Train-Once-for-All Personalization

Hong-You Chen[†][*] Yandong Li[◊] Yin Cui[◊] Mingda Zhang[◊] Wei-Lun Chao[†] Li Zhang[◊]
[†]The Ohio State University [◊]Google Research
[†]{chen.9301, chao.209}@osu.edu, [◊]{yandongli, yincui, mingdaz, zhl}@google.com

Abstract

We study the problem of how to train a "personalization-friendly" model such that given only the task descriptions, the model can be adapted to different end-users' needs, e.g.*, for accurately classifying different subsets of objects. One baseline approach is to train a "generic" model for classifying a wide range of objects, followed by class selection. In our experiments, we however found it suboptimal, perhaps because the model's weights are kept frozen without being personalized. To address this drawback, we propose **T**rain-once-for-**A**ll **PER**sonalization (*TAPER*), a framework that is trained just once and can later customize a model for different end-users given their task descriptions.* TAPER *learns a set of "basis" models and a mixer predictor, such that given the task description, the weights (not the predictions!) of the basis models can be on the fly combined into a single "personalized" model. Via extensive experiments on multiple recognition tasks, we show that* TAPER *consistently outperforms the baseline methods in achieving a higher personalized accuracy. Moreover, we show that* TAPER *can synthesize a much smaller model to achieve comparable performance to a huge generic model, making it "deployment-friendly" to resource-limited end devices. Interestingly, even without end-users' task descriptions,* TAPER *can still be specialized to the deployed context based on its past predictions, making it even more "personalization-friendly".*

1. Introduction

Recent years have witnessed multiple breakthroughs in visual recognition [10, 17, 23, 25, 36], thanks to the advance in deep learning and the accessibility to large datasets. Specifically, existing works have shown the possibility to train a gigantic and versatile "generic" model capable of classifying a wide range of over tens of thousands of objects [22, 33], rendering the promising future towards general-purposed AI.

[*]Work done as a student researcher at Google Research.

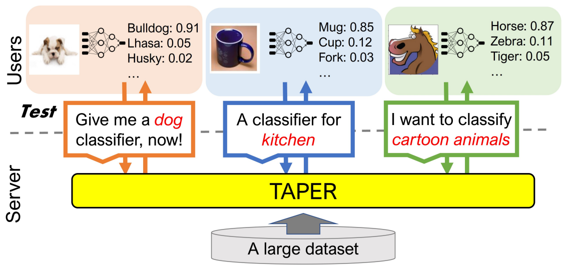


Figure 1. Examples of personalization via task description. We propose a useful formulation: train-once-for-all personalization. Our "personalization-friendly" framework TAPER can on the fly reply to each user's request with a personalized model promptly conditioned on the task description only.

However, from an end-user's perspective, we often do not need such a versatility *at once*. Instead, users more often look for models that are specialized to their requests, *e.g.*, for accurately classifying a few but frequently encountered or safety-critical objects in their environments. Taking ImageNet-1K [9] as an example, a ResNet-152 classifier [17] can achieve around 80% accuracy in recognizing each of the 1K objects, which, while exciting to the vision community, may sound terrible to a visually-impaired user who seeks to smoothly interact with a handful of everyday objects. A better solution for end-users is perhaps to construct "personalized" models dedicated to their needs, *e.g.*, train a 20-way classifier for everyday objects to attain an accuracy closer to 100%. Importantly, a personalized model usually requires a smaller capacity/size than a generic one, making it easier to deploy to resource-limited devices.

Personalization is by no means a new concept. A naïve way to achieve it is to retrain a new model upon request, using the corresponding data. Doing so, however, is hardly scalable from a service provider's point of view: the computation for training simply grows linearly with the number of users and their requests. The training latency can also degrade the user experience. Suppose the service provider has sufficient data and is capable of training a generic model, retraining may just sound superfluous: *if the objects the end-user cares about are already seen in training the generic model, why bother training on them again for personalization?* In this paper, we therefore ask:

Can we train a "personalization-friendly" model such that after deployed, it can be easily specialized and rapidly condensed based on the end-user's task description, without further training?

To begin with, we investigate a fairly simple idea, which is to train a (large) generic model, followed by *class selection* for personalization — chopping off the classes that are not of the user's interest from the classification head. While extremely straightforward without further training, this idea can already boost the aforementioned ResNet-152 to 95% accuracy on recognizing 20 classes. Nevertheless, this approach does not condense the model for computation and memory efficiency. One may resolve this problem by training a smaller generic model like ResNet-18, whose size is roughly $\frac{1}{5}$ of ResNet-152. However, with limited capacity, ResNet-18 after class selection can only attain 92% accuracy on classifying 20 classes. We hypothesize if we can somehow personalize the backbone weights as well, the model will be able to better utilize its capacity to tackle the shrunken scope of end-users' tasks.

To address these deficiencies while keeping the personalization process simple, we propose **T**rain-once-for-**A**ll **PER**sonalization (TAPER), a novel framework that is trained just once and can later head-to-toe customizes a condensed model on the fly for different end-users and requests, given their task descriptions.

At the core of TAPER is a set of shareable "basis" models inspired by [5, 12], and a "mixer" predictor. The basis models have the same neural network architecture, each of which is expected to capture a certain specialty and therefore can be smaller in size than a large generic model. The mixer predictor then takes the user's task description (*e.g.*, *"Classify bicycle, pedestrian, tree, obstacle for me."*) as input, and produces coefficients to linearly combine the weights (not predictions!) of the basis models, condensing them into a "personalized" model on the fly. As TAPER adapts to users by predicting corresponding coefficients, not by adjusting the bases, it requires no retraining and enjoys parameter efficiency (*e.g.*, for cloud services). Moreover, since the resulting personalized model is just like a basis model in size, it enjoys computation and memory efficiency during inference and is suitable for edge deployment.

