# Energy-Based Models for Deep Probabilistic Regression

## Supplementary Material

In this supplementary material, we provide additional details and results. It consists of Appendix A - Appendix F. Appendix A contains a detailed algorithm for our employed prediction procedure. Appendix B contains details on the illustrative 1D regression problem in Figure 2 in the main paper. Further details on the employed training and inference procedures are provided in Appendix C for the object detection experiments, and in Appendix D for the visual tracking experiments. Lastly, Appendix E and Appendix F contain details and full results for the experiments on age estimation and head-pose estimation, respectively. Note that equations, tables, figures and algorithms in this supplementary document are numbered with the prefix "S". Numbers without this prefix refer to the main paper.

## Appendix A Prediction Algorithm

Our prediction procedure (Section 3.3) is detailed in Algorithm S1, where $\lambda$ denotes the gradient ascent step-length, $\eta$ is a decay of the step-length and $T$ is the number of iterations. In our experiments, we fix $T$ (typically, $T = 10$) and select $\{\lambda, \eta\}$ using grid search on a validation set.

**Algorithm S1** Prediction via gradient-based refinement.

**Input:** $x^\star$, $\hat{y}$, $T$, $\lambda$, $\eta$.

$y \leftarrow \hat{y}$.
**for** $t = 1, \dots, T$ **do**
    `PrevValue` $\leftarrow f_\theta(x^\star, y)$.
    $\tilde{y} \leftarrow y + \lambda \nabla_y f_\theta(x^\star, y)$.
    `NewValue` $\leftarrow f_\theta(x^\star, \tilde{y})$.
    **if** `NewValue` $>$ `PrevValue` **then**
        $y \leftarrow \tilde{y}$.
    **else**
        $\lambda \leftarrow \eta\lambda$.
**Return** $y$.

## Appendix B Illustrative Example

The ground truth conditional target density $p(y|x)$ in Figure 2 is defined by a mixture of two Gaussian components (with weights 0.2 and 0.8) for $x < 0$, and

a log-normal distribution (with $\mu = 0.0$, $\sigma = 0.25$) for $x \geq 0$. The training data $\{(x_i, y_i)\}_{i=1}^{2000}$ was generated by uniform random sampling of $x$, $x_i \sim U(-3, 3)$. Both models were trained for 75 epochs with a batch size of 32 using the ADAM [14] optimizer.

The Gaussian model is defined using a DNN $f_\theta(x)$ according to,

$$\begin{aligned} p(y|x;\theta) &= \mathcal{N}\big(y; \mu_\theta(x),\, \sigma_\theta^2(x)\big), \\ f_\theta(x) &= [\, \mu_\theta(x) \;\; \log \sigma_\theta^2(x) \,]^\mathsf{T} \in \mathbb{R}^2. \end{aligned} \tag{S1}$$

It is trained by minimizing the negative log-likelihood, corresponding to the loss,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \frac{(y_i - \mu_\theta(x_i))^2}{\sigma_\theta^2(x_i)} + \log \sigma_\theta^2(x_i). \tag{S2}$$

The DNN $f_\theta$ is a simple feed-forward neural network, containing two shared fully-connected layers (dimensions: $1 \to 10$, $10 \to 10$) and two identical heads for $\mu$ and $\log \sigma^2$ of three fully-connected layers ($10 \to 10$, $10 \to 10$, $10 \to 1$).

Our proposed model $p(y|x;\theta) = e^{f_\theta(x,y)}/Z(x,\theta)$ (Eq. 1 in the paper) is defined using a feed-forward neural network $f_\theta(x, y)$ containing two fully-connected layers ($1 \to 10$, $10 \to 10$) for both $x$ and $y$, and three fully-connected layers ($20 \to 10$, $10 \to 10$, $10 \to 1$) processing the concatenated $(x, y)$ feature vector. It is trained using $M = 1024$ samples from a proposal distribution $q(y|y_i)$ (Eq. 5 in the paper) with $L = 2$ and variances $\sigma_1^2 = 0.1^2$, $\sigma_2^2 = 0.8^2$.

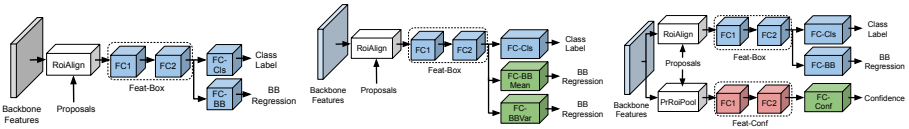
## Appendix C Object Detection

Here, we provide further details about the network architectures, training procedure, and hyperparameters used for our experiments on object detection (Section 4.1 in the paper).

### C.1 Network Architecture

We use the Faster-RCNN [21] detector with ResNet50-FPN [15] as our baseline. As visualized in Figure S1a, Faster-RCNN generates object proposals using a region proposal network (RPN). The features from the proposal regions are then pooled to a fixed-sized feature map using the RoiPool layer [7]. The pooled features are then passed through a feature extractor (denoted Feat-Box) consisting of two fully-connected (FC) layers. The output feature vector is then passed through two parallel FC layers, one which predicts the class label (denoted FC-Cls), and another which regresses the offsets between the proposal and the ground truth box (denoted FC-BB). We use the PyTorch implementation for Faster-RCNN from [17]. Note that we use the RoiAlign [9] layer instead of RoiPool in our experiments as it has been shown to achieve better performance [9].

