

Cross-Modal Fine-Tuning

Align then Refine

Tran Trong Khiem

AI lab training

2024/05/29

1 Introduction

2 ORCA

3 Implement ORCA

4 Enhancing Cross-Modal Fine-Tuning

5 MoNa

6 Proposed

7 Appendix

8 References

Introduction

Transfer learning :

- Reuse of a pre-trained model on a new problem.
- Models can apply what they have learned from large amounts of unlabeled data to downstream tasks.

Existing research:

- focuses on in-modality transfer within these well-studied areas.

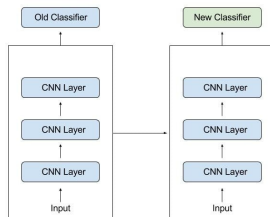


Figure 1: The early and middle layers are used and we only retrain the latter layers

Cross-Modal Fine-Tuning

Problem: Could **cross-modal fine-tuning** have immense impact on **less-studied areas** ?

- could we use pretrained BERT models to tackle genomics tasks, or vision transformers to solve PDEs?

ORCA:

- cross-modal fine-tuning workflow.
- prevent the **distortion of the pretrained weights**.
- exploit the knowledge encoded in the pretrained model.

1 Introduction

2 ORCA

3 Implement ORCA

4 Enhancing Cross-Modal Fine-Tuning

5 MoNa

6 Proposed

7 Appendix

8 References

ORCA work flow

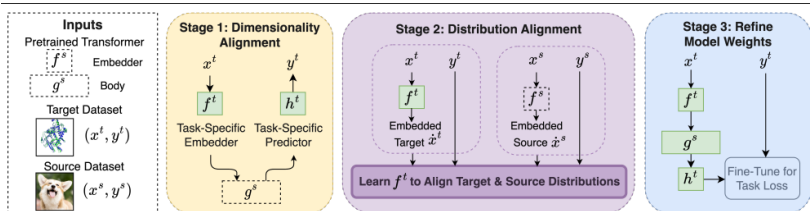


Figure 2: ORCA's three-stage fine-tuning workflow

ORCA:

- **Dimensionality Alignment:**
 - generate task-specific embedder and predictor.
- **Distribution Alignment:**
 - train the the embedding network.
- **Refine model weight:** fine-tuning to minimize the target loss.

Problem setup

Denote:

- A domain \mathcal{D} consists of a feature space \mathcal{X} , a label space \mathcal{Y} , and a joint probability distribution $P(\mathcal{X}, \mathcal{Y})$.
- The target domain \mathcal{D}^t and source (pretraining) domain \mathcal{D}^s
- $\mathcal{X}^t \neq \mathcal{X}^s$, $\mathcal{Y}^t \neq \mathcal{Y}^s$, and $P_t(\mathcal{X}^t, \mathcal{Y}^t) \neq P_s(\mathcal{X}^s, \mathcal{Y}^s)$
- embedder f that transforms input x into a **sequence of features**.
- model body g that applies a series of **pretrained attention layers to the embedded features**.
- predictor h that **generates the outputs with the desired shape**.

Goal:

- Given target data $\{(x_{t_i}, y_{t_i})\}_{i=1}^n$ sampled from a joint distribution P^t in domain \mathcal{D}^t ,
- learn a model m^t that correctly maps each input x^t to its label y^t

Dimensionality Alignment

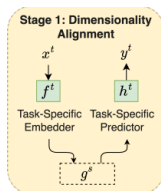


Figure 3: Dimensionality Alignment

Goal: addressing the problem of **dimensionality mismatch**.

Custom Embedding Network:

- The target embedder $f^t : \mathcal{X} \rightarrow \dot{\mathcal{X}}$ with $\dot{\mathcal{X}}$ is feature space.
- f^t is composed of a convolutional layer.

Custom Prediction Head

- The prediction head h^t take $\dot{y} \in \dot{\mathcal{Y}}$ as input and return a task-dependent output tensor.

Distribution Alignment

Goal: manipulate the target data so that they become **closer to the pretraining modality**.

- **train the embedder** before actually fine-tuning the model body.
- makes the **embedded target features** resemble the **source features**.
- **key idea**.

