DeepGUM: Learning Deep Robust Regression with a Gaussian-Uniform Mixture Model

Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda, Radu Horaud

DeepGUM: Learning Deep Robust Regression with a Gaussian-Uniform Mixture Model

Stéphane Lathuilière[1,3], Pablo Mesejo[1,2], Xavier Alameda-Pineda[1], and Radu Horaud[1]

[1] Inria Grenoble Rhône-Alpes, Montbonnot-Saint-Martin, France,
[2] University of Granada, Granada, Spain,
[3] University of Trento, Trento, Italy
firstname.name@inria.fr

Abstract. In this paper we address the problem of how to robustly train a ConvNet for regression, or deep robust regression. Traditionally, deep regression employ the L_2 loss function, known to be sensitive to outliers, i.e. samples that either lie at an abnormal distance away from the majority of the training samples, or that correspond to wrongly annotated targets. This means that, during back-propagation, outliers may bias the training process due to the high magnitude of their gradient. In this paper, we propose DeepGUM: a deep regression model that is robust to outliers thanks to the use of a Gaussian-uniform mixture model. We derive an optimization algorithm that alternates between the unsupervised detection of outliers using expectation-maximization, and the supervised training with *cleaned* samples using stochastic gradient descent. DeepGUM is able to adapt to a continuously evolving outlier distribution, avoiding to manually impose any threshold on the proportion of outliers in the training set. Extensive experimental evaluations on four different tasks (facial and fashion landmark detection, age and head pose estimation) lead us to conclude that our novel robust technique provides reliability in the presence of various types of noise and protection against a high percentage of outliers.

Keywords: Robust regression · Deep Neural Networks · Mixture model · Outlier detection

1 Introduction

For the last decade, deep learning architectures have undoubtably established the state of the art in computer vision tasks such as image classification [17, 36] or object detection [14, 32]. These architectures, e.g. ConvNets, consist of several convolutional layers, followed by a few fully connected layers and by a classification softmax layer with, for instance, a cross-entropy loss. ConvNets have also been used for regression, i.e. predict continuous as opposed to categorical output values. Classical regression-based computer vision methods have addressed human pose estimation [37], age estimation [29], head-pose estimation [8], or facial landmark detection [35], to cite a few. Whenever

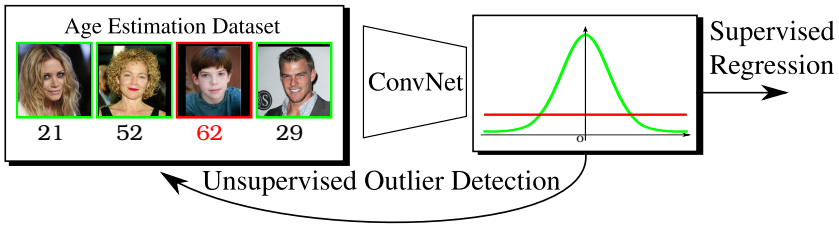


Fig. 1: A Gaussian-uniform mixture model is combined with a ConvNet architecture to downgrade the influence of wrongly annotated targets (outliers) on the learning process.

ConvNets are used for learning a regression network, the softmax layer is replaced with a fully connected layer, with linear or sigmoid activations, and L_2 is often used to measure the discrepancy between prediction and target variables. It is well known that L_2-loss is strongly sensitive to outliers, potentially leading to poor generalization performance [16]. While robust regression is extremely well investigated in statistics, there has only been a handful of methods that combine robust regression with deep architectures.

This paper proposes to mitigate the influence of outliers when deep neural architectures are used to learn a regression function, ConvNets in particular. More precisely, we investigate a methodology specifically designed to cope with two types of outliers that are often encountered: (i) samples that lie at an abnormal distance away from the other training samples, and (ii) wrongly annotated training samples. On the one hand, abnormal samples are present in almost any measurement system and they are known to bias the regression parameters. On the other hand, deep learning requires very large amounts of data and the annotation process, be it either automatic or manual, is inherently prone to errors. These unavoidable issues fully justify the development of robust deep regression.

The proposed method combines the representation power of ConvNets with the principled probabilistic mixture framework for outlier detection and rejection, e.g. Figure 1. We propose to use a Gaussian-uniform mixture (GUM) as the last layer of a ConvNet, and we refer to this combination as DeepGUM.[4] The mixture model hypothesizes a Gaussian distribution for inliers and a uniform distribution for outliers. We interleave an EM procedure within stochastic gradient descent (SGD) to downgrade the influence of outliers in order to robustly estimate the network parameters. We empirically validate the effectiveness of the proposed method with four computer vision problems and associated datasets: facial and fashion landmark detection, age estimation, and head pose estimation. The standard regression measures are accompanied by statistical tests that discern between random differences and systematic improvements.

The remainder of the paper is organized as follows. Section 2 describes the related work. Section 3 describes in detail the proposed method and the associated algorithm. Section 4 describes extensive experiments with several applications and associ-

[4] The code will be publicly available in case of manuscript acceptance.

ated datasets. Section 5 draws conclusions and discusses the potential of robust deep regression in computer vision.

2 Related Work

Robust regression has long been studied in statistics [16, 23, 30] and in computer vision [5, 24, 34]. Robust regression methods have a high *breakdown point*, which is the smallest amount of outlier contamination that an estimator can handle before yielding poor results. Prominent examples are the least trimmed squares, the Theil-Sen estimator or heavy-tailed distributions [13]. Several robust training strategies for artificial neural networks are also available [4, 26].

