Improving Face Pose Estimation using Long-Term Temporal Averaging for Stochastic Optimization

Nikolaos Passalis and Anastasios Tefas

Department of Informatics
Aristotle University of Thessaloniki, Thessaloniki, Greece
passalis@csd.auth.gr,tefas@aiia.csd.auth.gr

Abstract. Among the most crucial components of an intelligent system capable of assisting drone-based cinematography is estimating the pose of the main actors. However, training deep CNNs towards this task is not straightforward, mainly due to the noisy nature of the data and instabilities that occur during the learning process, significantly slowing down the development of such systems. In this work we propose a temporal averaging technique that is capable of stabilizing as well as speeding up the convergence of stochastic optimization techniques for neural network training. We use two face pose estimation datasets to experimentally verify that the proposed method can improve both the convergence of training algorithms and the accuracy of pose estimation. This also reduces the risk of stopping the training process when a bad descent step was taken and the learning rate was not appropriately set, ensuring that the network will perform well at any point of the training process.

Keywords: Pose Estimation, Deep Convolutional Neural Networks, Stochastic Optimization

1 Introduction

Unmanned Aerial Vehicles (UAVs), or *drones*, are capable of capturing spectacular aerial shots of various events, e.g., athletic events, concerts, etc., that would be otherwise difficult and expensive to obtain. This has led many media producers to use drones to film outdoor events instead of resorting to other more established, yet expensive means, such as hiring a helicopter and a crew to film the aerial shots. However, flying a drone in a professional filming setting requires at least two operators. The first one is responsible for controlling the flight path of the drone and avoiding possible hazards, while the second one for controlling the main shooting camera. Assisting parts of the flying and the shooting process using machine learning techniques, such as deep neural networks [11], and reinforcement learning techniques [14], has the potential to reduce the load of the human operators, increasing the quality of the captured shots and reducing the cost of the film productions.

Among the most crucial components of an intelligent system capable of assisting the filming process is estimating the pose of the main actors. For example, consider the problem of controlling the camera during a bicycle race event. After detecting a bicycle the drone must rotate its camera towards the bicyclist. To this end, the face pose, i.e., tilt and pan, of the bicyclist must be estimated in order to calculate the appropriate shooting angle according to the specifications of each shot type. Simple computer vision methods can be used to tackle this problem [24]. However, with the advent of deep convolutional neural networks (CNNs) it was established that it is possible to train CNNs capable of performing pose estimation significantly better than the conventional pose estimation methods [23].

Note that training deep CNNs is not always a straightforward task, slowing down the development of such systems. Several methods have been proposed to stabilize and smoothen the convergence of the training procedure [5], [6], [7], [9], [22]. During our experiments we also observed instabilities during the training process, even though a state-of-the-art stochastic optimization technique, the Adaptive Moment Estimation method (ADAM) [9], was used. If a slightly larger learning rate than the optimal was selected the training process was unstable. On the other hand, if the learning rate was too small, the optimization process slowed down significantly. Furthermore, the unstable behavior of CNNs used for pose estimation can be also partially attributed to the noisy nature of the data. After detecting a face using an object detector, such as the YOLO detector [19], or the SSD detector [13], the bounding box of the face is cropped, resized and then fed to the pose estimation CNN. However, the object detector is usually incapable of perfectly centering and determining the bounds of the face introducing a significant amount of noise into the aforementioned process.

The main contribution of this work is the proposal of a temporal averaging technique that is capable of stabilizing as well as speeding up the convergence of stochastic optimization for neural network training. The proposed technique uses an exponential running average on the parameters of the neural network to bias the current parameters towards a stabler state. As we show in Section 3, this is equivalent to first taking big descent steps to explore the solution space and then annealing towards stabler states. It was experimentally verified using two face pose estimation datasets that the proposed method can improve both the convergence of the utilized training algorithms and the accuracy of pose estimation. The more stable convergence of the algorithm also reduces the risk of stopping the training process when a bad descent step was taken and the learning rate was not appropriately set, ensuring that the network will perform well at any point of the training process (after a certain number of iterations have been performed).

The rest of the paper is structured as follows. First, the related work is briefly introduced and discussed in Section 2. Then, the proposed method is presented in detail in Section 3 and the experimental evaluation is provided in Section 4. Finally, conclusions are drawn and future work is discussed in Section 5.

