BAG-OF-FEATURES-BASED KNOWLEDGE DISTILLATION FOR LIGHTWEIGHT CONVOLUTIONAL NEURAL NETWORKS

Alexandros Chariton, Nikolaos Passalis and Anastasios Tefas

Computational Intelligence and Deep Learning Group, AIIA Lab
Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
E-mails: {charitona, passalis, tefas}@csd.auth.gr

ABSTRACT

Knowledge distillation enables us to transfer the knowledge from a large and complex neural network into a smaller and faster one. This allows for improving the accuracy of the smaller network. However, directly transferring the knowledge between enormous feature maps, as they are extracted from convolutional layers, is not straightforward. In this work, we propose an efficient mutual information-based approach for transferring the knowledge between feature maps extracted from different networks. The proposed method employs an efficient Neural Bag-of-Features formulation to estimate the joint and marginal probabilities and then optimizes the whole pipeline in an end-to-end manner. The effectiveness of the proposed method is demonstrated using a lightweight, fully convolutional neural network architecture, which aims toward high-resolution analysis and targets photonic neural network accelerators.

Index Terms— Knowledge Distillation, Bag-of-Features, Mutual Information, Convolutional Neural Networks

1. INTRODUCTION

Deep Learning (DL) led to a tremendous success in many different application areas [1]. However, DL models are usually computationally intensive, requiring specialized and energy-consuming hardware for inference. This has fueled the interest in developing more lightweight architectures [2, 3], methods for compressing existing architectures [4], as well as methods for mitigating the possible loss in accuracy due to employing lightweight architectures [5, 6], such as knowledge distillation [5], which works by transferring the knowledge encoded in a large and complex teacher model into a lightweight student model.

Even though many knowledge distillation methods have been proposed in the recent literature, most of them focus on transferring the knowledge either between vectors or matrices extracted from the two models. For example, knowledge distillation aims to "mimic" the soft probabilities extracted by the teacher model, hint-based transfer employs auxiliary dimensionality reduction layers to match the dimensionality of the representations extracted from the networks [7], while other methods, such as probabilistic knowledge transfer (PKT) [8], attempt to mimic other qualities, such as the similarities between samples, as they are encoded by the representations extracted from various layers of the networks. Even though the latter methods can handle vectors that have different dimensionality, they cannot directly handle the multiple feature vectors, as typically extracted from the convolutional layers of a neural network. To this end, they either employ aggressive pooling strategies, that can lead to loss of information, or flatten the feature maps into vectors with enormous dimensionality, reducing the effectiveness of the distillation process.

In this work, we aim to overcome these limitations by fully avoiding using pooling and/or flattening operators when transferring the knowledge between intermediate layers. It is worth noting that this problem is especially challenging, since the intermediate feature maps are not only typically very large, but they also differ in the number of channels used, since the student model is typically smaller than the teacher. To this end, we propose to directly optimize the student model to maximize the mutual information between the distribution of the feature vectors of the student and teacher, as they are extracted by a specific convolutional layer. However, measuring mutual information in high-dimensional spaces is especially challenging and usually histogram binning methods, among others, are used to approximate the mutual information in such spaces. To overcome this limitation, we propose employing, after appropriately adjusting it, a well-known model for extracting summary representations, the Bag-of-Features (BoF) model [9, 10].

The main contribution of this work is a fully differentiable and easy to optimize in a end-to-end manner formulation of mutual information for feature distribution between different networks. The proposed method employs a deep-learning ori-

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871391 (PlasmoniAC). 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.

ented variant of the BoF model [11] model to perform soft quantization, which in turn enables us to estimate the joint and marginal probabilities in a fully differentiable manner. Furthermore, BoF enables us to have adaptive histogram bins that are placed according to the density of the distribution and the task at hand, going beyond simple rule-based binning approaches. Then, the classical mutual information definition can be directly used as a criterion for optimizing the networks. It is worth noting that the proposed method is tailored towards a very specific part of knowledge distillation, i.e., transferring the knowledge between feature vectors extracted from different convolutional layers. As a result, it is orthogonal to most of the existing distillation methods and can be readily combined with them. Experiments conducted on two image datasets demonstrate the effectiveness of the proposed approach in the challenging domain of training photonic neural networks, further closing the accuracy gap between photonic and regular DL models.

The rest of the paper is structured as follows. In Section 2 we introduce and analytically derive the proposed method. Then, in Section 3 we present the experimental setup and results. Finally, Section 4 concludes this paper.

