A Comprehensive Analysis of Deep Regression
Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda, Radu Horaud

To cite this version:
Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda, Radu Horaud. A Comprehensive Analysis of Deep Regression. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020, 42 (9), pp.2065-2081. 10.1109/TPAMI.2019.2910523. hal-01754839

HAL Id: hal-01754839
https://inria.hal.science/hal-01754839
Submitted on 15 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
A Comprehensive Analysis of Deep Regression

Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda, Senior 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 from this trend. There is a large number of deep models, where small changes in the network architecture, or in the data pre-processing, together with the stochastic nature of the optimization procedures, produce notably different results, making extremely difficult to sift methods that significantly outperform others. This situation motivates the current study, in which we perform a systematic evaluation and statistical analysis of vanilla deep regression, i.e. convolutional neural networks with a linear regression top layer. This is the first comprehensive analysis of deep regression techniques. We perform experiments on four 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. Our results reinforce the hypothesis according to which, in general, a general-purpose network (e.g. VGG-16 or ResNet-50) adequately tuned can yield results close to the state-of-the-art without having to resort to more complex and ad-hoc regression models.


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], and have also been applied to newer tasks such as the analysis of image virality [15]. 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 [16], human pose estimation [17, 18] or facial landmark detection [19]. 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, [17], [18], [19] demonstrate the effectiveness of this strategy. Likewise for robust regression [20], often implying probabilistic models [21]. 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 [22], inverse piece-wise affine probabilistic model for head pose estimation [23] or a head-map estimator from which facial landmarks are extracted [24]. 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 [25, 26]. 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 systematic evaluation of deep architectures, focusing on different aspects. For instance, seminal papers exploring efficient back-propagation strategies [27], or evaluating ConvNets for visual recognition [28], were already published twenty years ago. Some articles provide general guidance and understanding on appropriate architectural choices [29], [30], [31] or on gradient-based training strategies for deep architectures [32], while others try to delve into the differences in performance of the variants of a specific (recurrent) model [33], or the differences in performance of several models applied to a specific problem [34], [35], [36]. Automatic ways to overcome the problem of choosing the optimal network architecture were also devised [37], [38]. 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, ≈ 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 [24], many published studies do not describe implementation, practical, or data preprocessing in detail [16], [18], [19], [22]. 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. [35] 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 [35] such that one understands the differences in performance from a thorough statistical perspective. [33] 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 box-plots 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 [29]. 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 preprocessing 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 [40] and ResNet-50 [41]. The VGG-16 model was termed Model D in [40]. We prefer VGG-16 to AlexNet [11] because it performs significantly better on ImageNet and has inspired several network architectures for various tasks [22], [42]. 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 $CB'$
denote the \(i^{th}\) convolution block (see [40] for the details on the number of layers and of units per layer, as well as the convolution parameters). Let \(F^i\) denote the \(i^{th}\) fully connected layer with a dropout rate of 50\%. \(F^l\) 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 \(CB^1 - CB^2 - CB^3 - CB^4 - CB^5 - F^l - FC^1 - FC^2 - SM\).

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 these three problems we selected four 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 [43] 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 [16, 23, 44, 45, 46]. During the recordings, the participants freely move their head and the corresponding head orientations lie in the intervals \([-60^\circ, 60^\circ]\) (pitch), \([-75^\circ, 75^\circ]\) (yaw), and \([-20^\circ, 20^\circ]\) (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 [23]. 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 [19] that consist of 5590 and 7876 face images, respectively. We combined both data sets and employed the same data partition as in [19]. 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.

Third, we use the Parse data set [47] 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 \([12^\circ, -12^\circ]\) with a step of 0.5\(^\circ\). This procedure is applied only to the training images.

Finally, we also use the MPII data set [48], that is a larger data set for human pose estimation. The data set contains around 25K images of over 40K people describing more than 400 human activities. This dataset represents a significant advance in terms of diversity and difficulty with respect to previous datasets for human body pose estimation. Importantly, the test set is not available for this dataset and score on the test set can be obtained only by submitting predictions to the MPII dataset research group. In addition, the number of submissions is limited. Therefore the MPII test set cannot be used in the context of this study. Therefore, we split the MPII training set into training, validation and test using 70\%, 20\% and 10\% of the original training set. Following MPII protocol, models are evaluated on the test set, considering only sufficiently separated individuals. Only mirroring is employed for data-augmentation in order to keep training time tractable in the context of our statistical analysis.

