WHAT DATA IS USEFUL FOR MY DATA: TRANSFER LEARNING WITH A MIXTURE OF SELF-SUPERVISED EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Transfer learning has proven to be a successful way to train high performing deep learning models in various applications for which little labeled data is available. In transfer learning, one pre-trains a model on a large dataset such as Imagenet, and fine-tunes its weights on the target domain. In our work, we claim that in the new era of ever increasing number of massive datasets, *selecting the relevant pre-training data* is a critical issue. We introduce a new problem in which available datasets are stored in one centralized location called a *dataserver*. We assume that a *client*, a target application with its own small labeled dataset, is only interested in fetching a subset of the server's data that is most relevant to its own target domain. We propose a novel method that aims to optimally select subsets of data from the dataserver given a particular target client. We perform data selection by employing a mixture of experts model in a series of *dataserver-client* transactions with a small computational cost. We show the effectiveness of our work in various transfer learning scenarios, demonstrating state-of-the-art performance on several target datasets and tasks such as image classification, object detection and instance segmentation. We will make our framework available as a web-service, serving data to users aiming to improve performance in their A.I. application.

1 INTRODUCTION

In the recent years, we have seen an explosive growth in both, the number and variety of A.I. applications. These range from generic image classification tasks, to surveillance, sports analytics, clothing recommendation, early disease detection, and mapping, among others. Yet, we are only in the beginning of our exploration of what is possible to achieve with Deep Learning.

One of the critical components of the new age A.I. applications is the need for labeled data. To achieve high-end performance, typically a massive amount of data needs to be used to train deep learning models. One way to mitigate the need for large-scale data annotation for each target application is via transfer learning in which a neural network is pre-trained (Chen et al., 2016; He et al., 2017; Shelhamer et al., 2017) on existing large-scale datasets and then fine-tuned on the target downstream task. While transfer learning is a well studied concept that has been proven successful in many domains (Chen et al., 2016; He et al., 2017; Shelhamer et al., 2017), deciding which data to pre-train the model on is an open research question that has received surprisingly little attention in the literature. We argue that this, however, is a crucial problem to be answered in light of the ever increasing scale of the available data.

A website of curated computer vision benchmarks [1] currently lists 367 public datasets, ranging from generic imagery, faces, fashion photos, to autonomous driving data. The sizes of datasets have also massively increased: the recently released OpenImages (Kuznetsova et al., 2018) contains 9M labeled images (600GB in size), and is 20 times larger compared to its predecessor MS-COCO (Lin et al., 2014) (330K images, 30GB). The video benchmark YouTube8m (Abu-El-Haija et al., 2016) (1.9B frames, 1.5TB), is 800 times larger compared to Davis (Caelles et al., 2018) (10k frames, 1.8GB), while the autonomous driving dataset nuScenes (Caesar et al., 2019) contains 100$\times$ the number of images than KITTI (Geiger et al., 2012).

[1] `https://www.visualdata.io/`

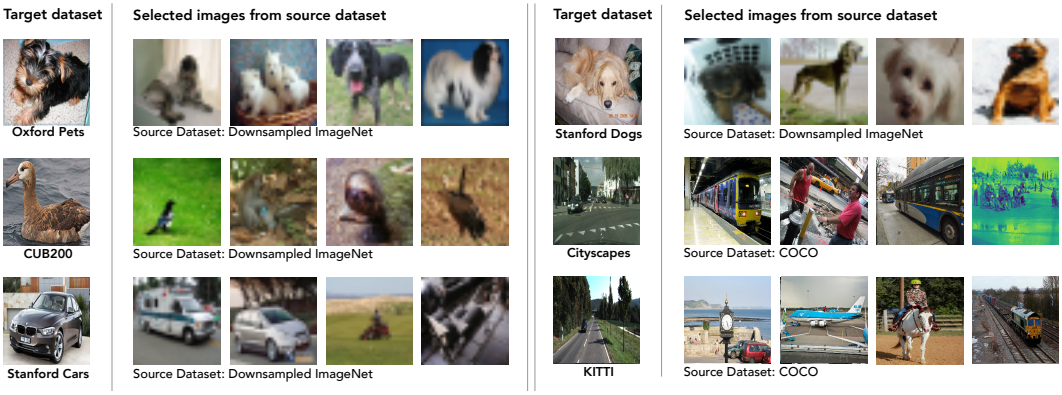

Figure 1: Different clients(target) and datasets in the dataserver (source). Images are randomly chosen from $\mathcal{S}_*$

It is evident that even downloading and storing all these datasets locally may not be affordable for everyone, let alone pre-training a model on this massive amount of data. Furthermore, for commercial applications data licensing may be a financial issue to consider. Recent works (He et al., 2018; Ngiam et al., 2018) have also shown that there is not a "the more the better" relationship between the amount of pre-training data and the downstream task performance. Instead, they demonstrated that selecting an appropriate subset of the pre-training data was important to achieve good performance on the target dataset.

