630 IEEE SIGNAL PROCESSING LETTERS, VOL. 27, 2020

Multi-Task Adapters for On-Device Audio Inference

Marco Tagliasacchi , Félix de Chaumont Quitry , and Dominik Roblek



***Abstract*—The deployment of deep networks on mobile devicesrequires 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 archi-tecture 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).**

***Index Terms*—Multi-task leaning, audio recognition.**

1. INTRODUCTION

HE availability of large annotated audio datasets (e.g., **T**AudioSet [1]) has enabled to train models that are ableto target a large number of audio classes. 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) [2]. 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 [3], [4], in which a shared common encoder computes

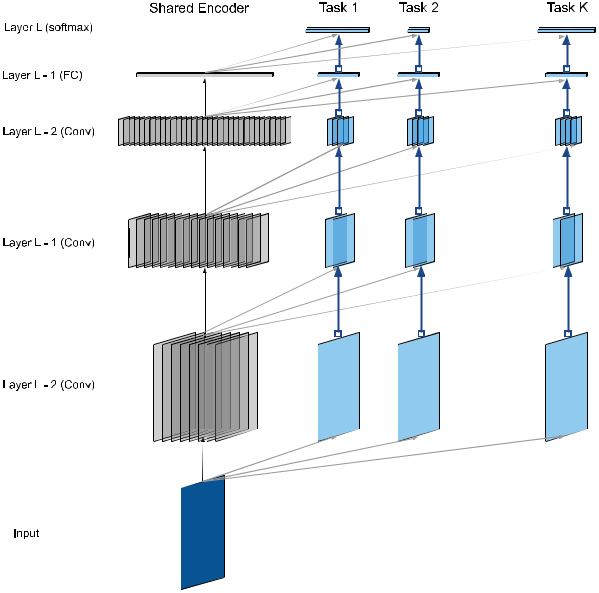
Manuscript received November 27, 2019; revised February 26, 2020; accepted April 9, 2020. Date of publication April 20, 2020; date of current version May 12, 2020. The associate editor coordinating the review of this manuscript and approving it for publication was Ville M. Hautamaki. *(Corresponding author:* *Marco Tagliasacchi.)*

The authors are with the Google Research, 8002 Zurich, Switzerland (e-mail:

[mtagliasacchi@google.com;](mailto:mtagliasacchi@google.com) [fcq@google.com;](mailto:fcq@google.com) [droblek@google.com)](mailto:droblek@google.com).

Digital Object Identifier 10.1109/LSP.2020.2988158

Fig. 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.



general-purpose audio embeddings, and task-specific fully con-nected 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 net-works in parallel to a shared encoder, as illustrated in Fig. 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. An important property of our model is that there are no inter-dependencies across tasks. As a consequence during inference one can decide to compute simultaneously either all tasks or a subset of them, depending on the available resource budget.

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

|  |  |
| --- | --- |
| TAGLIASACCHI *et al.*: MULTI-TASK ADAPTERS FOR ON-DEVICE AUDIO INFERENCE | 631 |

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 mech-anism, 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 en-coder network and learning task-specific adapters at dif-ferent 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 improve the level of accuracy of several tasks with respect to a multi-head model by only marginally increasing the cost.

1. 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 [5], [6] are common methods used to fine-tune a linear classifier on top of the em-beddings produced by a pre-trained network to address multiple tasks. An alternative approach consists of full fine-tuning [7], 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, the authors of [8] 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 net-work representation from one domain to another. This approach was later extended in [9], introducing a form of adapter that can be added in parallel to the main network architecture, and successfully applied to the NLP domain in [10]. An alternative approach is proposed in [11], in which a task-specific model patch is learned to produce different embeddings for different downstream tasks. 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.

Multi-task learning in the context of general-purpose audio has been less explored. The prevailing approach is to train a sin-gle model addressing multiple classes at once [2]. 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, [4] proposed a model architecture that addresses simultaneously three tasks. The task adaptation only occurs in the last layer of a multi-head model architecture. Similarly to our work, [3] address multi-task audio learning for deployment on embedded devices. Depending on their charac-teristics, 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 adapta-tion at different depths and in a task-specific manner.

