Probabilistic Regression with Huber Distributions

David Mohlin — Tobii AB / RPL, KTH
davmo@kth.se
Gerald Bianchi — Supervision during employment at Tobii
gerald.bianchi@gmail.com
Josephine Sullivan — RPL, KTH
sullivan@kth.se

Abstract

In this paper we describe a probabilistic method for estimating the position of an object along with its covariance matrix using neural networks. Our method is designed to be robust to outliers, have bounded gradients with respect to the network outputs, among other desirable properties. To achieve this we introduce a novel probability distribution inspired by the Huber loss. We also introduce a new way to parameterize positive definite matrices to ensure invariance to the choice of orientation for the coordinate system we regress over. We evaluate our method on popular body pose and facial landmark datasets and get performance on par or exceeding the performance of non-heatmap methods. Our code is available at github.com/Davmo049/Public_prob_regression_with_huber_distributions

1 Introduction

Estimating positions of objects is a well studied topic, due to its many applications. It is for example used for facial landmark estimation [18], autonomous driving [8, 12, 22] and body pose estimation [10, 26, 29, 34]. Regression can also appear as a component of more complicated systems such as predicting offsets of a bounding box relative to an anchor point for object detection [25]. Estimating uncertainties associated with these estimated positions has applications for example in time filtering, such as Kalman filters. Uncertainties can also be used for task specific problems, for example an autonomous vehicle should be able to come to a stop before it enters the region where an object is likely to be, not before it gets to the most likely position of the object.

Defining a loss function is a crucial step in designing a neural network. Robustness to outliers is an important property of the loss in order to reduce sensitivity to large errors. Robust loss functions for neural networks have been shown to improve performance of the model. In [11] they show that minimizing L1 and smooth L1 loss yields higher performances than the standard L2 loss. Motivated this they introduce the Wing loss, which decreases the impact large errors have. An extension of the Wing loss was presented in [35] in the context of heatmap regression. In [4], the author presents a generalization of several common loss functions with a robustness parameter which is automatically tuned during the training step.

This approach allows the optimization to determine how robust the loss should be depending on the training data

Many of the current approaches to regress positions can be divided into two categories: heatmap based methods and direct regression. Heatmap based methods estimate a heatmap of where the object could be over a quantized set, roughly corresponding to pixels in the image. This heatmap is then converted into a position through various heuristics such using the expected position, the most likely position or something similar. Direct regression does not introduce this intermediate representation and instead predict a vector in $\mathbb{R}^N$ directly from the latent representation of the network. Direct regression has been applied on for example depth estimation [4] and estimation of facial landmarks [11].

In general heatmap methods are more complicated, requiring heuristics for how to construct the loss and how to extract a position out of the heatmap. State of the art methods often also introduce other complex components such as using a cascade of deep neural networks [18, 23, 31] or multi-resolution networks [28].

While heatmap methods are currently the prevalent approach, we think that direct regression methods are still attractive for solving the problem of position estimation for three reasons. Firstly, direct regression methods do not quantize the output space into bins, which results in quantization errors and scales poorly to higher dimensions. Secondly, direct regression directly output the coordinates of the object, removing the need for a heuristic which convert heatmaps to coordinates. Finally, direct regression do not require complex network architectures to produce heatmaps, instead any standard network backbone such as resnet, inception, mobilenet or squeezenet can be used.

We evaluate this method on popular facial landmark (WFLW) and body pose (MPII) datasets. The results show that our method outperforms existing regression methods but gives slightly lower performance than the state of the art for heatmap based methods.

In this paper we introduce the Huber L_2 distribution a novel probability distribution, parameterized by a mean position and a covariance matrix. We fit neural networks to predict this prediction by minimizing the negative log likelihood between the predictions and annotations. We design the method to ensure the following desirable properties: i) unimodality of the distribution ii) using a distribution with exponential tail behaviour to make the method robust to outliers, iii) make the method invariant to the orientation of the coordinate system we regress over, iv) have bounded gradients to avoid too large parameter updates in gradient descent v) make the method loss have bounded hessians and make the loss convex for a region which we argue covers all reasonable outputs.

2 Related Work

Multivariate regression and estimating 2D and 3D position has been extensively studied in computer vision. Both with classic machine learning and deep learning techniques.

