Learning Backward-Compatible Representations

Niccolò Biondi

Simone Ricci

Federico Pernici

Alberto Del Bimbo

Università degli Studi di Firenze, Italy

ACM Multimedia 2024
Melbourne Australia, October 28, 2024

MICC
Media Integration and Communication Center

Outline of the Tutorial

Compatibility Learning: The Context

- ✓ Visual Search Systems Updating
- ✓ Backfiling (re-indexing)
- ✓ Compatibility Learning

Compatibility Learning: The Solutions

- ✓ Using Regularization
- ✓ Using Mapping
- ✓ Using Approaches
- ✓ Stationarity and Compatibility
- ✓ Using Softmax Outputs

Compatibility in Continual Learning

Compatibility with Foundation Models

- ✓ User-side Compatibility
- ✓ Third party-side Compatibility
- ✓ Negative Flips Reduction

Tutorial Presentation Schedule

October 28, 2024
2.30 pm – 6.00 pm

Part I 1h

COMPATIBILITY LEARNING: THE CONTEXT

- ✓ Visual Search Systems Updating
- ✓ Backfiling (re-indexing)
- ✓ Compatibility Learning

COMPATIBILITY LEARNING: THE SOLUTIONS

- ✓ Using Regularization
- ✓ Using Mapping
- ✓ Using Architectural Changes

Looking at the code

Part II 1h

COMPATIBILITY LEARNING: THE SOLUTIONS

- ✓ Stationarity and Compatibility
- ✓ Using Softmax Outputs

COMPATIBILITY IN CONTINUAL LEARNING

Looking at the code

Part III 1h

COMPATIBILITY WITH FOUNDATION MODELS

- ✓ User-side Compatibility

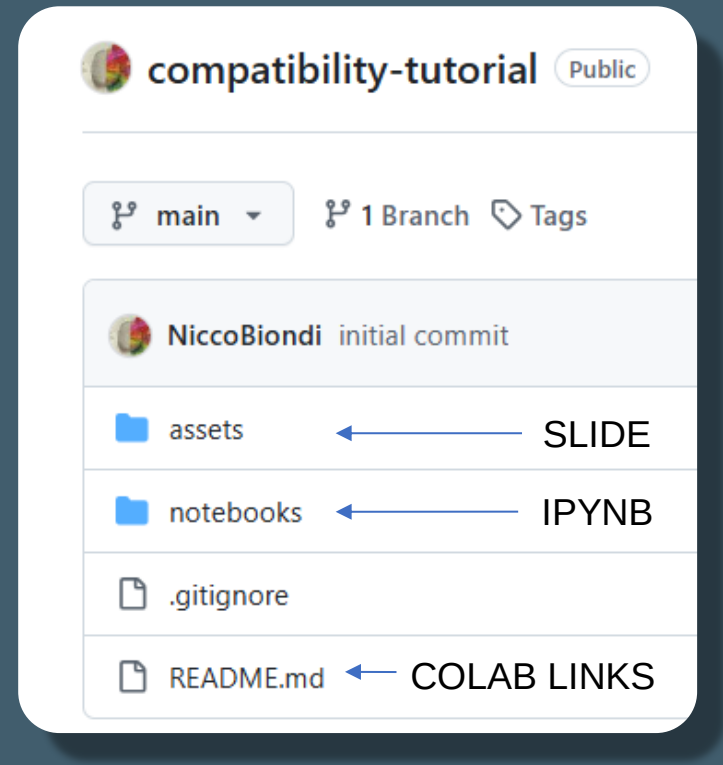
Looking at the code

- ✓ Third party-side Compatibility
- ✓ Negative Flips Reduction

GitHub Repo

You can get Slide and Practical Examples from our GitHub repository

github.com/NiccoBiondi/compatibility-tutorial



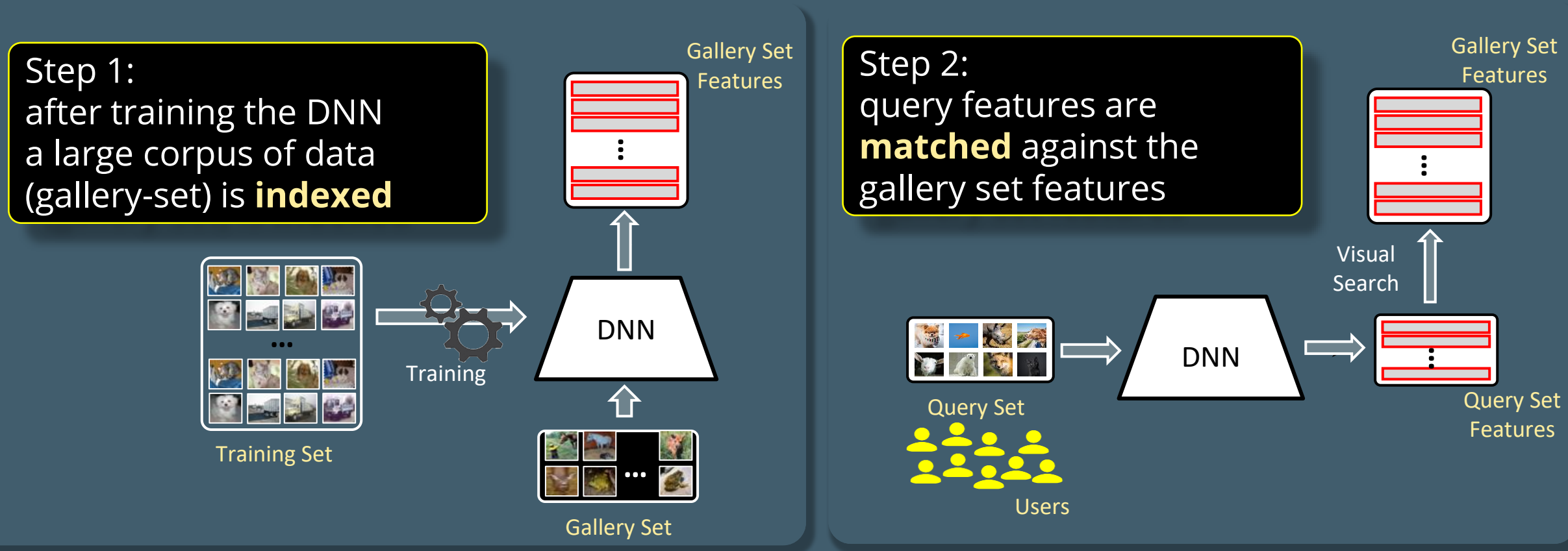
Compatible learning

The Context

Retrieval Systems

Visual search systems address the problem of finding similar data in a gallery-set

The core part includes an
online service of the embedding model and a large-scale vector database



Retrieval Systems Updating

In real-world applications novelties constantly emerge

Embedding models with **improved performance are released and updated continuously** to achieve a better user experience

Updating a DNN:

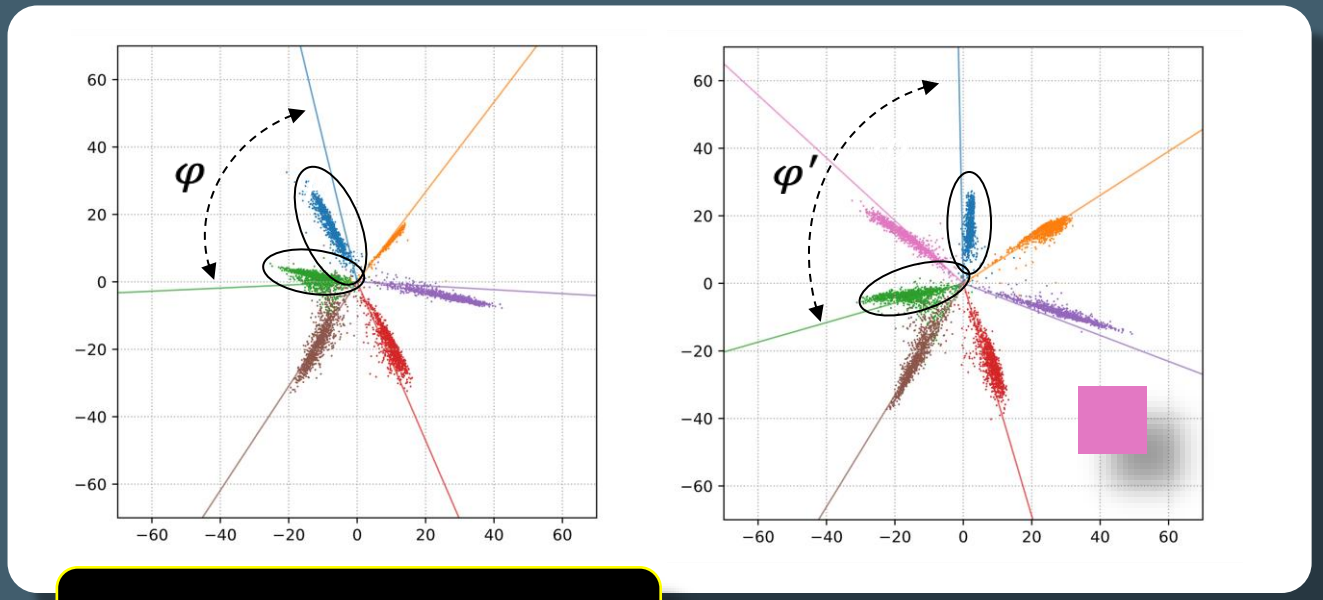
- ✓ More data
- ✓ Different network architectures
- ✓ Different training techniques

Main Issue

Updating a DNN model with a novel class **changes the internal feature representation**

Query features obtained with the new model **cannot be compared** with the old gallery features

MNIST dataset (2D representation space)

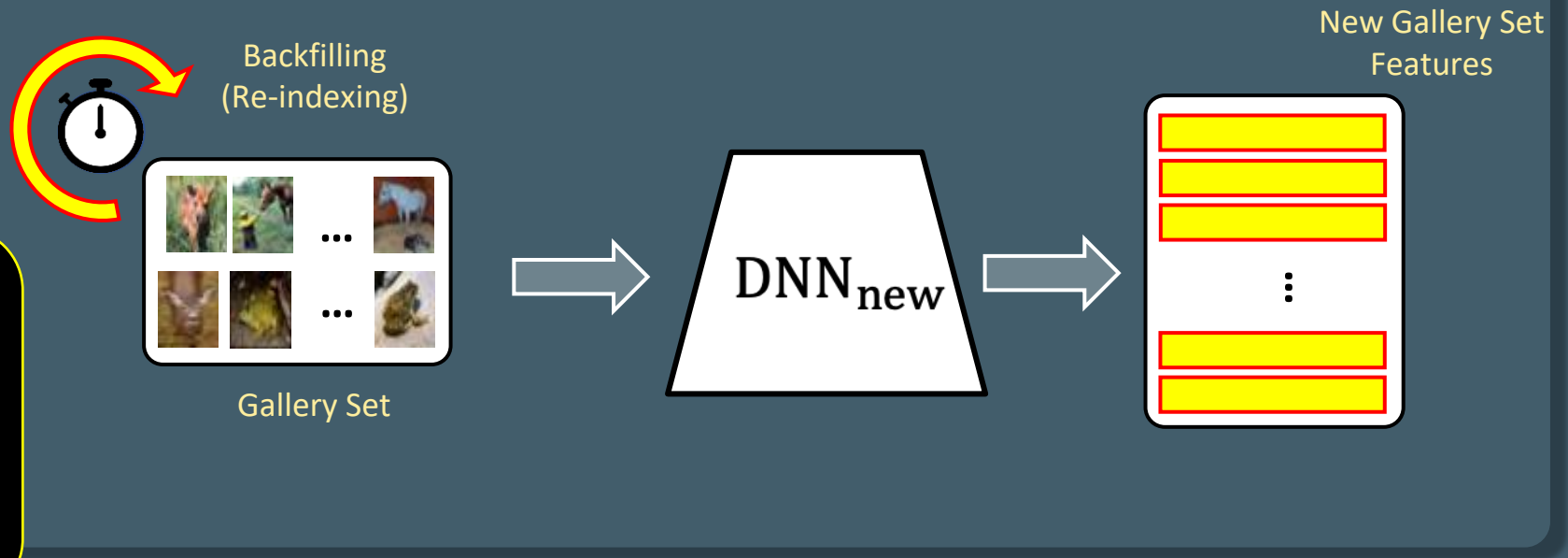


The geometry of the representation changes: $\varphi \neq \varphi'$

Backfilling 👎

For traditional model upgrades the old model is not replaced by the new one until the embeddings of all the images in the database are **re-computed by the new model**.....

Which takes days or weeks for a large amount of data



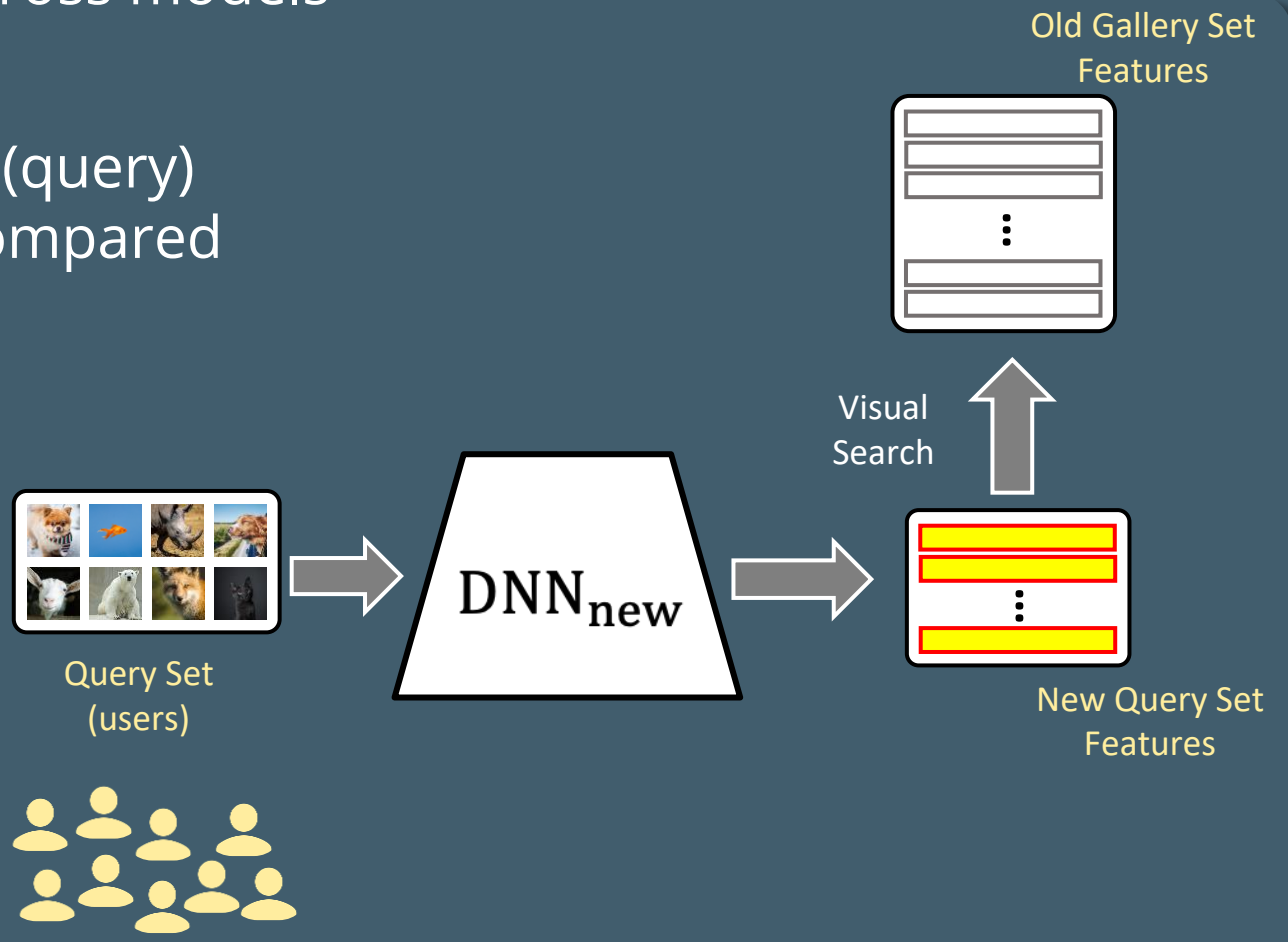
Issues:

- ✓ Time consuming
- ✓ Carbon Footprint
- ✓ Storage
- ✓ Privacy

Compatible Representation Learning 👍

Key idea: making the new representation incorporate the new information **while remaining aligned** with the old model
It is essentially transferring knowledge across models

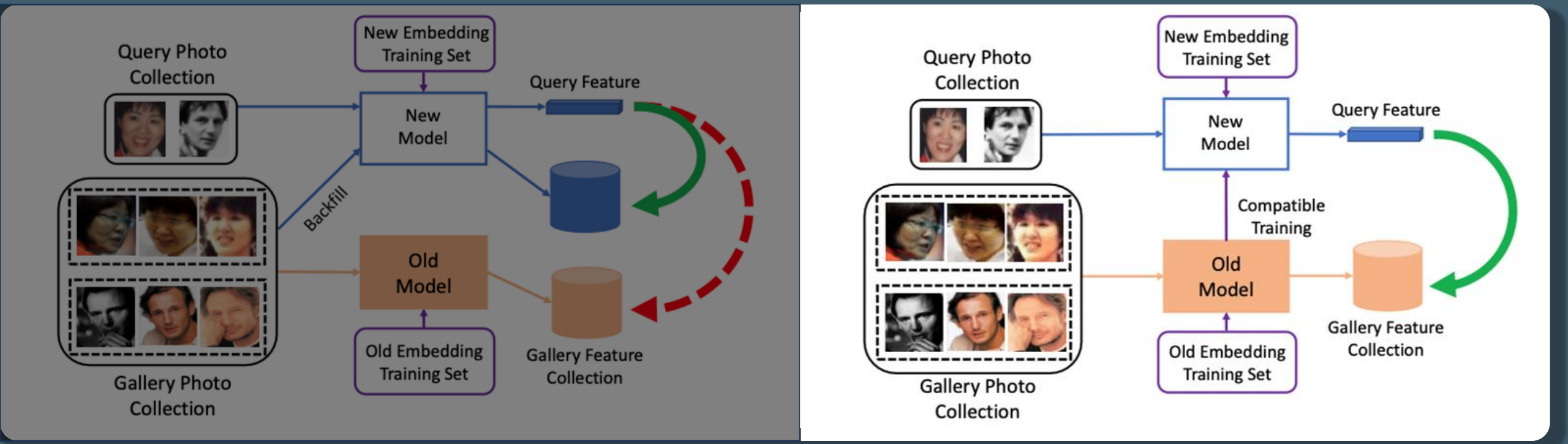
Compatible representations allow new (query) and old (gallery) features to be directly compared



Benefits:
- ✓ Avoid re-indexing gallery-set
- ✓ Privacy-preserving
- ✓ Memory-efficient

Backfilling vs Compatible Learning

With a backward compatible representation
direct comparison becomes possible



[*] Y. Shen et al., CVPR 2020

pictures
authored by Y. Shen

Backward Compatibility [*]

Backward Compatibility Criterion:

$$\mathrm{dist}\left(\phi_{\mathrm{new}}(\mathbf{x}_i), \phi_{\mathrm{old}}(\mathbf{x}_j)\right) \leq \mathrm{dist}\left(\phi_{\mathrm{old}}(\mathbf{x}_i), \phi_{\mathrm{old}}(\mathbf{x}_j)\right)$$
$$\forall\,(i,j) \in \left\{(i,j)\,|\,y_i = y_j\right\}$$
$$\text{and}$$
$$\mathrm{dist}\left(\phi_{\mathrm{new}}(\mathbf{x}_i), \phi_{\mathrm{old}}(\mathbf{x}_j)\right) \geq \mathrm{dist}\left(\phi_{\mathrm{old}}(\mathbf{x}_i), \phi_{\mathrm{old}}(\mathbf{x}_j)\right)$$
$$\forall\,(i,j) \in \left\{(i,j)\,|\,y_i \neq y_j\right\},$$

for all pairs of samples $\mathbf{x}_i$ *and* $\mathbf{x}_j$

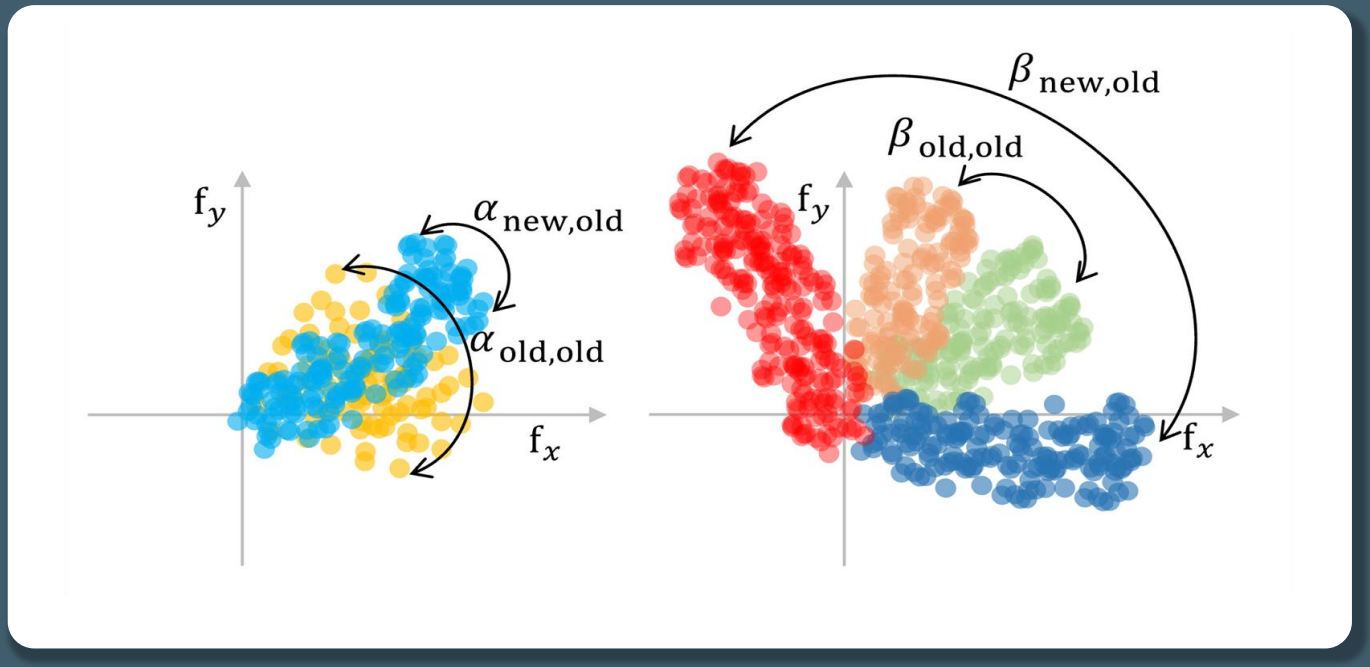
The new embedding when used to compare against the old embedding must be at least as good as the old one in separating images from different classes and grouping those from the same classes [*]

