DCTD: Deep Conditional Target Densities for Accurate Regression

Anonymous authors
Paper under double-blind review

Abstract

While deep learning-based classification is generally addressed using standardized approaches, a wide variety of techniques are employed for regression. In computer vision, one particularly popular such technique is that of confidence-based regression, which entails predicting a confidence value for each input-target pair (x, y). While this approach has demonstrated impressive results, it requires important task-dependent design choices, and the predicted confidences often lack a natural probabilistic meaning. We address these issues by proposing Deep Conditional Target Densities (DCTD), a novel and general regression method with a clear probabilistic interpretation. DCTD models the conditional target density $p(y|x)$ by using a neural network to directly predict the un-normalized density from (x, y). This model of $p(y|x)$ is trained by minimizing the associated negative log-likelihood, approximated using Monte Carlo sampling. We perform comprehensive experiments on four computer vision regression tasks. Our approach outperforms direct regression, as well as other probabilistic and confidence-based methods. Notably, our regression model achieves a 1.9% AP improvement over Faster-RCNN for object detection on the COCO dataset, and sets a new state-of-the-art on visual tracking when applied for bounding box regression.

1 Introduction

Supervised regression entails learning a model capable of predicting a continuous target value y from an input x, given a set of paired training examples. It is a fundamental machine learning problem with many important applications within computer vision and other domains. Common regression tasks within computer vision include object detection (Jiang et al., 2018; Zhou et al., 2019), head- and body-pose estimation (Xiao et al., 2018; Yang et al., 2019), age estimation (Rothe et al., 2016; Pan et al., 2018), visual tracking (Danelljan et al., 2019) and medical image registration (Niethammer et al., 2011; Chou et al., 2013), just to mention a few. While all of these tasks benefit from accurate regression of the target values, high accuracy can even be safety-critical in e.g. automotive and medical applications. Today, such regression problems are commonly tackled using Deep Neural Networks (DNNs), due to their ability to learn powerful feature representations from data.

While classification is generally addressed using standardized losses and output representations, a wide variety of different techniques are employed for regression. The most conventional strategy is to train a DNN to directly predict a target y given an input x (Lathuilière et al., 2019). In such *direct regression* approaches, the model parameters of the DNN are learned by minimizing a loss function, for example the L^2 or L^1 loss, penalizing the discrepancy between the predicted and ground truth target values. From a probabilistic perspective, this approach corresponds to creating a simple parametric model of the conditional target density $p(y|x)$, and minimizing the associated negative log-likelihood. The L^2 loss, for example, corresponds to a fixed-variance Gaussian model. More recent work (Kendall & Gal, 2017; Lakshminarayanan et al., 2017) has also explored learning more expressive models of $p(y|x)$, by letting a DNN instead output the full set of parameters of a certain family of probability distributions. These *probabilistic regression* approaches however still restrict the parametric model to fairly simple distributions in most cases, such as Gaussian (Chua et al., 2018) or Laplace (Gast & Roth, 2018; Ilg et al., 2018), limiting the expressiveness of the learned conditional target density. While these methods benefit from a clear probabilistic interpretation, they may thus not fully exploit the predictive power of the DNN.
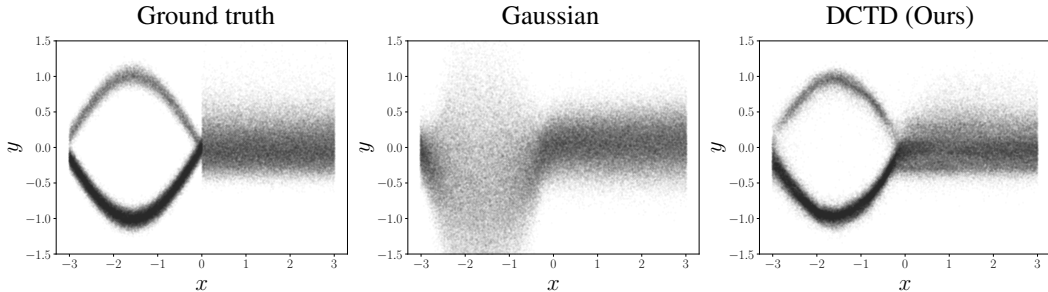
Figure 1: An illustrative 1D regression problem. The training data $\{(x_i, y_i)\}_{i=1}^{2000}$ is generated by the ground truth conditional target density $p(y|x)$. DCTD models $p(y|x)$ by predicting the unnormalized density from (x, y). In contrast to the Gaussian model $p(y|x, \theta) = \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x))$, DCTD can learn complex target densities directly from data, including multi-modal ones.

The quest for improved regression accuracy has also led to the development of more specialized methods, designed for a specific set of tasks. In computer vision, one particularly popular approach is that of *confidence-based regression*. Here, a DNN instead predicts a scalar confidence value for input-target pairs (x, y). The confidence can then be maximized w.r.t. y to obtain a target prediction for a given input x. The approach is commonly employed for image coordinate regression tasks within e.g. human pose estimation (Cao et al., 2017; Xiao et al., 2018) and object detection (Law & Deng, 2018; Zhou et al., 2019), where a 2D heatmap over image pixel coordinates y is predicted. Recently, the approach was also applied to the problem of bounding box regression by Jiang et al. (2018). Their proposed method, IoU-Net, obtained state-of-the-art accuracy on object detection, and was later also successfully applied to the task of visual tracking (Danelljan et al., 2019). The training of such confidence-based regression methods does however entail generating additional pseudo ground truth labels, by for example employing a Gaussian kernel (Xiao et al., 2018), and selecting an appropriate loss function. This both requires numerous design choices to be made, and limits the general applicability of the methods. Moreover, confidence-based regression methods do not allow for a natural probabilistic interpretation in terms of the conditional target density $p(y|x)$. In this work, we therefore set out to develop a method combining the general applicability and clear interpretation of probabilistic regression with the predictive power of confidence-based approaches.

Contributions We propose Deep Conditional Target Densities (DCTD), a novel and general regression method with a clear probabilistic interpretation. DCTD predicts the un-normalized conditional target density $p(y|x)$ from the input-target pair (x, y). It is trained by directly minimizing the associated negative log-likelihood by exploiting Monte Carlo approximations. At test time, targets are predicted by maximizing the conditional target density $p(y|x)$ through gradient-based refinement. Compared to confidence-based approaches, our DCTD requires no pseudo-labels and benefits from a direct probabilistic interpretation. Unlike existing probabilistic models, our approach can learn highly flexible target densities directly from data, as visualized in Figure 1.

We evaluate the proposed method on four diverse computer vision regression tasks: object detection, age estimation, head-pose estimation and visual tracking. Our DCTD method is found to outperform both the direct regression baselines, and popular probabilistic and confidence-based alternatives. Notably, our method achieves a 1.9% AP improvement over the FPN Faster-RCNN (Lin et al., 2017) baseline on the COCO dataset (Lin et al., 2014) when applied for object detection. It also sets a new state-of-the-art on standard benchmarks (Müller et al., 2018; 2016) when applied for bounding box regression in the ATOM visual tracking algorithm (Danelljan et al., 2019).

