Shannon Strikes Again! Entropy-based Pruning in Deep Neural Networks for Transfer Learning under Extreme Memory and Computation Budgets

Gabriele Spadaro[1] Riccardo Renzulli[1] Andrea Bragagnolo[3] Jhony H. Giraldo[2]
Attilio Fiandrotti[1] Marco Grangetto[1] Enzo Tartaglione[2]
[1] University of Turin, Computer Science Department, Italy
[2] LTCI, Télécom Paris - Institut Polytechnique de Paris, France
[3] Independent Researcher
gabriele.spadaro@unito.it

Abstract

Deep neural networks have become the de-facto standard across various computer science domains. Nonetheless, effectively training these deep networks remains challenging and resource-intensive. This paper investigates the efficacy of pruned deep learning models in transfer learning scenarios under extremely low memory budgets, tailored for TinyML models. Our study reveals that the source task's model with the highest activation entropy outperforms others in the target task. Motivated by this, we propose an entropy-based Efficient Neural Transfer with Reduced Overhead via PrunIng (ENTROPI) algorithm. Through comprehensive experiments on diverse models (ResNet18 and MobileNet-v3) and target datasets (CIFAR-100, VLCS, and PACS), we substantiate the superior generalization achieved by transfer learning from the entropy-pruned model. Quantitative measures for entropy provide valuable insights into the reasons behind the observed performance improvements. The results underscore ENTROPI's potential as an efficient solution for enhancing generalization in data-limited transfer learning tasks.

1. Introduction

Neural networks have emerged as a universal tool for various computer vision tasks, achieving state-of-the-art performance in image classification, object detection, and other visual recognition challenges [**?**, 5, 15]. However, harnessing the full potential of these networks often proves challenging due to their complexity and the immense data requirements for training. To address this, researchers have sought compact models to alleviate computational burdens and improve efficiency [1, 7, 13, 21]. Among such approaches, pruning has emerged as a promising technique, reducing neural network size by removing unnecessary connections [2, 4, 8, 11–13, 19, 23]. However, while pruning contributes to model efficiency, it may not always suffice, particularly in scenarios with limited data availability.

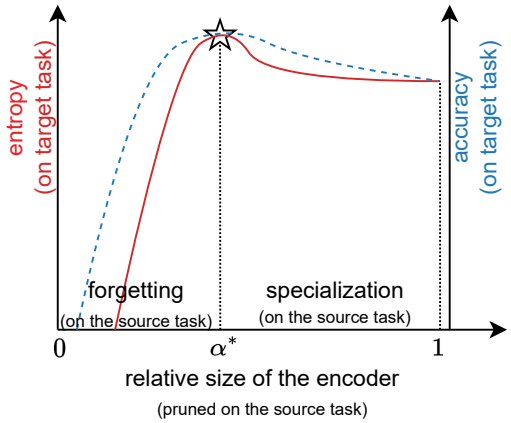

Figure 1. A pruned encoder exhibits better performance than a dense one in transfer learning scenarios, which correlates with the entropy of its representations. The peak of performance in the target task empirically correlates to the peak of the entropy.

In recent years, transfer learning has emerged as a pivotal approach to bridge the gap between data scarcity and effective model training, leveraging pre-trained models on large-scale datasets as a foundation for new tasks, leading to faster convergence and improved generalization [14, 22, 24]. Nonetheless, the substantial parameter inefficiencies arising from the large size of pre-trained models remain a challenge. To address this issue, some studies have explored the combination of pruning and transfer learning. For instance, DiffPruning [3] addresses parameter inefficiencies by extending the fixed pre-trained base model through a task-specific vector. Similarly, TransTailor [10] introduces pruning of the pre-trained model to enhance transfer learning, focusing on the structure mismatch between the model

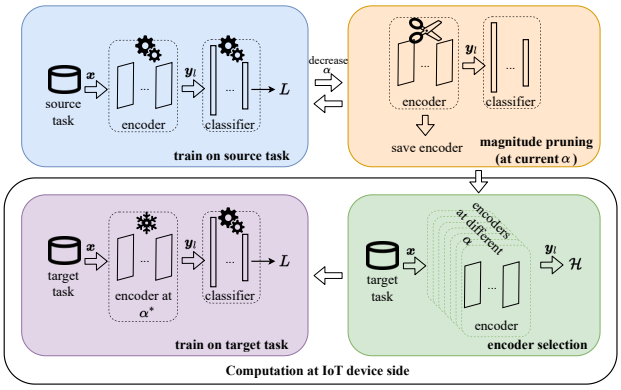

Figure 2. Schematic representation of the proposed ENTROPI algorithm. This approach trains a neural network in the source task in synergy with iterative magnitude pruning (with decreasing values of α). An encoder selection chooses the pruned model with the highest entropy, and finally, the pruned model is fine-tuned in the target task.