### 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 1000 experimental runs (\(\approx 110\) 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 \(\theta\) (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 [49], [50], [51], [52].

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 [53]. There are other methods to train neural networks, from derivative-free optimization [54], [55] to second-order methods [53], 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,

1. [https://team.inria.fr/perception/deep-regression/](https://team.inria.fr/perception/deep-regression/)
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 [53]. 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 [56] (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.

RMSProp [57] (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 [58] 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 [59] (Adaptive Moments) is an update to the RMSProp optimizer where momentum [60] 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 inertia).

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 [61]. 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 four data sets. We observe that, when employing VGG-16, the choice of the optimizer can lead to completely different loss values at convergence. Overall, we can state that the best training performance (in terms of loss value and convergence time) corresponds to AdaDelta and to Adam. In the light of the results described above, AdaGrad and RMSProp do not seem to be the optimizers of choice 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. Nevertheless, we observe that, with VGG-16, Adam is the best performing optimizer on FLD and one of the worst on MPII. Therefore, in case of optimization issues when tackling a new task, trying both Adam and AdaDelta seems to be a good option.

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, except on MPII, 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). With VGG-16 on MPII, note that the batch size has an impact on convergence speed but not on the final loss value. With ResNet-50, 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 (especially 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 refer to 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 $\alpha$ then the null hypothesis is rejected. Therefore, $\alpha$ 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 it. 

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 [66]. In our case, the visual inspection of the error measurements as well as the application of normality tests (in particular, the Lillifors 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 [67], Holm [68], Hochberg [69], Hommel [70], Holland [71], Rom [72], or Nemenyi [73]. Following the recommendation of Derrac et al. [66], 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 [39] 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
MSE
MSE
CB
ρ
-10
GAP
CB
FC
ρ
BN

= 9
applying the binomial distribution [74]. The number of
confidence interval for the
q
event that has occurred with a probability of 5% (or less) of
lies outside the 95% confidence interval, then a sampling
relates to the reliability of the estimation procedure, not to a
value of the population parameter. The 95% probability
interval from some future experiment encompasses the true
there is a 95% probability that the calculated confidence
interval covers the population parameter), but that
population parameter lies within it (i.e. a 95% probability
for a given realized interval there is 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 [74]. 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-fitting, we use holdout as model valida-
tion technique, 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 [19]). 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 [75]:
Loss (L) denotes the loss function employed at training
time. The baselines employ the Mean Squared Error (MSE).
Following [76], we compare the (MSE) with the Mean
Absolute Error (MAE) and the Huber loss (HUB).
Batch Normalization (BN) was introduced to lead to fast
and reliable network convergence [77]. 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
BN), or before the activation of FC^2 (denoted by BNB), or
we do not use batch normalization BN. In ResNet-50 we
consider only BN and BNB, 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 BNB, but not ResNet-50. This is due to the fact
that the original VGG-16 does not exploit batch normaliza-
tion, while ResNet-50 does. Using 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 BN (and
this is why the ResNet-50 baselines do not use BN), but
we include this comparison for completeness. Finally, we
compare the use of batch normalization with the more recent layer normalization \cite{torch} denoted by LN.

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

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

**Fine-tuning depth.** FT denotes the deepest block fixed during training, i.e. only the last layers are modified. For both architectures, we compare \( CB^2, CB^3, CB^4, CB^5 \). Note that the regression layer is always trained from scratch.

**Regressed layer.** RL denotes the layer after which the \( BN \) and \( REG \) layers are added: that is the layer that is regressed. \( \rho(L) \) denotes the model where the regression is performed on the output activations of the layer \( RL \), meaning that on top of that layer we directly add batch normalization and linear regression layers. For VGG-16, we compare \( \rho(CB^3), \rho(CF^1) \) and \( \rho(CF^2) \). For ResNet-50 we compare \( \rho(CB^3) \) and \( \rho(GAP) \).

