MULTI-TASK ADAPTERS FOR ON-DEVICE AUDIO INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

The deployment of deep networks on mobile devices requires to efficiently use the scarce computational resources, expressed as either available memory or computing cost. When addressing multiple tasks simultaneously, it is extremely important to share resources across tasks, especially when they all consume the same input data, e.g., audio samples captured by the on-board microphones. In this paper we propose a multi-task model architecture that consists of a shared encoder and multiple task-specific adapters. During training, we learn the model parameters as well as the allocation of the task-specific additional resources across both tasks and layers. A global tuning parameter can be used to obtain different multi-task network configurations finding the desired trade-off between cost and the level of accuracy across tasks. Our results show that this solution significantly outperforms a multi-head model baseline. Interestingly, we observe that the optimal resource allocation depends on both the task intrinsic characteristics as well as on the targeted cost measure (e.g., memory or computing cost).

1 INTRODUCTION

The availability of large annotated audio datasets (e.g., AudioSet (Gemmeke et al., 2017)) has enabled to train models that are able to target a large number of audio classes, ranging from speech and music, to coughing and baby crying. However, to achieve a good level of accuracy, it is necessary to use complex network architectures, characterized by a large number of parameters and floating point operations (FLOPs) (Hershey et al., 2017). For this reason, when models need to be deployed on mobile device, it is customary to train more focused detectors, each targeting a handful of classes. This approach has two main advantages: i) the resulting model is typically significantly less complex, thus lending itself to be deployed on device; ii) training can leverage task-specific datasets and data augmentation strategies, thus leading to a higher level of accuracy when deployed *in-the-wild*.

On the other side, training and deploying independent models for each task fails to leverage the fact that such models might be extracting common features, given that they all consume the same input. Indeed, it might be argued that especially in the early layers of the network architecture, models might be learning low-level features that are not task-specific. As a consequence, such independent models do not make optimal use of the scarce computational resources.

A common solution to this problem is to deploy a multi-head model (Georgiev et al., 2017; Lee et al., 2019), in which a shared common encoder computes general-purpose audio embeddings, and task-specific fully connected heads are added to target each task. However, when the number and heterogeneity of tasks increases, the audio embeddings might fail to capture all the information needed to solve all tasks. In this paper we propose a multi-task model that overcomes the aforementioned issue by adding task adapter networks in parallel to a shared encoder, as illustrated in Figure 1. The goal of such adapters is to learn task-specific features, possibly at different depths. The adapters have the same architecture as that of the shared encoder, but with a smaller number of channels in each layer. We designed the architecture in such a way that each layer in a task adapter receives as input the concatenation of the activations at the layer below, computed by both the shared encoder and the task adapter itself. As a consequence, there are no inter-dependencies across tasks, and during inference one can decide to compute simultaneously either all tasks or a subset of them, depending on the available resource budget.

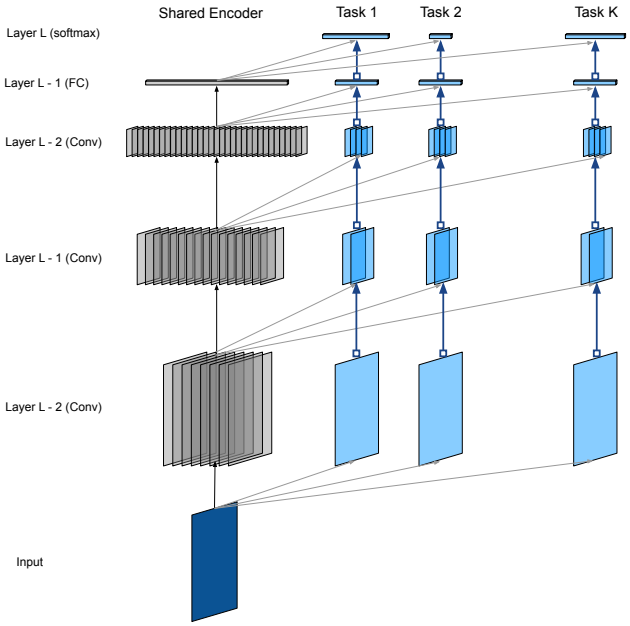


Figure 1: Overview of the proposed model architecture with task adapters. Each parallelogram denotes a 2-dimensional channel (time and frequency). Arrow with square ending denote gating variables that control which subset of the channels contribute to the input of the layer above.

