How to Train Your Energy-Based Model for Regression - Supplementary Material

Fredrik K. Gustafsson[1]
fredrik.gustafsson@it.uu.se
Martin Danelljan[2]
martin.danelljan@vision.ee.ethz.ch
Radu Timofte[2]
radu.timofte@vision.ee.ethz.ch
Thomas B. Schön[1]
thomas.schon@it.uu.se

[1] Department of Information Technology
Uppsala University
Sweden
[2] Computer Vision Lab
ETH Zürich
Switzerland

In this supplementary material, we provide additional details and results. It consists of Appendix A - Appendix D. Appendix A contains a detailed algorithm for our employed prediction strategy. Further experimental details are provided in Appendix B for 1D regression, and in Appendix C for object detection. Lastly, Appendix D contains details and further results for the visual tracking experiments. 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 2.2) is detailed in Algorithm S1, where λ denotes the gradient ascent step-length, η is a decay of the step-length and T is the number of iterations.

Algorithm S1 Prediction via gradient-based refinement.

```
Input: $x^\star$, $\hat{y}$, $T$, $\lambda$, $\eta$.
1: $y \leftarrow \hat{y}$.
2: for $t = 1, \dots, T$ 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:         $\lambda \leftarrow \eta\lambda$.
10: Return $y$.
```

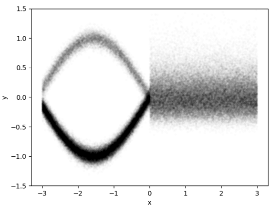
Figure S1: Visualization of the true $p(y|x)$ for the first 1D regression dataset.

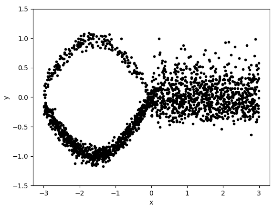
Figure S2: Training data $\{(x_i, y_i)\}_{i=1}^{2000}$ for the first 1D regression dataset.

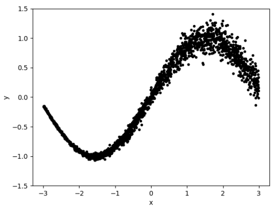
Figure S3: Training data $\{(x_i, y_i)\}_{i=1}^{2000}$ for the second 1D regression dataset.

Appendix B 1D Regression

Here, we provide details on the two synthetic datasets, the network architecture, the evaluation procedure, and hyperparameters used for our 1D regression experiments (Section 4.1). For all seven training methods, the DNN $f_\theta(x, y)$ was trained (by minimizing the associated loss $J(\theta)$) for 75 epochs with a batch size of 32 using the ADAM [11] optimizer.

B.1 Datasets

The ground truth $p(y|x)$ for the first dataset is visualized in Figure S1. It 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 $\mathcal{D}_1 = \{(x_i, y_i)\}_{i=1}^{2000}$ was generated by uniform random sampling of x in the interval $[-3, 3]$, and is visualized in Figure S2. The ground truth $p(y|x)$ for the second dataset is defined according to,

$$\begin{gathered} p(y|x) = \mathcal{N}\big(y; \mu(x), \sigma^2(x)\big), \\ \mu(x) = \sin(x), \quad \sigma(x) = 0.15(1 + e^{-x})^{-1}. \end{gathered} \tag{S1}$$

The training data $\mathcal{D}_2 = \{(x_i, y_i)\}_{i=1}^{2000}$ was generated by uniform random sampling of x in the interval $[-3, 3]$, and is visualized in Figure S3.

B.2 Network Architecture

The DNN $f_\theta(x, y)$ is a feed-forward network taking $x \in \mathbb{R}$ and $y \in \mathbb{R}$ as inputs. It consists of two fully-connected layers (dimensions: $1 \to 10$, $10 \to 10$) for x, one fully-connected layer ($1 \to 10$) for y, and four fully-connected layers ($20 \to 10$, $10 \to 10$, $10 \to 10$, $10 \to 1$) processing the concatenated (x, y) feature vector.