The state of the art methods for pose estimation on the body pose dataset MPII [2] have been heatmap based since 2014 [5, 23, 27, 30, 32, 33, 38]. Some innovations introduced since then are multi-stage losses [7], the hourglass network architecture [23] and exploiting additional training datasets [27, 37]. For the facial landmark dataset WFLW [36] heatmap based methods are also popular for achieving high performance [18, 24, 35].

An intermediate heatmap representation is not the basis of all position estimation methods. Carreira *et al.* [7] use regression to predict landmark positions, these predictions are then converted into heatmaps and concatenated to the original image to be sent through a second

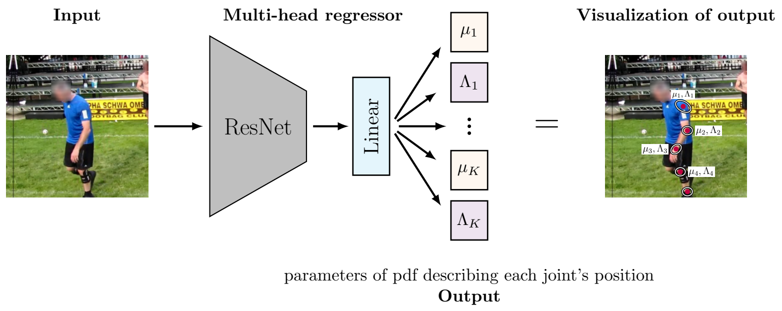

Figure 1: **Overview of our simple regression architecture.** In this paper we introduce a simple regression method that produces a probabilistic estimate of a d-dimensional vector. An input image is fed into any standard backbone network architecture with an additional final linear layer that outputs the parameters, mean μ and *precision* Λ, of the L_2 *multivariate Huber distribution* describing the likely values of the vector. This Huber distribution is introduced to allow robust estimation from noisy landmark datasets common in computer vision. Above depicts our method adapted to predict the location of body joints.

network for the final prediction. Feng *et al.* [11] introduce the Wing loss to focus more on small and medium size errors during the training when predicting facial landmarks, effectively making their method less affected by outliers. Sun *et al.* [29] modeled the position of body joints as a tree with the pelvis as a root and the position of a child joint being defined by an offset from its parent.

For the problem of predicting uncertainty for position it is common to construct a covariance matrix from the network output. The covariance matrix is symmetric positive definite by definition. One method to do this to estimate a diagonal covariance matrix [4], by applying a mapping from $\mathbb{R}$ to $\mathbb{R}^+$ on each value, such as a softplus function, it is possible to guarentee positive definiteness. A problem with this approach is that diagonal matrices is only a subset of all positive definite matrices.. Another common method is to predict parameters for a decomposition of the matrix using neural networks, followed by a reconstruction of the covariance matrix. Common decompositions to predict parameters of are the LDL decomposition[20] or the Cholesky decomposition [13, 18].

3 Method

For an input, $\mathbf{x} \in \mathbb{R}^D$, we want to predict $\mathbf{y} \in \mathbb{R}^d$ in a probabilistic fashion. We achieve this by training a neural network that outputs the parameters of a probability distribution for $\mathbf{y}$ over $\mathbb{R}^d$. In this section we introduce a novel distribution for this purpose and from it derive the loss we use for training our network. This loss and its parameterization from network output is designed to be invariant to the choice of orientation of the coordinate system we do regression in, have bounded gradients, is convex for the set of precision matrices with eigenvalues larger than a threshold corresponding to the smallest precision value deemed reasonable for the specific regression task and has bounded Hessians for the same set. These properties are proven in supplementary material C. Since one of the parameters of our distribution is a symmetric positive definite matrix, we also describe the procedure used to map the network's output to a symmetric positive definite matrix.