M-estimators, sampling methods, trimming methods and robust clustering are among the most used robust statistical methods. M-estimators [16] minimize the sum of a positive-definite function of the residuals and attempt to reduce the influence of large residual values. The minimization is carried our with weighted least squares techniques, with no proof of convergence for most M-estimators. Sampling methods [24], such as least-median-of-squares or random sample consensus (RANSAC), estimate the model parameters by solving a system of equations defined for a randomly chosen data subset. The main drawback of sampling methods is that they require complex data-sampling procedures and it is tedious to use them for estimating a large number of parameters. Trimming methods [30] rank the residuals and down-weight the data points associated with large residuals. They are typically cast into a (non-linear) weighted least squares optimization problem, where the weights are modified at each iteration, leading to iteratively re-weighted least squares problems. Robust statistics have also been addressed in the framework of mixture models and a number of robust mixture models were proposed, such as Gaussian mixtures with a uniform noise component [1, 7], heavy-tailed distributions [10], trimmed likelihood estimators [11, 27], or weighted-data mixtures [12]. Importantly, it has been recently reported that modeling outliers with an uniform component yields very good performance [7, 12].

Deep robust classification was recently addressed, e.g. [2] assumes that observed labels are generated from true labels with unknown noise parameters: a probabilistic model that maps true labels onto observed labels is proposed and an EM algorithm is derived. In [39] is proposed a probabilistic model that exploits the relationships between classes, images and noisy labels for large-scale image classification. This framework requires a dataset with explicit clean- and noisy-label annotations as well as an additional dataset annotated with a noise type for each sample, thus making the method difficult to use in practice. Classification algorithms based on a distillation process to learn from noisy data was recently proposed [20].

Recently, deep regression methods were proposed, e.g. [25, 28, 35, 37, 18]. Despite the vast robust statistics literature and the importance of regression in computer vision, at the best of our knowledge there has been only one attempt to combine robust regression with deep networks [3], where robustness is achieved by minimizing the Tukey's bi-weight loss function, i.e. an M-estimator. In this paper we take a radical different

approach and propose to use robust mixture modeling within a ConvNet. We conjecture that while *inlier noise* follows a Gaussian distribution, *outlier errors* are uniformly distributed over the volume occupied by the data. Mixture modeling provides a principled way to characterize data points individually, based on posterior probabilities. We propose an algorithm that interleaves a robust mixture model with network training, i.e. alternates between EM and SGD. EM evaluates data-posterior probabilities which are then used to weight the residuals used by the network loss function and hence to downgrade the influence of samples drawn from the uniform distribution. Then, the network parameters are updated which in turn are used by EM. A prominent feature of the algorithm is that it requires neither annotated outlier samples nor prior information about their percentage in the data. This is in contrast with [39] that requires explicit inlier/outlier annotations and with [3] which uses a fixed hyperparameter ($c = 4.6851$) that allows to exclude from SGD samples with high residuals.

3 Deep Regression with a Robust Mixture Model

We assume that the inlier noise follows a Gaussian distribution while the outlier error follows a uniform distribution. Let $\boldsymbol{x} \in \mathbb{R}^M$ and $\boldsymbol{y} \in \mathbb{R}^D$ be the input image and the output vector with dimensions M and D, respectively, with $D \ll M$. Let $\boldsymbol{\phi}$ denote a ConvNet with parameters $\boldsymbol{w}$ such that $\boldsymbol{y} = \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{w})$. We aim to train a model that detects outliers and downgrades their role in the prediction of a network output, while there is no prior information about the percentage and spread of outliers. The probability of $\boldsymbol{y}$ conditioned by $\boldsymbol{x}$ follows a Gaussian-uniform mixture model (GUM):

$$p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \pi \, \mathcal{N}(\boldsymbol{y}; \boldsymbol{\phi}(\boldsymbol{x}; \boldsymbol{w}), \boldsymbol{\Sigma}) + (1 - \pi) \, \mathcal{U}(\boldsymbol{y}; \gamma), \tag{1}$$

where π is the prior probability of an inlier sample, γ is the normalization parameter of the uniform distribution and $\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}$ is the covariance matrix of the multivariate Gaussian distribution. Let $\boldsymbol{\theta} = \{\pi, \gamma, \boldsymbol{\Sigma}\}$ be the parameter set of GUM. At training we estimate the parameters of the mixture model, $\boldsymbol{\theta}$, and of the network, $\boldsymbol{w}$. An EM algorithm is used to estimate the former together with the responsibilities r_n, which are plugged into the network's loss, minimized using SGD so as to estimate the later.

3.1 EM Algorithm

Let a training dataset consist of N image-vector pairs $\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N$. At each iteration, EM alternates between evaluating the expected complete-data log-likelihood (E-step) and updating the parameter set $\boldsymbol{\theta}$ conditioned by the network parameters (M-step). In practice, the E-step evaluates the posterior probability (responsibility) of an image-vector pair n to be an inlier:

$$r_n(\boldsymbol{\theta}^{(i)}) = \frac{\pi^{(i)} \mathcal{N}(\boldsymbol{y}_n; \boldsymbol{\phi}(\boldsymbol{x}_n, \boldsymbol{w}^{(c)}), \boldsymbol{\Sigma}^{(i)})}{\pi^{(i)} \mathcal{N}(\boldsymbol{y}_n; \boldsymbol{\phi}(\boldsymbol{x}_n, \boldsymbol{w}^{(c)}), \boldsymbol{\Sigma}^{(i)}) + (1 - \pi^{(i)}) \gamma^{(i)}}, \tag{2}$$