2. PROPOSED METHOD

Let $f(\mathbf{x}) \in \mathbb{B}^{N_o}$ denote a neural network, where $\mathbf{x}$ denotes the input to the network and N_o the output dimensionality of the network. In this work we focus on convolutional neural networks where the input is a multidimensional tensor $\mathbf{x} \in \mathbb{R}^{W \times H \times C}$, where W denotes the image width, H denotes the image height and C is the number of channels. Also, we assume that the networks are trained for classification tasks and one-hot encoding is used for the output neurons, i.e., N_o equals to the number of classes. Note that this is without loss of generality since the proposed method does not require any kind of ground-truth labels and it is applied directly on the convolutional layers of the network. Furthermore, we also use the notation $\mathbf{y}_k = f(\mathbf{x}, k) \in \mathbb{R}^{W_k \times H_k \times C_k}$ to refer to the output of the k-th convolutional layer of the network (for a given input $\mathbf{x}$), where W_k, H_k, and C_k refer to the width, height, and number of channels of the corresponding feature map. If the k-th layer is a fully connected one, then the same notation can be used, but the dimensionality of the output would be different, since the output is a vector instead of a feature map. However, note that in this work we focus only on transferring the knowledge between distributions defined by the feature maps extracted from the network. So we can assume that the output will be always a feature map when the notation $f(\cdot, k)$ is used.

Furthermore, let $f_T(\mathbf{x})$ denote the teacher network from which the knowledge is transferred to the student network denoted by $f_S(\mathbf{x})$. Similarly, we denote the output of the k-th layer of the student and teacher networks as $\mathbf{y}_{T,k} = f_T(\mathbf{x}, k)$ and $\mathbf{y}_{S,k} = f_S(\mathbf{x}, k)$ respectively. As we will discuss later, the receptive fields should be the same between the k-th layers of the teacher and student, i.e., the width W_k and height H_k should be the same for both networks for a specific layer used for knowledge transfer. However, there is no such limitation on the number of channels for which the number can be different between the networks. Therefore, more convolutional filters can be used for the teacher network and a smaller number of them can be employed for the more lightweight student network. Also, let $Y_{T,k}$ denote a random variable that refers to the feature vector distribution extracted from the k-th layer of the teacher network and $Y_{S,k}$ to the feature vector distribution extracted from the k-th layer of the student network. We refer to the i-th feature vector extracted from the teacher and and student networks as $\mathbf{y}_{T,k,i} \in \mathbb{R}^{N_{T,k}}$ and $\mathbf{y}_{S,k,i} \in \mathbb{R}^{N_{S,k}}$ respectively, where $N_{T,k}$ and $N_{S,k}$ is the number of convolutional filters used by the teacher and student networks. Note that the total number of feature vectors extracted from each layer is the same, equal to $N_k = W_k \cdot H_k$, while the feature vectors are *paired*, i.e., $\mathbf{y}_{T,k,i}$ and $\mathbf{y}_{S,k,i}$ are extracted from the same spatial location of the feature maps.

In this work, we aim to maximize the mutual information between the feature vector distributions of the k-th layer of teacher and student networks, i.e., between $Y_{T,k}$ and $Y_{S,k}$. For the rest of this section we will assume that all derivations refer to a specific layer k to simplify the used notation, unless explicitly stated. Therefore, we aim to maximize the mutual information defined as:

$$I(Y_T; Y_S|\mathbf{x}) = \int_{\mathcal{Y}_T} \int_{\mathcal{Y}_S} p_{(Y_T, Y_S)}(\mathbf{y}_T, \mathbf{y}_S|\mathbf{x}) \cdot \log\left(\frac{p_{(Y_T, Y_S)}(\mathbf{y}_T, \mathbf{y}_S|\mathbf{x})}{p_{Y_T}(\mathbf{y}_T|\mathbf{x}) p_{Y_S}(\mathbf{y}_S|\mathbf{x})}\right) d\mathbf{y}_T d\mathbf{y}_S, \tag{1}$$

where $\mathbf{y}_T$ and $\mathbf{y}_S$ belong to the feature spaces $\mathcal{Y}_T$ and $\mathcal{Y}_S$ of the teacher and student respectively. Also, $p_{(Y_T, Y_S)}(\cdot)$ refers to the joint probability of Y_T and Y_S, while the notation $p_{Y_T}(\cdot)$ and $p_{Y_S}(\cdot)$ is used to refer to the corresponding marginal probabilities. Note that we measure mutual information per input sample $\mathbf{x}$, i.e., all probabilities are conditioned on $\mathbf{x}$, since a different distribution is generated for each input sample. In this work, we employ a histogram-based approach for dividing the feature spaces in an adaptive manner and then treating this problem using discrete variables in order to tackle the especially challenging problem of estimating the aforementioned densities in high-dimensional spaces.