2 BACKGROUND & RELATED WORK

In supervised regression, the task is to learn to predict a target value $y^\star \in \mathcal{Y}$ from a corresponding input $x^\star \in \mathcal{X}$, given a training set of i.i.d. input-target examples, $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, $(x_i, y_i) \sim p(x, y)$. As opposed to classification, the target space $\mathcal{Y}$ is a continuous set, e.g. $\mathcal{Y} = \mathbb{R}^K$. In computer vision, the input space $\mathcal{X}$ often corresponds to the space of images, whereas the output space $\mathcal{Y}$ depends on the task at hand. Common examples include $\mathcal{Y} = \mathbb{R}_+$ in age estimation (Rothe et al., 2016), $\mathcal{Y} = \mathbb{R}^2$ in image keypoint estimation (Xiao et al., 2018), and $\mathcal{Y} = \mathbb{R}^4$ in object bounding box regression (Jiang et al., 2018).

Direct regression   Over the last decade, DNNs have been shown to excel at a variety of regression problems. Here, a DNN is viewed as a function $f_\theta : \mathcal{X} \to \mathcal{U}$, parameterized by a set of learnable weights $\theta \in \mathbb{R}^P$. The most conventional regression approach is to train a DNN to directly predict the targets, $y^\star = f_\theta(x^\star)$, called *direct regression*. The model parameters θ are learned by minimizing a loss $\ell(f_\theta(x_i), y_i)$ that penalizes the discrepancy between the prediction $f_\theta(x_i)$ and the ground truth target value y_i on training samples (x_i, y_i). The most common choices are the L^2 loss, $\ell(y^*, y) = \|y^* - y\|_2^2$, the L^1 loss, $\ell(y^*, y) = \|y^* - y\|_1$, and their close relatives (Huber, 1964; Lathuilière et al., 2019). From a probabilistic perspective, the choice of loss corresponds to minimizing the negative log-likelihood $-\log p(y|x, \theta)$ of a specific model $p(y|x, \theta)$ of the conditional target density. For example, the L^2 loss is derived from a fixed-variance Gaussian model, $p(y|x, \theta) = \mathcal{N}(y; f_\theta(x), \sigma^2)$.

Probabilistic regression   More recent work (Kendall & Gal, 2017; Lakshminarayanan et al., 2017; Chua et al., 2018) has explicitly taken advantage of this probabilistic perspective to achieve more flexible parametric models $p(y|x, \theta) = p(y|\phi_\theta(x))$, by letting the DNN output the parameters ϕ of a family of probability distributions $p(y|\phi)$. For example, a general 1D Gaussian model can be realized as $p(y|\phi_\theta(x)) = \mathcal{N}\big(y; \mu_\theta(x), \sigma_\theta^2(x)\big)$, where the DNN outputs the mean and log-variance as $f_\theta(x) = \phi_\theta(x) = [\,\mu_\theta(x) \quad \log \sigma_\theta^2(x)\,]^\mathsf{T} \in \mathbb{R}^2$. The model parameters θ are learned by minimizing the negative log-likelihood $-\sum_{i=1}^N \log p(y_i|x_i, \theta)$ over the training set $\mathcal{D}$. At test time, a target estimate $y^\star$ is obtained by first predicting the density parameter values $\phi_\theta(x^\star)$ and then, for instance, taking the expected value of $p(y|\phi_\theta(x))$. Previous work has applied Gaussian and Laplace models on computer vision tasks such as object detection (Feng et al., 2019; He et al., 2019) and optical flow estimation (Gast & Roth, 2018; Ilg et al., 2018). The aim of such probabilistic approaches is often not only to achieve accurate predictions, but also to provide an estimate of the aleatoric uncertainty (Kendall & Gal, 2017), which models noise and ambiguities inherent in the data itself. Our method also entails predicting a conditional target density $p(y|x, \theta)$ and minimizing the associated negative log-likelihood. However, our model $p(y|x, \theta)$ is not restricted to the functional form of any specific probability density (e.g. Gaussian or Laplace), but is instead directly defined by the DNN architecture itself, allowing for more expressive target densities.

Confidence-based regression   Another category of approaches reformulate the regression problem as $y^\star = \arg\max_y f_\theta(x, y)$, where $f_\theta(x, y) \in \mathbb{R}$ is a scalar confidence value predicted by the DNN. The idea is thus to predict a quantity $f_\theta(x, y)$, depending on both input x and target y, that can be maximized over y to obtain the final prediction $y^\star$. This maximization-based formulation is inherent in Structural SVMs (Tsochantaridis et al., 2005), but has also been adopted for DNNs. We term this family of approaches *confidence-based regression*. Different from direct regression, the predicted confidence $f_\theta(x, y)$ can encapsulate multiple hypotheses and other ambiguities. Confidence-based regression has been shown particularly suitable for image-coordinate regression tasks, such as hand keypoint localization (Simon et al., 2017) and body-part detection (Wei et al., 2016; Pishchulin et al., 2016; Xiao et al., 2018). In these cases, a CNN is trained to output a 2D heatmap over the image pixel coordinates y, thus taking full advantage of the translational invariance of the problem. A similar approach has also been employed to locate the two defining corners (Law & Deng, 2018) or four extreme points (Zhou et al., 2019) of bounding boxes in object detection. In computer vision, confidence prediction has also been successfully employed for tasks other than pure image-coordinate regression. Jiang et al. (2018) proposed the IoU-Net for bounding box regression in object detection, where a bounding-box $y \in \mathbb{R}^4$ and image x are both inputs to the DNN to predict a confidence $f_\theta(x, y)$. It employs a pooling-based architecture that is differentiable w.r.t. the bounding box y, allowing gradient-based maximization to obtain the final estimate $y^\star = \arg\max_y f_\theta(x, y)$. IoU-Net was later also applied to visual tracking (Danelljan et al., 2019).

In general, confidence-based approaches are trained using a set of generated *pseudo label* confidences $u_{x,y}$ and by employing a loss $\ell(f_\theta(x, y), u_{x,y})$. One strategy (Pishchulin et al., 2016; Law & Deng, 2018) is to treat the confidence prediction as a binary classification problem, where $u_{x,y}$ represents either the class, $u_{x,y} \in \{0, 1\}$, or its probability, $u_{x,y} \in [0, 1]$, and employ cross-entropy based losses ℓ. The other approach is to treat the confidence prediction as a direct regression problem itself by applying standard regression losses, such as L^2 (Simon et al., 2017; Danelljan et al., 2019; Wei et al., 2016) or the Huber loss (Jiang et al., 2018). In these cases, the pseudo label confidences $u_{x,y}$ can be constructed using a similarity measure S in the target value space, $u_{x_i,y} = S(y, y_i)$, for example defined as the Intersection over Union (IoU) between two bounding boxes (Jiang et al., 2018) or simply by a Gaussian kernel (Wei et al., 2016; Xiao et al., 2018). While these approaches

have demonstrated impressive results, existing confidence-based approaches require important design choices. In particular, the strategy for constructing the pseudo labels $u_{x,y}$ and the choice of loss $\ell(f_\theta(x,y), u_{x,y})$ are often crucial for performance and highly task-dependent. Moreover, the predicted confidence $f_\theta(x,y)$ can be difficult to interpret, since it has no natural connection to the conditional target density $p(y|x)$. In contrast, our approach is directly trained to predict $p(y|x)$ itself, and does not require generation of pseudo label confidences or choosing a specific loss.