where (i) denotes the EM iteration index and $\boldsymbol{w}^{(c)}$ denotes the currently estimated network parameters. The posterior probability of the n-th data pair to be an outlier is $1 - r_n(\boldsymbol{\theta}^{(i)})$. The M-step updates the mixture parameters $\boldsymbol{\theta}$ with:

$$\boldsymbol{\Sigma}^{(i+1)} = \sum_{n=1}^{N} r_n(\boldsymbol{\theta}^{(i)}) \boldsymbol{\delta}_n^{(i)} \boldsymbol{\delta}_n^{(i)\top}, \tag{3}$$

$$\pi^{(i+1)} = \sum_{n=1}^{N} r_n(\boldsymbol{\theta}^{(i)})/N, \tag{4}$$

$$\frac{1}{\gamma^{(i+1)}} = \prod_{d=1}^{D} 2\sqrt{3\left(C_{2d}^{(i+1)} - \left(C_{1d}^{(i+1)}\right)^2\right)}, \tag{5}$$

where $\boldsymbol{\delta}_n^{(i)} = \boldsymbol{y}_n - \boldsymbol{\phi}(\boldsymbol{x}_n; \boldsymbol{w}^{(c)})$, and C_1 and C_2 are the first- and second-order centered data moments computed using ($\delta_{nd}^{(i)}$ denotes the d-th entry of $\boldsymbol{\delta}_n^{(i)}$):

$$C_{1d}^{(i+1)} = \frac{1}{N} \sum_{n=1}^{N} \frac{(1 - r_n(\boldsymbol{\theta}^{(i)}))}{1 - \pi^{(i+1)}} \delta_{nd}^{(i)}, \; C_{2d}^{(i+1)} = \frac{1}{N} \sum_{n=1}^{N} \frac{(1 - r_n(\boldsymbol{\theta}^{(i)}))}{1 - \pi^{(i+1)}} \left(\delta_{nd}^{(i)}\right)^2. \tag{6}$$

The iterative estimation of γ as just proposed has an advantage over using a constant value based on the volume of the data, as done in robust mixture models [7]. Indeed, γ is updated using the actual volume occupied by the outliers, which increases the ability of the algorithm to discriminate between inliers and outliers.

Another prominent advantage of DeepGUM for robustly predicting multidimensional outputs is its flexibility for handling the granularity of outliers. Consider for example to problem of locating landmarks in an image. One may want to devise a method that disregards outlying landmarks and not the whole image. In this case, one may use a GUM model for each landmark category. In the case of two-dimensional landmarks, this induces $D/2$ covariance matrices of size 2 (D is the dimensionality of the target space). Similarly one may use an coordinate-wise outlier model, namely D scalar variances. Finally, one may use an image-wise outlier model, i.e. the model detailed above. This flexibility is an attractive property of the proposed model as opposed to [3] which uses a coordinate-wise outlier model.

3.2 Network Loss Function

As already mentioned we use SGD to estimate the network parameters $\boldsymbol{w}$. Given the updated GUM parameters estimated with EM, $\boldsymbol{\theta}^{(c)}$, the regression loss function is weighted with the responsibility of each data pair:

$$\mathcal{L}_{\text{DEEPGUM}} = \sum_{n=1}^{N} r_n(\boldsymbol{\theta}^{(c)}) ||\boldsymbol{y}_n - \boldsymbol{\phi}(\boldsymbol{x}_n; \boldsymbol{w})||_2^2. \tag{7}$$

With this formulation, the contribution of a training pair to the loss gradient vanishes

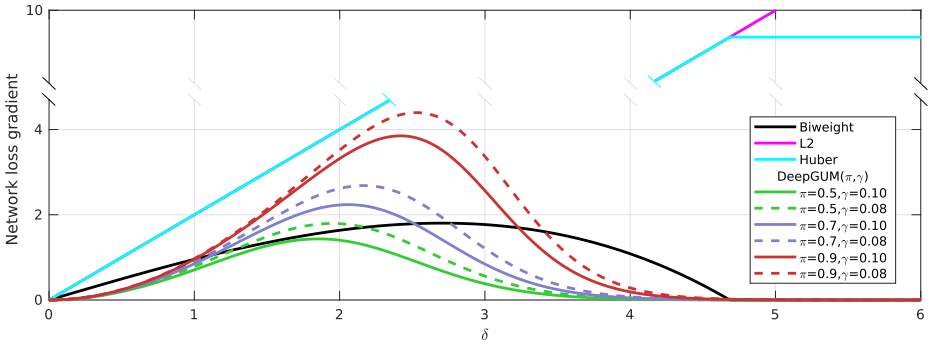


Fig. 2: Loss gradients for Biweight (black), Huber (cyan), L_2 (magenta), and DeepGUM (remaining colors). Huber and L_2 overlap up to $\delta = 4.6851$ (the plots are truncated along the vertical coordinate). DeepGUM is shown for different values of π and γ, although in practice they are estimated via EM. The gradients of DeepGUM and Biweight vanish for large residuals. DeepGUM offers some flexibility over Biweight thanks to π and γ.