We introduce a stage-wise training procedure to effectively learn the bases and the mixer predictor. We found that naïve end-to-end training for optimizing personalized accuracy often results in inferior bases that either generalize poorly or are not specialized. We thus dedicate each stage to one desired property, starting with training each basis to generically classify all classes, followed by specializing them to different but fixed portions of data. The final stage then jointly refines the bases, together with learning the mixer predictor, to synthesize classifiers for randomly sampled tasks on the fly to optimize personalized accuracy.

We validate TAPER on three visual recognition datasets, including ImageNet [9], iNaturalist [39], and DomainNet [31], each of which captures a different personalization scenario. TAPER consistently outperforms the baselines in achieving a higher personalized accuracy. For instance, on ImageNet, TAPER is able to synthesize a ResNet-18 to achieve 96% accuracy on classifying 20 classes, 4% higher than ResNet-18 with class selection. The accuracy is even higher than ResNet-152 with class selection while using $\frac{1}{5}$ of the model size. Interestingly, even without end-users' task descriptions, we show that TAPER can still be "self-specialized" to the deployed environment conditioned on its past predictions. Most importantly, none of these improvements require further training, making TAPER truly "personalization-friendly."

2. Related Work

Personalization. Unlike the standard machine learning (ML) learns a generic model to serve many users, personalization acknowledges users' characteristics and learns each a dedicated model. Its practical value is shown in many applications such as pose estimation [6], ads predictions [2], speech recognition [43], medical ML [15, 40], etc. More recently, personalization is studied in the context of federated learning, which focuses on how the users collaborate while training their own models under privacy concern [21, 29, 37]. Differently, our goal is to train a single "personalization-friendly" model. This concept is related to meta-learning [13, 18], while it mainly learns to adapt to many new tasks with few-shot data from unseen classes, not for train-once-for-all (each task still needs fine-tuning).

Conditional neural networks. Our implementation is inspired by recent architectures that dynamically adapt the networks based on the inputs [7, 41, 44]. Another approach is Mixture-of-Experts (MoE) [34, 35] that scales a model to be powerful and computational-heavy with a group of networks/layers. Given an input, MoE routes it to the related experts and combines their predictions. Our goal is to collapse into a compact model for each task. The motivations of these methods are different from ours. They specialized the network during the inference of an individual input (*e.g.*, "this image looks like an animal"), while we specialize based on the overall knowledge of the test environment a user prefers (*e.g.*, "I'm in a jungle"). We believe these different levels of personalization (inputs *vs.* tasks) are complimentary to each other for future consideration.

Another possible implementation is by a HyperNetwork [16] that learns another network to predict the high-dimensional personalized parameters directly. It remains challenging for modern deep networks due to the large output size and training difficulty [45]. Ours learns to combine several bases instead as a special case of HyperNetwork.

Ensembles and model interpolation. Combining several specialized models to serve more versatile inputs is a widely-used concept. For instance, model ensemble [1, 26, 28] combines several models' predictions for better precision or generalization. Recently, researchers found similar benefits by combining models on the weight space instead of on the outputs [20], motivated by less training cost. We extend the concept to personalize many tasks at once by predicting to combine the basis model parameters.

Other train-once-for-all tasks. Besides our train-once-for-all personalization, the idea of training once and getting several models is a practical approach in other contexts as well. For example, [4, 27] propose to train a model that later can serve on various platforms of different system resources, significantly reducing the training efforts for neural architecture searches. [19] snapshots the intermediate models in one pass of training and uses them for ensemble. [11] trains a single model that can dynamically adjust the strengths towards multiple loss functions in test time.

3. Approach

3.1. Problem definition

Define a task t as classification over a subset of classes $\mathcal{Y}_t \subset \mathcal{Y}$. The goal of personalization is to learn a predictor $f_t : \mathcal{X} \mapsto \mathcal{Y}_t$. To handle many tasks at the same time, we further assume we have the task description $\boldsymbol{d}_t$ for $\mathcal{Y}_t$, and we want to build a framework $h(\boldsymbol{d}_t)$ where given $\boldsymbol{d}_t$, it will output f_t. Generally, the task description should provide information about the classes within the task in the form of vector representation. We will leave the realizations and choices of $\boldsymbol{d}_t$ in subsection 3.5. We consider using a large-scale dataset with many classes covering $\mathcal{Y}$, to learn the personalized-friendly function $f_t = h(\boldsymbol{d}_t; \mathcal{V})$ parameterized by $\mathcal{V}$. h inferences on the task description as guidance for synthesizing a personalized model without further optimization, essentially *train-once-for-all personalization*.

Personalization in a server-user system As a motivating application of train-once-for-all personalization, the personalized model generator $h(\cdot, \mathcal{V})$ is useful for cloud service deployments in that the server learns $\mathcal{V}$ on a large-scale dataset and maintains it for serving many future users.

The users are ultimately performing the tasks on end devices such as mobile phones, laptops, drones, etc. The computation resource is often quite limited. This constrains the memory, power, and FLOPs budgets thus making it unfavorable for the users to train or inference large models on their ends. Specifically, train-once-for-all personalization enjoys the following aspects.

