# ADAPTIVE INFERENCE USING HIERARCHICAL CONVOLUTIONAL BAG-OF-FEATURES FOR LOW-POWER EMBEDDED PLATFORMS

*Nikolaos Passalis*[1]*, Jenni Raitoharju*[1]*, Anastasios Tefas*[2]*, and Moncef Gabbouj*[1]

[1]Faculty of Information Technology and Communication Sciences, Tampere University, Finland
[2]School of Informatics, Aristotle University of Thessaloniki, Greece
email: {nikolaos.passalis, jenni.raitoharju}@tuni.fi, tefas@csd.auth.gr, moncef.gabbouj@tuni.fi

## ABSTRACT

Using early exits provide a straightforward way to implement models that can adapt on-the-fly to the available computational resources. However, early exits in many cases suffer from significant limitations, which often prohibit their practical application, especially when placed on convolutional layers with narrow receptive fields. In this work, we propose a method capable of overcoming these limitations by a) using a Bag-of-Features (BoF)-based pooling approach, that allows for keeping more information regarding the distribution of the extracted feature vectors, while also maintaining more spatial information and b) employing a simple, yet effective, hierarchical approach for designing the exits, allowing for efficiently re-using the information that was already extracted by the previous layers. It is experimentally demonstrated that the proposed approach leads to significant performance improvements, allowing early exits to be a more practical tool that can be used in many real-world embedded applications.

***Index Terms***— Early Exits, Adaptive Inference, Bag-of-Features, Lightweight Deep Learning

## 1. INTRODUCTION

Recent advances in Deep Learning (DL) have led to a number of spectacular applications, ranging from autonomous cars [1] to intelligent buildings [2]. However, DL models are especially complex, often requiring expensive, powerful and energy-intensive hardware in order to successfully deploy them, significantly increasing the cost, reducing their flexibility and, as a result, slowing down the adoption of DL in many applications where these requirements cannot be easily met. These limitations are well understood in the relevant literature and many methods have been developed to partially overcome them, e.g., model compression and quantization [3], knowledge distillation [4], etc.

However, the vast majority of existing approaches aim to just learn a static, yet faster and more lightweight, DL model ignoring the fact that in many applications there is the need to dynamically adapt the inference process to the available computational resources. For example, consider a real-time security application where the persons that appear in a frame must be recognized. The time needed for the recognition process is proportional to the number of people depicted in the frame. Existing models are unable to dynamically adapt to the available workload, e.g., by providing faster (and possibly less accurate) predictions when the workload is high and slower (and possibly more accurate) predictions when the workload is low.

The most promising candidate for tackling this task is models with adaptive computational graphs [5, 6, 7], which provide a straightforward way to adapt on-the-fly to the available resources by choosing a different path on the model's computational graph. Other approaches [6, 7] achieve this by adding a number of early exits at various levels of the network and, as a result, are capable of providing estimations regarding the final output of the network at various points of the feed-forward process. However, using early exits does not always lead to acceptable performance [6], since they often employ an aggressive subsampling approach, e.g., global average pooling [8], ignoring both the spatial information and the distribution of the extracted feature vectors. On the other hand, the densely connected structure used in [7] requires a significant number of structural changes in the architecture of a network and cannot be easily used with existing neural networks. Furthermore, the approach proposed in [6] completely ignores the representations extracted by the previous exit layers, throwing away readily available information that can be potentially used to further increase the performance of the subsequent exit layers. It is also worth noting that early exits have been used in past for reducing training issues related to vanishing gradients [9].

The main contribution of this work is to propose a method capable of overcoming the aforementioned limitations. First, instead of using global average pooling for extracting a compact representation for the early exits [8], we propose using a Bag-of-Features (BoF)-based formulation [10, 11], that allows for a) keeping more information regarding the distribution of the extracted feature vectors and b) introducing more spatial information into the extracted representation by using a spatial segmentation scheme. The latter is especially impor-

tant for earlier exits, where the receptive field of the convolutional layers is usually smaller. Furthermore, we propose a simple, yet effective, hierarchical approach for designing the exits, allowing for efficiently re-using the information that was already extracted by earlier layers (which is ignored by existing formulations [6]). Note that the proposed method can work with any neural network architecture and requires no model-specific changes to the base network. We experimentally demonstrate using four different datasets and network architectures that the proposed approach can lead to significant performance improvements, transforming early exits into a practical tool that can be used in many real-world embedded applications.