trained on the source domain and the target task. However, despite these advances, state-of-the-art methods still grapple with high computational complexity. More specifically, this poses a critical challenge in developing and deploying TinyML models. As TinyML aims to run machine learning algorithms on resource-constrained devices such as microcontrollers, wearables, and Internet of Things (IoT) devices, these platforms often have limited memory capacities [17]. Reducing computation and memory requirements is critical, as optimizing on the full architecture for fine-tuning is in most cases impossible for devices having less than one MB for memory, and optimizing on the classifier layer only is one of the few viable options (see Table 2 for real measures).

In this paper, our objective is to identify a pruned model on the source domain that outperforms the dense network, leading to a more computationally efficient finetuning step for the target task. To achieve this, we adopt the model with the highest activation entropy, which results in superior performance in the target task. The compact nature of pruned models endows them with enhanced generalization capabilities and adaptability to novel tasks, owing to their more general expressivity for generated features, as illustrated in Fig. 1. Our claim is supported by quantitative measures of entropy, which shed light on the structural differences between pruned and dense models, providing insights into why transfer learning from pruned models yields superior performance. Consequently, we introduce an Efficient Neural Transfer with Reduced Overhead via PrunIng (ENTROPI) algorithm for transfer learning with pruned neural networks. Notably, we identify two regions based on the encoder size: a *specialization* region, where the source task benefits from the encoder's capacity to aid the classifier by projecting features into a lower-dimensional space, and a *forgetting* region, where the model complexity hampers the extraction of relevant information for the source task, resulting in deteriorated performance. Intriguingly, configurations close to the critical point α^*, where α is the fraction of non-zero parameters, demonstrate better performance across different target tasks, further reinforcing the effectiveness of our approach.

Our contributions can be summarized as follows: firstly, we introduce an entropy-based pruning mechanism for transfer learning, a novel approach that identifies the more suitable backbone through a simple forward-propagation evaluation on the target task. Secondly, we offer insights into the behavior of entropy concerning the source tasks, revealing that the encoder utilizes its additional complexity beyond the critical point α^* to facilitate feature projection into a smaller, more compact subspace, thereby simplifying the classification problem while becoming more specialized on the source task. Thirdly, we observe that despite potentially high entropy values for latent space representations, their *exact entropy* remains relatively small, confirming the previous point. Lastly, we present empirical validation through experiments conducted on popular architectures, including ResNet18 [5] and MobileNetv3 [6].

2. Efficient Neural Transfer with Reduced Overhead via PrunIng (ENTROPI)

Fig. 2 illustrates the overall scheme of the ENTROPI algorithm, which consists of four main phases: i) training on the source task, ii) iterative magnitude pruning at the current α, progressively reduced by a factor $\hat{\alpha}$ in each iterative step, iii) encoder selection for the target task according to the entropy estimation on the target training, and iv) training the classifier head for the target task.

2.1. Preliminaries

Let $\mathcal{E}(\cdot)$ be the encoder of the neural network in Fig. 2 and let $\boldsymbol{x}$ be some sample from our dataset $\mathcal{D}$. $y_{l,i}^{\boldsymbol{x}}$ is the output of a given i-th neuron at the l-th layer for the input $\boldsymbol{x} \in \mathcal{D}$ given as follows:

$$y_{l,i}^{\boldsymbol{x}} = \varphi\left[f(\boldsymbol{y}_{l-1}^{\boldsymbol{x}}, \boldsymbol{\theta}_{l,i})\right], \tag{1}$$

where $\boldsymbol{\theta}_{l,i}$ are the parameters associated to the l-th layer, $f(\cdot)$ is some affine function, $\varphi(\cdot)$ is the activation function, and $\boldsymbol{y}_{l-1}^{\boldsymbol{x}} \in \mathbb{R}^{N_{l-1}}$ is the input of such neuron with N_{l-1} the number of outputs provided by the layer $l-1$.

2.2. Entropy Estimation

Here, we provide details on how we compute the entropy of the activations inside the neural network model.