[*] Y. Shen et al., CVPR 2020

Backward Compatibility

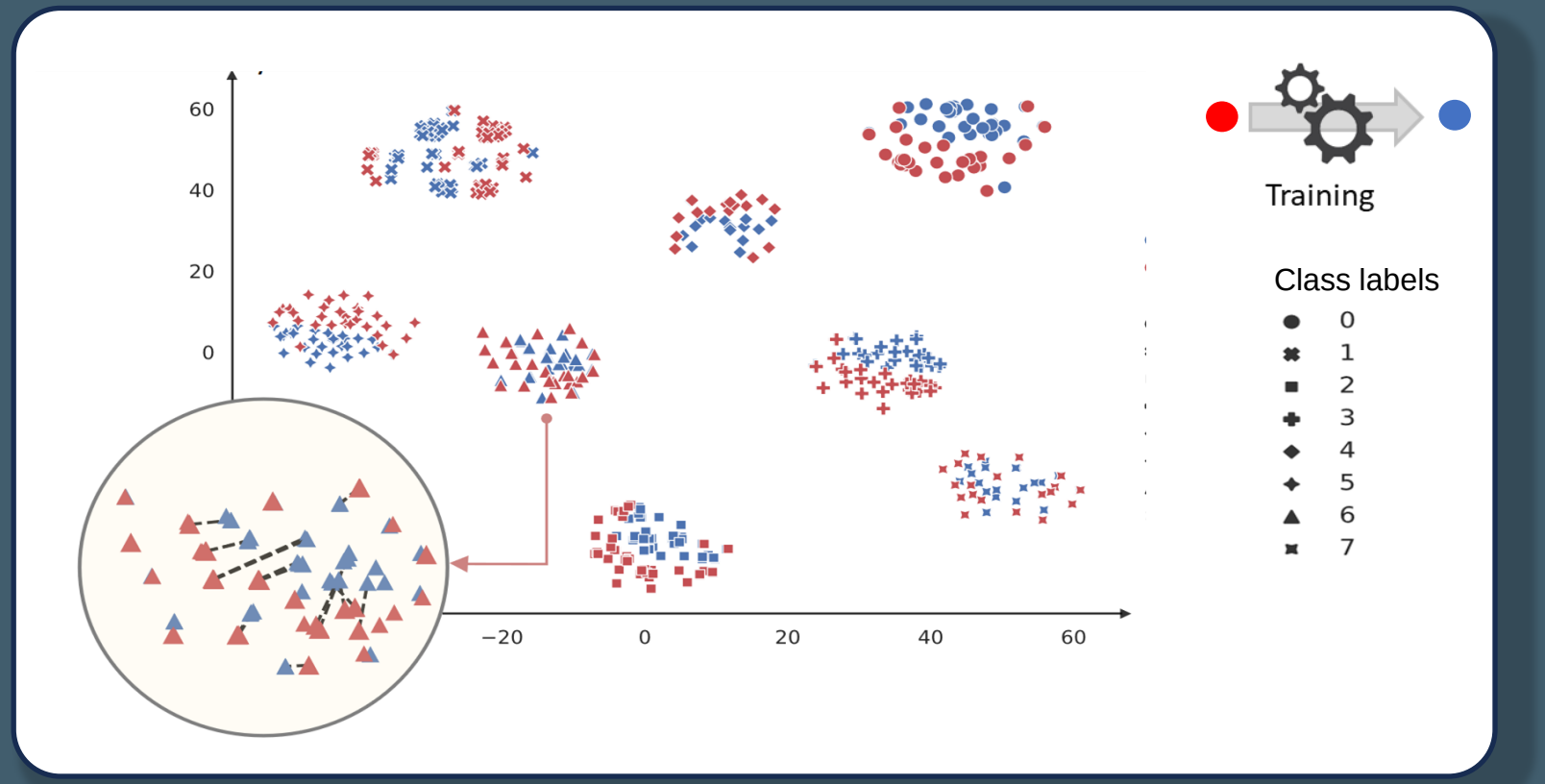
The new model must be at least as good as the old one in:

- ✓ **grouping** images from the same classes
  $(\alpha_{\text{new,old}} \leq \alpha_{\text{old,old}})$
- ✓ **separating** images from different classes
  $(\beta_{\text{new,old}} \geq \beta_{\text{old,old}})$



Main Goal

We are interested in learning a new representation that **does not change** its geometrical structure after updating, i.e. that **is aligned** to the old one



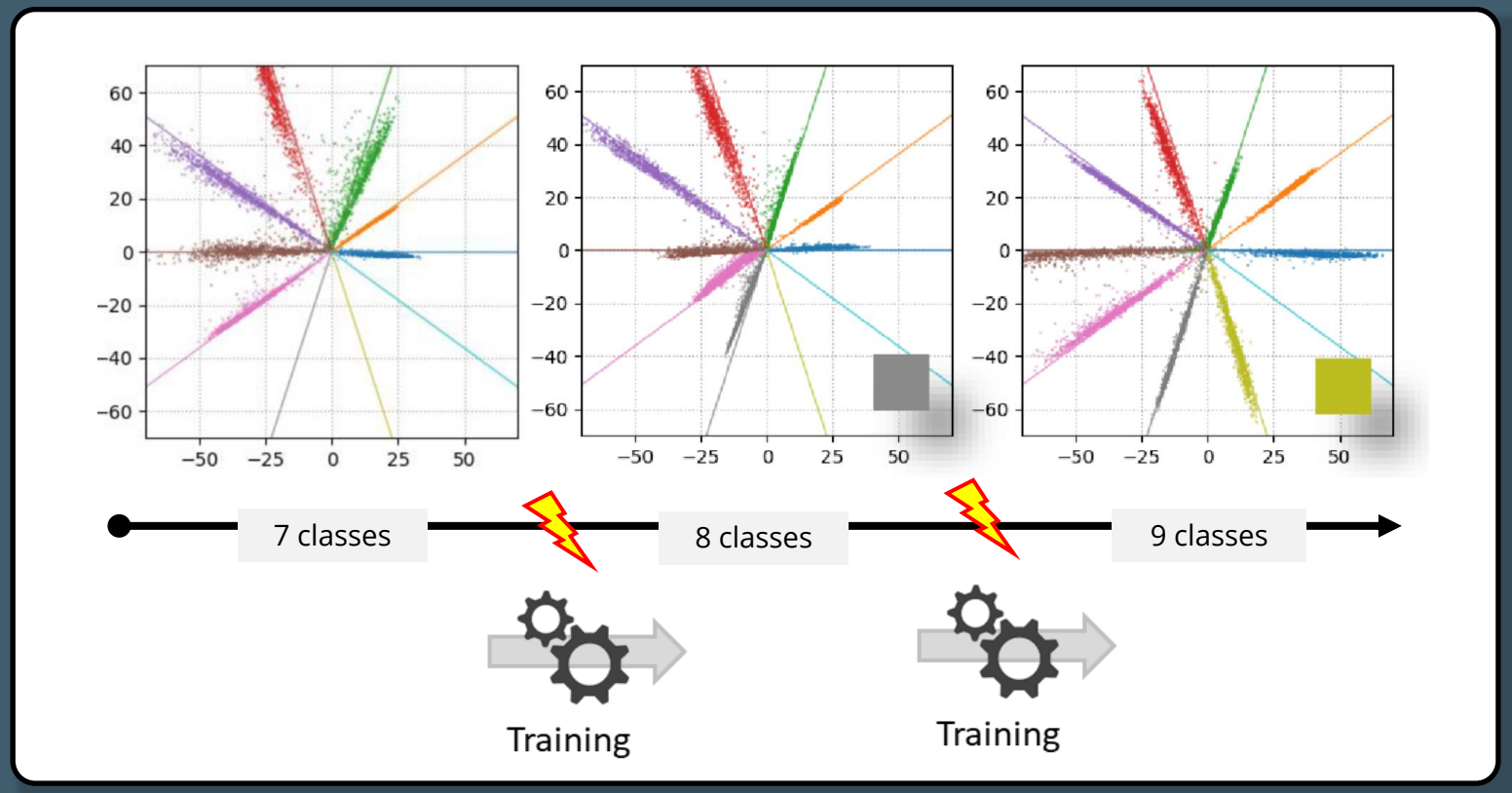
Same classes

picture
authored by Meng

Main Goal

We are interested in learning a new representation that **does not change** its geometrical structure after updating, i.e. that **is aligned** to the old one

MNIST dataset (2D representation space)

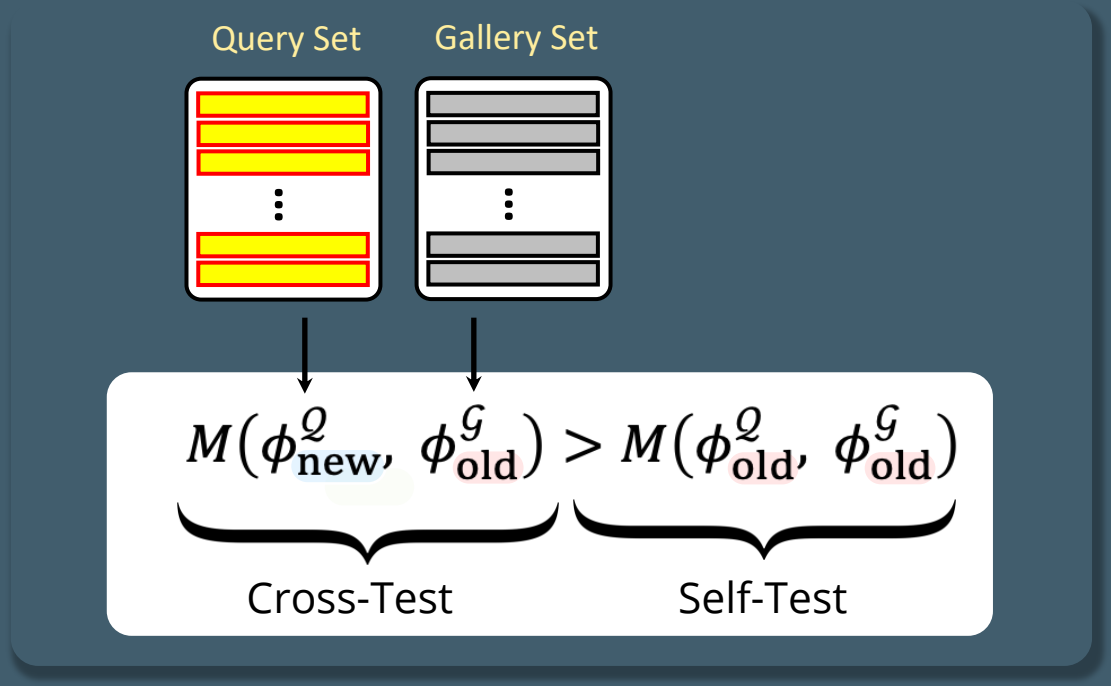


Adding new classes

Empirical Compatibility [*]

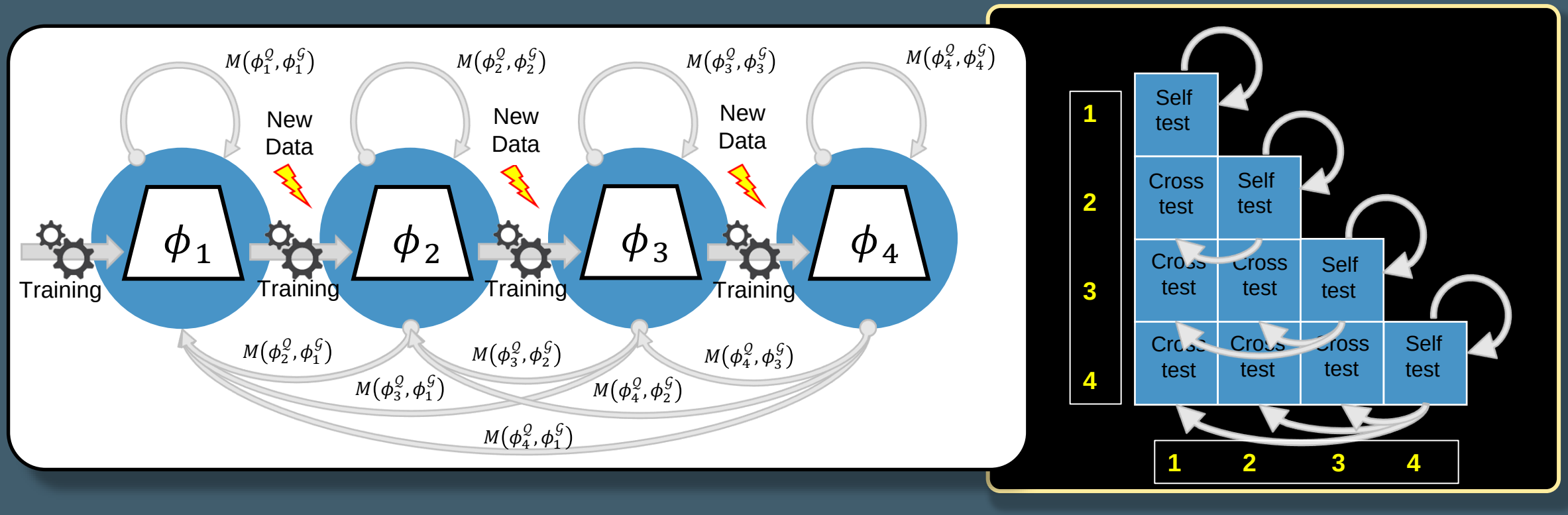
The Backward Compatibility criterion is intractable at large scale and in the open-set setting. It entails testing the gallery exhaustively

Empirical Compatibility criterion: backward compatibility is achieved when the accuracy using φ_{new} for queries without backfilling gallery images surpasses that of using φ_{old}



[*] Y. Shen et al., CVPR 2020

Multi-step Compatibility Matrix [*]

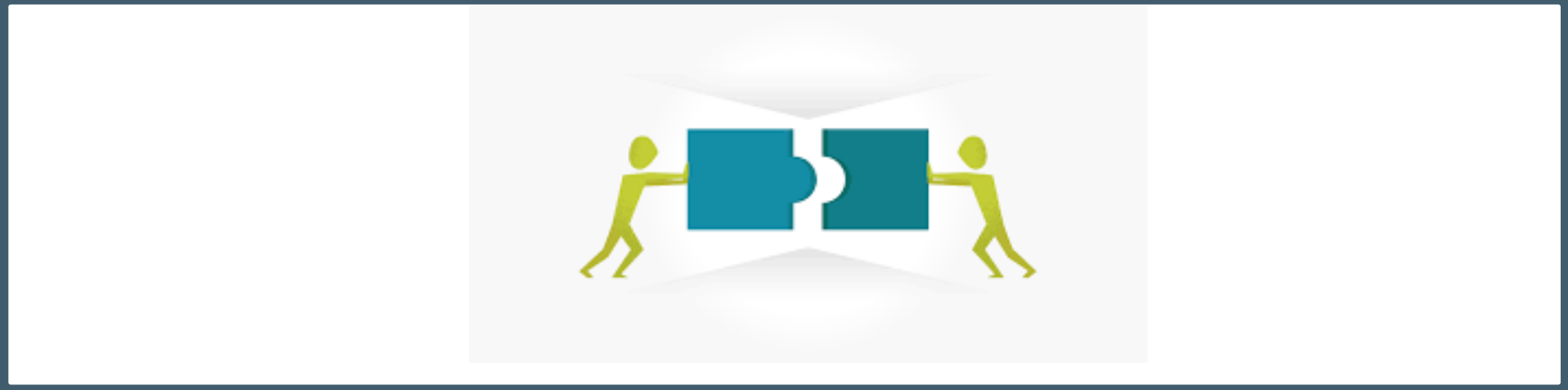


pictures
authored by F. Pernici

Compatibility between two models is achieved if the
cross-test is higher than the self-test of the older model

[*] N. Biondi et al. , IEEE TPAMI 2023

COMPATIBLE LEARNING



SOLUTIONS

Aligning the Representations

Using Regularization:

aims at upgrading models with additional loss functions that impose constrains with respect to the previous model

- ✓ BCT (CVPR2020)
- ✓ UniBCT (IJCAI2022)
- ✓ AdvBCT (CVPR2023) …

Using Mapping

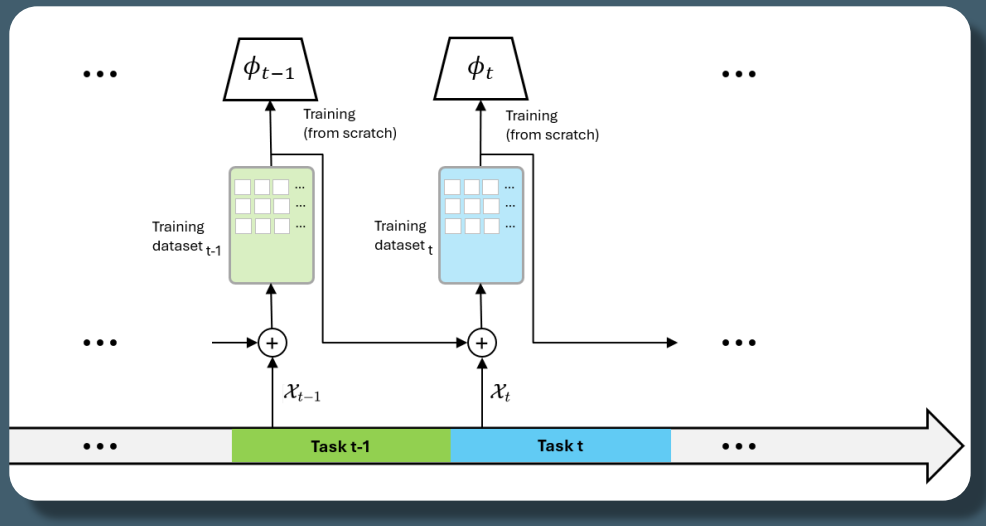
finds compatible mappings between the previous and upgraded models, where the upgraded models already exist

- ✓ CMC-RBT (BMVC2020)
- ✓ LCE (ICCV2021)
- ✓ FCT (CVPR2022) …

Using Architectural Changes:

impose constrains on the network architecture that force feature alignment

- ✓ CoReS (TPAMI2023)
- ✓ BT^2 (ICCV2023)
- ✓ OCA (ECCV2024)



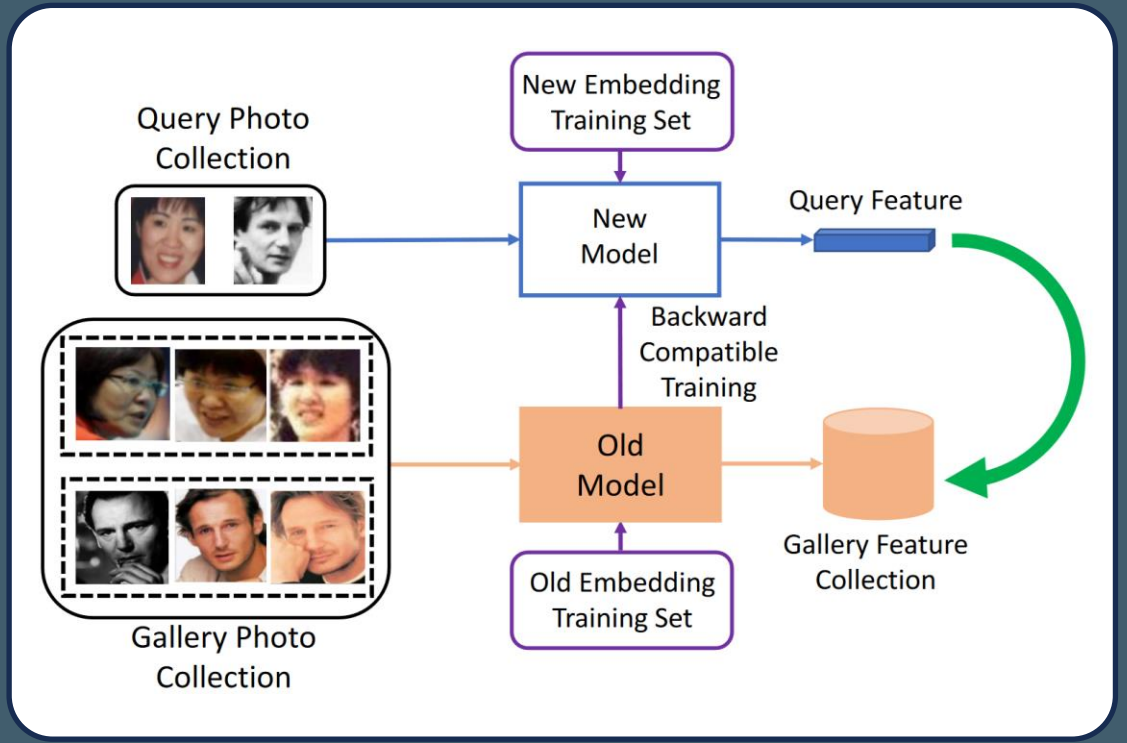
picture authored by N. Biondi

Backward Compatible Training (BCT) [*]