For the Gaussian and Laplace probabilistic models (Gaussian and Laplace in Table 2 in the paper), we replace the FC-BB layer in Faster-RCNN with




(a) Faster-RCNN. (b) Laplace/Gaussian. (c) IoU-Net/Ours.

Fig. S1: Network architectures for the different object detection networks used in our experiments (Section 4.1 in the paper). The backbone feature extractor (ResNet50-FPN), and the region proposal network (RPN) is not shown for clarity. We do not train the blocks in blue color, using the pre-trained Faster-RCNN weights from [17] instead. The blocks in red are initialized with the pre-trained Faster-RCNN weights and fine-tuned. The blocks in green on the other hand are trained from scratch.

two parallel FC layers, denoted FC-BBMean and FC-BBVar, which predict the mean and the log-variance of the distribution modeling the offset between the proposal and the ground truth box for each coordinate. This architecture is shown in Figure S1b. For the mixtures of $K = \{2, 4, 8\}$ Gaussians, we duplicate FC-BBMean and FC-BBVar $K$ times, and add an FC layer for predicting the $K$ component weights. For the cVAE, FC-BBMean and FC-BBVar instead outputs the mean and log-variance of a Gaussian distribution for the latent variable $z \in \mathbb{R}^{10}$. Duplicates of FC-BBMean and FC-BBVar, modified to also take sampled $z$ values as input, then predicts the mean and log-variance of the distribution modeling the bounding box offset.

For our confidence-based IoU-Net [12] models (IoU-Net and IoU-Net* in Table 2), we use the same network architecture as employed in the original paper, shown in Figure S1c. That is, we add an additional branch to predict the IoU overlap between the proposal box and the ground truth. This branch uses the PrRoiPool [12] layer to pool the features from the proposal regions. The pooled features are passed through a feature extractor (denoted Feat-Conf) consisting of two FC layers. The output feature vector is passed through another FC layer, FC-Conf, which predicts the IoU. We use an identical architecture for our approach, but train it to output $f_\theta(x, y)$ in $p(y|x; \theta) = e^{f_\theta(x,y)}/Z(x, \theta)$ instead.

### C.2 Training

We use the pre-trained weights for Faster-RCNN from [17]. Note that the bounding box regression in Faster-RCNN is trained using a direct method, with an Huber loss [11]. We trained the other networks in Table 2 in the paper (Gaussian, Gaussian Mixt. 2, Gaussian Mixt. 4, Gaussian Mixt. 8, Gaussian cVAE, Laplace, IoU-Net, IoU-Net* and Ours) on the MS-COCO [16] training split (*2017 train*) using stochastic gradient descent (SGD) with a batch size of 16 for 60k iterations. The base learning rate $lr_{\text{base}}$ is reduced by a factor of 10 after 40k and 50k iterations, for all the networks. We also warm up the training by linearly

increasing the learning rate from $\frac{1}{3}lr_{\text{base}}$ to $lr_{\text{base}}$ during the first 500 iterations. We use a weight decay of 0.0001 and a momentum of 0.9. For all the networks, we only trained the newly added layers, while keeping the backbone and the region proposal network fixed.

For the Gaussian, mixture of Gaussians, cVAE and Laplace models, we only train the final predictors (FC-BBMean and FC-BBVar), while keeping the class predictor (FC-Cls) and the box feature extractor (Feat-Box) fixed. We also tried fine-tuning the FC-Cls and Feat-Box weights for the Gaussian model, with different learning rate settings, but obtained worse performance on the validation set. The weights for both FC-BBMean and FC-BBVar were initialized with zero mean Gaussian with standard deviation of 0.001. All these models were trained with a base learning rate $lr_{\text{base}} = 0.005$ by minimizing the negative log-likelihood, except for the cVAE which is trained by maximizing the ELBO (using 128 sampled $z$ values to approximate the expectation).

For the IoU-Net, IoU-Net* and our proposed model, we only trained the newly added confidence branch. We found it beneficial to initialize the feature extractor block (Feat-Conf) with the corresponding weights from Faster-RCNN, i.e. the Feat-Box block. The weights for the predictor FC-Conf were initialized with zero mean Gaussian with standard deviation of 0.001. As in the original paper [12], we used a base learning rate $lr_{\text{base}} = 0.01$ for the IoU-Net and IoU-Net* networks. For our proposed model, we used $lr_{\text{base}} = 0.001$ due to the different scaling of the loss. Note that we did not perform any parameter tuning for setting the learning rates. We generate 128 proposals for each ground truth box during training. For the IoU-Net, we use the proposal generation strategy mentioned in the original paper [12]. That is, for each ground truth box, we generate a large set of candidate boxes which have an IoU overlap of at least 0.5 with the ground truth, and uniformly sample 128 proposals from this candidate set w.r.t. the IoU. For IoU-Net* and our model, we sample boxes from a proposal distribution (Eq. 5 in the paper) generated by $L = 3$ Gaussians with standard deviations of 0.0375, 0.075 and 0.15. The IoU-Net and IoU-Net* are trained by minimizing the Huber loss between the predicted IoU and the ground truth, while our model is trained by minimizing the negative log likelihood of the training data (Eq. 4 in the paper).