## 2. PROPOSED METHOD

Let $f(\mathbf{x})$ denote a pretrained neural network, where $\mathbf{x} \in \mathbb{R}^{W \times H \times C}$ is a given image, while $W$, $H$ and $C$ denote the width, height and number of channels of the corresponding input image. Also, let $\mathbf{y}_i = f_C(\mathbf{x}, i) \in \mathbb{R}^{W_i \times H_i \times C_i}$ denote the output of the $i$-th convolutional layer of the network, where $W_i$ and $H_i$ are the width and height of the $i$-th feature map extracted from the network, while $C_i$ denotes the number of filters in the $i$-th convolutional layer.

Using early exits in neural networks provide a way to estimate the final output of the network at various points of their computational graph, without having to feed-forward the whole network [6]. This allows the network to a) dynamically adapt to the available computational resources, as well as to possibly stop the inference process earlier, if we are confident enough of the output of the network. However, the size of the intermediate feature maps can be enormous. To this end, global average pooling can be used to extract a compact representation out of the intermediate feature maps of a network [8]: $\mathbf{s}_i^{avg} = \frac{1}{W_i H_i} \sum_{k=1}^{W_i} \sum_{l=1}^{H_i} [\mathbf{y}_i]_{kl} \in \mathbb{R}^{C_i}$, where the notation $[\mathbf{y}_i]_{kl}$ is used to refer to the feature vector extracted from the location $(k, l)$ of the $i$-th feature map. Then, this representation is fed to a fully connected layer, which is directly trained to predict the category of the input sample, allowing for estimating the final classification decision at various intermediate points of the network.

In this paper, we propose using a BoF-based aggregation approach for extracting the representation that is fed to the employed fully connected layer instead of simply performing global average pooling and possibly discarding valuable information [10, 12]. To this end, first a set of prototype vectors (also known as codewords) $\mathbf{v}_{ij} \in \mathbb{R}^{C_i}$ are employed to model the distribution of the feature vectors extracted from the $i$-th layer. The set of these vectors $\mathcal{V}_i = \{\mathbf{v}_{i1}, \mathbf{v}_{i2}, \ldots, \mathbf{v}_{iN_K}\}$, where $N_K$ is the number of codewords used, is called dictionary or codebook. A different codebook is used for each exit layer. Then, the probability of observing each feature vector $[\mathbf{y}_i]_{kl}$, extracted from the $i$-layer of the network, for a given image $\mathbf{x}$ can be estimated using Kernel Density Estimation [13] as:

$$p([\mathbf{y}_i]_{kl}|\mathbf{x}) = \sum_{j=1}^{N_K} [\mathbf{s}_i]_j K([\mathbf{y}_i]_{kl}, \mathbf{v}_{ij}) \in \mathbb{R}, \tag{1}$$

where $K(\cdot)$ is a kernel function and $\mathbf{s}_i \in \mathbb{R}^{N_K}$ are image-specific parameters that separately adjust the density estimation. Then, the parameters can be calculated using a maximum likelihood estimator [13]:

$$\mathbf{s}_i = \arg\max_{\mathbf{s}} \sum_{k=1}^{W_i} \sum_{l=1}^{H_i} \log \left( \sum_{j=1}^{N_K} [\mathbf{s}]_j K([\mathbf{y}_i]_{kl}, \mathbf{v}_{ij}) \right) \in \mathbb{R}^{N_K}. \tag{2}$$

