What explains the success of cross-modal fine-tuning with ORCA?

Paloma García-de-Herreros[*1] **Vagrant Gautam**[*1] **Philipp Slusallek**[1,2]
Dietrich Klakow[1] **Marius Mosbach**[3,4]

[1]Saarland University [2]DFKI [3]McGill University [4]Mila – Quebec AI Institute

{pgherreros,vgautam}@lsv.uni-saarland.de

Abstract

ORCA (Shen et al., 2023) is a recent technique for cross-modal fine-tuning, i.e., applying pre-trained transformer models to modalities beyond their training data. The technique consists primarily of training an embedder and fine-tuning the embedder and model. Despite its high performance on a variety of downstream tasks, we do not understand precisely how each of these components contribute to ORCA's success. Therefore, we run a series of ablations and find that embedder training does not help 2D tasks at all, contrary to what the original paper posits. In 1D tasks, some amount of embedder training is necessary but more is not better. In 4 out of 6 datasets we experiment with, it is model fine-tuning that makes the biggest difference. Through our ablations and baselines, we contribute a better understanding of the individual components of ORCA.

1 Introduction

Modern AI is based on a pipeline of pre-training general-purpose models on vast amounts of data and then adapting them to specific tasks. Examples across natural language processing (NLP) and computer vision (CV) typically focus on within-modality adaptation across, e.g., tasks or domains, but there is also a recent line of work that looks at leveraging pre-trained models *across* modalities, e.g., Frozen Pretrained Transformers (FPT) (Lu et al., 2021), ORCA (Shen et al., 2023), OmniPred (Song et al., 2024), Unified PDE Solver (UPS) (Shen et al., 2024), *inter alia*.

ORCA is a recent example of a method for cross-modal fine-tuning (Shen et al., 2023). It consists of a three-phase pipeline, shown in Figure 1. First, a pre-trained transformer model is chosen, and a custom embedder and predictor are created to support new tasks with any input and output dimensions. Second, a within-modality proxy dataset is chosen. The embedder is trained to minimize the distance between the target dataset and this proxy dataset, in order to map the target dataset into the embedding space of the model. Finally, all three components are fine-tuned on data from the target task.

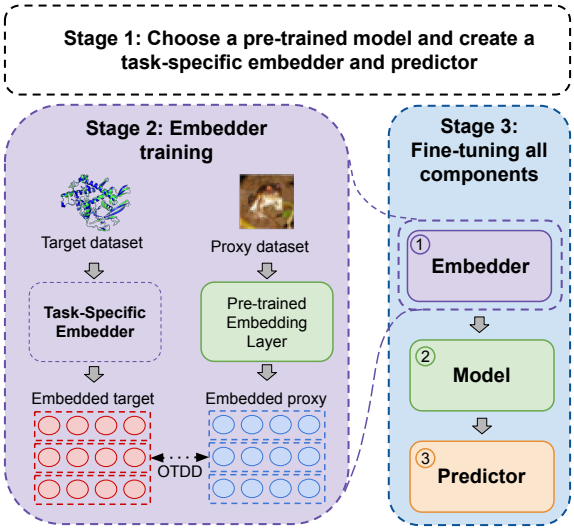


Figure 1: The ORCA pipeline. Stage 2 involves training the task-specific embedder. Stage 3 fine-tunes the embedder, the pre-trained encoder, and the predictor.