B.3 Evaluation

The training methods are evaluated in terms of the KL divergence $D_{\mathrm{KL}}(p(y|x) \parallel p(y|x;\theta))$ between the learned EBM $p(y|x;\theta) = e^{f_\theta(x,y)} / \int e^{f_\theta(x,\tilde{y})} d\tilde{y}$ and the true conditional density $p(y|x)$. To approximate $D_{\mathrm{KL}}(p(y|x) \parallel p(y|x;\theta))$, we compute $e^{f_\theta(x,y)}$ and $p(y|x)$ for all (x, y) pairs in a 2048×2048 uniform grid in the region $\{(x, y) \in \mathbb{R}^2 : x \in [-3, 3], y \in [-3, 3]\}$. We then normalize across all values associated with each x, employ the formula for KL

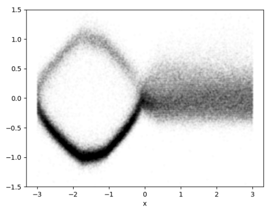

Figure S4: Example of $p(y|x;\theta)$ trained with NCE.

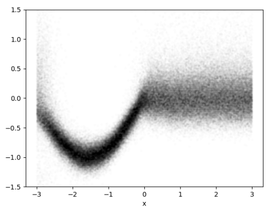

Figure S5: Example of $p(y|x;\theta)$ trained with DSM.

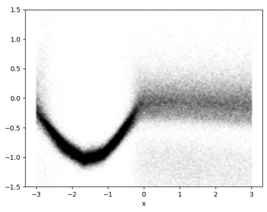

Figure S6: Example of $p(y|x;\theta)$ trained with SM.

divergence between two discrete distributions $q_1(y)$ and $q_2(y)$,

$$D_{\text{KL}}(q_1 \parallel q_2) = \sum_{y \in \mathcal{Y}} q_1(y) \log \frac{q_1(y)}{q_2(y)}, \tag{S2}$$

and finally average over all 2048 values of x. For each dataset and training method, we independently train the DNN $f_\theta(x,y)$ and compute $D_{\text{KL}}(p(y|x) \parallel p(y|x;\theta))$ 20 times. We then take the mean of the 5 best runs, and finally average this value for the two datasets.

B.4 Hyperparameters

The number of samples $M = 1024$ for all applicable training methods. All other hyperparameters were selected to optimize the performance, evaluated according to Section B.3.

ML-IS Following [7], we set $K = 2$ in the proposal distribution $q(y|y_i)$ in (4). After ablation, we set $\sigma_1 = 0.2$, $\sigma_2 = 1.6$.

KLD-IS We use the same proposal distribution $q(y|y_i)$ as for ML-IS. After ablation, we set $\sigma = 0.025$ in $p(y|y_i) = \mathcal{N}(y; y_i, \sigma^2 I)$.

ML-MCMC After ablation, we set the Langevin dynamics step-length $\alpha = 0.05$.

NCE To match ML-IS, we set $K = 2$ in the noise distribution $p_N(y|y_i)$ in (11). After ablation, we set $\sigma_1 = 0.1$, $\sigma_2 = 0.8$.

DSM After ablation, we set $\sigma = 0.2$ in $p_\sigma(\tilde{y}|y_i) = \mathcal{N}(\tilde{y}; y_i, \sigma^2 I)$.

NCE+ We use the same noise distribution $p_N(y|y_i)$ as for NCE. After ablation, we set $\beta = 0.025$.

B.5 Qualitative Results

An example of $p(y|x;\theta)$ trained using NCE on the first dataset is visualized in Figure S4. As can be observed, this is quite close to the true $p(y|x)$ visualized in Figure S1. Similar results are obtained with all four top-performing training methods. Examples of $p(y|x;\theta)$ instead trained using DSM and SM are visualized in Figure S5 and Figure S6, respectively. These do not approximate the true $p(y|x)$ quite as well, matching the worse performance in terms of D_{KL} reported in Table 1.