- **Scalability.** We propose a principle way based on a model generator to summarize a large number of tasks (in practice, possibly over millions) as a more scalable approach.
- **On-the-fly personalization.** By modeling $h(\boldsymbol{d}, \mathcal{V})$ as a translation from task descriptions to the model weight space, it allows a user to generate a personalized model without any training but just inference. This essentially bypasses the bottleneck of training cost and makes such a personalization system to be closer to a real-time API.
- **Condensed personalized models.** Our formulation provides an advantage that decouples the number of parameters of model generator $|\mathcal{V}|$ and the output personalized models. We can in theory use more parameters in $\mathcal{V}$ for a powerful generator and condense it into lightweight personalized models for final deployment.

3.2. A strong baseline: classifier selection

Given an input $\boldsymbol{x}$, we consider f as a general neural network $f(\boldsymbol{x}; \boldsymbol{\theta})$ that consists of a feature extractor parameterized by $\boldsymbol{\psi}$ with a linear classifier $\boldsymbol{w} = [\boldsymbol{w}^{(1)}, ..., \boldsymbol{w}^{(|\mathcal{Y}|)}]$ of $|\mathcal{Y}|$ vectors for output predictions over all classes in $\mathcal{Y}$. We denote by $\boldsymbol{\theta} = \{\boldsymbol{\psi}, \boldsymbol{w}\}$. Let the task specified by a user be a few-way classification task t.

One strong baseline to personalize and fulfill the aspects in subsection 3.1 is to assume a generic, non-personalized feature extractor is sufficient and build a personalized classifier $\boldsymbol{w}_t$ on top of it by selecting only the row vectors in $\boldsymbol{w}$ for the relevant classes. That is, the personalized parameters for task t are $\boldsymbol{\theta}_t = \{\boldsymbol{\psi}, \boldsymbol{w}_t\}$. As will be shown in section 4, by training a generic feature extractor along with $\boldsymbol{w}$ in a standard way followed by classifier selection to retrieve $\boldsymbol{w}_t$, it can largely outperform a non-personalized classifier. It serves as a surprisingly strong baseline for the train-once-for-all personalization.

However, we found it suboptimal since the features may also need to be personalized to focus on more dedicated relationships between the classes within a task. As we discussed in the introduction, there are two baseline solutions, to adapt and save $\boldsymbol{\psi}_t$ for every t, or not to personalize $\boldsymbol{\psi}$ but to use a larger and more powerful feature extractor. They both have obvious drawbacks — the former is not scalable in training cost for many tasks, and the latter is computationally unfavorable for end devices — contradicting the requirements of a cloud service system.

To this end, we are thus motivated in resolving such a dilemma. That is, can we have train-once-for-all personalization for the whole compact network?

3.3. Proposed TAPER: personalization with bases

Formulation: basis models. We propose TAPER to implement $\boldsymbol{\theta}_t = h(\cdot, \mathcal{V})$ for personalizing the whole network as $\boldsymbol{\theta}_t = \{\boldsymbol{\psi}_t, \boldsymbol{w}_t\}$. Inspired by multi-task learning [12], we assume the tasks share similarity (*e.g.*, superclasses, domains, styles, etc) — it is likely that we can represent each

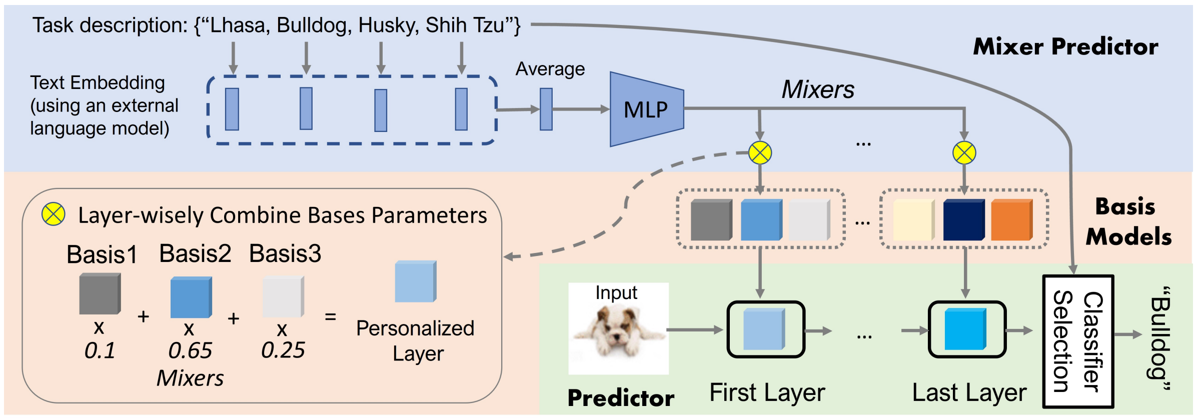

Task description: {"Lhasa, Bulldog, Husky, Shih Tzu"}
Mixer Predictor
Text Embedding (using an external language model)
Average
MLP
Mixers
Layer-wisely Combine Bases Parameters
Basis1 Basis2 Basis3
x 0.1 + x 0.65 + x 0.25 = Personalized Layer
Mixers
Basis Models
Input
Predictor
First Layer
Last Layer
Classifier Selection
"Bulldog"

Figure 2. **Overview of TAPER architecture.** A user first provides the task description (*e.g.*, a few classes of interest), which will be encoded into text embedding and fed into the *mixer predictor* to generate the *mixers*. The parameters of each layer are linear combinations of the *basis models* based on the mixers. The final outcome is a single basis personalized model, followed by *classifier selection*.

of the personalized model weight vectors $\boldsymbol{\theta}_t$ with combinations of a much smaller set of *basis* vectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_Q\}$, $|\boldsymbol{v}| = |\boldsymbol{\theta}|$. In our experiments, Q is typically small (*e.g.*, 10) compared to the number of tasks it possibly can handle (*e.g.*, for 20-way classification, there are $\binom{|\mathcal{Y}|}{20}$ combinations).