Denote:

- $f^s : \mathcal{X}^s \rightarrow \mathcal{X}$ is the pretrained source embedder.
- We can learn f^t by minimize $D(P(f^t(x^t), y^t) || P(f^s(x^s), y^s))$
- D is a metric for measuring distribution distance (MMD, OTDD, Euclidean).

Implement OTDD for Distribution Alignment

- Compute distance between two dataset : $\mathcal{D}^s = (\dot{x}^s, y^s)$ and $\mathcal{D}^t = (\dot{x}^t, y^t)$
- OTDD represents each class label as a distribution over the in-class features :

$$y \mapsto \alpha_y(X) = P(\dot{X} \mid Y = y)$$

- We can compute distance between feature-label pairs as :

$$d_Z((x, y), (x', y')) = [d_{\mathcal{X}}(x, x')^p + W_p^p(\alpha_y, \alpha_{y'})]^{\frac{1}{p}}$$

- we can finally use optimal transport to compute OTDD :

$$d_{OT}(\mathcal{D}^s, \mathcal{D}^t) = \min_{\pi \in \Pi(\alpha, \beta)} \int_{Z \times Z} d_Z(z^s, z^t)^p \pi(z^s, z^t)$$

- $z = (\dot{x}, y)$

1 Introduction

2 ORCA

3 Implement ORCA

4 Enhancing Cross-Modal Fine-Tuning

5 MoNa

6 Proposed

7 Appendix

8 References

Research questions

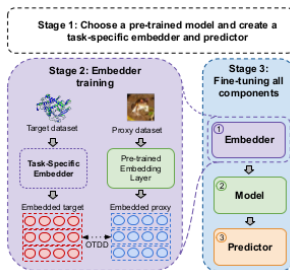


Figure 4: ORCA work flow

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

Experimental setup

Transformers pre-train model :

- 1 RoBERTa-base(Liu et al.,2019) for 1D tasks.
- 2 Swin-base(Liu et al.,2021) for 2D tasks.

Embedder training

- Using OTDD.

Target dataset

- 1 1D datasets : Satellite, DeepSEA, and ECG
- 2 2D datasets : NinaPro, CIFAR-100, and Darcy Flow.

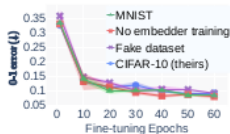
Proxy datasets

- 1 CIFAR-10 (Krizhevsky, 2009) for all 2D tasks.
- 2 CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) for all 1D tasks.

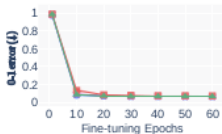
How does the choice of proxy dataset affect performance?

Experiment with the **choice of proxy dataset** for the tasks.

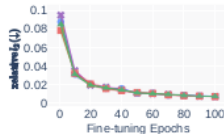
- Base line : embedder is **trained with different proxy datasets** or **not trained**.



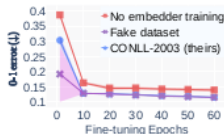
(a) NinaPro



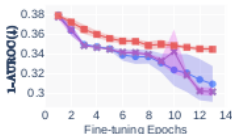
(b) CIFAR-100



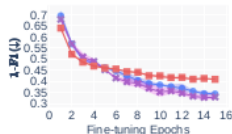
(c) Darcy Flow



(d) Satellite



(e) DeepSEA



(f) ECG

Figure 5: Per-epoch fine-tuning performance

How does the choice of proxy dataset affect performance?

Paper finding:

- **Embedder training** does **play a role in the 1D tasks**, but does **not matter for 2D tasks**.

Questions:

- ① Can we trust this conclusion (just comparing 4 dataset) ?
- ② Why are there differences between 2D tasks and 1D tasks?

Does doing (more) embedder training improve performance?