Regularization-based

BCT learns the new compatible model by **freezing the old classifier** and using it as a regularizer to align the new class prototypes to the old ones

The new model learns from both the new and the old data



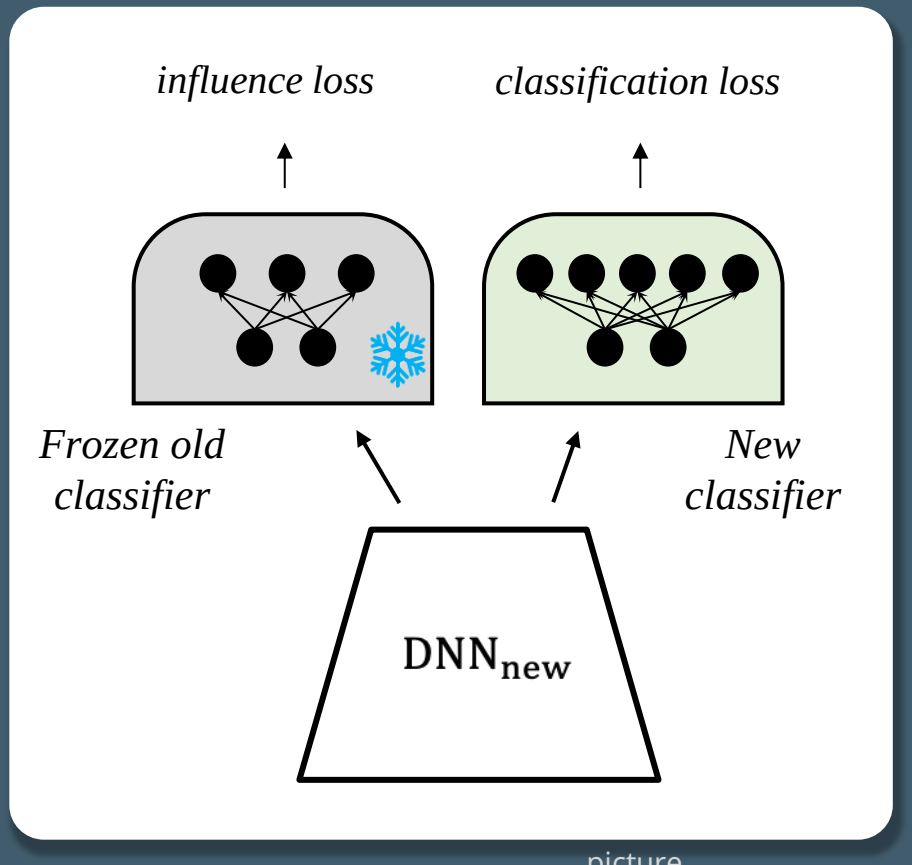
picture
authored by Y. Shen

[*] Y. Shen et al., CVPR 2020

Adds **influence loss** to the **classification loss** to bias the solution towards the old classifier

$$L_{\mathrm{BCT}}(w_c, w_\phi; \mathcal{T}_{\mathrm{new}}, \mathcal{T}_{\mathrm{BCT}}) = L(w_c, w_\phi; \mathcal{T}_{\mathrm{new}}) + \\ + \lambda L(w_{c\ \mathrm{old}}, w_\phi; \mathcal{T}_{\mathrm{BCT}})$$

- ✓ $\mathcal{T}_{\mathrm{BCT}} = \mathcal{T}_{\mathrm{old}}$ computes the influence loss only on the old training data
- ✓ $\mathcal{T}_{\mathrm{BCT}} = \mathcal{T}_{\mathrm{new}}$ computes the influence loss on both the old and new training data. For new classes, *synthesized* classifier weights are created by computing the average feature vector of $)_{\mathrm{old}}$ on the images in each class



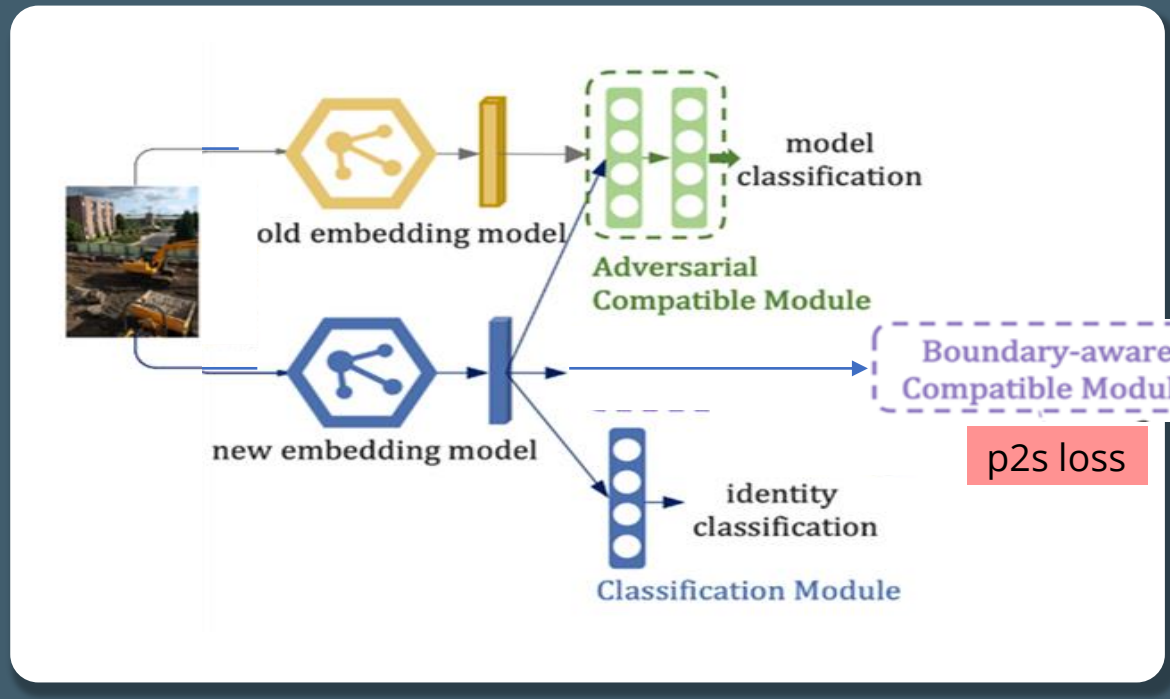
picture authored by Y. Shen

Adversarial Backward-Compatible Representation (AdvBCT) [*]

Regularization-based

AdvBCT complements the **classification loss** with **adversarial learning** and a **boundary-aware compatible module** to minimize discrepancy between the distributions of the old and new embeddings

- ✓ The **Discriminator** makes new and old models as similar as possible
- ✓ The **Boundary-aware compatible module** improves the compactness of the new feature
- ✓ The **Classification module** ensures the discrimination of new embeddings



picture authored by T.Pan

T. Pan et al. CVPR 2023

In total the **final loss** on embedding model is

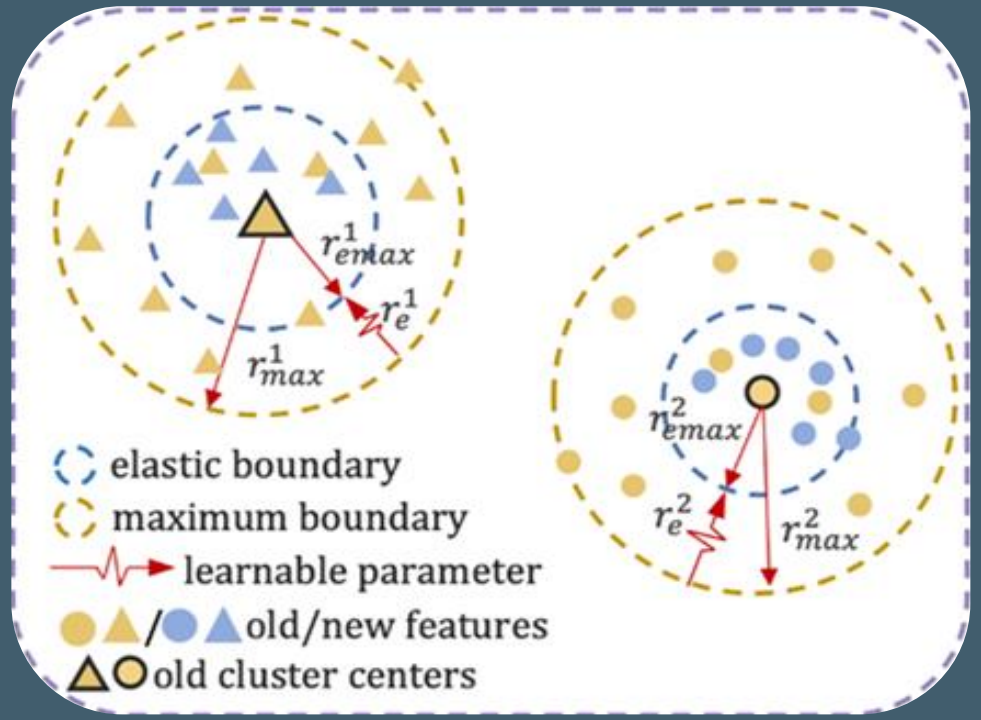
$$\mathcal{L}_{AdvBCT} = \mathcal{L}_{cls} + \lambda \mathcal{L}_{p2s} + \gamma \mathcal{L}_{adv}$$

$$\mathcal{L}_{adv} = E(\theta_n, \theta_d) = -\frac{1}{N} \sum_{i=1}^{N} \ell_i \log q_i$$

ℓ_i is the label
q_i the output of the discriminator

- ✓ **Constraints are not applied to all pairs of samples** but estimating the cluster centers and *max* and *min* boundaries
- ✓ The **point to set (p2s) loss** constrains distances between new embeddings and old cluster center in a dynamic way

Boundary-aware Compatible Module



$$\langle \phi_n(x_i), \phi_o(x_j) \rangle < \langle \phi_n(x_i), \phi_o(x_k) \rangle$$

$$\langle \phi_n(x_i), \phi_o(x_j) \rangle < \langle \phi_n(x_i), \phi_n(x_k) \rangle$$

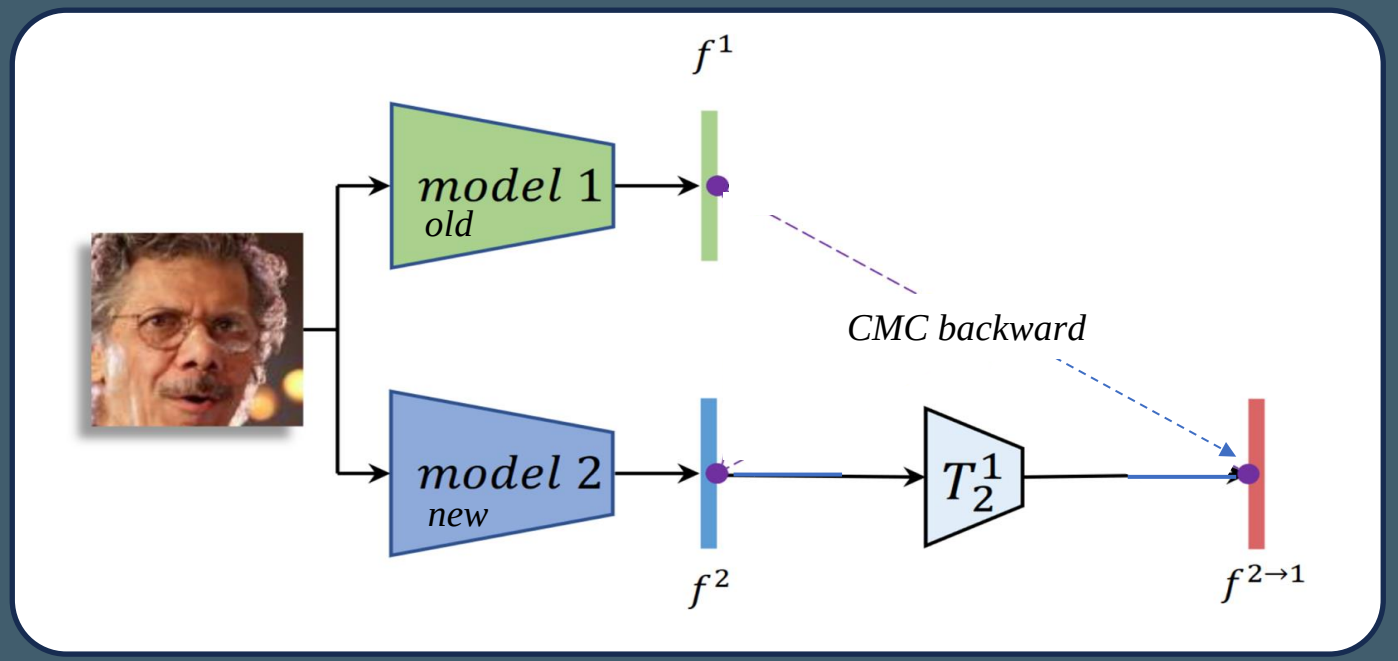
picture
authored by T.Pan

Learning Compatible Embeddings (LCE) [*]

Mapping-based

Compatibility in LCE is achieved by **learning transformations** that map embeddings and class centers from one feature space to another

The method can work in backward manner



[*] Q. Meng et al., ICCV 2021

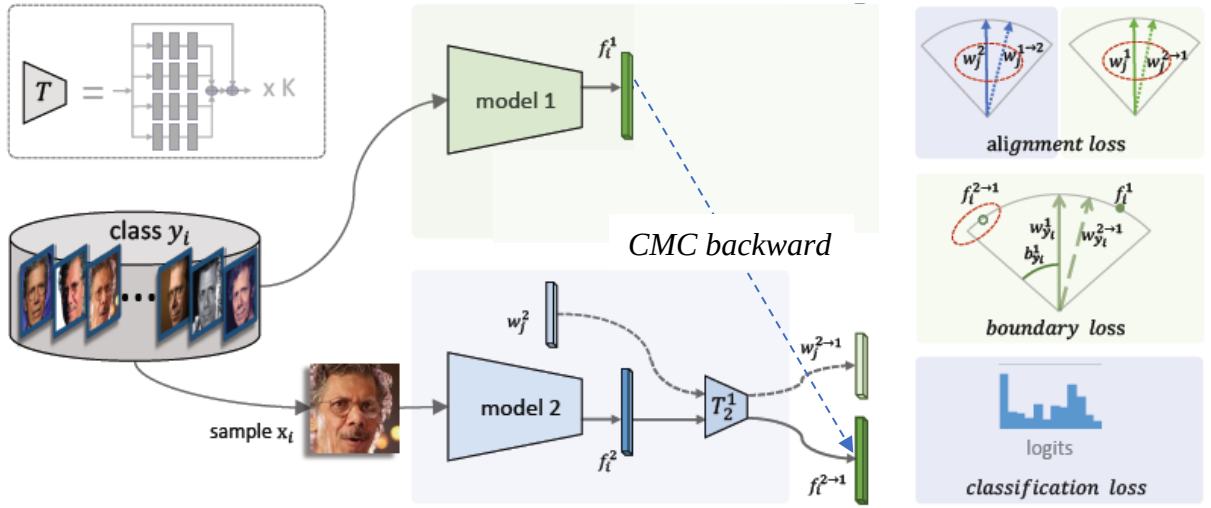
picture
authored by Q. Meng

Works in a point-to-set manner instead of employing point-wise constraints using:

- ✓ **Alignment loss** aligns class prototypes between two models
- ✓ **Boundary loss** enforces the mapped representation to have more compact intra-class distributions
- ✓ **Classification loss** to learn the data

$$L_{LCE} = \lambda_a L_a + \lambda_b L_b + L_c$$

CMC mode main drawback:
with multiple updates LCE requires the **composition of multiple mapping functions**

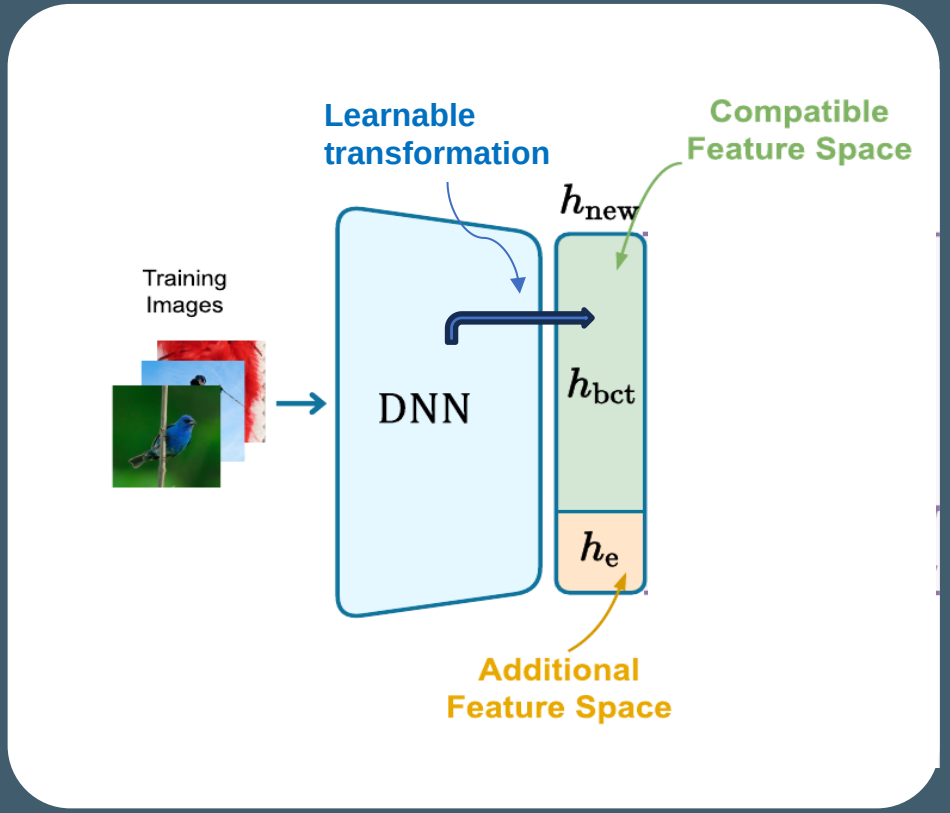


picture
authored by Q. Meng

Backward-compatible Training with Basis Transformation (BT^2) [*]

With Architectural changes

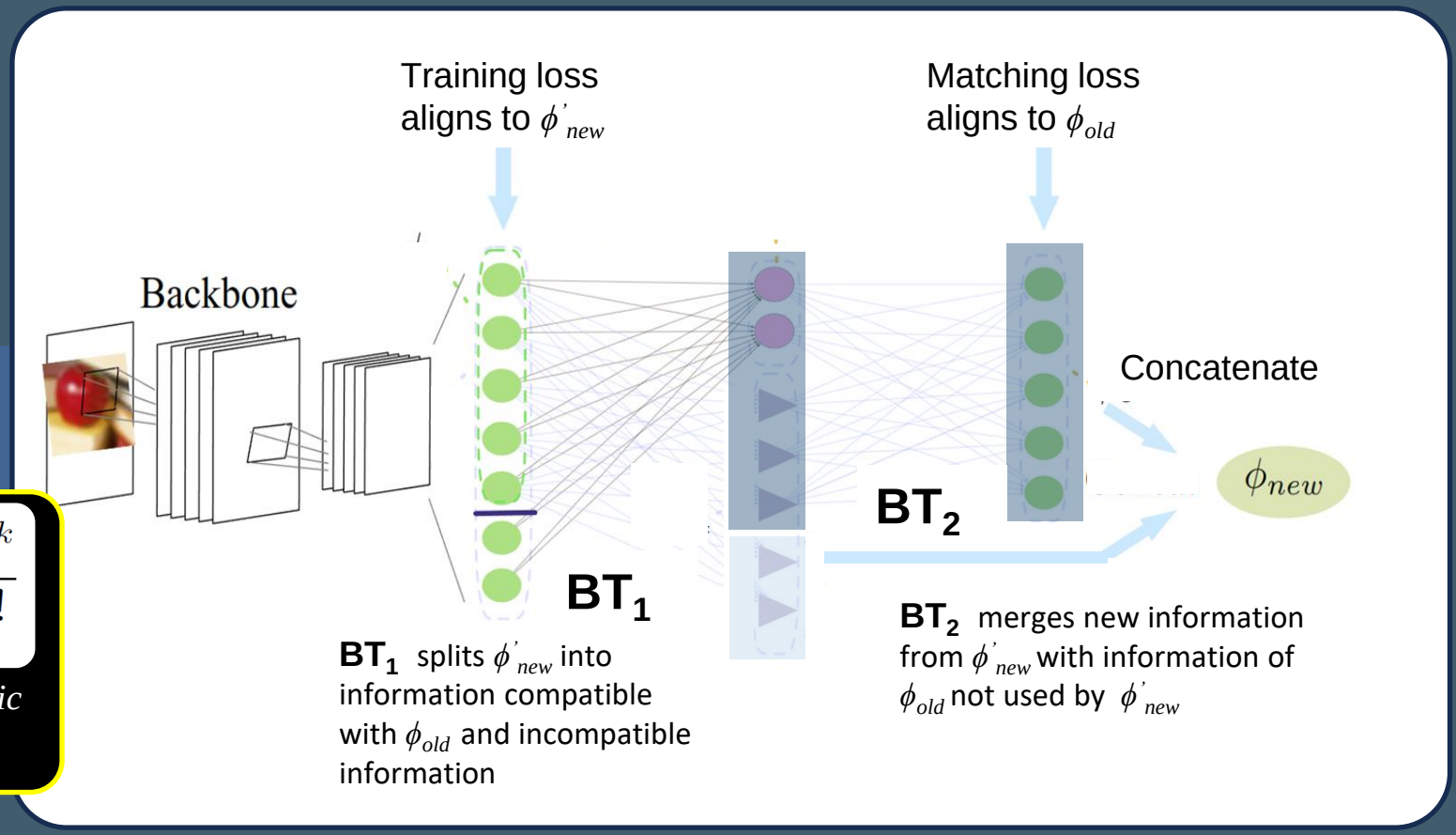
BT^2 implements BCT and mitigates the loss of performance of backward compatibility by **adding extra-dimensions** and using a **learnable transformation**