**Target & input representation (TIR).** The target representation can be either a heatmap or a low-dimensional vector. While in the first case, the input representation needs to keep the spatial correspondence with the input image, in the second case one can choose an input representation that does not maintain a spatial structure. In our study, we evaluate the performance obtained when using heatmap regression (HM) and when the target is a low-dimensional vector and the input is either the output of a global average pooling (GAP), global max pooling (GMP) or, in the case of VGG-16, the second fully connected layer (FC2). In the case of HM, we use architectures directly inspired by \cite{dai2015}. In particular, we add a \( 1 \times 1 \) convolution layer with batch normalization and ReLU activations, followed by a \( 1 \times 1 \) convolution with linear activations to \( CB^3 \) and \( CB^2 \) for VGG-16 and ResNet-50, respectively, predicting a heatmap of dimensions \( 56 \times 56 \) and \( 55 \times 55 \), respectively. Following common practice in heatmap regression, the ground-truth heatmaps are generated according to a Gaussian-like function \cite{dai2015, li2017}, and training is performed employing the MSE loss. Importantly, for the experiments with heatmap regression, the FT configuration needs to be specifically adapted. Indeed, the baseline fine-tuning would lead to no fine-tuning, since \( CB^3 \) and \( CB^4 \) are not used. Consequently, we employ the \( CB^2 \) fine-tuning configuration for both VGG-16 and ResNet-50 networks. At test time, the landmark locations are obtained as the arguments of the maxima over the predicted heatmaps.

The settings corresponding to our baselines are detailed in Table \ref{table:loss}. These baselines are chosen regarding common choices in the state-of-the-art (for \( L \), BN and DO) and in the light of preliminary experiments (not reported, for FR, LR and TIR). No important changes with respect to the original design were adopted. 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 Loss

Table \ref{table:loss} presents the results obtained when changing the training loss of the two base architectures and the four datasets. The results of Table \ref{table:loss} are quite uninformative, since the choice of the loss to be used highly depends on the dataset, and varies also with the base architecture. Indeed, while for MPII the best and worst choices are MAE and MSE respectively, MSE could be the best or worst for Biwi, depending on the architecture, and the Huber loss would be the optimal choice for FLD. All three loss functions are statistically equivalent for Parse, while for the other datasets significant differences are found. Even if the standard choice in the state-of-the-art is the \( L_2 \) loss, we strongly recommend to try all three losses when exploring a new dataset.

### 5.2 Batch Normalization

Table \ref{table:BN} shows the results obtained employing several normalization strategies. In the case of VGG-16, we observe that the impact of employing layer normalization (i.e. LN) or adding a batch normalization layer after the activations (i.e. BN), specially the former, is significant and beneficial compared to the other two alternatives. In the case of FLD, we notice that the problem with \( BN \) and \( BNB \) occurs at training since the final training MSE score is much higher than the one obtained with \( BN \). Interestingly, on the Biwi data set we observe that the training MSE is better with \( BN \) but \( BN \) performs better on the validation and the test sets.
We conclude that in the case of Biwi, BN does not help the optimization, but increases the generalization ability. The highly beneficial impact of normalization observed in VGG-16 clearly justifies its use. In particular, LN has empirically shown a better performance than BN, so its use is strongly recommended. However, this paper considers BN as a baseline, both for the positive results it provides and for being a more popular and consolidated normalization technique within the scientific community. The exact opposite trend is observed when running the same experiments with ResNet-50. In the case of VGG-16, we observe a more popular and consolidated normalization technique line, both for the positive results it provides and for being the safest option (best for Biwi/FLD, equivalent for Parse).

Table 4 shows results obtained with various fine-tuning depth values, as described in Section 5, for both VGG-16 and ResNet-50. The vertical gray line is the default configuration.

5.3 Dropout Ratio

We conclude that in the case of Biwi, BN does not help the optimization, but increases the generalization ability. The highly beneficial impact of normalization observed in VGG-16 clearly justifies its use. In particular, LN has empirically shown a better performance than BN, so its use is strongly recommended. However, this paper considers BN as a baseline, both for the positive results it provides and for being a more popular and consolidated normalization technique within the scientific community. The exact opposite trend is observed when running the same experiments with ResNet-50. Indeed, 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).

