TACKLING DIVERSE TASKS VIA CROSS-MODAL TRANSFER LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning large-scale pretrained models has led to remarkable progress in well-studied modalities such as vision and NLP. However, similar gains have not been observed in many other tasks due to an assumed lack of relevant pretrained models for these diverse modalities. In this work, we revisit this assumption by studying the cross-modal transfer ability of large-scale pretrained models. We introduce ORCA, a general cross-modal fine-tuning workflow that enables fast and automatic exploitation of existing pretrained models for diverse tasks. ORCA achieves task-specific adaptation by learning feature embeddings that minimize an optimal transport distance metric to map the data distribution in the end-task modality to the pretraining modality. We test ORCA on 13 tasks with varying modalities and input-output types. ORCA performs the best on 10 of them and is in the top three on the others. We further quantify the importance of embedding distance for downstream performance, highlight ORCA's utility for data-limited tasks, and demonstrate its compatibility with same-modality transfer.

1 INTRODUCTION

The success of machine learning (ML) in vision and natural language processing (NLP) has spurred its application beyond these traditional ML domains to diverse tasks such as solving partial differential equations (Li et al., 2021b), music modeling (Lewandowski et al., 2012), detecting cardiac disease (Hong et al., 2020), and many others. However, progress in these less-explored areas can be challenging due to (1) limited amounts of labeled data, (2) high computational cost and human effort for developing models from scratch, and (3) a lack of relevant large-scale pretrained models, which have in many cases obviated the first two issues in vision and NLP (e.g., Devlin et al., 2019; Carion et al., 2020; Dosovitskiy et al., 2021; Liu et al., 2021b; Radford et al., 2021).

There are two common approaches for practitioners to handle these issues: automated machine learning (AutoML) techniques (e.g., Roberts et al., 2021; Shen et al., 2022) that focus on designing task-specific networks in a data-efficient manner; and multimodal general-purpose methods that either propose flexible architectures applicable to various tasks (Jaegle et al., 2022a) or expand the set of modalities for which pretrained models exist (e.g., Reed et al., 2022; Lu et al., 2022a). However, both classes of approaches require training from scratch when applied to a new modality and proceed under the assumption of a lack of relevant pretrained models for these diverse problems.

In this work, we re-examine this assumption by considering the general problem of *cross-modal transfer*. Our goal is to exploit existing large-scale pretrained models in data-rich modalities for solving diverse downstream tasks. A few recent works have demonstrated the potential promise of cross-modal transfer by applying language transformers to vision (Kiela et al., 2019; Dinh et al., 2022; Lu et al., 2022b), referential games (Li et al., 2020c), and reinforcement learning (Reid et al., 2022). However, many of these approaches are ad-hoc (e.g., rely on manual prompt engineering or hand-craft new architecture components to solve specific tasks), and none of them yield models competitive with those trained from scratch. We tackle both shortcomings in our work.

We introduce a general-purpose, cross-modal transfer approach called **ORCA** (**O**ptimal t**R**ansport **C**ross-modal **A**daptation) that yields state-of-the-art results on a wide range of non-text and non-vision problems using pretrained transformers. Our key insight is to learn a task-specific feature embedding network before performing fine-tuning on the target task (see Figure 1 for the exact ORCA workflow). We train the embedder network to minimize the optimal transport dataset distance

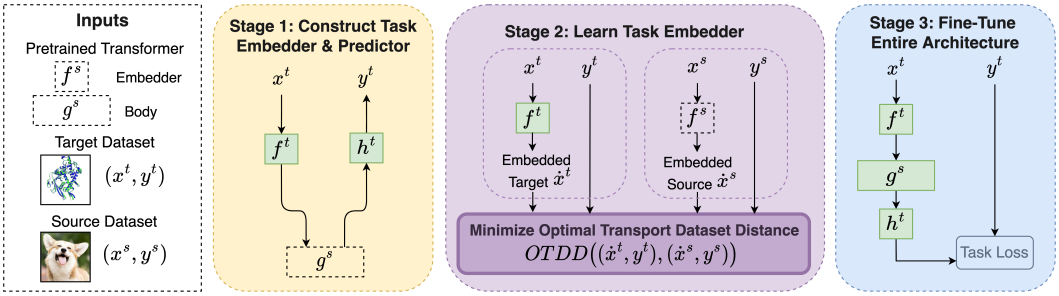

Figure 1: ORCA's three-stage cross-modal transfer workflow enables fast and automatic exploitation of large-scale pretrained models in data-rich modalities for solving diverse tasks. First, given target data (x^t, y^t) and a pretrained transformer body g^s, ORCA constructs an embedder architecture f^t to match the input dimensionality of g_s, and a predictor architecture h^t to convert the output of g^s back to the appropriate output space for the target task, e.g., classification logits or dense maps. Note that ORCA does *not* learn the weights for f_t or h_t during this stage. Next, ORCA learns the parameters for the embedder f^t by minimizing the OTDD between the target dataset and an in-modality source dataset. Finally, ORCA fine-tunes the entire architecture $\{f^t, g^s, h^t\}$.

(OTDD) (Alvarez-Melis & Fusi, 2020) between the feature-label distributions of the target data and data from the same modality as the pretraining data.[1] Intuitively, this step maps the feature distribution of an unfamiliar, cross-modal dataset back to that of a familiar, in-modal dataset. Using OTDD allows us to relax many distributional assumptions required by traditional domain adaptation methods. Moreover, this feature embedding step differentiates us technically from previous works that perform simple fine-tuning for cross-modal transfer (Lu et al., 2022b) and leads to significantly improved downstream performance (see Section 4).

We evaluate ORCA on a diverse set of 13 tasks with different input dimensions (1D and 2D), prediction types (point and dense), and modalities (vision, audio, electrocardiogram, mathematics, protein, genomics, cosmic-ray, and music). Our results show that ORCA outperforms various competitors, including task-specific hand-designed models, leading AutoML methods, and different fine-tuning approaches, ranking first on 10 tasks and in the top three on all tasks. We further perform experiments that reveal an empirical correlation between the embedding OTDD and the fine-tuning performance. Besides, we demonstrate ORCA's efficacy for both in-modal and limited-data tasks. Overall, our work not only explores the cross-modal transfer ability of pretrained models, but also establishes ORCA as a practical workflow for solving diverse prediction problems efficiently and automatically.

2 Related Work

In this section, we review several groups of related work in the areas of *AutoML*, *in-modal transfer learning* (unimodal domain adaptation, unimodal/multimodal fine-tuning, and general purpose methods), and *cross-modal transfer learning* (heterogeneous domain adaptation, task-specific fine-tuning, and FPT). Table 1 summarizes these groups along relevant axes, and contrasts them to ORCA.

AutoML for diverse tasks is a growing research area, as evidenced by the NAS-Bench-360 benchmark (Tu et al., 2022), along with several recent neural architecture search (NAS) methods that target this problem, e.g., AutoML-Zero (Real et al., 2020), XD (Roberts et al., 2021), and DASH (Shen et al., 2022)). In contrast to these NAS methods, ORCA takes a transfer learning approach in order to leverage existing pretrained models from data-rich modalities for more esoteric tasks, rather than repeatedly incurring the overhead of designing new architectures and training them from scratch. That said, given the shared underlying motivation, our experimental evaluation makes use of the diverse tasks comprising NAS-Bench-360, and compares ORCA with its expert and AutoML baselines. We also compare against DASH, the state-of-the-art method on this benchmark.

Unimodal domain adaptation (DA) is a form of transductive transfer learning where the source and target tasks are the same but the domains differ (Pan & Yang, 2009; Wang & Deng, 2018). Many DA methods assume that the target data has the same input space and support as the source data, and are

[1] We do not assume access to the pretraining data due to practical concerns about data access and computational efficiency. We instead work with publicly available proxy data from the pretraining modality, e.g., CIFAR-10 for models pretrained on ImageNet and CoNLL-2003 for models pretrained on larger text corpora.

Table 1: Summary of existing approaches aiming to develop models for diverse tasks.