[*] Y. Zhou et al. ICCV 2023

The learnable transformation is exploited to automatically **pick out the information from ϕ'_{new} that is non compatible** with ϕ_{old}

This extra information is encoded on the **dimension orthogonal** to ϕ_{old}



$$\mathrm{BT} = e^{A} = \sum_{k=0}^{\infty} \frac{A^k}{k!}$$

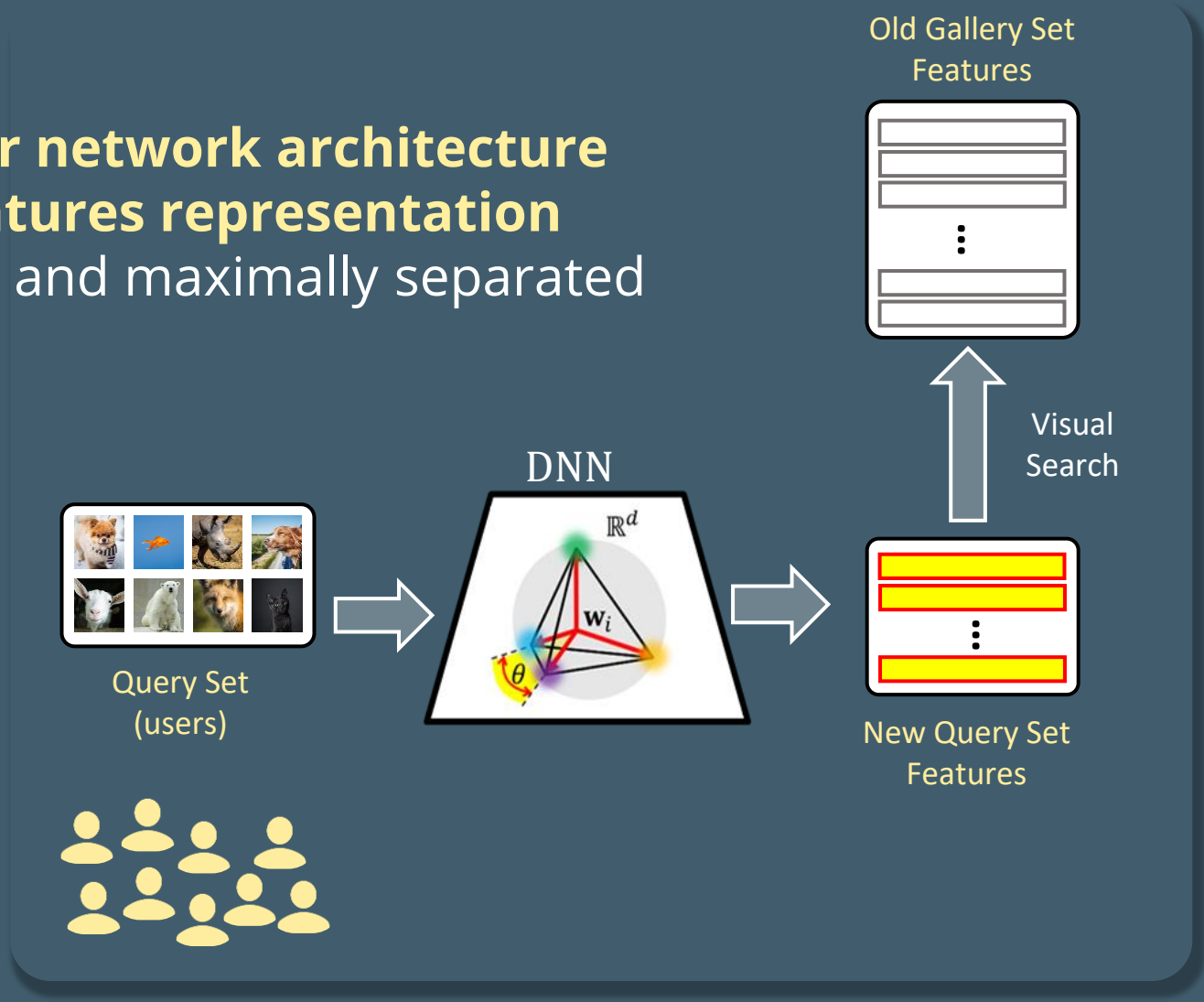
A is a left skew-symmetric matrix

Picture authored by Y. Zhou (modified)

Compatible Representation via Stationarity (CoReS) [*]

With Architectural changes

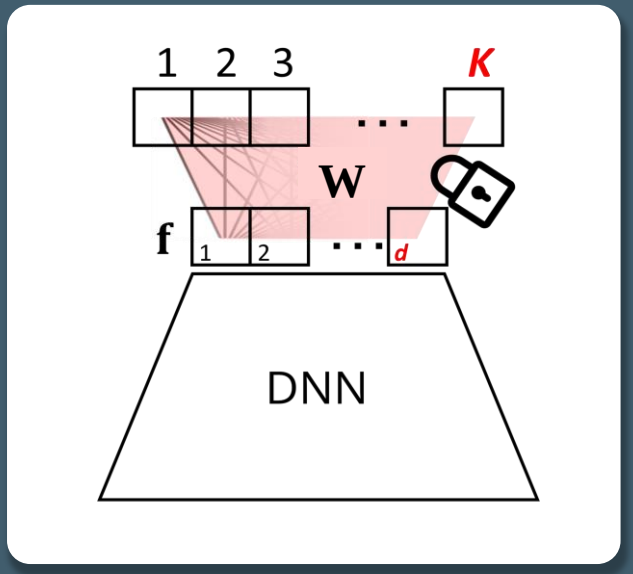
CoReS exploits the **Fixed Classifier network architecture with special configuration of features representation** to provide stationary embeddings and maximally separated representations



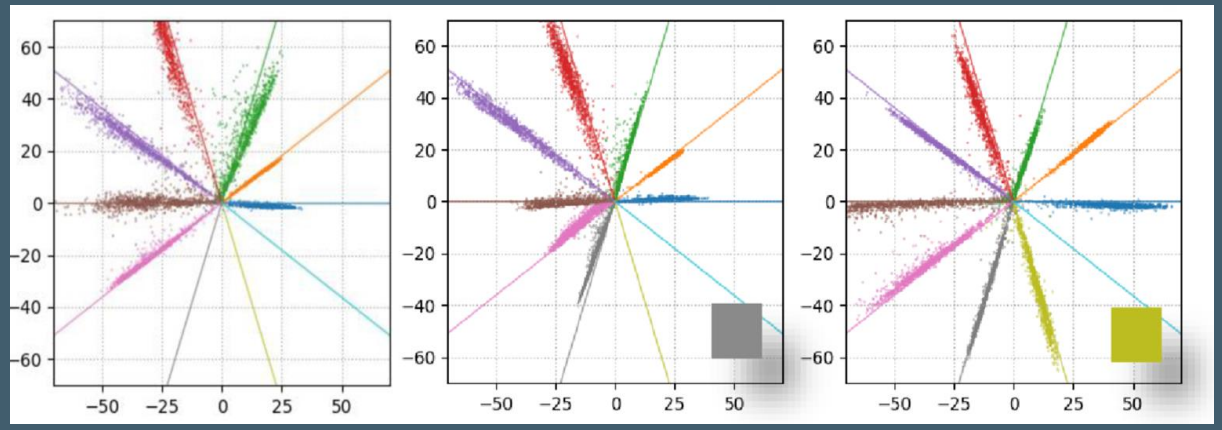
[*] N. Biondi et al., IEEE TPAMI 2023

CoReS Fixed classifiers

With **Fixed Classifiers** [*] the final layer of the CNN is **fixed** and the dynamic adjustment of the decision boundaries of class feature representations is demanded to the previous layers



Weights can be regarded **as fixed angular references** to which features align**.** This allows to define in advance where features will be projected, i.e. **enforce stationarity** over updatings

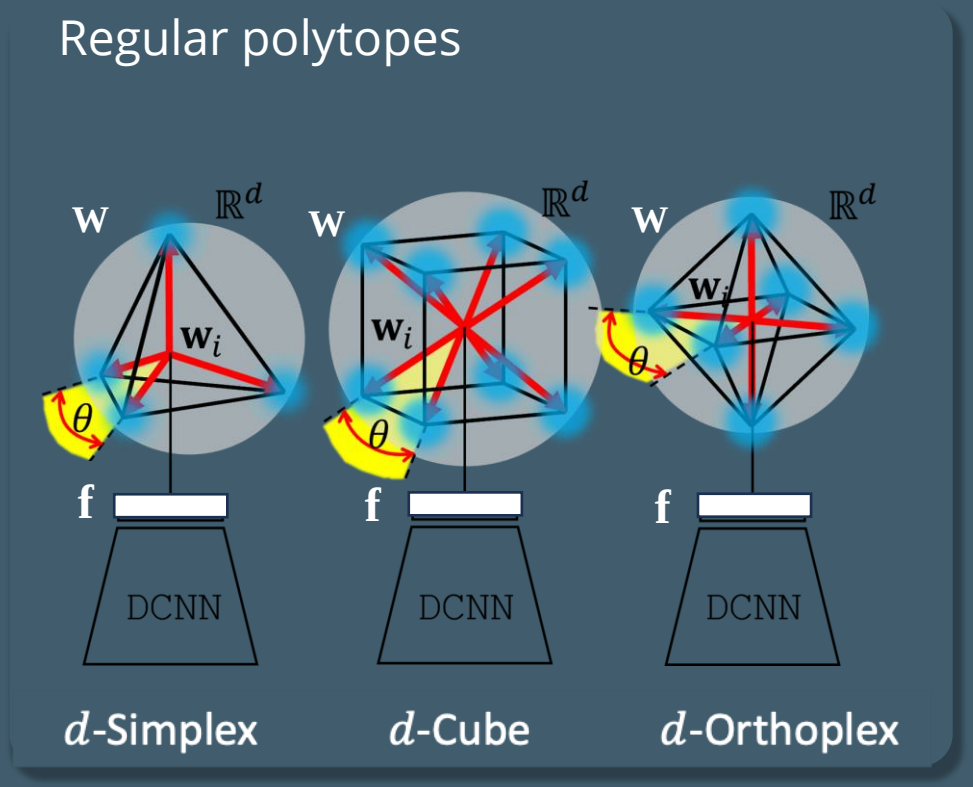


[*] E. Hoffer et al., ICLR 2018

CoReS Regular Polytopes

Maximally separated features are obtained by setting the weights of the Fixed Classifier to fixed values taken from the coordinate vertices of a **Regular Polytope**, following a highly symmetrical arrangement in the embedding space [*]

Dimension d	1	2	3	4	≥ 5
Number of Regular Polytopes	1	∞	5	6	**3**



[*] F. Pernici et al., IEEE TNNLS 2021

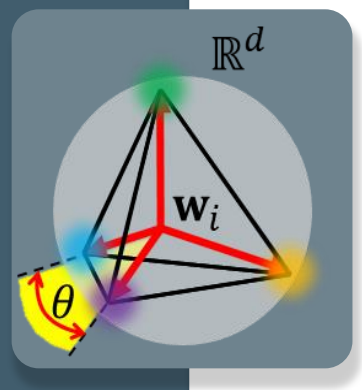
CoReS d-Simplex Fixed Classifier

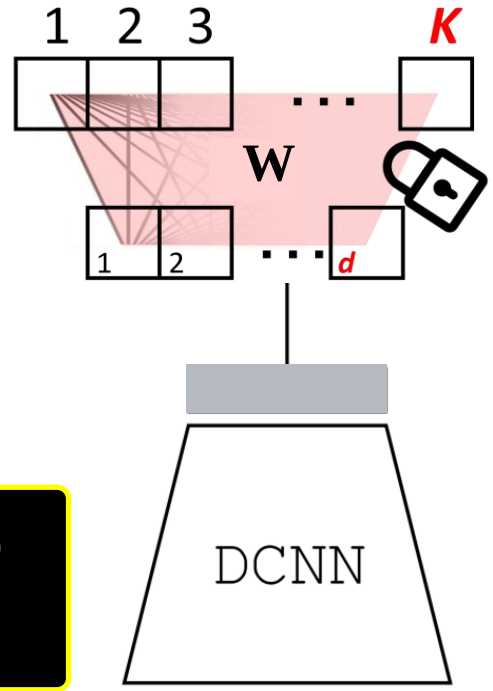
The d-**Simplex fixed classifier** of dimension d can accomodate a number of classes equal to the number of the d-Simplex vertices, i.e. $K = d + 1$

The d-Simplex weights

$$\mathbf{W} = \left[e_1, e_2, \ldots, e_{K-1}, \frac{1-\sqrt{K}}{K-1} \sum_{i=1}^{K-1} e_i \right]$$

e_i the standard basis of $\mathbb{R}^d$ with $i \in \{1, 2, \ldots, K-1\}$





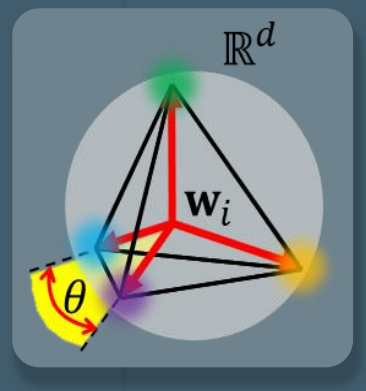
d-Simplex: same angle θ between classes

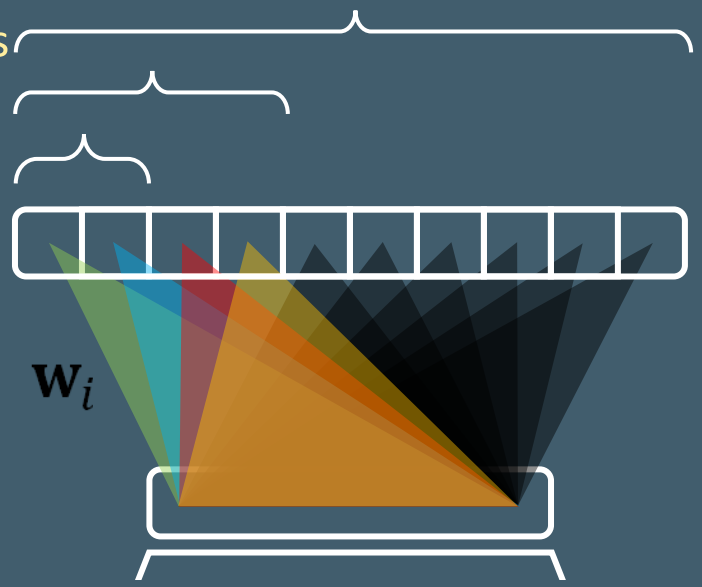
CoReS Class Pre-allocation

d-Simplex with **pre-allocated classes** allows stationarity when new classes are added with no changes to the spatial configuration of the features

No prior assumption on semantic similarity of future classes

$$\mathcal{L}_{\text{SCE}}(\phi_t) = -\sum_{B} \log \left(\frac{\exp\left(\mathbf{W}_{y_i}^{\top} \phi_t(\mathbf{x}_i)\right)}{\sum_{j=1}^{K_t} \exp\left(\mathbf{W}_j^{\top} \phi_t(\mathbf{x}_i)\right) + \sum_{j=K_t+1}^{K} \exp\left(\mathbf{W}_j^{\top} \phi_t(\mathbf{x}_i)\right)} \right)$$





Looking at the code

d-Simplex fixed classifier

Google Colab link

Jupyter Notebook link

Break 1

15’

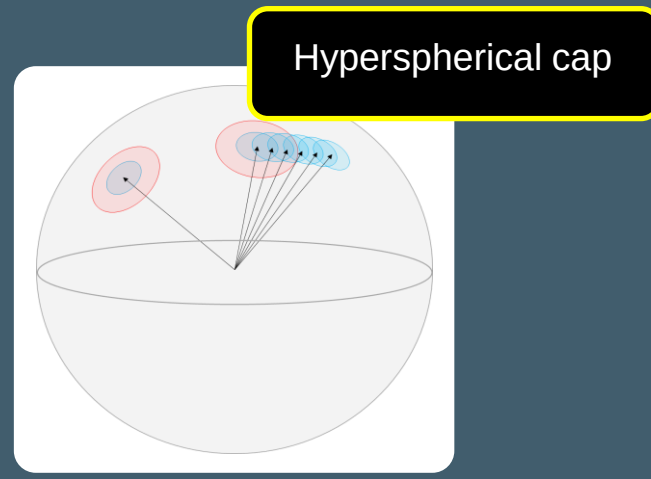
d-Simplex Stationarity Implies Compatibility

Theorem 1 (Stationarity implies Compatibility). *Let* $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K$ *be the class prototypes of a* d*-Simplex fixed classifier. Consider* ϕ_k *and* ϕ_t *as representation models learned up to the* k*-th and* t*-th tasks, respectively, within this classifier. The numbers of classes learned by each model are denoted by* K_k *and* K_t*, where* $K_k < K_t < K$*. Under the assumption that class hyperspherical caps shrink after model updates, it follows that* ϕ_k *and* ϕ_t *satisfy, on average, the compatibility inequalities as in Def. 1.*

Stationarity Implies Compatibility

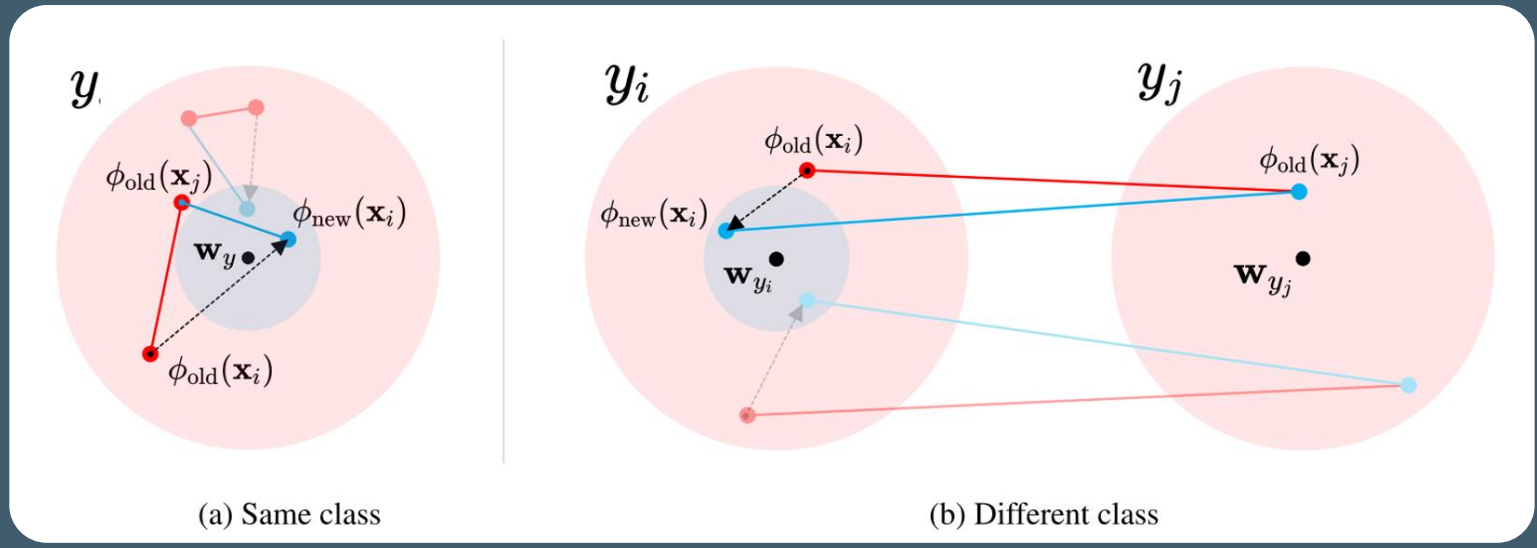
Assumptions

- ✓ Training dataset expands at each step: $\mathcal{T}_t = \mathcal{T}_k \cup \Delta\mathcal{T}$
- ✓ Class feature distribution within hyperspherical caps
- ✓ *d*-Simplex classifier with *K* pre-allocated classes
- ✓ Training shrinks hyperspherical caps (*Neural Scaling Laws*)