To this end, we employ the well known Bag-of-Features model. The BoF model works by defining a number of bins using prototype vectors called *codewords*. Then, each of the feature vectors are quantized to each bin. In this work, we use a soft binning approach [11]. More specifically, we quantize the feature vectors extracted by the teacher as:

$$d_{T,i,j|\mathbf{x}} = \frac{K(\mathbf{y}_{T,i}, \mathbf{v}_{T,j})}{\sum_{l=1}^{N_K} K(\mathbf{y}_{T,i}, \mathbf{v}_{T,l})}, \tag{2}$$

where N_K is the total number of codewords/histogram bins used and $K(\cdot)$ is a kernel used to measure the similarity between the feature vector $\mathbf{y}_{T,i}$ and the codeword $\mathbf{v}_{T,j}$. Note that $d_{T,i,j|\mathbf{x}}$ expresses the similarity between each feature vector i and each codeword $\mathbf{v}_{T,j}$. These similarity values are bounded between 0 and 1, while $||\mathbf{d}_{T,i|\mathbf{x}}||_1 = 1$, where $||\cdot||_1$ denotes the l_1 norm and $\mathbf{d}_{T,i|\mathbf{x}} \in \mathbb{R}^{N_K}$ is the bin membership vector for the i-th feature vector. The Gaussian kernel is usually used [12]:

$$K(\mathbf{y}_{T,i}, \mathbf{v}_{T,j}) = \exp\left(-||\mathbf{y}_{T,i} - \mathbf{v}_{T,j}||_2^2 / 2\sigma^2\right), \qquad (3)$$

where σ is the scaling factor for the kernel and $||\cdot||_2$ denotes the l_2 norm. The final histogram for the BoF representation is extracted by averaging over the membership vectors:

$$\mathbf{h}_{T|\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{d}_{T,i|\mathbf{x}} \in \mathbb{R}^{N_K}, \qquad (4)$$

where N is the total number of extracted feature vectors as described before. Similarly, we can calculate the vectors $\mathbf{h}_{S|\mathbf{x}} \in \mathbb{R}^{N_{K'}}$ and $\mathbf{d}_{S,i|\mathbf{x}} \in \mathbb{R}^{N'_K}$ for the student, where $N_{K'}$ is the number of codewords used for the student. The codewords can be selected by gathering the feature vectors extracted from several input samples and then clustering them using k-means [10]. The centers of the clusters can be then used as the codewords, while the scaling factor can be set according to the standard deviation of each cluster. In this work, we opt for using the discriminative way of training the codewords proposed in [12]. Therefore, after initializing the codewords using k-means, we used an additional auxiliary classification layer to finetune both the codewords and scaling factors for a few epochs, for betting fitting the data. The codewords can be either fitted once, ideally after pre-training the student network for a few epochs, or they can be occasionally re-fitted during the optimization of the student network to mitigate potential distribution shift phenomena. We have experimentally established that fitting the codewords once, after training for a few epochs, is usually enough for most problems, as shown in Section 3.

BoF provides a straightforward way to estimate the discrete marginal probabilities for the teacher over the N_K codewords as:

$$p_{Y_T}(n|\mathbf{x}) = h_{T,n|\mathbf{x}},\ n = 1 \dots N_K, \qquad (5)$$

where the notation $h_{T,n|\mathbf{x}}$ is used to refer to the n-th element of the vector $\mathbf{h}_{T|\mathbf{x}}$. Similarly, the marginal probability for the student is calculated as:

$$p_{Y_S}(m|\mathbf{x}) = h_{S,m|\mathbf{x}},\ m = 1 \dots N_{K'}. \qquad (6)$$

These probabilities now express the probability of a feature vector arriving at the n/m-th bucket, respectively. We slightly abuse the used notation and keep original subscripts to make the derivation easier to follow without cluttering the used notation. In order to estimate the joint distribution $p_{(Y_T,Y_S)}(n,m|\mathbf{x})$ we need to employ the membership vectors to calculate the co-occurrence statistics. Therefore, we first define the co-occurrence matrix as:

$$\mathbf{C}_x = \frac{1}{N} \mathbf{d}_{T,i|\mathbf{x}} \otimes \mathbf{d}_{S,i|\mathbf{x}} \in \mathbb{R}^{N_K \times N_{K'}}, \qquad (7)$$

where $\otimes$ refers to the outer product of two vectors. It is easy to see that the matrix $\mathbf{C}_x$ calculates the probability of two vectors to be drawn from the two distributions based on the membership vectors. Then, we can trivially estimate the joint probability distribution as $p_{(Y_T,Y_S)}(n,m|\mathbf{x}) = C_{x,n,m}$.