In this paper, we envision a new scenario in which all (public) datasets are stored in one centralized location, i.e., a *dataserver*, and made available for download per request by a *client*. A *client* can be anyone with its own A.I. application in mind, and has a small set of its own labeled target data. We assume that each client is only interested in downloading a subset of the server's data that is most relevant to its own target domain, limited to a pre-defined budget (maximum allowed size). We further want the transaction between the *dataserver* and the client to be both, extremely efficient computationally, as well as privacy-preserving. That is, the client's data should not be visible to the server, whereas the server aims to minimize the amount of computation per client, as it may serve possibly many clients in parallel.

We propose a novel method that aims to optimally select subsets of data from a large *dataserver* given a particular target *client*, in the context described above. In particular, we represent the server's data with a mixture of experts model trained with a simple self-supervised task. This allows us to distill all of the server's data, even when it consists of several datasets featuring different types of labels, into the weights of a small number of experts. These experts are then used on the client's side to determine the most important subset of the data that the server is to provide to the client. We show significant improvements in performance on all downstream tasks compared to pre-training on a randomly selected subset of the same size. Furthermore, we show that with only 20% or 40% of pre-training data, our method achieves comparable or better performance that pre-training on the entire server's dataset.

We implement our framework as a web platform, i.e., a *dataserver* that links to a variety of large datasets, and enables each client to only download the relevant subset of data. Our platform will be made available online upon acceptance.

2 Related Work

Transfer Learning. The success of deep learning and the difficulty of collecting large scale datasets has recently brought significant attention to the long existing history of transfer learning, cross-domain annotation and domain adaptation Pan & Yang (2009); Csurka (2017); Acuna et al. (2018); Sun et al. (2017); Acuna et al. (2019); Tremblay et al. (2018). Specifically in the context of neural networks, fine-tuning a pre-trained model in a new dataset is the most common strategy for knowledge transfer. Several works (Sun et al., 2017; Mahajan et al., 2018; Caron et al., 2019) have examined the idea of pre-training in an "enormous data" scenario. That is, pre-training on datasets that are $300\times$ (JFT (Sun et al., 2017)), and $3000\times$ (Instagram (Mahajan et al., 2018)) larger than the frequently used ImageNet Deng et al. (2009). Other work has tried to understand transfer learning in neural networks. In particular, Yosinski et al. studied factors affecting the transferability of representations learned on ConvNets with respect to network architectures, network layers, and training tasks. Zamir et al. examines the relationship between visual tasks and proposes a computational method for modelling the transferability between these. Cui et al. and Ngiam et al., on the other

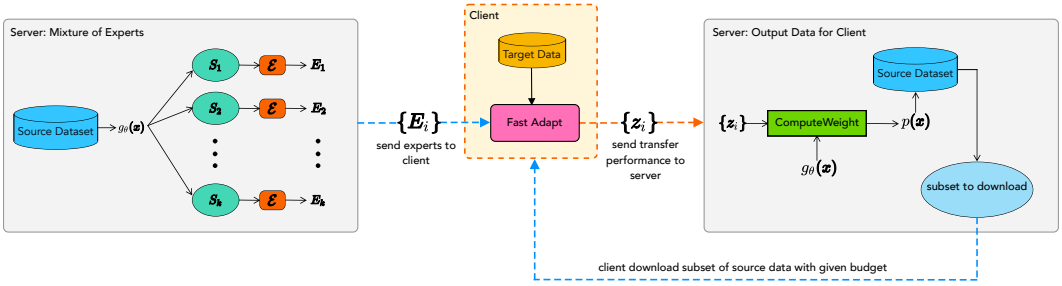

Figure 2: Overview of our method.

hand, study how the choice of pre-training data impacts performance on fine-grained classification tasks. Specifically, they show that pre-training on only relevant examples is important to achieve good performance. Our work builds on top of these observations but presents a scalable and efficient way to select the most useful subset of data in a distributed scenario where the transactions between a datacenter and a client should be both, computationally efficient and privacy-preserving. Furthermore, unlike most previous works that focus on classification, our approach can be used in a variety of tasks.

Federated Learning. (McMahan et al., 2016; Bonawitz et al., 2017) introduced a distributed ML approach with the goal of training a centralized model on decentralized data over a large number of client devices, (i.e mobile phones). Our work shares the similar idea of restricting the visibility of data in a client-server model. However, in our case the data is centralized in a server and the clients exploit the transfer learning scenario.

3 Our Approach

We define a new problem in which a *dataserver*, i.e., a centralized database that has access to a massive source dataset, aims to provide relevant subset of data to a *client* that wants to improve the performance of its model on a downstream task by pre-training the model on this subset. The *dataserver*'s dataset may or may not be completely labeled, and the types of labels (e.g., masks for segmentation, boxes for detection, or scene attributes) across data points may vary. The *client*'s dataset is considered to only have a small set of labeled examples, where further the task (and thus the type of its labels) may or may not be the same as any of the tasks defined on the *dataserver*' dataset(s). The main challenge is posed in requiring the *dataserver-client* transactions to be scalable (on the server side) with respect to the number of clients, and affordable for the resource-limited client (e.g., cannot pre-train on a massive dataset), as well as privacy preserving (client's data cannot be shared with the server, i.e., mimicking the case where the client has sensitive data such as hospital records). Only the most relevant data points should be transmitted from the server to the client.