Conclusion

Model Φ_t and Φ_k **satisfy** the compatibility inequalities in Expectation

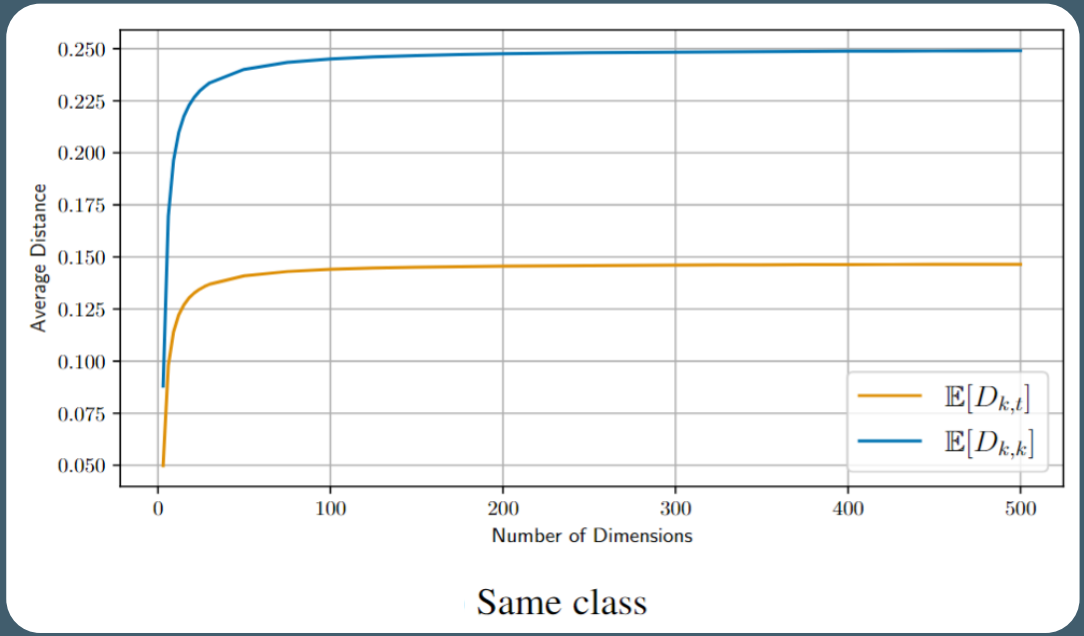


d-Simplex Implies Compatibility

A Montecarlo simulation

$$d\big(\phi_k(\mathbf{x}_i), \phi_t(\mathbf{x}_j)\big) \leq d\big(\phi_k(\mathbf{x}_i), \phi_k(\mathbf{x}_j)\big)$$

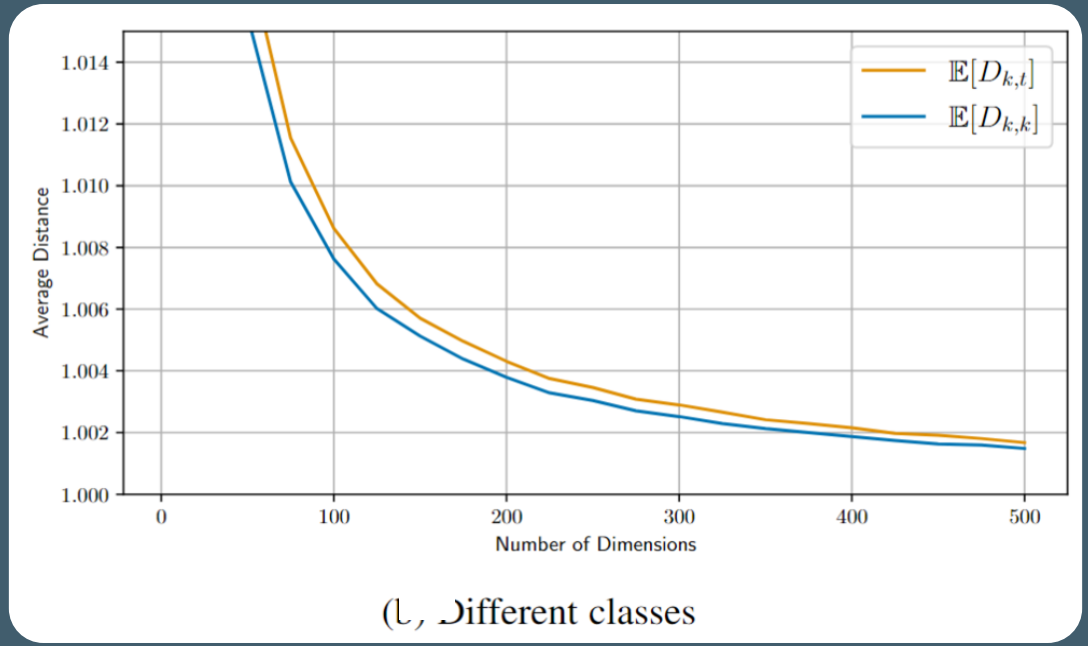
$$\mathbb{E}[D_{k,t}] \leq \mathbb{E}[D_{k,k}]$$



Same class

$$d\big(\phi_k(\mathbf{x}_i), \phi_t(\mathbf{x}_j)\big) \geq d\big(\phi_k(\mathbf{x}_i), \phi_k(\mathbf{x}_j)\big)$$

$$\mathbb{E}[D_{k,t}] \geq \mathbb{E}[D_{k,k}]$$

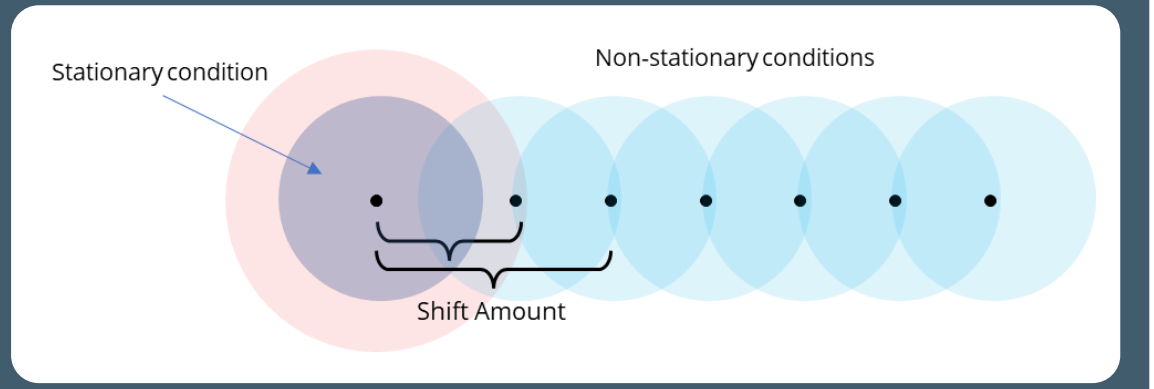


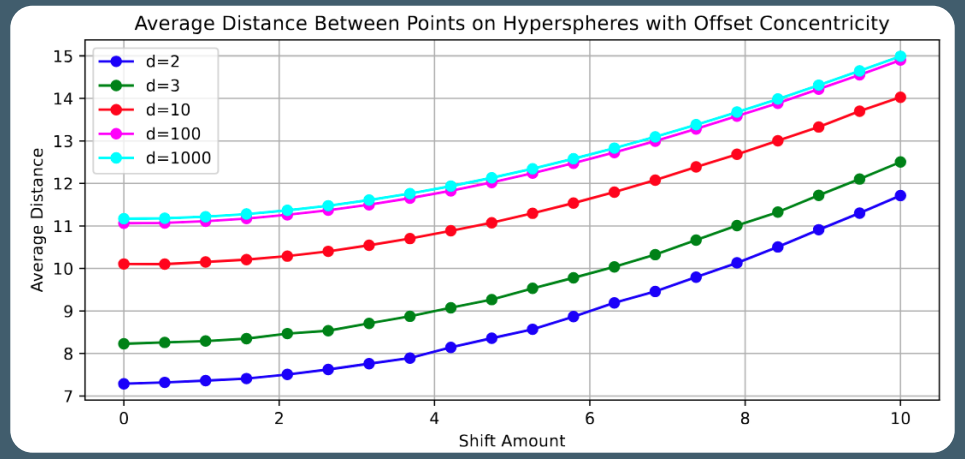
(b) Different classes

d-Simplex Implies Compatibility

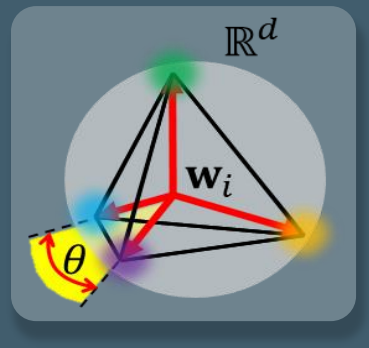
Compatibility provided by d-Simplex is optimal:

Features from the same class:
Stationarity is the optimal condition





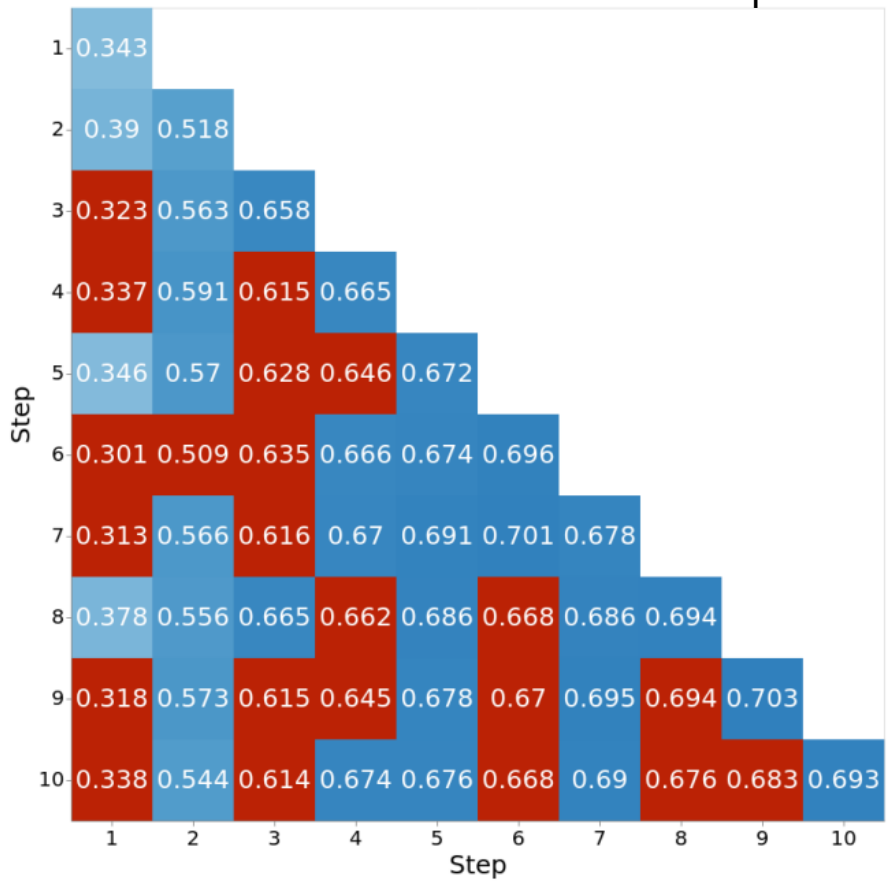
Features from different classes:
d-Simplex grants the maximum pairwise distance between classes



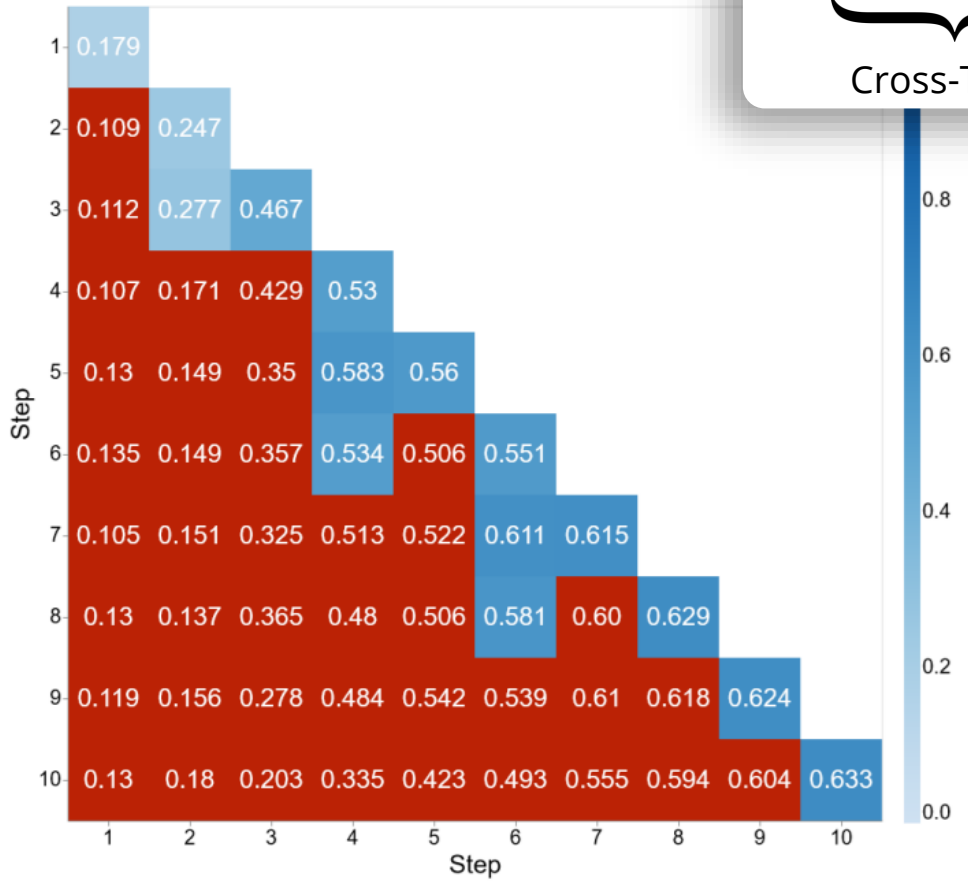
Comparative Evaluation CoReS vs BCT

CIFAR100/10 Dataset

Open set



(a) CoReS



(b) BCT [19]

[*]

$$M\underbrace{(\phi^{\mathcal{Q}}_{\text{new}}, \phi^{\mathcal{G}}_{\text{old}})}_{\text{Cross-Test}} > M\underbrace{(\phi^{\mathcal{Q}}_{\text{old}}, \phi^{\mathcal{G}}_{\text{old}})}_{\text{Self-Test}}$$

M = Verification Accuracy

[*] N. Biondi et al., IEEE TPAMI 2023

Compatibility Learning Metrics

From the Compatibility Matrix C

Average Compatibility: the normalized count of the number of times Compatibility is assessed over T steps

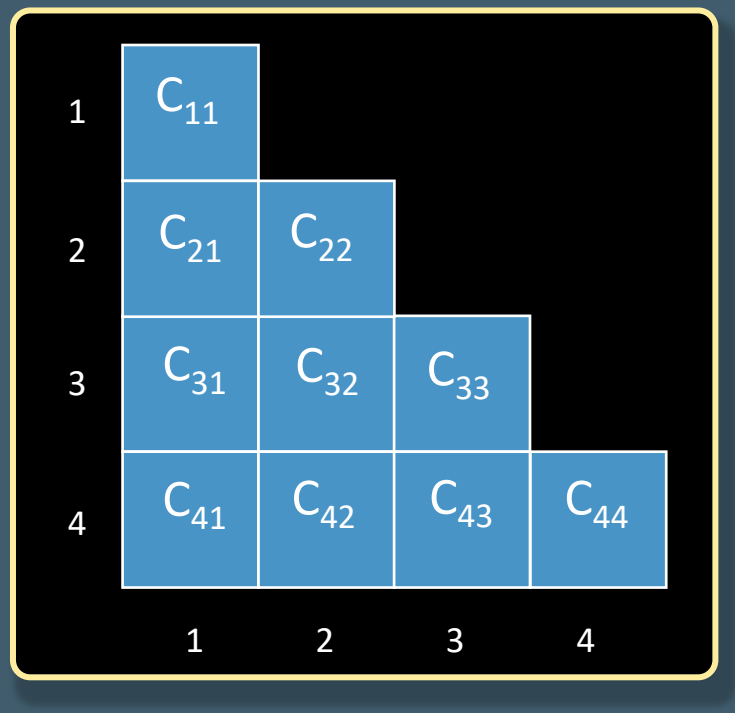
$$AC = \frac{2}{T(T-1)} \sum_{t=2}^{T} \sum_{k=1}^{t-1} \mathbb{1}\left(C_{t,k} > C_{k,k}\right),$$

Average Compatibility Accuracy:
the average measure of Accuracy assessed over tasks that satisfy the Compatibility Criterion

$$ACA = \frac{2}{T(T-1)} \sum_{t=2}^{T} \sum_{k=1}^{t-1} C_{t,k}$$
$$\text{s.t. } C_{t,k} > C_{k,k}$$

Average Accuracy: the average measure of Accuracy assessed over T steps

$$AA = \frac{2}{T(T+1)} \sum_{t=1}^{T} \sum_{k=1}^{t} C_{t,k},$$



Cifar100/10 dataset

Method	2 steps			5 steps			20 steps			50 steps		
	AC	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*
Baseline	0	29.63	0	0	13.60	0	0	04.22	0	0	02.29	0
BCT (Shen et al., 2020)	**1**	**46.40**	**50.21**	**0.40**	29.93	**11.78**	**0.13**	19.47	**05.20**	**0.01**	15.78	**0.84**
CoReS (Biondi et al., 2023)	**1**	42.49	39.88	**0.70**	**30.23**	**21.54**	**0.33**	**24.06**	**11.45**	**0.24**	**22.87**	**9.76**
ETF-CE (Yang et al., 2022)	0	38.37	0	0	27.26	0	0	20.43	0	0	**19.43**	0
ETF-DR (Yang et al., 2022)	0	36.04	0	0	24.66	0	0	18.83	0	0	16.87	0
LCE (Meng et al., 2021)	**1**	**43.48**	**40.71**	0.10	**32.63**	04.63	0	**20.95**	00.25	0	13.81	0.03
AdvBCT (Pan et al., 2023)	0	35.32	0	**0.40**	26.13	11.67	0.02	19.79	00.19	0	14.70	0.03

TinyImageNet200/ImageNet20 dataset

Method	2 steps			5 steps			20 steps			50 steps		
	AC	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*
Baseline	0	21.72	0	0	09.61	0	0	02.86	0	0	01.43	0
BCT (Shen et al., 2020)	**1**	**35.29**	**37.89**	**1**	**24.42**	**22.92**	**0.64**	**18.17**	**14.75**	**0.08**	15.41	**02.29**
CoReS (Biondi et al., 2023)	**1**	31.54	30.21	**1**	23.26	**21.55**	**0.72**	**19.25**	**16.67**	**0.55**	**17.06**	**09.79**
ETF-CE (Yang et al., 2022)	0	29.46	0	0.20	21.66	05.66	0.12	17.22	03.68	0.03	**16.14**	01.01
ETF-DR (Yang et al., 2022)	0	29.90	0	0	21.34	0	0.04	16.58	01.47	0.05	15.66	01.81
LCE (Meng et al., 2021)	**1**	**32.11**	**30.37**	0.60	**24.48**	16.92	0.02	16.51	00.90	0	11.10	0
AdvBCT (Pan et al., 2023)	0	24.90	0	**0.70**	18.99	13.93	0.26	14.65	04.41	0	09.34	0

Probability Simplex Projections [*]

Based on Softmax outputs

DNNs can achieve compatibility with no training with additional losses, mapping or modifications of the network architecture

Using the normalized softmax outputs as features for gallery and query

Features from	R@1 w R18 (ImageNet1K)	R@1 w R152 (ImageNet1K V2)	R@1 w R18 galleries and R152 queries	Compatible?
Penultimate Layer	70.22	75.36	00.10	No
Logits	51.56	79.29	60.53	Yes
Softmax Probability Outputs	63.39	78.29	76.15	Yes