(i) if the sample is an inlier with small error ($\|\boldsymbol{\delta}_n\|_2 \to 0, r_n \to 1$) or (ii) if the sample is an outlier ($r_n \to 0$). In both cases, the network will not back propagate any error. Consequently, the parameters $\boldsymbol{w}$ are updated only with inliers. This is graphically shown in Figure 2, where we plot the loss gradient as a function of a one-dimensional residual δ, for DeepGUM, Biweight, Huber and L_2. For fair comparison with Biweight and Huber, the plots correspond to a unit variance (i.e. standard normal, see discussion following eq. (3) in [3]). We plot the DeepGUM loss gradient for different values of π and γ to discuss different situations, although in practice all the parameters are estimated with EM. We observe that the gradient of the Huber loss increases linearly with δ, until reaching a stable point (corresponding to $c = 4.6851$ in [3]). Conversely, the gradient of both DeepGUM and Biweight vanishes for large residuals (i.e. $\delta > c$). Importantly, DeepGUM offers some flexibility as compared to Biweight. Indeed, we observe that when the amount of inliers increases (large π) or the spread of outliers increases (small γ), the importance given to inliers is higher, which is a desirable property. The opposite effect takes place for lower amounts of inliers and/or reduced outlier spread.

3.3 Training Algorithm

In order to train the proposed model, we assume the existence of a training and validation datasets, denoted $\mathcal{T} = \{\boldsymbol{x}_n^{\text{T}}, \boldsymbol{y}_n^{\text{T}}\}_{n=1}^{N_{\text{T}}}$ and $\mathcal{V} = \{\boldsymbol{x}_n^{\text{V}}, \boldsymbol{y}_n^{\text{V}}\}_{n=1}^{N_{\text{V}}}$, respectively. The training alternates between the unsupervised EM algorithm of Section 3.1 and the supervised SGD algorithm of Section 3.2, i.e. Algorithm 1. EM takes as input the training set, alternates between responsibility evaluation, (2) and mixture parameter update, (3), (4), (5), and iterates until convergence, namely until the mixture parameters do not evolve anymore. The current mixture parameters are used to evaluate the responsibilities of the validation set. The SGD algorithm takes as input the training and validation sets as

Algorithm 1 DeepGUM training

input: $\mathcal{T} = (\boldsymbol{x}_n^{\text{T}}, \boldsymbol{y}_n^{\text{T}})_{n=1}^{N_{\text{T}}}$, $\mathcal{V} = \{\boldsymbol{x}_n^{\text{V}}, \boldsymbol{y}_n^{\text{V}}\}_{n=1}^{N_{\text{V}}}$, and $\epsilon > 0$ (convergence threshold).
initialization: Run SGD on $\mathcal{T}$ to minimize (7) with $r_n = 1, \forall n$, until the convergence criterion on $\mathcal{V}$ is reached.
repeat
  EM algorithm: Unsupervised outlier detection
  repeat
    Update the r_n's with (2).
    Update the mixture parameters with (3), (4), (5).
  until The parameters $\boldsymbol{\theta}$ are stable.
  SGD: Deep regression learning
  repeat
    Run SGD to minimize $\mathcal{L}_{\text{DEEPGUM}}$ in (7).
  until Early stop with a patience of K epochs.
until $\mathcal{L}_{\text{DEEPGUM}}$ grows on $\mathcal{V}$.

well as the associated responsibilities. In order to prevent over-fitting, we perform early stopping on the validation set with a patience of K epochs.

Notice that the training procedure requires neither specific annotation of outliers nor the ratio of outliers present in the data. The procedure is initialized by executing SGD, as just described, with all the samples being supposed to be inliers, i.e. $r_n = 1, \forall n$. Algorithm 1 is stopped when $\mathcal{L}_{\text{DEEPGUM}}$ does not decrease anymore. It is important to notice that we do not need to constrain the model to avoid the trivial solution, namely all the samples are considered as outliers. This is because after the first SGD execution, the network can discriminate between the two categories. In the extreme case when DeepGUM would consider all the samples as outliers, the algorithm would stop after the first SGD run and would output the initial model.

Since EM provides the data covariance matrix $\boldsymbol{\Sigma}$, it may be tempting to use the Mahalanobis norm instead of the L_2 norm in (7). The covariance matrix is narrow along output dimensions with low-amplitude noise and wide along dimensions with high-amplitude noise. The Mahalanobis distance would give equal importance to low- and high-amplitude noise dimensions which is not desired. Another interesting feature of the proposed algorithm is that the posterior r_n weights the learning rate of sample n as its gradient is simply multiplied by r_n. Therefore, the proposed algorithm automatically selects a learning rate for each individual training sample.

4 Experiments

The purpose of the experimental validation is two-fold. First, we empirically validate DeepGUM with three datasets that are naturally corrupted with outliers. The validations are carried out with the following applications: fashion landmark detection (Section 4.1), age estimation (Section 4.2) and head pose estimation (Section 4.3). Second, we delve into the robustness of DeepGUM and analyze its behavior in comparison with

existing robust deep regression techniques by corrupting the annotations with an increasing percentage of outliers on the facial landmark detection task (Section 4.4).

We systematically compare DeepGUM with the standard L_2 loss, the Huber loss and the Biweight loss (used in [3]). In all these cases, we use the VGG-16 architecture [33] pre-trained on ImageNet [31]. We also tried to use the architecture proposed in [3], but we were unable to reproduce the results reported in [3] on the LSP and Parse datasets, using the code provided by the authors. Therefore, for the sake of reproducibility and for a fair comparison between different robust loss functions, we used VGG-16 in all our experiments. Following the recommendations from [19], we fine-tune the last convolutional block and both fully connected layers with a mini-batch of size 128 and learning rate set to 10^{-4}. The fine-tuning starts with 3 epochs of L_2 loss, before exploiting either the Biweight, Huber of DeepGUM loss. When using any of these three losses, the network output is normalized with the median absolute deviation (as in [3]), computed on the entire dataset after each epoch. Early stopping with a patience of $K = 5$ epochs is employed and the data is augmented using mirroring.