		Task-specific adaptation?	General-purpose workflow?	Supports transfer to different: input dim?	output dim?	modality?
Task-specific learning	Hand-designed models	✓				
	AutoML models	✓	✓			
In-modal transfer	Unimodal DA	✓		✓		
	Uni/Multimodal fine-tuning	✓		✓	✓	
	General-purpose models	✓	✓	✓	✓	
Cross-modal transfer	Heterogeneous DA	✓		✓		✓
	Task-specific fine-tuning	✓		✓	✓	✓
	FPT		✓	✓	✓	✓
	ORCA	✓	✓	✓	✓	✓

concerned with problems where the output spaces and the joint/marginal distributions differ, such as covariate and label shifts. Recent work considers more general settings such as different feature spaces (heterogeneous DA) or label spaces (universal DA). Our focus on cross-modal transfer goes one step further to the case where neither the input-space nor the output-space support overlaps.

Unimodal fine-tuning of pretrained models is a more flexible transfer approach that can be applied to downstream tasks with different label spaces or input spaces. Pretrained models are used for in-modality fine-tuning in NLP (e.g., Aghajanyan et al., 2021; Jiang et al., 2020), vision (e.g., Wei et al., 2022; Li et al., 2022), speech (e.g., Chen et al., 2022; Jiang et al., 2021), protein sequences (Jumper et al., 2021) and robotics (Ahn et al., 2022). Prompting (Liu et al., 2022) and adapter networks (He et al., 2022) have also been developed to improve the downstream performance of in-modality transfer. **Multimodal fine-tuning** expands the applicable modalities of a single pretrained model (e.g., Lu et al., 2019; Radford et al., 2021; Hu & Singh, 2021; Kim et al., 2021; Baevski et al., 2020; Alayrac et al., 2022) by learning embeddings of several data-rich modalities together. However, these approaches still focus on solving in-modality downstream tasks.

General-purpose models propose flexible architectures applicable to various tasks such as optical flow, point clouds, and reinforcement learning (Jaegle et al., 2021; 2022a; Reed et al., 2022). These approaches train multitask transformers from scratch using a large body of data from different tasks. Though more versatile than unimodal models, they still focus on transferring to problems within the pretraining modalities considered. Nonetheless, the success of transformers for in-modality fine-tuning motivates us to focus on adapting transformer-type architectures for cross-modal transfer.

Heterogeneous DA (HDA) considers nonequivalent feature spaces between the source and target domains. While most HDA methods are developed for same-modality-different-dimension transfer, e.g., between images of different resolutions, there are indeed a few works studying cross-modal tasks such as text-to-image (Yao et al., 2019; Li et al., 2020b). However, a crucial assumption that HDA makes is that the target and source tasks are the same. Thus, we operate in a much more flexible setting and consider knowledge transfer between drastically different domains with distinct tasks and label sets, such as applying Swin Transformers (Liu et al., 2021c) to solving partial differential equations or RoBERTa to classifying satellite images and electrocardiograms.

Cross-modal, task-specific fine-tuning is a recent line of research, with much of the work focused on transferring NLP models to other modalities, e.g., vision (Kiela et al., 2019), referential games (Li et al., 2020c), and reinforcement learning (Reid et al., 2022). These works provide initial evidence of the cross-modal transfer capabilities of pretrained models. However, they focus on hand-tailoring to a single modality, e.g., by adding ad-hoc encoders that transform agent messages (Li et al., 2020c) or decision trajectories (Reid et al., 2022) into tokens. Even when not relying on fine-tuning, work like LIFT (Dinh et al., 2022) that attempts cross-modal learning via prompting (Liu et al., 2021a) still require ad-hoc conversion of tasks to natural text.

Frozen Pretrained Transformers (FPT) (Lu et al., 2022b) is a general cross-modal fine-tuning workflow that transforms input features to be compatible with the pretrained models. Although FPT and ORCA are both general-purpose workflows, FPT does not account for differences between the target and pretraining modalities, which we show is necessary to achieve accurate predictive models and outperform existing baselines.

3 ORCA WORKFLOW

In this section, we first formalize the problem setup and then introduce the ORCA workflow for adapting pretrained transformers to diverse end tasks.

Problem Setup. A domain $\mathcal{D}$ consists of a features space $\mathcal{X}$, a label space $\mathcal{Y}$, and a joint probability distribution $P(\mathcal{X}, \mathcal{Y})$. In the cross-modal setting we study, the target (end-task) domain $\mathcal{D}^t$ and source (pretraining) domain $\mathcal{D}^s$ differ not only in the feature space but also the label space and by extension have differing probability distributions, i.e., $\mathcal{X}^t \neq \mathcal{X}^s$, $\mathcal{Y}^t \neq \mathcal{Y}^s$, and $P^t(\mathcal{X}^t, \mathcal{Y}^t) \neq P^s(\mathcal{X}^s, \mathcal{Y}^s)$. This is in contrast to the transductive transfer learning setting addressed by domain adaptation, where source and target domains share the label space and end task (Pan & Yang, 2009).

Given target data $\{x_i^t, y_i^t\}_{i \in [n^t]}$ sampled from a joint distribution P^t in domain $\mathcal{D}^t$, our goal is to learn a prediction model m^t that correctly maps each input x^t to its label y^t. We are interested in achieving this by leveraging pretrained transformers. Thus, we assume access to a model m^s that has been trained with data $\{x_i^s, y_i^s\}_{i \in [n^s]}$ in the source domain $\mathcal{D}^s$, where $(x_i^s, y_i^s) \sim P^s$. Then, given a predefined loss function l on the target data, we aim to develop m^t based on m^s such that $L(m^t) = \mathbb{E}_{(x^t, y^t) \sim P^t}[l(m^t(x^t), y^t)]$ is minimized.

This problem formulation does not define modality explicitly and includes both in-modal and cross-modal transfer problems. Given the generality of the tasks we wish to explore, it is hard to provide a precise mathematical definition, so we rely on semantics to differentiate the two settings. Intuitively, cross-modal domains, such as natural images vs. protein sequences, should be more distinct to each other than in-modal domains, such as photos taken in two different geographical locations.

Having defined the problem and learning objective, we can now introduce our methodology for transfer learning across modalities, which consists of three stages: (1) architecture design to support diverse input and output tensor dimensions, (2) embedder pretraining to match the source and target domain feature distributions, and (3) fine-tuning to minimize the target task loss.

3.1 Task-Specific Architecture Design

Applying pretrained models to another downstream problem usually requires addressing the problem of mismatched dimensions. To make ORCA work for input and output tensors of different dimensions, we decompose a transformer-based learner m into three parts (Figure 1 stage 1): an embedder f that transforms input x into a sequence of features, a model body g that applies a pretrained transformer (i.e., series of attention layers) to the embedded features, and a predictor h that generates predictions with the desired output shape. ORCA uses pretrained architecture and weights to initialize the model body g but replaces f and h with layers designed to match the target data with the pretrained model's embedding dimension. Next, we describe each module in detail.

Custom Embedding Network. Denote the feature space compatible with the pretrained model body as $\dot{\mathcal{X}}$. For a transformer with maximum sequence length S and embedding dimension D, $\dot{\mathcal{X}} = \mathbb{R}^{S \times D}$. The embedding network $f : \mathcal{X} \to \dot{\mathcal{X}}$ is designed to take in a tensor of arbitrary dimension from $\mathcal{X}$ and transform it to the feature space $\dot{\mathcal{X}}$. In ORCA, f is composed of a convolutional layer with input channel c_{in}, output channel c_{out}, kernel size k, and stride k, generalizing the patching operations used in vision transformers. We set c_{in} to the input channel of x and c_{out} to the embedding dimension D. To take full advantage of the representation power of the pretrained model, we choose the smallest k for which the product of output shape excluding the channel dimension $\leq S$. That is, when we flatten the non-channel dimensions of the output tensors after the convolution, pad and then transpose it, we can obtain sequence features with shape $S \times D$. Finally, we add a layer norm and a positional embedding to obtain $\dot{x}$.[2]

Pretrained Transformer Body. The model body g takes the embedding $\dot{x} \in \dot{\mathcal{X}}$ as input and outputs features $\dot{y} \in \dot{\mathcal{Y}}$; the dot is used to differentiate these intermediate representations from the raw inputs and labels. For transformer-based g, both the input and output feature spaces $\dot{\mathcal{X}}, \dot{\mathcal{Y}}$ are $\mathbb{R}^{S \times D}$.