### C.3 Analysis of Proposal Distribution

An extensive ablation study for the number of components $L$ and standard deviations $\{\sigma_l\}_{l=1}^{L}$ in the proposal distribution $q(y|y_i) = \frac{1}{L}\sum_{l=1}^{L}\mathcal{N}(y; y_i, \sigma_l^2)$ (Eq. 5 in the paper) is provided in Table S1, which is an extended version of Table 1 in the paper. We find that $L = 1$ downgrades performance, while there is no significant difference between $L = 2$ and $L = 3$. For $L \in \{2, 3\}$, the results are not particularly sensitive to the specific choice of $\{\sigma_l\}_{l=1}^{L}$, but benefits from including *both* small and relatively large values in $\{\sigma_l\}_{l=1}^{L}$.

Table S1: Impact of $L$ and $\{\sigma_l\}_{l=1}^{L}$ in the proposal distribution $q(y|y_i)$ (Eq. 5 in the paper), for the object detection task on the *2017 val* split of COCO [16].

| $L$ | $\{\sigma_l\}_{l=1}^{L}$ | AP (%) |
|---|---|---|
| 1 | {0.01875} | 38.07 |
| 1 | {0.0375} | 38.47 |
| 1 | {0.075} | 37.52 |
| 1 | {0.15} | 35.05 |
| 2 | {0.025, 0.1} | 38.97 |
| 2 | {0.0375, 0.15} | 39.05 |
| 2 | {0.04375, 0.175} | 39.07 |
| 2 | {0.05, 0.2} | 39.02 |
| 2 | {0.0125, 0.025} | 38.19 |
| 2 | {0.025, 0.05} | 38.65 |
| 2 | {0.075, 0.15} | 37.14 |
| 3 | {0.0125, 0.025, 0.05} | 38.61 |
| 3 | {0.025, 0.05, 0.1} | 38.95 |
| 3 | {0.0375, 0.075, 0.15} | **39.11** |
| 3 | {0.04375, 0.0875, 0.175} | 39.00 |
| 3 | {0.05, 0.1, 0.2} | 38.76 |
| 3 | {0.0625, 0.125, 0.25} | 37.96 |
| 3 | {0.075, 0.15, 0.3} | 37.42 |

### C.4 Inference

The inference in both the Gaussian and Laplace models is identical to the one employed by Faster-RCNN, except the output mean is taken as the prediction. Thus, we do not utilize the output variances during inference. For the mixture of $K = \{2, 4, 8\}$ Gaussians, we compute the mean of the distribution and take that as our prediction. Instead picking the component with the largest weight and taking its mean as the prediction resulted in somewhat worse validation performance. For cVAE, we approximately compute the mean (using 128 samples) of the distribution and take that as our prediction.

For IoU-Net and IoU-Net*, we perform IoU-Guided NMS as described in [12], followed by gradient-based refinement (Algorithm S1). For our proposed approach we adopt the same NMS technique, but guide it with the values $f_\theta(x, y)$ predicted by our network instead. We use a step-length $\lambda = 0.5$ and step-length decay $\eta = 0.1$ for IoU-Net. For IoU-Net* and our approach we perform the gradient-based refinement in the relative bounding box parametrization $y = (c_x/w_0, c_y/h_0, \log w, \log h)$ (see Section 4.1 in the paper). Here, we employ different step-lengths for position and size. For IoU-Net*, we use $\lambda = 0.002$ and $\lambda = 0.008$ respectively, with a decay of $\eta = 0.2$. For our proposed approach, we use $\lambda = 0.0001$ and $\lambda = 0.0004$ with $\eta = 0.5$. For all methods, these hyperparameters ($\lambda$ and $\eta$) were set using a grid search on the COCO validation split (*2017 val*). We used $T = 10$ refinement iterations for each of the three models. Note that since a given image $x$ can have multiple ground truth instances, mul-

Table S2: Impact of $L$ and $\{\sigma_l\}_{l=1}^{L}$ in the proposal distribution $q(y|y_i)$ (Eq. 5 in the paper), for the visual tracking task on the combined OTB [22] and NFS [13] datasets.

| $L$ | $\{\sigma_l\}_{l=1}^{L}$ | $\text{OP}_{0.50}$ (%) | $\text{OP}_{0.75}$ (%) | AUC (%) |
|---|---|---|---|---|
| 1 | {0.05} | 75.77 | 45.72 | 63.37 |
| 2 | {0.01, 0.1} | 77.25 | 46.09 | 61.48 |
| 2 | {0.03, 0.3} | 79.27 | 48.59 | 63.65 |
| 2 | {0.05, 0.5} | **79.90** | **48.71** | **64.10** |
| 2 | {0.07, 0.7} | 78.41 | 47.72 | 62.75 |

tiple bounding boxes are usually refined. The gradient-based refinement then moves each individual box $y$ towards the maximum of a *local* mode in $f_\theta(x, y)$. Thus, they will not converge to a single solution. Also note that $f_\theta(x, y)$ is class-conditional (as in the IoU-Net baseline), eliminating the risk of confusing neighboring objects of different classes.