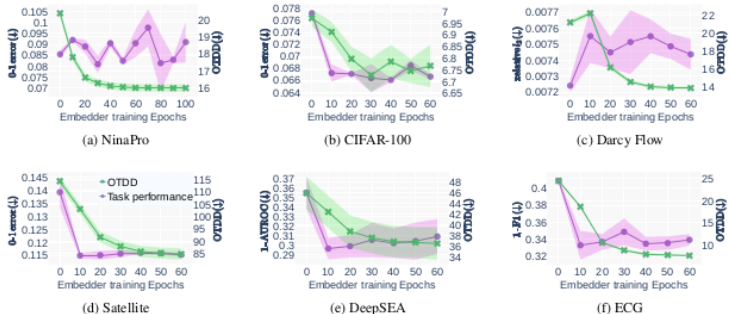


Figure 6: Per-epoch embedder training comparing OTDD

Paper finding:

- Embedder training is **unnecessary** in 2/6 tasks, training the embedder more can even lead to **worse task performance**.

What do the embedder and the pre-trained model contribute individually?

Experiment with freezing different parts of the pipeline:

- Freezing just the embedder, just the model, or both

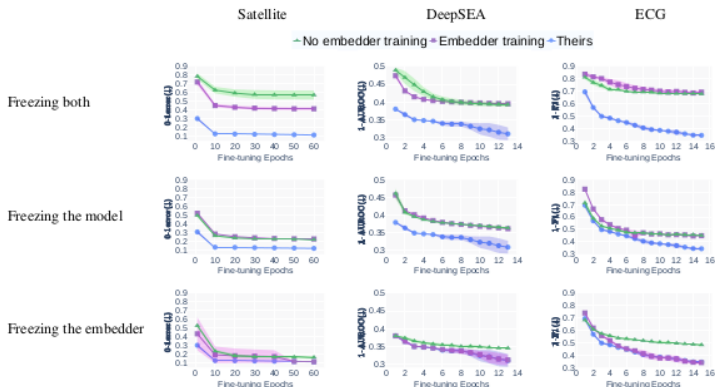


Figure 7: Freezing just the embedder, just the model, or both.

What do the embedder and the pre-trained model contribute individually?

Paper finding:

- 1 fine-tuning the pre- trained model is a **critical component** of ORCA.
- 2 while training the embedder is important for ORCA's success on these datasets.
 - it need not be fine-tuned beyond that.

Pre-training is not always necessary

Use RoBERTa models pre-trained on **different** amounts of English data: 10B, 100M, 10M, ...

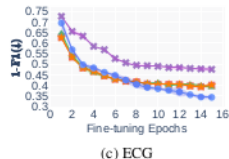
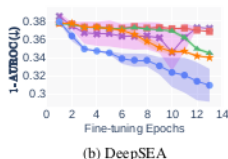
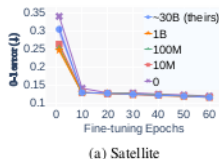


Figure 8: Effect of different amounts of pre-training data on downstream performance.

Paper finding:

- the amount of pre-training has a notice able effect only at certain(30B) scales.

Conclusion

In 1D task:

- ① some amount of **embedder training** is **necessary**.
- ② **more embedder training** can even **hurt performance** on the target task.
- ③ using a pre-trained model is **actually not necessary**.

In 2D task:

- ① **embedder training** does **not help at all**.

Questions:

- ① Can we trust this conclusion (just comparing 4 dataset) ?
- ② Why are there differences between 2D tasks and 1D tasks?

Idea for question about differences between 2D tasks and 1D tasks.

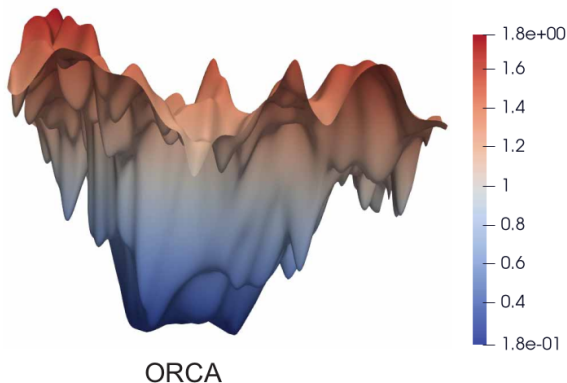


Figure 9: orca loss in Ninapro dataset

ORCA: unstable training, **trapped in unfavorable local optima.**

1 Introduction

2 ORCA

3 Implement ORCA

4 Enhancing Cross-Modal Fine-Tuning

5 MoNa

6 Proposed

7 Appendix

8 References

Introduction

ORCA:

- faces **instability during training**.
- potentially leading to suboptimal results as it is prone to getting **trapped in unfavorable local optima**.