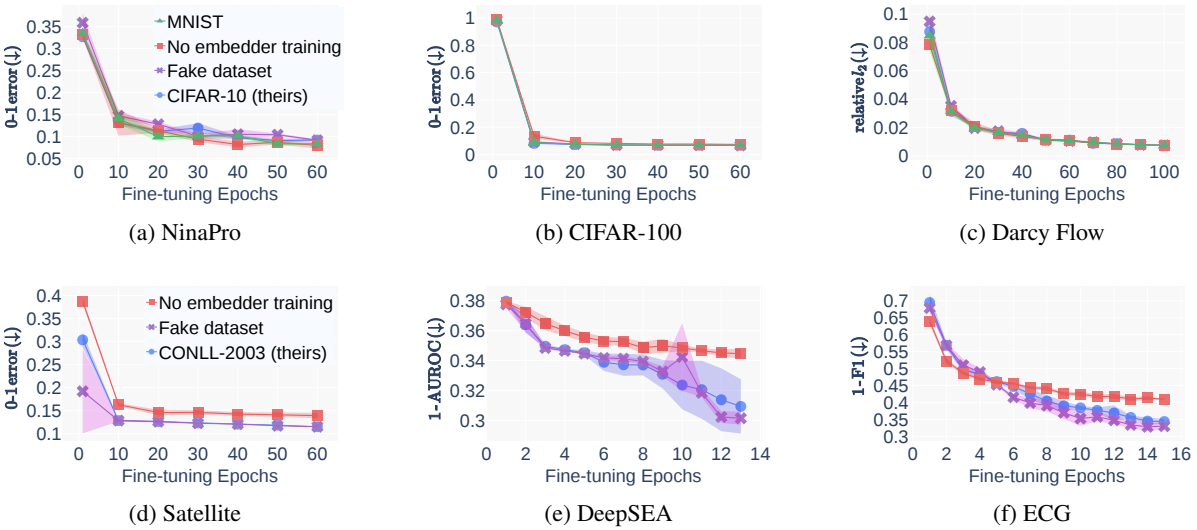
According to Shen et al. (2023), embedder training is the reason for ORCA's success. We expand on their ablations to better understand the contributions of ORCA's individual components, focusing on ablating the second and third stages of the pipeline. Our specific research questions are:

1. How does the choice of proxy dataset affect performance? (§3)
2. Does doing (more) embedder training improve performance? (§4)
3. What do the embedder and the pre-trained model contribute individually? (§5)
4. How much pre-training is necessary for cross-modal transfer? (§6)

By disentangling the contributions of embedder training and model fine-tuning, our results provide a more nuanced perspective on the success of cross-modal fine-tuning with ORCA. Additionally, our findings highlight the importance of strong baselines and careful ablations when making claims about *why* a method works.

[*] Equal contribution.

Proceedings of the Fifth Workshop on Insights from Negative Results in NLP, pages 8–16



(a) NinaPro (b) CIFAR-100 (c) Darcy Flow

(d) Satellite (e) DeepSEA (f) ECG

Figure 2: Per-epoch fine-tuning performance (↓) on 2D tasks (above) and 1D tasks (below) when the embedder is trained with different proxy datasets or not trained at all, i.e., naive fine-tuning.

2 Experimental setup

Unless otherwise specified, we follow the ORCA paper in using RoBERTa-base (Liu et al., 2019) and Swin-base (Liu et al., 2021) as the pre-trained transformers, a convolutional architecture for the embedder, and a linear transformation for the predictor (see Appendix C for details). We also use optimal transport dataset distance (OTDD; Alvarez-Melis and Fusi, 2020) as the loss function during embedder training. All our experiments use their publicly available code.[1] For training, we use the same hyperparameters as they do, except for the batch size when training on Satellite (64) and ECG (32) data. We evaluate on six target datasets that appear in the original paper, chosen to represent all pairs of dimensions and types, and we experiment with various proxy datasets. Dataset details are shown in Appendix B.

Target datasets. We select three 2D datasets (NinaPro, CIFAR-100, and Darcy Flow) and three 1D datasets (Satellite, DeepSEA, and ECG) from the NAS-Bench-360 benchmark (Tu et al., 2022). 2D and 1D refer to the input being either a matrix (2 dimensions) or a sequence (1 dimension).

Proxy datasets. The original paper uses CIFAR-10 (Krizhevsky, 2009) as the proxy dataset for all 2D tasks, and CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) for all 1D tasks. We experiment with additional proxy datasets to analyze their influence on overall performance.

For the 2D tasks, we compare to two other image datasets that maintain the same number of classes: MNIST (Deng, 2012), a different image dataset, and Fakedata[2], a dataset of randomly classified white noise images (Paszke et al., 2019). For the 1D tasks, we compare to a custom-created fake dataset classifying randomly generated language feature vectors into the same number of classes as CoNLL.