## Appendix D Visual Tracking

Here, we provide further details about the training procedure and hyperparameters used for our experiments on visual object tracking (Section 4.2 in the paper).

### D.1 Training

We adopt the ATOM [3] tracker as our baseline, and use the PyTorch implementation and pre-trained weights from [2]. ATOM trains an IoU-Net-based module to predict the IoU overlap between a candidate box and the ground truth, conditioned on the first-frame target appearance. The IoU predictor is trained by generating 16 candidates for each ground truth box. The candidates are generated by adding a Gaussian noise for each ground truth box coordinate, while ensuring a minimum IoU overlap of 0.1 between the candidate box and the ground truth. The network is trained by minimizing the squared error ($L^2$ loss) between the predicted and ground truth IoU.

Our proposed model is instead trained by sampling 128 candidate boxes from a proposal distribution (Eq. 5 in the paper) generated by $L = 2$ Gaussians with standard deviations of 0.05 and 0.5, and minimizing the negative log likelihood of the training data. An ablation study for the proposal distribution is found in Table S2. We use the training splits of the TrackingNet [18], LaSOT [4], GOT10k [10], and COCO datasets for our training. Our network is trained for 50 epochs, using the ADAM optimizer with a base learning rate of 0.001 which is reduced by a factor of 5 after every 15 epochs. The rest of the training parameters are exactly the same is in ATOM. The ATOM* model is trained by using the exact same proposal distribution, datasets and settings. It only differs by the loss, which is the same squared error between the predicted and ground truth IoU as in the original ATOM.

Fig. S2: Visualization of the conditional target density $p(y|x;\theta) \propto e^{f_\theta(x,y)}$ predicted by our network for the task of bounding box estimation in visual tracking. Since the target space $y \in \mathbb{R}^4$ is 4-dimensional, we visualize the density for different locations of the top-right corner as a heatmap, while the bottom-left is kept fixed at the tracker output (red box). Our network predicts flexible densities which qualitatively capture meaningful uncertainty in challenging cases.

### D.2 Inference

During tracking, the ATOM tracker first applies the classification head network, which is trained online, to coarsely localize the target object. 10 random boxes are then sampled around this prediction, to be refined by the IoU prediction network. We only alter the final bounding box refinement step of the 10 given random initial boxes, and preserve all other settings as in the original ATOM tracker. The original version performs $T = 5$ gradient ascent iterations with a step length of $\lambda = 1.0$. For our proposed model and the ATOM* version, we use $T = 10$ iterations, employing the bounding box parameterization described in Section 4.1. For our approach, we set the step length to $\lambda = 2 \cdot 10^{-4}$ for position and $\lambda = 10^{-3}$ for size dimensions. For ATOM*, we use $\lambda = 10^{-2}$ for position and $\lambda = 5 \cdot 10^{-2}$ for size dimensions. These parameters were set on the separate validation set. For simplicity, we adopt the vanilla gradient ascent strategy employed in ATOM for the two other methods as well. That is, we have no decay ($\eta = 1$) and do not perform checks whether the confidence score is increasing in each iteration.

### D.3 Qualitative Results

Illustrative examples of the target density $p(y|x;\theta) \propto e^{f_\theta(x,y)}$ predicted by our approach during tracking are visualized in Figure S2.

## Appendix E Age Estimation

In this appendix, further details on the age estimation experiments (Section 4.3 in the paper) are provided.

### E.1 Network Architecture

The DNN architecture $f_\theta(x, y)$ of our proposed model first extracts ResNet50 features $g_x \in \mathbb{R}^{2048}$ from the input image $x$. The age $y$ is processed by four fully-connected layers (dimensions: $1 \to 16$, $16 \to 32$, $32 \to 64$, $64 \to 128$), generating

Table S3: Impact of $\{\sigma_l\}_{l=1}^2$ in the proposal distribution $q(y|y_i)$ (Eq. 5 in the paper), for the age estimation task on our validation split of the UTKFace [25] dataset.

| $\{\sigma_l\}_{l=1}^2$ | MAE |
|---|---|
| {0.1, 10} | 7.62 |
| {0.1, 20} | **5.12** |
| {0.01, 20} | 5.36 |
| {0.1, 40} | 5.24 |

$g_y \in \mathbb{R}^{128}$. The two feature vectors $g_x$, $g_y$ are then concatenated to form $g_{x,y} \in \mathbb{R}^{2048+128}$, which is processed by two fully-connected layers ($2048+128 \rightarrow 2048$, $2048 \rightarrow 1$), outputting $f_\theta(x, y) \in \mathbb{R}$.

### E.2 Training

Our model is trained using $M = 1024$ samples from a proposal distribution $q(y|y_i)$ (Eq. 5 in the paper) with $L = 2$ and variances $\sigma_1^2 = 0.1^2$, $\sigma_2^2 = 20^2$. An ablation study for the variances is found in Table S3. The model is trained for 75 epochs with a batch size of 32, using the ADAM optimizer with weight decay of 0.001. The images $x$ are of size $200 \times 200$. For data augmentation, we use random flipping along the vertical axis and random scaling in the range $[0.7, 1.4]$. After random flipping and scaling, a random image crop of size $200 \times 200$ is also selected. The ResNet50 is imported from `torchvision.models` in PyTorch with the pretrained option set to true, all other network parameters are randomly initialized using the default initializer in PyTorch.