As shown in [13], the image specific parameters can be trivially estimated, giving rise to the well known soft-BoF formations [12, 14]. Therefore, the representation extracted from the $i$-th layer of the network is calculated as:

$$\mathbf{s}_i = \frac{1}{W_i H_i} \sum_{k=1}^{W_i} \sum_{l=1}^{H_i} \mathbf{u}_{ikl} \in \mathbb{R}^{N_K}, \tag{3}$$

where $[\mathbf{u}_{ikl}]_j = \frac{K([\mathbf{y}_i]_{kl}, \mathbf{v}_{ij})}{\sum_{m=1}^{N_K} K([\mathbf{y}_i]_{kl}, \mathbf{v}_{im})} \in [0, 1]$. The histogram vector $\mathbf{s}_i$ essentially provides a compact summary that describes the semantic content of an image at various levels of granularity, maintaining more information regarding the actual *distribution* of the vectors $[\mathbf{y}_i]_{kl}$ than the average representation ($\mathbf{s}_i^{avg}$).

In this work, the BoF model is implemented using a normalized RBF layer, followed by a recurrent accumulation layer, as proposed in [10], while a hyperbolic (sigmoid) kernel is used to compute the similarity between each feature vector and the codewords [15]:

$$K([\mathbf{y}_i]_{kl}, \mathbf{v}) = \frac{1}{2} \left( \tanh(\alpha [\mathbf{y}_i]_{kl}^T \mathbf{v} + \beta) + 1 \right) \tag{4}$$

where $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, while $\alpha$ and $\beta$ are the kernel parameters (typically set to $\alpha = 1$ and $\beta = 0$). The kernel is also scaled to $0 \ldots 1$ to ensure that it is compatible with the employed quantization process.

Furthermore, the quantization process can be repeated at various spatial levels, similarly to many spatial pyramid aggregation schemes [16], giving rise to the Spatial BoF [10], allowing for keeping more spatial information. This is especially important for the representations extracted from the earlier layers, due the significantly smaller effective receptive field of the earlier convolutional layers.

Apart from modeling the distribution of the features extracted from the exit layers using the BoF model, we also propose building an incremental hierarchical representation that builds upon the representations from the previous exit layers instead of relying only on the representation extracted from the current layer. Therefore, the final representation $\mathbf{s}^h$

extracted from the $i$-th layer is calculated as:

$$\mathbf{s}_i^h = \begin{cases} \mathbf{s}_i & \text{if } i = 1 \\ \mathbf{s}_i \frown \mathbf{s}_{i-1}^h & \text{if } i > 1 \end{cases}, \quad (5)$$

where the notation $\mathbf{a} \frown \mathbf{b}$ is used to denote the concatenation of vectors $\mathbf{a}$ and $\mathbf{b}$. Note that the impact on the computational complexity of the model is minimal, given that the representation extracted from the previous layers can be readily cached and re-used in the subsequent calculations. Therefore, this approach provides a straightforward way to exploit the previous computations, that were already performed, to further increase the prediction accuracy of the exit layers, as also experimentally demonstrated in Section 3.

## 3. EXPERIMENTAL EVALUATION

The proposed method is evaluated using four different network architectures, ranging from an extremely lightweight two-layer Convolutional Neural Network (CNN) to a state-of-the-art lightweight MobileNet [17], as well as a wide range of datasets: a) the MNIST image classification dataset [18], b) the Fashion MNIST fashion product classification dataset [19], c) the CIFAR-10 object recognition dataset [20], as well as the more challenging d) FER-2013 facial expression dataset [21].

The following network architectures were evaluated: a) CNN-1, which is composed of: $3 \times 3$ convolution with 32 filters, $2 \times 2$ max pooling, $3 \times 3$ convolution with 64 filters, $2 \times 2$ max pooling, fully connected layer with 1024 neurons, dropout $p = 0.5$, and a final fully connected classification layer, b) CNN-2, which follows the same architecture as CNN-1, but uses twice the number of filters in the first two convolutional layers, c) CNN-3, which is composed of: $3 \times 3$ convolution with 16 filters, $3 \times 3$ convolution with 32 filters, $2 \times 2$ max pooling, $3 \times 3$ convolution with 64 filters, $3 \times 3$ convolution with 128 filters, $2 \times 2$ max pooling, fully connected layer with 1024 neurons, dropout $p = 0.5$, and a final fully connected classification layer, and d) a MobileNet with the first convolutional layer appropriately tuned for each dataset. All the networks were trained using the categorical cross-entropy loss, while the ReLU activation function was used for all the layers. The Adam algorithm was used for the optimization [22], while the number of epochs and learning rates ($\eta$) are reported for each network in Table 1.