6.4 Fine Tuning Depth

Table 5 shows 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 a behavior of CB4 similar to the one observed for batch normalization. Indeed, CB4 may not be the best choice in terms of optimization (CB2 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 data sets but MPII, where CB3 shows a statistically significant better performance. In addition, the results shown in Table 5 also discourage to use CB2. 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 CB3), ResNet-50 on Biwi selects the model CB4 by a solid margin in a statistically significant manner, and the same happens with MPII where CB4 and CB2 represent better choices. Therefore, we suggest that, 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 CB2 should not necessarily be included, since for all data sets this option is significantly worse than the other ones.

5.5 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, MPII and Parse point to use GAP, while results on FLD point to use CB5. However, we can observe that the confidence intervals of ρ(GAP) and ρ(CB5) 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.

5.6 Target and Input Representations

Table 8 shows the results obtained for different target and input representations on VGG-16 and ResNet-50. Regarding VGG-16, we observed that except for FLD, the best option is to keep the flatten layer, and therefore to discard the use of pooling layers (GMP or GAP). The case of FLD is different, since the results would suggest that it is significantly better to use either GAP or GMP. Similar conclusions can be drawn from the results obtained with ResNet-50 in the sense that the standard configuration (GAP) is the optimal one for Biwi and Parse, but not for FLD. In the case of FLD, the results suggest to choose GMP. As indicated above, since heatmap regression represents, in general, the state of the art in human-pose estimation and landmark localization, we also tested HM on FLD, Parse and MPII. The first conclusion that can be drawn from Table 8 is that HM produces worse results than the other TIR. A more detailed inspection of Table 8 reveals an overall optimization problem: HM leads to less good local minima. Importantly, this occurs under a wide variety of optimization settings (tested but not reported). We also realized that, current deep regression strategies exploiting heatmaps do not exploit vanilla regressors, but ad-hoc, complex and cascaded struc-
TABLE 8: Impact of the target and input representations (TIR) on VGG-16 and ResNet-50.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>TIR</th>
<th>VGG-16</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MAE test</td>
<td>MSE train</td>
<td>MSE valid</td>
<td>MSE test</td>
<td>MAE test</td>
<td>MSE train</td>
<td>MSE valid</td>
<td>MSE test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Biwi</td>
<td>GMP</td>
<td>[3.97 4.08]</td>
<td>[4.03 4.11]</td>
<td>[18.25 18.99]</td>
<td>[21.19 22.38]</td>
<td>[3.64 3.75]</td>
<td>[1.48 1.50]</td>
<td>[20.62 21.23]</td>
<td>[17.80 18.88]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GAP</td>
<td>[3.99 4.09]</td>
<td>[2.75 2.80]</td>
<td>[20.71 21.40]</td>
<td>[21.03 22.07]</td>
<td>[3.60 3.71]</td>
<td>[1.25 1.27]</td>
<td>[21.49 22.25]</td>
<td>[17.15 18.14]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FC²</td>
<td>[3.66 3.79]</td>
<td>[4.33 4.41]</td>
<td>[12.18 12.56]</td>
<td>[18.77 20.20]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLDS</td>
<td>GMP</td>
<td>[2.50 2.64]</td>
<td>[3.02 3.06]</td>
<td>[7.65 7.81]</td>
<td>[9.56 11.07]</td>
<td>[1.75 1.82]</td>
<td>[2.17 2.19]</td>
<td>[5.21 5.30]</td>
<td>[4.67 5.22]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GAP</td>
<td>[2.53 2.67]</td>
<td>[5.18 5.23]</td>
<td>[9.06 9.22]</td>
<td>[9.69 10.64]</td>
<td>[1.96 2.05]</td>
<td>[5.21 5.25]</td>
<td>[6.85 6.95]</td>
<td>[5.79 6.39]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FC²</td>
<td>[2.61 2.76]</td>
<td>[9.43 9.54]</td>
<td>[10.77 11.03]</td>
<td>[10.35 11.62]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HM</td>
<td>[3.08 3.22]</td>
<td>[17.94 18.16]</td>
<td>[18.08 18.46]</td>
<td>[14.74 16.60]</td>
<td>[2.25 2.35]</td>
<td>[8.37 8.46]</td>
<td>[8.99 9.17]</td>
<td>[7.65 8.39]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parse</td>
<td>GMP</td>
<td>[5.55 6.06]</td>
<td>[1.17 1.18]</td>
<td>[30.00 30.67]</td>
<td>[52.17 61.86]</td>
<td>[5.74 6.51]</td>
<td>[0.89 0.90]</td>
<td>[36.39 37.17]</td>
<td>[54.63 69.10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GAP</td>
<td>[5.61 6.07]</td>
<td>[1.48 1.50]</td>
<td>[31.46 32.15]</td>
<td>[51.33 60.57]</td>
<td>[4.56 5.68]</td>
<td>[1.04 1.04]</td>
<td>[29.30 30.09]</td>
<td>[43.58 55.21]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FC²</td>
<td>[6.00 5.59]</td>
<td>[2.38 2.41]</td>
<td>[27.74 28.60]</td>
<td>[41.09 52.15]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HM</td>
<td>[19.06 21.19]</td>
<td>[455.8 460.3]</td>
<td>[475.4 485.8]</td>
<td>[674.3 806.1]</td>
<td>[18.90 21.32]</td>
<td>[358.6 363.1]</td>
<td>[458.1 469.7]</td>
<td>[631.8 841.5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPII</td>
<td>GMP</td>
<td>[9.55 9.90]</td>
<td>[73.96 74.72]</td>
<td>[215.4 221.0]</td>
<td>[166.6 179.4]</td>
<td>[8.71 9.06]</td>
<td>[114.2 115.5]</td>
<td>[191.2 196.4]</td>
<td>[142.1 154.5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GAP</td>
<td>[9.60 9.98]</td>
<td>[63.56 64.21]</td>
<td>[212.0 217.0]</td>
<td>[163.5 178.7]</td>
<td>[7.27 8.06]</td>
<td>[23.35 23.56]</td>
<td>[147.9 152.1]</td>
<td>[110.9 119.0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FC²</td>
<td>[8.42 8.78]</td>
<td>[73.96 74.62]</td>
<td>[165.3 170.4]</td>
<td>[130.7 142.3]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HM</td>
<td>[27.50 28.28]</td>
<td>[1489 1500]</td>
<td>[1419 1443]</td>
<td>[1384 1448]</td>
<td>[17.28 18.28]</td>
<td>[885.0 896.4]</td>
<td>[993.1 1016]</td>
<td>[773.2 859.0]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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, we vary the 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 [22], [86], where the authors crop a 64 × 64 and a 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 these pre-processing steps are shown in Figures 4a-4h.