In our approach, we represent the *dataserver*'s data using a mixture of experts learned (only once) on a self-supervised task. This naturally partitions the datasets into K different subsets of data and produces specialized neural networks whose weights encode the representation of each of those subsets. These experts are cached on the server and shared with each client, and used as a proxy to determine the importance of data points for the client's task. In particular, the experts are downloaded by the client and fast-adapted on the client's dataset. We experimentally validate that the accuracy of each adapted expert indicates the usefulness of the data partition used to train the expert on the *dataserver*. The server then uses these accuracies to construct the final subset of its data that is relevant for the client. In Figure 2, we present an illustration of our framework, and summarize the method in Algorithm 2.

In Section 3.1, we formalize our problem. In Section 3.2, we describe how we can obtain expert models through mixture of experts and analyze the different choices of representation learning algorithms for the experts (server side). In Section 3.3.1, we propose how to exploit the experts' performance on the *client*'s task for data selection.

3.1 Problem and Task Definition

Let $\mathbb{X}$ denote the input space (images in this paper), and $\mathbb{Y}_a$ a set of labels for a given task a. Generally, we will assume that multiple tasks, each associated with a different set of labels, are available, and denote this by $\mathbb{Y}$. Consider also two different distributions over $\mathbb{X} \times \mathbb{Y}$, called the source domain $\mathcal{D}_s$ and target domain $\mathcal{D}_t$. Let $\mathcal{S}$ (server) and $\mathcal{T}$ (client) be two sample sets drawn i.i.d from $\mathcal{D}_s$ and $\mathcal{D}_t$, respectively. We assume that $|\mathcal{S}| \gg |\mathcal{T}|$. Our problem then relies on finding

Algorithm 1 Server modules

Initialize representation learning algorithm $\mathcal{E}$, number of experts K
$g_\theta \leftarrow$ HARDGATING($\mathcal{S}, K$) ▷ Section 3.2: partition $\mathcal{S}$ into local subsets to obtain gating

procedure MOE($\mathcal{S}, \mathcal{E}, K$):
    For $i = 1, ..., K$
        Run $\mathcal{E}$ on $\{x \in \mathcal{S} | g_\theta(x)_i = 1\}$ to obtain expert e_{θ_i}
    return $\{e_{\theta_i}\}$

procedure OUTPUTDATA($\mathcal{S}, \boldsymbol{z}$):
    $\boldsymbol{w} \leftarrow Softmax(Normalize(\boldsymbol{z}))$
    $\boldsymbol{p}(\boldsymbol{x}) = \sum_{i=1}^{K} \boldsymbol{w}_i \, \boldsymbol{g}_{i\theta}(\boldsymbol{x}) \frac{1}{|S_i|}$
    Sample $\mathcal{S}_*$ from $\mathcal{S}$ at rate according to $\boldsymbol{p}$
    return $\mathcal{S}_*$

Algorithm 2 Overview of our framework.

Input: $\mathcal{S}, \mathcal{T}$
    $\{e_{\theta_i}\} \leftarrow$ MOE($\mathcal{D}_\mathcal{S}, \mathcal{E}, K$)
    $\boldsymbol{z} \leftarrow$ FASTADAPT($\mathcal{T}, \{e_{\theta_i}\}$)
    $\mathcal{S}_* \leftarrow$ OUTPUTDATA($\mathcal{S}, \boldsymbol{z}, b$)
    return $\mathcal{S}_*$
Output: $\mathcal{S}_* \in \mathcal{S}$ to download

Algorithm 3 Client module

procedure FASTADAPT($\mathcal{D}_\mathcal{T}, \{e_{\theta_i}\}$):
    Initialize logits $\boldsymbol{z} \in \mathbb{R}^K$
    For $i = 1, ..., K$
        $\boldsymbol{z}_i \leftarrow$ PERFORMANCE($e_{\theta_i}, \mathcal{T}$) ▷ Section 3.3.1: Evaluate transfer performance of E_i on $\mathcal{T}$
    return $\boldsymbol{z}$

the subset $\mathcal{S}_* \subset \mathcal{P}(S)$, where $\mathcal{P}(S)$ is the power set of $\mathcal{S}$, such that $\mathcal{S}_* \cup \mathcal{T}$ minimizes the risk of a model h on the target domain:

$$\mathcal{S}_* = \underset{\hat{\mathcal{S}} \subset \mathcal{P}(S)}{\arg\min} \; \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}) \sim \mathcal{D}_t}[L(h_{\hat{\mathcal{S}} \cup \mathcal{T}}(\boldsymbol{x}), \boldsymbol{y})] \tag{1}$$

Here, $h_{\hat{\mathcal{S}} \cup \mathcal{T}}$ indicates that h is trained on the union of data $\hat{\mathcal{S}}$ and $\mathcal{T}$. Intuitively, we are trying to find the subset of data from $\mathcal{S}$ that helps to improve the performance of the model on the target dataset. However, what makes our problem particularly challenging and unique is that we are restricting the visibility of the data between the *dataserver* and the *client*. This means that fetching the whole sample set $\mathcal{S}$ is prohibitive for the client, as it is uploading its own dataset to the server. We tackle this problem by representing the *dataserver*'s dataset with a set of classifiers that are agnostic of the client (Sec. 3.2)., and use these to optimize equation 1 on the client's side (Sec. 3.3.1).