For every task, $\{\boldsymbol{v}_q\}$ are combined into a personalized model $\boldsymbol{\theta}_t$ with a combination vector $\boldsymbol{\alpha}_t$, we call it *mixers*,

$$\boldsymbol{\theta}_t(\boldsymbol{\alpha}_t, \mathcal{V}) = \sum_q \boldsymbol{\alpha}_t[q] \times \boldsymbol{v}_q, \tag{1}$$

where the mixers $\boldsymbol{\alpha}_t \in \Delta^{Q-1}$ is a Q-dimensional vector on the $(Q-1)$-simplex for convexly combining the basis models into a personalized model. Both $\boldsymbol{\alpha}$ and $\mathcal{V}$ are learned.

By adjusting only the mixers for a user, we can then quickly condense the bases into a compact personalized model for the user's future use. We note that the bases are trained to be combined layer by layer element-wisely on weights, not on the activation. This is starkly different from the mixture of experts [35] that maintains several experts and aggregates their predictions, where the model size and computation cost scale with the number of experts. In contrast, TAPER outputs a single basis model of size $|\boldsymbol{\theta}| = |\boldsymbol{v}|$ and does not scale with Q. TAPER fulfills the requirements in subsection 3.1: scalable, on-the-fly personalization, and lightweight. Unlike the baseline, it adapts the whole network, governs by the mixers over the set of bases. An overview of the architecture is provided in Figure 2. We will discuss training TAPER in subsection 3.4.

Mixer predictor. Our goal is to generate $\boldsymbol{\theta}_t = h(\boldsymbol{d}_t, \mathcal{V})$ given the task description. The task description vector is translated into the mixers by a *mixer predictor* network $\boldsymbol{\alpha}_t = g(\boldsymbol{d}_t; \boldsymbol{\phi})$, parameterized by $\boldsymbol{\phi}$, for selecting the relevant bases dedicated to the task and combining them into a condensed personalized model. We adopt a simple 4-layer multilayer perceptron (MLP) which is shared by all tasks.

Block-wise mixers. So far, we assume to use a single mixers vector $\boldsymbol{\alpha}_t$ for the whole network. A slight relaxation is to allow each component of the network to have its own mixers such that it provides more freedom for $\boldsymbol{\alpha}$ and $\mathcal{V}$ to jointly learn to combine layer-wisely. In our experiments on ResNet-18, we use one mixer vector for each of the 4 blocks, *i.e.*, now $|\boldsymbol{\alpha}_t| = 4Q$ instead of Q.

3.4. Training TAPER

Objective. Building upon Equation 1, let the loss of a task to be $\mathcal{L}_t$, we define TAPER objective function as

$$\min_{\boldsymbol{\phi}, \mathcal{V}=\{\boldsymbol{v}_q\}_{q=1}^{Q}} \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}_t(\boldsymbol{\theta}_t),$$
$$\text{where } \boldsymbol{\theta}_t = \sum_q \boldsymbol{\alpha}_t[q] \times \boldsymbol{v}_q, \quad \boldsymbol{\alpha}_t = \sigma(g(\boldsymbol{d}_t; \boldsymbol{\phi})), \tag{2}$$

where we implement $\boldsymbol{\alpha}$ to be a convex combination by a softmax function $\sigma(\cdot)$ in our experiments, as a form of regularization [12] to avoid it becoming unbounded. Both the basis models and the mixer predictor are to be learned.

Naïve approach. Equation 2 can be optimized end-to-end in standard deep learning frameworks (*e.g.*, Tensorflow) by initializing each basis with different random weights[1]. One concern is that an individual basis does not learn much about the general knowledge since each basis is likely selected by a few tasks and not trained on enough data, resulting in poor generalization. To better leverage the capacity of more bases, we provide a simple multi-stage training recipe.

Improved three-stage training. A better strategy is to first have each base be generally knowledgeable and then

[1]We note that the bases $\{\boldsymbol{v}_q\}$ cannot all be initialized with the same weights otherwise it reduces to a single basis network.

specialize them. This is inspired by the recent practice of few-shot learning [38, 42], which shows it is very important to initialize the model which will be specialized by a well-trained backbone. The training is in three stages.

- **Stage 1: single basis pre-training.** We begin with a single network $\boldsymbol{\theta}^{(0)}$ to learn the general representation of the whole dataset in a standard way, *e.g.*, with cross-entropy.
- **Stage 2: specialized basis models.** Next, we want to prepare Q specialized networks as the initialization for the Q bases. We split the dataset into Q shards based on classes or domains. For each shard, we copy $\boldsymbol{\theta}^{(0)}$ as the initialization, fine-tune it, and collect the "expert" model as $\boldsymbol{v}_q$. We note that the purpose is just to burn in each basis different domain knowledge as warm starts.
- **Stage 3: learning to mix the bases for tasks.** We jointly learn both the bases $\{\boldsymbol{v}_q\}_{q=1}^{Q}$ and the mixer predictor $g(\boldsymbol{d}; \phi)$ to combine them for all the tasks, guided by the task descriptions. Note that, we use the classifier $\boldsymbol{w}_t$ selected for each task, building upon subsection 3.2.

Despite its simplicity, we found it crucial for addressing the dilemma in the naïve approach when more bases are used. It warm-starts TAPER with well-pre-trained specialized bases thus the mixer predictor only needs to learn to "mix" them for a few epochs. This makes the developed cycles much shorter and more flexible. For instance, when the developers collect a new dataset for augmenting the existing bases, it only requires fine-tuning from $\boldsymbol{\theta}^{(0)}$, adding it as a new basis, and re-train the mixer predictor.