2 Related Work

Several methods have been proposed for training deep neural networks as well as for improving the convergence of stochastic gradient descent. For example, using batch normalization [7], rectifier activation units [5], and residual connections [6], [22], allows for effectively dealing with the problem of *vanishing gradients*. Furthermore, advanced optimization techniques, such as the ADAGRAD algorithm [1], and the ADAM algorithm [9], adjust the learning rate for each parameter separately effectively dealing with gradients of different magnitudes and allowing for improving the convergence speed. Each of these techniques deal with a specific problem that arises during the training of deep neural networks. The method proposed in this paper is complementary to these methods since it addresses a different problem, i.e., improves the stability of the training procedure regardless of the selected learning rate. This is also demonstrated in Section 4 where the proposed method is combined with some of the aforementioned methods to improve the convergence speed and the stability of the training process.

Parameter averaging techniques were also proposed in some works to deal with the noisy stochastic updates of gradient descent [18], [20], where after completing the training of the neural network the weights of the network are replaced with the average weights, as calculated during the training process. A more deliberate technique was proposed in [9], where an exponential moving average over the parameters of the network is used to ensure that higher weight is given to the recent states of the network. A similar approach is also adopted for deep reinforcement learning tasks [12]. The method proposed in this paper is different from the method proposed in [9], since the weights of the network are not averaged after each iteration. Instead, a number of descent steps are taken, e.g., 10 optimization steps, and after them the parameters of the networks are updated. This allows for better exploring the solution space and increasing the convergence speed, while maintaining the stability that the averaging process offers, as demonstrated in Section 4.

3 Proposed Method

In this Section the used notation is introduced and the proposed Long-Term Temporal Averaging (LT-TA) algorithm is presented in detail. Then, we examine the behavior of the LT-TA algorithm by analyzing the proposed parameter update technique.

Let θ denote the parameters of the neural network that is to be optimized towards minimizing a loss function $\mathcal{L}$. The notation θ_t is used to denote the parameters after t optimization iterations. Also, let $f(\theta, x, \eta)$ be an optimization method that provides the updates for the parameters of the neural network, where x is the training data and η the hyper-parameters of the optimization method, e.g., the learning rate. Any optimization technique can be used ranging from the simple Stochastic Gradient Descent method [4], to more advanced techniques, such as the ADAM method [9].

Algorithm 1 Long-Term Temporal Averaging Algorithm

Input: A training set of data $\mathcal{X}$, the initial parameters of the network θ_0, and an optimization method $f(\cdot)$ along with its hyper-parameteres η

Parameters: the target update rate α_∞, the update rate decay m, the exploration window N_S, and the number of iterations N

Output: The optimized parameters θ

1: **procedure** PAST ALGORITHM
2: $\theta_{past} \leftarrow \theta_0$
3: **for** $t \leftarrow 1; t \leq N; t++$ **do**
4: Sample a batch x from $\mathcal{X}$
5: $\theta_t \leftarrow f(\theta_{t-1}, x, \eta)$
6: **if** $\mod(t, N_S) = 0$ **then**
7: $\alpha \leftarrow \alpha_\infty + (1 - \alpha_\infty)\exp(-m(t/Ns))$
8: $\theta_{past} \leftarrow \alpha\theta_t + (1 - \alpha)\theta_{past}$
9: $\theta_t \leftarrow \theta_{past}$
return θ_{past}

The proposed method is shown in Algorithm 1. The proposed algorithm keeps track of a exponentially averaged version of the parameters of the network denoted by θ_{past}. These parameters are initially set to the current state of the network (line 2) and represent the stable state of the network. During the optimization (lines 3-5) the proposed algorithm performs regular optimization updates (line 5). However, every N_S iterations the current weights of the network are annealed towards the previous (past) stable state θ_{past}. As we demonstrate later in this Section this is equivalent to performing large exploration descent steps during the N_S iterations and then slowing down the learning in order to update the stable version of the network. The rate a, used for the updating the stable parameters of the network (line 8), is determined using an exponential decay strategy (line 7). During the initial iterations less weight is given to the past state of the network, since usually we start with a randomly initialized neural network. However, as the optimization progress the past weights of the network converge towards their stable state. Therefore, the update rate is decayed towards a_∞. For all the conducted experiments, a_∞ is set to 0.5, while the decay rate m is set to 10^{-3}. Thus, for the initial iterations the update rate is close to 1, while as the training procedure converges it is slowly decayed to 0.5. After the performing N iterations the algorithm returns the stable version of the optimized parameters θ_{past}.