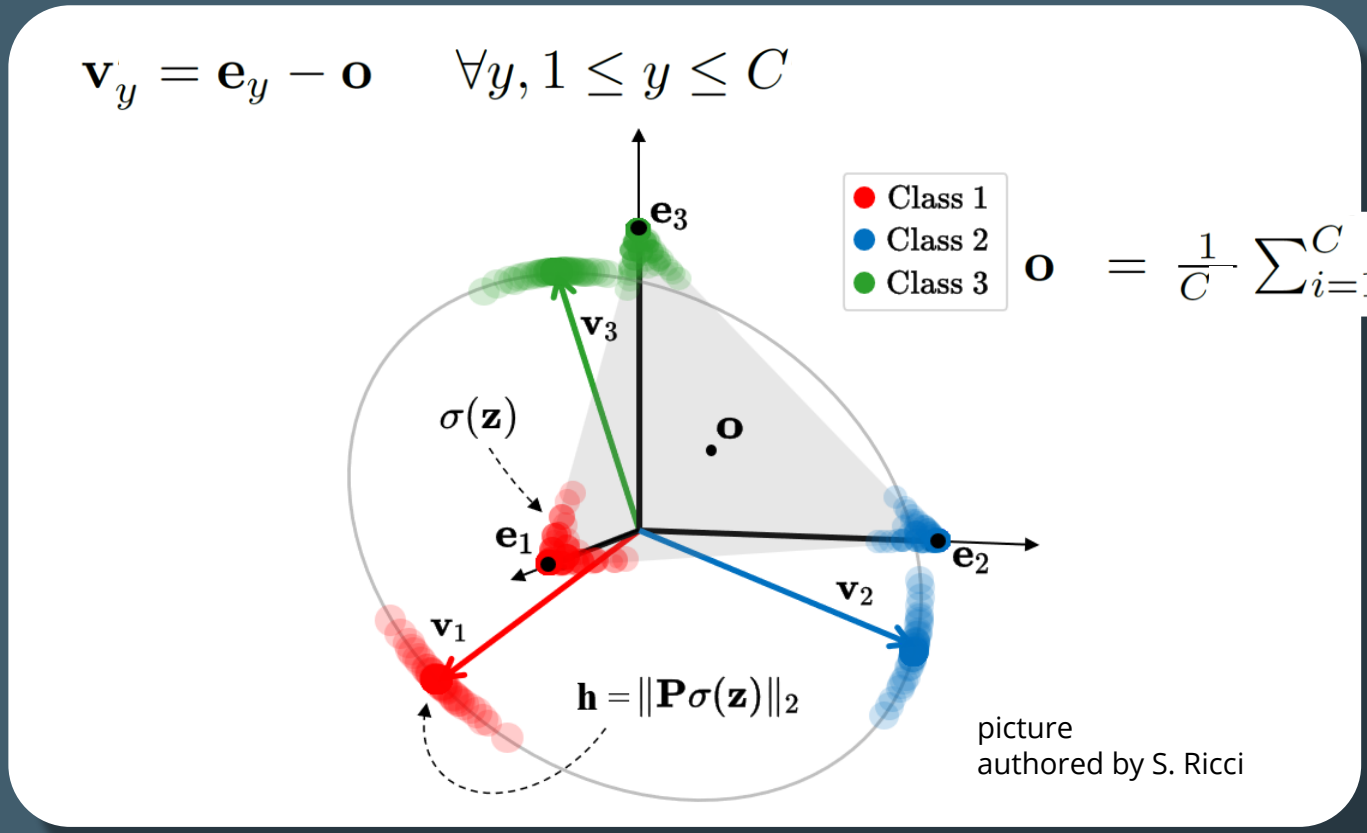
[*] S. Ricci et al., ICLR 2025 (sub)

Probability Simplex Projections [*]

Probability Simplex (grey area) is a geometrical configuration such that its vertices $\boldsymbol{e}_i$ are the standard basis of the representation space and components of its points sum up to 1

Softmax probabilities converge to the **Probability Simplex** vertices

Normalization of softmax outputs projects softmax outputs onto the unit hypersphere of the representation space

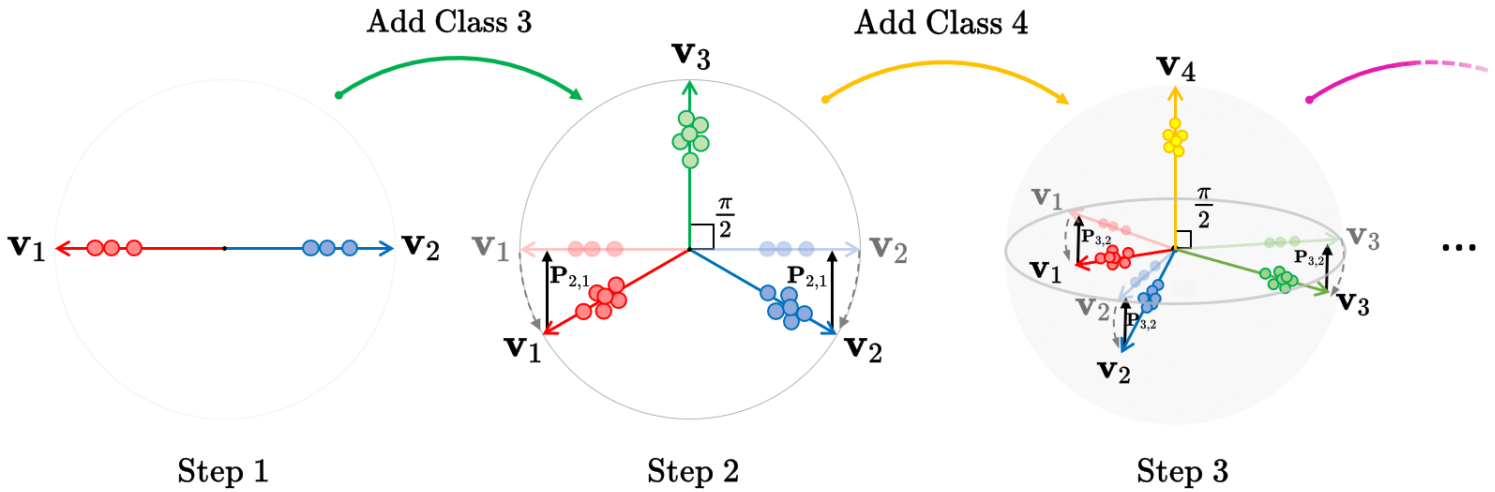


[*] S. Ricci et al., ICLR 2025 (sub)

Probability Simplex Projections are Stationary

Softmax features keep **stationary** during updates up to a projection

The class prototypes of the new classes are **orthogonal** with respect to the class prototypes from the previous step



$$\mathbf{V}_2 = \begin{bmatrix} \mathbf{e}_1 - \mathbf{o} \\ \mathbf{e}_2 - \mathbf{o} \end{bmatrix} = \begin{bmatrix} 1 - \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 1 - \frac{1}{2} \end{bmatrix}$$

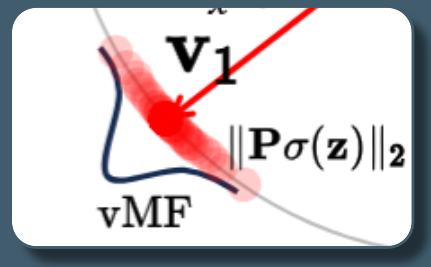
$$\mathbf{P}_{3,2} = [\mathbf{V}_2 | \mathbf{0}] = \begin{bmatrix} 1 - \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 1 - \frac{1}{2} & 0 \end{bmatrix}$$

Probability Simplex Compatibility Theorem

Theorem (***Probability Simplex Compatibility Theorem***). *Consider two models trained independently, the first with C^k classes at step k and the second with C^t classes at step t, where $C^t \geq C^k$. Assuming that the normalized softmax features of each class have a von Mises-Fisher distribution on the hypersphere, with the distribution at step t having a higher concentration around the mean than at step k, it follows that the two independently trained models satisfy the Compatibility inequalities.*

Theorem Proof

Features of each class **approximate the von Mises Fisher (vMF) distribution** on the hypersphere $\mathbf{X} \sim \text{vMF}(\boldsymbol{\mu}_1, \kappa_1), \quad \mathbf{Y} \sim \text{vMF}(\boldsymbol{\mu}_2, \kappa_2) \quad \text{with } \mathbf{X}, \mathbf{Y} \in S^{d-1}$



The verification of compatibility can be **analytically determined** according to a **closed-form solution** $\mathbb{E}[1 - \cos\theta] = 1 - \mathbb{E}\left[\mathbf{X}^\top \mathbf{Y}\right] = 1 - m_d\left(\kappa_1\right) m_d\left(\kappa_2\right) \cos\alpha \quad \text{with } \cos\alpha = \boldsymbol{\mu}_1^\top \boldsymbol{\mu}_2$

where $m_d(\kappa) \overset{\text{def}}{=} \dfrac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)}$

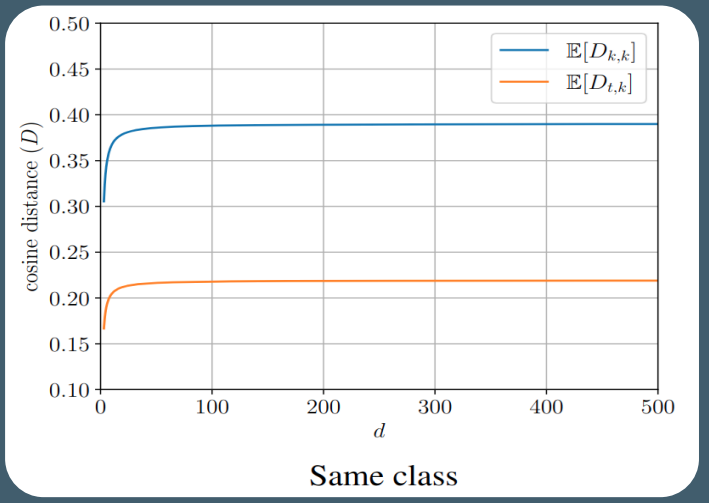
The expected cosine distance between features **X** and **Y** is a function of the angle between the mean of the vMF distributions (representing class prototypes) and their concentrations k_1, k_2

The vMF distribution of the updated model's class features have a **higher concentration**:

$k_2 > k_1 \Rightarrow m_d(k_2) > m_d(k_1)$

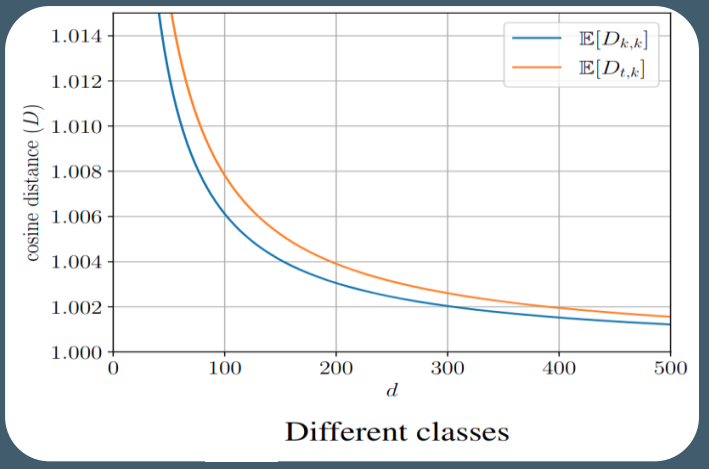
$$\mathbb{E}[1 - \cos\theta] = 1 - m_d(\kappa_1)\, m_d(\kappa_2) \cos\alpha$$

Features of the **same class** have stationarity prototypes (i.e., cos α = 1) ⇒ the expected cosine distance between features **X** and **Y** **decreases**



Same class

For features from **different classes** the angle between prototypes is always greater than π/2 (i.e., cos α < 0) ⇒ the expected cosine distance between features **X** and **Y** **increases**

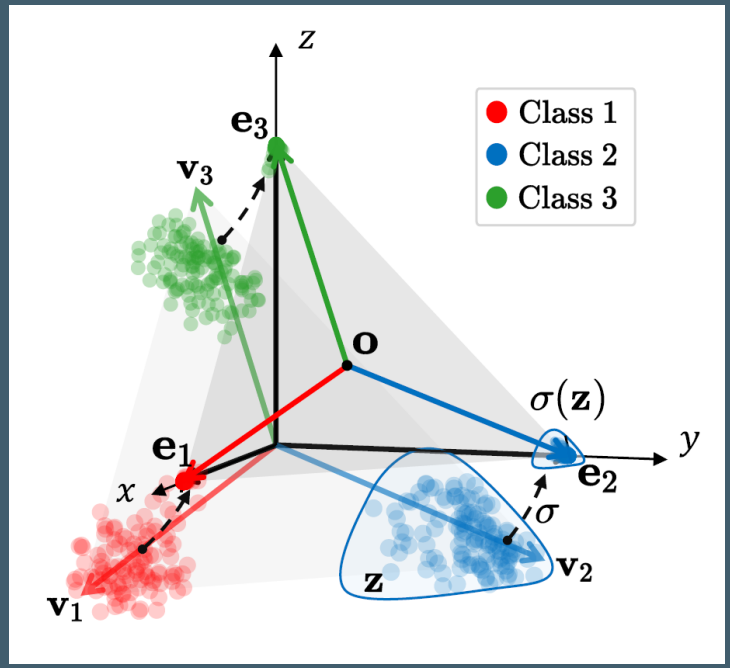


Different classes

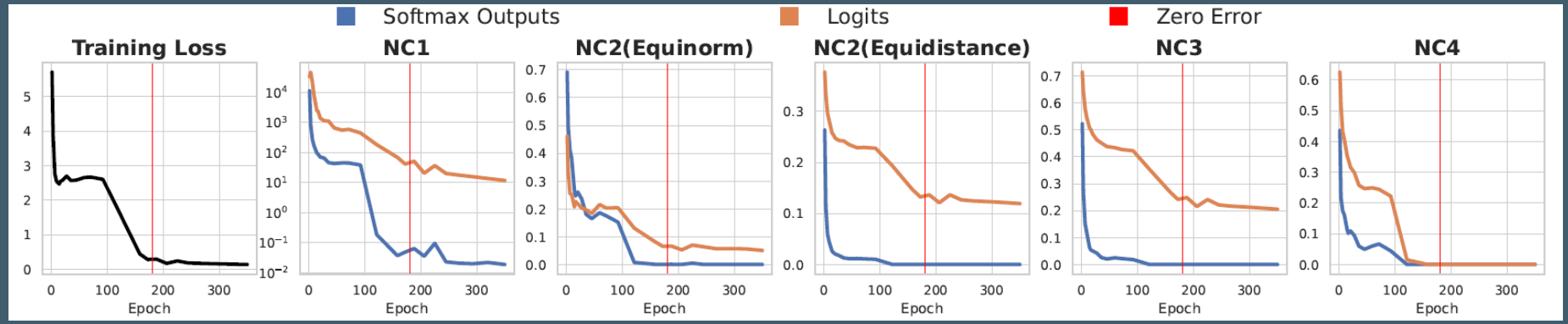
Logits Simplex Projections (LSP)

During training also logits configure in a simplex with the same prototype vectors …

Proposition 3. *As the softmax outputs approach the vertices of the Probability Simplex, the corresponding logits vectors assume a simplex configuration, with class prototypes aligning to the vectors specified in Eq. 3.*



… with more spread in the class features distribution
→ this leads to a more robust and generalizable representation



Cifar100/10 dataset

	2 steps			5 steps			20 steps			50 steps		
METHOD	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*
Baseline	0	29.63	0	0	13.60	0	0	04.22	0	0	02.29	0
BCT (Shen et al., 2020)	**1**	**46.40**	**50.21**	0.40	29.93	11.78	0.13	19.47	05.20	0.01	15.78	0.84
CoReS (Biondi et al., 2023)	**1**	42.49	39.88	**0.70**	30.23	21.54	0.33	**24.06**	11.45	0.24	**22.87**	9.76
ETF-CE (Yang et al., 2022)	0	38.37	0	0	27.26	0	0	20.43	0	0	19.43	0
ETF-DR (Yang et al., 2022)	0	36.04	0	0	24.66	0	0	18.83	0	0	16.87	0
LCE (Meng et al., 2021)	**1**	**43.48**	**40.71**	0.10	**32.63**	04.63	0	20.95	00.25	0	13.81	0.03
AdvBCT (Pan et al., 2023)	0	35.32	0	0.40	26.13	11.67	0.02	19.79	00.19	0	14.70	0.03
PSP	**1**	36.31	29.05	**0.90**	26.04	**21.76**	**0.52**	20.56	**12.99**	**0.39**	19.42	**10.76**
LSP	**1**	41.14	36.38	**0.70**	**30.36**	**21.91**	**0.44**	**24.11**	**13.26**	**0.36**	**22.68**	**11.25**

TinyImageNet200/ImageNet20 dataset

	2 steps			5 steps			20 steps			50 steps		
METHOD	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*
Baseline	0	21.72	0	0	09.61	0	0	02.86	0	0	01.43	0
BCT (Shen et al., 2020)	**1**	**35.29**	**37.89**	**1**	24.42	**22.92**	0.64	18.17	14.75	0.08	15.41	02.29
CoReS (Biondi et al., 2023)	**1**	31.54	30.21	**1**	23.26	21.55	0.72	**19.25**	16.67	0.55	**17.06**	09.79
ETF-CE (Yang et al., 2022)	0	29.46	0	0.20	21.66	05.66	0.12	17.22	03.68	0.03	16.14	01.01
ETF-DR (Yang et al., 2022)	0	29.90	0	0	21.34	0	0.04	16.58	01.47	0.05	15.66	01.81
LCE (Meng et al., 2021)	**1**	32.11	**30.37**	0.60	**24.48**	16.92	0.02	16.51	00.90	0	11.10	0
AdvBCT (Pan et al., 2023)	0	24.90	0	0.70	18.99	13.93	0.26	14.65	04.41	0	09.34	0
PSP	**1**	29.88	25.05	**0.90**	21.93	17.99	**0.91**	17.53	**15.33**	**0.90**	16.63	**14.95**
LSP	**1**	**32.44**	29.26	**1**	**25.10**	**23.48**	**0.83**	**20.46**	**17.34**	**0.80**	**19.48**	**16.01**

[*] S. Ricci et al., ICLR 2025 (sub)

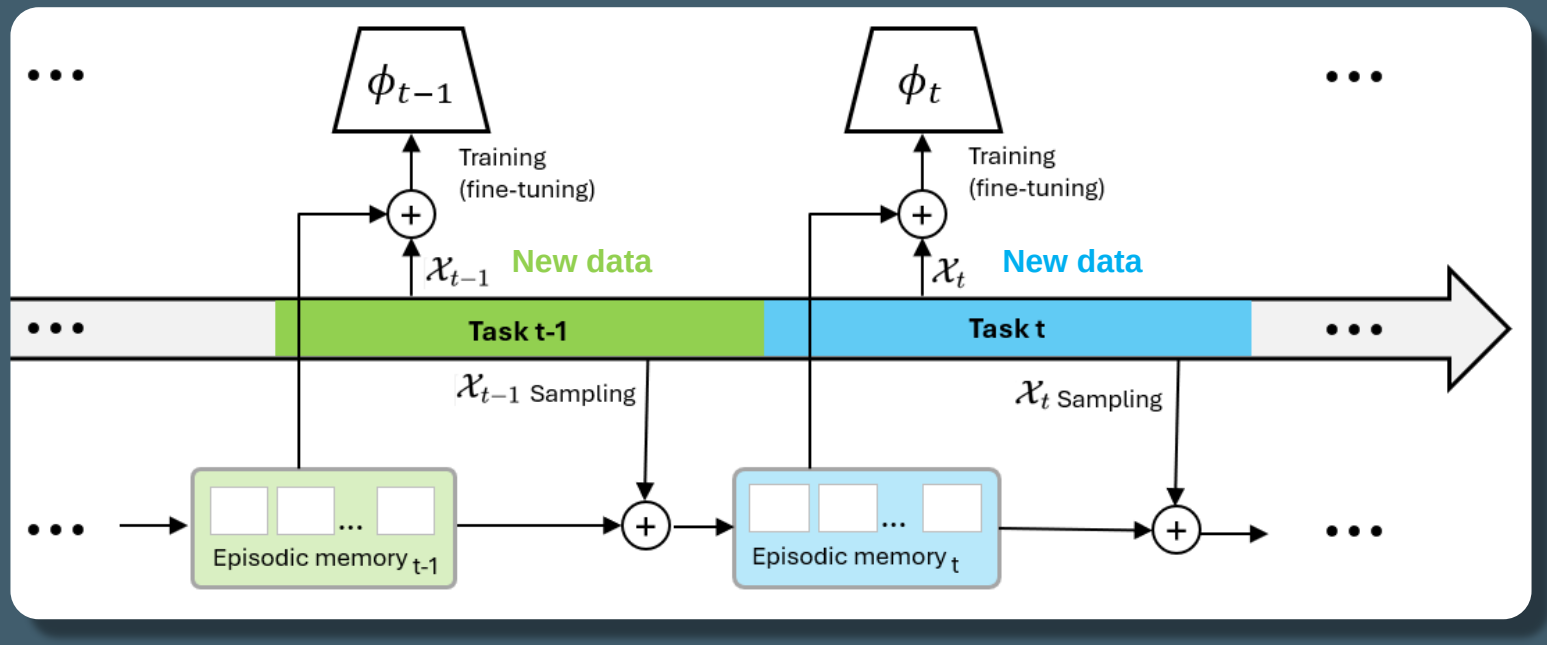
Compatibility

Issues and Solutions

Compatibility in Continual Learning

Compatibility training using old and new data **is time-consuming** as **t**raining datasets are larger and larger and models more and more complex.
It **might be unfeasible** if previous training data are no more available