Custom Prediction Head. Finally, the prediction head h must take $\dot{y} \in \dot{\mathcal{Y}}$ as input and return a task-dependent output tensor. Different tasks often specify different types of outputs, e.g., classification tasks require logits in $\mathbb{R}^K$ where K is the number of classes, and dense prediction tasks require dense maps with the same spatial dimension as the input and per index logits corresponding to K classes. Thus, it is crucial to define task-specific output modules and fine-tune them when

[2] As a concrete example, consider an image tensor with shape (C_{in}, H_{in}, W_{in}). We first choose stride k for the convolution such that $H_{out} \times W_{out} \approx S$ to get an output tensor with shape (D, H_{out}, W_{out}). Then, we flatten it to shape $(D, H_{out} \times W_{out})$, pad along the last dimension to shape (D, S), and transpose.

transferring to new tasks. In our workflow, we use the simplest possible instantiation of the predictor modules. For classification, we apply average pooling along the sequence length dimension (or take the classification token of language models) to obtain 1D tensors with length D and then use a linear layer that maps D to K. For dense prediction, we apply a linear layer to the sequence outputs so the resulting tensor has shape $(S, k^{\dim(\mathcal{Y})-1}K)$, where $k^{\dim(\mathcal{Y})-1}$ is the downsampling factor of the embedder convolution kernel with stride k. This upsamples by the same factor that the embedder convolution downsampled. Then, we can mold the tensor to the desired output dimension.[3]

With these modifications, we now have an architecture based on the pretrained model but is also compatible with our target task. Next, we turn our attention to pretraining the embedder for task-specific adaptation via matching source and target embedding distributions.

3.2 Embedding Learning

Intuitively, transferring knowledge across similar modalities should be easier than across distant ones. Hence, given a target task in a new modality, we aim to manipulate the task data so that they become closer to the pretraining modality. We use the optimal transport dataset distance (OTDD) (Alvarez-Melis & Fusi, 2020) to measure the closeness between datasets in different domains. Unlike classic OT-based or maximum-mean-discrepancy-based metrics (Gretton et al., 2012) which only measure the difference between *feature* distributions, OTDD allows us to compare datasets using the *label* information, thus distinguishing us from unsupervised or semi-supervised DA methods that utilize the OT distance (Courty et al., 2017; Yan et al., 2018). More importantly, OTDD works even if the label sets are unrelated or disjoint, which makes it particularly suitable for cross-modal distance estimation. OTDD has been studied for dataset transformation (Alvarez-Melis & Fusi, 2021) but not for training or fine-tuning models as an optimization objective.

Formally, let $f^s : \mathcal{X}^s \to \dot{\mathcal{X}}$ denote the pretrained embedder (the part of m^s that transforms the source data to sequence features) and $f^t : \mathcal{X}^t \to \dot{\mathcal{X}}$ be a randomly initialized target embedder with architecture discussed in the previous section. We train f^t to minimize the expected OTDD between the embedding-label distributions $\big(f^t(x^t), y^t\big)$ and $\big(f^s(x^s), y^s\big)$. That is, for both datasets, we first represent each class label as a distribution over the in-class features: $y \mapsto P(\dot{\mathcal{X}}|\mathcal{Y} = y)$. This transforms the source and target label sets into the shared space of distributions over $\dot{\mathcal{X}}$. Then, we can define the distance $d_{\mathcal{Y}}(y^t, y^s)$ between different labels using the p-Wasserstein distance associated with a metric $d_{\dot{\mathcal{X}}}$ over the feature space, e.g., the l_2 distance $\|\dot{x}^t - \dot{x}^s\|_2^2$. This allows us to measure the difference between distributions in $\dot{\mathcal{X}} \times \mathcal{Y}$ using the following p-Wasserstein metric:

$$d_{\dot{\mathcal{X}}\times\mathcal{Y}}\big((\dot{x}^t, y^t), (\dot{x}^s, y^s)\big) = \big(d_{\dot{\mathcal{X}}}(\dot{x}^t, \dot{x}^s)^p + d_{\mathcal{Y}}(y^t, y^s)^p\big)^{1/p}. \tag{1}$$

Plugging this into the OT formulation leads to the OTDD over $\dot{\mathcal{X}} \times \mathcal{Y}$, which we optimize to learn f^t.

In Appendix A.1, we provide a more detailed exposition on OTDD and its computational complexity. We also show empirically that learning the embedder takes much shorter time than fine-tuning. We additionally highlight two points for implementing embedding learning with OTDD in practice. First, as stated in the introduction, we do not have to use the exact pretraining data to represent a model's source domain, as they are often not publicly available and can be too large to compute OTDD efficiently. Instead, we can use a smaller in-modal dataset as the proxy dataset. Second, classification tasks naturally come with discrete labels required for computing OTDD. For dense prediction tasks where labels are high-dimensional maps, we perform clustering on the dense maps to generate pseudo labels, which not only preserves the intrinsic distribution of the target data but also speedups OTDD computation.

3.3 Fine-tuning Model Parameters

After training the embedder, we perform full fine-tuning by updating all model parameters to minimize the target loss. This step further aligns the embedder and predictor with the pretrained model to improve downstream performance. We perform an ablation study comparing ORCA to standard

[3] As a concrete example, for an image tensor with embedding convolution kernel size k, the linear layer will yield an output of shape (S, k^2K), which we transpose, pad, and reshape to (k^2K, H_{out}, W_{out}). Finally, we apply pixelshuffle (Shi et al., 2016) to get an output of shape (K, H_{in}, W_{in}).

fine-tuning without feature matching in Section 4.2.1 and show that our approach improves prediction accuracy and reduces performance variance. There are orthogonal lines of work that study how to best fine-tune a pretrained model (e.g., Liu et al., 2022; He et al., 2022). We compare with one strategy used in FPT (Lu et al., 2022b) in Section 4.2.2 but leave further exploration for future work.

4 EXPERIMENTS

Having introduced how ORCA addresses the dimension mismatch between the target and source datasets via architecture design and tackles the distribution mismatch via embedder learning, we now proceed with showing its empirical effectiveness by evaluating it on a diverse suite of tasks. In the following, we will first demonstrate that ORCA can outperform hand-designed and AutoML-discovered models often by a large margin. Then, we analyze key components of ORCA to better understand the mechanism underlying cross-modal transfer.

Experiment Protocol. While our workflow accepts a wide range of pretrained transformers as model bodies, we use RoBERTa (Liu et al., 2019b) and Swin Transformers (Liu et al., 2021c), which are representatives of the most researched language and vision modalities, to exemplify ORCA's efficacy. We choose CoNLL-2003[4] for RoBERTa and CIFAR-10 for Swin as the proxy source datasets. We use the base models for both architectures, which have around 100 million parameters.

For each task, we first apply the hyperparameter tuning algorithm ASHA (Li et al., 2020a) to the standard fine-tuning baseline ("Fine-tuning" in Table 3) to identify suitable batch size, optimizer, learning rate, and weight decay. These hyperparameters are then applied to all fine-tuning related baselines as well as ORCA. We use the model implementations and pretrained weights available in the Hugging Face transformers library (Wolf et al., 2019) and manage our experiments with the Determined AI platform. All experiments are performed on NVIDIA V100 GPUs and results are averaged over 5 random seeds. For other experiment details, see Appendix A.3.

4.1 CAN PRETRAINED MODELS TRANSFER ACROSS MODALITY TO SOLVE DIVERSE TASKS?

In this section, we highlight the most important observation of this work: cross-modal fine-tuning with ORCA can solve a variety of tasks effectively and efficiently. To demonstrate this, we evaluate ORCA on 13 tasks detailed below. We first include 10 tasks from NAS-Bench-360[5], which covers problems such as PDE solving, protein folding, and cardiac disease detection. This benchmark contains tasks for 1D and 2D classification, 2D dense prediction, but not 1D dense prediction, so we added JSB Chorales, a music modeling dataset widely used for evaluating recurrent networks (Chung et al., 2017; Bai et al., 2018). We also added ListOps (parsing math expressions) (Tay et al., 2021) and Homology (classifying protein structure) (Rao et al., 2019) for comparison with FPT. Together, these 13 tasks represent a wide collection of modalities for comprehensive evaluation.

Following the taxonomy in Table 1, we consider three classes of baselines: (1) hand-designed expert architectures for each task, as identified by Tu et al. (2022), Rao et al. (2019), and Tay et al. (2021); (2) general-purpose models, as represented by Perceiver IO (Jaegle et al., 2022b); and (3) AutoML baselines, as represented by those evaluated in NAS-Bench-360 and DASH (Shen et al., 2022). We will compare with FPT later, the only remaining approach with a general workflow from Table 1.