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 [12]. However, our approach differs from multiple angles: i) a single-task learning model is discussed in [12], while we focus on multi-task learning, thus investigating how allocation is performed across tasks;

1. 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 III); iv) we evaluate the model in the context of audio tasks, while [12] is mostly concerned with vision tasks.
2. 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 Fig. 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 *fk,i*(*·*), *i* = 1*, . . . , 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*, . . . , K*, the task specific encoders. The function *fk,i*(*·*) produces as output a tensor of size *Ti* *×* *Fi* *×* *Ck,i*. Note that the number of temporal frames *Ti* and frequency bins *Fi* is the same for all values of *k*. For the task-specific encoders, we include a num-ber of task-specific channels *Ck,i* = max(1*, αiC*0*,i* ), where *C*0*,i* and *αi* are hyperparameters that determine the maximumachievable complexity of the model. Although it is possible to use a different value of *αi* at each layer, throughout the rest of this paper we assume *αi* = *α*, *i* = 1*, . . . , L*.

632 IEEE SIGNAL PROCESSING LETTERS, VOL. 27, 2020

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| In the shared encoder, *f*0*,i* receives as input only the output | | | | | | | | | | | | | | | | | | | | | | both of the shared encoder and of the task-adapter channels | |  |
| of the previous layer. However, in each task-adapter encoder, | | | | | | | | | | | | | | | | | | | | | | (i.e., *ck,i* = *Ck,i*). Conversely, when increasing *λ* the use of | |  |
| *fk,i*, *k* = 0, receives as input the concatenation of the outputs of | | | | | | | | | | | | | | | | | | | | | | additional channels is penalized, thus inducing the network to | |  |
| *f*0*,i−*1and *fk,i−*1along the channel dimension. Therefore, the | | | | | | | | | | | | | | | | | | | | | | use fewer channels. Note that | *σ*(a*k−*1*,i*)1is upper bounded |  |
| cost of computing *fk,i*, *k* = 0, can be expressed as: | | | | | | | | | | | | | | | | | | |  |  |  | by *α C*0*,i−*1 , therefore when *α* | 1, the second term in equa- |  |
|  |  |  | cost*k,i* = *ηi,k* | | | | *·* | *Ck,i* | *·* | (*C*0*,i* | | *−* | 1 + *Ck,i* | | | *−* | | 1) | (1) | | | tion (5) is dominated by the constant *C*0*,i−*1, and *Ckadapters* | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | is proportional to the l-1 norm of the gating variable vector, | |  |
| (with *C*0*,*0 = 1, and *Ck,*0 = 0, for *k* = 0). That is, the cost is | | | | | | | | | | | | | | | | | | | | | | thus promoting a sparse solution in which only a subset of the | |  |
| proportional to the number of output channels *Ck,i* multiplied | | | | | | | | | | | | | | | | | | | | | | channels are used. |  |  |
| by the number of input channels (*C*0*,i−*1 + *Ck,i−*1). The cost | | | | | | | | | | | | | | | | | | | | | |  |  |  |
| scaling factor *ηi,k* is a constant value that can be computed based | | | | | | | | | | | | | | | | | | | | | | IV. EXPERIMENTS | |  |
| on: i) the intrinsic architecture of the layer; ii) the known sizes | | | | | | | | | | | | | | | | | | | | | |  |
| **Audio front-end**: In our work we consistently use the same | |  |
| *Ti × Fi*; iii) the target cost measure, i.e., FLOPs or number of | | | | | | | | | | | | | | | | | | | | | |  |
| audio frontend, which processes input sequences sampled at | |  |
| parameters as described in [12]. | | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| The proposed method aims at learning how to scale the | | | | | | | | | | | | | | | | | | | | | | 16 kHz, with a window size of 25 ms and a hop size equal to 10 ms | |  |
| number of channels to be used in each layer of the task adapters | | | | | | | | | | | | | | | | | | | | | | to compute the short-time Fourier transform (STFT), aggregated | |  |
| encoder, i.e., to determine *ck,i* *≤* *Ck,i*, subject to a constraint | | | | | | | | | | | | | | | | | | | | | | in *F* = 64 mel-spaced frequency bins in the range 60–7800 Hz. | |  |
| Finally, we take the logarithm of the resulting spectrogram. | |  |
| on the total cost. To this end, we introduce a gating mechanism | | | | | | | | | | | | | | | | | | | | | |  |
| that controls the flow of the activations in the task adapters en- | | | | | | | | | | | | | | | | | | | | | | **Audio tasks**: We evaluate the proposed multi-task adapters | |  |
| coders. Namely, for each layer of the task adapters we introduce | | | | | | | | | | | | | | | | | | | | | | architecture addressing simultaneously 8 different audio-based | |  |
| *Ck,i* additional trainable variablesa*k,i* = [*ak,i,*1*, . . . , ak,i,Ck,i* ], | | | | | | | | | | | | | | | | | | | | | | tasks, covering both speech and non-speech related tasks. In all | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ˜ |  |  |  | cases, the model receives as input a spectrogram slice of size | |  |
| which modulate the output of each channel, that is, *fk,i,c*(*x*) = | | | | | | | | | | | | | | | | | | | | | | 96 *×* 64, so that the receptive field is equal to *T* = 975 ms. We | |  |
| *σ*(*ak,i,c*) | | *·* | *fk,i,c*(*x*), where *σ*( | | | | | | ) is a non-linear function that | | | | | | | | | | | | |  |