3 How does the choice of proxy dataset affect performance?

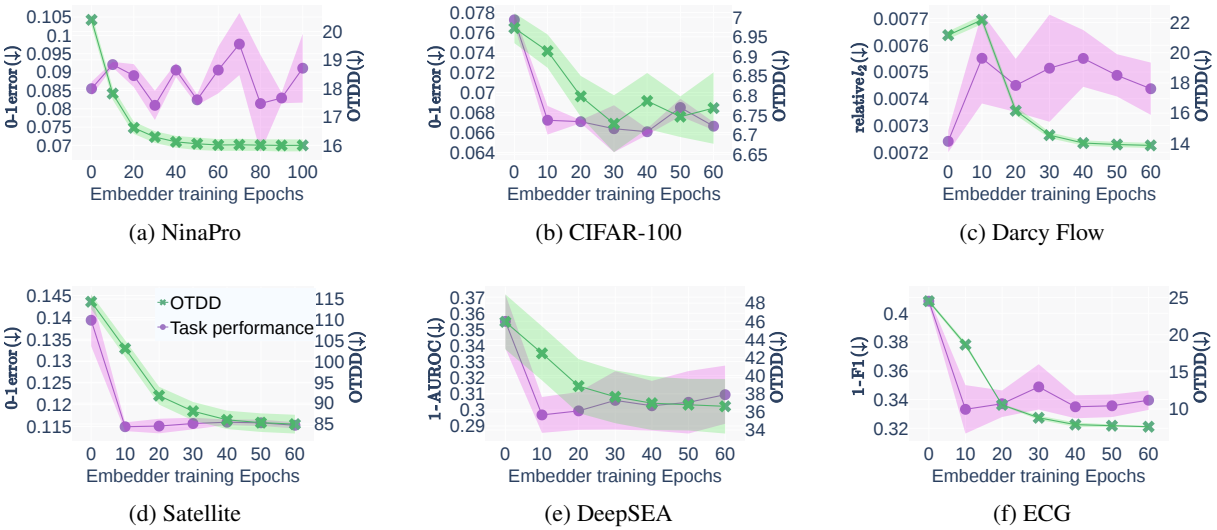
In this section, we experiment with the choice of proxy dataset for the tasks. As a baseline, we compare to just fine-tuning the embedder, model and predictor, without training the embedder first.

As Figure 2 shows, all fine-tuning curves for the 2D datasets (first row) overlap, indicating that the choice of proxy dataset is not important. Even fake data as a proxy dataset results in the same performance. Similarly, for the 1D tasks (second row), there is no real difference between using CoNLL and fake embeddings. Together, this shows that **the choice of proxy dataset for embedder training does not matter for ORCA to work**.

Comparing to a naive fine-tuning baseline allows us to evaluate the claim that "ORCA consistently outperforms naive fine-tuning" (Shen et al., 2023). We find that **embedder training does play a role in the 1D tasks, but does not matter for 2D tasks**, even in the early stages of fine-tuning.

[1] https://github.com/sjunhongshen/ORCA/

[2] From torchvision.datasets.



(a) NinaPro (b) CIFAR-100 (c) Darcy Flow

(d) Satellite (e) DeepSEA (f) ECG

Figure 3: Per-epoch embedder training comparing OTDD (↓) (metric minimized during this stage) to downstream task performance (↓).

4 (More) embedder training is not the secret to ORCA's success

The previous results motivate us to more closely examine the role of embedder training in ORCA. In this stage, the OTDD metric is used to quantify the distance between the proxy and target embeddings. The authors minimize OTDD, claiming that "as the dataset distance decreases, the fine-tuning accuracy increases" (Shen et al., 2023).

However, when we examine the relationship between OTDD and downstream task performance, we find that **embedder training is unnecessary in two out of six tasks** (Figures 3a and 3c). For the remaining four tasks, **training the embedder more can even lead to worse task performance**.

As this section and the previous one show that embedder training does not affect final performance on the 2D tasks, we focus on the 1D tasks for our remaining experiments.