3.2 DATASERVER

We here introduce our representation of the *dataserver*. This representation is computed once and stored on the server.

3.2.1 DATASET REPRESENTATION WITH A MIXTURE OF EXPERTS

We choose to represent the *dataserver*'s data $\mathcal{S}$ using the mixture of experts model (Jacobs et al., 1991). In this model, one makes a prediction as:

$$y(\boldsymbol{x}) = \sum_{i=1}^{K} g_\theta(\boldsymbol{x}) e_{\theta_i}(\boldsymbol{x}) \tag{2}$$

Here, $\boldsymbol{g}_\theta$ denotes a gating function, $\boldsymbol{e}_{\theta_i}$ denotes the i-th expert model given an input $\boldsymbol{x}$, θ are learnable weights of the model, and K corresponds to the number of experts. One can think of the gating function as softly assigning data points to each of the experts, which try to make the best guess on their assigned data points. In our work, we propose to choose the data relevant to the *client* by 1) estimating the relevance of each expert on the *client*'s dataset, and 2) use the gating function as a means to measure relevance of the original data points. We explain this in more detail in Sec 3.3.1. In this section, we focus our description on how we train the experts.

Learning the mixture of experts model is done by defining an objective $\mathcal{L}$ and using maximum-likelihood estimation (MLE):

$$\theta = \underset{\theta}{\arg\min} \, \mathbb{E}_{(\boldsymbol{x},\hat{\boldsymbol{y}}) \sim \mathcal{S}}[\mathcal{L}(\boldsymbol{y}(\boldsymbol{x}), \hat{\boldsymbol{y}})] \tag{3}$$

We discuss the choices for the objective $\mathcal{L}$ in Sec 3.2.2, dealing with the fact that the labels across the source datasets may be defined for different tasks.

While this objective can be trained end-to-end, the computational cost of doing so on a massive dataset is extremely high, particularly when K is relatively large (we need to backpropagate gradients to every expert on every training example). A straightforward way to alleviate this issue is to associate each expert with a local cluster defined by a hard gating, as used in (Hinton et al., 2015; Gross et al., 2017). In practice, we define a gating function g that partitions the dataset into mutually exclusive subsets, and train one expert per subset. This makes training easy to parallelize as each expert is trained independently on its own subset of data.

In our experiments, we use two simple partitioning schemes to determine the gating: (1) superclass partition, and (2) unsupervised partition. For superclass partition, we represent each class c in the source dataset as the mean of the image features f_c for category c, and perform k-means clustering over $\{f_c\}$. This gives a partitioning where each cluster is a superclass containing a subset of similar categories. For unsupervised partitioning, we partition the source dataset using k-means clustering on the feature space of a pretrained neural network (i.e. features extracted from the penultimate layer of a network pre-trained on ImageNet).

3.2.2 Training the Experts

We discuss two different scenarios to train the experts. In the simplified scenario, the tasks defined for both the server's and client's datasets are the same, e.g., classification. In this case we simply train a classifier for each subset of the data in $\mathcal{S}$. We next discuss the more challenging case where the tasks are different.

Ideally, we would like to learn a representation that can generalize to a variety of downstream tasks and can therefore be used in a task agnostic fashion. To this end, we use a self-supervised method on a pretext task to train the mixtures of experts. In self-supervision one leverages a simple surrogate task that can be used to learn a meaningful representation. Furthermore, it does not require any manually labeled data to train the experts which means that the *dataserver*'s dataset may or may not be labeled beforehand. This is useful if the client desires to obtain raw data and label the relevant subset on its own.

To be specific, we select image rotation as a pseudo-task for self-supervision. In particular, we follow (Gidaris et al., 2018) which demonstrated to be a simple yet powerful proxy for representation learning. Formally, given an image $\boldsymbol{x}$, we define its corresponding label $\hat{\boldsymbol{y}}$ by performing a set of geometric transformations $\{\boldsymbol{r}(\cdot, j)\}_{j=0}^{3}$ on $\boldsymbol{x}$, where r is an image rotation operator, and j defines a particular rotation by one of the following predefined degrees $\{0, 90, 180, 270\}$. We then minimize the following learning objective for the mixture of experts:

$$\mathcal{L}(\boldsymbol{x}) = \frac{1}{4} \sum_{j=0}^{3} \log \boldsymbol{y}_j(\boldsymbol{r}(\boldsymbol{x}, j)) \tag{4}$$

3.3 Server-Client Transaction

In this section, we describe the transaction between the server and client that determines the relevant subset of the server's data. The client first downloads the experts and uses these experts to measure their performance on the client's dataset. Since there is likely a domain gap between the source and the target datasets, we perform a quick adaptation of the experts on the client's side (Sec 3.3.1). The performance of each expert is sent back to the server, which uses this information as a proxy to determine which data points are relevant to the client (Sec. 3.3.2). We describe these steps in more detail in the following subsections.