In order to evaluate the methods, we report the mean absolute error (MAE) between the regression target and the network output over the test set. In addition, we complete the evaluation with statistical tests that allow to point out when the differences between methods are systematic and statistically significant or due to chance. Statistical tests are run per-image regression errors and therefore can only be applied to the methods for which the code is available, and not to average errors reported in the literature; in the latter case, only MAE are made available. In practice, we use the non-parametric Wilcoxon signed-rank test [38] to assess whether the null hypothesis (the median difference between pairs of observations is zero) is true or false. We denote the statistical significance with *, ** or ***, corresponding to a p-value (the conditional probability of, given the null hypothesis is true, getting a test statistic as extreme or more extreme than the calculated test statistic) smaller than $p = 0.05$, $p = 0.01$ or $p = 0.001$, respectively. We only report the statistical significance of the methods with the lowest MAE. For instance, A*** means that *the probability that method* A *is equivalent to any other method is less than* $p = 0.001$.

4.1 Fashion Landmark Detection

Visual fashion analysis presents a wide spectrum of applications such as cloth recognition, retrieval, and recommendation. We employ the fashion landmark dataset (FLD) [21] that includes more than $120K$ images, where each image is labeled with eight landmarks. The dataset is equally divided in three subsets: upper-body clothes (6 landmarks), full-body clothes (8 landmarks) and lower-body clothes (4 landmarks). We randomly split each subset of the dataset into test ($5K$), validation ($5K$) and training ($\sim 30K$). Two metrics are used: the mean absolute error (MAE) of the landmark localization and the percentage of failures (landmarks detected further from the ground truth than a given threshold). We employ *landmark-wise* r_n.

Table 1 reports the results obtained on the upper-body subset of the fashion landmark dataset (additional results on full-body and lower-body subsets are included in the

Table 1: Mean absolute error on the upper-body subset of FLD, per landmark and in average. The landmarks are left (L) and right (R) collar (C), sleeve (S) and hem (H). The results of DFA are from [22] and therefore do not take part in the statistical comparison.

Method	Upper-body landmarks						
	LC	RC	LS	RS	LH	RH	Avg.
DFA [22] (L_2)	15.90	15.90	30.02	29.12	23.07	22.85	22.85
DFA [22] (5 VGG)	10.75	10.75	20.38	19.93	15.90	16.12	15.23
L_2	12.08	12.08	18.87	18.91	16.47	16.40	15.80
Huber [15]	14.32	13.71	20.85	19.57	20.06	19.99	18.08
Biweight [3]	13.32	13.29	21.88	21.84	18.49	18.44	17.88
DeepGUM	11.97^{***}	11.99^{***}	18.59^{***}	18.50^{***}	16.44^{***}	16.29^{***}	15.63^{***}

supplementary material). We report the mean average error (in pixels) for each landmark individually, and the overall average (last column). While for the first subset we can compare with the very recent results reported in [22], for the other there are no previously reported results. Generally speaking, we outperform all other baselines in average, but also in each of the individual landmarks. The only exception is the comparison against the method utilizing five VGG pipelines to estimate the position of the landmarks. Although this method reports slightly better performance than DeepGUM for some columns of Table 1, we recall that we are using one single VGG as front-end, and therefore the representation power cannot be the same as the one associated to a pipeline employing five VGG's trained for tasks such as pose estimation and cloth classification that clearly aid the fashion landmark estimation task.

Interestingly, DeepGUM yields better results than L_2 regression and a major improvement over Biweight [3] and Huber [15]. This behavior is systematic for all fashion landmarks and statistically significant (with $p < 0.001$). In order to better understand this behavior, we computed the percentage of outliers detected by DeepGUM and Biweight, which are 3% and 10% respectively (after convergence). We believe that within this difference (7% corresponds to $2.1K$ images) there are mostly "difficult" inliers, from which the network could learn a lot (and does it in DeepGUM) if they were not discarded as happens with Biweight. This illustrates the importance of rejecting the outliers while keeping the inliers in the learning loop, and exhibits the robustness of DeepGUM in doing so. Figure 3 displays a few landmarks estimated by DeepGUM.

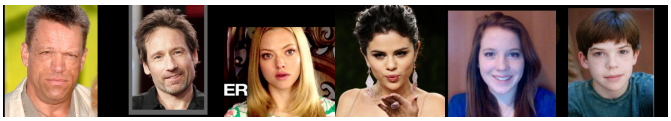
4.2 Age Estimation

Age estimation from a single face image is an important task in computer vision with applications in access control and human-computer interaction. This task is closely related to the prediction of other biometric and facial attributes, such as gender, ethnicity, and hair color. We use the cross-age celebrity dataset (CACD) [6] that contains $163,446$ images from $2,000$ celebrities. The images are collected from search engines

Fig. 3: Sample fashion landmarks detected by DeepGUM.

Method	MAE
L_2	5.75
Huber [15]	5.59
Biweight [3]	5.55
Dex [29]	5.25
DexGUM***	5.14
DeepGUM***	5.08

14 14 14 16 20 23



49 51 60 60 60 62

Fig. 4: Results on the CACD dataset: (left) mean absolute error and (right) images considered as outliers by DeepGUM, the annotation is displayed below each image.

using the celebrity's name and desired year (from 2004 to 2013). The dataset splits into 3 parts, $1,800$ celebrities are used for training, 80 for validation and 120 for testing. The validation and test sets are manually cleaned whereas the training set is noisy. In our experiments, we report results using *image-wise* r_n.