5 Which components of ORCA are really necessary?

To better understand how the fine-tuning phase affects the multiple components of ORCA, we experiment with freezing different parts of the pipeline: the embedder, the pre-trained model, or both. We compare our results with the original setup.

Row 1 of Figure 4 shows the results of freezing both the embedder and the pre-trained model, and only fine-tuning the predictor. Across all datasets, the frozen versions perform much worse than the original setup, regardless of embedder training. This indicates that these datasets are not simple enough to be solved by training a simple predictor.

In row 2, we freeze only the pre-trained model, but fine-tune the embedder and the predictor. These frozen versions also perform much worse than the original setup, indicating that **fine-tuning the pre-trained model is a critical component of ORCA**, regardless of dataset and embedder training.

Finally, in row 3, we only freeze the embedder, allowing the fine-tuning stage to affect both the model and the predictor. As we already saw in Figure 2, training the embedder is important across all three datasets. However, once this training is done, even if it is frozen, adapting the pre-trained model is sufficient for good task performance. This shows that **while training the embedder is important for ORCA's success on these datasets, it need not be fine-tuned beyond that**.

6 Pre-training is not always necessary

Our previous results show that fine-tuning the model is necessary for good downstream task performance, but they do not show whether using *pre-trained* models is necessary for this. To answer this question, we use RoBERTa models pre-trained on different amounts of English data. Specifically, we compare the original RoBERTa-base model to a randomly initialized model with no training data, along with three variants trained on less data (Warstadt et al., 2020), shown in Appendix F.

Figure 5 shows that performance varies widely

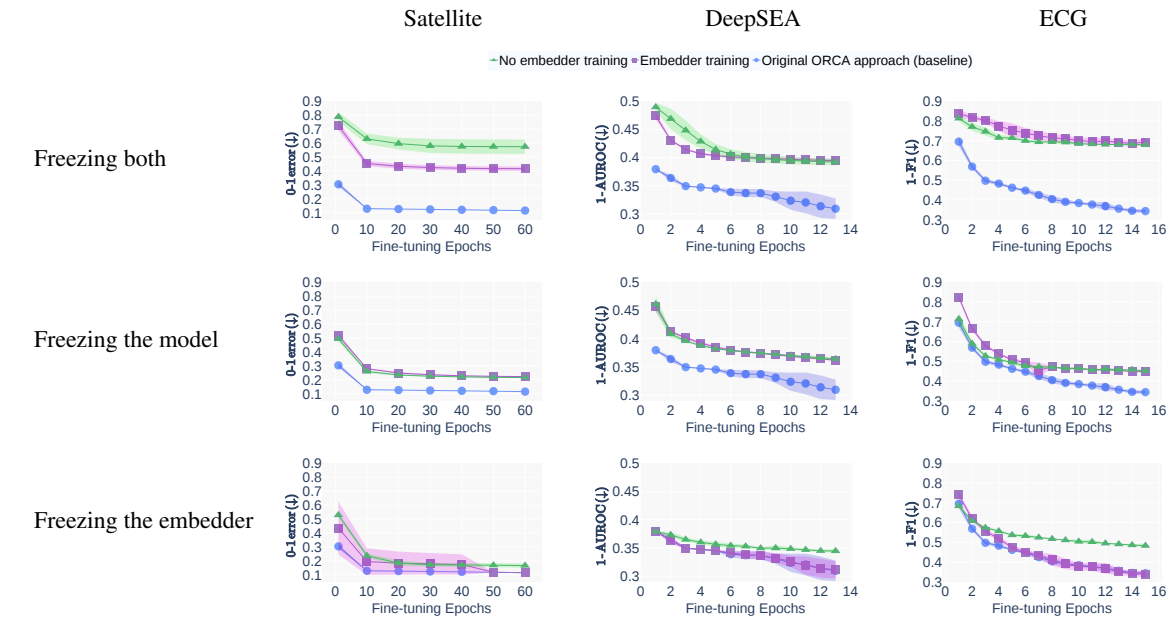
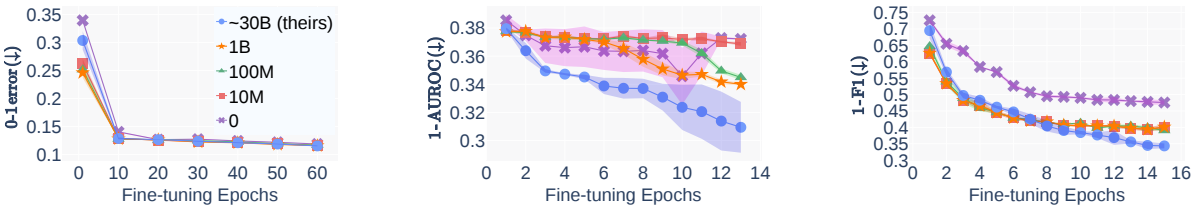