3.3.1 Fast Adapt to a Target Dataset (on Client)

Single Task on Server and Client: We first discuss the case where the dataset task is the same for both the client and the server, e.g., classification. While the task may be the same, the label set may not be (classes may differ across domains). An intuitive way to adapt the experts would be to remove their classification head that was trained on the server, and learn a small decoder network on top of the experts's penultimate representations on the client's dataset, as in (Zamir et al., 2018). For classification tasks, we learn a simple linear layer on top of each pre-trained expert's representation for a few epochs. We then evaluate the target's task performance on a held-out validation set using the adapted experts. We denote the accuracy for each expert i as z_i.

Diverse Tasks on Server and Client: To generalize to unseen tasks and be further able to handle cases where the labels are not available on the client's side, we propose to evaluate the performance of the common self-supervised task used to train the experts on the server's data. Intuitively, if the expert performs well in the self-supervised task on the target dataset, then the data it was trained on is likely relevant for the client. Specifically, we use the self-supervised experts trained to learn image rotation, and evaluate the proxy task performance of predicting image rotation angles on the target images:

$$z_i = \frac{1}{|\mathcal{T}|} \sum_{x \in \mathcal{T}} \left[\arg\max_j \{e_{\theta_i}(\boldsymbol{r}(\boldsymbol{x}, j))\}_{j=0}^{3} = j \right] \tag{5}$$

Note that in this case we do not adapt the experts on the target dataset (we only perform inference).

3.3.2 Data Selection (on Server)

We now aim to assign a weighting to each of the data points in the source domain $\mathcal{S}$ to reflect how well the source data contributed to the transfer learning performance. The accuracies $\boldsymbol{z}_i$ from the client's FastAdapt step for each expert are normalized to $[0, 1]$ and fed into a *softmax* function with temperature $T = 0.1$. These are then used as importance weights $\boldsymbol{w}_i$ for estimating how relevant is the representation learned by a particular expert for the target task's performance. We leverage this information to weigh the individual data points $\boldsymbol{x}$. More specifically, each source data $\boldsymbol{x}$ is assigned a probabilistic weighting:

$$\boldsymbol{p}(\boldsymbol{x}) = \sum_{i=1}^{K} \boldsymbol{w}_i \, \boldsymbol{g}_{i_\theta}(\boldsymbol{x}) \frac{1}{|S_i|} \tag{6}$$

Here, $|S_i|$ represents the size of the subset that an expert e_{θ_i} was trained on. Intuitively, we are weighting the set of images associated to the i-th expert and uniformly sampling from it. We construct our dataset by sampling examples from $\mathcal{S}$ at a rate according to $\boldsymbol{p}$.

3.4 Relation to Domain Adaptation

If we assume that the client and server tasks are exactly the same then our problem can be interpreted as doing domain adaptation in each of the subset $\hat{\mathcal{S}}$ and the following generalization bound from Ben-David et al. (2009) can be used:

$$\varepsilon_{\mathcal{T}}(h) < \varepsilon_{\hat{\mathcal{S}}}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\hat{\mathcal{S}}, \mathcal{T}) \tag{7}$$

where ε represents the risk of a hypothesis function $h \in \mathcal{H}$ and $d_{\mathcal{H}\Delta\mathcal{H}}$ is the $\mathcal{H}\Delta\mathcal{H}$ divergence Ben-David et al. (2009), which relies on the capacity of $\mathcal{H}$ to distinguish between data points from $\hat{\mathcal{S}}$ and $\mathcal{T}$, respectively.

Let us further assume that the risk of the hypothesis function h on any subset $\hat{\mathcal{S}}$ is similar such that $\varepsilon_{\hat{\mathcal{S}}}(h) \approx \varepsilon_{\mathcal{S}}(h)$ for every $\hat{\mathcal{S}} \subset \mathcal{P}(\mathcal{S})$ and $h \in \mathcal{H}$. Under this assumption, minimizing equation 1 is equivalent to finding the subset $\mathcal{S}_*$ that minimizes the divergence with respect to $\mathcal{T}$. Formally,

$$\mathcal{S}_* = \arg\min_{\hat{\mathcal{S}}} d_{\mathcal{H}\Delta\mathcal{H}(\hat{\mathcal{S}}, \mathcal{T})} \tag{8}$$

In practice, it is hard to compute $d_{\mathcal{H}\Delta\mathcal{H}}$ and this is often approximated by a so called *proxy $\mathcal{A}$-distance* Ben-David et al. (2007); Chen et al. (2015); Ganin et al. (2015). A classifier that discriminates between the two domains and whose risk ε is used to approximate the second part of the equation.

$$\hat{d}_{\mathcal{H}} \approx \hat{d}_{\mathcal{A}} \approx 2(1 - 2\varepsilon) \tag{9}$$

Note that doing so would require having access to $\mathcal{S}$ and $\mathcal{T}$ in at least one of the two sides (i.e to train the new discriminative classifier) and this is prohibitive in our scenario. In our case, we compute the domain confusion between $\hat{\mathcal{S}}$ and $\mathcal{T}$ by evaluating the performance of expert e_i on the target domain. We argue that this proxy task performance (or error rate) is an appropriate proxy distance that serves the same purpose but does not violate the data visibility condition. Intuitively, if the features learned on the subset cannot be discriminated from features on the target domain, the domain confusion is maximized. We empirically show the correlation between the domain classifier and our proposed proxy task performance in our experiments.

