A Trade-off : Between High Accuracy and Energy Saving

Jiasong Guo
Shaghai Jiao Tong University
7_GUO_7@sjtu.edu.cn

Ruoyu Cheng
Shaghai Jiao Tong University
roy_account@sjtu.edu.cn

Yixiong Wang
Shaghai Jiao Tong University
wyx_ei@sjtu.edu.cn

ABSTRACT

Recent works related to neural networks have paid more attention to energy consumption. Small networks are energy saving compared with big networks yet they may sacrifice the accuracy of computation outcomes and big ones are on the contrary. Previous work proposed a big/small network system for trade-off between energy consuming and accuracy, the determiner of which receives the softmax layer as the input and output the score margin. If the score is smaller than the threshold, it calls the big network, otherwise outputs the classification result directly. Other works use machine learning methods to train the threshold using the raw data.

In this paper, we propose a machine learning based determiner that receives the outcomes of classification layer of small networks combined with data set size and complexity of small network in order to make the determiner generalized. Besides, considering the trade-off between energy consuming and accuracy, we propose a metric to evaluate the performance of such determiners. Gradient-Boost Decision Tree (GBDT), Random Forest and Decision Tree are considered in determiner selection. Experiments results suggest that the one with the best performance is GBDT. It achieves 98.19% accuracy with penalty on calling the big network (i.e., $Acc - \lambda f$), which is 1.21% better than the traditional method. We also proposed an $\varepsilon -$ greedy based algorithm to determine the threshold.

1 INTRODUCTION

Nowadays, deep neural network (DNNs) are so significant for their higher performance in large-scale problems. [6] [8]. In server and mobile applications, the increasing complexity of the problem poses a challenge to the energy efficiency of DNN implementations, as DNNs are now used not only on servers but also on many commercial mobile devices. [5] Ultra-low-power DNNs can be implemented with dedicated hardware. When DNN is implemented as dedicated hardware, its total energy consumption is usually controlled by off-chip memory access to synaptic connection information, because hardware specialization reduces the energy consumption of neuron computing. [2] [3] As the complexity of the