The mutual information between the feature vector distributions for the teacher and student model can be re-defined using a discrete formulation and the probabilities estimated before as:

$$I(Y_T; Y_S|\mathbf{x}, k) = \sum_{n=1}^{N_K} \sum_{m=1}^{N_{K'}} \Bigg(p_{(Y_T,Y_S)}(n,m|\mathbf{x}) \cdot \log\left(\frac{p_{(Y_T,Y_S)}(n,m|\mathbf{x})}{p_{Y_T}(n|\mathbf{x}) p_{Y_S}(m|\mathbf{x})}\right)\Bigg). \qquad (8)$$

The aforementioned criterion can be applied at the same time to multiple layer pairs between the student and teacher networks leading to the following loss:

$$\mathcal{L}_{MI} = -\sum_{k \in \mathcal{K}} \alpha_k I(Y_T; Y_S|\mathbf{x}, k), \qquad (9)$$

where $\mathcal{K}$ denotes the set of layers that will be used for knowledge transfer and α_k is a hyper-parameter for controlling the relative weight of the k-th layer during the optimization. Note that the proposed mutual information loss can be defined only for feature distributions, i.e., it requires feature maps to be calculated. To this end, it can be also combined with other loss functions, e.g., cross-entropy loss for classification tasks or distillation loss [5] for further improving the robustness of the student network, as: $\mathcal{L} = \mathcal{L}_{task} + \mathcal{L}_{MI}$, where $\mathcal{L}_{task}$ describes the loss for solving the task at hand. Note that the whole formulation is fully differential and can be optimized in an end-to-end fashion using regular back-propagation.

3. EXPERIMENTAL EVALUATION

The experimental evaluation of the proposed method is provided in this Section. First, we introduce the used datasets and evaluation setup, and then we proceed by reporting the results of the conducted experiments. We used two well known image datasets for the conducted experiments, CIFAR-10 and CIFAR-100 datasets [13]. For both datasets we normalized the input images using standard scaling, while we employed image augmentation when training the teacher networks and extended that to the student network for the CIFAR-100 dataset (horizontal flips and random crop with a padding of 4 pixels).

The proposed method was evaluated using a lightweight VGG-based architecture that was proposed for high resolution analysis [14]. This architecture is fully convolutional, allowing it to scale to arbitrary input sizes. The base receptive field of the architecture is 32×32, while it consists of four 3×3 convolutional layers with 16, 16, 24, and 16 filters.

For all layers padding of 1 is used, while 2×2 pooling is used after the second and fourth layer. The output is fed into a final classification layer that flattens the output of the previous layers using a convolutional layer with N_C filters of size 8×8, where N_C is the number of used classes. The aforementioned architecture was used for defining the student networks. For the teacher networks we used the same architecture after increasing the number of filters by 3 times. We used the ReLU activation function for the intermediate layers [15] and the softmax function for the output layer. For the student networks, we target a physically realizable activation function, the photonic sinusoidal activation [16], to evaluate the effectiveness of the proposed method on such more challenging scenarios where the network is deployed on photonic neuromorphic neural network accelerators. The networks were optimized using the Adam optimizer with a learning rate of 0.0001 using the default parameters [17].

Instead of maximizing the mutual information for all the layers at once, we opted for a layer-wise optimization setup, which led to slightly better results in our experiments. More specifically, we first optimize the mutual information between the first layers of the networks for 50 epochs, next we proceed to the second layers for another 50 epochs, etc. This process is repeated for the four layers of the network for a total of 200 epochs. We also pretrained the networks for 10 epochs (20 for the CIFAR100 dataset) before learning the BoF representation in order to reduce the impact of potential distribution shifts during the training process. The number of BoF codewords was set to $N_K = N_{K'} = 12$ for all the layers. Using more codewords can improve the performance, especially for more complex datasets. However, we opted for using a small number of codewords in order to keep the complexity of the proposed method as low as possibly. At this point, it is worth noting that the proposed method does have any impact on the inference time/complexity, since it is only employed during the training phase. As a result, it does not affect the inference complexity of the models.