4 EXPERIMENTS

4.1 TOY EXPERIMENT - DOMAIN CONFUSION

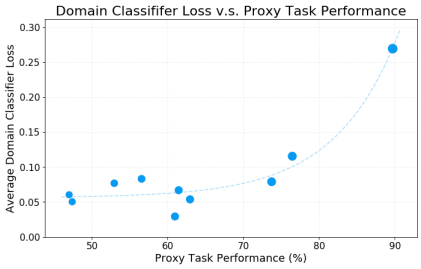

Figure 3: Relationship between domain classifier and proxy task performance on subsets $\hat{\mathcal{S}}$.

To see how well the performance of the proxy task reflects the domain confusion, we perform an experiment comparing the proxy task performance and $\hat{d}_{\mathcal{A}}(\hat{\mathcal{S}}, \mathcal{T})$. To estimate $\hat{d}_{\mathcal{A}}$, we follow the same idea from Ben-David et al. (2007); Chen et al. (2015); Ganin et al. (2015) and for each subset $\hat{\mathcal{S}}$, we estimate the domain confusion. Figure 3 shows the domain confusion vs the proxy task performance using OxfordIIIT-Pets (Parkhi et al., 2012) dataset as the target domain. In this plot, the highest average loss corresponds to the subset with the highest domain confusion (i.e., $\mathcal{S}_i$ that is the most indistinguishable from the target domain). Notice that this correlates with the expert that gives the highest proxy task performance.

4.2 EXPERIMENTAL SETUP

We perform experiments in classification, detection, and instance segmentation tasks on two server datasets and seven client datasets. In our experiments, we first train expert models on the server dataset $\mathcal{S}$, and then use the experts to select an optimal $\mathcal{S}_*$ for each target dataset as described in Section 3.3.1. We evaluate the performance on the target task by pre-training on the selected subset $\mathcal{S}_*$, and use this as an initialization for training over the target dataset. For all self-supervised experts, we use *ResNet18* (He et al., 2015), and train our models to predict image rotations.

4.2.1 IMAGE CLASSIFICATION SETUP

For classification tasks, we use the Downsampled ImageNet (Chrabaszcz et al., 2017) as our server dataset. This is a variant of ImageNet (Deng et al., 2009) resized to 32×32 resolution, with 1,281,167 training images from 1,000 classes. We consider several small classification datasets to be used as target datasets (Nilsback & Zisserman, 2008; Wah et al., 2011; Parkhi et al., 2012; Krause et al., 2013; Khosla et al., 2011). We use ResNet18 (He et al., 2015) as the base network architecture, and an input size of 32×32 for all classification datasets. Once the subsets are selected, we pre-train on the selected $\mathcal{S}_*$ and evaluate the transfer performance by fine-tuning on client (target) datasets.

4.2.2 OBJECT DETECTION AND INSTANCE SEGMENTATION SETUP

For detection and segmentation experiments, we use MS-COCO (Lin et al., 2014) as our server dataset. We evaluate the results using the standard metrics on Cityscapes (Cordts et al., 2016) and KITTI (Geiger et al., 2012) as the target datasets. We use Mask R-CNN models with ResNet-FPN-50 backbone, and follow the same training procedure as (He et al., 2017) for all experiments. We keep all hyperparameters fixed across all training runs and vary the choice of server data used for pre-training.

4.3 RESULTS AND ANALYSIS

We begin by investigating the impact of pre-training data sampled using our approach on the downstream performance. In Table 1, we summarize our result for classification, object detection, and instance segmentation tasks by subsampling 20%, 40% of the source dataset to be used for pre-training. By carefully selecting a similar subset of pre-training data using our approach, we see an improvement on all downstream tasks performance compared with pre-training on randomly selected subset of the same size. Moreover, when using 20% or 40% of pre-train data, we see comparable or improved performance of using the selected subset compared to pre-training on the entire 100% of pre-train data.

For classification tasks, we compare our method with the approach recently proposed by Ngiam et al. where they sample data based on the probability over source dataset classes computed by pseudo-

	Target Task	**Classification (% accuracy)**		**Detection (% box AP)**		**Segmentation (% mask AP)**	
	Source Dataset	Downsampled ImageNet		COCO		COCO	
	Target Dataset	Oxford-IIIT Pets	CUB200 Birds	Cityscapes	KITTI	Cityscapes	KITTI
0%	Random Initialization	32.4	25.1	36.2	21.8	32.0	17.8
100%	Entire Dataset	79.1	57.0	41.8	28.6	36.5	22.1
20%	Uniform Sample	71.1	48.6	38.1	22.2	34.3	18.9
	(Ngiam et al., 2018)	81.3	54.3	—	—	—	—
	Ours	82.0	54.8	40.7	27.3	36.1	21.0
40%	Uniform Sample	76.0	52.7	39.8	23.4	34.4	18.8
	(Ngiam et al., 2018)	81.0	57.4	—	—	—	—
	Ours	81.5	57.3	42.2	26.7	36.7	21.2

Table 1: Transfer learning results on classification, object detection, and instance segmentation. Each row corresponds to data selection method, and we indicate the size of the subset (e.g., either 20% or 40% of the entire source dataset). Each column corresponds to a target dataset.