	ML-IS	ML-MCMC-1	ML-MCMC-4	ML-MCMC-8	KLD-IS	NCE	DSM	**NCE+**
λ_{pos}	0.0004	0.000025	0.000025	0.000025	0.0004	0.0004	0.000025	0.0008
λ_{size}	0.0016	0.0001	0.0001	0.0001	0.0016	0.0016	0.0001	0.0032

Table S1: Used step-lengths λ_{pos} and λ_{size} for the object detection experiments.

σ	0.0075	0.015	0.0225	0.03	0.0375
AP (%) $\uparrow$	38.32	39.19	**39.38**	39.33	39.23

Table S2: Ablation study for KLD-IS, on the *2017 val* split of COCO [13].

α	0.000001	0.00001	0.0001
AP (%) $\uparrow$	36.14	**36.19**	36.04

Table S3: Ablation study for ML-MCMC-1, on the *2017 val* split of COCO [13].

Appendix C Object Detection

Here, we provide details on the prediction procedure and hyperparameters used for our object detection experiments (Section 4.2). We employ an identical network architecture and training procedure as described in [7], only modifying the loss when using a different method than ML-IS to train $f_\theta(x,y)$.

C.1 Prediction

Predictions $y^\star$ are produced by performing guided NMS [9] followed by gradient-based refinement (Algorithm S1), taking the Faster-RCNN detections as initial estimates $\hat{y}$. As in [7], we run $T = 10$ gradient ascent iterations. We fix the step-length decay to $\eta = 0.5$, which is the value used in [7]. For each trained model, we select the gradient ascent step-length λ to optimize performance in terms of AP on the *2017 val* split of COCO [13]. Like [7], we use different step-lengths for the bounding box position (λ_{pos}) and size (λ_{size}). We start this ablation with $\lambda_{\text{pos}} = 0.0001$, $\lambda_{\text{size}} = 0.0004$. The used step-lengths for all training methods are given in Table S1.

C.2 Hyperparameters

The number of samples $M = 128$ for all applicable training methods. All other hyperparameters were selected to optimize performance in terms of AP on the *2017 val* split of COCO [13].

ML-IS Following [7], we set $K = 3$ in the proposal distribution $q(y|y_i)$ in (4) with $\sigma_1 = 0.0375$, $\sigma_2 = 0.075$, $\sigma_3 = 0.15$.

KLD-IS We use the same proposal distribution $q(y|y_i)$ as for ML-IS. Based on the ablation study in Table S2, we set $\sigma = 0.0225$ in $p(y|y_i) = \mathcal{N}(y; y_i, \sigma^2 I)$.

ML-MCMC Based on the ablation study in Table S3, we set the Langevin dynamics step-length $\alpha = 0.00001$.

NCE To match ML-IS, we set $K = 3$ in the noise distribution $p_N(y|y_i)$ in (11). Based on the ablation study in Table S4, we set $\sigma_1 = 0.075$, $\sigma_2 = 0.15$, $\sigma_3 = 0.3$.

$\{\sigma_k\}_{k=1}^3$	$\{0.0125, 0.025, 0.05\}$	$\{0.025, 0.05, 0.1\}$	$\{0.05, 0.1, 0.2\}$	$\{0.075, 0.15, 0.3\}$	$\{0.1, 0.2, 0.4\}$
AP (%) $\uparrow$	38.58	38.95	39.12	**39.17**	39.05

Table S4: Ablation study for NCE, on the *2017 val* split of COCO [13].

σ	0.0375	0.075	0.15
AP (%) ↑	36.11	**36.12**	36.05

Table S5: Ablation study for DSM, on the *2017 val* split of COCO [13].

β	0.05	0.1	0.15
AP (%) ↑	39.27	**39.36**	39.32

Table S6: Ablation study for NCE+, on the *2017 val* split of COCO [13].