In Table 2, we report the prediction error for each method on each task. ORCA achieves the lowest error rate on 10 of 13 tasks and is the most effective in terms of aggregated performance. This is also supported by the performance summary in Figure 2. More specifically, we outperform all hand-designed architectures on all tasks except ECG,

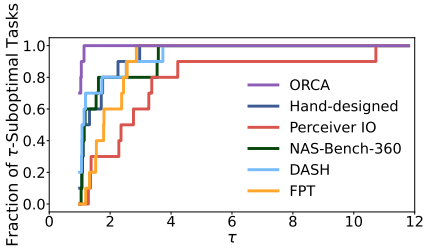

Figure 2: Aggregate performance of ORCA and baselines for Table 2 results measured by performance profiles (Dolan & Moré, 2002). Larger values (larger fractions of tasks on which a method is within τ-factor of the best) are better. ORCA being in the top left corner means it is often the best.

[4]CoNLL-2003 is for named entity recognition. It is used to interpret language models (Jawahar et al., 2019).

[5]NAS-Bench-360 is designed for testing how well ML algorithms can generalize and is a core component of the 2022 AutoML Decathlon competition. For a summary of included tasks, see Table 7 in the Appendix

Table 2: Prediction errors (lower is better) for 13 tasks across diverse application domains. On 10/13 problems, ORCA ranks the first among all hand-designed expert models, AutoML and general-purpose methods. NAS-Bench-360 refers to the aggregated performance (task-wise best) of all baselines evaluated in the benchmark, including DARTS (Liu et al., 2019a), DenseNAS (Fang et al., 2020), XGBoost (Chen & Guestrin, 2016), and 3 others. FPT refers to fine-tuning the layer norms of RoBERTa or Swin. For a list of hand-designed models, see Table 7 in the Appendix. Bold numbers indicate the best average performance.

	CIFAR-100 0-1 error (%)	Spherical 0-1 error (%)	Darcy Flow relative ℓ_2	PSICOV MAE_8	Cosmic 1-AUROC	NinaPro 0-1 error (%)	FSD50K 1- mAP
Hand-designed	19.39±0.20	67.41±0.76	8E-3±1E-3	3.35±0.14	0.24±0.015	8.73±0.9	0.62±0.004
NAS-Bench-360	23.39±0.01	48.23±2.87	2.6E-2±1E-3	2.94±0.13	0.22±0.035	7.34±0.76	0.60±0.001
DASH	24.37±0.81	75.44±2.38	7.9E-3±2E-3	3.30±0.16	0.25±0.02	**6.60±0.33**	0.60±0.008
Perceiver IO	70.04±0.44	82.57±0.19	2.4E-2±1E-2	8.06±0.06	0.48±0.01	22.22±1.80	0.72±0.002
FPT-Swin	10.11±1.18	76.38±4.89	2.1E-2±1.3E-3	4.66±0.054	0.32±0.0021	15.69±2.33	0.67±0.0068
ORCA-SWIN	**6.53±0.079**	**29.85±0.72**	**7.3E-3±6.8E-5**	**1.91±0.038**	**0.21±0.0050**	7.54±0.39	**0.56±0.013**

	ECG 1 - F1 score	Satellite 0-1 error (%)	DeepSEA 1- AUROC	JSB Chorales NLL	ListOps 0-1 error (%)	Homology 0-1 error (%)
Hand-designed	**0.28±0.00**	19.8±0.00	0.30±0.024	8.43±0.00	62.73±0.00	88±0.00
NAS-Bench-360	0.33±0.02	12.51±0.24	0.32±0.01	-	-	-
DASH	0.32±0.007	12.28±0.5	**0.28±0.013**	6.13±0.006	57.33±0.14	89.71±0.54
Perceiver IO	0.66±0.01	15.93±0.008	0.38±0.004	-	-	-
FPT-RoBERTa	0.50±0.0098	20.83±0.24	0.37±0.0002	2.72±0.026	64.35±0.79	91.48±1.14
ORCA-ROBERTA	0.29±0.0052	**11.59±0.18**	0.29±0.006	**2.44±0.056**	**51.90±2.18**	**87.21±0.50**

where we rank second but do much better than the other methods. We also beat all AutoML baselines on all tasks except DeepSEA and NinaPro, where ORCA is second and third, respectively. The improvements from ORCA come at a small computational overhead associated with pretraining the embedder to match the source and target modalities. Table 6 in the Appendix shows the time needed for embedder learning with OTDD, which is a small portion (10.2% on average) of the fine-tuning time. ORCA's efficiency and its state-of-the-art results on 10 tasks make it a practical tool for model development in diverse areas.

Our experiments further validate the findings in Lu et al. (2021) that pretrained transformers can learn knowledge transferable to seemingly unrelated tasks. In the following, we delve into the mechanism of ORCA to provide intuition for necessary components of successful cross-modal learning.

4.2 KEY FACTORS FOR SUCCESSFUL CROSS-MODAL TRANSFER

Here, we dissect the success of cross-modal transfer with ORCA through a series of ablation studies. As a preview, we identify three aspects to be key to its success: feature embedding with OTDD, full fine-tuning of all model weights, and suitable pretrained model selection.

4.2.1 MATCHING FEATURE DISTRIBUTIONS HELPS ADAPTATION

In our first set of experiments, we observe that OTDD can indeed represent the modality discrepancy between the target and source data. For instance, using the vision model Swin as the model body, the change in OTDD values before and after embedding learning is much larger for out-of-domain tasks like Spherical (15.9→11.78) than the in-modality task CIFAR-100 where OTDD remains low (2.01→1.94). We are thus interested in studying how minimizing OTDD can affect fine-tuning.

To isolate the impact of embedding learning, we compare the performance of ORCA with that of standard fine-tuning (updating all weights without pretraining the embedder) on all evaluated tasks. The top two rows of Table 3 show that ORCA consistently outperforms naive fine-tuning and often by a large margin. This suggests that closing the gap between a new modality and the pretraining modality can facilitate a model's adaptation to a new task.

From a task-wise perspective, we also train the embedder for different number of epochs before fine-tuning to see how optimizing the OTDD to various levels of convergence affects downstream performance. Figure 3 plots the fine-tuning accuracy along with the final OTDD objective for different levels of embedder pretraining. Evidently, as the dataset distance decreases, the final fine-tuning accuracy increases. This correlation supports the effectiveness of embedder learning for cross-modal transfer. In addition, we observe that learning the embedder prior to fine-tuning can stabilize training, as the performance variance of ORCA is consistently lower than that of standard fine-tuning.

Table 3: Prediction errors of ORCA and standard fine-tuning on 13 tasks. We consider updating all parameters (full fine-tuning setting) and updating only the layer norms (FPT setting). ORCA is better in both settings.

	CIFAR-100	Spherical	Darcy Flow	PSICOV	Cosmic	NinaPro	FSD50K
ORCA	**6.53±0.079**	**29.85±0.72**	**7.3E-3±6.8E-5**	**1.91±0.038**	**0.21±0.0050**	**7.54±0.39**	**0.56±0.013**
Fine-tuning	7.67±0.55	55.26±1.63	**7.3E-3±1.1E-4**	1.92±0.039	0.24±0.0080	8.35±0.75	0.63±0.014
ORCA (layernorm)	7.99±0.098	42.45±0.21	2.1E-2±7.4E-4	4.97±0.14	0.25±0.0020	15.99±1.92	0.64±0.0093
Fine-tuning (layernorm)	10.11±1.18	76.38±4.89	2.1E-2±1.3E-3	4.66±0.054	0.32±0.0021	15.69±2.33	0.67±0.0068

	ECG	Satellite	DeepSEA	JSB Chorales	ListOps	Homology
ORCA	**0.29±0.0052**	**11.59±0.18**	**0.29±0.006**	**2.44±0.056**	**51.90±2.18**	**87.21±0.50**
Fine-tuning	0.44±0.0056	13.86±1.47	0.51±0.0001	2.59±0.098	72.79±1.96	89.31±0.60
ORCA (layernorm)	0.47±0.007	20.54±0.49	0.36±0.0070	2.68±0.13	63.62±0.75	90.65±1.66
Fine-tuning (layernorm)	0.50±0.0098	20.83±0.24	0.37±0.0002	2.72±0.026	64.35±0.79	91.48±1.14

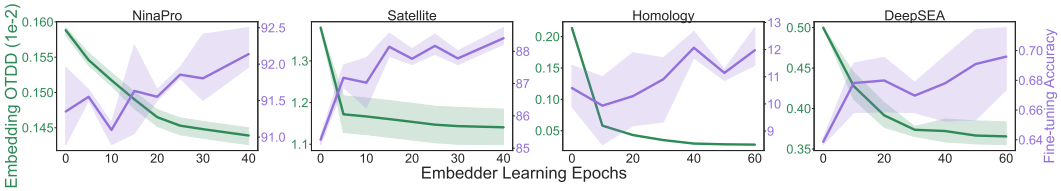

Figure 3: Final model accuracy and embedding OTDD vs. number of embedding training epochs when applying ORCA to four tasks. As the embedder learns to map the target data to the source modality better (smaller final OTDD), we generally obtain models with better downstream performance.