Generally, tasks might be characterized by a different level of intrinsic difficulty, and require adaptation at different layers in the network. A fixed allocation of extra channels is likely to be suboptimal when costs are explicitly taken into account. Thus, our key contribution is to let the network learn which additional channels to use in each layer of each task adapter network, subject to a global cost constraint. Note that cost can be expressed either in terms of number of parameters or FLOPs, depending on the requirements imposed by the deployment on device (respectively due to memory or battery constraints).

Our solution consists of introducing an explicit gating mechanism, controlled by a small set of trainable variables that determine whether each channel of the task adapters is used as input to the layer above. By learning such gating variables, the model can effectively decide to turn off some of the channels in the task adapter networks, thus learning how to allocate the available budget to tasks and layers. In summary, we propose the following main contributions:

- We propose a model that addresses multiple audio tasks simultaneously, sharing representations via a common encoder network and learning task-specific adapters at different depths, which are able to augment the common representation and achieving higher accuracy.
- We propose a learnable gating mechanism that allows one to sweep different trade-offs between accuracy and overall cost, by selectively turning off some of the channels in the task adapter networks.
- We evaluate the proposed model simultaneously on eight different audio tasks, ranging from keyword spotting to audio scene recognition, speaker identification, etc. Our empirical results show that it is possible to significantly improve the level of accuracy of several tasks with respect to a multi-head model by only marginally increasing the cost.

2 Related work

Learning representations that can be re-used for multiple tasks has received a great deal of attention in the recent literature. Domain adaptation and transfer learning (Yosinski et al., 2014; Donahue et al., 2014) are common methods used to fine-tune a linear classifier on top of the embeddings

produced by a pre-trained network to address multiple tasks. An alternative approach consists of full fine-tuning (Cui et al., 2018), in which a pre-trained network is used as starting point for the training process. However, when multiple tasks need to be addressed, neither solution is particularly suitable. In the first case, task-adaptation is limited to the output layer of the network, which might not be sufficient when tasks require heterogeneous representations. In the second case, full fine-tuning might lead to very different models for each task, due to catastrophic forgetting. To overcome these limitations, Rebuffi et al. (2017) address the problem of adapting a common representation to different visual domains. They propose to use residual adapter modules, i.e., parametric modules that can steer the internal network representation from one domain to another. This approach was later extended in (Rebuffi et al., 2018), introducing a form of adapter that can be added in parallel to the main network architecture, and successfully applied to the NLP domain in (Houlsby et al., 2019). An alternative approach is proposed in (Mudrakarta et al., 2019), in which a task-specific model patch is learned to produce different embeddings for different downstream tasks. The patch can take the form of either batch-norm parameters or a subset of the weights of spatial convolution filters. All these methods allow to adapt the network by changing a small number of weights. At the same time, during inference the whole network has to be reevaluated from scratch when moving from one task to the other, due to the dependencies introduced in the computation graph. This is in constrast with our model, which is able to target simultaneously multiple tasks at once.

Most of the works above deal with vision-related tasks. Multi-task learning in the context of general-purpose audio has been less explored. The prevailing approach is to train a single model addressing multiple classes at once (Hershey et al., 2017). However, this approach does not benefit from the availability of task-specific datasets, and model capacity might not be tailored to the subset of classes of interest. Recently, Lee et al. (2019) proposed a model architecture that addresses simultaneously three tasks. Unlike our approach, they start directly from time-domain waveforms. In addition, the task adaptation only occurs in the last layer of a multi-head model architecture. Similarly to our work, Georgiev et al. (2017) address multi-task audio learning for deployment on embedded devices. Depending on their characteristics, tasks can be processed by a multi-head model, in which only the last layer is task-specific, or have its own task-specific network. Conversely, our model can accommodate task adaptation at different depths and in a task-specific manner.

In our work we deliberately keep the task adapters of individual tasks separate from each other, so that it is possible to select the subset of tasks to evaluate depending on the available budget. This is in constrast to the approach explored by Cheung et al. (2019), which superimpose multiple models into a single entangled model, from which task-specific models can be later retrieved. At the same time, this approach seems to be more suitable for server-side inference, where the overall model complexity is less critical. In addition, while in our work we focus on a single modality, we recognize the importance of handling multiple modalities at once. For example, Kaiser et al. (2017) explored the case in which multiple tasks belong to different modalities (e.g., image, speech, text), showing that they might still benefit from sharing part of the network architecture when training is performed concurrently on all tasks. Also in this case the resulting model is large and not specifically tailored to be deployed on mobile devices.