|  |  |  |  |  |  |  | *·* |  |  |  |  |  |  |  |  |  |  |  | + | . | use the *Speech Commands* (SPC) dataset [13] to evaluate key- | |  |
| maps its input to non-negative real numbers, i.e., R *→* R | | | | | | | | | | | | | | | | | | | |  | word spotting on 35 distinct keywords. *LibriSpeech* (LSP) [14] | |  |
| In our work, we use a clipped ReLU nonlinearity defined as | | | | | | | | | | | | | | | | | | | | | | contains audio books read by 251 different speakers. We use the | |  |
| follows: | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 100 hours training set to evaluate a speaker identification task. | |  |
|  |  |  |  | *σ*(*a*; *s*) = min(1*, ReLU* (*s · a* + 0*.*5)) | | | | | | | | | | | | | | | (2) | | |  |
|  |  |  |  | The *Spoken Language Identification* (LID) dataset [15] contains | |  |
| The slope of the non-linearity *s* is progressively increased | | | | | | | | | | | | | | | | | | | | | | samples that belong to three different languages: English, Span- | |  |
| ish and German, while the *MUSAN* (MUS) dataset [16] distin- | |  |
| during training, in such a way that, as *s* *→ ∞*, (2) acts as a gating | | | | | | | | | | | | | | | | | | | | | |  |
| guishes across three classes, namely music, speech and noise. | |  |
| function. Note that when the gating non-linearity is driven to be | | | | | | | | | | | | | | | | | | | | | | We also use two datasets released in the context of the recent | |  |
| either 0 or 1, it is *locked* at this value, as the gradients are equal | | | | | | | | | | | | | | | | | | | | | |  |
| DCASE2018 Challenge, *Bird Audio Detection* [17] (BSD), and | |  |
| to zero. Therefore, it performs a hard selection of those channels | | | | | | | | | | | | | | | | | | | | | |  |
| *TUT Urban Acoustic Scenes 2018* [18] (TUT), which contains | |  |
| that are contributing to the network output and those that can be | | | | | | | | | | | | | | | | | | | | | |  |
| labeled audio samples from 10 different urban environments. | |  |
| discarded. The number of active channels in the *i*-th layer of the | | | | | | | | | | | | | | | | | | | | | |  |
| Finally, we consider two tasks based on the NSynth dataset [19]. | |  |
| *k*-th task adapter is equal to: | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | *NSynthPitch* (NPI) contains notes played by different musical in- | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | *Ck,i* |  |  |  |  |  |  |  |  |  |  |  |  |  | struments at 128 different pitch levels, while *NSynthInstrument* | |  |
|  |  |  |  |  |  |  |  | 1*σ*(*ak,i,c*)*>*0 | | | | | | |  |  |  |  | (3) | | |  |
|  |  |  |  |  | *ck,i* = | | |  |  |  |  | (NIF) distinguishes 11 different families of musical instruments. | |  |
|  |  |  |  |  |  |  |  | *c*=1 |  |  |  |  |  |  |  |  |  |  |  |  |  | For all datasets, we consider the default train/test split, and | |  |
| During training, we jointly learn both the parameters of the | | | | | | | | | | | | | | | | | | | | | | provide results on slices extracted from the test set only. Note | |  |
| network and the gating variables. This is achieved by optimizing | | | | | | | | | | | | | | | | | | | | | | that the choice of the tasks used for the evaluation is consistent | |  |
| the following loss function: | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  | with the selected temporal granularity. | |  |
|  |  |  |  |  | *K* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **Model architecture**: For both the shared encoder and the | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | task-adapter networks, we use a convolutional neural network | |  |
|  |  |  |  |  | *L* = |  |  | *XE* | | |  | *adapters* | | | |  |  |  | (4) | | |  |
|  |  |  |  |  | *wk* | | *Lk* |  |  | + *λCk* | | |  |  |  |  |  | with *L* = 5 layers. Each convolutional layer is followed by | |  |
|  |  |  |  |  | *k*=1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | max-pooling, to reduce the time-frequency dimensions by a | |  |
| where | *XE* | | | is the cross-entropy | | | | | | | loss for the | | | |  |  | -th task, | |  | is | |  |
|  |  |  |  |  |  |
| *Lk* | |  | *k* | | *wk* | factor of two at each layer, a ReLU non-linearity and batch- | |  |
|  |  | *adapters* | | |  |  |  |  |  |  |
| an optional weighting term, and *Ck* | | | | | | | | | | | |  |  | is a penalty term that | | | | | | | | normalization. Finally, a global max-pooling layer is followed | |  |
| captures the cost of the *k*-th task adapter for a given configuration | | | | | | | | | | | | | | | | | | | | | | by a fully-connected layer. For each task, the output softmax | |  |
| of the gating variables: | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | layer receives as input the embeddings produced by the encoder, | |  |
|  |  |  |  |  | *L* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | concatenated with the embeddings produced by the task-adapter | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | network. |  |  |
| *adapters* | | |  |  | *ηi,k ·* | *σ*(a*k,i*) 1 *·* (*C*0*,i−*1 + | | | | | | | | |  |  | *σ*(a*k−*1*,i*)1)*.* | | | |  |  |  |
| *Ck* |  |  | = | |  |  |  | We consider a scenario in which the shared encoder is trained | |  |
|  |  |  |  |  | *i*=1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