### E.3 Prediction

For this experiment, we use a slight variation of Algorithm S1, which is found in Algorithm S2. There, $T$ is the number of gradient ascent iterations, $\lambda$ is the stepsize, $\Omega_1$ is an early-stopping threshold and $\Omega_2$ is a degeneration tolerance. Following IoU-Net, we set $T = 5$, $\Omega_1 = 0.001$ and $\Omega_2 = -0.01$. Based on the validation set, we select $\lambda = 3$. We refine a single estimate $\hat{y}$, predicted by each baseline model.

### E.4 Baselines

All baselines are trained for 75 epochs with a batch size of 32, using the ADAM optimizer with weight decay of 0.001. Identical data augmentation and parameter initialization as for our proposed model is used.

**Direct** The DNN architecture of *Direct* first extracts ResNet50 features $g_x \in \mathbb{R}^{2048}$ from the input image $x$. The feature vector $g_x$ is then processed by two fully-connected layers ($2048 \rightarrow 2048$, $2048 \rightarrow 1$), outputting the prediction $\hat{y} \in \mathbb{R}$. It is trained by minimizing either the Huber or $L^2$ loss.

**Algorithm S2** Prediction via gradient-based refinement (variation).

**Input:** $x^\star$, $\hat{y}$, $T$, $\lambda$, $\Omega_1$, $\Omega_2$.

$y \leftarrow \hat{y}$.
**for** $t = 1, \ldots, T$ **do**
  $\texttt{PrevValue} \leftarrow f_\theta(x^\star, y)$.
  $y \leftarrow y + \lambda \nabla_y f_\theta(x^\star, y)$.
  $\texttt{NewValue} \leftarrow f_\theta(x^\star, y)$.
  **if** $|\texttt{PrevValue} - \texttt{NewValue}| < \Omega_1$ **or** $(\texttt{NewValue} - \texttt{PrevValue}) < \Omega_2$ **then**
    **Return** $y$.
**Return** $y$.

Table S4: Full results for the age estimation experiments. Gradient-based refinement using our proposed method consistently improves MAE (lower is better) for the age predictions outputted by a number of baselines.

| Method | MAE |
|---|---|
| Niu et al. [19] | 5.74 ± 0.05 |
| Cao et al. [1] | 5.47 ± 0.01 |
| Direct - Huber | 4.80 ± 0.06 |
| Direct - Huber + Refinement | 4.74 ± 0.06 |
| Direct - L2 | 4.81 ± 0.02 |
| Direct - L2 + Refinement | **4.65** ± 0.02 |
| Gaussian | 4.79 ± 0.06 |
| Gaussian + Refinement | 4.66 ± 0.04 |
| Laplace | 4.85 ± 0.04 |
| Laplace + Refinement | 4.81 ± 0.04 |
| Softmax - CE & L2 | 4.78 ± 0.05 |
| Softmax - CE & L2 + Refinement | **4.65** ± 0.04 |
| Softmax - CE, L2 & Var | 4.81 ± 0.03 |
| Softmax - CE, L2 & Var + Refinement | 4.69 ± 0.03 |

**Gaussian** The Gaussian model is defined using a DNN $f_\theta(x)$ according to,

$$\begin{aligned} p(y|x;\theta) &= \mathcal{N}\big(y; \mu_\theta(x),\, \sigma_\theta^2(x)\big), \\ f_\theta(x) &= [\, \mu_\theta(x) \,\log \sigma_\theta^2(x)\,]^\mathsf{T} \in \mathbb{R}^2. \end{aligned} \tag{S3}$$

It is trained by minimizing the negative log-likelihood, corresponding to the loss,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \frac{(y_i - \mu_\theta(x_i))^2}{\sigma_\theta^2(x_i)} + \log \sigma_\theta^2(x_i). \tag{S4}$$

The DNN architecture of $f_\theta(x)$ first extracts ResNet50 features $g_x \in \mathbb{R}^{2048}$ from the input image $x$. The feature vector $g_x$ is then processed by two heads of two fully-connected layers ($2048 \rightarrow 2048$, $2048 \rightarrow 1$) to output $\mu_\theta(x)$ and $\log \sigma_\theta^2(x)$. The mean $\mu_\theta(x)$ is taken as the prediction $\hat{y}$.

**Laplace** The Laplace model is defined using a DNN $f_\theta(x)$ according to,

$$p(y|x;\theta) = \frac{1}{2\beta_\theta(x)} \exp\Big\{ - \frac{|y - \mu_\theta(x)|}{\beta_\theta(x)} \Big\}, \\ f_\theta(x) = [\, \mu_\theta(x) \, \log \beta_\theta(x) \,]^\mathsf{T} \in \mathbb{R}^2. \tag{S5}$$

It is trained by minimizing the negative log-likelihood, corresponding to the loss,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \mu_\theta(x_i)|}{\beta_\theta(x_i)} + \log \beta_\theta(x_i). \tag{S6}$$

The DNN architecture of $f_\theta(x)$ first extracts ResNet50 features $g_x \in \mathbb{R}^{2048}$ from the input image $x$. The feature vector $g_x$ is then processed by two heads of two fully-connected layers ($2048 \to 2048$, $2048 \to 1$) to output $\mu_\theta(x)$ and $\log \beta_\theta(x)$. The mean $\mu_\theta(x)$ is taken as the prediction $\hat{y}$.