3.5. Task descriptions

In subsection 3.1, we assume the personalized model generator h takes a vector representation of the task and outputs the corresponding personalization model. This is realistic for some applications where (1) the users may not have training data while (2) the task that the user wants to perform can beforehand be pre-defined by the user's preference. The task descriptions not only instruct $h(\boldsymbol{d}, \mathcal{V})$ what kind of personalized model it should generate but also more importantly, for the $h(\boldsymbol{d}, \mathcal{V})$ to leverage the relationships between tasks during training.

The task description can be a flexible design choice. As an example, considering a classification task from ImageNet, a simple way is to create the bag-of-word (BoW) vector for a task, *i.e.*, a 1000-way binary vector with the bits turned on for the corresponding class indexes. The mixer $g(\boldsymbol{d}_t; \boldsymbol{\phi})$ in TAPER can gradually realize the relationships among classes during training.

Another way is to explicitly leverage the semantics of the classes by extracting the "textual class names" (*e.g.*, *"Red wolf"* or *"Buckeye"*), encode each of them into a text embedding via an external pre-trained language model, and average over classes into a vector representation $\boldsymbol{d}_t$. In our experiments, we pre-compute the 1024-dim text embedding for each class following the prompt ensemble approach in [32] and keep them frozen. Using textual embedding takes the advantage of large-scale language modeling and is more convenient as a compact fixed-dimension representation, unlike BoW depends on the class size. We provide experiments on the choice of task descriptions in subsection 4.5. Interestingly, we show in subsection 5.1 that it also allows the users to use flexible free language descriptions (*e.g.*, *"A fish of deep water having a light organ"*) instead of specifying the class name (*"flashlight fish"*).

Table 1. Summary of the datasets in experiments.

Dataset	Train/Val Size	#Class	Task
ImageNet	1.3M/50K	1K	General object recognition
iNaturalist-21	2.7M/100K	10K	Species classification
DomainNet	410K/177K	345	Object recognition with domains

4. Experiments

4.1. Settings

Datasets. To validate the effectiveness of TAPER on three large-scale visual recognition datasets, including ImageNet [9], iNaturalist (2021) [39], and DomainNet [31], each of which captures a different personalization scenario. All of them are single-label classification tasks and the resolution is 224×224. The summary is in Table 1. For each dataset, we construct the tasks as 20-way classification by sampling from the label space $\mathcal{Y}$. Each image from the training/validation set is randomly assigned with a task description as discussed in subsection 3.5 for training (sampled every epoch) and evaluation, respectively. The goal is to accurately predict the labels from the whole $\mathcal{Y}$ and the metric is the standard top-1 accuracy. More details for each dataset are provided in the corresponding subsections.

Implementation details. We use the training process similar to the standard on ImageNet [17] for all the datasets, including data pre-processing/augmentation and learning rate schedule (initial learning rate is 0.1 and decay by 0.1 every 30 epochs). We use the SGD optimizer with momentum $= 0.9$, batch size $= 128$, and weight decay $= 0.0001$. Our experiments are implemented using JAX [3]. We train on randomly-initialized **ResNet-18** networks [17] with cross-entropy by default.

For TAPER, each of the basis models uses the same architecture, and each layer is linearly combined via the mixers. The mixer predictor is a 4-layer MLP (with batchnorms and ReLU non-linearity between each layer) which maps the 1024-dim task description text embedding to the block-wise mixers, as discussed in subsection 3.3. For our proposed three-stage training in subsection 3.4, we train each stage sequentially for $100/5/20$ epochs, for the 3 stages, respectively. For a fair comparison, we, therefore, train 125 epochs for the baseline approaches (subsection 3.2). *We provide more details in the supplement materials.*

4.2. Train-once-for-all on ImageNet

We first use ImageNet to develop our method. In reality, the tasks may not be random combinations of classes but somehow have correlations depending on the use case. For instance, a user in a driving scene may ask *"Classify bicycle, pedestrian, tree, obstacle for me."* Another user may ask for a classifier for a kitchen or for different types of electronics; *e.g.*, *"coffee pot"* and *"espresso maker"* are more likely in the same task. To simulate this more realistic/meaningful scenario without losing generality, we assign each image a k-way task ($k = 20$ by default) by sampling from classes that are the nearest $2k$ synsets in the WordNet knowledge graph [30] based on its ground-truth label (which is included in the task as well[2]). We use 10 bases for ImageNet experiments. For stage 2 of TAPER training, we simply divide the dataset into 10 chunks by sharding the 1000 classes (*i.e.*, 100 classes per chunk). It is just to initialize the bases as slightly different specialists.

We then train the mixer predictor jointly with bases to personalize conditioned on the task description. The results of TAPER and the baseline approaches using different sizes of networks are in Table 2.

Personalization is valuable. Our first observations of the two baseline approaches in subsection 3.2 are: (1) increasing the network depths without personalization (ignoring the tasks) improves the accuracy but saturates at around 80%, while (2) simply post-processing a ResNet-18 with classifier selection already achieves 92.2%, *i.e.*, a +22.3% gain. This demonstrates the importance of personalization and the feasibility of train-once-for-all personalization.