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. [19] 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 $\varepsilon$% in each direction. We compare four different expanding ratios: 0%, 5%, 15%, and 50%. These are denoted with $\varepsilon$-0, $\varepsilon$-5, $\varepsilon$-15, and $\varepsilon$-50 respectively, see Figures 4i to 4m.

Parse. When using this data set in [18], the images were resized to $224 \times 224$ 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 $224 \times 224$ pixels. This strategy is referred to as $120 \times 80$. We also consider directly resizing into $224 \times 224$ images, hence without keeping the aspect ratio, and padding with the mean value of ImageNet ($\alpha$-Pad), or keeping the aspect ratio ($\alpha$-Pad). Examples are shown in Figures 4n to 4q. The two last strategies are also employed using zero-padding ($\alpha$-0Pad and $\alpha$-0Pad).

MPII. In the case of the MPII data set, similarly to the Biwi and the FLD data sets, the question of the amount of context information is examined. Similarly to the FLD dataset, we adopt the following procedure: the bounding box is expanded by $\varepsilon$% on each direction. We compare three different expanding ratios: 0%, 5%, 15% denoted respectively by $\varepsilon$-0, $\varepsilon$-5, and $\varepsilon$-15. Examples of these pre-processing steps are shown in Figures 4f to 4k. Note that, contrary to the FLD dataset, we do not evaluate larger expanding ratios, since the MPII dataset contains many images where several persons would appear within the bounding box if a larger expanding ratio were employed.