Finally, the proposed method for determining how to size the adapters based on the available budget is related to the MorphNet solution previously appeared in (Gordon et al., 2018). However, our approach differs from multiple angles: i) a single-task learning model is discussed in (Gordon et al., 2018), while we focus on multi-task learning, thus investigating how allocation is performed across tasks; ii) we introduce explicit gating variables instead of re-using batch-norm scaling variables. This has the advantage of applying the solution also to layers in which batch norm might not be used (e.g., fully connected layers); iii) we adopt a different relaxation of the discrete cost allocation problem (further discussed in Section 3); iv) we evaluate the model in the context of audio tasks, while (Gordon et al., 2018) is mostly concerned with vision tasks.

3 METHODS

We consider a model architecture that receives one audio recording as input and produces as output predictions for K downstream tasks simultaneously. The architecture consists of a shared encoder and K task-adapter encoders. The underlying idea is that the shared encoder provides a general purpose representation for the audio inputs, which might be suitable for different downstream tasks.

However, higher level of accuracy might be achieved by refining the representations computed at different depths adding task-specific adapters in the form of additional channels.

The overall architecture of the model is illustrated in Figure 1. Both the shared encoder and each of the task adapters consist of the same number of convolutional layers, followed by a global max-pooling layer and a fully connected layer, for a total of L layers. Let $f_{k,i}(\cdot)$, $i = 1, \ldots, L$, denote the function computed by the generic layer at depth i. To simplify the notation, we denote with $k = 0$ the shared encoder and with $k = 1, \ldots, K$, the task specific encoders. The function $f_{k,i}(\cdot)$ produces as output a tensor of size $T_i \times F_i \times C_{k,i}$. Note that the number of temporal frames T_i and frequency bins F_i is the same for all values of k. For the task-specific encoders, we include a number of task-specific channels $C_{k,i} = max(1, \lfloor \alpha_i C_{0,i} \rfloor)$, where $C_{0,i}$ and α_i are hyperparameters that determine the maximum achievable complexity of the model. Although it is possible to use a different value of α_i at each layer, throughout the rest of this paper we assume $\alpha_i = \alpha$, $i = 1, \ldots, L$.

In the shared encoder, $f_{0,i}$ receives as input only the output of the previous layer. However, in each task-adapter encoder, $f_{k,i}$, $k \neq 0$, receives as input the concatenation of the outputs of $f_{0,i-1}$ and $f_{k,i-1}$ along the channel dimension. Therefore, the cost of computing $f_{k,i}$, $k \neq 0$, can be expressed as:

$$\text{cost}_{k,i} = \eta_{i,k} \cdot C_{k,i} \cdot (C_{0,i-1} + C_{k,i-1}) \tag{1}$$

(with $C_{0,0} = 1$, and $C_{k,0} = 0$, for $k \neq 0$). That is, the cost is proportional to the number of output channels $C_{k,i}$ multiplied by the number of input channels $(C_{0,i-1} + C_{k,i-1})$. The cost scaling factor $\eta_{i,k}$ is a constant value that can be computed based on: i) the intrinsic architecture of the layer; ii) the known sizes $T_i \times F_i$; iii) the target cost measure, i.e., FLOPs or number of parameters.

The proposed method aims at learning how to scale the number of channels to be used in each layer of the task adapters encoder, i.e., to determine $c_{k,i} \leq C_{k,i}$, subject to a constraint on the total cost. To this end, we introduce a gating mechanism that controls the flow of the activations in the task adapters encoders. Namely, for each layer of the task adapters we introduce $C_{k,i}$ additional trainable variables $\mathbf{a}_{k,i} = [a_{k,i,1}, \ldots, a_{k,i,C_{k,i}}]$, which modulate the output of each channel:

$$\tilde{f}_{k,i,c}(x) = \sigma(a_{k,i,c}) \cdot f_{k,i,c}(x), \tag{2}$$

where $\sigma(\cdot)$ is a non-linear function that maps its input to non-negative real numbers, i.e., $\mathbb{R} \rightarrow \mathbb{R}^+$. In our work, we use a clipped ReLU nonlinearity defined as follows:

$$\sigma(a; s) = \min(1, ReLU(s \cdot a + 0.5)) \tag{3}$$

The slope of the non-linearity s is progressively increased during training, in such a way that, as $s \rightarrow \infty$, (3) acts as a gating function. Note that when the gating non-linearity is driven to be either 0 or 1, it is *locked* at this value, as the gradients are equal to zero. Therefore, it performs a hard selection of those channels that are contributing to the network output and those that can be discarded. The number of active channels in the i-th layer of the k-th task adapter is equal to:

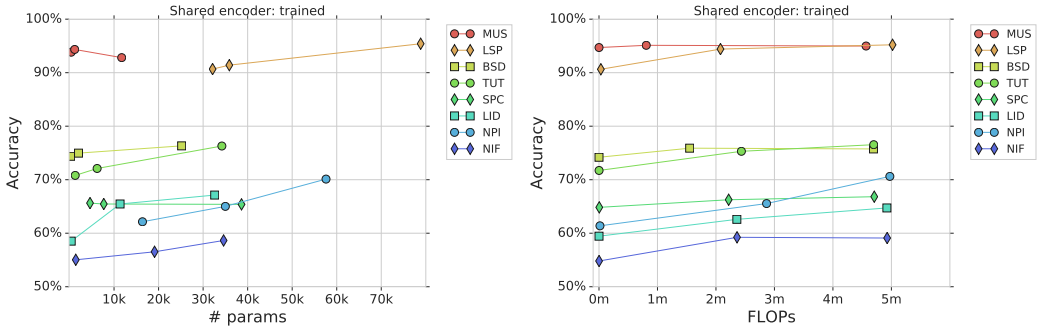
$$c_{k,i} = \sum_{c=1}^{C_{k,i}} \mathbb{1}_{\sigma(a_{k,i,c})>0} \tag{4}$$

During training, we jointly learn both the parameters of the network and the gating variables. This is achieved by optimizing the following loss function:

$$\mathcal{L} = \sum_{k=1}^{K} w_k \left[\mathcal{L}_k^{XE} + \lambda \mathcal{C}_k^{adapters} \right] \tag{5}$$

(a) Accuracy vs. cost.

		Num. params		FLOPs		
Task	Multi-head	$\lambda = 10^{-2}$	$\lambda = 10^{-4}$	$\lambda = 10^{-2}$	$\lambda = 10^{-4}$	Single-task
MUS	0.94	0.94 (+0.5%)	0.93 (-1.1%)	0.95 (+0.4%)	0.95 (+0.3%)	0.98
LSP	0.91	0.91 (+0.8%)	0.95 (+5.2%)	0.94 (+4.2%)	0.95 (+5.0%)	0.98
BSD	0.74	0.75 (+0.8%)	0.76 (+2.6%)	0.76 (+2.3%)	0.76 (+2.1%)	0.73
TUT	0.71	0.72 (+1.8%)	0.76 (+7.8%)	0.75 (+5.0%)	0.77 (+6.7%)	0.82
SPC	0.66	0.65 (-0.3%)	0.65 (-0.4%)	0.66 (+2.2%)	0.67 (+3.1%)	0.75
LID	0.59	0.65 (+11.9%)	0.67 (+14.7%)	0.63 (+5.3%)	0.65 (+8.9%)	0.64
NPI	0.62	0.65 (+4.6%)	0.70 (+12.8%)	0.66 (+6.8%)	0.71 (+15.0%)	0.79
NIF	0.55	0.57 (+2.7%)	0.59 (+6.6%)	0.59 (+8.1%)	0.59 (+7.8%)	0.63
Mean	0.71	0.73 (+2.9%)	0.75 (+6.0%)	0.74 (+4.3%)	0.75 (+6.1%)	0.79



(b) Number of parameters

(c) FLOPs

Figure 2: Accuracy vs. cost for the multi-task learning scenario.

where $\mathcal{L}_k^{XE}$ is the cross-entropy loss for the k-th task, w_k is an optional weighting term, and $\mathcal{C}_k^{adapters}$ is a penalty term that captures the cost of the k-th task adapter for a given configuration of the gating variables:

$$\mathcal{C}_k^{adapters} = \sum_{i=1}^{L} \eta_{i,k} \cdot \|\mathbf{a}_{k,i}\|_1 \cdot (C_{0,i-1} + \|\mathbf{a}_{k-1,i}\|_1). \tag{6}$$

The Lagrange multiplier λ controls indirectly the target cost, i.e., when $\lambda = 0$ the optimizer minimizes the cross-entropy loss $\mathcal{L}_k^{XE}$ only, thus potentially using all the available capacity, both of the shared encoder and of the task-adapter channels (i.e., $c_{k,i} = C_{k,i}$). Conversely, when increasing λ the use of additional channels is penalized, thus inducing the network to use fewer channels. Note that $\|\mathbf{a}_{k-1,i}\|_1$ is upper bounded by $\alpha\lfloor C_{0,i-1}\rfloor$, therefore when $\alpha \ll 1$, the second term in equation (6) is dominated by the constant $C_{0,i-1}$, and $\mathcal{C}_k^{adapters}$ is proportional to the l-1 norm of the gating variable vector, thus promoting a sparse solution in which only a subset of the channels are used.