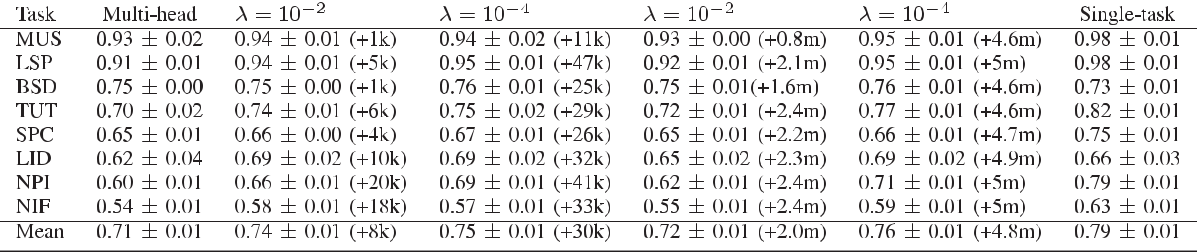
1. together with the task adapters. The number of channels in each

|  |  |
| --- | --- |
| The Lagrange multiplier *λ* controls indirectly the target cost, | layer is equal to [6*,* 12*,* 24*,* 48*,* 96], for a total of 65k parameters |
| i.e., when *λ* = 0 the optimizer minimizes the cross-entropy loss | anzd 6M FLOPs. We set *α* = 1*/*3 so that the number of task- |
| *LkXE* only, thus potentially using all the available capacity, | specific channels is equal to [2*,* 4*,* 8*,* 16*,* 32]. Hence, the total |

|  |  |
| --- | --- |
| TAGLIASACCHI *et al.*: MULTI-TASK ADAPTERS FOR ON-DEVICE AUDIO INFERENCE | 633 |

TABLE I

ACCURACY VS. COST



number of gating variables is equal to (2 + 4 + 8 + 16 + 32) *×*

1. = 496. In our experiments we set the task-specific loss weight to be uniform across tasks *wk* = 1*/K*. Learning the weights [20] is an interesting direction which is left to future work.