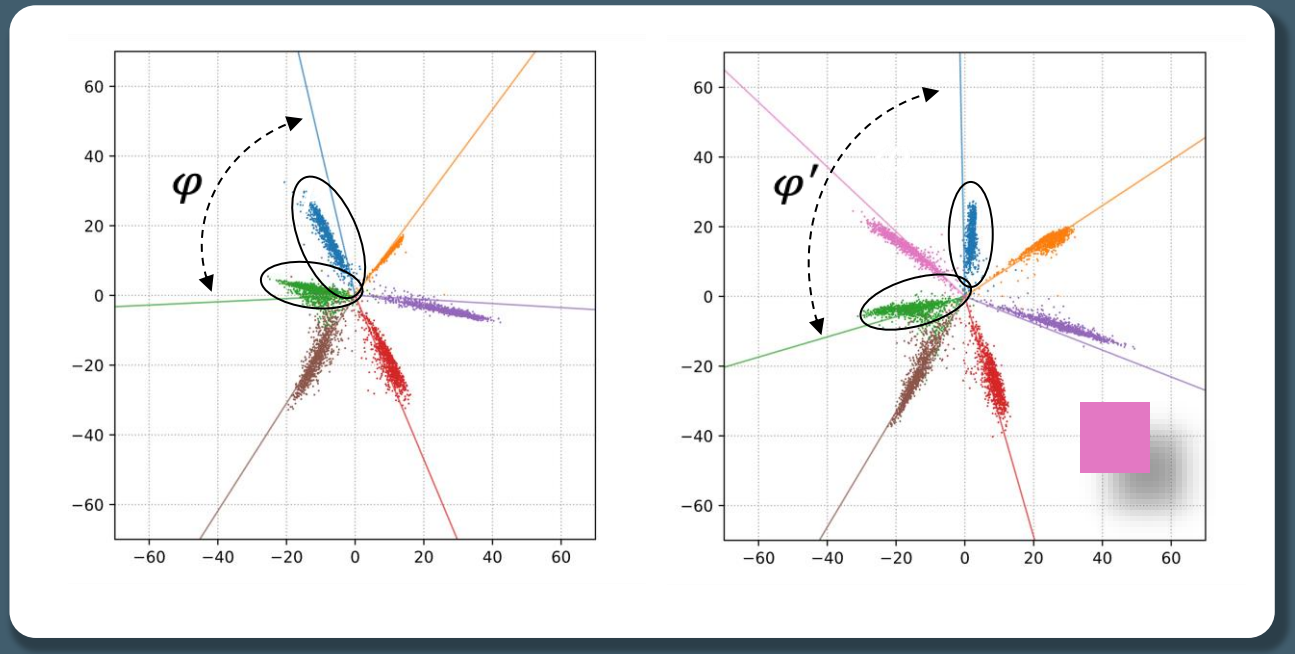
Compatibility in Continual Learning: learning compatible representation where at each step the model is trained using **only the new data** and a **small replay memory**



picture
authored by N. Biondi

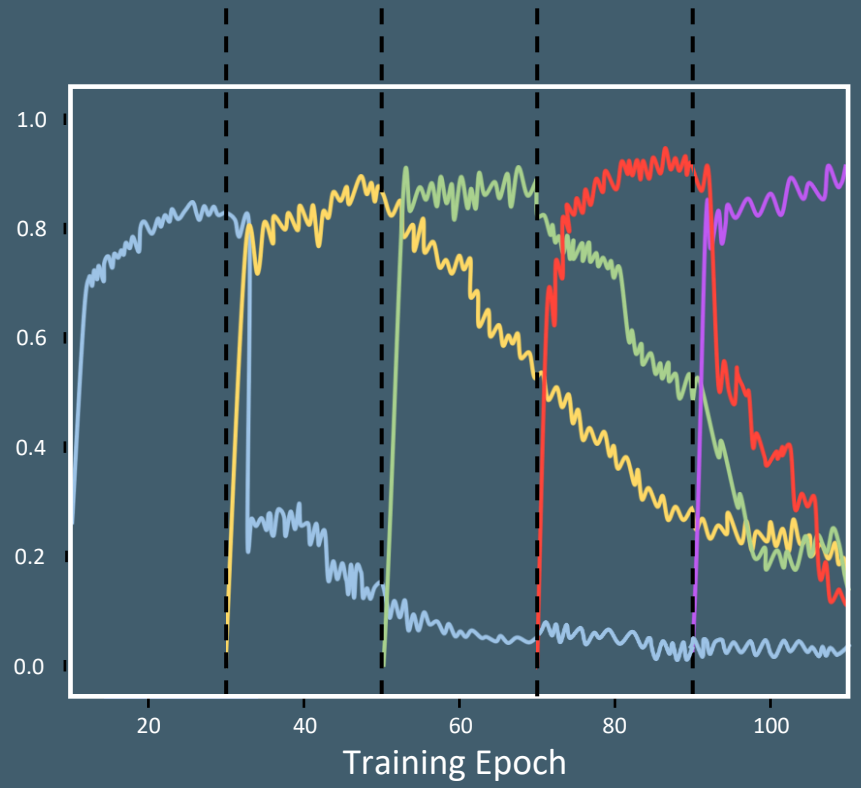
Main Issues

Updating a DNN model with novel classes changes the internal feature representation



$\varphi \neq \varphi'$

Performance degradation occurs when new knowledge is assimilated (*catastrophic forgetting*)



pictures
authored by F. Pernici and N. Biondi

Proposals for Solutions

Regularization-based approaches:
Use distillation and experience replay to mitigate forgetting and additional losses to preserve Compatibility at model upgrading

- ✓ CVS (CVPR2023)
- ✓ C^2R (CVPR2024)

Mapping-based approaches:
Use distillation and experience replay to mitigate forgetting and finds compatible mappings between the previous and upgraded model

- ✓ FAN (ECCV2020)

Approaches with **architectural changes**:
Use distillation and experience replay to mitigate forgetting and impose constrains on the network architecture that force feature alignment

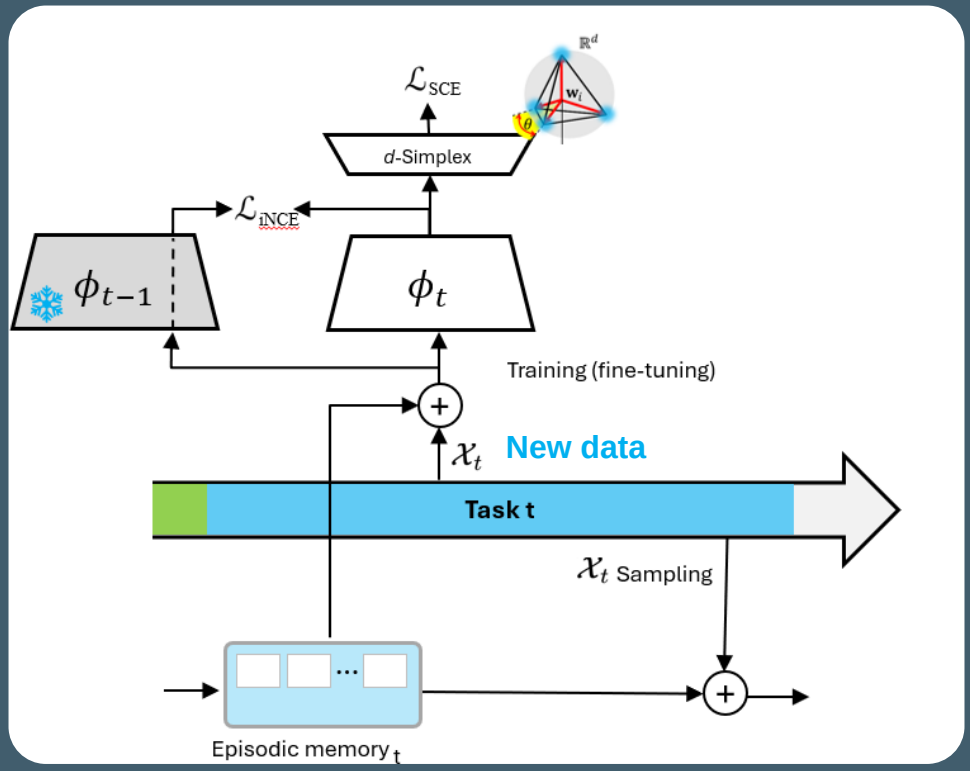
- ✓ d-Simplex-FD (ACM TOMM2023)
- ✓ d-Simplex-HOC (CVPR2024)

d-Simplex with Higher Order Compatibility Loss (d-Simplex HOC)[*]

With Architectural changes

Training according to d-Simplex using Cross-Entropy loss aligns features of each class to the first-order moment of their distribution. Cannot capture **higher-order dependencies** between model updates

- ✓ **d-Simplex classifier** to keep feature geometric configuration stable during model updates
- ✓ **Higher-Order Compatibility (HOC) loss** combines Cross-Entropy loss and Contrastive loss



[*] N. Biondi et al., CVPR 2024

HOC loss

$$\mathcal{L}_{\text{HOC}}(\phi_t) = \lambda\,\mathcal{L}_{\text{SCE}}(\phi_t) + (1-\lambda)\,\mathcal{L}_{i\text{NCE}}(\phi_t, \phi_{t-1}), \quad \lambda \in [0,1]$$

$$\mathcal{L}_{i\text{NCE}}(\phi_t, \phi_{t-1}) = -\sum_{B} \log\left(\frac{\Delta\big(\phi_{t-1}(\mathbf{x}_i), \phi_t(\mathbf{x}_i)\big)}{\sum_{j\neq i}\Delta\big(\phi_{t-1}(\mathbf{x}_i), \phi_t(\mathbf{x}_j)\big)}\right)$$

where $\Delta\big(\phi_{t-1}(\mathbf{x}_i), \phi_t(\mathbf{x}_j)\big) = \exp\left(\rho\ \frac{\phi_{t-1}(\mathbf{x}_i)\phi_t(\mathbf{x}_j)}{||\phi_{t-1}(\mathbf{x}_i)||\,||\phi_t(\mathbf{x}_j))||}\right)$

A **convex combination** of the Cross-Entropy loss and a Contrastive loss

Contrastive loss based on the ρ-scaled cosine similarity between $\varphi_{t-1}(\mathbf{x}_i)$ and $\varphi_t(\mathbf{x}_j)$

The Contrastive loss approximates the **mutual information** between ϕ_t and ϕ_{t-1}
It captures **higher-order dependencies** in the representation between model updates

Looking at the code

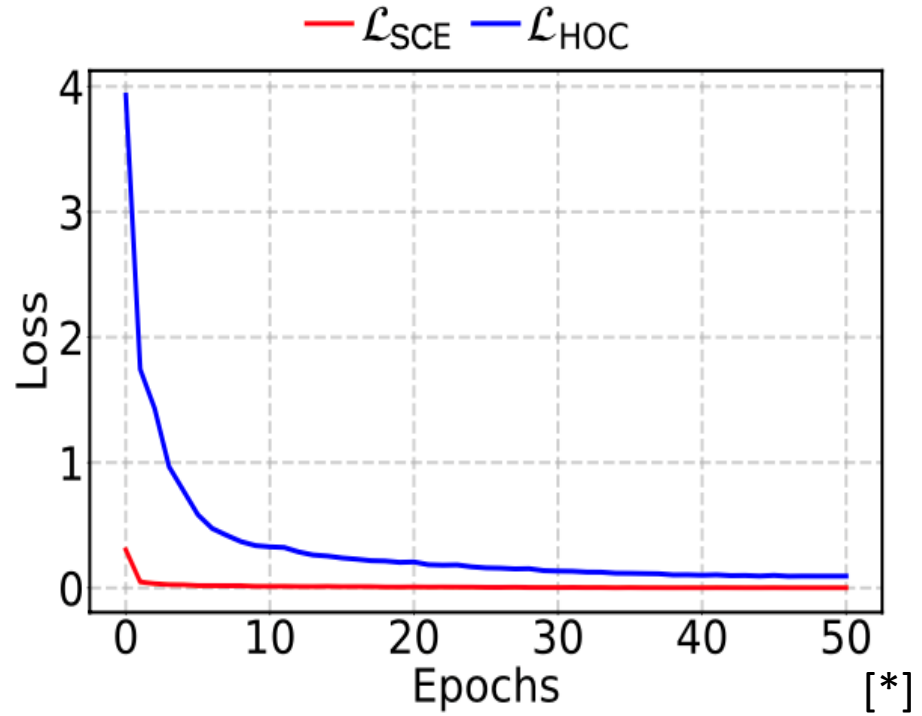
Effects of HOC loss

Google Colab link

Jupyter Notebook link

HOC Loss vs Cross Entropy Loss

MNIST dataset



LeNet++ trained with the d-Simplex fixed classifier on
5 MNIST classes and fine-tuned with the other 5 classes

Rapid convergence when using Cross-Entropy loss may lead to the **model's inability to learn higher-order dependencies** between consecutive model updates

[*] N. Biondi et al., IEEE TPAMI 2024 (sub)

HOC Loss Equivalence with Compatibility

While capturing higher-order dependencies, the HOC loss **preserves compatibility** between learned representations

Proposition 1 (Loss Equivalence). *Let ϕ_t, ϕ_{t-1} be two representation models learned according to a d-Simplex fixed classifier where the updated model ϕ_t is obtained by fine-tuning ϕ_{t-1}. Then, training ϕ_t with the HOC loss is equivalent to training ϕ_t using cross-entropy loss under compatibility constraints.* [*]

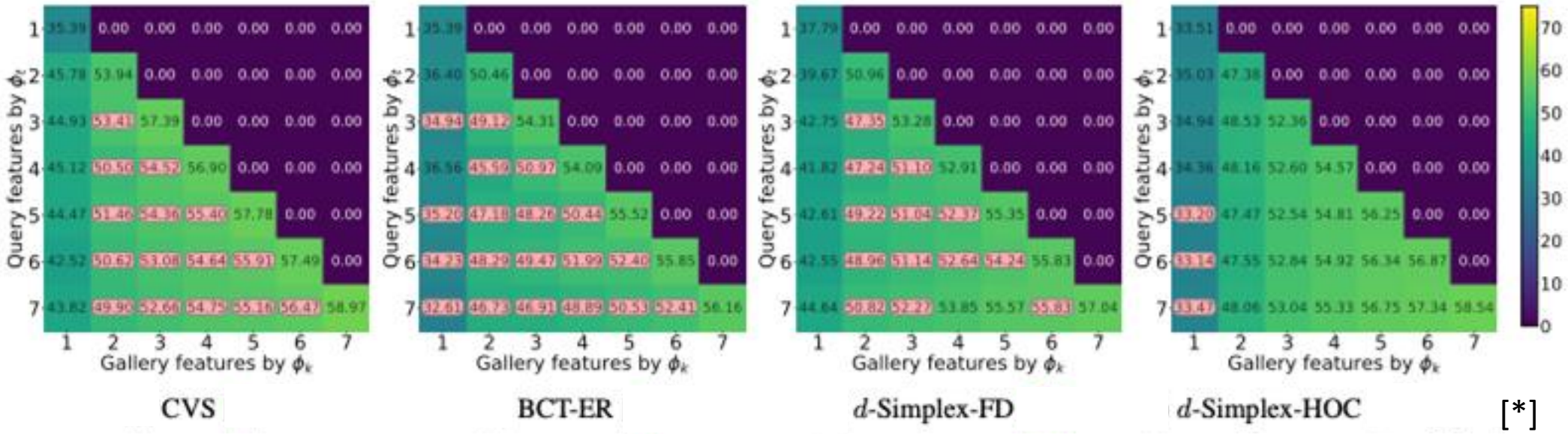
[*] N. Biondi et al., CVPR 2024

Comparative Evaluation

ResNet32
CIFAR100/10 Dataset

$$M\underbrace{\left(\phi^{\mathcal{Q}}_{\text{new}},\ \phi^{\mathcal{G}}_{\text{old}}\right)}_{\text{Cross-Test}} > M\underbrace{\left(\phi^{\mathcal{Q}}_{\text{old}},\ \phi^{\mathcal{G}}_{\text{old}}\right)}_{\text{Self-Test}}$$

open set



[*]

[*] N. Biondi et al., IEEE TPAMI 2024 (sub)

Compatible Continual Learning

ResNet32
CIFAR100/10 dataset

100 training classes: init 10 classes [*]

Method	2 tasks			7 tasks			16 tasks			31 tasks		
	AC	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*
ER baseline	1.00	46.67	39.25	0.19	46.75	7.79	0.04	44.91	1.69	0.02	40.78	0.93
FAN [19]	1.00	48.37	43.87	0.14	47.59	5.63	0.15	46.10	6.69	0.07	43.12	3.42
BCT-ER	1.00	47.11	41.14	0.10	46.82	3.48	×	43.63	×	0.01	41.96	0.40
LCE-ER	1.00	49.78	50.02	0.24	45.15	10.38	0.28	41.98	12.61	0.14	38.45	5.93
AdvBCT-ER	1.00	46.28	39.71	×	35.66	×	0.01	34.89	0.39	0.02	33.93	1.04
CVS [7]	1.00	**51.31**	**52.89**	0.29	**51.69**	12.70	0.13	**49.38**	5.65	0.07	46.33	2.93
d-Simplex-FD [20]	1.00	48.76	44.96	0.38	49.67	17.31	0.45	48.42	21.32	0.34	44.82	15.37
d-Simplex-HOC (this paper)	1.00	49.98	48.48	**0.86**	48.21	**42.41**	**0.90**	47.56	**43.20**	**0.49**	**47.34**	**23.44**

M: Search accuracy 1:N search

ResNet18
TinyImageNet /ImageNet20 dataset

200 training classes; init 50 classes

Method	2 tasks			7 tasks			16 tasks			31 tasks		
	AC	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*
ER baseline	1.00	34.87	34.87	×	24.63	×	×	21.87	×	<0.01	19.15	0.05
FAN [19]	1.00	33.94	33.94	×	33.94	×	×	22.94	×	<0.01	20.11	0.10
BCT-ER	1.00	34.67	34.67	×	24.27	×	0.01	22.77	0.22	<0.01	19.67	0.04
LCE-ER	1.00	34.21	34.21	×	22.19	×	×	16.82	×	0.02	14.80	0.34
AdvBCT-ER	1.00	34.32	34.32	×	19.77	×	×	19.02	×	<0.01	16.61	0.04
CVS [7]	1.00	**37.14**	**37.14**	0.10	**33.20**	3.29	0.05	29.79	1.62	0.01	24.71	0.35
d-Simplex-FD [20]	×	32.20	×	0.14	30.52	4.43	0.05	29.17	1.55	0.02	27.09	0.44
d-Simplex-HOC (this paper)	1.00	36.85	36.85	**0.81**	31.45	**25.96**	**0.82**	**31.91**	**26.72**	**0.77**	**32.35**	**25.14**

AC Average Compatibility: Normalized count of Compatibility over T tasks

AA Average Accuracy: Average Accuracy over T tasks

ACA Average Compatibility Accuracy: Average Accuracy over tasks that satisfy the Compatibility Criterion

ResNet50
pre-trained Imagenet 1k
CUB 180/20 dataset

180 training classes; init 60 classes)

Method	2 tasks			7 tasks			16 tasks			31 tasks		
	AC	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*	*AC*	*AA*	*ACA*
ER baseline	1.00	**46.83**	46.26	0.48	43.77	20.80	0.25	42.85	10.55	0.07	38.54	3.08
FAN [19]	1.00	45.58	43.71	0.62	43.77	26.25	0.35	41.46	14.49	0.10	36.65	4.16
BCT-ER	1.00	44.90	43.71	0.62	**45.15**	27.06	0.26	41.37	11.03	0.08	38.26	4.00
LCE-ER	1.00	46.54	**47.62**	0.10	39.61	3.91	0.09	35.25	3.47	0.05	30.95	1.82
AdvBCT-ER	1.00	43.37	41.67	0.38	40.55	15.45	0.13	37.95	4.76	0.07	36.25	2.73
CVS [7]	1.00	45.58	46.09	0.62	44.75	27.28	0.23	**43.12**	9.87	0.14	39.67	5.96
d-Simplex-FD [20]	1.00	39.40	36.74	0.86	39.74	35.08	0.60	39.09	24.03	0.55	39.42	21.97
d-Simplex-HOC (this paper)	1.00	41.61	42.52	**0.90**	41.50	**37.72**	**0.73**	41.17	**30.01**	**0.60**	**42.01**	**25.46**

Biondi et al., IEEE TPAMI 2024 (sub)

Break 2

15’