4 Experiments

Audio front-end: In our work we consistently use the same audio frontend, which processes input sequences sampled at 16 kHz, with a window size of 25 ms and a hop size equal to 10 ms to compute the short-time Fourier transform (STFT), and then computes $F = 64$ mel-spaced frequency bins in the range 60–7800 Hz. Finally, we take the logarithm of the resulting spectrogram.

Audio tasks: We evaluate the proposed multi-task adapters architecture addressing simultaneously 8 different audio-based tasks, covering both speech and non-speech related tasks. In all cases, the model receives as input a spectrogram slice of size 96×64, so that the receptive field is equal to $T = 975$ ms. We use the *Speech Commands* (SPC) dataset (Warden, 2018) to evaluate keyword spotting on 35 distinct keywords. *LibriSpeech* (LSP) (Panayotov et al., 2015) contains audio books read by 251

different speakers. We use the 100 hours training set to evaluate a speaker identification task. The *Spoken Language Identification* (LID) dataset (Tomasz, 2018) contains samples that belong to three different languages: English, Spanish and German, while the *MUSAN* (MUS) dataset (Snyder et al., 2015) distinguishes across three classes, namely music, speech and noise. We also use two datasets released in the context of the recent DCASE2018 Challenge, *Bird Audio Detection* (Stowell et al., 2018) (BSD), and *TUT Urban Acoustic Scenes 2018* (Mesaros et al., 2018) (TUT), which contains labeled audio samples from 10 different urban environments. Finally, we consider two tasks based on the NSynth dataset (Engel et al., 2017). *NSynthPitch* (NPI) contains notes played by different musical instruments at 128 different pitch levels, while *NSynthInstrument* (NIF) distinguishes 11 different families of musical instruments. For all datasets, we consider the default train/test split, and provide results on slices extracted from the test set only. Note that the choice of the tasks used for the evaluation is consistent with the selected temporal granularity. As such, we do not consider speech recognition tasks, which generally require a much finer temporal granularity.

Model architecture: For both the shared encoder and the task-adapter networks, we use a convolutional neural network with $L = 5$ layers. Each convolutional layer is followed by max-pooling, to reduce the time-frequency dimensions by a factor of two at each layer, a ReLU non-linearity and batch-normalization. Finally, a global max-pooling layer is followed by a fully-connected layer. For each task, the output softmax layer receives as input the embeddings produced by the encoder, concatenated with the embeddings produced by the task-adapter network.

We consider two scenarios for the shared encoder architecture. In the *multi-task learning* scenario, the encoder is trained together with the task adapters. In this case, the number of channels in each layer is equal to $[6, 12, 24, 48, 96]$, for a total of 65k parameters and 6M FLOPs. In the *transfer learning* scenario, we consider embeddings produced by an encoder pre-trained using a self-supervised learning method (Audio2Vec - CBoW), as described in (Tagliasacchi et al., 2019). In this case we use $[8, 16, 32, 64, 128]$ channels in each layer and the encoder weights are frozen during training, for a total of 125k parameters and 18M FLOPs. The maximum number of channels in the task-adapter networks is determined by setting $\alpha = 0.2$.

The loss function is minimized with stochastic gradient descent using the Adam optimizer with a learning rate equal to 10^{-3}. The batch size is set equal to 256 samples, that is, 256 / 8 = 32 samples from each task in a batch. Training is stopped after 1 million batch iterations, when the level of accuracy of all tasks is saturated.

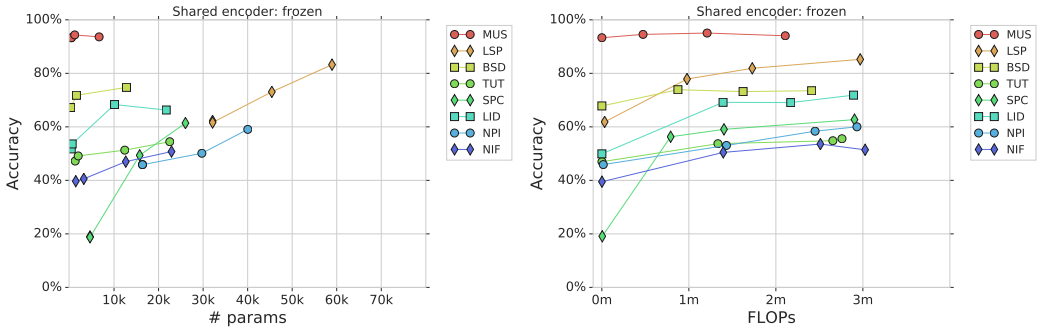
Baselines: As a baseline, we consider a multi-head architecture which consists of a shared encoder and 8 different fully connected layers, one for each task. We also include results obtained by training a task specific model. In this case, we use a model with $[8, 16, 32, 64, 128]$ channels in each layer for a total of 125k parameters and 18M FLOPs up to the embedding layer. The number of parameters of the output softmax layer depends on the number of output classes. For example, the *LibriSpeech* head requires additional $251 \times 128 \simeq 32k$ parameters, the *MUSAN* head only $3 \times 128 = 384$ parameters. The FLOPs cost of this layer is negligible when compared to the rest of the network.