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 4 a 5-pixel error for the right elbow location may be acceptable in the case of $\alpha$-0Pad but may correspond to a confusion with a shoulder location in the case of $\alpha$-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 $120 \times 80$. 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 3 to 9 are not directly comparable with those of Table 10. In the case of FLD, $\varepsilon$-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 $\varepsilon$-5 to allow direct comparison with Tables 3 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 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 $\varepsilon$-0 strategy is significantly better and the $\varepsilon$-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, $\alpha$-1Pad and $\alpha$-0Pad with VGG-16 are better than the rest, but without statistical differences between them. A similar behavior is found in ResNet-50 including $120 \times 80$.

The last three rows of Table 10 are devoted to the data pre-processing of MPII. The conclusion is coherent among the two base architectures: the strategy $\varepsilon$-5 is significantly better and the strategy $\varepsilon$-15 is significantly worse. We hy-

<table>
<thead>
<tr>
<th>Data Set &amp; Mrr.</th>
<th>VGG-16</th>
<th>ResNet-50</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE test</td>
<td>MSE train</td>
</tr>
<tr>
<td>Biwi Yes</td>
<td>[3.66 3.79]**</td>
<td>[4.43 4.41]</td>
</tr>
<tr>
<td></td>
<td>[5.49 5.67]</td>
<td>[9.09 9.36]</td>
</tr>
<tr>
<td>Biwi No</td>
<td>[2.61 2.76]**</td>
<td>[4.93 5.94]</td>
</tr>
<tr>
<td></td>
<td>[3.06 3.24]</td>
<td>[14.13 14.38]</td>
</tr>
<tr>
<td>FLD Yes</td>
<td>[4.90 5.59]**</td>
<td>[2.36 2.41]</td>
</tr>
<tr>
<td></td>
<td>[5.08 5.76]</td>
<td>[2.31 2.36]</td>
</tr>
<tr>
<td>FLD No</td>
<td>[8.42 8.78]**</td>
<td>[73.86 74.78]</td>
</tr>
<tr>
<td></td>
<td>[9.40 9.74]</td>
<td>[95.32 96.72]</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Pre-processing</th>
<th>VGG-16</th>
<th>ResNet-50</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE test</td>
<td>MSE train</td>
</tr>
<tr>
<td>FLD c-0</td>
<td>[2.25 2.36]**</td>
<td>[7.61 7.69]</td>
</tr>
<tr>
<td>FLD c-5</td>
<td>[8.61 8.76]</td>
<td>[9.45 9.54]</td>
</tr>
<tr>
<td>FLD c-15</td>
<td>[2.35 2.48]</td>
<td>[8.42 8.50]</td>
</tr>
<tr>
<td>FLD c-50</td>
<td>[9.96 10.20]</td>
<td></td>
</tr>
<tr>
<td>FLD α-0P</td>
<td>[9.99 11.52]**</td>
<td>[8.47 8.62]</td>
</tr>
<tr>
<td>FLD α-5P</td>
<td>[9.72 11.13]</td>
<td>[9.77 9.95]</td>
</tr>
<tr>
<td>FLD α-15P</td>
<td>[8.54 9.56]**</td>
<td>[6.22 6.35]</td>
</tr>
<tr>
<td>FLD 120x80</td>
<td>[8.39 9.33]**</td>
<td>[8.23 8.39]</td>
</tr>
<tr>
<td>MPPI Yes</td>
<td>[8.42 8.78]</td>
<td>[73.96 74.72]</td>
</tr>
<tr>
<td></td>
<td>[8.15 8.44]</td>
<td>[51.32 52.29]</td>
</tr>
<tr>
<td></td>
<td>[10.43 10.83]**</td>
<td>[154.8 136.8]</td>
</tr>
</tbody>
</table>

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 [16] for details). We report the median and best run of the worst and best data pre-processing strategies for each baseline network.