**Softmax** The DNN architecture of *Softmax* first extracts ResNet50 features $g_x \in \mathbb{R}^{2048}$ from the input image $x$. The feature vector $g_x$ is then processed by two fully-connected layers ($2048 \to 2048$, $2048 \to C$), outputting logits for $C = 101$ discretized classes $\{0, 1, \dots, 100\}$. It is trained by minimizing either the cross-entropy (CE) and $L^2$ losses, $J = J_{CE} + 0.1 J_{L^2}$, or the CE, $L^2$ and variance [20] losses, $J = J_{CE} + 0.1 J_{L^2} + 0.05 J_{Var}$. The prediction $\hat{y}$ is computed as the softmax expected value.

### E.5 Full Results

Full experiment results, extending the results found in Table 4 (Section 4.3 in the paper), are provided in Table S4.

## Appendix F Head-Pose Estimation

In this appendix, further details on the head-pose estimation experiments (Section 4.4 in the paper) are provided.

### F.1 Network Architecture

The DNN architecture $f_\theta(x, y)$ of our proposed model first extracts ResNet50 features $g_x \in \mathbb{R}^{2048}$ from the input image $x$. The pose $y \in \mathbb{R}^3$ is processed by four fully-connected layers (dimensions: $3 \to 16$, $16 \to 32$, $32 \to 64$, $64 \to 128$), generating $g_y \in \mathbb{R}^{128}$. The two feature vectors $g_x$, $g_y$ are then concatenated to form $g_{x,y} \in \mathbb{R}^{2048+128}$, which is processed by two fully-connected layers ($2048 + 128 \to 2048$, $2048 \to 1$), outputting $f_\theta(x, y) \in \mathbb{R}$.

Table S5: Impact of $\{\sigma_l\}_{l=1}^2$ in the proposal distribution $q(y|y_i)$ (Eq. 5 in the paper), for the head-pose estimation task on our validation split of the BIWI [5] dataset.

| $\{\sigma_l\}_{l=1}^2$ | Average MAE |
|---|---|
| {0.1, 20} | 6.96 |
| {1, 20} | **5.08** |
| {1, 30} | 5.24 |
| {2, 20} | 7.02 |
| {1, 10} | 7.56 |

### F.2 Training

Our model is trained using $M = 1024$ samples from a proposal distribution $q(y|y_i)$ (Eq. 5 in the paper) with $L = 2$ and variances $\sigma_1^2 = 1^2$, $\sigma_2^2 = 20^2$ for yaw, pitch and roll. An ablation study for the variances is found in Table S5. The model is trained for 75 epochs with a batch size of 32, using the ADAM optimizer with weight decay of 0.001. The images $x$ are of size $64 \times 64$. For data augmentation, we use random flipping along the vertical axis and random scaling in the range $[0.7, 1.4]$. After random flipping and scaling, a random image crop of size $64 \times 64$ is also selected. The ResNet50 is imported from `torchvision.models` in PyTorch with the pretrained option set to true, all other network parameters are randomly initialized using the default initializer in PyTorch.

### F.3 Prediction

For this experiment, we also use the prediction procedure detailed in Algorithm S2. Again following IoU-Net, we set $T = 5$, $\Omega_1 = 0.001$ and $\Omega_2 = -0.01$. Based on the validation set, we select $\lambda = 0.1$. We refine a single estimate $\hat{y}$, predicted by each baseline model.

### F.4 Baselines

All baselines are trained for 75 epochs with a batch size of 32, using the ADAM optimizer with weight decay of 0.001. Identical data augmentation and parameter initialization as for our proposed model is used.

**Direct** The DNN architecture of *Direct* first extracts ResNet50 features $g_x \in \mathbb{R}^{2048}$ from the input image $x$. The feature vector $g_x$ is then processed by two fully-connected layers ($2048 \rightarrow 2048$, $2048 \rightarrow 3$), outputting the prediction $\hat{y} \in \mathbb{R}^3$. It is trained by minimizing either the Huber or $L^2$ loss.

Table S6: Full results for the head-pose estimation experiments. Gradient-based refinement using our proposed method consistently improves the average MAE for yaw, pitch, roll (lower is better) for the predicted poses outputted by a number of baselines.