Let $\boldsymbol{y}_l^{\boldsymbol{x}} \in \mathbb{R}^{N_l}$ represent the output of the l-th layer in our neural network, with N_l denoting the number of outputs for that layer. We assume that N_l is constant for all $\boldsymbol{x} \in$

$\mathcal{D}$, *i.e.*, the dimensionality of the input is always constant. Therefore, we define the quantization index $q_{l,i}^{\boldsymbol{x}}$ as follows:

$$q_{l,i}^{\boldsymbol{x}} \triangleq \left\lfloor \frac{y_{l,i}^{\boldsymbol{x}}}{\Delta} + \frac{1}{2} \right\rfloor, \tag{2}$$

where Δ is the step size defined as:

$$\Delta = \frac{\max_{\boldsymbol{x},i}\{y_{l,i}^{\boldsymbol{x}}\} - \min_{\boldsymbol{x},i}\{y_{l,i}^{\boldsymbol{x}}\}}{M}, \tag{3}$$

and M being the quantization levels. Therefore, we can reconstruct the original output through $\tilde{y}_{l,i}^{\boldsymbol{x}} = \Delta q_{l,i}^{\boldsymbol{x}}$.

Let us define the *state* of the pattern $\boldsymbol{x}$ at the output of the encoder as its quantized representation. We can extend the quantization index for the whole layer as follows:

$$\boldsymbol{q}_l^{\boldsymbol{x}} = [q_{l,1}^{\boldsymbol{x}}, q_{l,2}^{\boldsymbol{x}}, \cdots, q_{l,N_l}^{\boldsymbol{x}}]. \tag{4}$$

Let $|\mathcal{D}|$ be the cardinality of our dataset, *i.e.*, the number of samples in $\mathcal{D}$. For every possible state $\boldsymbol{\kappa}$ for l, we can count its number of occurrences within $|\mathcal{D}|$, estimating the frequency distribution for every possible state $\boldsymbol{\kappa}$ as follows:

$$p(\boldsymbol{\kappa}, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \mathbb{1}_{\boldsymbol{\kappa}}(\boldsymbol{q}_l^{\boldsymbol{x}}), \tag{5}$$

where $\mathbb{1}(\cdot)$ is the indicator function. At this point, we can use the definition in (5) to write the entropy $\mathcal{H}(\mathcal{D})$ for the output of the encoder as follows:

$$\mathcal{H}(\mathcal{D}) = -\sum_{\boldsymbol{\kappa}} p(\boldsymbol{\kappa}, \mathcal{D}) \log[p(\boldsymbol{\kappa}, \mathcal{D})]. \tag{6}$$

We know by construction that this quantity is bounded between zero and $\log\left(M^{N_l}\right)$.

2.3. Pruning has an Impact on the Entropy

To reduce the number of parameters in a deep neural network, a prevalent approach involves thresholding based on specific hyperparameters, which determines the number of parameters to be eliminated [2, 19]:

$$\theta_{l,i,j} = \begin{cases} \theta_{l,i,j} & |\theta_{l,i,j}| > \mathcal{Q}_{|\boldsymbol{\theta}|}(1-\alpha), \\ 0 & \text{otherwise}, \end{cases} \tag{7}$$

where $\mathcal{Q}_{|\boldsymbol{\theta}|}(\cdot)$ is the quantile function for the ℓ_1 norm of the parameters, and $(1-\alpha) \in [0, 1]$ is the fraction of parameters to be removed. It is important to highlight that numerous attempts have been made to propose more effective pruning strategies; however, iterative magnitude pruning, despite being relatively costly, offers the best trade-off in terms of approach complexity and performance. This phenomenon has been extensively discussed in the literature, as evidenced by studies such as [16,20]. Consequently, we have chosen to adopt iterative magnitude pruning as our pruning approach, and we refrain from conducting an ablation study on alternative methods.

During the pruning process, as we approach a certain relative size α^*, we can anticipate that the entropy of $\boldsymbol{y}$ will be upper-bounded. Specifically, we can establish that:

$$\mathcal{H}(\mathcal{D}) \leq \sum_{i=1}^{N_l} \mathcal{H}_i(\mathcal{D}) = -\sum_{i=1}^{N_l} \sum_{\kappa=1}^{M} p_i(\kappa, \mathcal{D}) \log[p_i(\kappa, \mathcal{D})], \tag{8}$$

where the term $p_i(\kappa, \mathcal{D})$ is determined by

$$p_i(\kappa, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \delta_{\kappa, q_{l,i}^{\boldsymbol{x}}}, \tag{9}$$

where $\delta_{i,j}$ is the Kronecker delta. Therefore, expanding (2), we get:

$$q_{l,i}^{\boldsymbol{x}} = \left\lfloor \frac{\varphi\left[f(\boldsymbol{y}_{l-1}^{\boldsymbol{x}}, \boldsymbol{\theta}_{l,i})\right]}{\Delta} + \frac{1}{2} \right\rfloor. \tag{10}$$