Two exit layers were used for all the conducted experiments. Each network was first trained using each dataset (referred to as "Base Network" in Table 1) and then the weights were fixed, and two exit layers were added and trained for the same number of iterations. All the exit layers were simultaneously trained by combining the corresponding losses. The first exit layer was placed after the 1st convolutional layer for the CNN-1 and CNN-2, after the 2nd convolutional layer for the CNN-3 and after the 5th convolutional layer for the MobileNet. The second exit layer was placed after the 2nd convolutional layer for the CNN-1 and CNN-2, after the 4th convolutional layer for the CNN-3 and after the 7th convolutional layer for the MobileNet. The second exit layer was trained either by directly using the representation extracted from that layer or using the hierarchical representation that exploits the information extracted from the first exit layer (denoted by "Hierarchical" in Table 1). A separate fully connected layer was used for each hierarchical exit.

Apart from directly evaluating the performance of each network ("Base Network"), we also evaluated a) a standard feature aggregation method proposed for early exits (global average pooling, denoted by "Global Pooling" in Table 1) [8], b) the BoF representation (the number of codewords for each of the two exit layers reported in parentheses) and c) two variants of the Spatial BoF pooling (segmentation into 4 spatial regions [10]) with different number of codewords (again reported in parentheses). Note that altering the number of codewords allows the proposed method to better adapt to the needs of each application. The total multiply-accumulate operations (Million MAC, MMAC) is also reported for each network up to the corresponding exit.

Several interesting conclusions can be drawn from the results reported in Table 1. First, using the proposed BoF-based pooling for the first exit layer reduced the classification error in all the cases, without significantly affecting the number of MMAC operations needed. For example, for a MobileNet trained on the CIFAR-10 data the error drops from 24% to almost 10%, while the MMAC operations increase by less than 1.5%. The same is also true for the rest of datasets. The ability of the proposed method to a) tune the length of the extracted representation to the needs of the applications, as well as b) better model the spatial information contained in the extracted feature vectors allows for increasing the accuracy even more. For example, for the MNIST dataset, the error using the standard Global Pooling drops from 51% to less than 5%, while only slightly increasing the MMAC operations. Also, note that using spatial segmentation in the first exit layer is especially important, since the receptive field of the first convolutional layers is relatively narrow.

The proposed method also outperforms the plain average pooling approach when used in the second exit of the network. Note that the improvements obtained using spatial segmentation on the second exit are smaller, since the extracted feature map already captures significant part of the spatial information contained in the original image due its larger receptive field. Nonetheless, the proposed Spatial BoF always achieves the best classification performance. Finally, combining the representation extracted from both exit layers, i.e., using the proposed hierarchical representation scheme, improves the performance for the BoF model. For example, for the MNIST dataset this allows for reducing the classification error from 2.93% to just 1.57%, while for the CIFAR-10 dataset it even slightly outperforms the original pre-trained network (8.28% vs. 8.40%). This is even more impressive