4.2.2 FINE-TUNING ALL MODEL PARAMETERS WHEN COMPUTATION ALLOWS

As discussed in Section 2, Frozen Pretrained Transformers (FPT) (Lu et al., 2022b) is a related approach for cross-modal transfer that showed pretrained language models contain knowledge relevant for out-of-modality tasks. While FPT presented a general fine-tuning pipeline that transfers GPT-2 to tasks like CIFAR-10, Homology, and ListOps, the resulting models were as good as those directly trained on the target data. FPT differs from ORCA in that (1) it does not pretrain the embedding layers for task-specific adaptation and (2) it only fine-tunes the layer norms. To isolate the impact these two components, we evaluate ORCA with fine-tuning the layer norms vs. FPT on our task set.

The bottom rows of Table 3 show ORCA with fine-tuning just the layer norms outperforms FPT, indicating pretraining the embedding layers boosts the cross-modal performance of FPT. However, this performance gain is smaller than that seen in the full fine-tuning setting, which implies that full fine-tuning can take better advantage of the learned embeddings. Also, fine-tuning a subset of the model weights is less effective than full fine-tuning in all tasks except for DeepSEA. This exception might be due to the fact that full fine-tuning without learned embeddings is more prone to overfitting.

In terms of runtime, we find that fine-tuning just the layer norms only results in less than 2x speedups compared with full fine-tuning, despite the fact that we are updating significantly fewer parameters (see Appendix Table 8 for detailed statistics). This is unsurprising since gradients are still back-propagated through the entire network. Therefore, when computational resources and time allow, we recommend using ORCA with full fine-tuning to achieve better downstream performance.

4.2.3 PRETRAINING MODALITY CAN AFFECT TRANSFER PERFORMANCE

For experiments in Table 2, we chose pretrained models for each task based on the input dimension, i.e., we use RoBERTa for all 1D tasks and Swin for all 2D tasks. To better understand how the pretraining modality affects fine-tuning, we switch the model bodies and apply ORCA. This is easy to implement because ORCA is model-agnostic and the embedder architecture handles all necessary input transformation to obtain sequence features. As shown in Table 4, fine-tuned RoBERTa outperforms fine-tuned Swin on the 1D task, and the final OTDD objective for RoBERTa is also smaller than that of Swin. We hypothesize that this is because the considered DeepSEA data (genomics sequences) are structured more like language than images with discrete

Table 4: Test errors of ORCA on DeepSEA and Spherical with language- and image-pretrained model bodies. Numbers in the parenthesis represent the OTDD after embedding learning. Smaller OTDD leads to better performance.

Error (OTDD)	DeepSEA (1D)	Spherical (2D)
RoBERTa (1D)	**0.295±0.006 (37.40)**	68.28±0.017 (19.54)
Swin (2D)	0.361±0.001 (64.83)	**29.85±0.072(11.78)**

units of information and general grammatical rules. The FPT paper observes a similar trend for Homology. As for the 2D tasks, we again notice that models with better fine-tuning accuracy have smaller OTDDs. This suggests a way of selecting pretrained models from a predefined model hub for each task, e.g., by comparing the optimized OTDDs and picking the one with the smallest value.

4.3 APPLICATIONS: LOW-DATA REGIME AND IN-MODALITY TRANSFER

One of our motivations for transferring pretrained models to various modalities is to utilize existing model resources to help tasks in data-limited regimes, where training models from scratch can be challenging. To this end, we investigate whether ORCA can facilitate fine-tuning large-scale models on small target datasets.

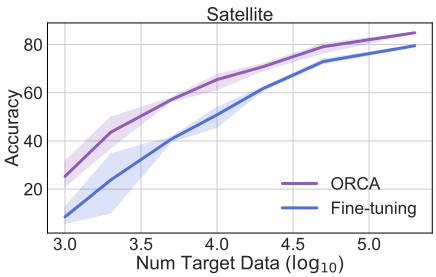

Figure 4: Prediction accuracy of ORCA and the fine-tuning baseline trained with different amounts of target data. ORCA has a larger performance gain in low-data regime.

Indeed, for standard fine-tuning, a small amount of data often cannot give enough signal to update the pretrained weights. However, using ORCA, it is possible to obtain a good feature embedder with the same amount of data, which can then reduce the difficulty of fine-tuning. In Figure 4, we vary the amount of target data used for fine-tuning and plot the final performance for both ORCA and the fine-tuning baseline. We make the following observations. First, the performance gain of ORCA increases as the amount of available data decreases. This means that standard fine-tuning does suffer from limited data, but fine-tuning with ORCA can considerably alleviate the problem and lead to better downstream results. Second, using ORCA allows us to match the performance of standard fine-tuning with three times the amount of the data. Consequently, ORCA can greatly benefit model development in practical domains where data collection is costly, e.g., chemistry or medical imaging.

Lastly, a natural question to ask is whether ORCA can also tackle in-modality tasks. While we design ORCA to enable cross-modal transfer, we hypothesize that it should facilitate same-modality transfer if two domains have large dataset distance. To validate this, we test ORCA on DomainNet datasets, which are commonly used to evaluate homogeneous DA methods (Peng et al., 2019). From Table 5, we can see that ORCA achieves significantly better performance than the fine-tuning baseline, which shows that the feature matching of ORCA can also help in-domain generalization.

Table 5: We use the dataset splits in Tan et al. (2020), which removed some mislabeled outliers, and report the prediction errors for ORCA and fine-tuning (using Swin-base).

	Real	Painting	Sketch	Clipart
ORCA	**96.71±0.02**	**94.71±0.13**	**94.93±0.24**	**93.61±0.54**
Fine-tuning	93.33±1.33	75.79±0.86	83.00±0.13	86.01±2.62

Discussion and Future Work. We identify several interesting directions to pursue based on our experiment results. First, it is worth studying the effect of pretraining modality further and come up with a systematic way of selecting the pretrained models. Then, we can incorporate model selection into ORCA for a more flexible cross-modal transfer pipeline. Second, in our workflow, we use the standard fine-tuning paradigm with an additional embedder learning stage to improve downstream performance. While vanilla fine-tuning is generally effective, we believe that there is still large room for improvement, e.g., by combining ORCA with more sophisticated transfer techniques such as adapters (He et al., 2022) and prompting (Liu et al., 2022). Lastly, we currently only evaluate ORCA on supervised 1D and 2D tasks. It is thus important to validate it on more diverse settings, such as high-dimensional problems or reinforcement learning (Reid et al., 2022).

5 CONCLUSION

In this paper, we argue that an important step towards developing more general ML methods is to study how we can reuse existing models effectively for new and less-explored tasks. To this end, we propose a novel framework that allows transferring pretrained transformers to distinct downstream modalities. Our method, ORCA, can map target data from an arbitrary end task's modality to a model's pretraining modality to improve fine-tuning performance. We believe that this work not only signals the potential of large-scale pretraining for diverse tasks but also lays out a path for a largely uncharted data-centric paradigm in machine learning.

REFERENCES

Badri Adhikari. DEEPCON: protein contact prediction using dilated convolutional neural networks with dropout. *Bioinformatics*, 36(2):470–477, 07 2019.

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and S. Gupta. Better fine-tuning by reducing representational collapse. *ArXiv*, abs/2008.03156, 2021.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Jayant Joshi, Ryan C. Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego M Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, and Mengyuan Yan. Do as i can, not as i say: Grounding language in robotic affordances. *ArXiv*, abs/2204.01691, 2022.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. *ArXiv*, abs/2204.14198, 2022.