Apart from DeepGUM, L_2, Biweight and Huber, we also compare to the age estimation method based on deep expectation (Dex) [29], which was the winner of the Looking at People 2015 challenge. This method uses the VGG-16 architecture and poses the age estimation problem as a classification problem followed by a softmax expected value refinement. We report results with two different approaches using Dex. First, our implementation of the original Dex model. Second, we add the GUM model on top the the Dex architecture; we termed this architecture DexGUM.

The table in Figure 4 reports the results obtained on the CACD test set for age estimation. We report the mean absolute error (in years) for size different methods. We can easily observe that DeepGUM exhibits the best results: 5.08 years of MAE (0.7 years better than L_2). Importantly, the architectures using GUM (DeepGUM followed by DexGUM) are the ones offering the best performance. This claim is supported by the results of the statistical tests, which say that DexGUM and DeepGUM are statistically better than the rest (with $p < 0.001$), and that there are no statistical differences between them. This is further supported by the histogram of the error included in the supplementary material. DeepGUM considered that 7% of images were outliers and thus these images were undervalued during training. The images in Figure 4 correspond to outliers detected by DeepGUM during training, and illustrate the ability of DeepGUM

to detect outliers. Since the dataset was automatically annotated, it is prone to corrupted annotations. Indeed, the age of each celebrity is automatically annotated by subtracting the date of birth from the picture time-stamp. Intuitively, this procedure is problematic since it assumes that the automatically collected and annotated images show the right celebrity and that the times-tamp and date of birth are correct. Our experimental evaluation clearly demonstrates the benefit of a robust regression technique to operate on datasets populated with outliers.

4.3 Head Pose Estimation

The McGill real-world face video dataset [8] consists of 60 videos (a single participant per video, 31 women and 29 men) recorded with the goal of studying unconstrained face classification. The videos were recorded in both indoor and outdoor environments under different illumination conditions and participants move freely. Consequently, some frames suffer from important occlusions. The yaw angle (ranging from -90° to 90°) is annotated using a two-step labeling procedure that, first, automatically provides the most probable angle as well as a degree of confidence, and then the final label is chosen by a human annotator among the plausible angle values. Since the resulting annotations are not perfect it makes this dataset suitable to benchmark robust regression models. As the training and test sets are not separated in the original dataset, we perform a 7-fold cross-validation. We report the fold-wise MAE average and standard deviation as well as the statistical significance corresponding to the concatenation of the test results of the 7 folds. Importantly, only a subset of the dataset is publicly available (35 videos over 60).

In Table 2, we report the results obtained with different methods and employ a dagger to indicate when a particular method uses the entire dataset (60 videos) for training. We can easily notice that DeepGUM exhibits the best results compared to the other ConvNets methods (respectively 0.99°, 0.50° and 0.20° lower than L_2, Huber and Biweight in MAE). The last three approaches, all using deep architectures, significantly outperform the current state-of-the-art approach [9]. Among them, DeepGUM is significantly better than the rest with $p < 0.001$.

4.4 Facial Landmark Detection

We perform experiments on the LFW and NET facial landmark detection datasets [35] that consist of 5590 and 7876 face images, respectively. We combined both datasets and employed the same data partition as in [35]. Each face is labeled with the positions of five key-points in Cartesian coordinates, namely left and right eye, nose, and left and right corners of the mouth. The detection error is measured with the Euclidean distance between the estimated and the ground truth position of the landmark, divided by the width of the face image, as in [35]. The performance is measured with the failure rate of each landmark, where errors larger than 5% are counted as failures. The two aforementioned datasets can be considered as outlier-free since the average failure

Table 2: Mean average error on the McGill dataset. The results of the first half of the table are directly taken from the respective papers and therefore no statistical comparison is possible. †Uses extra training data.

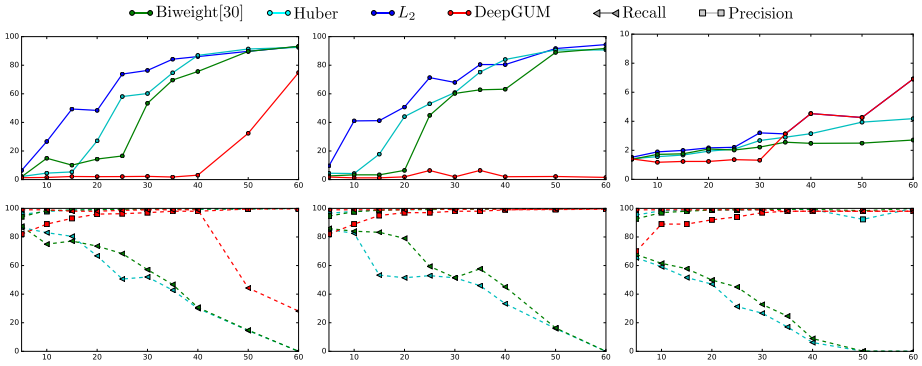
Method	MAE	RMSE
Xiong et al. [40]†	-	29.81 ± 7.73
Zhu and Ramanan [41]†	-	35.70 ± 7.48
Demirkus et al. [8]†	-	12.41 ± 1.60
Drouard et al. [9]	12.22 ± 6.42	23.00 ± 9.42
L_2	8.60 ± 1.18	12.03 ± 1.66
Huber [15]	8.11 ± 1.08	11.79 ± 1.59
Biweight [3]	7.81 ± 1.31	11.56 ± 1.95
DeepGUM***	7.61 ± 1.00	11.37 ± 1.34

rate reported in the literature falls below 1%. Therefore, we artificially modify the annotations of the datasets for facial landmark detection to find the breakdown point of DeepGUM. Our purpose is to study the robustness of the proposed deep mixture model to outliers generated in controlled conditions. We use three different types of outliers:

- Normally Generated Outliers (*NGO*): A percentage of landmarks is selected, regardless of whether they belong to the same image or not, and shifted a distance of d pixels in a uniformly chosen random direction. The distance d follows a Gaussian distribution, $\mathcal{N}(25, 2)$. *NGO* simulates errors produced by human annotators that made a mistake when clicking, thus annotating in a slightly wrong location.
- Local - Uniformly Generated Outliers (*l-UGO*): It follows the same philosophy as *NGO*, sampling the distance d from a uniform distribution over the image, instead of a Gaussian. Such errors simulate human errors that are not related to the human precision, such as not selecting the point or misunderstanding the image.
- Global - Uniformly Generated Outliers (*g-UGO*): As in the previous case, the landmarks are corrupted with uniform noise. However, in *g-UGO* the landmarks to be corrupted are grouped by image. In other words, we do not corrupt a subset of all landmarks regardless of the image they belong to, but rather corrupt all landmarks of a subset of the images. This strategy simulates problems with the annotation files or in the sensors in case of automatic annotation.

The first and the second types of outlier contamination employ *landmark-wise* r_n, while the third uses *image-wise* r_n.

The plots in Figure 5 report the failure rate of DeepGUM , Biweight, Huber and L_2 (top) on the clean test set and the outlier detection precision and recall of all except for L_2 (bottom) for the three types of synthetic noise on the corrupted training set. The precision corresponds to the percentage of training samples classified as outliers that are true outliers; and the recall corresponds to the percentage of outliers that are classified as such. The first conclusion that can be drawn directly from this figure are that, on the one hand, Biweight and Huber systematically present a lower recall than DeepGUM. In other words, DeepGUM exhibits the highest reliability at identifying and, therefore,



(a) *l-UGO* (b) *g-UGO* (c) *NGO*

Fig. 5: Evolution of the failure rate (top) when augmenting the noise for the 3 types of outliers considered. We also display the corresponding precisions and recalls in percentage (bottom) for the outlier class. Best seen in color.

ignoring outliers during training. And, on the other hand, DeepGUM tends to present a lower failure rate than Biweight, Huber and L_2 in most of the scenarios contemplated.

Regarding the four most-left plots, *l-UGO* and *g-UGO*, we can clearly observe that, while for limited amounts of outliers (i.e. $< 10\%$) all methods report comparable performance, DeepGUM is clearly superior to L_2, Biweight and Huber for larger amounts of outliers. We can also safely identify a breakdown point of DeepGUM on *l-UGO* at $\sim 40\%$. This is inline with the reported precision and recall for the outlier detection task. While for Biweight and Huber, both decrease when increasing the number of outliers, these measures are constantly around 99% for DeepGUM (before 40% for *l-UGO*). The fact that the breakdown point of DeepGUM under *g-UGO* is higher than 50% is due to fact that the a priori model of the outliers (i.e. uniform distribution) corresponds to the way the data is corrupted.

For *NGO*, the corrupted annotation is always around the ground truth, leading to a failure rate smaller than 7% for all methods. We can see that all four methods exhibit comparable performance up to 30% of outliers. Beyond that threshold, Biweight outperforms the other methods in spite of presenting a progressively lower recall and a high precision (i.e. Biweight identifies very few outliers, but the ones identified are true outliers). This behavior is also exhibited by Huber. Regarding DeepGUM, we observe that in this particular setting the results are aligned with L_2. This is because the SGD procedure is not able to find a better optimum after the first epoch and therefore the early stopping mechanism is triggered and SFD output the initial network, which corresponds to L_2. We can conclude that the strategy of DeepGUM, consisting in removing all points detected as outliers, is not effective in this particular experiment. In other words, having more noisy data is better than having only few clean data in this particular case of 0-mean highly correlated noise. Nevertheless, we consider an attractive property of DeepGUM the fact that it can automatically identify these particular cases and return an acceptable solution.

5 Conclusions

This paper introduced a deep robust regression learning method that uses a Gaussian-uniform mixture model. The novelty of the paper resides in combining a probabilistic robust mixture model with deep learning in a jointly trainable fashion. In this context, previous studies only dealt with the classical L_2 loss function or Tukey's Biweight function, an M-estimator robust to outliers [3]. Our proposal yields better performance than previous deep regression approaches by proposing a novel technique, and the derived optimization procedure, that alternates between the unsupervised task of outlier detection and the supervised task of learning network parameters. The experimental validation addresses four different tasks: facial and fashion landmark detection, age estimation, and head pose estimation. We have empirically shown that DeepGUM (i) is a robust deep regression approach that does not need to rigidly specify *a priori* the distribution (number and spread) of outliers, (ii) exhibits a higher breakdown point than existing methods when the outliers are sampled from a uniform distribution (being able to deal with more than 50% of outlier contamination without providing incorrect results), and (iii) is capable of providing comparable or better results than current state-of-the-art approaches in the four aforementioned tasks. Finally, DeepGUM could be easily used to remove undesired samples that arise from tedious manual annotation. It could also deal with highly unusual training samples inherently present in automatically collected huge datasets, a problem that is currently addressed using error-prone and time-consuming human supervision.

Acknowledgments. This work was supported by the European Research Council via the ERC Advanced Grant VHIA (Vision and Hearing in Action) #113340.