3.1 The L_2 multivariate Huber distribution

We define the L_2 *multivariate Huber distribution* for $\mathbf{y} \in \mathbb{R}^d$ as

$$p(\mathbf{y} \mid \mu, \Lambda, \delta) \propto \exp\left(-h_\delta\left(\|\Lambda^{1/2}(\mathbf{y}-\mu)\|_2\right)\right) \tag{1}$$

where $\Lambda \in \mathbb{R}^{d\times d}$ is a symmetric positive definite matrix, $\mu \in \mathbb{R}^d$ and

$$h_\delta(y) = \begin{cases} y^2/2 & \text{if } |y| \leq \delta \\ \delta(|y| - \delta/2) & \text{otherwise} \end{cases} \tag{2}$$

The function h_δ is the well known Huber function parametrized by $\delta \in [0, \infty)$. Intuitively μ is the mean position and Λ is the precision multiplied by a constant, see supplementary material A.2 for details. Our multivariate Huber distribution is similar, but not identical to the multivariate Huber distribution defined in [3]. In the latter the Huber loss is applied independently to each dimension of the vector as opposed to in our distribution (equation (1)) where the Huber loss is applied on the L_2 norm. Our distribution has a tail density which is $\mathcal{O}(\exp(-\|y\|_2))$, compared to a Gauss distribution which has $\mathcal{O}(\exp(-\|y\|_2^2))$. This makes maximum likelihood estimators of μ and Λ less dependent on outliers, compared to a Gauss distribution. This is an important property for robust estimation in the presence of mislabelings and heavy-tailed noise. See the section G for visualizations of the distribution.

When we estimate the parameters of the Huber distribution we use the negative log-likelihood of the distribution as the training loss. A slightly different parametrization of equation (1) where we instead estimate $A = \Lambda^{1/2}$, $\nu = \Lambda^{1/2}\mu$ gives this loss nice mathematical properties. The multivariate Huber distribution becomes, including the normalising constant and dropping the subscript in $\|\cdot\|_2$:

$$p_{\text{huber}}(\mathbf{y} \mid \nu, A, \delta) = \frac{|A|}{c_d(\delta)} \exp\left(-h_\delta\left(\|A\mathbf{y} - \nu)\|\right)\right) \tag{3}$$

where $|\cdot|$ denotes the determinant and c_d is a constant depending only on the dimensionality of $\mathbf{y}$ and δ. See supplementary A.1 for details about c_d. Note that in our experiments we set δ to be constant.

3.2 Loss based on the L_2 multivariate Huber distribution

Assume we have labelled training data, $\mathcal{D}$, where an example $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ has input $\mathbf{x} \in \mathbb{R}^D$ and a corresponding position vector $\mathbf{y} \in \mathbb{R}^d$. Our goal is to train a neural network that will map an input $\mathbf{x}$ to a probabilistic estimate of its position. We learn a function, f, defined by parameters $\Theta \in \mathbb{R}^p$ and encoded as a neural network that outputs a vector of length $(d + d(d+1)/2)$ given input $\mathbf{x}$. We then apply a function q to the output vector to return the parameters ν and A of a L_2 multivariate Huber distribution that is

$$q(f(\mathbf{x}, \Theta)) = (\nu_{\mathbf{x},\Theta}, A_{\mathbf{x},\Theta}) \quad \text{where} \quad q : \mathbb{R}^{d+d(d+1)/2} \to \mathbb{R}^d \times \mathbb{R}^{d\times d} \tag{4}$$

The parameters, Θ, can be found by maximimizing the likelihood of the training data w.r.t. the multivariate Huber distribution of equation (3) or equivalently minimizing the negative log-likelihood of the training data:

$$\arg\max_{\Theta} \prod_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} p_{\text{huber}}(\mathbf{y} \mid \nu_{\mathbf{x},\Theta}, A_{\mathbf{x},\Theta}) = \arg\min_{\Theta} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} -\log\left(p_{\text{huber}}(\mathbf{y} \mid \nu_{\mathbf{x},\Theta}, A_{\mathbf{x},\Theta})\right) \tag{5}$$

When dropping the Θ subscript to reduce the notation clutter in equation (5) and using the fact that δ is constant we get the loss

$$\mathfrak{L}(\mathbf{y}, \nu_{\mathbf{x}}, A_{\mathbf{x}}) = -\log(|A_{\mathbf{x}}|) + h_{\delta}(\|A_{\mathbf{x}}\mathbf{y} - \nu_{\mathbf{x}})\|) \tag{6}$$

In our experiments we regress for multiple keypoints simultaneously. We do this by increasing the number of outputs of the network proportional to the number of keypoints and sum the losses, equation (6), for each keypoint into one total loss. In a probabilistic framework this corresponds to the assumption that the position of each keypoint is independent.