David Alvarez-Melis and Nicoló Fusi. Geometric dataset distances via optimal transport. *ArXiv*, abs/2002.02923, 2020.

David Alvarez-Melis and Nicoló Fusi. Dataset dynamics via gradient flows in probability space. In *ICML*, 2021.

Alexei Baevski, Henry Zhou, Abdel rahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *ArXiv*, abs/2006.11477, 2020.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *ArXiv*, abs/1803.01271, 2018.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *ArXiv*, abs/2005.12872, 2020.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, et al. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 2022.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *ArXiv*, abs/1609.01704, 2017.

Taco Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. In *International Conference on Machine Learning*, 2018.

Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1853–1865, 2017.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NIPS*, 2013.

Angus Dempster, Franccois Petitjean, and Geoffrey I. Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34:1454–1495, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Shashank Rajput, Michael Gira, Jy yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. *ArXiv*, abs/2206.06565, 2022.

Elizabeth D. Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

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. *ArXiv*, abs/2010.11929, 2021.

Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Densely connected search space for more flexible neural architecture search. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10625–10634, 2020.

Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. Fsd50k: an open dataset of human-labeled sound events. *ArXiv*, abs/2010.00475, 2021.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alex Smola. A kernel two-sample test. *JMLR*, 13:723–773, 2012.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. *ArXiv*, abs/2110.04366, 2022.

Shenda Hong, Yanbo Xu, Alind Khare, Satria Priambada, Kevin O. Maher, Alaa Aljiffry, Jimeng Sun, and Alexey Tumanov. Holmes: Health online model ensemble serving for deep learning models in intensive care units. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

Ronghang Hu and Amanpreet Singh. Unit: Multimodal multitask learning with a unified transformer. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1419–1429, 2021.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver: General perception with iterative attention. In *ICML*, 2021.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Andrew Brock, Evan Shelhamer, Olivier J. H'enaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver io: A general architecture for structured inputs & outputs. *ArXiv*, abs/2107.14795, 2022a.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J Henaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=fILj7WpI-g.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does bert learn about the structure of language? In *ACL*, 2019.

Dongwei Jiang, Wubo Li, Ruixiong Zhang, Miao Cao, Ne Luo, Yang Han, Wei Zou, Kun Han, and Xiangang Li. A further study of unsupervised pretraining for transformer based speech recognition. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6538–6542. IEEE, 2021.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *ArXiv*, abs/1911.03437, 2020.

David Josephs, Carson Drake, Andrew M Heroy, and John Santerre. semg gesture recognition with a simple model of attention. *Machine Learning for Health*, pp. 126–138, 2020.

John M. Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Zídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andy Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David A. Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583 – 589, 2021.

Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, and Davide Testuggine. Supervised multimodal bitransformers for classifying images and text. *ArXiv*, abs/1909.02950, 2019.

Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *ICML*, 2021.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *ArXiv*, abs/2001.04451, 2020.

Nicolas Boulanger Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv: Learning*, 2012.

Feng Li, Hao Zhang, Hu-Sheng Xu, Siyi Liu, Lei Zhang, Lionel M. Ni, and Heung yeung Shum. Mask dino: Towards a unified transformer-based framework for object detection and segmentation. *ArXiv*, abs/2206.02777, 2022.

Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-Tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. *Proceedings of Machine Learning and Systems*, 2:230–246, 2020a.

Shuang Li, Binhui Xie, Jiashu Wu, Ying Zhao, Chi Harold Liu, and Zhengming Ding. Simultaneous semantic alignment network for heterogeneous domain adaptation. In *Proceedings of the 28th ACM international conference on multimedia*, pp. 3866–3874, 2020b.

Yaoyiran Li, E. Ponti, Ivan Vulic, and Anna Korhonen. Emergent communication pretraining for few-shot machine translation. In *COLING*, 2020c.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *Proceedings of the 9th International Conference on Learning Representations*, 2021a.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021b.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019a.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021a.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys (CSUR)*, 2022.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019b.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. *ArXiv*, abs/2111.09883, 2021b.

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. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021c.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, 2019.

Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. Unified-io: A unified model for vision, language, and multi-modal tasks. *ArXiv*, abs/2206.08916, 2022a.

Kevin Lu, Aditya Grover, P. Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *ArXiv*, abs/2103.05247, 2021.

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Frozen pretrained transformers as universal computation engines. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36 (7):7628–7636, Jun. 2022b.

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

Ofir Pele and Michael Werman. Fast and robust earth mover's distances. *2009 IEEE 12th International Conference on Computer Vision*, pp. 460–467, 2009.

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*, pp. 1406–1415, 2019.

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 *ICML*, 2021.

Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John F. Canny, P. Abbeel, and Yun S. Song. Evaluating protein transfer learning with tape. *bioRxiv*, 2019.

Esteban Real, Chen Liang, David R. So, and Quoc V. Le. Automl-zero: Evolving machine learning algorithms from scratch. In *ICML*, 2020.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley D. Edwards, Nicolas Manfred Otto Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *ArXiv*, abs/2205.06175, 2022.

Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning? *ArXiv*, abs/2201.12122, 2022.

Nicholas Carl Roberts, Mikhail Khodak, Tri Dao, Liam Li, Christopher Re, and Ameet Talwalkar. Rethinking neural operations for diverse tasks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

Junhong Shen, Mikhail Khodak, and Ameet Talwalkar. Efficient architecture search for diverse tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1874–1883, 2016.

Shuhan Tan, Xingchao Peng, and Kate Saenko. Class-imbalanced domain adaptation: An empirical odyssey. In *ECCV Workshops*, 2020.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *ArXiv*, abs/2011.04006, 2021.

Renbo Tu, Nicholas Roberts, Mikhail Khodak, Junhong Shen, Frederic Sala, and Ameet Talwalkar. NAS-bench-360: Benchmarking neural architecture search on diverse tasks. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2022.

Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312: 135–153, 2018.

Yixuan Wei, Han Hu, Zhenda Xie, Zheng Zhang, Yue Cao, Jianmin Bao, Dong Chen, and Baining Guo. Contrastive learning rivals masked image modeling in fine-tuning via feature distillation. *ArXiv*, abs/2205.14141, 2022.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

Yuguang Yan, Wen Li, Hanrui Wu, Huaqing Min, Mingkui Tan, and Qingyao Wu. Semi-supervised optimal transport for heterogeneous domain adaptation. In *IJCAI*, 2018.

Yuan Yao, Yu Zhang, Xutao Li, and Yunming Ye. Heterogeneous domain adaptation via soft transfer network. In *Proceedings of the 27th ACM international conference on multimedia*, pp. 1578–1586, 2019.

Keming Zhang and Joshua S. Bloom. deepcr: Cosmic ray rejection with deep learning. *The Astrophysical Journal*, 889(1):24, 2020.

Jian Zhou and Olga G. Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature Methods*, 12:931–934, 2015.

A APPENDIX

A.1 EMBEDDING LEARNING WITH OPTIMAL TRANSPORT DATASET DISTANCE

A.1.1 LITERATURE REVIEW

Due to the limited space, we do not give a full review of the optimal transport dataset distance (OTDD) (Alvarez-Melis & Fusi, 2020) in the main text. Here, we briefly recall the optimal transport (OT) distance and explain OTDD in detail.

Consider a complete and separable metric space $\mathcal{X}$ and let $\mathcal{P}(\mathcal{X})$ be the set of probability measures on $\mathcal{X}$. For $\alpha, \beta \in \mathcal{P}(\mathcal{X})$, let $\Pi(\alpha, \beta)$ be the set of joint probability distributions on $\mathcal{X} \times \mathcal{X}$ with marginals α and β in the first and second dimensions respectively. Then given a cost function $c(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$, the classic OT distance with cost c is defined by:

$$\text{OT}_c(\alpha, \beta) := \min_{\pi \in \Pi(\alpha, \beta)} \int_{\mathcal{X} \times \mathcal{X}} c(x, y) d\pi(x, y). \tag{2}$$

When $\mathcal{X}$ is equipped with a metric $d_{\mathcal{X}}$, we can use $c(x, y) = d_X(x, y)^p$ for some $p \geq 1$ and obtain the p-Wasserstein distance, $W_p(\alpha, \beta) := (\text{OT}_{d_{\mathcal{X}}^p}(\alpha, \beta))^{\frac{1}{p}}$.