To better understand how the proposed algorithm works consider the first N_S iterations when the simple stochastic gradient descent algorithm with learning

rate η is used to provide the optimization updates:

$$\begin{aligned}
\theta_1 &= \theta_0 - \eta \frac{\partial \mathcal{L}}{\partial \theta_0} \\
\theta_2 &= \theta_1 - \eta \frac{\partial \mathcal{L}}{\partial \theta_1} \\
&\vdots \\
\theta_{N_S-1} &= \theta_{N_S-2} - \eta \frac{\partial \mathcal{L}}{\partial \theta_{N_S-2}} \\
\theta_{N_S} &= \theta_{N_S-1} - \eta \frac{\partial \mathcal{L}}{\partial \theta_{N_S-1}}
\end{aligned} \tag{1}$$

It is easy to see that after N_S optimization steps the weights of the network can be expressed as a weighted sum over the descent steps:

$$\theta_{N_S} = \theta_{past} - \eta \sum_{i=0}^{N_S-1} \frac{\partial \mathcal{L}}{\partial \theta_i} \tag{2}$$

since $\theta_{past} = \theta_0$. After these iterations the exponentially averaged copy of the parameters of the network is updated (line 8 of Algorithm 1) as:

$$\theta_{past} = (1-\alpha)\theta_{past} + \alpha\theta_{N_S-1} = \theta_{past} - \alpha\eta \sum_{i=0}^{N_S} \frac{\partial \mathcal{L}}{\partial \theta_i} \tag{3}$$

Therefore, updating the exponentially averaged copy of the parameters θ_{past} is equivalent to lowering the learning rate of the previous updates to $\alpha\eta$ while updating the parameters θ_{past}. However this is *not equivalent* to performing optimization with the lowered learning rate. To understand this note that the intermediate states $\theta_1, \theta_2, \ldots, \theta_{N_S}$ are calculated using the original learning rate η instead of the lowered rate $\alpha\eta$:

$$\theta_i = \theta_{i-1} - \eta \frac{\partial \mathcal{L}}{\partial \theta_{i-1}} \tag{4}$$

That is, during the N_S steps the proposed algorithm explores the solution space by taking large steps towards the descent direction, while the stable state θ_{past} is updated using a lowered learning rate. This increases the convergence speed, while ensuring that relatively large descent steps that overshoot the local minima will not significantly affect the stability of the training procedure.

4 Experiments

Two datasets were used to evaluate the proposed method: the Annotated Facial Landmarks in the Wild dataset (AFLW) [10], and the Head Pose Image Dataset (HPID) [3]. The Annotated Facial Landmarks in the Wild (AFLW) dataset [10],

is a large-scale dataset for facial landmark localization. The 75% of the images were used to train the models, while the rest 25% for evaluating the accuracy of the models. The face images were cropped according to the supplied annotations and then resized to 32×32 pixels. Face images smaller than 16×16 pixels were not used for training or evaluating the model. Some examples of cropped images are shown in Figure 1. The Head Pose Image Dataset (HPID) [3] is a smaller dataset that contains 2790 face images of 15 subjects in various poses taken in a constrained environment. Some sample images are shown in Figure 2. Again, all images were resized to 32×32 pixels before feeding them to the used CNN. For both datasets the horizontal pose (pan) is to be predicted. The AFLW dataset provides continuous pose targets, while the HPID dataset provides discrete targets (13 steps).

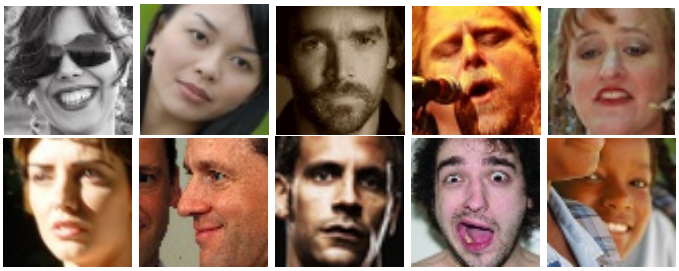

Fig. 1. Cropped face images from the ALFW dataset

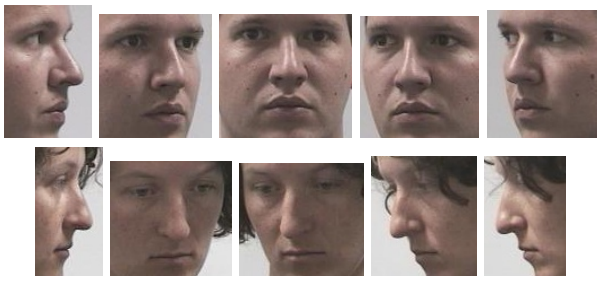

Fig. 2. Cropped face images from the HPID dataset

During the training the following data augmentation techniques were used:

1. random vertical flip with probability 0.5
2. random horizontal shift up to 5%
3. random vertical shift up to 5%

4. random zoom up to 5%
5. random rotation up to 10°

The vertical and horizontal shifts simulate the behavior of face detectors that are usually unable to perfectly align the face in the images, while the zooming and the rotation transformations increase the invariance of the network to various shooting specifications.