Regression-by-classification   A regression problem can also be treated as a classification problem by first discretizing the target space $\mathcal{Y}$ into a finite set of C classes. Standard techniques from classification, such as softmax and the cross-entropy loss, can then be employed. Rothe et al. (2016) additionally computed the softmax expected value to obtain a more fine-grained prediction, and applied their method to the task of age estimation. Ruiz et al. (2018) applied the same method to head-pose estimation, but also added an L^2 loss term for the softmax expected value during training. Again for age estimation, Pan et al. (2018) then added an additional loss term penalizing the softmax variance. A hierarchical classification approach has also been proposed for both age estimation (Yang et al., 2018) and head-pose estimation (Yang et al., 2019). The discretization of the target space $\mathcal{Y}$ often complicates exploiting its inherent neighborhood structure. This has been addressed by exploring ordinal regression methods for 1D problems (Cao et al., 2019; Diaz & Marathe, 2019). Finally, classification into coarse discrete bins can be combined with direct regression, a technique often utilized in 2D (Redmon et al., 2016; Liu et al., 2016) and 3D (Shi et al., 2019; Qi et al., 2018) object detection. While our approach can be seen as a generalization of the softmax model for classification to the continuous target space $\mathcal{Y}$, it does not suffer from the aforementioned drawbacks of regression-by-classification. On the contrary, our model naturally allows the network to exploit the full structure of the continuous target space $\mathcal{Y}$.

3 REGRESSION USING DEEP CONDITIONAL TARGET DENSITIES

In this work, we take the probabilistic view of regression by creating a model $p(y|x, \theta)$ of the conditional target density $p(y|x)$. Instead of defining $p(y|x, \theta)$ by letting a DNN predict the parameters of a certain family of probability distributions (e.g. Gaussian or Laplace), we construct a versatile model that can better leverage the predictive power of DNNs. To that end, we take inspiration from confidence-based regression approaches and let a DNN predict a scalar value for any input-target pair (x, y). Unlike confidence-based methods however, this prediction has a clear probabilistic interpretation. Specifically, we view a DNN as a function $f_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, parameterized by $\theta \in \mathbb{R}^P$, that maps an input-target pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ to a scalar value $f_\theta(x, y) \in \mathbb{R}$. Then, we define the Deep Conditional Target Density (DCTD) according to,

$$p(y|x, \theta) = \frac{e^{f_\theta(x,y)}}{Z(x, \theta)}, \qquad Z(x, \theta) = \int e^{f_\theta(x,y)} dy, \tag{1}$$

where $Z(x, \theta)$ is the normalizing constant. We train our DCTD model $p(y|x, \theta)$ by minimizing the negative log-likelihood $-\log p(\{y_i\}_i | \{x_i\}_i, \theta) = \sum_{i=1}^N -\log p(y_i|x_i, \theta)$, where each term is given by,

$$-\log p(y_i|x_i, \theta) = \log\left(\int e^{f_\theta(x_i,y)} dy\right) - f_\theta(x_i, y_i). \tag{2}$$

The training thus requires the evaluation of the normalizing constant $Z(x, \theta)$, involving the integral in equation 2. This can be achieved using effective finite approximations. In some tasks, such as image-coordinate regression, this is naturally performed by a grid approximation, utilizing the dense prediction already employed in many such methods. In this work, we however investigate a more generally applicable technique, namely Monte Carlo approximations. This procedure, when employed for training the network, is detailed in Section 3.1.

At test time, given an input $x^\star$, our model in equation 1 allows evaluating the conditional target density $p(y|x^\star, \theta)$ for any y by first approximating the constant $Z(x^\star, \theta)$ and then predicting the scalar $f_\theta(x^\star, y)$ using the DNN. This enables the computation of, for instance, means and variances of the target value y. In this work, we focus on finding the most likely prediction, $y^\star = \arg\max_y p(y|x^\star, \theta) = \arg\max_y f_\theta(x^\star, y)$, which does not require the evaluation of $Z(x, \theta)$ during inference. Thanks to the auto-differentiation capabilities of modern deep learning frameworks, we can apply gradient-based techniques to find the final prediction by simply maximizing the network output $f_\theta(x^\star, y)$ w.r.t. y. We elaborate on this procedure for prediction in Section 3.2.

3.1 TRAINING

Our model $p(y|x, \theta) = e^{f_\theta(x,y)}/Z(x, \theta)$ of the conditional target density is trained by minimizing the negative log-likelihood $\sum_{i=1}^{N} - \log p(y_i|x_i, \theta)$. To evaluate the integral in equation 2, we employ a Monte Carlo approximation. Specifically, each term $- \log p(y_i|x_i, \theta)$ is approximated by sampling values $y^{(k)}$ from a proposal distribution $q(y|y_i)$ that depends on the ground truth target y_i,

$$- \log p(y_i|x_i, \theta) \approx \log \left(\frac{1}{M} \sum_{k=1}^{M} \frac{e^{f_\theta(x_i, y^{(k)})}}{q(y^{(k)}|y_i)} \right) - f_\theta(x_i, y_i), \quad y^{(k)} \sim q(y|y_i). \quad (3)$$

The final loss $J(\theta)$ is then obtained by averaging over all training samples (x_i, y_i) in the mini-batch,

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \log \left(\frac{1}{M} \sum_{m=1}^{M} \frac{e^{f_\theta(x_i, y^{(i,m)})}}{q(y^{(i,m)}|y_i)} \right) - f_\theta(x_i, y_i), \quad (4)$$

where $\{y^{(i,m)}\}_{m=1}^{M}$ are M samples drawn from $q(y|y_i)$. Qualitatively, minimizing $J(\theta)$ encourages the DNN to output large values $f_\theta(x_i, y_i)$ for the ground truth target y_i, while minimizing the predicted value $f_\theta(x_i, y)$ at all other targets y. In ambiguous or uncertain cases, the DNN can output small values everywhere or large values at multiple hypotheses, but at a cost of a higher loss.

As seen in equation 4, the DNN f_θ is applied both to the input-target pair (x_i, y_i), and all input-sample pairs $\{(x_i, y^{(i,m)})\}_{m=1}^{M}$ during training. While this can seem inefficient, most applications in computer vision employ network architectures that first extract a deep feature representation for the input x_i. The DNN can thus be designed to combine this input feature with the target y at a late stage, meaning that the input feature extraction process, which becomes the main computational bottleneck, needs to be performed only once for each x_i. In practice, we found our training strategy to not add any significant computational overhead compared to the baselines.