Figure 4: Performance (↓) when freezing both the embedder and model (top row), just the model (middle row) or just the embedder (bottom row), before full fine-tuning. We also evaluate the impact of training (purple squares) vs. not training (green triangles) the embedder before freezing. All ablations are compared to the original ORCA approach without freezing (blue circles).



(a) Satellite (b) DeepSEA (c) ECG

Figure 5: Effect of different amounts of pre-training data on downstream performance (↓).

depending on the dataset. For Satellite, all models perform the same, showing that the task is simple enough to be solved even without pre-training. With DeepSEA and ECG, on the other hand, pre-training data on the scale of 30B tokens results in clearly better performance. These results highlight the importance of comparing to a no pre-training baseline, for ORCA—and indeed all cross-modal fine-tuning work—to ensure that pre-training is actually necessary for the success of the method.

Until the 30B data scale, however, DeepSEA performance remains within the variance of simply fine-tuning a randomly-initialized model, whereas ECG does benefit from even a small amount of pre-training. This shows that even for non-trivial tasks, **the amount of pre-training has a noticeable effect only at certain scales**.

7 Conclusion

We perform a series of ablations to investigate how the different components of ORCA, a recently-proposed method for cross-modal fine-tuning, affect its performance. Contrary to the original results, we find that embedder training does not help 2D tasks at all, compared to just fine-tuning without training the embeddder. In 1D tasks, some amount of embedder training is necessary, but unlike the claim in the original paper, more embedder training can even hurt performance on the target task. When we freeze various components of the ORCA pipeline, we find that fine-tuning the model is crucial for good task performance. Finally, we find that for one of the 1D tasks, using a pre-trained model is actually not necessary, indicating the importance of no pre-training baselines in evaluations of cross-modal transfer.

References

Krizhevsky Alex. 2009. Learning multiple layers of features from tiny images. *https://www. cs. toronto. edu/kriz/learning-features-2009-TR. pdf*.

David Alvarez-Melis and Nicolo Fusi. 2020. Geometric dataset distances via optimal transport. In *Advances in Neural Information Processing Systems*, volume 33, pages 21428–21439. Curran Associates, Inc.

Manfredo Atzori, Arjan Gijsberts, Simone Heynen, Anne-Gabrielle Mittaz Hager, Olivier Deriaz, Patrick Van Der Smagt, Claudio Castellini, Barbara Caputo, and Henning Müller. 2012. Building the ninapro database: A resource for the biorobotics community. In *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 1258–1265. IEEE.

Gari D Clifford, Chengyu Liu, Benjamin Moody, H Lehman Li-wei, Ikaro Silva, Qiao Li, AE Johnson, and Roger G Mark. 2017. Af classification from a short single lead ecg recording: The physionet/computing in cardiology challenge 2017. In *2017 Computing in Cardiology (CinC)*, pages 1–4. IEEE.

Li Deng. 2012. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142.

EA Feingold, PJ Good, MS Guyer, S Kamholz, L Liefer, K Wetterstrand, FS Collins, TR Gingeras, D Kampa, EA Sekinger, et al. 2004. The encode (encyclopedia of dna elements) project. *Science*, 306(5696):636–640.

Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. Technical report.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2020. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

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

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022.

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. 2021. Pretrained transformers as universal computation engines. *arXiv preprint*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

François Petitjean, Jordi Inglada, and Pierre Gancarski. 2012. Satellite image time series analysis under time warping. *IEEE Transactions on Geoscience and Remote Sensing*, 50(8):3081–3095.

Junhong Shen, Liam Li, Lucio M. Dery, Corey Staten, Mikhail Khodak, Graham Neubig, and Ameet Talwalkar. 2023. Cross-modal fine-tuning: Align then refine. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31030–31056. PMLR.

Junhong Shen, Tanya Marwah, and Ameet Talwalkar. 2024. Ups: Towards foundation models for pde solving via cross-modal adaptation. *arXiv preprint arXiv:2403.07187*.

Shu Shen, Kang Gu, Xin-Rong Chen, Ming Yang, and Ru-Chuan Wang. 2019. Movements classification of multi-channel semg based on cnn and stacking ensemble learning. *IEEE Access*, 7:137489–137500.

Xingyou Song, Oscar Li, Chansoo Lee, Daiyi Peng, Sagi Perel, Yutian Chen, et al. 2024. OmniPred: Language models as universal regressors. *arXiv preprint*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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

Alex Warstadt, Yian Zhang, Xiaocheng Li, Haokun Liu, and Samuel R. Bowman. 2020. Learning which features matter: RoBERTa acquires a preference for linguistic generalizations (eventually). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 217–235, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural

language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

A Limitations

Choice of datasets. We only experiment with three 2D datasets and three 1D datasets, and we do not consider the experiments from the original paper on tabular data, where our findings may not hold. Additionally, due to the widely varying patterns we find in our results, we believe that this is not sufficient for our findings to generalize beyond these specific datasets to the modalities that they represent. This points to a limitation of cross-modal fine-tuning work in general, which would benefit from a larger set of datasets, and in particular, more challenging tasks, as we find that the Satellite dataset is very simple.

Choice of pre-trained models. Our experiments focus on 1D tasks, for which we only experiment with encoder-only architectures (specifically RoBERTa-type models) even though other encoder-only models and even other architectures (e.g., encoder-decoder and decoder-only models) could also be used. We caution against claims about generalization of our results for these tasks to pre-trained models beyond just RoBERTa.

Ablating stage one. Our experiments focus on stages two and three of the ORCA pipeline, but stage one, i.e., the creation of the task-specific embedder and predictor, is not something we vary. In Shen et al. (2023) and in our work, the task-specific embedder consists of a convolutional layer, a layer norm, and a positional embedding, and the predictor consists of a linear projection. It would be interesting to test a much simpler method of converting dimensions in the embedder than a convolutional architecture, e.g., a linear projection, which we leave to future work.

Evaluating what is being transferred. In Section 5, we show that pre-training is necessary for some cross-modal transfer, but we still do not know exactly what is being transferred. The cross-modal transfer literature posits that pre-trained knowledge is somehow exploited in downstream tasks, but since we do not know how to quantify "knowledge" in this setting, we cannot make this claim. It is just as plausible that models pre-trained on tokens beyond a certain scale find better, more general solutions that are a good initialization for adapting to a new task. One way to further probe the transfer hypothesis would be by limiting the number of parameters that are allowed to change during fine-tuning, e.g., by using parameter-efficient fine-tuning with LoRA. We leave an exploration of this to future work.

B Dataset details

Table 1 shows the target and original proxy datasets considered, along with their dimension, type, number of classes, and the metric used to measure target task performance. The tasks are classified into two types, taking into account whether the task's output is a singular prediction (point) or multiple predictions (dense). The target datasets are described in more detail below.

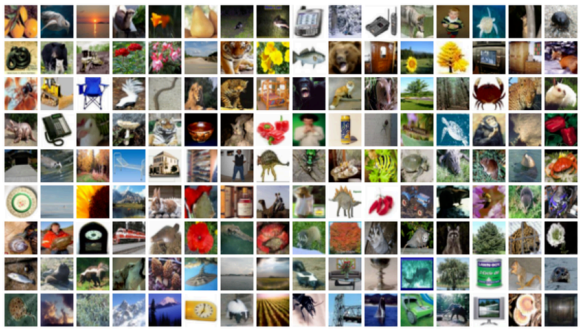


Figure 6: CIFAR-100 examples.