TAPER outperforms the strong baseline, with much smaller networks. The baselines keep the features frozen. Our TAPER better leverages the model capacity and outputs a fully-personalized network for every task — the adapted ResNet-18 outperforms the classifier selection with a ResNet-152, using only roughly $\frac{1}{5}$ of parameters. We note that, although the baseline uses a single feature extractor, it does not have an advantage on parameter efficiency from the users' perspective since it still needs to be copied and delivered to each user's end device. TAPER's ResNet-18 outperforms the baseline counterpart by 3.6%.

Different number of classes in a task. Before we extend our study to other datasets, we first verify the effects of the number of classes in a task. TAPER takes a task vector representation as input and in theory, can handle tasks with different class sizes in one network. In Table 3, we consider training and cross-evaluate TAPER in two scenarios: the tasks are either fixed 20-way or dynamically drawn with $5 \sim 100$ ways. We observe that TAPER can handle all the cases reasonably well, where it is slightly better if training and evaluation are matched on the same scenario. For simplicity, later we will focus on the fixed 20-way scenario.

[2]Note that, we encode each class and average over classes as the task embedding thus it will not leak the ground-truths.

Table 2. Accuracy (%) on ImageNet with 20-way tasks.

Method	Network	#Parameters per task	Classifier selection	Accuracy
Baseline	ResNet-152	60.4M	✗	78.4
	ResNet-152	58.4M	✓	95.1 (+16.7)
Baseline	ResNet-101	44.7M	✗	77.6
	ResNet-101	42.7M	✓	94.8 (+17.2)
Baseline	ResNet-18	11.4M	✗	69.9
	ResNet-18	10.9M	✓	92.2 (+22.3)
TAPER	ResNet-18	10.9M	✓	**95.8**

Table 3. TAPER on ImageNet with different classes per task.

Training/Evaluation	Fixed 20-way	Dynamic $[5, 100]$-way
Baseline	92.2 ± 0.36	88.5 ± 0.78
Fixed 20-way	95.8 ± 0.45	93.6 ± 0.85
Dynamic $[5, 100]$-way	95.2 ± 0.71	95.0 ± 0.68

Table 4. Accuracy (%) on iNaturalist with 20-way tasks.

Method	Network	#Parameters per task	Classifier selection	Accuracy
Baseline	ResNet-101	63.1M	✗	84.3
	ResNet-101	42.7M	✓	97.7 (+13.4)
Baseline	ResNet-18	16.0M	✗	72.3
	ResNet-18	10.9M	✓	90.8 (+18.5)
TAPER	ResNet-18	10.9M	✓	95.9

4.3. Fine-grained species classification

Another concrete use case of personalization is fine-grained predictions in a specific topic. For instance, an entomologist might want to classify different kinds of moths. TAPER is particularly helpful for supporting scientific research in the wild that has constraints on computation resources or Internet bandwidth. We simulate such a scenario on the iNaturalist (2021) datasets that have $10,000$ species from 11 super-categories such as *"Mammals"* and *"Reptiles"*. We construct each image a 20-way task description by sampling other classes from the same super-category. We use $Q = 3 \times 11$ bases for TAPER here.

In Table 4, we again see TAPER's superiority — comparable performance and fewer parameters compared to the baseline. Notably, here we see the clear benefits of classifier selection. When the number of classes is large, cutting the classes that are not likely of the user's interest can save significant parameters and achieve higher accuracy.

Table 5. **Personalization with tasks specifying both classes and domains.** Test accuracy on DomainNet per domain is reported.

Method	Network	#Parameters per task	Classifier selection	*Real*	*Painting*	*Clipart*	*Quickdraw*	*Infograph*	*Sketch*	Avg. over domains
Baseline	ResNet-101	43.4M	✗	75.1	61.0	73.9	70.0	34.1	62.5	62.8
	ResNet-101	42.7M	✓	93.5	85.8	92.1	94.3	63.2	85.6	85.8 (+23.0)
Baseline	ResNet-18	11.1M	✗	74.2	59.4	72.1	69.8	32.0	62.1	61.6
	ResNet-18	10.9M	✓	93.3	84.8	91.1	94.0	61.4	84.5	84.9 (+23.3)
TAPER (1 basis/domain)	ResNet-18	10.9M	✓	96.0	90.9	94.6	96.7	74.0	90.6	90.5
TAPER (3 bases/domain)	ResNet-18	10.9M	✓	**96.7**	**92.4**	**95.7**	**97.5**	**77.6**	**91.9**	**92.0**

Table 6. **Ablation study for different design choices of TAPER.** The indentation with different symbols denotes adding (+) / removing (–) a component, or using a variant (○). We report the mean$\pm$std based on 3 runs on ResNet-18. *Accuracy here is averaged over examples.

Design choices	Methods / Datasets (#Bases Q)	ImageNet (10)	iNaturalist (33)	DomainNet* (18)	Avg. Accuracy
①	Standard, w/o personalization	69.9 $\pm$0.25	72.3 $\pm$0.52	65.8 $\pm$0.23	69.3
②	+ Classifier selection: a strong baseline	92.2 $\pm$0.36	90.8 $\pm$0.75	88.4 $\pm$0.54	90.5
③	TAPER w/ naïve training & classifier selection	81.8 $\pm$2.45	75.7 $\pm$3.01	78.6 $\pm$1.44	78.7
④	TAPER at Stage 1	69.8 $\pm$0.34	72.3 $\pm$0.46	65.8 $\pm$0.26	69.3
⑤	+ Stage 2 & classifier selection	91.2 $\pm$1.56	89.3 $\pm$2.45	88.5 $\pm$1.16	89.7
⑥	+ uniform weight average	86.1 $\pm$1.52	15.8 $\pm$7.55	87.2 $\pm$2.66	63.0
⑦	+ fine-tuning w/o task description	92.1 $\pm$0.56	91.0 $\pm$0.76	88.4 $\pm$0.44	90.5
⑧	+ Stage 3 (complete TAPER)	95.8 $\pm$0.45	95.9 $\pm$0.72	94.1 $\pm$0.63	95.3
⑨	○ BoW task description	94.9 $\pm$0.51	93.1 $\pm$0.81	93.5 $\pm$0.74	93.8
⑩	– Block-wise mixers	94.0 $\pm$0.24	93.1 $\pm$1.20	91.7 $\pm$0.39	92.9
⑪	– classifier selection	84.3 $\pm$0.57	81.0 $\pm$1.75	87.5 $\pm$0.64	84.3