Compared to confidence-based regression, a significant advantage of our approach is that there is no need for generating task-dependent pseudo label confidences or choosing between different losses. The only design choice of our method is the proposal distribution $q(y|y_i)$. Note however that since the loss (equation 4) explicitly adapts to $q(y|y_i)$, this choice has no effect on the overall behaviour of the loss, only on the quality of the sampled approximation. We found a simple mixture of a few equally weighted Gaussian components, all centered at the target label y_i, to consistently perform well in our experiments. Specifically we set,

$$q(y|y_i) = \frac{1}{L} \sum_{l=1}^{L} \mathcal{N}(y; y_i, \sigma_l^2), \quad (5)$$

where the variances σ_l^2 are hyperparameters selected based on a validation set for each experiment.

3.2 PREDICTION

Given an input $x^\star$ at test time, the trained DNN f_θ can be used to evaluate the full target density $p(y|x^\star, \theta) = e^{f_\theta(x^\star,y)}/Z(x^\star, \theta)$, by employing the aforementioned techniques to approximate $Z(x^\star, \theta)$. In many applications, the most likely prediction $y^\star = \arg\max_y p(y|x^\star, \theta)$ is however the single desired output. For DCTD, this is obtained by directly maximizing the DNN output, $y^\star = \arg\max_y f_\theta(x^\star, y)$, thus not requiring $Z(x^\star, \theta)$ to be evaluated. By designing the DNN f_θ to be differentiable w.r.t. the target y, the gradient $\nabla_y f_\theta(x^\star, y)$ can be efficiently evaluated using the auto-differentiation tools implemented in modern deep learning frameworks. We can therefore perform gradient ascent to find a local maximum of $f_\theta(x^\star, y)$. The gradient ascent refinement is performed either on a single initial estimate $\hat{y}$, or on a set of random initializations $\{\hat{y}_k\}_{k=1}^{K}$ to obtain a final accurate prediction $y^\star$. As noted in Section 3.1, this prediction procedure can be made highly efficient in practice by extracting the deep feature representation for $x^\star$ only once. Back-propagation is then performed only through a few final layers of the DNN f_θ in order to evaluate the gradient $\nabla_y f_\theta(x^\star, y)$. Moreover, the gradient computation for a set of target candidates $\{\hat{y}_k\}_{k=1}^{K}$ can be parallelized on the GPU by simple batching, requiring no significant overhead. Please refer to Appendix B for a detailed algorithm of this prediction procedure.

Table 1: Results for the object detection task on the test-dev split of the COCO dataset. Our approach significantly outperforms the baseline Faster-RCNN and the confidence-based IoU-Net.

Formulation Approach	Direct Faster-RCNN	Gaussian	Laplace	Confidence IoU-Net	Confidence IoU-Net[†]	**DCTD Ours**
AP (%)	37.2	36.7	37.1	38.3	38.2	**39.1**
$AP_{50}(\%)$	**59.2**	58.7	59.1	58.3	58.4	58.5
$AP_{75}(\%)$	40.3	39.6	40.2	41.4	41.4	**41.8**

4 EXPERIMENTS

We perform comprehensive experiments on four different computer vision tasks. Our DCTD method is compared both to baseline regression methods and to state-of-the-art models. All experiments are implemented in PyTorch (Paszke et al., 2017). Code will be made available upon publication.

4.1 OBJECT DETECTION

We first perform experiments for visual object detection, the problem of estimating a bounding box for each object in the image from a set of given classes. Specifically, we compare our regression method to other techniques for the task of bounding-box regression, by integrating them into an existing object detection pipeline. To this end, we use the Faster-RCNN (Ren et al., 2015) framework, which serves as a popular baseline in the object detection field due to its strong state-of-the-art performance. It uses one network head for classification and the second for regressing the bounding box using the direct method. We also compare our approach to the confidence-based IoU-Net (Jiang et al., 2018). It extends Faster-RCNN with an additional branch that predicts the IoU overlap between a target box y and the ground truth. The IoU prediction branch uses differentiable region pooling (Jiang et al., 2018), allowing the initial bounding box predicted by the Faster-RCNN to be refined using gradient-based maximization of the predicted IoU confidence.

For our approach, we employ an identical architecture as used in IoU-Net for a fair comparison. Instead of training the network to output the IoU, we predict the exponent $f_\theta(x, y)$ in equation 1, trained by minimizing the negative log-likelihood (NLL) in equation 4. We parametrize the bounding box as $y = (c_x/w_0, c_y/h_0, \log w, \log h) \in \mathbb{R}^4$, where (c_x, c_y) and (w, h) denote the center coordinate and size respectively. The reference size (w_0, h_0) is set to that of the ground truth during training and the initial box during inference. For the proposal distribution (equation 5) we employ $L = 3$ isotropic Gaussians with standard deviation $\sigma_l = 0.05 \cdot 2^l$. In addition to the standard IoU-Net, we compare with a version (denoted IoU-Net[†]) employing the same proposal distribution and inference settings as in our approach. For both our method and IoU-Net[†], we set the refinement step-length using grid search on a separate validation set. We also compare with a Gaussian and a Laplace probabilistic model for bounding box regression by modifying the Faster-RCNN regression head to predict both the mean and log-variance of the distribution, and adopting the NLL loss.

Our experiments are performed on the large-scale COCO benchmark (Lin et al., 2014). As per the official guideline, we use the *2017 train* split ($\approx 118\,000$ images) for training and the *2017 val* split ($\approx 5\,000$ images) as the validation set for setting the hyperparameters. The results are reported on the *2017 test-dev* split ($\approx 20\,000$ images), in terms of the standard COCO metrics AP (mean Average Precision AP_T over 10 IoU thresholds $T \in [0.5, 0.95]$), AP_{50}, and AP_{75}. We initialize all networks in our comparison with the pre-trained Faster-RCNN weights, using the ResNet50-FPN (Lin et al., 2017) backbone and re-train *only* the newly added layers for a fair comparison. Further details are provided in Appendix C. The results are shown in Table 1. Our DCTD approach obtains the best results, outperforming both Faster-RCNN and IoU-Net by 1.9% and 0.8% in AP, respectively.

4.2 AGE ESTIMATION

In age estimation, we are given a cropped image $x \in \mathbb{R}^{h \times w \times 3}$ of a person's face, and the task is to predict his/her age $y \in \mathbb{R}_+$. We utilize the UTKFace (Zhang et al., 2017) dataset, specifically the subset of $16\,434$ images used by Cao et al. (2019). In this subset, ground truth age labels $y_i \in [21, 60]$. We also utilize the dataset split employed by Cao et al. (2019), with $3\,287$ test images and $11\,503$ images for training. Additionally, we use $1\,644$ of the training images for validation. Methods are evaluated in terms of the Mean Absolute Error (MAE). The DNN architecture $f_\theta(x, y)$

Table 2: Results for the age estimation experiments. Refinement using DCTD consistently improves MAE (lower is better) for the age predictions outputted by a number of baselines.

+DCTD	Cao et al. (2019)	Direct	Gaussian	Laplace	Softmax (CE, L^2)	Softmax (CE, L^2, Var)
	5.47 ± 0.01	4.81 ± 0.02	4.79 ± 0.06	4.85 ± 0.04	4.78 ± 0.05	4.81 ± 0.03
✓	-	$\mathbf{4.65} \pm 0.02$	4.66 ± 0.04	4.81 ± 0.04	$\mathbf{4.65} \pm 0.04$	4.69 ± 0.03

Table 3: Results for the head-pose estimation experiments. Refinement using DCTD consistently improves the average MAE for Yaw, Pitch and Roll for the predicted pose outputted by our baselines.