**Table 1**. Evaluation Results

| Aggregation Method | Exit - 1 Error | Exit - 1 Million MAC | Exit - 2 Error | Exit - 2 Million MAC | Hierarchical Error | Hierarchical Million MAC |
|---|---|---|---|---|---|---|
| **MNIST Dataset** | | | | | | |
| Base Network (CNN-1, 50 epochs, $\eta = 0.001$) | | | | | 0.70% | 4.10 |
| Global Pooling | 51.09% | 0.21 | 5.89% | 2.45 | 5.82% | 2.46 |
| BoF (16, 16) | 14.97% | 0.31 | 5.70% | 2.48 | 3.98% | 2.57 |
| Spatial BoF (8, 8) | 4.79% | 0.26 | 3.18% | 2.47 | 1.73% | 2.51 |
| Spatial BoF (64, 32) | **2.59%** | 0.58 | **2.93%** | 2.51 | **1.57**% | 2.87 |
| **Fashion MNIST Dataset** | | | | | | |
| Base Network (CNN-2, 50 epochs, $\eta = 0.001$) | | | | | 8.09% | 12.66 |
| Global Pooling | 32.08% | 0.43 | 17.30% | 9.37 | 16.49% | 9.37 |
| BoF (16, 16) | 17.55% | 0.61 | 14.74% | 9.42 | 13.00% | 9.60 |
| Spatial BoF (8, 8) | 15.56% | 0.52 | 14.51% | 9.40 | 11.91% | 9.48 |
| Spatial BoF (64, 32) | **13.44%** | 0.61 | **12.69%** | 9.42 | **10.67**% | 9.60 |
| **CIFAR-10 Dataset** (CNN-3) | | | | | | |
| Base Network (CNN-3, 50 epochs, $\eta = 0.001$, 50 epochs, $\eta = 0.0001$) | | | | | 14.01% | 17.38 |
| Global Pooling | 56.27% | 4.04 | 20.05% | 14.09 | 19.74% | 14.09 |
| BoF (64, 64) | 37.22% | 4.46 | 22.02% | 14.30 | 20.67% | 14.71 |
| Spatial BoF (32, 32) | 33.40% | 4.25 | 20.30% | 14.19 | 19.23% | 14.41 |
| Spatial BoF (64, 64) | **29.73%** | 4.46 | **19.69%** | 14.30 | **18.79**% | 14.72 |
| **CIFAR-10 Dataset** (MobileNet v.2) | | | | | | |
| Base Network (MobileNet v.2, 50 epochs, $\eta = 0.001$, 50 epochs, $\eta = 0.0001$) | | | | | 8.40% | 94.60 |
| Global Pooling | 24.51% | 44.34 | 12.81% | 64.15 | 12.27% | 64.15 |
| BoF (128, 128) | 10.64% | 44.88 | 8.61% | 64.95 | 8.47% | 65.48 |
| Spatial BoF (64, 64) | 10.55% | 44.61 | 8.70% | 64.55 | 8.57% | 64.83 |
| Spatial - BoF (256, 256) | **9.85%** | 45.42 | **8.36%** | 65.75 | **8.28**% | 66.85 |
| **FER-2103 Dataset** | | | | | | |
| Base Network (MobileNet v.2, 50 epochs, $\eta = 0.001$, 50 epochs, $\eta = 0.0001$) | | | | | 37.11% | 211.52 |
| Global Pooling | 57.43% | 98.45 | 51.85% | 143.01 | 51.16% | 143.02 |
| BoF (128, 128) | 46.98% | 99.64 | 43.44% | 144.80 | 42.32% | 146.00 |
| Spatial BoF (64, 64) | 42.41% | 98.75 | 41.29% | 143.46 | 40.79% | 143.77 |
| Spatial BoF (256, 256) | **44.47%** | 100.85 | **40.99%** | 146.60 | **39.57**% | 149.03 |

**Table 2**. Performance evaluation using Raspberry PI 3 Model B+ (first exist, CNN-1, averaged over 50 runs)

| Approach | FPS | Speedup | Class. Efficiency |
|---|---|---|---|
| Base Network | 45.00 | - | - |
| Average Pooling | 192.12 | 4.27 | 2.09 |
| Spatial BoF (8, 8) | 140.92 | 3.13 | **2.98** |
| Spatial BoF (64, 32) | 131.94 | 2.93 | 2.88 |

given than the hierarchical exit requires 66.85 MMAC operations instead of 94.60 MMAC (that are required for the full forward pass).