Now consider the case of finite datasets with features in $\mathcal{X}$ and labels in a finite set $\mathcal{Y}$. Each dataset can be considered a discrete distribution in $\mathcal{P}(\mathcal{X} \times \mathcal{Y})$. To define a distance between datasets, a natural approach is to define an appropriate cost function on $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$ and consider the optimal transport distance. Indeed, for any metric $d_{\mathcal{Y}}$ on $\mathcal{Y}$ and any $p \geq 1$, $\mathcal{Z}$ can be made a complete and separable metric space with metric

$$d_{\mathcal{Z}}((x, y), (x', y')) = (d_{\mathcal{X}}(x, x')^p + d_{\mathcal{Y}}(y, y')^p)^{\frac{1}{p}} \tag{3}$$

It is usually not clear how to define a natural distance metric in $\mathcal{Y}$, so instead we proceed by representing each class $y \in \mathcal{Y}$ by $P(\mathcal{X}|\mathcal{Y} = y)$, the conditional distribution of features $\mathcal{X}$ given $\mathcal{Y} = y$. More specifically, for a dataset $\mathcal{D} \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$, denote this map from classes to conditional distributions by $F(\mathcal{D}, \cdot) : \mathcal{Y} \to \mathcal{P}(\mathcal{X})$. Then we can transform any dataset over $\mathcal{X} \times \mathcal{Y}$ into one over $\mathcal{X} \times \mathcal{P}(\mathcal{X})$ via $G(\mathcal{D}) := (\text{proj}_{\mathcal{X}}, F(\mathcal{D}, \text{proj}_Y))$.

As discussed above, W_p is a natural notion of distance in $\mathcal{P}(\mathcal{X})$, so by substituting $\mathcal{Y} \mapsto \mathcal{P}(\mathcal{X})$ and $d_{\mathcal{Y}} \mapsto W_p$ in Equation 3, we can define the (p-)optimal transport dataset distance between datasets $\mathcal{D}_A$ and $\mathcal{D}_B$ by

$$\text{OTDD}(\mathcal{D}_A, \mathcal{D}_B) := \text{OT}_{(d_{\mathcal{X}}^p \times W_p^p)^{\frac{1}{p}}}(G(\mathcal{D}_A), G(\mathcal{D}_B)) \tag{4}$$

A.1.2 COMPUTATIONAL CONSIDERATIONS

As we aim for a practical fine-tuning workflow, computational cost is a crucial concern. While Alvarez-Melis & Fusi (2020) proposed two variants of OTDD—the exact one and a Gaussian approximation, we observe from our experiments that optimizing the exact OTDD leads to better performance. In the following, we will focus on analyzing the computational cost of the exact OTDD.

Given datasets with D-dimensional feature vectors, estimating vanilla OT distances can be computationally expensive and has a worst-case complexity of $O(D^3 \log D)$ (Pele & Werman, 2009). However, adding an entropy regularization term $\epsilon H(\pi|\alpha \otimes \beta)$ to Equation 2, where H is the relative entropy and ϵ controls the time-accuracy trade-off, can be solved efficiently with the Sinkhorn algorithm (Cuturi, 2013). This reduces OT's empirical complexity to $O(D^2)$ and makes the time cost for computing OTDD manageable for ORCA's workflow.

During implementation of ORCA, we also observed memory issues for computing OTDD using the entire target and source datasets on GPUs. To alleviate this, we propose a class-wise subsampling strategy for approximating OTDD on GPUs (Algorithm 1). In short, we split the K-class target dataset into K datasets based on the labels and compute the class-wise OTDD between each single-class target dataset and the *entire source dataset*. Each class-wise OTDD can be approximated with the average of batch samples similar to how stochastic gradient descent approximates gradient descent. After that, we approximate the OTDD between the target and source datasets using the

Algorithm 1 Efficient approximation of OTDD using class-wise subsampling.

Input: target dataset $\{x^t, y^t\}$, number of target classes K^t, source dataset $S = \{x^s, y^s\}$, subsample size b, subsample round R
for each class $i \in [K^t]$ in the target dataset **do**
  Compute class weight $w_i = \frac{\text{number of target data in class } i}{\text{total number of target data}}$
  Generate data loader D_i consisting of data in class i
end for
for $i \in [K^t]$ **do**
  for $r \in [R]$ **do**
    Subsample b target data points D_{ir} uniformly at random from D_i
    Compute class-wise distance $d_{ir} = OTDD(D_{ir}, S)$
  end for
  Approximate class-wise OTDD by $d_i = \frac{1}{R}\sum_{i=1}^{R} d_{ir}$
end for
Approximate OTDD by $d = \sum_{i=1}^{K^t} w_i \cdot d_i$

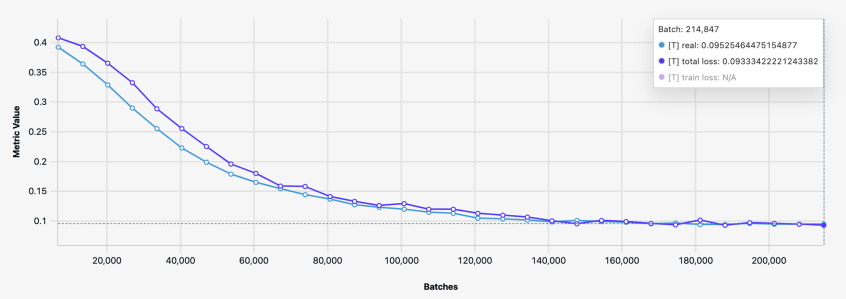

Figure 5: Screenshot of OTDD curves during embedding learning for the task ListOps. x-axis is the number of optimization steps, y-axis represents OTDD (1E-2). We use Algorithm 1 to approximate the exact OTDD as the loss function for optimization on GPU (purple curve). We also track the actual OTDD on CPU (blue curve). We can see that the proposed algorithm works well, which allows us to perform embedding learning efficiently.

weighted sum of the K class-wise OTDDs. To verify that the approximation works empirically, we track the approximated OTDD (computed on GPUs) and the actual OTDD (computed on CPUs) and visualize the loss curves during ORCA's embedder learning process (Figure 5). We can see that the estimated value adheres to the actual value.

Leveraging both the Sinkhorn algorithm and class-wise approximation, the embedder learning process only takes up a small fraction of the total fine-tuning time in practice, as shown in Table 6. Hence, we invest a reasonable time budget but achieve significantly improved cross-domain transfer performance using ORCA.

Table 6: We record the runtime (in hours) of ORCA's embedding learning stage and the fine-tuning stage for each task. Then, we compute the ratio between the two. Averaged across tasks, embedding learning with OTDD only takes about 10% of the time needed for fine-tuning. All experiments are performed on NVIDIA V100 GPUs.

	CIFAR-100	Spherical	Darcy Flow	PSICOV	Cosmic	NinaPro	FSD50K
Embedding	1.6	1.8	0.18	0.35	0.2	0.3	0.21
Fine-tuning	9.2	9.3	0.86	1.1	3.5	1.1	12.5
Embedding/Fine-tuning	17%	19%	20%	31%	5%	27%	2%
	ECG	Satellite	DeepSEA	JSB Chorales	ListOps	Homology	
Embedding	0.6	0.26	0.14	0.1	0.2	0.16	
Fine-tuning	23.1	37.5	14.8	0.13	17.8	6.0	
Embedding/Fine-tuning	3%	1%	1%	76%	1%	2%	

A.2 INFORMATION ABOUT EVALUATION TASKS

Table 7: Summary about each evaluation task and the hand-designed expert models used. The top 10 rows are for the 10 datasets in NAS-Bench-360 (Tu et al., 2022).