CIFAR-100: Standard Image Classification. (Alex, 2009) The dataset consists of 32x32 color images divided into 100 classes, based on the object represented by the image. Some examples can be seen in Figure 6.

Darcy Flow: Solving Partial Differential Equations (PDEs). (Li et al., 2020) The only regression task considered. Although, for the training stages, the dataset is divided into a total of 10 inferred classes. The dataset consists of 2D grids specifying the initial conditions of a fluid, as an output the same 2D grid on a later time is predicted.

DeepSEA: Predicting Functional Effects From Genetic Sequences. (Feingold et al., 2004) The dataset consists of a collection of genomic profiles to estimate the behavior of chromatin proteins,

Dim.	Target dataset	Type	Metric	# classes	Proxy dataset	# classes
2D	NinaPro	Point	0-1 error (↓)	18		
	CIFAR-100	Point	0-1 error (↓)	100	CIFAR-10	10
	Darcy Flow	Dense	relative l_2 (↓)	10		
1D	Satellite	Point	0-1 error (↓)	24		
	DeepSEA	Point (multi-label)	1 - AUROC (↓)	36	CoNLL-2003[3]	7
	ECG	Point	1 - F1 (↓)	4		

Table 1: Target datasets of each type along with the proxy datasets used for them in ORCA (Shen et al., 2023)

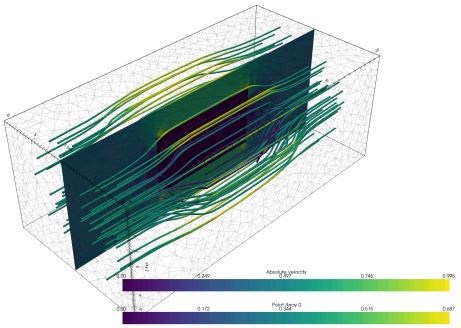


Figure 7: Example from the Darcy Flow dataset.

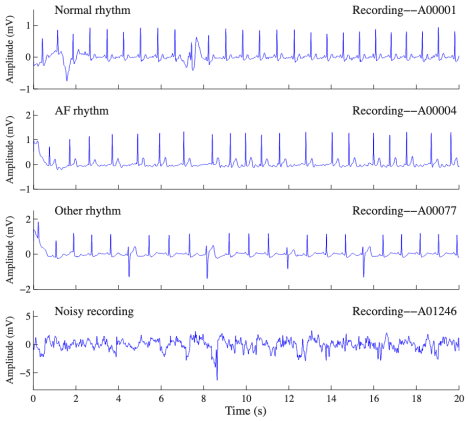


Figure 8: Examples of ECG recordings of the 4 different classes

classifying it into 36 classes.

ECG: Detecting Heart Disease. (Clifford et al., 2017) The dataset is formed by recordings of up to a minute of Electrocardiograms classified into four classes: normal, disease, other, or noisy rhythms. Figure 8 shows an example of each of the classes.

NinaPro: Classifying Electromyography Signals. (Atzori et al., 2012) A subset of NinaPro BD5 is taken, to classify the electromyography (sEMG) signals of a collection of hand movements in 18 classes. Some examples of the movements can be seen in Figure 9.

Satellite: Satellite Image Time Series Analysis. (Petitjean et al., 2012) The dataset consists of satellite image time series (SITS), tracking the land changes over the years, classifying them into 24 land cover types.

[3]We were unable to replicate the exact workflow to create the language features passed to the model, so we used the ones provided in the original ORCA GitHub.

C Embedder and predictor details

As described in Figure 1, in the first stage of the ORCA workflow (Shen et al., 2023), a task-specific embedder and predictor are created to support any combination of input-output dimensions. Throughout all our experiments, we kept the same architectures used in the original paper, which we will explain in this section.