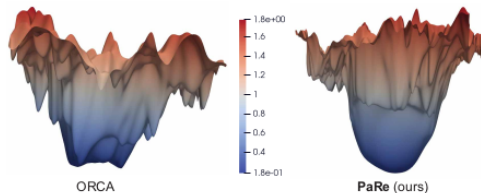


Figure 10: Loss landscape

Patch Replacement(PaRe):

- Motivated by **traditional data augmentation** techniques like Mixup (Zhang et al., 2017) and CutMix (Yunet al., 2019)

Patch Replacement(PaRe)

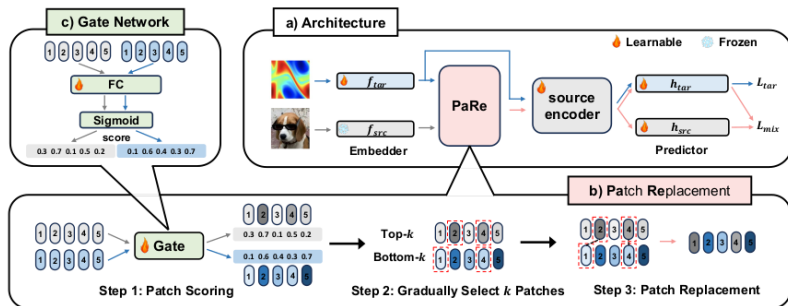


Figure 11: PaRe Overview

Embedder:

- pre-trained embedder $f_s : \mathcal{X}^s \rightarrow \tilde{\mathcal{X}}^s$
- target embedder f_t is randomly initialized $f_t : \mathcal{X}^t \rightarrow \tilde{\mathcal{X}}^t$

PaRe

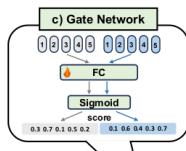


Figure 12: Patch scoring

Patch scoring:

- The source embeddings $\tilde{X}_s \in \mathbb{R}^{N \times D}$ contain N patches, where $\tilde{X}_s = \{\tilde{x}_{s1}, \tilde{x}_{s2}, \dots, \tilde{x}_{sN}\}$.
- Score each patch \tilde{x}_{si} from the source and \tilde{x}_{ti} from the target using a gate network.

$$S_s = \sigma(F_C(\tilde{X}_s))$$

- The **higher the score**, the more **critical information** the patch contains

PaRe

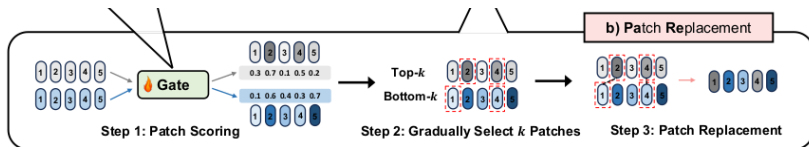


Figure 13: Pa Re

Patches replacement

- keep the positions of the top $(N - k)$ **target patches** with the **highest scores fixed**.
- replace the bottom (k) **target patches** with the lowest scores with the **top (k) source patches** with the highest scores.
- k **linearly decreases** with the number of training epochs:

$$k = k_0 \left(1 - \frac{\text{ep}_{\text{current}}}{\text{ep}_{\text{total}}} \right)$$

Loss function

Denote:

- Output of patches replacement \tilde{X}^m for **source and target**.
- Output of **source encoder** $g(\tilde{X}^m)$.
- source prediction $p^{ms} = h_s(g(\tilde{X}^m))$ and target prediction $p^{mt} = h_t(g(\tilde{X}^m))$.
- Weight $\lambda = \frac{k}{k_0}$.