3.3 Predicting a symmetric positive definite matrix

The function q, mapping the network's output to the distribution's parameters, should both span all possible output parameters as well as fulfill the parameter's constraints. For the mean vector $\nu \in \mathbb{R}^d$, we can simply let ν correspond to the first d numbers output by f. Generating the matrix $A_{\mathbf{x}}$ is trickier as it needs to be symmetric positive definite. We also want to ensure the procedure we use to construct $A_{\mathbf{x}}$ does not bias this matrix to have certain eigenvectors for certain eigenvalues. This motivates the following approach.

Let the vector $\mathbf{v}$ correspond to the last $d(d+1)/2$ numbers output by f. From $\mathbf{v}$ we can create a symmetric matrix B with a bijection π

$$B_{i,j} = \begin{cases} v_{\pi(i,j)}/\sqrt{2} & \text{if } i > j \\ v_{\pi(j,i)}/\sqrt{2} & \text{if } i < j \\ v_{\pi(i,i)} & \text{if } i = j \end{cases} \tag{7}$$

where

$$\pi : \{(i,j) \mid i,j \in \{1,\ldots,d\} \ \&\ i \leq j\} \longrightarrow \{1,\ldots,d(d+1)/2\} \tag{8}$$

Next we perform an eigenvalue decomposition of $B = V^T \operatorname{diag}(\lambda_1,\ldots,\lambda_d)V$. B being symmetric and real ensures V is orthonormal (ON) and each eigenvalue λ_i is real. We then construct a positive definite matrix by applying a function $g : \mathbb{R} \to \mathbb{R}^+$ on each eigenvalue independently.

$$g(\lambda) = \begin{cases} \lambda & \text{if } \lambda > \theta \\ \theta \exp(\lambda/\theta - 1) & \text{otherwise} \end{cases} \tag{9}$$

The value of θ corresponds to the smallest reasonable precision for the task. For example when doing regression on an image one should not need to be able to output distributions with a standard deviation larger than the size of the image.

We can then construct our output matrix as

$$A = V^T \operatorname{diag}(g(\lambda_1),\ldots,g(\lambda_d))V \tag{10}$$

This procedure is not biased to output certain eigenvectors for certain eigenvalues if $\mathbf{v}$ is not biased toward certain directions. This is because the mapping to create B is an isometry between $\mathbb{R}^{d(d+1)/2}$ w.r.t the L_2 norm and the set of symmetric matrices with respect to the Frobenius norm. Since the Frobenius norm only depends on the eigenvalues, not the eigenvectors this means that the eigenvectors of A and B will be unbiased.

It is not completely straightforward to do backward propagation through the above sequence of steps, but the analytical expression and proof of its correctness is available in supplementary material B.

4 Experiments

We evaluate the effectiveness of our probabilistic Huber loss and approach on the problem of keypoint prediction for images. The two datasets we use are the facial landmark dataset WFLW [36] and the 2D body keypoint dataset MPII [2] where the goal is to estimate the 2D position of each landmark in the image. The facial landmark dataset demands high precision while the human pose dataset has a large degree of variation. We run our method 5 times with different random seeds and report the mean performance of the runs.

4.1 Implementation details

Preprocessing Both MPII and WLFW provide bounding boxes for each person/face considered as well as the annotated keypoints corresponding to each face. The input to our network at test time is constructed by creating a square crop centered at the center of the bounding box. The length of the cropped region is proportional to the distance between the bounding box corners and its center. The crop is then scaled, keeping the aspect ratio intact, to the network's expected input size.

For training time we do the same except we also use standard data augmentations such as mirroring the image, using random rotations, scaling, translations and perspective distortions. All of these operations can be combined into an affine transform. We construct the input image sampling pixels using bilinear interpolation and edge replication using this transform. By only sampling pixel values once we avoid creating excessive blur. We use the same affine transform to compute where each annotated landmark will be in the input image.

Finally we apply an affine normalization for each landmark such that each normalized landmark is zero mean with identity covariance across all training samples.