Results: We evaluate the proposed model architecture by computing the classification accuracy of each of the 8 tasks. We consider cost expressed either as number of task-specific parameters, or task-specific FLOPs. Figure 2 shows the results for the *multi-task learning* scenario. We let the parameter λ vary, so as to target different cost levels, and report the task accuracy in each case. The leftmost point in each curve represents the multi-head baseline, and the other two points are obtained by setting $\lambda = 10^{-2}, 10^{-4}$, when expressing number of parameters in thousands and FLOPs in millions. Note that the x-axis in both figures represents the task-specific cost only, i.e., it does not include the cost of computing the shared encoder. For this reason the curves do not start at zero, because we include the task-specific cost of the softmax layer of head, particularly noticeable when considering cost based on the number of parameters.

Overall, the average accuracy across tasks grows from 0.71 to 0.75 by adding task-specific adapters (+6% in relative terms). However, there are significant differences across tasks. For example, *MUSAN* starts from a higher level of accuracy in the multi-head model, and no improvement is observed when adding task-specific adapters. Conversely, *NSynthPitch* is quite different from all other tasks, and the shared encoder is unable to capture the features necessary to solve this task. As a result, a relative +15% / + 13% is observed when cost is measured in terms of number of parameters and FLOPs, respectively. For 6 out of the 8 tasks, the proposed model achieves a level of accuracy which

(a) Accuracy vs. cost.

		Num. params		FLOPs		
Task	Multi-head	$\lambda = 10^{-2}$	$\lambda = 10^{-4}$	$\lambda = 10^{-2}$	$\lambda = 10^{-4}$	Single-task
MUS	0.93	0.94 (+1.2%)	0.94 (+0.4%)	0.95 (+1.9%)	0.94 (+0.8%)	0.98
LSP	0.62	0.73 (+17.5%)	0.83 (+33.8%)	0.82 (+32.4%)	0.85 (+37.8%)	0.98
BSD	0.67	0.72 (+6.8%)	0.75 (+11.2%)	0.73 (+7.9%)	0.74 (+8.4%)	0.73
TUT	0.47	0.51 (+8.6%)	0.54 (+15.3%)	0.55 (+16.7%)	0.56 (+18.3%)	0.82
SPC	0.19	0.49 (+160%)	0.61 (+223%)	0.59 (+209%)	0.63 (+228%)	0.75
LID	0.52	0.68 (+32.0%)	0.66 (+28.1%)	0.69 (+38.4%)	0.72 (+44.0%)	0.64
NPI	0.46	0.50 (+9.2%)	0.59 (+28.6%)	0.58 (+27.2%)	0.60 (+30.8%)	0.79
NIF	0.40	0.47 (+18.4%)	0.51 (+27.9%)	0.54 (+35.6%)	0.51 (+30.2%)	0.63
Mean	0.53	0.63 (+31.6%)	0.68 (+46.0%)	0.68 (+46.2%)	0.69 (+49.8%)	0.79



(b) Number of parameters

(c) FLOPs

Figure 3: Accuracy vs. cost for the transfer learning scenario.

is in-between the multi-head baseline and independent single-task models. When comparing with the latter, one needs to bear in mind that the overall complexity of the single task models is significantly larger than the architecture evaluated in the multi-task learning scenario, also when $\lambda \to 0$ and all gates are open. To further bridge the gap, one could increase α, thus allocating additional task-specific channels, in exchange for additional complexity. For two tasks, namely *Bird Audio Detection* and *Spoken Language Identification*, the proposed multi-task architecture outperforms the corresponding single-task baselines. This can be explained by the fact that both tasks have a relatively small dataset and the single-task model is likely to overfit. Conversely, when trained jointly with other tasks, this acts as a form of regularization, thus leading to higher accuracy on the test set.