| Method | Yaw MAE | Pitch MAE | Roll MAE | Avg. MAE |
|---|---|---|---|---|
| Yang et al. [24] | 4.24 | 4.35 | 4.19 | 4.26 |
| Gu et al. [8] | 3.91 | 4.03 | 3.03 | 3.66 |
| Yang et al. [23] | 2.89 | 4.29 | 3.60 | 3.60 |
| Direct - Huber | 2.78 ± 0.09 | 3.73 ± 0.13 | 2.90 ± 0.09 | 3.14 ± 0.07 |
| Direct - Huber + Refine. | 2.75 ± 0.08 | 3.70 ± 0.11 | 2.87 ± 0.09 | 3.11 ± 0.06 |
| Direct - L2 | 2.81 ± 0.08 | **3.60** ± 0.14 | 2.85 ± 0.08 | 3.09 ± 0.07 |
| Direct - L2 + Refine. | 2.78 ± 0.08 | 3.62 ± 0.13 | 2.81 ± 0.08 | 3.07 ± 0.07 |
| Gaussian | 2.89 ± 0.09 | 3.64 ± 0.13 | 2.83 ± 0.09 | 3.12 ± 0.08 |
| Gaussian + Refine. | 2.84 ± 0.08 | 3.67 ± 0.12 | 2.81 ± 0.08 | 3.11 ± 0.07 |
| Laplace | 2.93 ± 0.08 | 3.80 ± 0.15 | 2.90 ± 0.07 | 3.21 ± 0.06 |
| Laplace + Refine. | 2.89 ± 0.07 | 3.81 ± 0.13 | 2.88 ± 0.06 | 3.19 ± 0.06 |
| Softmax - CE & L2 | 2.73 ± 0.09 | 3.63 ± 0.13 | 2.77 ± 0.11 | 3.04 ± 0.08 |
| Softmax - CE & L2 + Refine. | **2.67** ± 0.08 | 3.61 ± 0.12 | **2.75** ± 0.10 | **3.01** ± 0.07 |
| Softmax - CE, L2 & Var | 2.83 ± 0.12 | 3.79 ± 0.10 | 2.84 ± 0.11 | 3.15 ± 0.07 |
| Softmax - CE, L2 & Var + Refine. | 2.76 ± 0.10 | 3.74 ± 0.09 | 2.83 ± 0.10 | 3.11 ± 0.06 |

**Gaussian** The Gaussian model is defined using a DNN $f_\theta(x)$ according to,

$$
\begin{aligned}
p(y|x;\theta) = \mathcal{N}\big(y; \mu_\theta(x), \Sigma_\theta(x)\big), \quad \Sigma_\theta(x) = \operatorname{diag}\big(\sigma^2_\theta(x)\big), \\
y = [\, y_1 \; y_2 \; y_3 \,]^\mathsf{T} \in \mathbb{R}^3, \\
\mu_\theta(x) = [\, \mu_{1,\theta}(x) \; \mu_{2,\theta}(x) \; \mu_{3,\theta}(x) \,]^\mathsf{T} \in \mathbb{R}^3, \\
\sigma^2_\theta(x) = [\, \sigma^2_{1,\theta}(x) \; \sigma^2_{2,\theta}(x) \; \sigma^2_{3,\theta}(x) \,]^\mathsf{T} \in \mathbb{R}^3, \\
f_\theta(x) = [\, \mu_\theta(x)^\mathsf{T} \; \log \sigma^2_\theta(x)^\mathsf{T} \,]^\mathsf{T} \in \mathbb{R}^6.
\end{aligned}
\tag{S7}
$$

It is trained by minimizing the negative log-likelihood, corresponding to the loss,

$$
J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left( \sum_{k=1}^{3} \frac{(y_{k,i} - \mu_{k,\theta}(x_i))^2}{\sigma^2_{k,\theta}(x_i)} + \log \sigma^2_{k,\theta}(x_i) \right). \tag{S8}
$$

The DNN architecture of $f_\theta(x)$ first extracts ResNet50 features $g_x \in \mathbb{R}^{2048}$ from the input image $x$. The feature vector $g_x$ is then processed by two heads of two fully-connected layers ($2048 \to 2048$, $2048 \to 3$) to output $\mu_\theta(x) \in \mathbb{R}^3$ and $\log \sigma^2_\theta(x) \in \mathbb{R}^3$. The mean $\mu_\theta(x)$ is taken as the prediction $\hat{y}$.

**Laplace** Following [6], the Laplace model is defined using a DNN $f_\theta(x)$ according to,

$$
\begin{aligned}
p(y|x;\theta) = \prod_{k=1}^{3} \beta_{k,\theta}(x)^{-\frac{1}{2}} \exp\Big\{ -\frac{1}{2} \Big( \sum_{k=1}^{3} \frac{(y_k - \mu_{k,\theta}(x))^2}{\beta_{k,\theta}(x)} \Big)^{\frac{1}{2}} \Big\}, \\
y = [\, y_1 \; y_2 \; y_3 \,]^\mathsf{T} \in \mathbb{R}^3, \\
\mu_\theta(x) = [\, \mu_{1,\theta}(x) \; \mu_{2,\theta}(x) \; \mu_{3,\theta}(x) \,]^\mathsf{T} \in \mathbb{R}^3, \\
\beta_\theta(x) = [\, \beta_{1,\theta}(x) \; \beta_{2,\theta}(x) \; \beta_{3,\theta}(x) \,]^\mathsf{T} \in \mathbb{R}^3, \\
f_\theta(x) = [\, \mu_\theta(x)^\mathsf{T} \; \log \beta_\theta(x)^\mathsf{T} \,]^\mathsf{T} \in \mathbb{R}^6.
\end{aligned}
\tag{S9}
$$

It is trained by minimizing the negative log-likelihood, corresponding to the loss,

$$
J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \Big\{ \Big( \sum_{k=1}^{3} \frac{(y_{k,i} - \mu_{k,\theta}(x_i))^2}{\beta_{k,\theta}(x_i)} \Big)^{\frac{1}{2}} + \sum_{k=1}^{3} \log \beta_{k,\theta}(x_i) \Big\}. \tag{S10}
$$

The DNN architecture of $f_\theta(x)$ first extracts ResNet50 features $g_x \in \mathbb{R}^{2048}$ from the input image $x$. The feature vector $g_x$ is then processed by two heads of two fully-connected layers ($2048 \to 2048$, $2048 \to 3$) to output $\mu_\theta(x) \in \mathbb{R}^3$ and $\log \beta_\theta(x) \in \mathbb{R}^3$. The mean $\mu_\theta(x)$ is taken as the prediction $\hat{y}$.