<table>
<thead>
<tr>
<th>Pitch</th>
<th>Yaw</th>
<th>Roll</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-16 64-0Pad 224</td>
<td>10.11/4.75</td>
<td>4.70/3.92</td>
</tr>
<tr>
<td>ResNet-50 224</td>
<td>4.73/4.15</td>
<td>2.98/2.49</td>
</tr>
<tr>
<td>VGG-16 224</td>
<td>4.51/4.02</td>
<td>4.68/3.74</td>
</tr>
<tr>
<td>ResNet-50 64-Re</td>
<td>5.98/5.22</td>
<td>2.39/2.37</td>
</tr>
<tr>
<td>Liu et al. [16]</td>
<td>6.1</td>
<td>6.0</td>
</tr>
<tr>
<td>Liu et al. [16]†</td>
<td>4.5</td>
<td>4.3</td>
</tr>
<tr>
<td>Mukherjee et al. [43]</td>
<td>5.18</td>
<td>5.67</td>
</tr>
<tr>
<td>Drouard et al. [86]†</td>
<td>5.43</td>
<td>4.24</td>
</tr>
<tr>
<td>Lathuiery et al. [25]</td>
<td>4.68</td>
<td>3.12</td>
</tr>
</tbody>
</table>

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.

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.
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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Left Eye</th>
<th>Right Eye</th>
<th>Nose</th>
<th>Left Mouth</th>
<th>Right Mouth</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>10.44/4.02</td>
<td>6.83/4.02</td>
<td>5.22/7.63</td>
<td>6.43/6.02</td>
<td>6.83/5.22</td>
<td>7.15/5.38</td>
</tr>
<tr>
<td>VGG-16</td>
<td>1.20/0.80</td>
<td>0.40/0.00</td>
<td>3.21/2.41</td>
<td>3.61/2.81</td>
<td>3.61/2.41</td>
<td>2.41/1.69</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>0.80/1.20</td>
<td>0.00/0.00</td>
<td>0.00/0.40</td>
<td>2.01/0.80</td>
<td>1.20/1.20</td>
<td>0.80/0.72</td>
</tr>
<tr>
<td>Sun et al.</td>
<td>0.67</td>
<td>0.33</td>
<td>0.00</td>
<td>1.16</td>
<td>0.67</td>
<td>0.57</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Head</th>
<th>Torso</th>
<th>Upper Legs</th>
<th>Lower Legs</th>
<th>Upper Arms</th>
<th>Lower Arms</th>
<th>Full Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-16</td>
<td>68.5/70.49</td>
<td>85.25/83.61</td>
<td>77.05/79.51</td>
<td>63.11/63.93</td>
<td>45.90/45.08</td>
<td>40.16/42.62</td>
<td>60.66/61.64</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>57.4/65.66</td>
<td>68.9/73.8</td>
<td>71.3/74.6</td>
<td>55.7/62.3</td>
<td>39.3/45.1</td>
<td>36.1/41.0</td>
<td>53.1/58.5</td>
</tr>
<tr>
<td>VGG-16</td>
<td>77.05/68.85</td>
<td>90.16/88.52</td>
<td>80.33/81.15</td>
<td>63.11/67.21</td>
<td>50.82/52.46</td>
<td>44.26/50.00</td>
<td>64.43/65.90</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>67.2/68.9</td>
<td>78.7/80.3</td>
<td>77.9/77.9</td>
<td>65.6/63.9</td>
<td>45.1/50.0</td>
<td>46.7/45.9</td>
<td>61.6/62.1</td>
</tr>
<tr>
<td>Andriluka et al.</td>
<td>72.7</td>
<td>86.3</td>
<td>66.3</td>
<td>60.0</td>
<td>54.0</td>
<td>35.6</td>
<td>59.2</td>
</tr>
<tr>
<td>Yang &amp; Ramanan</td>
<td>82.4</td>
<td>82.9</td>
<td>68.8</td>
<td>60.5</td>
<td>63.4</td>
<td>42.4</td>
<td>63.6</td>
</tr>
<tr>
<td>Pishchulin et al.</td>
<td>77.6</td>
<td>90.7</td>
<td>80.0</td>
<td>70.0</td>
<td>59.3</td>
<td>37.1</td>
<td>66.1</td>
</tr>
<tr>
<td>Johnson et al.</td>
<td>76.8</td>
<td>87.6</td>
<td>74.7</td>
<td>67.1</td>
<td>67.3</td>
<td>45.8</td>
<td>67.4</td>
</tr>
<tr>
<td>OuYang et al.</td>
<td>89.3</td>
<td>89.3</td>
<td>78.0</td>
<td>72.0</td>
<td>67.8</td>
<td>47.8</td>
<td>71.0</td>
</tr>
<tr>
<td>Belagiannis et al.</td>
<td>91.7</td>
<td>98.1</td>
<td>84.2</td>
<td>79.3</td>
<td>66.1</td>
<td>41.5</td>
<td>73.2</td>
</tr>
</tbody>
</table>