Finally, we also report the number of images processed per second (FPS) using a CPU-based embedded platform, a Raspberry PI 3 Model B+ (clocked at 1.4GHz), using the PyTorch library. The performance evaluation results (FPS, speed up over using the final output of the network, along with a classification efficiency metric (speedup multiplied by accuracy)) are reported in Table 2. Note that even though a non-optimized implementation was employed for the proposed BoF method, it was indeed capable of providing a significant speedup, demonstrating its potential for practical embedded applications.

## 4. CONCLUSIONS

A method capable of improving the performance of convolutional neural networks that employ early exits was proposed in this paper. The proposed method employs a BoF-based formulation to keep more information regarding the distribution of the extracted feature vectors. At the same time, it also uses a simple, yet effective, hierarchical approach to provide a way for efficiently re-using the information that was already extracted by the previous layers. It was experimentally demonstrated that the proposed method can indeed increase the accuracy of early exits, maintaining their ability to readily adapt to the available computational resources.

## 5. REFERENCES


[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436, 2015.

[2] Peng Li, Zhikui Chen, Laurence Tianruo Yang, Qingchen Zhang, and M Jamal Deen, "Deep convolutional computation model for feature learning on big data in internet of things," *IEEE Trans. on Industrial Informatics*, vol. 14, no. 2, pp. 790–798, 2018.

[3] Song Han, Huizi Mao, and William J Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proc. Int. Conf. on Learning Representations*, 2016.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean, "Distilling the knowledge in a neural network," in *Proc. NIPS Deep Learning and Representation Learning Workshop*, 2015.

[5] Andreas Veit and Serge Belongie, "Convolutional networks with adaptive inference graphs," in *Proc. European Conf. on Computer Vision*, 2018, pp. 3–18.

[6] Yue Bai, Shuvra S Bhattacharyya, Antti P Happonen, and Heikki Huttunen, "Elastic neural networks: A scalable framework for embedded computer vision," in *Proc. European Signal Processing Conf.*, 2018, pp. 1472–1476.

[7] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger, "Multi-scale dense networks for resource efficient image classification," in *Proc. Int. Conf. on Learning Representations*, 2018.

[8] Yi Zhou, Yue Bai, Shuvra S Bhattacharyya, and Heikki Huttunen, "Elastic neural networks for classification," *arXiv preprint arXiv:1810.00589*, 2018.

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Computer Vision and Pattern Recongition*, 2015, pp. 1–9.

[10] Nikolaos Passalis and Anastasios Tefas, "Learning bag-of-features pooling for deep convolutional neural networks," in *Proc. IEEE Int. Conf. on Computer Vision*, 2017, pp. 5766–5774.

[11] Nikolaos Passalis and Anastasios Tefas, "Training lightweight deep convolutional neural networks using bag-of-features pooling," *IEEE Trans. on Neural Networks and Learning Systems*, 2018.

[12] Nikolaos Passalis and Anastasios Tefas, "Neural bag-of-features learning," *Pattern Recognition*, vol. 64, pp. 277–294, 2017.

[13] Subhabrata Bhattacharya, Rahul Sukthankar, Rong Jin, and Mubarak Shah, "A probabilistic representation for efficient large scale visual recognition tasks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2011, pp. 2593–2600.

[14] Jan C Van Gemert, Jan-Mark Geusebroek, Cor J Veenman, and Arnold WM Smeulders, "Kernel codebooks for scene categorization," in *Proc. European Conf. on Computer Vision*, 2008, pp. 696–709.

[15] Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis, "Temporal logistic neural bag-of-features for financial time series forecasting leveraging limit order book data," *arXiv preprint arXiv:1901.08280*, 2019.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. of the European Conf. on Computer Vision*, 2014, pp. 346–361.

[17] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[19] Han Xiao, Kashif Rasul, and Roland Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[20] Alex Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., University of Toronto, 2009.

[21] Octavio Arriaga, Matias Valdenegro-Toro, and Paul Plöger, "Real-time convolutional neural networks for emotion and gender classification," *arXiv preprint arXiv:1710.07557*, 2017.

[22] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. for Learning Representations*, 2015.