+DCTD	Yang et al. (2019)	Direct	Gaussian	Laplace	Softmax (CE, L^2)	Softmax (CE, L^2, Var)
	3.60	3.09 ± 0.07	3.12 ± 0.08	3.21 ± 0.06	3.04 ± 0.08	3.15 ± 0.07
✓	-	3.07 ± 0.07	3.11 ± 0.07	3.19 ± 0.06	$\mathbf{3.01} \pm 0.07$	3.11 ± 0.06

of our DCTD first extracts ResNet50 (He et al., 2016) features $g_x \in \mathbb{R}^{2048}$ from the input image x. The age y is processed by four fully-connected layers, generating $g_y \in \mathbb{R}^{128}$. The two feature vectors are then concatenated and processed by two fully-connected layers, outputting $f_\theta(x, y) \in \mathbb{R}$. We apply our DCTD to refine the age predicted by baseline models, using the gradient ascent maximization of $f_\theta(x, y)$ (Section 3.2). All baseline DNN models employ a similar architecture, including an identical ResNet50 for feature extraction and the same number of fully-connected layers to output either the age $y \in \mathbb{R}$ (*Direct*), mean and variance parameters for Gaussian and Laplace distributions, or to output logits for C discretized classes (*Softmax*). The results are found in Table 2. We observe that age refinement provided by our DCTD method consistently improves the accuracy of the predictions generated by the baseline methods. Further details are provided in Appendix D.

4.3 HEAD-POSE ESTIMATION

In head-pose estimation, we are given an image $x \in \mathbb{R}^{h \times w \times 3}$ of a person, and are tasked with predicting the orientation $y \in \mathbb{R}^3$ of his/her head, where y is the Yaw, Pitch and Roll angles. We utilize the BIWI (Fanelli et al., 2013) dataset, specifically the processed dataset provided by Yang et al. (2019), in which the images have been cropped to faces detected using MTCNN (Zhang et al., 2016). We also employ protocol 2 as defined by Yang et al. (2019), with $10\,613$ images for training and $5\,065$ images for testing. Additionally, we use $1\,010$ training images for validation. The methods are evaluated in terms of the average MAE for Yaw, Pitch and Roll. The network architecture of the DNN f_θ defining our DCTD takes the image $x \in \mathbb{R}^{h \times w \times 3}$ and orientation $y \in \mathbb{R}^3$ as inputs, but is otherwise identical to the age estimation case (Section 4.2). Our DCTD model is again evaluated by applying the optimization-based refinement to the predicted orientation $y \in \mathbb{R}^3$ outputted by a number of baseline models. We use the same baselines as for age estimation, and apart from minor changes required to increase the output dimension from 1 to 3, identical network architectures are also used. The results are found in Table 3, and also in this case we observe that refinement using DCTD consistently improves upon the baselines. Further details are provided in Appendix E.

4.4 VISUAL TRACKING

Lastly, we evaluate our approach on the problem of generic visual object tracking. The task is to estimate the bounding box of a target object in every frame of a video. The target object is defined by a given box in the first video frame. We employ the recently introduced ATOM (Danelljan et al., 2019) tracker as our baseline. Given the first-frame annotation, ATOM trains a classifier to first roughly localize the target in a new frame. The target bounding box is then determined using an IoU-Net based module, which is also conditioned on the first-frame target appearance using a modulation-based architecture. We train our network to predict the conditional target density through $f_\theta(x, y)$ in equation 1, using a network architecture identical to the baseline ATOM tracker. In particular, we employ the same bounding box parameterization as for object detection (Section 4.1) and sample $M = 128$ boxes during training from a proposal distribution (equation 5) generated by $L = 2$ Gaussians with standard deviations of 0.05 and 0.5. During tracking, we follow the same procedure as in ATOM, sampling 10 boxes in each frame followed by gradient ascent to refine the estimate generated by the classification module.

We demonstrate results on two standard tracking benchmarks: TrackingNet (Müller et al., 2018) and UAV123 (Müller et al., 2016). TrackingNet contains challenging videos sampled from YouTube,

Table 4: Results on the two tracking datasets TrackingNet and UAV123. The asterisk * indicate an approximate value (± 1), taken from the plot in the corresponding paper, due to the unavailability of the raw results. Our approach outperforms the baseline ATOM and other state-of-the-art trackers.

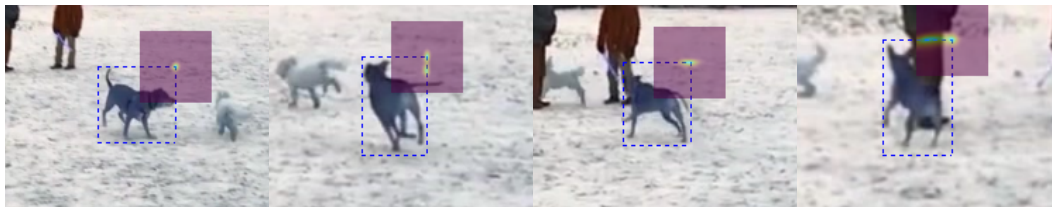| Dataset	Metric	SiamFC Bertinetto et al. (2016)	MDNet Nam & Han (2016)	DaSiamRPN Zhu et al. (2018)	SiamRPN++ Li et al. (2019)	ATOM Danelljan et al. (2019)	ATOM†	**Ours**
	Precision (%)	53.3	56.5	59.1	**69.4**	64.8	66.7	68.9
TrackingNet	Norm. Prec. (%)	66.6	70.5	73.3	**80.0**	77.1	78.3	79.5
	Success (%)	57.1	60.6	63.8	73.3	70.3	72.1	**73.7**
	$OP_{0.50}$ (%)	-	-	73.6	75*	78.9	79.6	**80.1**
UAV123	$OP_{0.75}$ (%)	-	-	41.1	56*	55.7	56.0	**59.8**
	AUC (%)	-	52.8	58.4	61.3	65.0	65.0	**66.5**

Figure 2: Visualization of the conditional target density $p(y|x,\theta) \sim e^{f_\theta(x,y)}$ 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 (blue box). Our network predicts flexible densities, expressing meaningful uncertainties in challenging cases.

with a test set of 511 videos. The main metric is the Success, defined as the average IoU overlap with the ground truth. UAV123 contains 123 videos captured from a UAV, and includes small and fast-moving objects. We report the overlap precision metric (OP_T), defined as the percentage of frames having bounding box IoU overlap larger than a threshold T. The final AUC score is computed as the average OP over all thresholds $T \in [0, 1]$. Hyperparameters are set on the OTB (Wu et al., 2015) and NFS (Galoogahi et al., 2017) datasets, containing 100 videos each. Due to the significant challenges imposed by the limited supervision and generic nature of the tracking problem, there are not any competitive baselines employing direct bounding box regression. Current state-of-the-art employ either confidence-based regression, as in ATOM, or anchor-based bounding box regression techniques (Zhu et al., 2018; Li et al., 2019). We therefore only compare with the ATOM baseline and include other recent state-of-the-art methods in the comparison. As in section 4.1, we compare with a version of the IoU-Net based ATOM (denoted ATOM†) employing the same training and inference settings as our final approach. The results are shown in Table 4. Our approach achieves a significant 1.6% and 1.5% absolute improvement over ATOM on the overall metric on TrackingNet and UAV123 respectively. Note that the improvements are most prominent for high-accuracy boxes, indicated by the $OP_{0.75}$ score. Moreover, our approach outperforms the recent SiamRPN++ (Li et al., 2019), which employs anchor-based bounding box regression (Ren et al., 2015; Redmon & Farhadi, 2016) and a much deeper backbone network (ResNet50) compared to ours (ResNet18). Figure 2 visualizes the conditional target density generated by our approach for tracking.