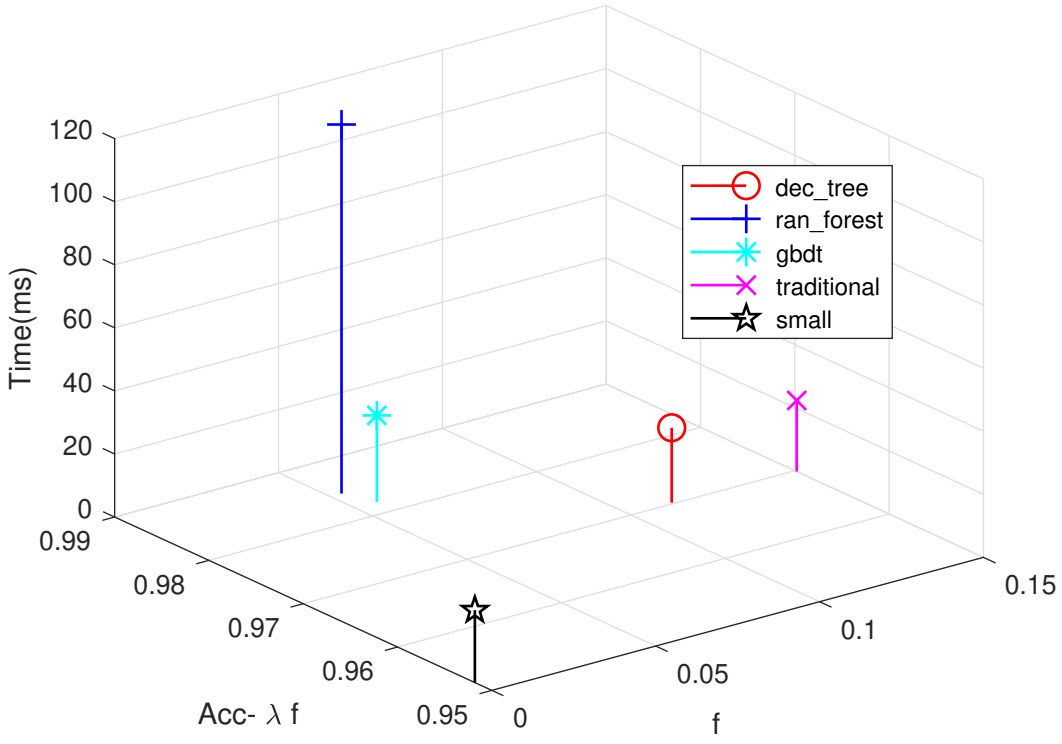

Fig. 1. The 3-D visualization for different determiners. The models are decision forest, random forest, GBDT, traditional threshold method and only small networks. The performance is evaluated by burst time, frequency to call the big network and $Acc - \lambda f$: a metric we propose in this paper. The shorter burst time, the higher $Acc - \lambda f$ and the smaller frequency indicates better performance.

network increases, this trend is expected to intensify, because large DNNs for more complex problems require larger sets of connection information, which increases the energy consumption of the memory.

An existing method for this is to design two networks, because there are many cases where DNNs with smaller sizes (which we call little DNNs) can equally perform well compared to big DNNs [11]. Given an input for inference (e.g., an input image for object recognition), the little DNN is first executed to give an inference result. If the result is believed to be accurate, then it is taken as the inference result. If it is not certain to take it as the final result, then the big DNN is utilized to give the inference result.

But determining when to use large DNNs can be a challenging problem. This is because large DNNs should be able to estimate whether the results of small DNNs perform correctly for large DNNs. Too frequent false positives will reduce the accuracy of the reasoning. On the other hand, the negative factor of excessive error is also very problematic, because the excessive energy-consuming large DNN execution will make it lose the opportunity to reduce energy.

So if we are given a relatively small network and a big network, how can we tradeoff between the small and the big is a question. Small network stands for lower accuracy but lower energy consumption while the big one stands for more accuracy but higher energy consumption. In this paper, we utilize a machine-learning method to design out determiner. We also define $Acc - \lambda f$ for the goal and $Acc - \lambda f - px$ for every single image. We propose a generalized determiner for all given small and big networks with different sizes and complexities.

2 RELATED WORK & MOTIVATION

Nowadays, DNN hardware acceleration is getting more and more attention. Chakradhar et al. propose a configurable hardware accelerator which can be adapted to different types of parallelism (convolution and intra/inter-output parallelism) in neural networks. [1] Chen et al. present a hardware implementation of neural networks called DianNao where a neuron is implemented with multipliers, a tree structure of adders, and local memory components for input data, synaptic weights, and output. They report that, when the hardware accelerator is utilized, the off-chip memory becomes the dominant source of energy consumption in DNN execution. [2] Park et al. propose a low power hardware IP core called KBrain which can perform both supervised/unsupervised learning and inference. [12]

Some of previous work has explored a trade-off between inference accuracy and energy consumption. Du et al. propose an approximate multi-layer perceptron with inexact multipliers which reduce energy consumption at a cost of degraded accuracy. [4]

In the field of deep learning architectures, there are several methods of utilizing multiple neural networks in a hierarchical structure and the construction of a neural network from coarse to fine, whole or cascade. [13] [15]

Because there is no efficient determiner, so we want to design a determiner which fits all given small network and a big network. We propose a high-performance determiner to optimize this situation (balancing between energy and accuracy) as shown in Fig 2.