TABLE 14: Comparison on the MPII single person body-pose data set. PCKh scores are reported. We report the median and best run of the baseline networks.

<table>
<thead>
<tr>
<th>Method</th>
<th>Total PCKh</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-16</td>
<td>61.6/62.5</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>61.1/67.5</td>
</tr>
<tr>
<td>Tompson et al.</td>
<td>83</td>
</tr>
<tr>
<td>Belagiannis et al.</td>
<td>79</td>
</tr>
<tr>
<td>Newell et al.</td>
<td>81</td>
</tr>
<tr>
<td>Chu et al.</td>
<td>85</td>
</tr>
<tr>
<td>Chen et al.</td>
<td>81</td>
</tr>
<tr>
<td>Yang et al.</td>
<td>72</td>
</tr>
<tr>
<td>Nie et al.</td>
<td>82</td>
</tr>
</tbody>
</table>

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 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 correctly 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 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.
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.

Finally, Table [14] reports the results on the MPII the test set using the PCKh score. PCKh score is a variant of the PCP score employed for the Parse dataset, where the success threshold is set to 50% of the ground-truth head height. On this dataset, we observe that the state-of-the-art clearly outperforms vanilla deep regression, suggesting that, in some cases, adopting more complex and task-specific approaches can lead to better performance. In particular, in the case of full-body pose estimation, in opposition to our vanilla deep regression models, state of the art models exploit several assumptions specific to the human body structure. For instance, instead of regressing only the joints coordinates, recent approaches jointly estimate the limbs locations in order to better constrain the model [79], [82]. A cascade regression strategy is also often employ to enforce consistency among predicted joint locations [79], [81], [83], [84], [85]. Furthermore, at training time, state of the art methods usually exploit additional landmark visibility annotation [79], [84] that are not used by our general-purpose model. As conclusion, much better results are obtained, but at the cost of introducing more complex and ad-hoc approaches.

8 Overall Discussion

Figure [5] 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 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 [3], CB when fine-tuning ResNet-50 (see Table [5] and heatmap regression (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 MPII, and up to 6% success rate in Parse). Although, in Figure 5, 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] and [14] for Biwi, FLD and Parse, respectively, and the second best performing method in the corresponding original papers [23], [18], and [19]. Note that we do not display the horizontal line in the case of the MPII.

The reason for this choice is twofold. First, the scores obtained are significantly lower than those obtained by state-of-the-art methods. Consequently, the magnitude among variants of vanilla deep regression models (around 65% PCKh score) cannot be fairly compared with the variation of the state of the art (around 85% PCKh score). Second, because of the MPII evaluation protocol constraints explained in Sec. 2, we report the score on different training and test sets. 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 5 (with the exceptions of TIR in Biwi, data-preprocessing in Parse, and the loss in MPII). 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 TIR 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 TIR 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 particular, 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 intervals 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, in three out of four problems, correctly fine-tuned deep regression networks can 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 to understand deep regression for computer vision and to discern systematic behaviors from those due to chance.

References

Stéphane Lathuilière received the M.Sc. degree in applied mathematics and computer science from ENSIMAG, Grenoble Institute of Technology (Grenoble INP), France, in 2014. He worked towards his Ph.D. in mathematics and computer science in the Perception Team at Inria, and obtained it from Université Grenoble Alpes (France) in 2018. He is currently a Post-Doctoral fellow at the University of Trento. 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 with 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.