We know that for a sufficiently small Δ and a sufficiently large $\mathcal{D}$, in a model trained to *extract* information from the input, the entropy of the deeper layers is smaller or equal than one of the shallower ones (because of information filtering) [18]. We expect that, when having low pruning regimes at high α, the entropy of the signal is low as the encoder $\mathcal{E}(\cdot)$ is also partially aiding for the classification (or rather, for the target task), projecting the signal into a low-dimensional space. As α drops, however, the encoder will not have enough parameters to effectively project the input signal: the signal becomes sparser, and the entropy grows. This behavior is however also upper-bounded by the pruning ratio: in a fully-connected layer, if the number of parameters is lower than the size of the input, then necessarily the entropy of the layer can not be at its largest value and is bounded by a constant value in a low pruning ratio regime, while it grows linearly with increasing pruned parameters.

To summarize, we expect two different trends for the entropy of the output of the encoder: i) as α decreases (until a critical α^*) the entropy of the output's representations grows, as the encoder, having progressively fewer and fewer parameters, becomes unable to effectively project the signal into a low-dimensionality space, useful for the target task, and ii) for sizes smaller than α^*, the trend reverts: the entropy drops as the parameter complexity is insufficient to propagate enough information, and with that also the performance on the trained task drops.

3. Experiments and Results

This section presents the experimental setup and results of the paper. We conducted transfer learning experiments by training and pruning two Convolutional Neural Network (CNN) models from scratch on the CIFAR-10 dataset. We

Table 1. Top-1 accuracy (↑) on transfer learning for ResNet18 and MobileNet-v3.

Dataset	ResNet18		MobileNet-v3	
	Dense	Pruned	Dense	Pruned
CIFAR-100	$14.46_{\pm 0.27}$ ($\alpha = 1.0$)	$\mathbf{21.83}_{\pm 0.09}$ ($\alpha = 0.0077$)	$13.91_{\pm 0.02}$ ($\alpha = 1.0$)	$\mathbf{17.36}_{\pm 0.09}$ ($\alpha = 0.0038$)
VLCS	$48.73_{\pm 1.25}$ ($\alpha = 1.0$)	$\mathbf{49.51}_{\pm 1.01}$ ($\alpha = 0.0186$)	$45.22_{\pm 0.94}$ ($\alpha = 1.0$)	$\mathbf{53.84}_{\pm 0.79}$ ($\alpha = 0.0158$)
PACS	$18.57_{\pm 8.94}$ ($\alpha = 1.0$)	$\mathbf{19.49}_{\pm 8.14}$ ($\alpha = 0.0085$)	$30.16_{\pm 1.61}$ ($\alpha = 1.0$)	$\mathbf{32.59}_{\pm 2.62}$ ($\alpha = 0.0106$)

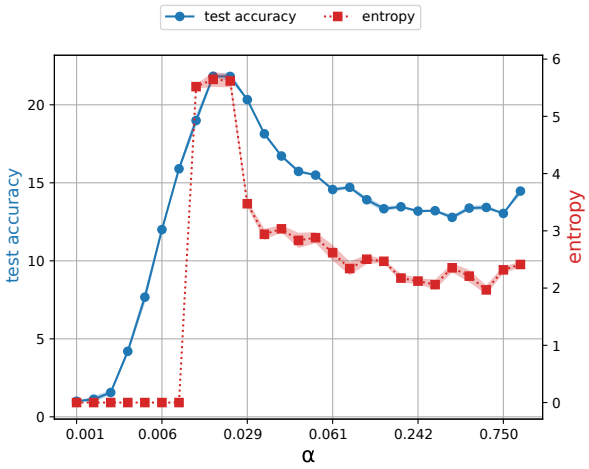

Figure 3. Entropy and Top-1 test accuracy on Cifar-100 using ResNet18 at different α values.

trained with batches of size 128 using Adam, with learning rate 10^{-5} and exponential decay of 0.99. Subsequently, we used the pruned models' backbones as feature extractors and added new classification layers. The entire backbone was frozen, and the models were trained on three other datasets: PACS, VLCS, and CIFAR-100. Specifically, we evaluated two different CNN architectures: ResNet18 and MobileNetv3. To obtain the pruned backbones, we employed a batch size of 128 and used Stochastic Gradient Descent (SGD) as the optimizer with a momentum of 0.9 and a weight decay of 10^{-4}. The learning rate of 0.1 is decayed with a cosine-annealing schedule during the training duration of 90 epochs. For all the experiments we have employed $M = 4$ and we progressively decrease α by 0.25.