Deploying a deep learning model on a drone with limited processing and memory resources imposes significant restrictions on the complexity of the model. To this end, a lightweight CNN with less than 300K parameters is used. The architecture of the used deep convolutional neural network is shown in Table 1. Local contrast normalization (LCN) is used after each of the two convolutional blocks [8], while the dropout technique is used to regularize the learning process [21]. Finally, the ADAM algorithm [9], with learning rate $\eta = 0.001$, and the default hyper-parameters ($\beta_1 = 0.9$ and $\beta_2 = 0.999$) is used for optimizing the network using mini-batches of 32 samples.

Table 1. Architecture of the used CNN

Layer Type	**Output Shape**
Input	$32 \times 32 \times 3$
Convolutional (3×3)	$30 \times 30 \times 16$
Convolutional (3×3)	$28 \times 28 \times 16$
Max Pooling (2×2)	$14 \times 14 \times 16$
LCN + Dropout $(p = 0.5)$	$14 \times 14 \times 16$
Convolutional (3×3)	$12 \times 12 \times 32$
Convolutional (3×3)	$10 \times 10 \times 32$
Max Pooling (2×2)	$5 \times 5 \times 32$
LCN + Dropout $(p = 0.5)$	$5 \times 5 \times 32$
Dense	256
Dense	1

First, the ALFW dataset is used to evaluate the proposed method. The results are shown in Table 2. The proposed Long-Term Temporal Averaging method (abbreviated as LT-TA) outperforms both the plain Temporal Averaging (abbreviated as TA), as well as the baseline learning technique, i.e., using only the ADAM algorithm without any temporal averaging. For both the TA and the LT-TA techniques the weight update parameter is exponentially decayed to 0.5 (Algorithm 1). The proposed LT-TA method reduces both the training and the testing mean angular error as well as it stabilizes the learning process by reducing the error deviation during the last training iterations (the train error deviation during the last 5,000 iterations is reported). These results are also confirmed by the learning curve depicted in Figure 3, where the mean angular error during the training process is ploted. Note how the proposed LT-TA method stabilizes the

convergence of the training process (the error spikes are significantly reduced). Using a smaller learning rate can also have similar effect, but it also slows down the convergence.

The evaluation results for the HPID dataset are reported in Table 3. Again, the proposed method reduces both the train and the test error, while reducing the instabilities of the algorithm during the training process (the deviation is reduced from 0.21 to 0.11). This is also confirmed by the learning curves shown in Figure 4, where the LT-TA method reduces the fluctuations of the error (especially in the last iterations).

Table 2. ALFW Dataset Evaluation: Comparing the proposed Long-Term Temporal Averaging (LT-TA) method to the plain Temporal Averaging (TA) and the baseline learning methods. The mean angular error is reported. The train error deviation refers to the error deviation during the last 5,000 iterations.

Method	**Train Error**	**Train Deviation**	**Test Error**
Baseline	7.41	0.32	8.16
TA	7.24	0.18	8.01
LT-TA	**6.98**	**0.17**	**7.72**

Table 3. HPID Dataset Evaluation: Comparing the proposed Long-Term Temporal Averaging (LT-TA) method to the plain Temporal Averaging (TA) and the baseline learning methods. The mean angular error is reported. The train error deviation refers to the error deviation during the last 5,000 iterations.

Method	**Train Error**	**Train Deviation**	**Test Error**
Baseline	4.57	0.21	5.99
TA	4.02	0.21	5.89
LT-TA	**3.63**	**0.11**	**5.74**

5 Conclusions

In this work we proposed a Long-Term Temporal Averaging technique that first takes big descent steps to explore the solution space and then uses an exponential running average on the parameters of the neural network to bias the current parameters towards stabler states. The more stable convergence of the algorithm also reduces the risk of stopping the training process when a bad descent step was taken and the learning rate was not appropriately set, ensuring that the network will perform well at any point of the training process (after a certain number of iterations have been performed). It is demonstrated, using two face