	ML-IS	ML-MCMC-1	ML-MCMC-4	ML-MCMC-8	KLD-IS	NCE	DSM	**NCE+**
AP (%) ↑	39.11	36.19	36.24	36.25	**39.38**	39.17	36.12	39.36
AP_{50}(%) ↑	57.95	57.34	57.45	57.28	**58.07**	57.96	57.29	57.99
AP_{75}(%) ↑	41.97	38.77	38.81	38.88	42.47	42.07	38.84	**42.63**
Training Cost ↓	1.03	2.47	7.05	13.3	**1.02**	1.04	3.84	1.09

Table S7: Comparison of training methods for the object detection experiments, on the *2017 val* split of COCO [13]. NCE+ and KLD-IS achieve the best performance.

DSM Based on the ablation study in Table S5, we set $\sigma = 0.075$ in $p_\sigma(\tilde{y}|y_i) = \mathcal{N}(\tilde{y}; y_i, \sigma^2 I)$.

NCE+ We use the same noise distribution $p_N(y|y_i)$ as for NCE. Based on the ablation study in Table S6, we set $\beta = 0.1$.

C.3 Detailed Results

A comparison of the training methods on the *2017 val* split of COCO [13] is provided in Table S7.

Appendix D Visual Tracking

Here, we provide detailed results and hyperparameters for our visual tracking experiments (Section 5). We employ an identical network architecture, training procedure and prediction procedure for DiMP-KLD-IS, DiMP-NCE and DiMP-NCE+, only the loss is modified.

D.1 Training Parameters

DiMP-KLD-IS is obtained by combining the DiMP [3] method for center point regression with the PrDiMP [5] bounding box regression approach, and modifying a few training parameters. Specifically, we change the batch size from 10 to 20, we change the LaSOT sampling weight from 0.25 to 1.0, we change the number of samples per epoch from 26000 to 40000, and we add random horizontal flipping with probability 0.5. Since we increase the batch size, we also freeze conv1, layer1 and layer2 of the ResNet backbone to save memory.

	SiamFC [1]	MDNet [16]	UPDT [2]	DaSiamRPN [19]	ATOM [4]	SiamRPN++ [12]	DiMP [3]	SiamRCNN [17]	PrDiMP [5]	DiMP-KLD-IS	DiMP-NCE	**DiMP-NCE+**
Precision ↑	53.3	56.5	55.7	59.1	64.8	69.4	68.7	**80.0**	70.4	73.3	69.8	73.7
Norm. Prec. ↑	66.6	70.5	70.2	73.3	77.1	80.0	80.1	**85.4**	81.6	83.5	82.4	83.7
Success (AUC) ↑	57.1	60.6	61.1	63.8	70.3	73.3	74.0	**81.2**	75.8	78.1	77.1	78.7

Table S8: Full results on the TrackingNet [15] test set, in terms of precision, normalized precision, and success (AUC). Our proposed DiMP-NCE+ is here only outperformed by the very recent SiamRCNN [17]. SiamRCNN is however slower than DiMP-NCE+ (5 FPS vs 30 FPS) and employs a larger backbone network (ResNet101 vs ResNet50).

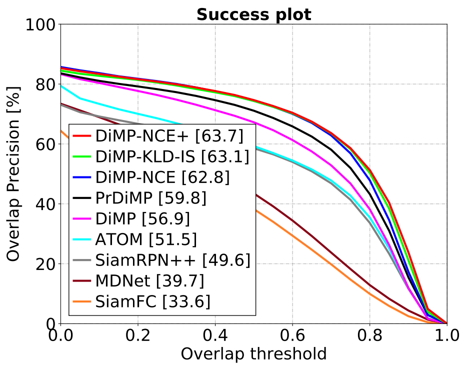

Figure S7: Success plot on LaSOT [6].

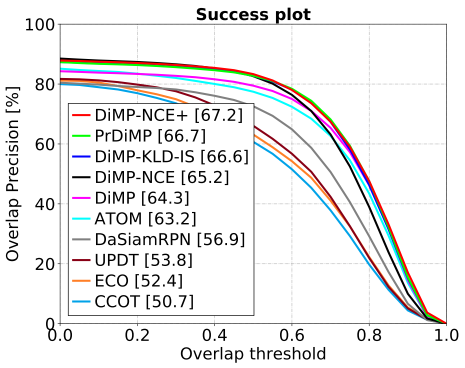

Figure S8: Success plot on UAV123 [14].