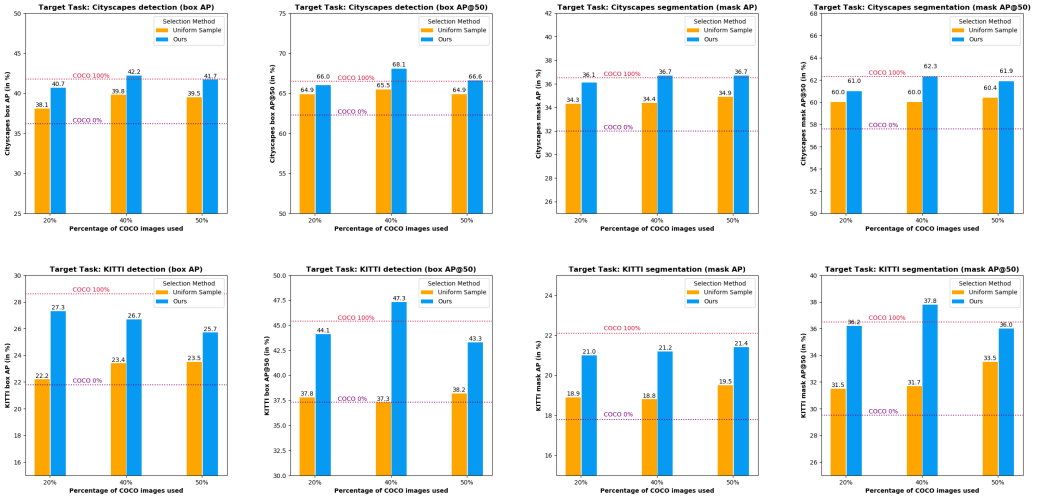

Figure 4: Transfer learning on object detection and instance segmentation. We report results on Cityscapes (top row) and KITTI (bottom row) when sampling {20%, 40%, 50%} of MS-COCO images (server).

labeling the target dataset with a classifier trained on the source dataset. Note that this approach is limited to the classification task, and cannot handle diverse tasks. Furthermore, it does not scale to a growing *dataserver*. Our approach achieves comparable results to Ngiam et al. in classification, and can be additionally applied to source datasets with no classification labels such as MS-COCO or even datasets which are not labeled.

Figure 4 shows the AP (average precision averaged over intersection-over-union (IoU) overlap thresholds 0.5:0.95) and AP@50 (average precision computed at IoU threshold 0.5) for object detection and segmentation after fine-tuning the Mask R-CNN on Cityscapes and KITTI dataset. A general trend is that performance is improved by pre-training for the instance segmentation task using COCO compared to ImageNet pre-training (COCO 0%). This suggests that a pre-training task other than classification is beneficial to improve transfer performance on localization tasks such as

Size	**Selection Method**	box AP	mask AP	mask AP_{50}	car	truck	rider	bicycle	person	bus	mcycle	train
0%	—	36.2	32.0	57.6	49.9	30.8	23.2	17.1	30.0	52.4	17.9	35.2
20%	Uniform Sample	38.1	34.3	60.0	50.0	34.2	24.7	19.4	32.8	52.0	18.9	42.1
	Ours	40.7	36.1	61.0	51.3	35.4	25.9	20.4	33.9	56.9	20.8	44.0
40%	Uniform Sample	39.8	34.4	60.0	50.7	31.8	25.4	18.3	33.3	55.2	21.2	38.9
	Ours	42.2	36.7	62.3	51.8	36.9	26.4	19.8	33.8	59.2	22.1	44.0
50%	Uniform Sample	39.5	34.9	60.4	50.8	34.8	26.3	18.9	33.2	55.5	20.8	38.7
	Ours	41.7	36.7	61.9	51.7	37.2	26.9	19.6	34.2	56.7	22.5	44.5
100%	—	41.8	36.5	62.3	51.5	37.2	26.6	20.0	34.0	56.0	22.3	44.2

Table 2: Transfer to object detection and instance segmentation with Mask R-CNN on Cityscapes. Each row corresponds to a selection method and the percentage of MS-COCO images used for pre-training.