Loss function:

- Calculate the mixed loss \mathcal{L}_{mix} using mixed embeddings \tilde{X}^m .

$$\mathcal{L}_{\text{mix}} = (1 - \lambda)\mathcal{L}_{\text{tar}}(p^{mt}, y_t) + \lambda\mathcal{L}_{\text{src}}(p^{ms}, y_s)$$

Experiment Setup

Pre-train model:

- 1 RoBERTa (Liu et al., 2019) for 1D tasks.
- 2 Swin Transformers (Liu et al., 2021) for 2D tasks.

Proxy datasets:

- 1 2D classification tasks: CIFAR10, Tiny-ImageNet.
- 2 2D dense prediction tasks: VOC.
- 3 1D tasks : CoNLL-2003.

Overall results

NAS-Bench-360 :

- comprises four 2D classification tasks, three 2D dense prediction tasks, and three 1D tasks.
- PaRe achieves the best performance across all tasks.

	CIFAR-100 0-1 error (%)	Spherical 0-1 error (%)	Darcy Flow relative ℓ_2	PSICOV MAE _s	Cosmic 1-AUROC	NinaPro 0-1 error (%)	FSD50K 1-mAP	ECG 1-F1 score	Satellite 0-1 error (%)	DeepSEA 1-AUROC
Hand-designed	19.39	67.41	8.00E-03	3.35	0.127	8.73	0.62	0.28	19.80	0.30
NAS-Bench-360	23.39	48.23	2.60E-03	2.94	0.229	7.34	0.60	0.34	12.51	0.32
DASH	24.37	71.28	7.90E-03	3.30	0.190	6.60	0.60	0.32	12.28	0.28
Perceiver IO	70.04	82.57	2.40E-02	8.06	0.485	22.22	0.72	0.66	15.93	0.38
FPT	10.11	76.38	2.10E-02	4.66	0.233	15.69	0.67	0.50	20.83	0.37
NFT	7.67	55.26	7.34E-03	1.92	0.170	8.35	0.63	0.44	13.86	0.51
ORCA	6.53	29.85	7.28E-03	1.91	0.152	7.54	0.56	0.28	11.59	0.29
PaRe	6.25	25.55	7.00E-03	0.99	0.121	6.53	0.55	0.28	11.18	0.28

Figure 14: Prediction errors (↓) across 10 diverse tasks on NAS-Bench-360.

Overall results(.cnt)

PDEBench:

- comprises multiple scientific ML-related datasets, with a focus on the physics domain.

	Advection 1D	Burgers 1D	Diffusion-Reaction 1D	Diffusion-Sorption 1D	Navier-Stokes 1D	Darcy-Flow 2D	Shallow-Water 2D	Diffusion-Reaction 2D
PINN	6.70E-01	3.60E-01	6.00E-03	1.50E-01	7.20E-01	1.80E-01	8.30E-02	8.40E-01
FNO	1.10E-02	3.10E-03	1.40E-03	1.70E-03	6.80E-02	2.20E-01	4.40E-03	1.20E-01
U-Net	1.10E+00	9.90E-01	8.00E-02	2.20E-01	-	-	1.70E-02	1.60E+00
ORCA	9.80E-03	1.20E-02	3.00E-03	1.60E-03	6.20E-02	8.10E-02	6.00E-03	8.20E-01
PaRe	2.70E-03	8.30E-03	2.60E-03	1.60E-03	6.62E-02	8.06E-02	5.70E-03	8.18E-01

Figure 15: Normalized Root Mean Squared Errors (nRMSEs, ↓) across 8 tasks of PDEBench

Limitation

- ① Determining the most suitable source modality proxy dataset based on the target modality dataset remains a **challenge**.
- ② a modality-agnostic **data augmentation** method is **necessary** to **prevent model overfitting** and enhance cross-modal fine-tuning.

- 1 Introduction
- 2 ORCA
- 3 Implement ORCA
- 4 Enhancing Cross-Modal Fine-Tuning
- 5 MoNa**
- 6 Proposed
- 7 Appendix
- 8 References

Introduction

The **cross-modal transfer** is not as straightforward as the **in-modality transfer** due to two challenges:

- 1 The input and label space are different across modalities.
- 2 The knowledge required for addressing tasks in **different modality** may also **differ**.

Key problem:

- What knowledge **from source modality** is **transferred via the pretrained model** ?
- How does it **benefit the target modality** ?