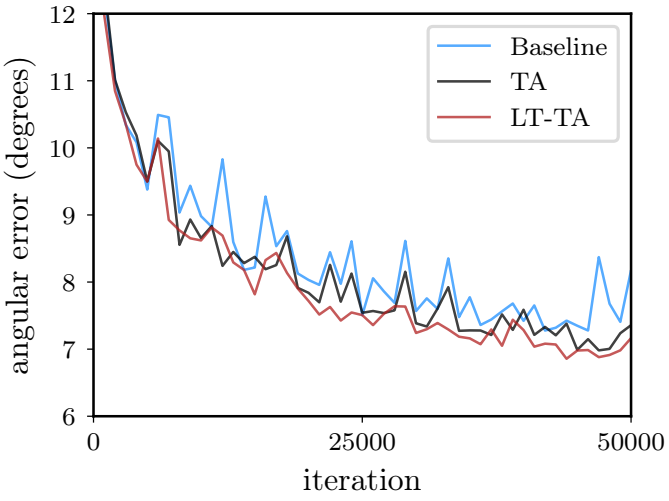

Fig. 3. ALFW Dataset Evaluation: Comparing the mean angular error during the training process

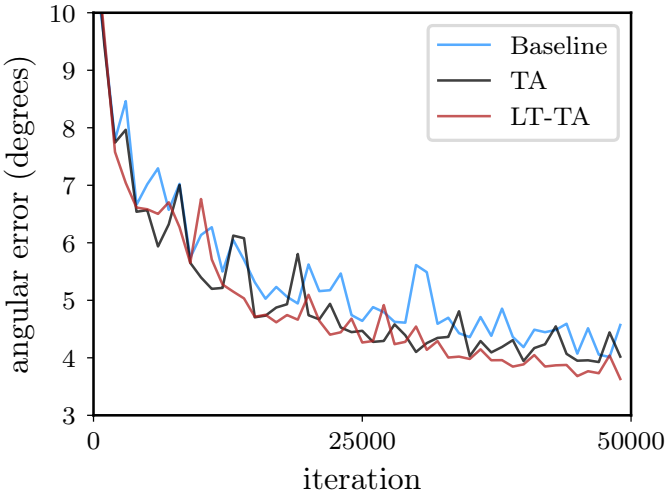

Fig. 4. HPID Dataset Evaluation: Comparing the mean angular error during the training process

image dataset for pose estimation, that the proposed technique is capable of stabilizing as well as speeding up the convergence of stochastic optimization techniques for neural network training.

There are several interesting future research directions. First, the proposed technique can be evaluated under a wider range of learning scenarios, e.g., different learning rates, network architectures, datasets, etc. This also includes several other tasks that are needed for intelligent drone-based cinematography, such as face recognition [2], [16], and learning compact representations [15], [17], that can be used to develop lightweight models for tasks running on-board. Furthermore, the update rate for the exponentially averaged parameters can be dynamically determined. For example, the relative change of the loss function can be used to adjust the update rate, or second order statistics can be also taken into account, similar to [9].

6 Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

References

1. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 12(Jul), 2121–2159 (2011)
2. Goudelis, G., Tefas, A., Pitas, I.: Emerging biometric modalities: a survey. Journal on Multimodal User Interfaces 2(3), 217–235 (2008)
3. Gourier, N., Hall, D., Crowley, J.L.: Estimating face orientation from robust detection of salient facial structures
4. Haykin, S., Network, N.: A comprehensive foundation. Neural Networks 2(2004), 41 (2004)
5. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international Conference on Computer Vision. pp. 1026–1034 (2015)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
7. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning. pp. 448–456 (2015)
8. Jarrett, K., Kavukcuoglu, K., LeCun, Y., et al.: What is the best multi-stage architecture for object recognition? In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2146–2153 (2009)
9. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

10. Koestinger, M., Wohlhart, P., Roth, P.M., Bischof, H.: Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In: First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies (2011)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)
12. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
13. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: Proceedings of the European Conference on Computer Vision. pp. 21–37 (2016)
14. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature 518(7540), 529–533 (2015)
15. Nousi, P., Tefas, A.: Deep learning algorithms for discriminant autoencoding. Neurocomputing (2017)
16. Passalis, N., Tefas, A.: Learning neural bag-of-features for large-scale image retrieval. IEEE Transactions on Systems, Man, and Cybernetics: Systems (2017)
17. Passalis, N., Tefas, A.: Neural bag-of-features learning. Pattern Recognition 64, 277–294 (2017)
18. Polyak, B.T., Juditsky, A.B.: Acceleration of stochastic approximation by averaging. SIAM Journal on Control and Optimization 30(4), 838–855 (1992)
19. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 779–788 (2016)
20. Ruppert, D.: Efficient estimations from a slowly convergent robbins-monro process. Tech. rep., Cornell University Operations Research and Industrial Engineering (1988)
21. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1), 1929–1958 (2014)
22. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. arXiv preprint arXiv:1505.00387 (2015)
23. Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1653–1660 (2014)
24. Zhu, X., Ramanan, D.: Face detection, pose estimation, and landmark localization in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2879–2886 (2012)