Test time data-augmentations Test time augmentations are frequently applied to improve performance on MPII and WFLW [5, 23, 27, 30, 38]. We do this by combining the outputs of our model for a non-augmented input and a mirrored input. We convert the predictions for the mirrored input to the same coordinate system as the non-augmented output by using the affine transforms used for preprocessing.

The standard way to combine test time augmentations is by averaging the predictions. Since our method outputs probability distributions, we can fuse our predictions using the maximum likelihood (ML) point. Empirically the two methods performed similarly, this could be because a mirrored and non-mirrored input produces similar covariance matrices. Finding the ML point for multiple independent Huber can be done by using Majorize/Minimize (MM) of quadratic functions for quick and guaranteed convergence. See Supplementary material E for details.

Other implementation details For our experiments we use the ResNet family of convolutional networks[15] as our backbone network. We use an input size of 224×224 pixels.

For this input we get an output with a spatial resolution of 7×7. Conventionally this spatial resolution is then reduced to 1×1 by average pooling. Pooling is often motivated by a desire to have invariance to small translation, scale and/or rotation changes. However, this property is undesirable for regression and we replace this average pooling with channel wise convolutions whose parameters are learned. This change improved performance by 0.24 NME for WFLW, see supplementary material D for corresponding tables.

For the regression head we use a linear mapping from the latent space to a $5K$ dimensional output, where K is the number of keypoints we want to estimate and 5 is the number of parameters we need to parameterize our L_2 multivariate Huber distributions for two dimensions. The δ parameter of the loss is set to be 1.

4.2 Evaluation metrics

There are standard evaluation metrics associated with the datasets WFLW and MPII. For WFLW it is the *Normalized Mean Error* (NME) between the predicted and ground truth landmark coordinates where the normalization is performed w.r.t. the interoccular distance. The standard evaluation metric for MPII is PCKh(@0.5). The metric measures the percentage of the predicted keypoint locations whose distance to its ground truth location are within 50% of the head segment's length. We now review of how we quantitatively evaluate our probability distributions.

Quantitative evaluation of uncertainty and error We investigate whether the estimated precision matrices for test samples are correlated with the actual prediction error. To achieve this we use a similar approach to [18]. First for each test sample we compute its expected error from the predicted covariance. Next we group samples with similar expected errors into bins. For the presented plots we use a bin size of 734. Finally we plot the average expected error against the average empirical error for each bin.

5 Results

5.1 Performance relative to other methods

Table 1 reports the performance of our best performing networks for WFLW and MPII compared to those of recent high-performing approaches which involve landmark heatmaps in some form or other and those which do not. Our best-performing networks have a ResNet101 backbone architecture, are trained using our probabilistic Huber loss with a ν parametrization for 200 epochs and use test-time augmentation with our probabilistic fusion. For WFLW our best performing network ranks ∼4th, w.r.t. the NME metric, of all published methods and is the best performing method which directly regresses from a compact encoding of the input image. For MPII our results are respectable, given our streamlined approach, but significantly below state of the art performance. Once again our approach is best amongst single stage regression from a compact representation.

5.2 Ablation experiments

Loss In this set of ablation experiments we compare our probabilistic L_2 multivariate Huber loss, equation (6), to other probabilistic based losses. In particular we compare to losses using the negative log-likelihood (NLL) of the Gauss, Laplace and Charbonnier distributions:

$$\mathfrak{L}_{\text{Gauss}}(\mathbf{y},\nu,A) = -\log(|A|) + \|Ay-\nu\|_2^2/2 \tag{11}$$

$$\mathfrak{L}_{\text{Laplace}}(\mathbf{y},\nu,A) = -\log(|A|) + \|Ay-\nu\|_2 \tag{12}$$

$$\mathfrak{L}_{\text{Charb.}}(\mathbf{y},\nu,A) = -\log(|A|) + \sqrt{\|Ay-\nu\|_2^2+1} - 1 \tag{13}$$

Table 1: Comparison of our regression approach to state of art methods on WLFW and MPII. Method marked with † construct heatmaps of predictions as additional input for multi stage regression. Entries marked with * use additional data.