MoNa : Improves the **cross-modal transfer** with two-stage training.

- 1 leverages meta learning to learn an **optimal target embedder**.
- 2 vanilla finetuning.

Notations

Model Architecture: g_θ

- An embedder $e(\cdot; \theta_e)$.
- A transformer encoder $f(\cdot; \theta_f)$.
- A predictor $h(\cdot; \theta_h)$.
- The parameter of full model as $\theta = \{\theta_e, \theta_f, \theta_h\}$.
- Pretrained weights of the source model as $\theta_0^S = \{\theta_{e_0}^S, \theta_{f_0}^S, \theta_{h_0}^S\}$.
- Target model $\theta^T = \{\theta_e^T, \theta_f^T, \theta_h^T\}$.
- **Vanilla finetuning** :

$$\theta_T^* = \arg \min_{\theta_T} \sum_{i=1}^{n_T} \ell(g_\theta^T(x_i^T), y_i^T),$$

Distortion of learned source modality knowledge

Problem:

- There **lacks a general metric** measuring the degree of **knowledge reuse** during transfer.

Expectation:

- Smaller distortion** if more source knowledge is **reused** to solve target task.

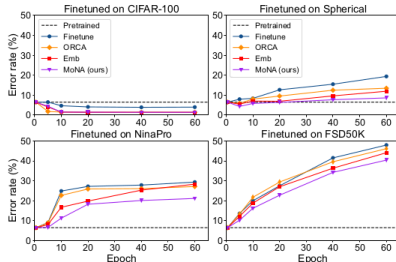


Figure 16: Linear probing results on CIFAR-10.

Distortion of learned source modality knowledge(.cnt)

Finding:

- More epochs leads to larger distortion of source knowledge on all target modalities except for CIFA-100.
- A large discrepancy may hinder the effectiveness of cross-modal transfer.
- Embedder training:
 - Mitigates the knowledge misalignment between target and source.
 - Reduces the model distortion during its adaptation towards target tasks.
- The key to effective transfer:
 - Learn a target embedding function $e^T : X \rightarrow \hat{X}$.
 - makes the target conditional distribution $P(Y^t|\hat{X})$ more aligned with the source knowledge.

MoNA: Modality Knowledge Alignment

MoNa:

- 1 Stage 1: Target embedder training.
- 2 Stage 2: Full finetuning.

Target embedder training:

- The **inner-loop** is the optimization of the model on target dataset.

$$\theta^{\mathcal{T}*}(\phi_e) = \arg \min L_{\text{inner}}(x_t, y_t; \phi_e)$$

- Where L_{inner} is the target loss.
- Virtually update **model weight** : $\theta^{\mathcal{T}} = \theta^{\mathcal{T}} - \alpha \nabla \mathcal{L}_{\text{inner}}$
- Outer-loop : find optimal embedder parameters ϕ_e^* .
 - resulting optimal target encoder.
 - generates high quality representations of source data.

MoNA: Modality Knowledge Alignment(.cnt)

Outer-loop:

- Compute source features $\{f_i^S = f(e(x_i^s, \theta_e^s), \theta_f^T)\}$
- Measures the source discriminability of the induced encoder :

$$\begin{aligned}\mathcal{L}_{\text{outer}} &= \mathcal{L}_{\text{align}} + \mathcal{L}_{\text{uniform}} \\ &= -\mathbb{E}_{i,j:y_i=y_j} \left[\|f_i - f_j\|_2^2 \right] - \log \mathbb{E}_{i,j} \left[e^{-2\|f_i - f_j\|_2^2} \right]\end{aligned}$$

- To prevent the embedder from **overly focusing on source modality**, we have :

$$\mathcal{L}'_{\text{outer}} = \lambda \mathcal{L}_{\text{outer}} + \mathcal{L}_{\text{inner}}$$

- Update the **target embedder** as :

$$\phi_e = \phi_e - \beta \nabla \mathcal{L}'_{\text{outer}}$$

MoNA: Modality Knowledge Alignment(.cnt)

Algorithm 1 MoNA: Modality Knowledge Alignment

Input: Source pretrained model $g_{\theta_0^S}$; Learning rate α, β ;
Maximum iterations I_1, I_2 .