Task name	# Data	Data dim.	Type	License	Learning objective	Expert arch.
CIFAR-100	60K	2D	Point	CC BY 4.0	Classify natural images into 100 classes	DenseNet-BC (Huang et al., 2017)
Spherical	60K	2D	Point	CC BY-SA	Classify spherically projected images into 100 classes	S2CN (Cohen et al., 2018)
NinaPro	3956	2D	Point	CC BY-ND	Classify sEMG signals into 18 classes corresponding to hand gestures	Attention Model (Josephs et al., 2020)
FSD50K	51K	2D	Point (multi-label)	CC BY 4.0	Classify sound events in log-mel spectrograms with 200 labels	VGG (Fonseca et al., 2021)
Darcy Flow	1100	2D	Dense	MIT	Predict the final state of a fluid from its initial conditions	FNO Li et al. (2021a)
PSICOV	3606	2D	Dense	GPL	Predict pairwise distances between residuals from 2D protein sequence features	DEEPCON (Adhikari, 2019)
Cosmic	5250	2D	Dense	Open License	Predict propablistic maps to identify cosmic rays in telescope images	deepCR-mask (Zhang & Bloom, 2020)
ECG	330K	1D	Point	ODC-BY 1.0	Detect atrial cardiac disease from a ECG recording (4 classes)	ResNet-1D (Hong et al., 2020)
Satellite	1M	1D	Point	GPL 3.0	Classify satellite image pixels' time series into 24 land cover types	ROCKET (Dempster et al., 2020)
DeepSEA	250K	1D	Point (multi-label)	CC BY 4.0	Predict chromatin states and binding states of RNA sequences (36 classes)	DeepSEA (Zhou & Troyanskaya, 2015)
JSB Chorales	229	1D	Dense	CC BY-SA	Predict the next note from sheet music	Dilated TCN (Bai et al., 2018)
ListOps	55K	1D	Point	MIT	Model hierarchically structured data in a longcontext scenario	Reformer (Kitaev et al., 2020)
Homology	12K	1D	Point		Predict the fold for a protein	LSTM (Rao et al., 2019)

A.3 EXPERIMENT DETAILS

Below, we summarize details for implementing ORCA and evaluating it on the selected 13 tasks. The code and configuration file for reproducing each experiment can be found in the supplementary material. We will also release ORCA's best checkpoint for each task later.

A.3.1 PRETRAINED MODELS

We evaluated ORCA with two pretrained models in our experiments. In Table 2, for all 2D tasks including CIFAR-100, Spherical, Darcy Flow, PSICOV, Cosmic, NinaPro, and FSD50K, we use the following model. As Swin has a pretrained resolution, we reshape the inputs for our tasks to the resolution before feeding them into the model.

Name	Pretrain	Resolution	Num Params	FLOPS	FPS
Swin-base (Liu et al., 2021c)	ImageNet-22K	224×224	88M	15.4G	278

For all 1D tasks including ECG, Satellite, DeepSEA, JSB Chorales, ListOps,and Homology, we use the following model:

Name	Pretrain	Num Params	FLOPS
RoBERTa-base (Liu et al., 2019b)	Five English-language corpora	125M	1.64E20

We use the Hugging Face transformers library Wolf et al. (2019) to implement the pretrained models.

A.3.2 TASK DATA PREPARATION

For all the NAS-Bench-360 tasks, each dataset is preprocessed and split using the script available on `https://github.com/rtu715/NAS-Bench-360`, with the training set being used for hyperparameter tuning, embedding learning, and fine-tuning. We obtain the data

processing script for JSB data from https://github.com/locuslab/TCN, for ListOps from https://github.com/kzl/universal-computation, and for Homology from https://github.com/songlab-cal/tape.

A.3.3 HYPERPARAMETER TUNING

As ORCA is both task-agnostic and model-agnostic, it can be applied to fine-tuning a variety of pretrained transformers on drastically different end tasks with distinct datasets. Hence, it is hard to define one set of fine-tuning hyperparameters for all (model, task) pairs. At the same time, optimizing large-scale pretrained transformers can be challenging due to their large model sizes, as the downstream performance depends largely on the hyperparameters used. For instance, using a large learning rate can distort pretrained weights and lead to catastrophic forgetting. Therefore, in our experiments, given a (model, task) pair, we first apply hyperparameter tuning using the Asynchronous Successive Halving Algorithm (ASHA) (Li et al., 2020a) to the *standard fine-tuning setting* (i.e., after initializing the embedder and predictor architectures, directly updating all model weights to minimize the task loss) to identify a proper training configuration. Then, we use the same set of hyperparameters found for all our experiments for the particular (model, task) combination. Note that even though we did not explicitly state this in the main text, the hyperparameter tuning stage can be directly integrated into the ORCA workflow between stage 1 and stage 2. In this sense, ORCA is still an automated cross-modal transfer workflow that works for diverse tasks and different pretrained models.

The configuration space for ASHA is as follows:

- Batch size: 32, 128, 512, 1024 for Swin; 16, 56, 256, 512 for RoBERTa
- Optimizer: SGD, Adam, AdamW
- Learning rate: 1E-2, 1E-3, 1E-4, 1E-5, 1E-6
- Weight decay: 1E-2, 1E-3, 1E-4, 1E-5, 1E-6

Note that to fit each experiment on a single GPU, we set a fixed batch size (32 for Swin and 16 for Roberta) and vary the gradient accumulation step instead of actually varying the batch size, but the effect is the same.

A.3.4 ORCA ONLY: EMBEDDING LEARNING WITH OTDD

After initializing the embedder architecture for each task, we train it to minimize the OTDD between the embedded target features and embedded source features.

For source datasets, we use CIFAR-10 for Swin and CONLL-2003 for RoBERTa. We sample 5000 data points to compute OTDD. In practice, we can pass the source data through the pretrained embedder once and save all the embedded features, so we don't have to pay the cost of obtaining the source features each time we fine-tune a new model.

For classification tasks, we directly use the labels provided by the end task to compute OTDD. For dense tasks, we perform K-Means clustering on the target data to obtain pseudolabels for OTDD computation. The number of clusters is set to the number of classes of the source dataset, e.g., 10 for 2D tasks that use CIFAR-10 as the source dataset.

To compute the embedding learning objective, we use the OTDD implementation of the original paper provided here: https://github.com/microsoft/otdd. As for the hyperparameters, we use the batch size, learning rate, optimizer, and weight decay obtained from A.3.3. The others are fixed across different tasks:

- Embedding learning epochs: 60
- Learning rate scheduler: decay by 0.2 every 20 epochs

A.3.5 FINE-TUNING

Besides the searched hyperparameters, we also fix the following hyperparameters for fine-tuning.

- Fine-tuning epochs: 100 for Swin tasks, 60 for RoBERTa tasks

- Learning rate scheduler: we use the linear decay with min_lr = 0 and 5 warmup epochs

When fine-tuning is finished, we evaluate the performance of all models following the NAS-Bench-360 protocol. We first report results of the target metric for each task by running the model of the *last* epoch on the test data. Then, we report aggregate results via performance profiles (Dolan & Moré, 2002), a technique that considers both outliers and small performance differences to compare methods across multiple tasks robustly. In such plots, each curve represents one method. The τ on the x-axis denotes the fraction of tasks on which a method is no worse than a τ-factor from the best. The performance profile for our experiments is shown in Figure 2.

A.3.6 ADDITIONAL RESULTS

In Table 3, we compare with the FPT setting, which only fine-tunes the layer norms of the pretrained transformer models. As we have shown already, the downstream performance of fine-tuning only a subset of the parameters is less competitive than fine-tuning all parameters. Below, we show that the time saved for updating only layer norms is also not that significant. Therefore, we suggest performing full fine-tuning when time and computational resources allow.

Table 8: We record the total runtime (in hours) for four settings: ORCA with full fine-tuning, ORCA with tuning layer norms, full fine-tuning (without embedding learning), and fine-tuning layer norms (FPT). We can see that tuning the layer norms does not bring significant benefit in terms of reducing the model development time, but it sacrifices the downstream performance of the resulting models.

	CIFAR-100	Spherical	Darcy Flow	PSICOV	Cosmic	NinaPro	FSD50K
ORCA	10.8	11.1	1.04	1.45	3.7	1.4	12.71
ORCA (layernorm)	8.7	8.9	0.76	1.15	3.5	1.0	8.96
Fine-tuning	9.2	9.3	0.86	1.1	3.2	1.1	12.5
Fine-tuning (layernorm)	7.1	7.1	0.58	0.8	3.0	0.7	8.75
	ECG	Satellite	DeepSEA	JSB Chorales	ListOps	Homology	
ORCA	23.7	37.76	14.94	0.23	18.0	6.16	
ORCA (layernorm)	18.0	25.56	11.24	0.2	13.2	4.56	
Fine-tuning	23.1	37.5	14.8	0.13	17.8	6.0	
Fine-tuning (layernorm)	17.4	25.3	11.1	0.1	13.0	4.4	