(a) WFLW

Method	Heatmaps	NME
CFSS[39]	✗	9.07
DVLN[37]	✗	10.84
LAB[36]	✓	5.27
Wing[11]	✗	5.11
DeCaFa[9]	✓	4.62
AVS[24]	✓	4.39
AWing[35]	✓	**4.36**
LUVLi[18]	✓	4.37
Ours	✗	4.62

(b) MPII

Method	Heatmaps	PCKh(@0.5)
Tompson *et al.* [32]	✓	79.6
Tompson *et al.* [33]	✓	82.0
Newell *et al.* [23]	✓	90.9
Yang *et al.* [38]	✓	92.0
Bin *et al.* * [5]	✓	**94.1**
Carreira *et al.* [7]	Partial†	81.3
Sun *et al.* [29] (Stage 0)	✗	79.6
Sun *et al.* [29] (Stage 1)	Partial†	86.4
Lathuilière *et al.* [19]	✗	61.6
Ours	✗	85.4

We also compare the performance when we constrain the losses examined to have the identity, diagonal and full covariance matrices. Note when the covariance, A, is set to the identity matrix the above losses reduce to the mean squared error loss, mean error loss and the Charbonnier loss respectively. Our loss is reduced to the standard Huber loss when A is set to be the identity matrix. The δ parameter was set to be the constant 1 for all Huber losses. Tuning this value might increase the performance for these methods. The results of these ablation experiments on WFLW are presented in tables 2 (standard dataset evaluation metric) and 3 (NLL) and on MPII in table 4 (standard dataset evaluation metric).

Table 2: WFLW NME

Distribution	Identity covariance (↓)	Diagonal covariance (↓)	Full covariance (↓)
Huber (ours)	5.52 ± 0.02	4.91 ± 0.03	4.91 ± 0.04
Gauss	6.04 ± 0.10	5.65 ± 0.06	5.40 ± 0.17
Laplace	5.16 ± 0.03	4.90 ± 0.07	4.89 ± 0.02
Charbonnier	5.64 ± 0.03	4.95 ± 0.02	4.96 ± 0.02

Table 3: NLL WFLW

Distribution	Diagonal covariance (↓)	Full covariance (↓)
Huber (ours)	-307.0 ± 1.3	-310.0 ± 1.4
Gauss	-274.0 ± 2.1	-279.3 ± 5.6
Laplace	-306.6 ± 2.8	-309.9 ± 1.2
Charbonnier	-305.9 ± 1.1	-308.9 ± 0.8

By comparing column 1 with column 2 in table 2 we see that estimating the covariance matrix jointly with the position improves the performance of the position estimate. By

Table 4: MPII PCKh@0.5

Distribution	Identity covariance (↑)	Diagonal covariance (↑)	Full covariance (↑)
Huber (ours)	81.9 ± 0.1	85.1 ± 0.1	85.0 ± 0.1
Gauss	77.0 ± 0.1	81.3 ± 0.6	81.2 ± 0.5
Laplace	82.8 ± 0.1	85.0 ± 0.1	85.0 ± 0.1
Charbonnier	80.2 ± 0.1	84.7 ± 0.1	84.7 ± 0.1

comparing column 1 with column 2 in table 3 we see that modeling the full covariance matrix, compared to modeling a diagonal matrix, consistently improves the NLL by approximately 3 units. The performance of the position estimate does not change significantly when modeling a full covariance matrix compared to only modeling a diagonal matrix, as can be seen by comparing column 2 with column 3 in table 2.

From tables 3 and 2 it is apparent that the Gauss loss has significantly worse performance compared to the other losses. This could be due to the fact that the tails for a Gauss distribution decays with $\mathcal{O}(e^{-r^2})$, whereas the other distributions have tails decaying with $\mathcal{O}(e^{-|r|})$. A distribution with quickly decaying tails tends to be less robust to outliers.

Other ablation experiments The supplementary material contains the results of more ablation studies, see section D, w.r.t. the network architecture, the effect of replacing the final global pooling layer of the ResNet with trainable convolutions and the effect of training from a random initialization versus pre-training on large image repositories. A summary of the results are: both ResNet50 and ResNet101 produce better results than ResNet18, replacing the final global pooling with a convolution improves performance, pre-training improves performance and estimating the distributions mean indirectly by using the ν parameterization has similar performance to estimating the mean directly with the μ paramteterization, as described in section 3.1.