Output: Model for the target task: g_{θ^T} .

Stage 1: Target embedder training

for $iter = 1, 2, \dots, I_1$ **do**

 Initialize the target model g_{θ^T} with ϕ_e and $\theta_{f_0}^S$.

 Virtually update: $\theta^{T*} = \theta^T - \alpha \nabla_{\theta^T} \mathcal{L}_{inner}$.

 Compute source features $\{f_i^s\}$ with $\theta_{e_0}^S$ and θ_f^{T*} .

 Obtain outer-loop loss using Eq. (4).

 Update target embedder: $\phi_e \leftarrow \phi_e - \beta \nabla_{\phi_e} \mathcal{L}'_{outer}$.

end for

Stage 2: Full finetuning

for $iter = 1, 2, \dots, I_2$ **do**

 Initialize the target model g_{θ^T} with ϕ_e and $\theta_{f_0}^S$.

 Update target model towards Eq. (1).

end for

Figure 17: Modality Knowledge Alignment.

Experiments Setup

Pre-train model:

- 1 RoBERTa (Liu et al., 2019) for 1D tasks.
- 2 Swin Transformers (Liu et al., 2021) for 2D tasks.

Source datasets:

- 1 2D classification tasks: CIFAR10
- 2 1D tasks : CoNLL-2003.

NAS-Bench-360 :

- comprises four 2D classification tasks, three 2D dense prediction tasks, and three 1D tasks.

PDEBench:

- comprises multiple scientific ML-related datasets, with a focus on the physics domain.

Experiments Result

NAS-Bench-360:

Model	CIFAR-100	Spherical	Darcy Flow	PSICOV	Cosmic	NinaPro	FSD50K	ECG	Satellite	DeepSEA
ORCA	6.53	29.85	7.28 E-3	1.91	0.152	7.54	0.56	0.28	11.59	0.29
MoNA	6.48	27.13	6.80 E-3	0.99	0.121	7.28	0.55	0.27	11.13	0.28
PaRe	6.25	25.55	7.00 E-3	0.99	0.121	6.53	0.55	0.28	11.18	0.28

Table 1: Prediction errors (↓) on ten tasks of NAS-Bench-360.

PDEBench:

Model	Advection	Burgers	Diffusion-Reaction	Diffusion-Sorption	Navier-Stokes	Darcy-Flow	Shallow-Water	Diffusion-Reaction
ORCA	9.80 E-3	1.20 E-2	3.00 E-3	1.60 E-3	6.20 E-2	8.10 E-2	6.00 E-3	8.20 E-1
MoNA	8.8 E-3	1.14 E-2	2.80 E-3	1.60 E-3	5.40 E-2	7.9 E-2	5.70 E-3	8.18 E-1
PaRe	2.70 E-3	8.30 E-3	2.60 E-3	1.60E-3	6.62 E-2	8.06 E-2	5.70 E-3	8.18 E-1

Table 2: Normalized Root Mean Squared Errors (nRMSEs, ↓) across 8 tasks of PDEBench.

- 1 Introduction
- 2 ORCA
- 3 Implement ORCA
- 4 Enhancing Cross-Modal Fine-Tuning
- 5 MoNa
- 6 Proposed**
- 7 Appendix
- 8 References

Idea

Problem setup:

- Transformers pre-train body model g_s in domain D^s .
- Dataset $\{x_i^t, y_i^t\}_{i=1}^N$ in domain D^t .
- Dataset $\{x_i^s, y_i^s\}_{i=1}^N$ in domain D^s

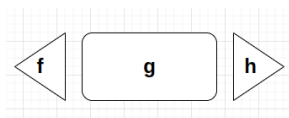


Figure 18: proposed model

embedder: f extract feature from input $f(\cdot)$

- random init for f_t .
- using pre-train and freezing for f_s

predictor: h generate output $h(\cdot)$.

Idea(cnt.)

Proposed workflow:

- ➊ Stage 1 : Contrastive training for embedder.
- ➋ Stage 2 : Patch-replacement fine-tuning.