We compared the proposed method with three other methods: a) using regular training using the cross-entropy loss, b) using the knowledge distillation process [5], which can only transfer the knowledge between the output layers of the network, and b) using the PKT method that can transfer the knowledge between arbitrary-sized representations extracted from neutral networks [8]. To ensure a fair comparison the same number of training epochs was used for all methods. For distillation, we used a temperature of 2. For all the conducted experiments, we used the same weight value for all layers, i.e., $a_1 = a_2 = ... = a$, where we set $a = 4$ after performing validation experiments. Similarly, the distillation and PKT losses were weighted by 0.5 in the final loss function. For the PKT approach, we flattened the feature maps before transferring the knowledge, since it cannot directly handle feature maps composed of multiple feature vectors, while PKT loss was also combined with cross-entropy.

Table 1. Evaluation on CIFAR-10 and CIFAR100 datasets (mean accuracy and standard deviation of five runs are reported). Best results are reported in bold, second best results are underlined.

Method	**CIFAR 10**	**CIFAR100**
Cross-entropy	73.03 ± 1.03	41.80 ± 0.90
KD [5]	73.40 ± 0.46	42.77 ± 0.65
PKT [8]	72.80 ± 0.54	41.33 ± 1.08
Proposed	$\underline{74.60 \pm 0.59}$	$\underline{42.85 \pm 0.72}$
Proposed + KD	$\mathbf{74.73 \pm 0.79}$	$\mathbf{43.33 \pm 0.19}$

The experimental results are reported for CIFAR-10 in Table 1. Note that the employed architecture, which is tailored towards mobile deployment on embedded devices with restricted resources [14], has a quite constrained learning capacity, since training with the regular cross-entropy achieves an accuracy of about 73%. Training with the knowledge distillation approach (abbreviated as "KD") can improve the performance slightly. The PKT method on the other hand, despite being designed to transfer the knowledge between representation of arbitrary dimensionality, fails to improve the performance. We suspect that this is due to the high dimensionality of the flattened feature maps used before applying PKT, which reduces the effectiveness of density estimation used by PKT. Note that this does not affect the proposed method, since there is no need to flatten the extracted feature maps and the individual feature vectors are handled directly. Moreover, PKT is known to overly regularize the training process when the layers are not correctly matched [18]. On the other hand, the proposed method improves the accuracy over 1% over the next best-performing method. Furthermore, when combined with the KD method further improvements are observed. Repeating the experiments using ReLU activations for the student leads to almost the same results, i.e., about 75% for the proposed method and 73.5% for the baseline network. Finally, similar results are also reported for the CIFAR-100 dataset, in which the proposed method outperforms the rest of the evaluated methods.

4. CONCLUSIONS

In this work, we presented a fully differentiable and easy to optimize in a end-to-end manner formulation of mutual information, which can be used for transferring the knowledge between feature distributions of different networks. The effectiveness of the proposed method was demonstrated using two image datasets, especially compared to the PKT method that also relies on an information-theoric formulation, but fails to work with enormous feature maps. The proposed method is model-agnostic and can be combined with most of the other distillation methods. Finally, it can be also used with other network architectures that generate multiple feature vectors, such as 1-D convolution layers [19], and recurrent ones [20].

5. REFERENCES

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[3] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al., "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.

[4] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 126–136, 2018.

[5] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al., "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[6] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

[7] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[8] Nikolaos Passalis and Anastasios Tefas, "Learning deep representations with probabilistic knowledge transfer," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 268–284.

[9] Josef Sivic and Andrew Zisserman, "Video google: A text retrieval approach to object matching in videos," in *IEEE International Conference on Computer Vision*, 2003, vol. 3, pp. 1470–1470.

[10] Hervé Jégou, Matthijs Douze, and Cordelia Schmid, "Improving bag-of-features for large scale image search," *International journal of Computer Vision*, vol. 87, no. 3, pp. 316–336, 2010.

[11] Nikolaos Passalis and Anastasios Tefas, "Learning bag-of-features pooling for deep convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5755–5763.

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

[13] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," 2009.

[14] Maria Tzelepi and Anastasios Tefas, "Graph embedded convolutional neural networks in human crowd detection for drone flight safety," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 2, pp. 191–204, 2019.

[15] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[16] Nikolaos Passalis, George Mourgias-Alexandris, Apostolos Tsakyridis, Nikos Pleros, and Anastasios Tefas, "Training deep photonic convolutional neural networks with sinusoidal activations," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 3, pp. 384–393, 2019.

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

[18] Nikolaos Passalis, Maria Tzelepi, and Anastasios Tefas, "Heterogeneous knowledge distillation using information flow modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2339–2348.

[19] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman, "1D convolutional neural networks and applications: A survey," *Mechanical Systems and Signal Processing*, vol. 151, pp. 107398, 2021.

[20] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.