**Softmax** The DNN architecture of *Softmax* first extracts ResNet50 features $g_x \in \mathbb{R}^{2048}$ from the input image $x$. The feature vector $g_x$ is then processed by three heads of two fully-connected layers ($2048 \to 2048$, $2048 \to C$), outputting logits for $C = 151$ discretized classes $\{-75, -74, \dots, 75\}$ for the yaw, pitch and roll angles (in degrees). It is trained by minimizing either the cross-entropy (CE) and $L^2$ losses, $J = J_{CE} + 0.1 J_{L^2}$, or the CE, $L^2$ and variance [20] losses, $J = J_{CE} + 0.1 J_{L^2} + 0.05 J_{Var}$. The prediction $\hat{y}$ is obtained by computing the softmax expected value for yaw, pitch and roll.

### F.5 Full Results

Full experiment results, extending the results found in Table 5 (Section 4.4 in the paper), are provided in Table S6.

## References


1. Cao, W., Mirjalili, V., Raschka, S.: Rank-consistent ordinal regression for neural networks. arXiv preprint arXiv:1901.07884 (2019) 9
2. Danelljan, M., Bhat, G.: PyTracking: Visual tracking library based on PyTorch. https://github.com/visionml/pytracking (2019), accessed: 30/04/2020 6
3. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: ATOM: Accurate tracking by overlap maximization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4660–4669 (2019) 6

4. Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., Ling, H.: LaSOT: A high-quality benchmark for large-scale single object tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5374–5383 (2019) 6
5. Fanelli, G., Dantone, M., Gall, J., Fossati, A., Van Gool, L.: Random forests for real time 3d face analysis. International Journal of Computer Vision (IJCV) **101**(3), 437–458 (2013) 11
6. Gast, J., Roth, S.: Lightweight probabilistic deep networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3369–3378 (2018) 13
7. Girshick, R.B.: Fast R-CNN. Proceedings of the IEEE International Conference on Computer Vision (ICCV) pp. 1440–1448 (2015) 2
8. Gu, J., Yang, X., De Mello, S., Kautz, J.: Dynamic facial analysis: From bayesian filtering to recurrent neural network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1548–1557 (2017) 12
9. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. Proceedings of the IEEE International Conference on Computer Vision (ICCV) pp. 2980–2988 (2017) 2
10. Huang, L., Zhao, X., Huang, K.: GOT-10k: A large high-diversity benchmark for generic object tracking in the wild. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2019) 6
11. Huber, P.J.: Robust estimation of a location parameter. The Annals of Mathematical Statistics pp. 73–101 (1964) 3
12. Jiang, B., Luo, R., Mao, J., Xiao, T., Jiang, Y.: Acquisition of localization confidence for accurate object detection. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 784–799 (2018) 3, 4, 5
13. Kiani Galoogahi, H., Fagg, A., Huang, C., Ramanan, D., Lucey, S.: Need for speed: A benchmark for higher frame rate object tracking. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 1125–1134 (2017) 6
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 2
15. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2117–2125 (2017) 2
16. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 740–755 (2014) 3, 5
17. Massa, F., Girshick, R.: maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark (2018), accessed: 04/09/2019 2, 3
18. Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., Ghanem, B.: TrackingNet: A large-scale dataset and benchmark for object tracking in the wild. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 300–317 (2018) 6
19. Niu, Z., Zhou, M., Wang, L., Gao, X., Hua, G.: Ordinal regression with multiple output cnn for age estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4920–4928 (2016) 9
20. Pan, H., Han, H., Shan, S., Chen, X.: Mean-variance loss for deep age estimation from a face. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5285–5294 (2018) 10, 13

21. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **39**, 1137–1149 (2015) 2
22. Wu, Y., Lim, J., Yang, M.H.: Object tracking benchmark. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **37**(9), 1834–1848 (2015) 6
23. Yang, T.Y., Chen, Y.T., Lin, Y.Y., Chuang, Y.Y.: FSA-Net: Learning fine-grained structure aggregation for head pose estimation from a single image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1087–1096 (2019) 12
24. Yang, T.Y., Huang, Y.H., Lin, Y.Y., Hsiu, P.C., Chuang, Y.Y.: SSR-Net: A compact soft stagewise regression network for age estimation. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) (2018) 12
25. Zhang, Z., Song, Y., Qi, H.: Age progression/regression by conditional adversarial autoencoder. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5810–5818 (2017), https://susanqq.github.io/UTKFace/ 8