4.4. Personalization with domain description

The task information passed to the TAPER mixer predictor can be a flexible description of the tasks of users' interest. We go beyond classes and consider if the users provide domain information related to the image styles. For instance, a user may ask: *"help me classify flowers in **paintings**"* or *"I want a **cartoon** animals classifier"*.

We investigate such a use case on the DomainNet dataset that provides 6 domains of image styles over 345 common objects. Here, we prepare the task descriptions by attaching the domain name before each of the class names, *e.g.*, *"This is a **sketch airplane**."*, encoding each class to retrieve the textual embedding, and averaging over the classes within the task. Each task is from one domain but with different 20 class combinations. We perform stage 2 training on the division of domains. The test accuracy per domain is summarized in Table 5. We see TAPER consistently outperforms the baselines for all the domains, more on the harder domains (*e.g.*, *Infograph* and *Sketch*). This echo to why TAPER improves by using 1 basis per domain (intuitively, it may depend more on domains) — the ideal features are likely domain-specific. By adding up to 3 bases per domain, TAPER can further improve since it has more freedom to personalize by considering both the domains and classes.

4.5. Remarks on design choices

Here we verify our design choices proposed in subsection 3.3 and subsection 3.4. Please refer to the indexes in Table 6. We observe:

- TAPER with naïve training (③) outperforms a non-personalized network (①) but not the classifier selection baseline (②), even ③ is attached with classifier selection already. We hypothesize the bases are not properly trained and poor in generalization.
- As sanity checks, TAPER's stage 1 (④) is basically ① but trained less (*i.e.*, training more cannot improve). Stage 2 (⑤) is slightly worse than ② as expected since the models are specialized on a shard of the dataset. Simply averaging them on weights (⑥) will not become a stronger model but fine-tuning it (⑦) can recover it to ②.
- From ⑧ to ⑪, TAPER leverages task descriptions to personalize the features by the mixer predictor thus outperforming the baseline (②). Text embedding is better for task descriptions compared to BoW vectors (⑨). It is preferred to have mixers block-wise (⑩).
- Removing classifier selection from TAPER (⑪) has a big impact. However, comparing ⑪ to ① and ④, we validate that TAPER indeed learns personalized features.
- Complete TAPER (⑧) performs the best consistently.

Table 7. Free languages descriptions of classes.

Users' free languages	*"Did you mean this?"*
WordNet Definitions	Class names
"a drinking glass with a base and stem"	*"goblet"*
"live associated with sea anemones"	*"anemones fish"*
"a tall elegant chest of drawers"	*"chiffonier"*

5. Applications, Discussions, and Conclusion

We consider practical use cases and discussions. We provide more studies and evaluations in the supplementary materials, including the effects of the number of bases/tasks.

5.1. Class descriptions in free languages

So far we use a pre-trained language encoder to embed the class names via prompts. Since the language encoder can handle general textual descriptions, this allows the users to enjoy more flexibility in their descriptions. To demonstrate such an advantage, we still train TAPER with class name descriptions, but in evaluation, we replace them with free languages that do not describe the class names explicitly for encoding, by using the definitions in the WordNet dictionary. See examples in Table 7. Perhaps surprisingly, TAPER is robust to such replacement. In ImageNet experiments, it achieves 94.2% accuracy, slightly dropped from 95.8% in Table 2. We also compared the mixers predicted from class names and free languages for each class — we see a high 0.92 cosine similarity; they select similar bases.

5.2. Self-improvement without task descriptions

So far, we have assumed that the task description is provided for personalization. We show TAPER can provide some training-free personalization even without a description but given the unlabeled test data of the task. This is useful in some scenarios such as a smart surveillance camera keeps collecting images from the same environment and wants to refine its classifier for future predictions. Concretely, assuming we have trained the TAPER model,

1. Begin with a standard, non-personalized classifier (*e.g.*, the stage 1 model) to predict a batch of test data.
2. Extract the top most common pseudo labels and use them to construct the task description.
3. Use the mixer predictor to combine a personalized model and repeat from (2) over time.

We demonstrate with a case of a 20-way task sampled from ImageNet in Figure 3: it can gradually estimate the task and improve along with seeing more test data.

5.3. Analysis

Visualization. To understand if the bases and the mixers are learned to tailor different tasks, we visualize their mixers α_t and pairwise cosine similarity of the parameters of

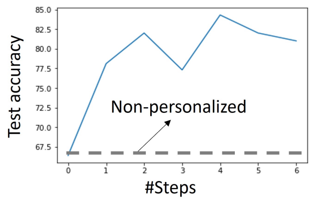
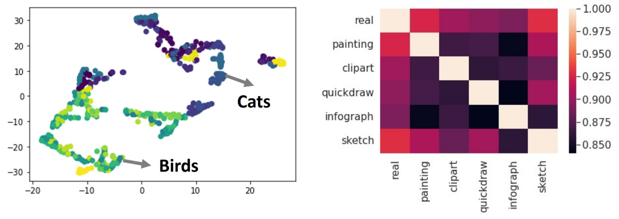


Figure 3. **Self-improvement without task descriptions.** In each step, we predict the top common classes in the test batch, retrieve the task embedding, and re-generate the personalized model.