5.3 Accuracy of probability estimates

Figure 3 displays plots of the expected error predicted for the test data in WFLW plotted against the actual average error as described in section 4.2. We see that the network which is trained for 50 epochs predicts covariances which are well aligned with the empirical error, on average. The network which is trained for 200 epochs consistently underestimate the variance. In the supplementary material D we can see that the 200 epochs network gets better NME performance but worse NLL performance. These observations are consistent with prior work [14, 21] which shows that assigned probabilities generally start to overfit earlier than the point estimate.

Qualitatively uncertainties tend to correspond to occlusion, strange poses or other ambiguities. In figure 2 we show two examples. In the left image most body parts are clearly visible and estimated uncertainties are low with low errors to ground truth locations. In the right image it is ambiguous which person to estimate the joint locations for, in addition to this most persons are also occluded. For the right image the errors are generally large with corresponding large estimated uncertainties.

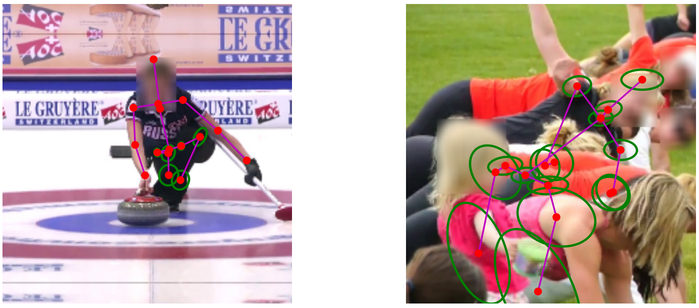

Figure 2: Qualitative illustration of results on MPII. We show the predictions of our method. For each joint we show the mean position estimate (red) and corresponding uncertainties (the green ellipses correspond to one standard deviation). The test images above are specifically chosen to highlight the output uncertainties and how they correspond to prediction quality.

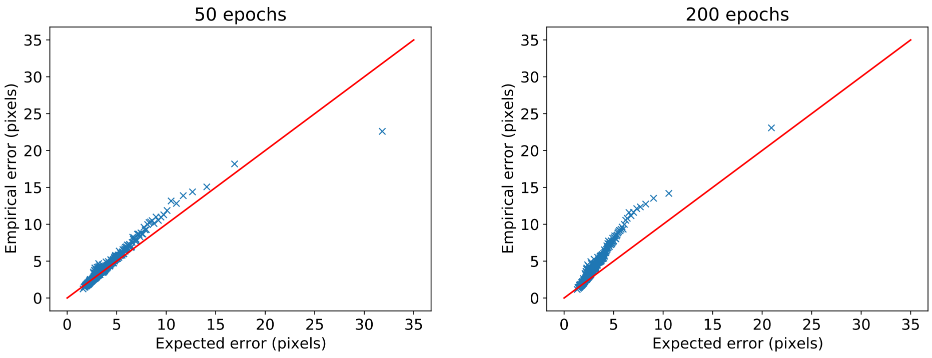

Figure 3: Relationship between empirical errors and expected errors based on the estimated precision matrix. Blue crosses correspond to the average empirical error compared to the average estimated error based on the variance of the output. The red line is the identity function. For a well calibrated method the blue crosses should be close to the red line.

6 Conclusion & Future work

In this paper we have presented a way to enforce the output of neural networks to be positive definite matrices. We have used this mapping to parameterize a unimodal probability distribution over $\mathbb{R}^N$. Furthermore we show that the NLL loss with respect to this parameterization has bounded gradients among other desirable properties.

We have evaluated this method on standard face and body keypoint regression datasets and conclude that our method outperform other pure regression methods, however, the state of the art methods for this type of data still remain heatmap based.

One potential direction of research could be to use the network to estimate δ parameter of the Huber distribution. Another direction could be to evaluate this method on higher dimensional data, such as regressing over 3d positions, an ubiquitous problem for real world applications.

References

[1] Spectrum of symmetrizable matrix. https://math.stackexchange.com/questions/578891/spectrum-of-symmetrizable-matrix. Accessed: 2020-05-21.

