulated bodies," in *NIPS*, 2007, pp. 1129–1136.

[45] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504 – 507, 2006.

[46] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why Does Unsupervised Pre-training Help Deep Learning?" *JMLR*, vol. 11, pp. 625–660, 2010.

[47] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How Transferable Are Features in Deep Neural Networks?" in *NIPS*, 2014.

[48] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?" *IEEE TMI*, vol. 35, no. 5, pp. 1299–1312, 2016.

[49] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, 2016.

[50] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[51] P. Mesejo, O. Ibánez, E. Fernández-Blanco, F. Cedrón, A. Pazos, and A. B. Porto-Pazos, "Artificial neuron–glia networks learning approach based on cooperative coevolution," *International journal of neural systems*, vol. 25, no. 04, 2015.

[52] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *JMLR*, vol. 12, no. 7, pp. 2121–2159, 2011.
[53] T. Tieleman and G. Hinton, "Rrmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.
[54] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.
[55] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[56] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *ICML*, 2013, pp. 1139–1147.
[57] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.
[58] R. Fisher, *Statistical methods for research workers*. Edinburgh Oliver & Boyd, 1925.
[59] R. Nuzzo, "Scientific method: Statistical errors," *Nature*, vol. 506, no. 7487, pp. 150–152, 2014.
[60] B. Vidgen and T. Yasseri, "P-values: Misunderstood and misused," *Frontiers in Physics*, vol. 4, p. 6, 2016.
[61] J. A. C. Sterne, D. R. Cox, and G. D. Smith, "Sifting the evidence—what's wrong with significance tests?Another comment on the role of statistical methods," *BMJ*, vol. 322, no. 7280, pp. 226–231, 2001.
[62] J. Derrac, S. Garca, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms." *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.
[63] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, pp. 52–64, 1961.
[64] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.
[65] Y. Hochberg, "A Sharper Bonferroni Procedure for Multiple Tests of Significance," *Biometrika*, vol. 75, no. 4, pp. 800–802, 1988.
[66] G. Hommel, "A stagewise rejective multiple test procedure based on a modified Bonferroni test," *Biometrika*, vol. 75, no. 2, pp. 383–386, 1988.
[67] B. Holland, "An improved sequentially rejective Bonferroni test procedure," *Biometrics*, vol. 43, pp. 417–423, 1987.
[68] D. Rom, "A sequentially rejective test procedure based on a modified Bonferroni inequality," *Biometrika*, vol. 77, pp. 663–665, 1990.
[69] P. Nemenyi, "Distribution-free multiple comparisons," Ph.D. dissertation, Princeton University, 1963.
[70] W. Conover, *Practical Nonparametric Statistics*. Kirjastus: John Wiley and Sons (WIE), 1998.
[71] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *JMLR*, vol. 15, pp. 1929–1958, 2014.
[72] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015, pp. 448–456.
[73] V. Drouard, R. Horaud, A. Deleforge, S. Ba, and G. Evangelidis, "Robust head-pose estimation based on partially-latent mixture of linear regressions," *IEEE TIP*, vol. 26, no. 3, pp. 1428 – 1440, 2017.
[74] M. Andriluka, S. Roth, and B. Schiele, "Pictorial structures revisited: People detection and articulated pose estimation," in *CVPR*, 2009, pp. 1014–1021.
[75] Y. Yang and D. Ramanan, "Articulated human detection with flexible mixtures of parts," *IEEE TPAMI*, vol. 35, no. 12, pp. 2878–2890, 2013.
[76] L. Pishchulin, A. Jain, M. Andriluka, T. Thormählen, and B. Schiele, "Articulated people detection and pose estimation: Reshaping the future," in *CVPR*. IEEE, 2012, pp. 3178–3185.
[77] S. Johnson and M. Everingham, "Clustered pose and nonlinear appearance models for human pose estimation," in *BMVC*, 2010.
[78] W. Ouyang, X. Chu, and X. Wang, "Multi-source deep learning for human pose estimation," in *CVPR*, 2014, pp. 2329–2336.

Stéphane Lathuilière received the M.Sc. degree in applied mathematics and computer science from ENSIMAG, Grenoble Institute of Technology (Grenoble INP), France, in 2014. Currently he is a PhD student in the PERCEPTION team at INRIA Grenoble Rhone-Alpes. His research interests include machine learning for activity recognition, deep models for regression and reinforcement learning for audio-visual fusion in robotics.

Pablo Mesejo received the M.Sc. and Ph.D. degrees in computer science respectively from Universidade da Coruña (Spain) and Università degli Studi di Parma (Italy), where he was an Early Stage Researcher within the Marie Curie ITN MIBISOC. He was a post-doc at the AL-CoV team of Université d'Auvergne (France) and the Mistis team of Inria Grenoble Rhône-Alpes (France), before joining the Perception team as a Starting Researcher Position. His research interests include computer vision, machine learning and computational intelligence techniques applied mainly to biomedical image analysis problems.

Xavier Alameda-Pineda received M.Sc. in Mathematics (2008), in Telecommunications (2009) and in Computer Science (2010). He obtained his Ph.D. in Mathematics and Computer Science from Université Joseph Fourier in 2013. Since 2016, he is a Research Scientist at Inria Grenoble, with the Perception Team. He served as Area Chair at ICCV 2017 and is the recipient of several paper awards. His scientific interests lie in computer vision, machine learning and signal processing for human behavior understanding and robotics.

Radu Horaud received the B.Sc. degree in Electrical Engineering, the M.Sc. degree in Control Engineering, and the Ph.D. degree in Computer Science from the Institut National Polytechnique de Grenoble, France. Since 1998 he holds a position of director of research with INRIA Grenoble Rhône-Alpes, where he is the founder and head of the PERCEPTION team. His research interests include computer vision, machine learning, audio signal processing, audiovisual analysis, and robotics. Radu Horaud and his collaborators received numerous best paper awards. In 2013, Radu Horaud was awarded an ERC Advanced Grant for his project *Vision and Hearing in Action* (VHIA) and in 2017 he was awarded an ERC Proof of Concept Grant for this project VHIALab.