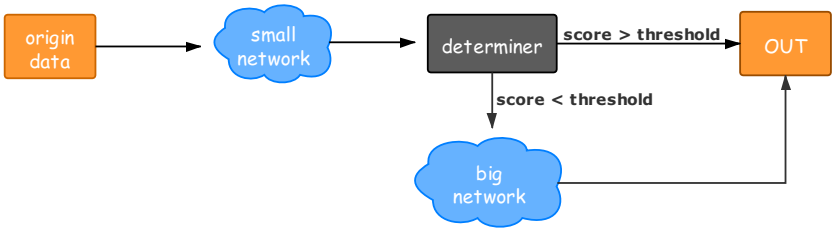

Fig. 2. **Our architecture**

3 PROPOSED APPROACH

In this section, we first propose our generalized model of the determiner adapting to various small networks and datasets. We propose a flow diagram of data pre-processing to obtain the training data for our determiner and utilize several ML-based models for training. In the end, we explore the threshold turning algorithm to strike a balance between the overall accuracy and the frequency of calling the big network (indicating the energy consumption).

3.1 Model of Determiner

Instead of making inference by the first-order score margin ($1^{st}score - 2^{nd}score$), our idea is to fully utilize the classifier layer, as well as the complexity of small network and size of the dataset. The input $\mathbf{x}$ for our determiner is supposed to be $(class_1, class_2, ..., class_{10}, complexity_1, complexity_2, size)$, which is the concatenation of three factors mentioned above. And the output $\mathbf{y} = score \in (0,1)$ is the confidence of not calling the big network. Since our target is not maximizing the accuracy but rather balancing the accuracy and energy consumption, selecting a reasonable label for the train set is the key step. A

natural intuition is to label each sample as the correctness of the small network output, but this is not energy-efficient due to the tendency of calling big network whenever the small network may go wrong. Then $Acc - \lambda f$ would be suitable for the trade-off, but these two features are only for a batch of samples. Finally, we combine two ideas and label each sample as $Acc - \lambda f - px$ (see Fig.3), where f denotes the frequency of calling the big network, and px is a penalty for wrong prediction.

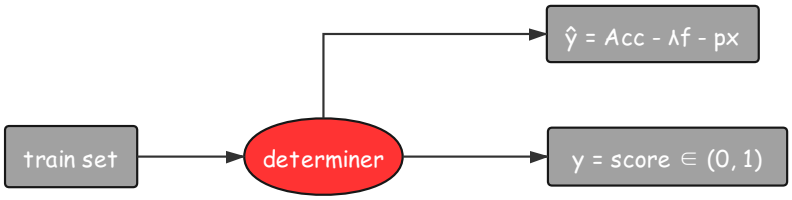

Fig. 3. **The model of determiner**

3.2 Data Pre-processing and Determiner Training

Different from those end-to-end models, our determiner needs specialized inputs and labels, thus data pre-processing technique is required to obtain the train set. To enhance the generalization ability, we randomly pick $1, 2, ..., 6 \times 10000$ images to train 8 small networks with different structures respectively. Afterwards, 10000 test images generate 10000 samples, each sample is labeled with the corresponding x together with same accuracy Acc and frequency f. We leave the last 10000 samples as test set, and 470000 samples are used to train our determiner. The flow diagram of data pre-processing is shown in Fig4. It is noticeable that currently we are still using the first-order score margin as determiner, so a grid search for finding the optimal threshold is necessary for more precise labels.

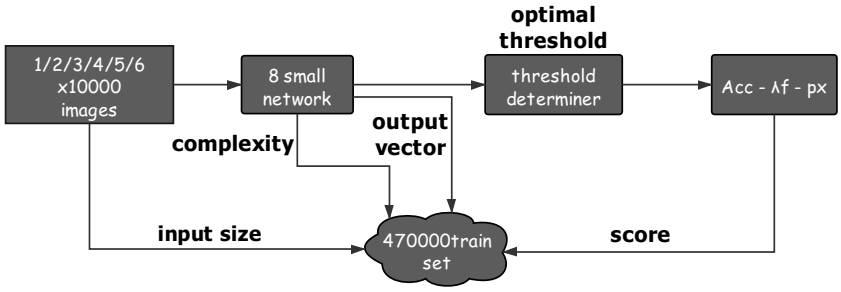

Fig. 4. **Flow diagram of data pre-processing**

We adopt three common regression models: decision tree, random forest and gradient-boost decision tree (GBDT). Decision tree is energy-efficient but less accurate, while random forest and gradient-boost decision tree improve the confidence of the determiner but take more energy. We optimize the parameters and train all models with the previous 470000 samples.