References

1. Banfield, J.D., Raftery, A.E.: Model-based gaussian and non-gaussian clustering. Biometrics pp. 803–821 (1993)
2. Bekker, A.J., Goldberger, J.: Training deep neural-networks based on unreliable labels. In: ICASSP. pp. 2682–2686 (2016)
3. Belagiannis, V., Rupprecht, C., Carneiro, G., Navab, N.: Robust optimization for deep regression. In: ICCV (2015)
4. Beliakov, G., Kelarev, A.V., Yearwood, J.: Robust artificial neural networks and outlier detection. Technical report. CoRR **abs/1110.0169** (2011)
5. Black, M.J., Rangarajan, A.: On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. IJCV **19**(1), 57–91 (1996)
6. Chen, B.C., Chen, C.S., Hsu, W.H.: Cross-age reference coding for age-invariant face recognition and retrieval. In: ECCV (2014)
7. Coretto, P., Hennig, C.: Robust improper maximum likelihood: tuning, computation, and a comparison with other methods for robust Gaussian clustering. JASA **111**, 1648–1659 (2016)
8. Demirkus, M., Precup, D., Clark, J.J., Arbel, T.: Hierarchical temporal graphical model for head pose estimation and subsequent attribute classification in real-world videos. CVIU pp. 128–145 (2015)
9. Drouard, V., Horaud, R., Deleforge, A., Ba, S., Evangelidis, G.: Robust head-pose estimation based on partially-latent mixture of linear regressions. TIP **26**, 1428–1440 (2017)

10. Forbes, F., Wraith, D.: A new family of multivariate heavy-tailed distributions with variable marginal amounts of tailweight: application to robust clustering. Statistics and Computing **24**(6), 971–984 (2014)
11. Galimzianova, A., Pernus, F., Likar, B., Spiclin, Z.: Robust estimation of unbalanced mixture models on samples with outliers. TPAMI **37**(11), 2273 – 2285 (2015)
12. Gebru, I.D., Alameda-Pineda, X., Forbes, F., Horaud, R.: Em algorithms for weighted-data clustering with application to audio-visual scene analysis. IEEE TPAMI **38**(12), 2402–2415 (2016)
13. Gelman, A., Carlin, J., Stern, H., Rubin, D.: Bayesian Data Analysis. Chapman & Hall/CRC Texts in Statistical Science, Taylor & Francis (2003)
14. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: CVPR (2014)
15. Huber, P.J.: Robust estimation of a location parameter. The annals of mathematical statistics pp. 73–101 (1964)
16. Huber, P.: Robust Statistics. Wiley (2004)
17. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. In: NIPS (2012)
18. Lathuilière, S., Juge, R., Mesejo, P., Muñoz Salinas, R., Horaud, R.: Deep Mixture of Linear Inverse Regressions Applied to Head-Pose Estimation. In: CVPR (2017)
19. Lathuilière, S., Mesejo, P., Alameda-Pineda, X., Horaud, R.: A comprehensive analysis of deep regression. arXiv preprint arXiv:1803.08450 (2018)
20. Li, Y., Yang, J., Song, Y., Cao, L., Luo, J., Li, J.: Learning from Noisy Labels with Distillation. arXiv preprint arXiv:1703.02391 (2017)
21. Liu, Z., Luo, P., Qiu, S., Wang, X., Tang, X.: Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In: CVPR (2016)
22. Liu, Z., Yan, S., Luo, P., Wang, X., Tang, X.: Fashion Landmark Detection in the Wild. In: ECCV (2016)
23. Maronna, R.A., Martin, D.R., Yohai, V.J.: Robust statistics. John Wiley & Sons (2006)
24. Meer, P., Mintz, D., Rosenfeld, A., Kim, D.Y.: Robust regression methods for computer vision: A review. IJCV **6**(1), 59–70 (1991)
25. Mukherjee, S., Robertson, N.: Deep Head Pose: Gaze-Direction Estimation in Multimodal Video. TMM **17**(11), 2094–2107 (2015)
26. Neuneier, R., Zimmermann, H.G.: How to train neural networks. In: Neural Networks: Tricks of the Trade, pp. 373–423. Springer Berlin Heidelberg (1998)
27. Neykov, N., Filzmoser, P., Dimova, R., Neytchev, P.: Robust fitting of mixtures using the trimmed likelihood estimator. CSDA **52**(1), 299–308 (2007)
28. Ranjan, R., Patel, V.M., Chellappa, R.: Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. CoRR **abs/1603.01249** (2016)
29. Rothe, R., Timofte, R., Van Gool, L.: Deep expectation of real and apparent age from a single image without facial landmarks. IJCV (2016)
30. Rousseeuw, P.J., Leroy, A.M.: Robust regression and outlier detection, vol. 589. John Wiley & sons (2005)
31. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV **115**(3), 211–252 (2015)
32. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., Lecun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. In: ICLR (2014)
33. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)

34. Stewart, C.V.: Robust parameter estimation in computer vision. SIAM Review **41**(3), 513–537 (1999)
35. Sun, Y., Wang, X., Tang, X.: Deep convolutional network cascade for facial point detection. In: CVPR (2013)
36. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR (2015)
37. Toshev, A., Szegedy, C.: DeepPose: Human Pose Estimation via Deep Neural Networks. In: CVPR (2014)
38. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin pp. 80–83 (1945)
39. Xiao, T., Xia, T., Yang, Y., Huang, C., Wang, X.: Learning from massive noisy labeled data for image classification. In: CVPR (2015)
40. Xiong, X., De la Torre, F.: Supervised descent method and its applications to face alignment. In: CVPR. pp. 532–539 (06 2013)
41. Zhu, X., Ramanan, D.: Face detection, pose estimation, and landmark localization in the wild. In: CVPR. pp. 2879–2886 (2012)