(a) t-SNE of mixers (b) Bases cosine similarity

Figure 4. **Visualization.** (a) Predicted mixers of tasks contain *"Bird"* and *"Cat"* in ImageNet (each task colored by the sum of sorted class IDs). (b) Similarity matrix of the basis parameters learned on DomainNet. One basis for each domain.

bases $\mathcal{V}$ in Figure 4. We see different tasks indeed leverage different specialized bases.

Limitations: the price of personalization. Given the text embedding of the class names, can TAPER extend to classes not in training? To construct the classifier for unseen classes, we follow the zero-shot learning literature [14] to learn an extra mapping from the image feature space to the text embedding space on seen classes in ImageNet-1K, and evaluate unseen classes from ImageNet-21K. We observe it can hardly have such free lunch — using 10 bases is worse than using one. We hypothesize two reasons: (1) plainly fitting seen classes (better) inevitably degrades unseen performance, consistent with previous studies [8, 24]. (2) The relationships between text and vision may not be learned yet in training or have changed significantly in new classes. For instance, it might learn both *"Crown"* and *"Daisy"* in training, but *"Crown daisy"* is visually different from them. This will be our future study. Practically, developers might consider expanding the training dataset, using text with detailed visual descriptions, or augmenting TAPER with advanced optimization that promotes generalization.

5.4. Conclusion

We propose a new framework, train-once-for-all personalization, named TAPER, that is trained just once and can support many end-users given their task descriptions only. TAPER is simple and general. Personalization with TAPER is scalable, training-free, compact, and effective on various applications we evaluated.

References

[1] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. *arXiv preprint arXiv:2002.06470*, 2020. 3

[2] Mikhail Bilenko and Matthew Richardson. Predictive client-side profiles for personalized advertising. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 413–421, 2011. 2

[3] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. 5, 12

[4] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. 3

[5] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. In *CVPR*, 2016. 2

[6] James Charles, Tomas Pfister, Derek Magee, David Hogg, and Andrew Zisserman. Personalizing human video pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3063–3072, 2016. 2

[7] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[8] Yinbo Chen, Zhuang Liu, Huijuan Xu, Trevor Darrell, and Xiaolong Wang. Meta-baseline: Exploring simple meta-learning for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9062–9071, 2021. 8

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1, 2, 5

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 1

[11] Alexey Dosovitskiy and Josip Djolonga. You only train once: Loss-conditional training of deep networks. In *International Conference on Learning Representations*, 2020. 3

[12] An Evgeniou and Massimiliano Pontil. Multi-task feature learning. In *NeurIPS*, 2007. 2, 3, 4

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017. 2

[14] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. *Advances in neural information processing systems*, 26, 2013. 8

[15] Jeremy Goecks, Vahid Jalili, Laura M Heiser, and Joe W Gray. How machine learning will transform biomedicine. *Cell*, 181(1):92–101, 2020. 2

[16] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 5, 11, 12

[18] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021. 2

[19] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free. In *International Conference on Learning Representations*, 2017. 3

[20] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018. 3

[21] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019. 2

[22] Armand Joulin, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache. Learning visual features from large weakly supervised data. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 67–84, Cham, 2016. Springer International Publishing. 1

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. 1

[24] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022. 8

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1

[26] Ekaterina Lobacheva, Nadezhda Chirkova, Maxim Kodryan, and Dmitry P Vetrov. On power laws in deep ensembles. *Advances In Neural Information Processing Systems*, 33:2375–2385, 2020. 3

[27] Wei Lou, Lei Xun, Amin Sabet, Jia Bi, Jonathon Hare, and Geoff V Merrett. Dynamic-ofa: Runtime dnn architecture switching for performance scaling on heterogeneous embedded platforms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3110–3118, 2021. 3

[28] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[29] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020. 2

[30] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 6, 11

[31] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019. 2, 5

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 5, 11

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. 1

[34] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021. 2

[35] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. 2, 4

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 1

[37] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017. 2

[38] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision*, pages 266–282. Springer, 2020. 5

[39] Grant Van Horn, Elijah Cole, Sara Beery, Kimberly Wilber, Serge Belongie, and Oisin Mac Aodha. Benchmarking representation learning for natural world image collections. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12884–12893, 2021. 2, 5

[40] Jeremy C Weiss, Sriraam Natarajan, Peggy L Peissig, Catherine A McCarty, and David Page. Machine learning for personalized medicine: predicting primary myocardial infarction from electronic health records. *Ai Magazine*, 33(4):33–33, 2012. 2

[41] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[42] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8808–8817, 2020. 5

[43] Dong Yu and Jinyu Li. Recent progresses in deep learning based acoustic models. *IEEE/CAA Journal of automatica sinica*, 4(3):396–409, 2017. 2

[44] Mingda Zhang, Chun-Te Chu, Andrey Zhmoginov, Andrew Howard, Brendan Jou, Yukun Zhu, Li Zhang, Rebecca Hwa, and Adriana Kovashka. Basisnet: Two-stage model synthesis for efficient inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3081–3090, 2021. 2

[45] Andrey Zhmoginov, Mark Sandler, and Maksym Vladymyrov. Hypertransformer: Model generation for supervised and semi-supervised few-shot learning. In *International Conference on Machine Learning*, pages 27075–27098. PMLR, 2022. 2