Results. The results are shown in Table 1, where all the experiments are averaged over three seeds. We observe that for the three datasets, we consistently observe an improvement in the performance when compared to employing the dense model, with Top-1 gains up to the 8%. Interestingly, we observe that the chosen values of α are extremely low, in the order of 0.01. This clearly indicates that having sparser backbones (which exhibit as well maximum entropy on the target task) improves performance.

Ablation on α and computation estimation. We conduct an ablation study where we employ all pruned encoders (having hence a study for different values of α) using ResNet18 and the target dataset CIFAR-100. Fig. 3 shows the results of this ablation study, where all the points are averaged on three different seeds. We observe that the accuracy on the test set reaches its peak in correspondence with the maximum entropy, calculated on the target train set. This further validates our thesis that training an off-the-shelf pruned model selected using ENTROPI is advantageous. We remark that the achieved performance is still far from the baselines as we constrain our optimization problems to be at an extremely low memory budget. Table 2 compares FLOPs and back-propagation memory footprint for the tested architectures on VLCS. We clearly observe that in our challenging setup, where we are allowed only to tune the classification head, the memory footprint required is extremely low, in the order of tens of kB, while for full fine-tuning, we are required even hundreds of MB. This makes the proposed approach suitable in the most extreme contexts, like embedded AI or even TinyML [9].

Table 2. FLOPs and memory for backpropagation (↓) on VLCS.

Model	Frozen	FLOPs (M)	Mem. Foot. (MB)
ResNet18	✓	**0.16**	**0.03**
	✗	30.03	131.49
MobileNet-v3	✓	**0.41**	**0.08**
	✗	229.05	48.23

4. Conclusions

In this paper, we conducted a study on transfer learning scenarios, aiming to identify an effective pruning strategy for enhancing performance in the target task. Our key finding was that among the models from the source task, the one with the highest activation entropy consistently outperformed others in the target task. Building on this observation, we introduced the ENTROPI algorithm. To validate the effectiveness of the proposed approach, we conducted extensive experiments on diverse models, including ResNet18 and MobileNetv3, and tested on various target datasets: CIFAR-100, VLCS, and PACS. The results consistently showcased the superiority of transfer learning from the entropy-pruned model, highlighting the potential of our approach in data-limited transfer learning tasks, and paving the way to the employment of large architectures for downstream tasks adaptation directly on IoT devices.

References

[1] Andrea Bragagnolo, Enzo Tartaglione, and Marco Grangetto. To update or not to update? neurons at equilibrium in deep models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22149–22160. Curran Associates, Inc., 2022. 1

[2] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. 1, 3

[3] Demi Guo, Alexander Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, 2021. 1

[4] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, 2015. 1

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016. 1, 2

[6] 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 *IEEE/CVF International Conference on Computer Vision*, 2019. 2

[7] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, 1990. 1

[8] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017. 1

[9] Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and song han. On-device training under 256KB memory. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 4

[10] Bingyan Liu, Yifeng Cai, Yao Guo, and Xiangqun Chen. Transtailor: Pruning the pre-trained model for improved transfer learning. *AAAI Conference on Artificial Intelligence*, 2021. 1

[11] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018. 1

[12] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *IEEE/CVF International Conference on Computer Vision*, 2017. 1

[13] D. Molchanov, A. Ashukha, and D. Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, 2017. 1

[14] Partha Pratim Ray. A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, 34(4):1595–1623, 2022. 1

[15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015. 1

[16] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020. 3

[17] Ramon Sanchez-Iborra and Antonio F Skarmeta. TinyML-enabled frugal smart objects: Challenges and opportunities. *IEEE Circuits and Systems Magazine*, 2020. 2

[18] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017. 3

[19] Enzo Tartaglione, Andrea Bragagnolo, Attilio Fiandrotti, and Marco Grangetto. Loss-based sensitivity regularization: Towards deep sparse neural networks. *Neural Networks*, 2022. 1, 3

[20] Enzo Tartaglione, Andrea Bragagnolo, and Marco Grangetto. Pruning artificial neural networks: A way to find well-generalizing, high-entropy sharp minima. In *International Conference on Artificial Neural Networks*, 2020. 3

[21] K. Ullrich, M. Welling, and E. Meeds. Soft weight-sharing for neural network compression. In *International Conference on Learning Representations*, 2019. 1

[22] Yinghao Wang, Rémi Nahon, Enzo Tartaglione, Pavlo Mozharovskyi, and Van-Tam Nguyen. Optimized preprocessing and tiny ml for attention state classification, 2023. 1

[23] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. In *International Conference on Learning Representations*, 2018. 1

[24] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 2021. 1