Stage 1: Contrastive training for embedder.

- ➊ Step 1 : Data augmentation.
- ➋ Step 2 : Training target embedder.

Step 1: Data augmentation.

- Pertubing data T times with **multi-scale gaussian noise** $\{\sigma_i\}_{i=1}^K$.

$$x_i^t = \sqrt{1 - \sigma^2} x_i^{t-1} + \sigma * \epsilon$$

$$\epsilon \sim \mathcal{N}(0, I)$$

- We have Expanded Dataset $\{x_i, y_i\}_{i=1}^M$ with $M = N \times K \times T$

Step 2 : Contrastive training for target embedder

We have:

- feature $z_i = f(x_i)$ is a embedded vector.
- **Similarity** between two output:

$$\cos(z_i, z_j) = \frac{z_i z_j}{||z_i|| * ||z_j||}$$

- **Objective** : Similarity **high in the same label, low in the different label**.
- Loss function for label Y :

$$\mathcal{L}_Y = -\log \frac{\sum_j^{y_j=Y} \sum_i^{y_i=Y} \cos(z_i, z_j)}{\sum_i^{y_i=Y} \sum_{j=1}^M \lambda_{i,j} \cos(z_i, z_j)}$$

- $\lambda_{i,j} \propto \frac{1}{\text{correlation}(y_i, y_j) + \epsilon}$ and $\lambda_{i,j} = 1$ if $y_i = y_j$.

Patch-replacement fine-tuning

Stage 2: Patch-replacement fine-tuning.

- Using self-attention for scoring each feature patch.

Training options:

- ❶ Freezing embedder.
- ❷ Training embedder.

1 Introduction

2 ORCA

3 Implement ORCA

4 Enhancing Cross-Modal Fine-Tuning

5 MoNa

6 Proposed

7 Appendix

8 References

MMD

Define: MMD is a distance (difference) between feature means.

Denote:

- X and $\phi(X) \in \mathcal{F}$ is the a feature map.
- Assuming \mathcal{F} satisfies the necessary conditions:
 - X, Y such that $k(X, Y) = \langle \phi(X), \phi(Y) \rangle_{\mathcal{F}}$

Feature Mean:

- Given $\mathcal{X} \sim P$ we have feature means :

$$\mu_P = \mathbb{E}_{X \sim P}[\phi(X)]$$

Maximum mean discrepancy:

$$\text{MMD}(P, Q) = \|\mathbb{E}_{X \sim P}[\phi(X)] - \mathbb{E}_{Y \sim Q}[\phi(Y)]\|_{\mathcal{F}} = \|\mu_P - \mu_Q\|$$

Optimal transport(OP): Comparing by ‘transporting’



Figure 19: Optimal transport

Optimal transport

- a method to find least-cost schemes to **transport dirt and rubble from one place to another**.
- $OT_c(\alpha, \beta) := \min_{\pi \in \Pi(\alpha, \beta)} \int_{X \times X} c(x, y) d\pi(x, y)$.
 - $\Pi(\alpha, \beta)$ be the set of joint probability distributions on $X \times X$.
- $W_p(\alpha, \beta) \hat{=} OT(\alpha, \beta)^{1/p}$ is **called the p -Wasserstein distance**.

1 Introduction

2 ORCA

3 Implement ORCA

4 Enhancing Cross-Modal Fine-Tuning

5 MoNa

6 Proposed

7 Appendix

8 References

References

- 1 Junhong Shen, Liam Li, Lucio M. Dery , Corey Staten, Mikhail Khodak, Graham Neubig, Ameeta Talwalkar; Cross-Modal Fine-Tuning: Align then Refine
- 2 MMD.
- 3 OTDD
- 4 Paloma García-de-Herreros, Vagrant Gautam, Philipp Slusallek, Dietrich Klakow, Marius Mosbach, What explains the success of cross-modal fine-tuning with ORCA?
- 5 Lincan Cai, Shuang Li, Wenxuan Ma, Jingxuan Kang , Binhui Xie, Zixun Sun, Chengwei Zhu, Enhancing Cross-Modal Fine-Tuning with Gradually Intermediate Modality Generation.