Task-specific Embedding Network The architecture is composed of a convolutional layer with an input channel of the target dataset and an output channel of the dimension of the pre-trained model embedding space. The kernel size and stride can be treated as a hyperparameter, but in all our experiments for the 2D tasks both are set to four and, for the 1D tasks, are computed based on the input and target sequence length. After this, a layer norm and a positional embedder are added to obtain the final representation.

Task-specific Predictor Given the diversity of the tasks considered, two different architectures are implemented depending on the target task type. For

Algorithm 1 Efficient approximation of OTDD using class-wise subsampling from (Shen et al., 2023)

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 $w_i = \frac{number\ of\ target\ data\ in\ class\ i}{total\ number\ of\ target\ data}$
  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 $d = \sum_{i=1}^{K^t} w_i \dot{d}_i$

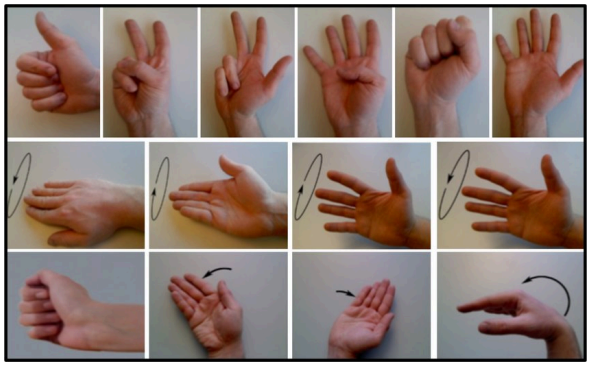


Figure 9: Samples of movements in NinaPro BD5 (Shen et al., 2019), the dataset contains the electromyography signals of the movements.

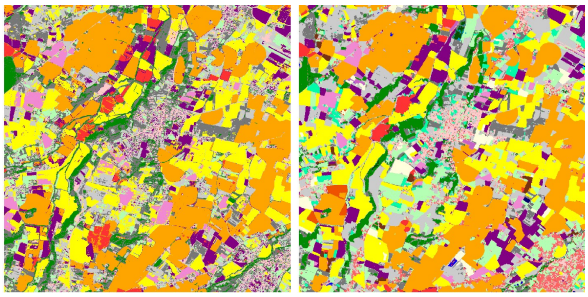


Figure 10: Example of Satellite (Petitjean et al., 2012)

the point tasks, average pooling along the sequence length dimension is applied, to obtain 1D tensors with the same length as the dimension of the pre-trained model embedding space. Then to map to the number of classes of the target dataset, a linear layer is used. For dense tasks, a linear layer is applied to the sequence outputs to adjust the tensor shape. Then, this tensor is molded to the desired output dimension.

D OTDD approximation implementation

Following the original ORCA implementation (Shen et al., 2023), we also used an approximation of OTDD using class-wise subsampling, as described in Algorithm 1.

As described in the original paper, to tackle potential memory issues when computing OTDD, the dimensionality of the feature vectors is reduced by taking the average along the sequence length dimension. On top of that, the target dataset is divided into subsets based on the labels, each of these subsets will be approximated with the average of batch samples (the number of maximum samples taken from each class is determined for every dataset). Then the OTDD between each class representative and a sample of the proxy dataset (5000 samples for CIFAR-10 and 2000 for CONLL 2003) is computed. Finally, the overall OTDD is approximated by the weighted sum of the OTDD of all the classes in the task dataset.

E Experimental Details

We run our experiments using a single 80GB NVIDIA A100 GPU. As in the original paper (Shen et al., 2023), we implemented the base models using the Huggingface Transformers library (Wolf et al., 2020).

F Details on pre-trained RoBERTa models

Table 2 provides information about the amount of training data seen by the different RoBERTa variants released by Warstadt et al. (2020).

Model	Training data
roberta-base	~30B
roberta-base-1B-2	1B
roberta-base-100M-3	100M
roberta-base-10M-3	10M
roberta-base-random	0

Table 2: Models for pre-trained knowledge comparison, and their training data in number of tokens.