The loss function is minimized with stochastic gradient de-scent 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. We do not explicitly rebalance the class labels during training and we do not enforce that all labels are represented in each 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 architec-ture which consists of a shared encoder and 8 different fully connected layers, one for each task. Note that this baseline can be seen as a corner case of the proposed multi-task adapter model, which is obtained when *λ* is sufficiently large so that no task-specific parameters are used, apart from those in the output softmax layer. For this layer the number of parameters depends on the number of output classes. For example, the *LibriSpeech*

head requires additional 251 *×* 128 32*k* parameters, the *MU-SAN* head only3 *×* 128 = 384parameters. The FLOPs cost ofthis layer is negligible when compared to the rest of the network. We also report results for independent single-task models having the same model architecture, but with a single head. We also include results obtained by training independent task-specific models. 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.

**Results**: We evaluate the proposed model architecture bycomputing the classification accuracy of each of the 8 tasks. We consider cost expressed either as number of task-specific parameters, or task-specific FLOPs. Table I summarizes our main results. We let the parameter *λ* vary, so as to target different cost levels. In each case we report the task accuracy averaged across 5 replicas (*±σ*). and the additional number of task-specific parameters.

Overall, the average accuracy across tasks grows adding task-specific parameters. When using number of parameters as cost measure, the accuracy goes from 0.71 to 0.74 (+8k parameters) and 0.75 (+30k parameters). When using FLOPs, to 0.72 (+2.0m FLOPs) and 0.76 (+4.8m FLOPS). At the same time, we observe significant differences across tasks. For ex-ample, *MUSAN* starts from a higher level of accuracy in the multi-head model, and marginal improvements are 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, accuracy increases from 0.60 to 0.69 (0.71) when cost is measured in terms of number of parameters (FLOPs). For 6 out of the 8 tasks, the proposed model achieves a level of accuracy which 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 *λ* *→* 0 and all gates are open. To further bridge the gap, one could increase *α*, thus allocating additional task-specific channels, in exchangefor additional complexity. For two tasks, namely *Bird Audio* *Detection* and *Spoken Language Identification*, the proposedmulti-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.

It is also interesting to observe how the model decides to allocate the additional budget available for task adapters, in-specting the status of the gates upon training convergence. We observed that harder tasks (e.g., *NSynthPitch*) use a larger number of additional channels than simple tasks (e.g., *MUSAN*), as measured by the task-specific additional parameters in Table I. In addition, 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.

1. CONCLUSION

In this paper we propose a multi-task learning model that is able to address a wide variety of audio tasks. 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.