5 CONCLUSION

We proposed Deep Conditional Target Densities (DCTD), a novel and generally applicable regression method with a clear probabilistic interpretation. It directly models the conditional target density $p(y|x)$ by predicting the un-normalized density through a DNN $f_\theta(x, y)$, taking the input-target pair (x, y) as input. The model is trained by minimizing the associated negative log-likelihood, employing a Monte Carlo approximation of the normalizing constant. At test time, targets are predicted by maximizing the DNN output $f_\theta(x, y)$ w.r.t. y via gradient-based refinement. Experiments performed on four diverse computer vision applications demonstrate the high accuracy and wide applicability of our method. However, this work constitutes an initial investigation of DCTD. Future directions include exploring better architectural designs, studying other regression applications, and investigating DCTD's potential for aleatoric uncertainty estimation.

REFERENCES

Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *ECCV workshop*, 2016.

Wenzhi Cao, Vahid Mirjalili, and Sebastian Raschka. Rank-consistent ordinal regression for neural networks. *arXiv preprint arXiv:1901.07884*, 2019.

Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7291–7299, 2017.

Chen-Rui Chou, Brandon Frederick, Gig Mageras, Sha Chang, and Stephen Pizer. 2D/3D image registration using regression learning. *Computer Vision and Image Understanding*, 117(9):1095–1106, 2013.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4759–4770, 2018.

Martin Danelljan and Goutam Bhat. PyTracking: Visual tracking library based on PyTorch. https://github.com/visionml/pytracking, 2019. Accessed: 12/08/2019.

Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: Accurate tracking by overlap maximization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4660–4669, 2019.

Raul Diaz and Amit Marathe. Soft labels for ordinal regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. In *CVPR*, 2019.

Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, and Luc Van Gool. Random forests for real time 3d face analysis. *International Journal of Computer Vision (IJCV)*, 101(3): 437–458, 2013.

Di Feng, Lars Rosenbaum, Fabian Timm, and Klaus Dietmayer. Leveraging heteroscedastic aleatoric uncertainties for robust real-time lidar 3D object detection. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1280–1287. IEEE, 2019.

Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *ICCV*, 2017.

Jochen Gast and Stefan Roth. Lightweight probabilistic deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3369–3378, 2018.

Ross B. Girshick. Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

Jinwei Gu, Xiaodong Yang, Shalini De Mello, and Jan Kautz. 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.

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 (CVPR)*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2888–2897, 2019.

Lianghua Huang, Xin Zhao, and Kaiqi Huang. GOT-10k: A large high-diversity benchmark for generic object tracking in the wild. *arXiv preprint arXiv:1810.11981*, 2018.

Peter J Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, pp. 73–101, 1964.

Eddy Ilg, Ozgun Cicek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Bro. Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 652–667, 2018.

Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 784–799, 2018.

Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5574–5584, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6402–6413, 2017.

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)*, 2019.

Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 734–750, 2018.

Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 740–755, 2014.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2117–2125, 2017.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 21–37. Springer, 2016.

Francisco Massa and Ross Girshick. 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.

Matthias Müller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *ECCV*, 2016.

Matthias Müller, Adel Bibi, Silvio Giancola, Salman Al-Subaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, 2018.

Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016.

Marc Niethammer, Yang Huang, and François-Xavier Vialard. Geodesic regression for image time-series. In *International conference on medical image computing and computer-assisted intervention*, pp. 655–662. Springer, 2011.

Zhenxing Niu, Mo Zhou, Le Wang, Xinbo Gao, and Gang Hua. 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.

Hongyu Pan, Hu Han, Shiguang Shan, and Xilin Chen. 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.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS - Autodiff Workshop*, 2017.

Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V. Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 4929–4937, 2016.

Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum PointNets for 3D object detection from RGB-D data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 918–927, 2018.

Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, 2016.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015.

Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision*, 126(2-4): 144–157, 2016.

Nataniel Ruiz, Eunji Chong, and James M Rehg. Fine-grained head pose estimation without keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2074–2083, 2018.

Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–779, 2019.

Tomas Simon, Hanbyul Joo, Iain A. Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 4645–4653, 2017.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, 2005.

Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 4724–4732, 2016.

Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *TPAMI*, 37(9):1834–1848, 2015.

Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 466–481, 2018.

Tsun-Yi Yang, Yi-Hsuan Huang, Yen-Yu Lin, Pi-Cheng Hsiu, and Yung-Yu Chuang. SSR-Net: A compact soft stagewise regression network for age estimation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

Tsun-Yi Yang, Yi-Ting Chen, Yen-Yu Lin, and Yung-Yu Chuang. 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. URL `https://github.com/shamangary/FSA-Net`.

Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5810–5818, 2017. URL `https://susanqq.github.io/UTKFace/`.

Xingyi Zhou, Jiacheng Zhuo, and Philipp Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 850–859, 2019.

Zheng Zhu, Qiang Wang, Li Bo, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, 2018.

APPENDIX A   ILLUSTRATIVE EXAMPLE

The ground truth conditional target density $p(y|x)$ in Figure 1 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 (Kingma & Ba, 2014) optimizer.

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

$$p(y|x, \theta) = \mathcal{N}\big(y; \mu_\theta(x),\, \sigma_\theta^2(x)\big), \qquad f_\theta(x) = [\,\mu_\theta(x) \quad \log \sigma_\theta^2(x)\,]^{\mathsf{T}} \in \mathbb{R}^2.$$

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

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

The DNN f_θ 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 μ and $\log \sigma^2$ of three fully-connected layers ($10 \to 10$, $10 \to 10$, $10 \to 1$).

The DCTD model 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 $q(y|y_i)$ (equation 5) with $L = 2$ and variances $\sigma_1^2 = 0.1^2$, $\sigma_2^2 = 0.8^2$.

APPENDIX B   PREDICTION USING DEEP CONDITIONAL TARGET DENSITIES

The procedure for prediction described in Section 3.2 is further detailed in Algorithm 1, where T denotes the number of gradient ascent iterations, λ is the step-length and η is a decay of the step-length.