Pre-Training Selection Method		Target Dataset				
		Stanford Dogs	Stanford Cars	Oxford-IIIT Pets	Flowers 102	CUB200 Birds
0%	Random Initialization	23.66	18.60	32.35	48.02	25.06
100%	Entire Dataset	64.66	52.92	79.12	84.14	56.99
20%	Uniform Sample	52.84	42.26	71.11	79.87	48.62
	Fast Adapt (SP+TS)	72.21	44.40	81.41	81.75	54.00
	Fast Adapt (SP+SS)	73.46	44.53	82.04	81.62	54.75
	Fast Adapt (UP+SS)	66.97	44.15	79.20	80.74	52.66
40%	Uniform Sample	59.43	47.18	75.96	82.58	52.74
	Fast Adapt (SP+TS)	68.66	50.67	80.76	83.31	58.84
	Fast Adapt (SP+SS)	69.97	51.40	81.52	83.27	57.25
	Fast Adapt (UP+SS)	67.16	49.52	79.69	83.51	57.44

Table 3: Ablation experiments on gating and expert training. SP stands for Superclass Partition, UP for Unsupervised Partition, TS for Task-Specific experts (experts trained on classif. labels), and SS for Self-Supervised experts (experts trained to predict image rotation). Results reported are top-1 accuracy for all datasets.

detection and segmentation, and shows the importance of training data. Next, we can see that pre-training using subsets selected by our approach is 2-3% better than the uniform sampling baseline, and that using 40% or 50% of COCO yields comparable (or better) performance to using 100% of data for the downstream tasks on Cityscapes. Table 2 further shows the instance segmentation performance on the 8 object categories for Cityscapes.

In Table 3, we compare different instantiations of our approach on five classification datasets. For all instantiations, pre-training on our selected subset significantly outperforms the pre-training on a randomly selected subset of the same size. Our result in Table 3 shows that under the same superclass partition, the subsets obtained through sampling according to the transferability measured by self-supervised experts (SP+SS) yield a similar downstream performance compared to sampling according to the transferability measured by the task-specific experts (SP+TS). This suggests that self-supervised training for the experts can successfully be used as a proxy to decide which data points from the source dataset are most useful for the target dataset.

5 Conclusion

In this work, we propose a novel method that aims to optimally select subsets of data from a large dataserver given a particular target client. In particular, we represent the server's data with a mixture of experts trained on a simple self-supervised task. These are then used as a proxy to determine the most important subset of the data that the server should send to the client. We experimentally show that our method is general and can be applied to any pre-training and fine-tuning scheme and that our approach even handles the case where no labeled data is available (only raw data). We hope that our work opens a more effective way of performing transfer learning in the era of massive datasets.

REFERENCES

Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Apostol Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *ArXiv*, abs/1609.08675, 2016.

David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. 2018.

David Acuna, Amlan Kar, and Sanja Fidler. Devil is in the edges: Learning semantic boundaries from noisy annotations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In B. Schölkopf, J. C. Platt, and T. Hoffman (eds.), *Advances in Neural Information Processing Systems 19*, pp. 137–144. MIT Press, 2007. URL http://papers.nips.cc/paper/2983-analysis-of-representations-for-domain-adaptation.pdf.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79:151–175, 2009.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *ACM Conference on Computer and Communications Security*, 2017.

Sergi Caelles, Alberto Montes, Kevis-Kokitsi Maninis, Yuhua Chen, Luc Van Gool, Federico Perazzi, and Jordi Pont-Tuset. The 2018 davis challenge on video object segmentation. *ArXiv*, abs/1803.00557, 2018.

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *ArXiv*, abs/1903.11027, 2019.

Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Leveraging large-scale uncurated data for unsupervised pre-training of visual features. *ArXiv*, abs/1905.01278, 2019.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016. URL http://dblp.uni-trier.de/db/journals/corr/corr1606.html#ChenPK0Y16.

Minmin Chen, Kilian Q Weinberger, Zhixiang Xu, and Fei Sha. Marginalizing stacked linear denoising autoencoders. *The Journal of Machine Learning Research*, 16(1):3849–3875, 2015.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets, 2017.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3213–3223, 2016.

Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017.

Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge J. Belongie. Large scale fine-grained categorization and domain-specific transfer learning. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4109–4118, 2018.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Franois Laviolette, Mario Marchand, and Victor S. Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17:59:1–59:35, 2015.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1v4N2l0-.

Sam Gross, Marc'Aurelio Ranzato, and Arthur Szlam. Hard mixtures of experts for large scale weakly supervised vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6865–6873, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

Kaiming He, Ross B. Girshick, and Piotr Dollár. Rethinking imagenet pre-training. *CoRR*, abs/1811.08883, 2018. URL http://arxiv.org/abs/1811.08883.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv:1811.00982*, 2018.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 181–196, 2018.

H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *ArXiv*, abs/1602.05629, 2016.

Jiquan Ngiam, Daiyi Peng, Vijay Vasudevan, Simon Kornblith, Quoc V. Le, and Ruoming Pang. Domain adaptive transfer learning with specialist models, 2018.

M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, April 2017. ISSN 0162-8828. doi: 10.1109/TPAMI.2016.2572683. URL https://doi.org/10.1109/TPAMI.2016.2572683.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.

Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 969–977, 2018.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

Amir Roshan Zamir, Alexander Sax, William B. Shen, Leonidas J. Guibas, Jagannath Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3712–3722, 2018.