3.3 Threshold Turning Algorithm for the determiner

The threshold of the determiner affects both the frequency of calling the big network and the overall accuracy. An appropriate threshold is crucial for our architecture. However, traditional threshold finding algorithm would be easily trapped in a local optimum, thus we propose a new threshold turning algorithm for better performance in algorithm 1.

The basic idea is to find the threshold by binary search method with descending step size so that the search is gradually becoming fine-grained. We denote q as the performance of threshold, i.e., $Acc - \lambda f$ corresponding to threshold φ. We tend to turn φ towards the opposite direction of gradient. Meanwhile, we explore the search space and let q drop with a possibility of ε to find the global optimum. This $\varepsilon - greedy$ method provides an opportunity to jump out of local optimum, but the possibility ε has to decrease for quicker convergence. When the step size δ is smaller than a specific value Δ, the algorithm terminates and we select the best result from the saved records.

Algorithm 1 threshold turning with $\varepsilon - greedy$

$\varphi \leftarrow \Phi$ // initial threshold;
$\delta \leftarrow 2^{maxDepth} \times \Delta$ // initial step size;
$q \leftarrow 0$;
$q' \leftarrow 0$;
flag $\leftarrow$ True // denote the turning direction;
while $\delta \geq \Delta$ **do**
  $q' \leftarrow q$;
  $q \leftarrow TestAndCompute(\varphi)$;
  if $q < q'$ && $\varepsilon - greedy$ **then**
    flag $\leftarrow \neg$ flag;
    $\delta \leftarrow \delta/2$;
  end if
  if $flag$ **then**
    $\varphi \leftarrow \varphi + \delta$;
  else
    $\varphi \leftarrow \varphi - \delta$;
  end if
end while

4 EXPERIMENT

4.1 Experimental Setup

We carry out all of our experiments on MNIST [9]. The dataset contains 60000 train samples and 10000 test samples with 28×28 pixels in each figure. To accomadate the relatively simple dataset, we choose LeNet [9] as our big network. Besides, we use 8 small lighted-weighted networks [11] considering the trade-off between energy consuming and accuracy. The configurations for all the 9 networks(LeNet and 8 small networks) are listed in Table1. It is easy to see that all the networks performs really good on MNIST, yet LeNet still outstands by a small margin.

4.2 Performance of Determiners

Now that we have derived the training data described in section 3.2 (we set $\lambda = 0.1$ for all the experiments in this paper), we now train the determiner. Three machine learning based regression models

Table 1. Network Configuration

Name	LeNet	Mini1(m1)	Mini2(m2)	Mini3(m3)	Mini4(m4)
Input	28×28	28×28	28×28	28×28	28×28
	Conv2d $5 \times 5 \times 20$	Conv2d $5 \times 5 \times 10$	Conv2d $5 \times 5 \times 7$	Conv2d $5 \times 5 \times 2$	Conv2d $2 \times 2 \times 2$
	MaxPool2d 2×2	MaxPool2d 2×2	MaxPool2d 2×2	MaxPool2d 2×2	MaxPool2d 2×2
	Conv2d $5 \times 5 \times 50$	Conv2d $5 \times 5 \times 25$	Conv2d $5 \times 5 \times 15$	Conv2d $5 \times 5 \times 5$	Conv2d $2 \times 2 \times 5$
	MaxPool2d 2×2	MaxPool2d 2×2	MaxPool2d 2×2	MaxPool2d 2×2	MaxPool2d 2×2
	Linear 500	Linear 250	Linear 150	Linear 50	Linear 50
	Linear 84	Linear10	Linear 10	Linear 10	Linear 10
	Linear 10				
Size(MB)	2.05	0.55	0.25	0.05	0.08
Top-1 Accruacy	99.16	98.86	98.73	97.99	97.81