Algorithm 1 Prediction via optimization-based refinement

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

1: $y \leftarrow \hat{y}$.
2: **for** $t = 1, \ldots, T$ **do**
3:     PrevValue $\leftarrow f_\theta(x^\star, y)$.
4:     $\tilde{y} \leftarrow y + \lambda \nabla_y f_\theta(x^\star, y)$.
5:     NewValue $\leftarrow f_\theta(x^\star, \tilde{y})$.
6:     **if** NewValue $>$ PrevValue **then**
7:         $y \leftarrow \tilde{y}$.
8:     **else**
9:         $\lambda \leftarrow \lambda \cdot \eta$.
10: **Return** y.

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)

C.1   NETWORK ARCHITECTURE

We use the Faster-RCNN (Ren et al., 2015) detector with ResNet50-FPN (Lin et al., 2017) as our baseline. 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 (Girshick, 2015). The pooled features are then passed through a feature extractor (denoted
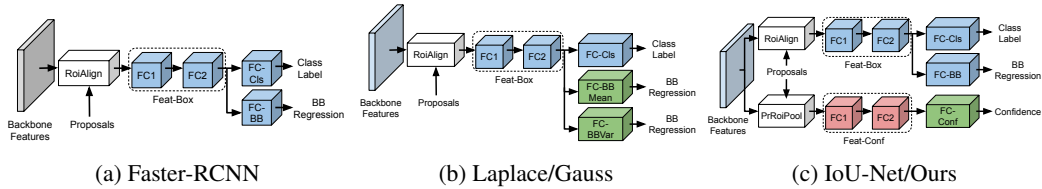
Figure 3: Network architectures for the different detection networks used in our experiments in Section 4.1. The backbone feature extractor (ResNet50-FPN), and the region proposal network (RPN) is not shown for clarity. We do not train the blocks in blue color, using the pre-trained Faster-RCNN weights from Massa & Girshick (2018) 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.

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 Massa & Girshick (2018). Note that we use the RoiAlign (He et al., 2017) layer instead of RoiPool in our experiments as it has been shown to achieve better performance (He et al., 2017).

For the Gaussian and Laplace probabilistic models (Gaussian and Laplace in Table 1), we replace the FC-BBReg layer in Faster-RCNN with parallel two 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.

For our confidence-based IoU-Net (Jiang et al., 2018) models (IoU-Net and IoU-Net[†] in Table 1), we use the same network architecture as employed in the original paper. 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 (Jiang et al., 2018) 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 equation 1 instead. Illustrations of the architectures are found in Figure 3.

C.2 TRAINING

We use the pre-trained weights for Faster-RCNN from Massa & Girshick (2018). Note that the bounding box regression in Faster-RCNN is trained using a direct method, with an Huber loss (Huber, 1964). We trained the other networks in Table 1 (Gaussian, Laplace, IoU-Net, IoU-Net[†] and DCTD) on the MS-COCO (Lin et al., 2014) training split (*2017 train*) using stochastic gradient descent (SGD) with a batch size of 16 for 60k iterations. The base learning rate lr_{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_{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 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, 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. Both Gaussian and Laplace models were trained with a base learning rate $lr_{\text{base}} = 0.005$ by minimizing the negative log-likelihood.

For the IoU-Net, IoU-Net[†] and our DCTD 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 mentioned in

the original paper, we used a base learning rate $lr_{\text{base}} = 0.01$ for the IoU-Net and IoU-Net† networks. For our DCTD network, 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. 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 DCTD, we sample boxes from a proposal distribution (equation 5) generated by $L = 3$ Gaussians with standard deviations of 0.05, 0.1, and 0.2. The IoU-Net and IoU-Net† are trained by minimizing the Huber loss between the predicted IoU and the ground truth, while DCTD is training by minimizing the negative log likelihood of the training data.

C.3 INFERENCE

The inference in both Gaussian and Laplace models is identical to the one employed by Faster-RCNN. Thus, we do not utilize the predicted variances for inference. For IoU-Net and IoU-Net†, we perform IoU-Guided NMS as described in (Jiang et al., 2018), followed by optimization-based refinement (Algorithm 1). For our approach we adopt the same NMS technique, but guide it with the values $f_\theta(x, y)$ predicted by or 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). 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 DCTD approach, we use $\lambda = 0.0001$ and $\lambda = 0.0004$ with $\eta = 0.5$. For all methods, these hyperparameters (λ and η) were set using a grid search on the MS-COCO validation split (*2017 val*). We used $T = 10$ refinement iterations for each of the three models.

APPENDIX D AGE ESTIMATION

In this appendix, further details on the age estimation experiments (Section 4.2) are provided.

D.1 DCTD NETWORK ARCHITECTURE

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

D.2 DCTD TRAINING

The DCTD model is trained using $M = 1024$ samples from a proposal distribution $q(y|y_i)$ (equation 5) with $L = 2$ and variances $\sigma_1^2 = 0.1^2$, $\sigma_2^2 = 20^2$. It 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 200×200. 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 200×200 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.

D.3 DCTD PREDICTION

For this experiment, we use a slight variation of Algorithm 1, which is found in Algorithm 2. There, T is the number of gradient ascent iterations, λ is the stepsize, Ω_1 is an early-stopping threshold and Ω_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.

Algorithm 2 Prediction via optimization-based refinement (variation)

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

1: $y \leftarrow \hat{y}$.
2: **for** $t = 1, \dots, T$ **do**
3:      PrevValue $\leftarrow f_\theta(x^\star, y)$.
4:      $y \leftarrow y + \lambda \nabla_y f_\theta(x^\star, y)$.
5:      NewValue $\leftarrow f_\theta(x^\star, y)$.
6:      **if** $|\text{PrevValue} - \text{NewValue}| < \Omega_1$ **or** $(\text{NewValue} - \text{PrevValue}) < \Omega_2$ **then**
7:          **Return** y.
8: **Return** y.

D.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 DCTD 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.

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

$$p(y|x, \theta) = \mathcal{N}\big(y; \mu_\theta(x), \sigma_\theta^2(x)\big), \qquad f_\theta(x) = [\,\mu_\theta(x) \quad \log \sigma_\theta^2(x)\,]^\mathsf{T} \in \mathbb{R}^2.$$

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

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

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 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,

$$p(y|x, \theta) = \frac{1}{2\beta_\theta(x)} \exp\left(-\frac{|y - \mu_\theta(x)|}{\beta_\theta(x)} \right), \qquad f_\theta(x) = [\,\mu_\theta(x) \quad \log \beta_\theta(x)\,]^\mathsf{T} \in \mathbb{R}^2.$$

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

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

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 $\log \beta_\theta(x)$. 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 ($2048 \rightarrow 2048$, $2048 \rightarrow C$), 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 (Pan et al., 2018) losses, $J = J_{CE} + 0.1 J_{L^2} + 0.05 J_{Var}$. The prediction $\hat{y}$ is computed as the softmax expected value.

D.5   FULL RESULTS

Full experiment results, expanding the results found in Table 2 (Section 4.2), are provided in Table 5.

Table 5: Full results for the age estimation experiments. Refinement using DCTD consistently improves MAE (lower is better) for the age predictions outputted by a number of baselines.