[2] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[3] Aleksandr Aravkin. *Robust methods for Kalman filtering/smoothing and bundle adjustment*. PhD thesis, University of Washington, 2010.

[4] J. T. Barron. A general and adaptive robust loss function. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[5] Yanrui Bin, Xuan Cao, Xinya Chen, Yanhao Ge, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, Changxin Gao, and Nong Sang. Adversarial semantic data augmentation for human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[6] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[7] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] Arnaud Dapogny, Kevin Bailly, and Matthieu Cord. DeCaFA: deep convolutional cascade for face alignment in the wild. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[10] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision (IJCV)*, 61(1):55–79, 2005.

[11] Zhen-Hua Feng, Josef Kittler, Muhammad Awais, Patrik Huber, and Xiao-Jun Wu. Wing loss for robust facial landmark localisation with convolutional neural networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] Ross Girshick. Fast r-cnn. In *Proceedings of the International Conference on Computer Vision (ICCV)*, December 2015.

[13] Nitesh B. Gundavarapu, Divyansh Srivastava, Rahul Mitra, Abhishek Sharma, and Arjun Jain. Structured aleatoric uncertainty in human pose estimation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[14] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[17] Harold W Kuhn. A note on fermat's problem. *Mathematical programming*, 4(1):98–107, 1973.

[18] Abhinav Kumar, Tim K Marks, Wenxuan Mou, Ye Wang, Michael Jones, Anoop Cherian, Toshiaki Koike-Akino, Xiaoming Liu, and Chen Feng. LUVLi face alignment: Estimating landmarks' location, uncertainty, and visibility likelihood. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[19] Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda, and Radu Horaud. A comprehensive analysis of deep regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 42(9):2065–2081, 2019.

[20] Katherine Liu, Kyel Ok, William Vega-Brown, and Nicholas Roy. Deep inference for covariance estimation: Learning gaussian noise models for state estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[21] David Mohlin, Gérald Bianchi, and Josephine Sullivan. Probabilistic Orientation Estimation with Matrix Fisher Distributions. In *Advances in Neural Information Processing Systems (NeurIPs)*, 2020.

[22] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[24] Shengju Qian, Keqiang Sun, Wayne Wu, Chen Qian, and Jiaya Jia. Aggregation via separation: Boosting facial landmark detector with semi-supervised style translation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[25] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[26] Nikolaos Sarafianos, Bogdan Boteanu, Bogdan Ionescu, and Ioannis A. Kakadiaris. 3d human pose estimation: A review of the literature and analysis of covariates. *Computer Vision and Image Understanding (CVIU)*, 2016.

[27] Zhihui Su, Ming Ye, Guohui Zhang, Lei Dai, and Jianda Sheng. Cascade feature aggregation for human pose estimation. *arXiv preprint arXiv:1902.07837*, 2019.

[28] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[29] Xiao Sun, Jiaxiang Shang, Shuang Liang, and Yichen Wei. Compositional human pose regression. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[30] Wei Tang, Pei Yu, and Ying Wu. Deeply learned compositional models for human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[31] Zhiqiang Tang, Xi Peng, Kang Li, and Dimitris N. Metaxas. Towards efficient U-Nets: A coupled and quantized approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 42(8):2038–2050, 2020.

[32] Jonathan Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in Neural Information Processing Systems (NeurIPs)*, 2014.

[33] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[34] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[35] Xinyao Wang, Liefeng Bo, and Li Fuxin. Adaptive wing loss for robust face alignment via heatmap regression. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[36] Wayne Wu, Chen Qian, Shuo Yang, Quan Wang, Yici Cai, and Qiang Zhou. Look at boundary: A boundary-aware face alignment algorithm. In *CVPR*, June 2018.

[37] Wenyan Wu and Shuo Yang. Leveraging intra and inter-dataset variations for robust face alignment. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017.

[38] Wei Yang, Shuang Li, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Learning feature pyramids for human pose estimation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[39] Shizhan Zhu, Cheng Li, Chen Change Loy, and Xiaoou Tang. Face alignment by coarse-to-fine shape searching. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[40] Xiangyu Zhu, Zhen Lei, Xiaoming Liu, Hailin Shi, and Stan Z Li. Face alignment across large poses: A 3d solution. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.