Name	Mini5(m5)	Mini6(m6)	Mini7(m7)	Mini8(m8)
Input	28×28	28×28	28×28	28×28
	Conv2d $5 \times 5 \times 20$	Conv2d $5 \times 5 \times 10$	Conv2d $5 \times 5 \times 10$, stride 2	Conv2d $5 \times 5 \times 10$, stride 4
	MaxPool2d 2×2	MaxPool2d 2×2	MaxPool2d 2×2	MaxPool2d 2×2
	Linear 500	Linear 250	Linear 250	Linear 250
	Linear10	Linear 10	Linear 10	Linear 10
Size(MB)	1.72	1.49	0.39	0.11
Top-1 Accruacy	98.77	97.99	98.05	95.18

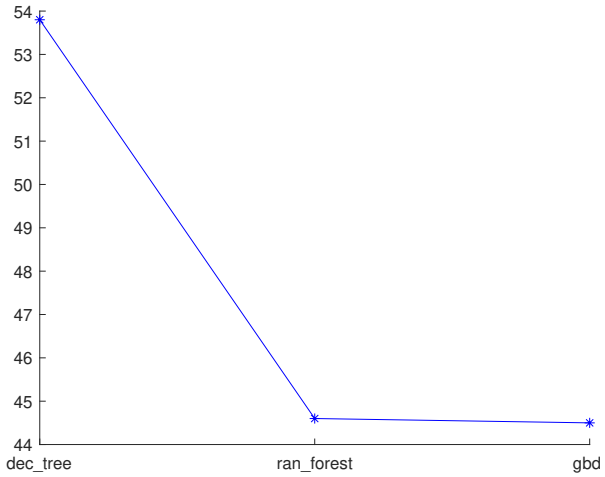

Fig. 5. MSE of Determiners

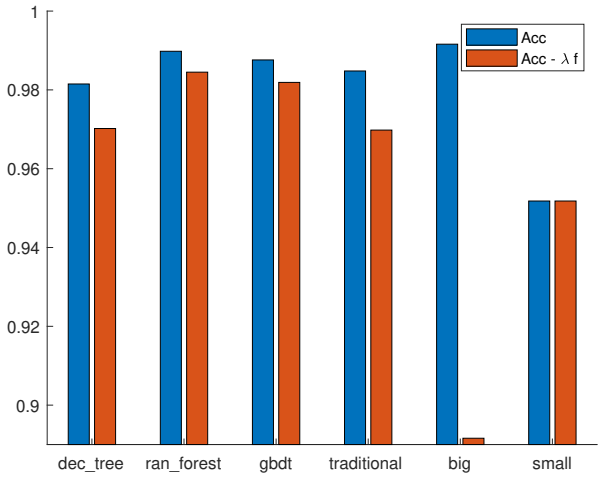

Fig. 6. Performance of Determiners

including decision tree [7], Gradient-Boost Decision Tree (GBDT) [14] and random forest [10] are tested in this experiment. The data are split into 10 folders for train and test with 8 folders for training and 2 folders for test. We evaluate the models by comparing their mean square errors (MSEs). The result is shown in figure 5. Note that lower MSE indicates better performance, The decision tree reaches the highest MSE while random forest and GBDT reaches similar MSE, we can assume that the ultimte

selection of determiners is between the two models.

However, the MSE reflects the performance of the determiners only in an indirect. We tend to assess the models via the metric we propose in section 3.1 and the result is shown in figure 6. Here comes the interesting part. In figure 6, we can see that a system with only big network reaches the highest accuracy (Acc) yet it has the lowest $Acc - \lambda f$ which indicates that it performs really bad under our criteria. This is quite natural as we take energy consuming as a significant factor and λ here is the trade-off. The Acc for the traditional method is 98.48% and $Acc - \lambda f$ is 96.98%. The random forest is better than the traditonal one by 0.5% Acc and 1.47% $Acc - \lambda f$ and GBDT 0.28% Acc and 1.21% $Acc - \lambda f$. The data here also verifies our assumption mentioned above.