634 IEEE SIGNAL PROCESSING LETTERS, VOL. 27, 2020

REFERENCES

1. J. F. Gemmeke *et al.*, “Audio Set: An ontology and human-labeled dataset for audio events,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 776–780.
2. S. Hershey *et al.*, “CNN architectures for large-scale audio classification,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 131– 135.
3. P. Georgiev, S. Bhattacharya, N. D. Lane, and C. Mascolo, “Low-resource multi-task audio sensing for mobile and embedded devices via shared deep neural network representations,” in *Proc. ACM. Interactive, Mobile,* *Wearable Ubiquitous Technol.*, vol. 39, no. 4, 2017, Art. no. 50. [Online].Available: <https://doi.org/10.1145/3131895>
4. T. Lee, T. Gong, S. Padhy, A. Ndirango, and A. Rouditchenko, “Label-efficient audio classification through multitask learnign and self-supervision,” in *Proc. Int. Conf. Mach. Learn.*, 2019, *arXiv:1910.12587*. [Online]. Available: [https://openreview.net/pdf?id=BkecJjCEuN](https://openreview.net/pdf\{?\}id$=$BkecJjCEuN)
5. J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in *Proc. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328. [Online]. Available: <http://arxiv.org/abs/1411.1792>
6. J. Donahue *et al.*, “DeCAF: A deep convolutional activation feature for generic visual recognition,” in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. I-647–I-655. [Online]. Available: <http://arxiv.org/abs/1310.1531>
7. Y. Cui, Y. Song, C. Sun, A. Howard, and S. Belongie, “Large scale fine-grained categorization and domain-specific transfer learning,” in *Proc.* *Comput. Vision Pattern Recognit. Conf.*, 2018, pp. 4109–4118. [Online].Available: <http://arxiv.org/abs/1806.06193>
8. S.-A. Rebuffi, H. Bilen, and A. Vedaldi, “Learning multiple visual domains with residual adapters,” in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 506– 516. [Online]. Available: <http://arxiv.org/abs/1705.08045>
9. S.-A. Rebuffi, H. Bilen, and A. Vedaldi, “Efficient parametrization of multi-domain deep neural networks,” in *Proc. Comput. Vision Pattern* [*Recognit. Conf.*, 2018, pp. 8119–8127. [Online]. Available: http://arxiv.](http://arxiv.org/abs/1803.10082)[org/abs/1803.10082](http://arxiv.org/abs/1803.10082)
10. N. Houlsby *et al.*, “Parameter-efficient transfer learning for NLP,” in *Proc.* *36th Int. Conf. Mach. Learn.*, 2019, pp. 2790–2799.
11. P. K. Mudrakarta, M. Sandler, A. Zhmoginov, and A. Howard, “K for the price of 1: Parameter efficient multi-task and transfer learning,” in *Proc.* *Int. Conf. Learn. Representations*, Oct. 2019, *arXiv:1810.10703*. [Online].Available: <http://arxiv.org/abs/1810.10703>
12. A. Gordon *et al.*, “MorphNet: Fast & simple resource-constrained structure learning of deep networks,” in *Proc. Comput. Vision Pattern Recognit.* [*Conf.*, 2018, pp. 1586–1595. [Online]. Available: https://arxiv.org/pdf/](https://arxiv.org/pdf/1711.06798.pdf)[1711.06798.pdf](https://arxiv.org/pdf/1711.06798.pdf)
13. P. Warden, “Speech commands: A dataset for limited-vocabulary speech

[recognition,” 2018, *arXiv:1804.03209*. [Online]. Available: https://arxiv.](https://arxiv.org/pdf/1804.03209.pdf) [org/pdf/1804.03209.pdf](https://arxiv.org/pdf/1804.03209.pdf)

1. V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “LibriSpeech: An ASR corpus based on public domain adio books,” in *Proc. IEEE Int. Conf.* [*Acoust., Speech Signal Process.*, 2015. [Online]. Available: http://www.](http://www.gutenberg.org)[gutenberg.org](http://www.gutenberg.org)
2. Tomasz, “Spoken Language Identification,” 2018. [Online]. Available: <https://www.kaggle.com/toponowicz/spoken-language-identification>
3. D. Snyder, G. Chen, and D. Povey, “MUSAN: A music, speech, and noise

[corpus,” 2015, *arXiv:1510.08484*. [Online]. Available: http://www.itl.nist.](http://www.itl.nist.gov/iad/mig/tests/sre/) [gov/iad/mig/tests/sre/](http://www.itl.nist.gov/iad/mig/tests/sre/)

1. D. Stowell, M. Wood, H. Pamuła, Y. Stylianou, and H. Glotin, “Automatic acoustic detection of birds through deep learning: The first bird audio detection challenge,” *Methods Ecology Evol.*, vol. 10, no. 3, pp. 368–380, Mar. 2018.
2. A. Mesaros, T. Heittola, and T. Virtanen, “Detection and Classification of Acoustic Scenes and Events,” in *Proc. Detection Classification Acoust.* *Scenes Events Workshop*, 2018, pp. 9–13.
3. J. Engel *et al.*, “Neural audio synthesis of musical notes with WaveNet autoencoders,” in *Proc. 34th Int. Conf. Mach Learn.*, vol. 70, Aug. 2017, pp. 1068–1077.
4. R. Cipolla, Y. Gal, and A. Kendall, “Multi-task learning using uncertainty to weigh losses for scene geometry and semantics,” in *Proc. Soc. Conf.* *Comput. Vision Pattern Recognit.*, 2018, pp. 7482–7491.