Figure 3 show similar results for the transfer learning scenario. In this case, the pre-trained encoder is frozen and, as already observed in (Tagliasacchi et al., 2019), it provides good representations only for some of the tasks. For example, a simple multi-head model achieves a level of accuracy equal to 0.19 on the *Speech Commands* task. By adding task-specific adapters, the accuracy grows above 0.60, regardless of the adopted cost measure. In general, much larger improvements are observed in this case, with an average relative increase in accuracy above 40%.

It is also interesting to observe how the model decides to allocate the additional budget available for task adapters, inspecting the status of the gates upon training convergence. Figure 4a illustrates this aspect for two tasks, namely *Bird Audio Detection* and *NSynthPitch*. Each sub-figure shows how the gates evolve by decreasing the value of λ, thus relaxing the cost constraint. First, we observe that different tasks require a different level of adaptation. In this example, *NSynthPitch* uses a larger number of additional channels. Second, the status of the gates depend on the selected cost measure. When considering FLOPs, the last fully layer is relatively inexpensive. Thus, most of the gates are kept open. Conversely, when considering number of parameters requires a more parsimonious use of the fully connected layer, as this accounts for a large fraction of the total cost.

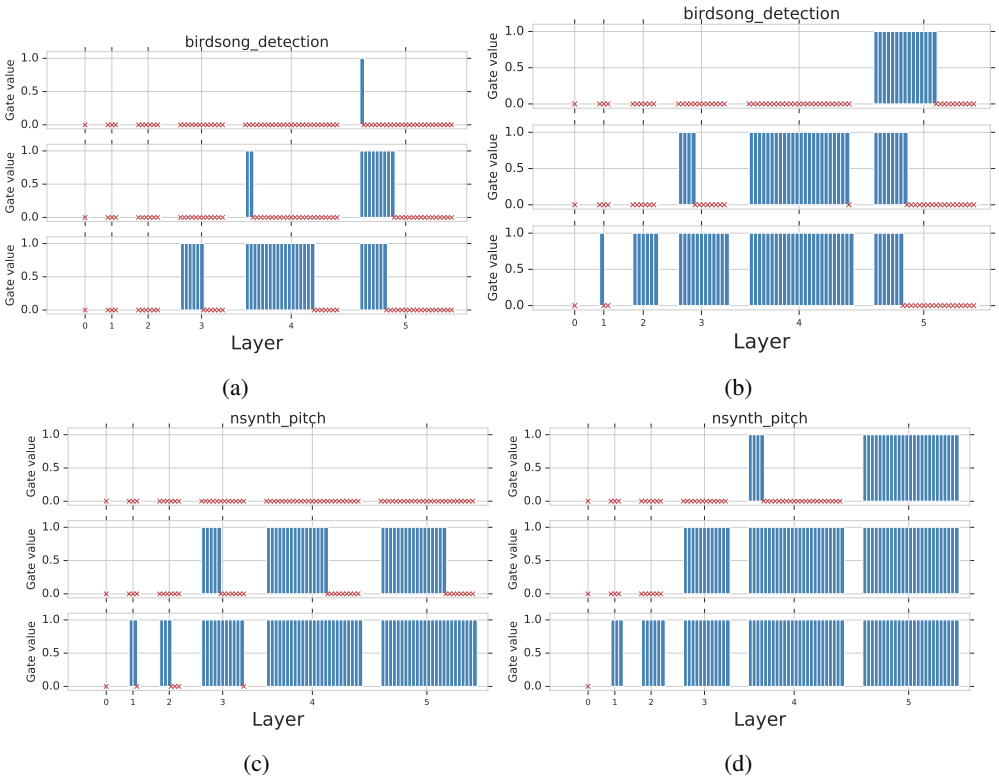


Figure 4: Gate values - Cost measure: Number of parameters (a) and (c); FLOPs (b) and (d). Crosses represent closed gates corresponding to unused channels. Within each sub-figure, the Lagrange multiplier λ increases going top-to-bottom, allowing more gates to stay open.

5 Conclusion

In this paper we propose a multi-task learning model that is able to address a wide variety of audio tasks. Unlike previously proposed approaches, our model can compute simultaneously either all tasks at once, or a subset of them, depending on the available computational resource budget. The allocation of the task-specific resources is handled jointly with training, by learning which additional channels should be used for each task and layer. Experimental results show that the proposed model outperforms a multi-head architecture baseline and approaches the accuracy achievable when using separate task-specific models. Our future research will pursue two different directions: i) on the one hand, we will explore how tasks can be group together, so as to share a common representation within each group; ii) on the other hand, we will relax the constraint on the input audio front-end, investigating how different tasks may benefit from task-specific time-to-frequency transformations.