4.3 Energy Consuming

The idea for evaluating energy consuming is to check the real electric consuming and the emission of carbon dioxide. However, due to the limitation of time and correlated equipment, we roughly assess the consumption by checking the time cost of the determiners. We test well-trained models with 10000 samples and compare the time they call the determiners. The result is shown in table 2. The time consuming for random forest is 5 times more than the traditional method , GBDT and decision tree is similar to the traditional one. Considering the trade-off between energy consuming and accuracy and the assumption we propose in section 4.2, we assert that GBDT is the best choice for determiners.

Table 2. Time Cosuming

Name	traditional	decision tree	GBDT	random forest	big	small
Time(ms)	22.55	23.75	27.41	117.01	33.08	22.96

4.4 Threshold Turning

Once the determiner is fixed, we have to determine the threshold. This is the very essential part of the determiner as we intend to find the best recall point to maximize $Acc - \lambda f$, low threshold saves the energy yet it scarifies the accuracy while high threshold does the opposite. The approach we use in this experiment is shown as algorithm 1. After many iterations of the algorithm, we finally fix the threshold for GBDT as 0.8352.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose a novel determiner which tries to optimize the trade-off between energy and accuracy, while adapts to all given small networks. In order to avoid being trapped in local minimum, we also present a new threshold turning algorithm for better performance. Experiments on MNIST show that our proposed structure performs better than traditional first-order score margin by 0.5% *Acc* and 1.4% $Acc - \lambda f$ for random forest model and 0.28% *Acc* and 1.21% $Acc - \lambda f$ for GBDT model. As future work, we will work on more challenging tasks and utilize complex deep neural networks for experiment. In addition, the complexity of the determiner is another trade-off which is worth considering.

References

[1] Srimat Chakradhar, Murugan Sankaradas, Venkata Jakkula, and Srihari Cadambi. A dynamically configurable coprocessor for convolutional neural networks. In *ACM SIGARCH Computer Architecture News*, volume 38, pages 247–257. ACM, 2010.

[2] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In *ACM Sigplan Notices*, volume 49, pages 269–284. ACM, 2014.

[3] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, et al. Dadiannao: A machine-learning supercomputer. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 609–622. IEEE Computer Society, 2014.

[4] Zidong Du, Krishna Palem, Avinash Lingamneni, Olivier Temam, Yunji Chen, and Chengyong Wu. Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 201–206. IEEE, 2014.

[5] Jeff Gehlhaar. Neuromorphic processing: a new frontier in scaling computer architecture. *ACM SIGPLAN Notices*, 49(4):317–318, 2014.

[6] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[7] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207. Citeseer, 1996.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[11] Eunhyeok Park, Dongyoung Kim, Soobeom Kim, Yong-Deok Kim, Gunhee Kim, Sungroh Yoon, and Sungjoo Yoo. Big/little deep neural network for ultra low power inference. In *Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis*, pages 124–132. IEEE Press, 2015.

[12] Seongwook Park, Kyeongryeol Bong, Dongjoo Shin, Jinmook Lee, Sungpill Choi, and Hoi-Jun Yoo. 4.6 a1. 93tops/w scalable deep learning/inference processor with tetra-parallel mimd architecture for big-data applications. In *2015 IEEE International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers*, pages 1–3. IEEE, 2015.

[13] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[14] Jerry Ye, Jyh-Herng Chow, Jiang Chen, and Zhaohui Zheng. Stochastic gradient boosted distributed decision trees. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 2061–2064. ACM, 2009.

[15] Erjin Zhou, Haoqiang Fan, Zhimin Cao, Yuning Jiang, and Qi Yin. Extensive facial landmark localization with coarse-to-fine convolutional network cascade. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 386–391, 2013.