Method	MAE
OR-CNN (Niu et al., 2016)	5.74 ± 0.05
CORAL-CNN (Cao et al., 2019)	5.47 ± 0.01
Direct - Huber	4.80 ± 0.06
Direct - Huber + DCTD	4.74 ± 0.06
Direct - L2	4.81 ± 0.02
Direct - L2 + DCTD	$\mathbf{4.65} \pm 0.02$
Gaussian	4.79 ± 0.06
Gaussian + DCTD	4.66 ± 0.04
Laplace	4.85 ± 0.04
Laplace + DCTD	4.81 ± 0.04
Softmax - CE & L2	4.78 ± 0.05
Softmax - CE & L2 + DCTD	$\mathbf{4.65} \pm 0.04$
Softmax - CE, L2 & Var	4.81 ± 0.03
Softmax - CE, L2 & Var + DCTD	4.69 ± 0.03

APPENDIX E HEAD-POSE ESTIMATION

In this appendix, further details on the head-pose estimation experiments (Section 4.3) are provided.

E.1 DCTD NETWORK ARCHITECTURE

The DNN architecture $f_\theta(x, y)$ of the DCTD 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: $3 \rightarrow 16$, $16 \rightarrow 32$, $32 \rightarrow 64$, $64 \rightarrow 128$), generating $g_y \in \mathbb{R}^{128}$. The two feature vectors g_x, g_y are then concatenated to form $f_{x,y} \in \mathbb{R}^{2048+128}$, 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 DCTD TRAINING

The DCTD model is trained using $M = 1024$ samples from a proposal distribution $q(y|y_i)$ (equation 5) with $L = 2$ and variances $\sigma_1^2 = 1^2$, $\sigma_2^2 = 20^2$ for Yaw, Pitch and Roll. It 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 64×64. 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 64×64 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.

E.3 DCTD PREDICTION

For this experiment, we also use the prediction procedure detailed in Algorithm 2. 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.

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 DCTD 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.

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

$$p(y|x,\theta) = \mathcal{N}\big(y; \mu_\theta(x), \Sigma_\theta(x)\big), \quad \Sigma_\theta(x) = \mathrm{diag}\big(\sigma_\theta^2(x)\big),$$

$$y = [\,y_1 \quad y_2 \quad y_3\,]^\mathsf{T} \in \mathbb{R}^3,$$

$$\mu_\theta(x) = [\,\mu_{1,\theta}(x) \quad \mu_{2,\theta}(x) \quad \mu_{3,\theta}(x)\,]^\mathsf{T} \in \mathbb{R}^3,$$

$$\sigma_\theta^2(x) = [\,\sigma_{1,\theta}^2(x) \quad \sigma_{2,\theta}^2(x) \quad \sigma_{3,\theta}^2(x)\,]^\mathsf{T} \in \mathbb{R}^3,$$

$$f_\theta(x) = [\,\mu_\theta(x)^\mathsf{T} \quad \log \sigma_\theta^2(x)^\mathsf{T}\,]^\mathsf{T} \in \mathbb{R}^6,$$

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

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \bigg(\sum_{k=1}^{3} \frac{(y_{k,i} - \mu_{k,\theta}(x_i))^2}{\sigma_{k,\theta}^2(x_i)} + \log \sigma_{k,\theta}^2(x_i) \bigg).$$

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 ($2048 \rightarrow 2048$, $2048 \rightarrow 3$) to output $\mu_\theta(x) \in \mathbb{R}^3$ and $\log \sigma_\theta^2(x) \in \mathbb{R}^3$. The mean $\mu_\theta(x)$ is taken as the prediction $\hat{y}$.

Laplace   Following Gast & Roth (2018), the Laplace model is defined using a DNN $f_\theta(x)$ according to,

$$p(y|x,\theta) = \prod_{k=1}^{3} \beta_{k,\theta}(x)^{-\frac{1}{2}} \exp\bigg\{ -\frac{1}{2} \bigg(\sum_{k=1}^{3} \frac{(y_k - \mu_{k,\theta}(x))^2}{\beta_{k,\theta}(x)} \bigg)^{\frac{1}{2}} \bigg\},$$

$$y = [\,y_1 \quad y_2 \quad y_3\,]^\mathsf{T} \in \mathbb{R}^3,$$

$$\mu_\theta(x) = [\,\mu_{1,\theta}(x) \quad \mu_{2,\theta}(x) \quad \mu_{3,\theta}(x)\,]^\mathsf{T} \in \mathbb{R}^3,$$

$$\beta_\theta(x) = [\,\beta_{1,\theta}(x) \quad \beta_{2,\theta}(x) \quad \beta_{3,\theta}(x)\,]^\mathsf{T} \in \mathbb{R}^3,$$

$$f_\theta(x) = [\,\mu_\theta(x)^\mathsf{T} \quad \log \beta_\theta(x)^\mathsf{T}\,]^\mathsf{T} \in \mathbb{R}^6,$$

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

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

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 ($2048 \rightarrow 2048$, $2048 \rightarrow 3$) 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 ($2048 \rightarrow 2048$, $2048 \rightarrow C$), outputting logits for $C = 151$ discretized classes $\{-75, -74, \ldots, 75\}$ 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 (Pan et al., 2018) 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.

E.5   FULL RESULTS

Full experiment results, expanding the results found in Table 3 (Section 4.3), are provided in Table 6.

APPENDIX F   VISUAL TRACKING

Here, we provide further details about the training procedure and hyperparameters used for our experiments on visual object tracking (Section 4.4).

Table 6: Full results for the head-pose estimation experiments. Refinement using DCTD 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	Av. MAE
SSR-Net-MD (Yang et al., 2018)	4.24	4.35	4.19	4.26
VGG16 (Gu et al., 2017)	3.91	4.03	3.03	3.66
FSA-Caps-F (Yang et al., 2019)	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 + DCTD	2.75 ± 0.08	3.70 ± 0.11	2.87 ± 0.09	3.11 ± 0.06
Direct - L2	2.81 ± 0.08	$\mathbf{3.60} \pm 0.14$	2.85 ± 0.08	3.09 ± 0.07
Direct - L2 + DCTD	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 + DCTD	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 + DCTD	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 + DCTD	$\mathbf{2.67} \pm 0.08$	3.61 ± 0.12	$\mathbf{2.75} \pm 0.10$	$\mathbf{3.01} \pm 0.07$
Softmax - CE, L2 & Var	2.83 ± 0.12	3.79 ± 0.10	2.84 ± 0.11	3.15 ± 0.07
Softmax - CE, L2 & Var + DCTD	2.76 ± 0.10	3.74 ± 0.09	2.83 ± 0.10	3.11 ± 0.06

F.1 TRAINING

We adopt the ATOM (Danelljan et al., 2019) tracker as our baseline, and use the PyTorch implementation and pre-trained weights from Danelljan & Bhat (2019). 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 DCTD model is instead trained by sampling 128 candidate boxes from a proposal distribution (equation 5) generated by $L = 2$ Gaussians with standard deviations of 0.05 and 0.5, and minimizing the negative log likelihood of the training data. We use the training splits of TrackingNet (Müller et al., 2018), LaSOT (Fan et al., 2019), GOT10k (Huang et al., 2018), and MS-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.

F.2 INFERENCE

During tracking, the ATOM tracker first apply 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 DCTD-based 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.