References

Brian Cheung, Alex Terekhov, Yubei Chen, Pulkit Agrawal, and Bruno Olshausen. Superposition of many models into one. Technical report, 2019. URL `https://arxiv.org/pdf/1902.05522v1.pdf`.

Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large Scale Fine-Grained Categorization and Domain-Specific Transfer Learning. In *Computer Vision and Pattern Recognition Conference (CVPR)*, jun 2018. URL `http://arxiv.org/abs/1806.06193`.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *International Conference on Machine Learning (ICML)*, oct 2014. URL `http://arxiv.org/abs/1310.1531`.

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. Technical report, apr 2017. URL http://arxiv.org/abs/1704.01279.

Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio Set: An ontology and human-labeled dataset for audio events. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 776–780. IEEE, mar 2017. ISBN 978-1-5090-4117-6. doi: 10.1109/ICASSP.2017.7952261. URL http://ieeexplore.ieee.org/document/7952261/.

Petko Georgiev, Sourav Bhattacharya, Nicholas D Lane, and Cecilia Mascolo. Low-resource Multi-task Audio Sensing for Mobile and Embedded Devices via Shared Deep Neural Network Representations. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 39(4):39, 2017. URL https://doi.org/0000001.0000001.

Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. MorphNet: Fast & Simple Resource-Constrained Structure Learning of Deep Networks. In *Computer Vision and Pattern Recognition Conference (CVPR)*, 2018. URL https://arxiv.org/pdf/1711.06798.pdf.

Shawn Hershey, Sourish Chaudhuri, Daniel P.W. Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, Malcolm Slaney, Ron J Weiss, and Kevin Wilson. CNN architectures for large-scale audio classification. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 131–135, 2017. ISBN 9781509041176. doi: 10.1109/ICASSP.2017.7952132.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, PMLR*, pp. 2790–2799, feb 2019. URL http://arxiv.org/abs/1902.00751.

Łukasz Kaiser, Google Brain, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Google Research, Llion Jones, and Jakob Uszkoreit. One Model To Learn Them All. Technical report, 2017. URL https://github.com/tensorflow/tensor2tensor.

Tyler Lee, Ting Gong, Suchismita Padhy, Anthony Ndirango, and Andrew Rouditchenko. Label-efficient audio classification through multitask learnign and self-supervision. In *International Conference on Machine Learning (ICML)*, 2019. URL https://openreview.net/pdf?id=BkecJjCEuN.

Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Detection and Classification of Acoustic Scenes and Events. In *Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, pp. 9–13, 2018. doi: 10.5281/zenodo.1228142. URL https://doi.org/10.5281/zenodo.1228142,.

Pramod Kaushik Mudrakarta, Mark Sandler, Andrey Zhmoginov, and Andrew Howard. K For The Price Of 1: Parameter Efficient Multi-task And Transfer Learning. In *International Conference on Learning Representations (ICLR)*, oct 2019. URL http://arxiv.org/abs/1810.10703.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. LibriSpeech: An ASR Corpus Based on Public Domain Adio Books. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2015. URL http://www.gutenberg.org.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Neural Information Processing Systems (NIPS)*, may 2017. URL http://arxiv.org/abs/1705.08045.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Computer Vision and Pattern Recognition Conference (CVPR)*, mar 2018. URL http://arxiv.org/abs/1803.10082.

David Snyder, Guoguo Chen, and Daniel Povey. MUSAN: A Music, Speech, and Noise Corpus. Technical report, 2015. URL http://www.itl.nist.gov/iad/mig/tests/sre/.

Dan Stowell, | Mike Wood, | Hanna Pamuła, Yannis Stylianou, and Hervé Glotin. Automatic acoustic detection of birds through deep learning: the first Bird Audio Detection challenge. Technical report, 2018. URL https://arxiv.org/pdf/1807.05812.pdf.

Marco Tagliasacchi, Beat Gfeller, Félix de Chaumont Quitry, and Dominik Roblek. Self-supervised audio representation learning for mobile devices. Technical report, may 2019. URL http://arxiv.org/abs/1905.11796.

Tomasz. Spoken Language Identification, 2018.

Pete Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. Technical report, 2018. URL https://arxiv.org/pdf/1804.03209.pdf.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Neural Information Processing Systems (NIPS)*, nov 2014. URL http://arxiv.org/abs/1411.1792.