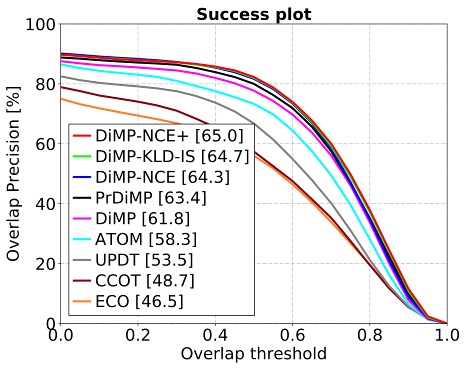

Figure S9: Success plot on NFS [10].

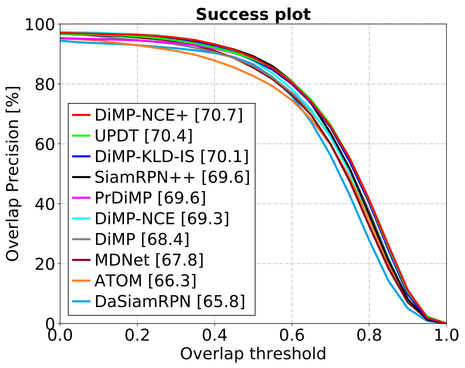

Figure S10: Success plot on OTB-100 [18].

D.2 Hyperparameters

The number of samples $M = 128$ for all three training methods.

DiMP-KLD-IS Following PrDiMP, we set $K = 2$ in the proposal distribution $q(y|y_i)$ in (4) with $\sigma_1 = 0.05$, $\sigma_2 = 0.5$, and we set $\sigma = 0.05$ in $p(y|y_i) = \mathcal{N}(y; y_i, \sigma^2 I)$.

DiMP-NCE Matching DiMP-KLD-IS, we set $K = 2$ in the noise distribution $p_N(y|y_i)$ in (11) with $\sigma_1 = 0.05$, $\sigma_2 = 0.5$. A quick ablation study on the validation set of GOT-10k [8] did not find values of σ_1, σ_2 resulting in improved performance.

DiMP-NCE+ We use the same noise distribution $p_N(y|y_i)$ as for NCE. We set $\beta = 0.1$, as this corresponded to the best performance on the object detection experiments (Table S6).

D.3 Detailed Results

Full results on the TrackingNet [15] test set, in terms of all three TrackingNet metrics, are found in Table S8. Success plots for LaSOT, UAV123, NFS and OTB-100 are found in Figure S7-S10, showing the overlap precision OP_T as a function of the overlap threshold T.

References

[1] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision (ECCV) Workshops*, 2016.

[2] Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 483–498, 2018.

[3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 6182–6191, 2019.

[4] 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)*, pages 4660–4669, 2019.

[5] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7183–7192, 2020.

[6] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5374–5383, 2019.

[7] Fredrik K Gustafsson, Martin Danelljan, Goutam Bhat, and Thomas B Schön. Energy-based models for deep probabilistic regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, August 2020.

[8] Lianghua Huang, Xin Zhao, and Kaiqi Huang. GOT-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[9] 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)*, pages 784–799, 2018.

[10] Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1125–1134, 2017.

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

[12] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4282–4291, 2019.

[13] 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)*, pages 740–755, 2014.

[14] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for UAV tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 445–461, 2016.

[15] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. TrackingNet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 300–317, 2018.

[16] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4293–4302, 2016.

[17] Paul Voigtlaender, Jonathon Luiten, Philip HS Torr, and Bastian Leibe. Siam R-CNN: Visual tracking by re-detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6578–6588, 2020.

[18] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(9):1834–1848, 2015.

[19] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 101–117, 2018.