A Paradigm Shift in AI

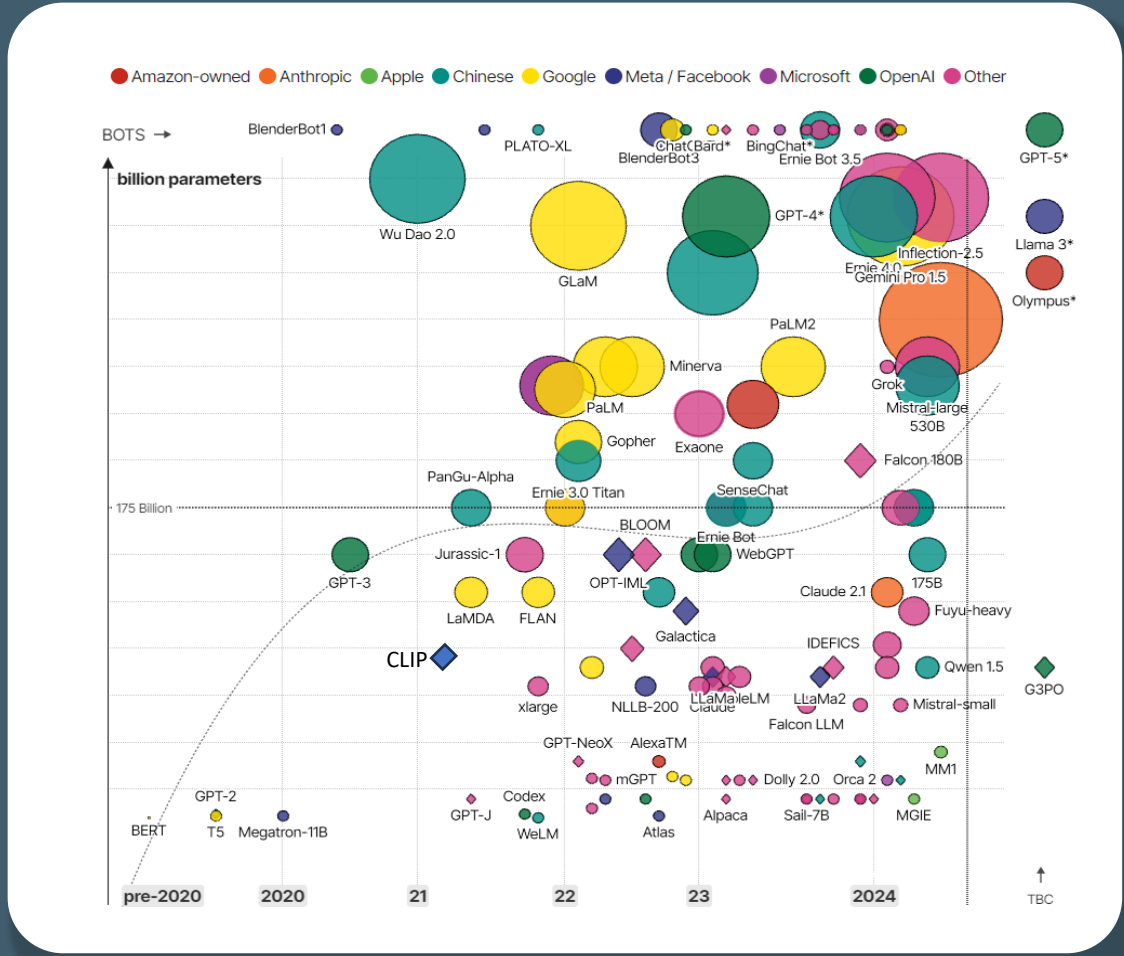
Foundation Models

Pre-trained Large Models

Large models pre-trained by third parties are more and more a fundamental building block for applications

The representation from the **Large Model's encoder** is exploited to develop specific applications

Used for both **Discriminative** and **Generative** tasks



picture authored by D. McCandless

Foundation Models updating

When updated Foundation Models may exhibit distinct issues that involve Compatibility

User–side

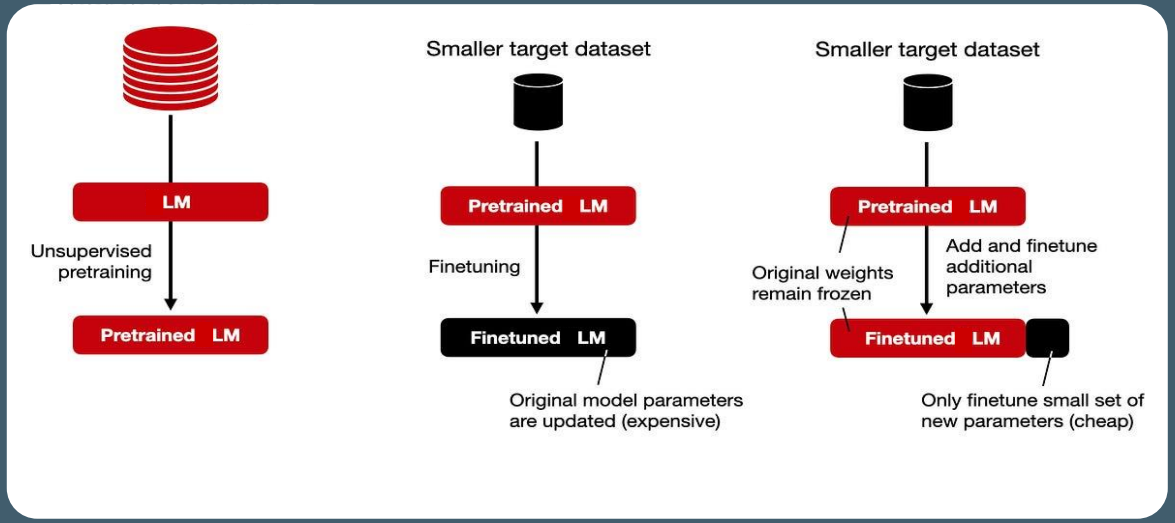
The now-standard practice is downloading and fine-tuning representations from the large pre-trained models. Compatibility with the user's fine-tuned version should ***allow seamless replacement***

Third parties-side

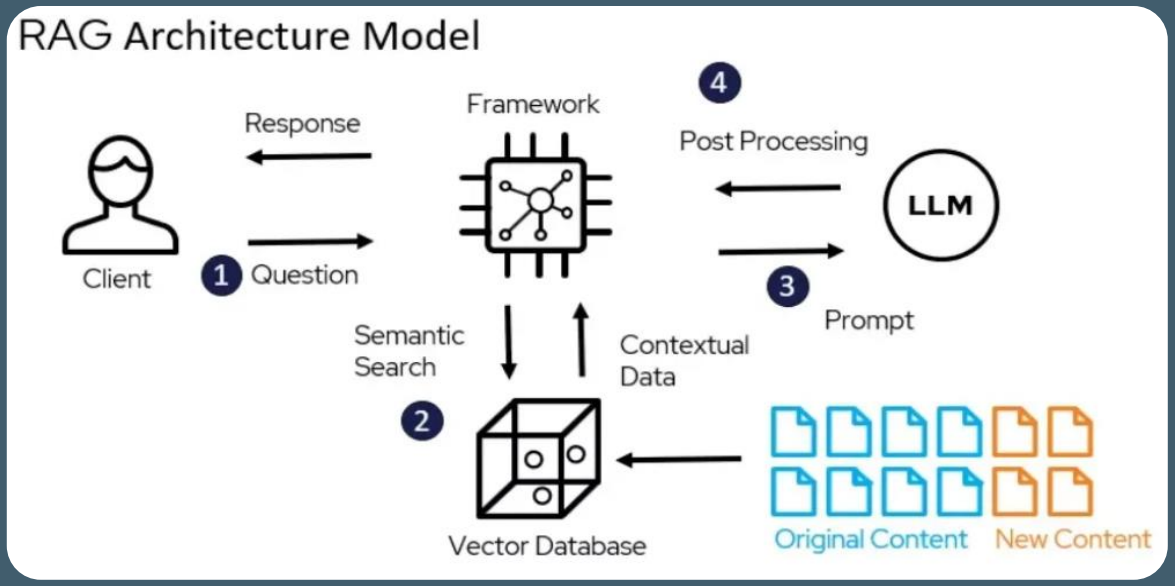
When the foundation model is updated Compatibility with the previous versions of the model should ***maintain consistency*** *and minimize disruption of the user experience*

User-side Adaptation

With a lightweight **adapter** or a **fine-tuned version** of the foundation model
(mostly for Discriminative tasks)



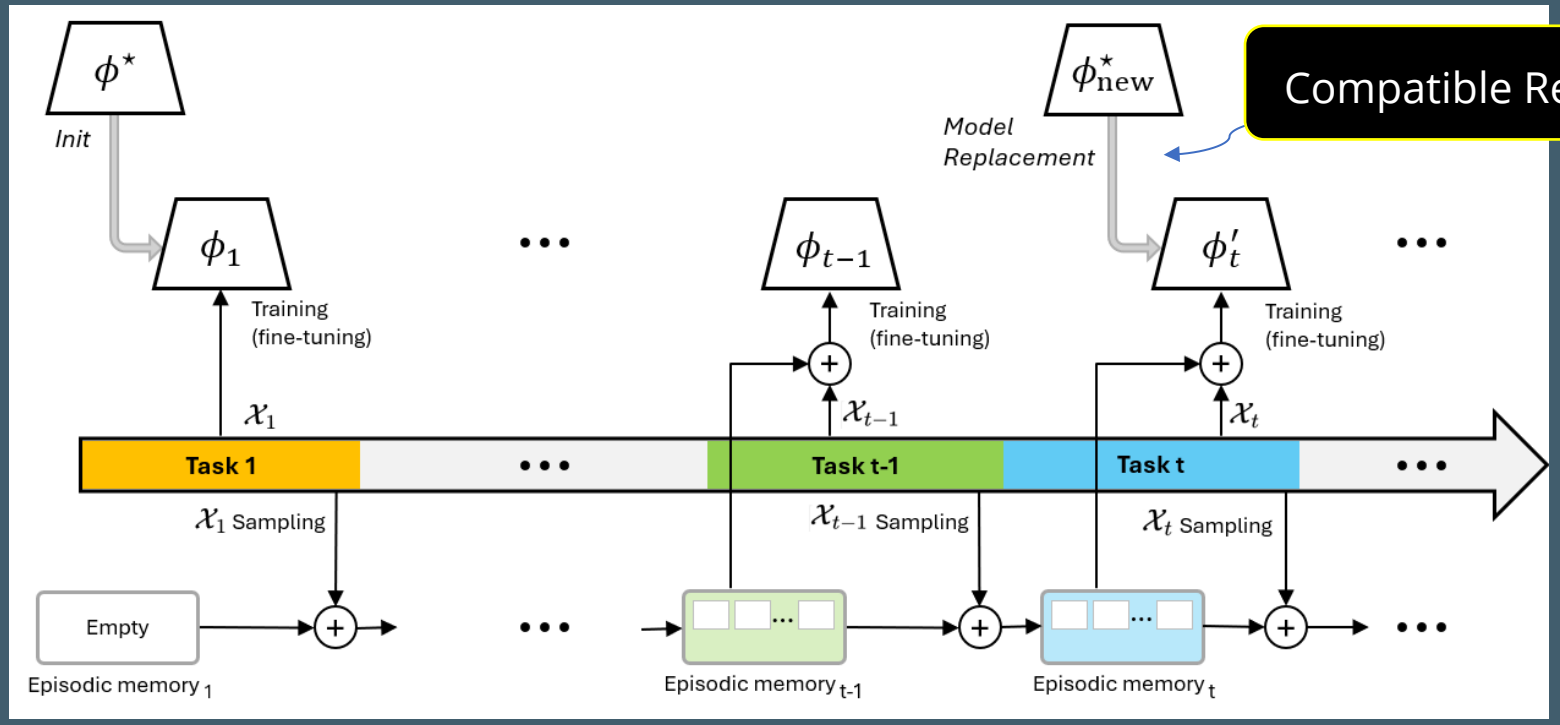
With **RAG (Retrieval Augmented Generation) -based applications** where the context of the foundation model is expanded with a set of user's private documents
(mostly for Generative tasks)



User-side Compatibility

The now-standard practice is downloading and fine-tuning representations from large pre-trained models

Compatibility at the user-side should allow **seamless replacement** with the new improved versions of the large model



Looking at the code

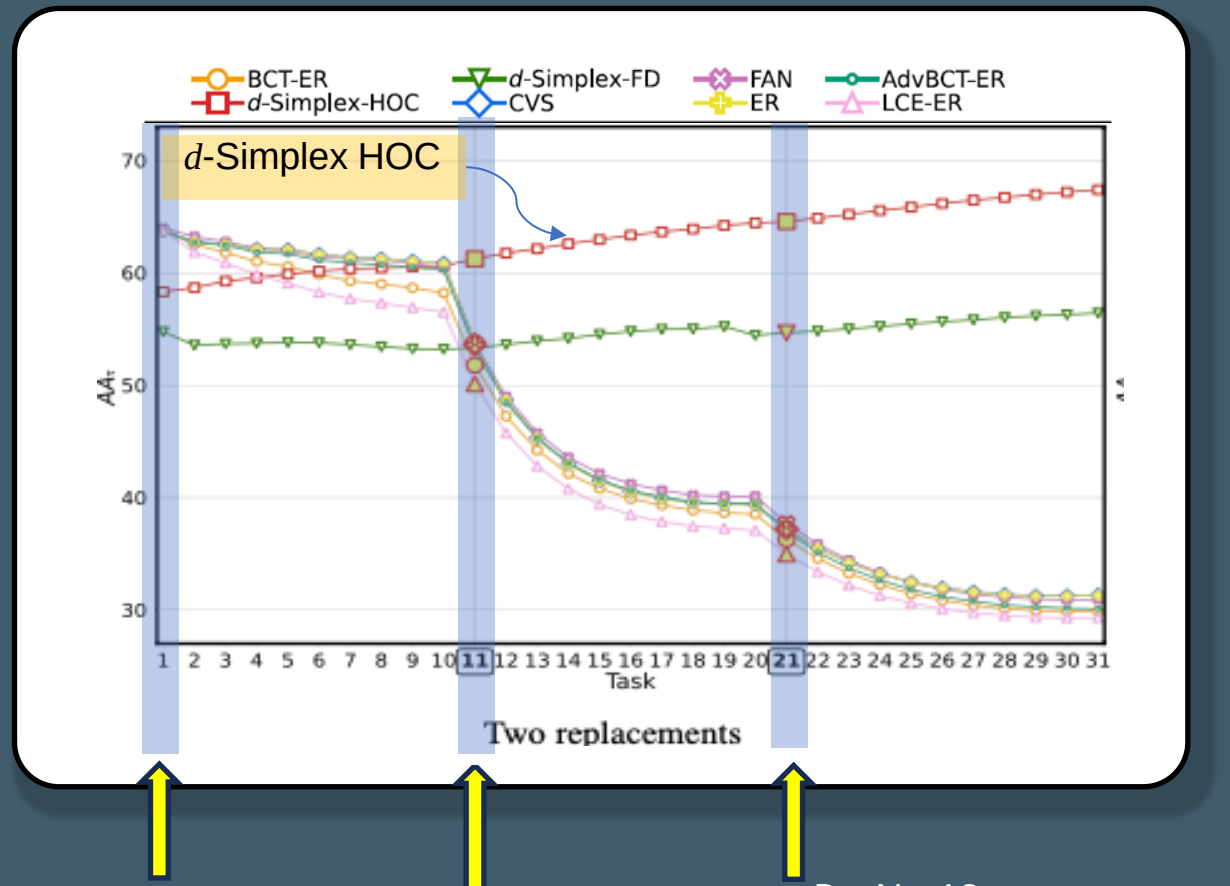
Large Model replacement

Google Colab link

Jupyter Notebook link

Updating with Replacement [*]

CIFAR10 dataset



d-Simplex fixed classifier with class pre-allocation provides **higher compatibility**

Fine-tuning
CIFAR100R

ResNet18
pre-trained
100 ImageNet32 classes

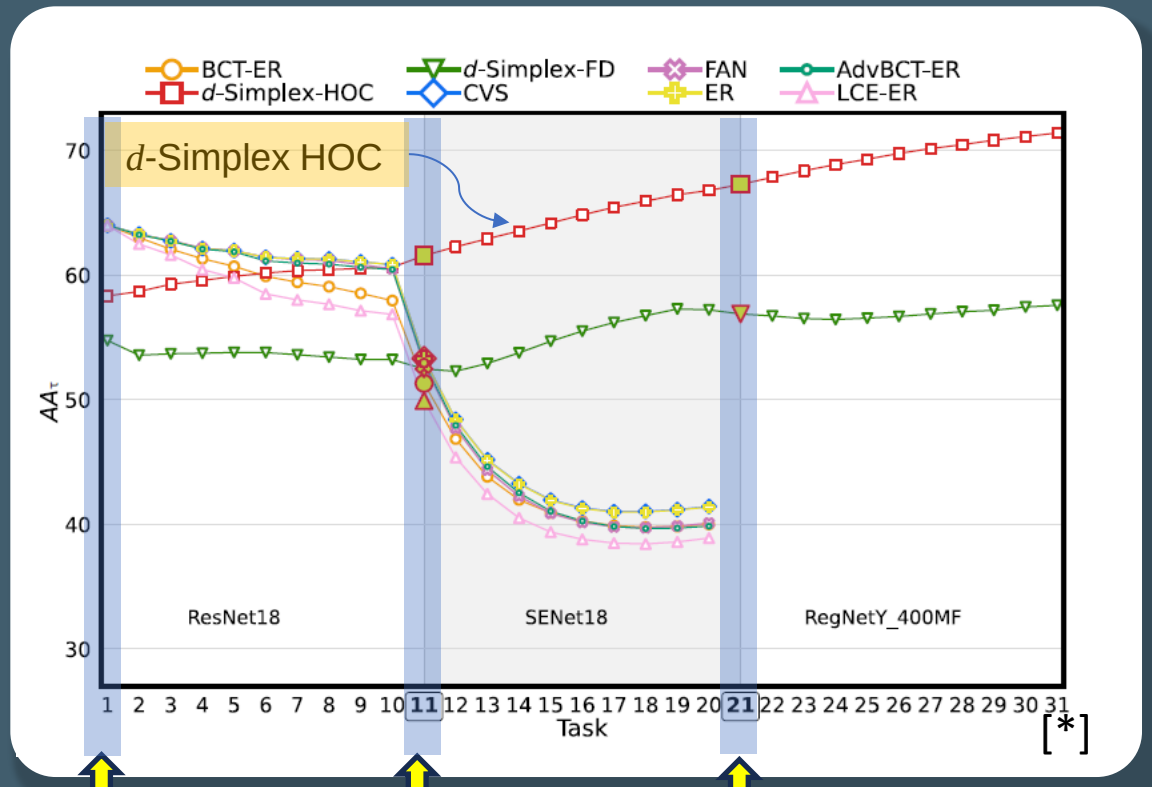
ResNet18
trained
300 ImageNet32 classes

ResNet18
trained
600 ImageNet32 classes

[*] N. Biondi et al., CVPR 2024

Updating with Replacement [*]

CIFAR10 dataset



Fine-tuning
CIFAR100R

ResNet18
pre-trained
100 ImageNet32 classes

Senet18
trained
300 ImageNet32 classes

RegNetY_400MF
trained
600 ImageNet32 classes

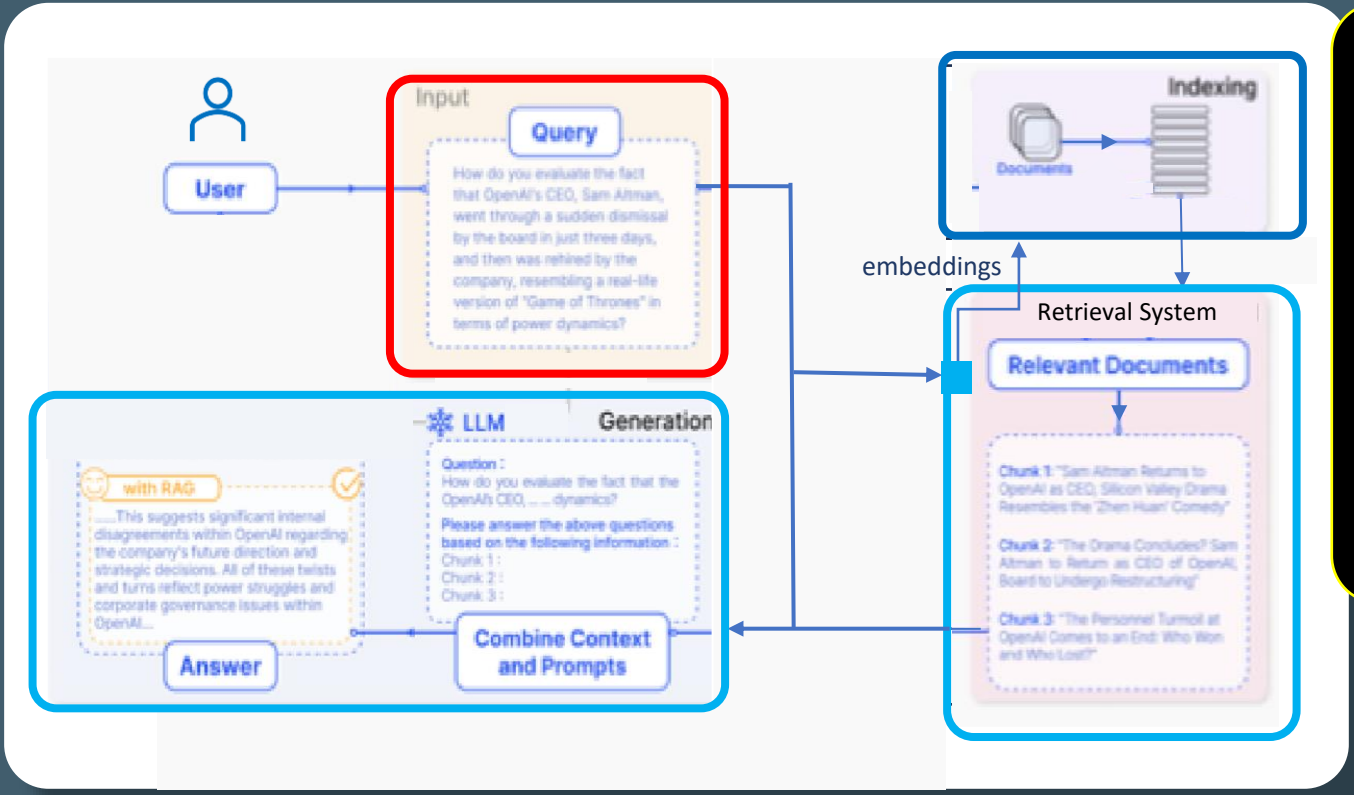
d-Simplex fixed classifier with class pre-allocation provides **higher compatibility** with **seamless model replacement**

The **fixed matrix** of the *d*-Simplex acts as a shared interface between the pre-trained model and the fine-tuned user model

[*] N. Biondi et al., CVPR 2024

Updating With RAG

RAG: a promising solution by **incorporating knowledge from external databases** to enhance the accuracy and credibility of the generation

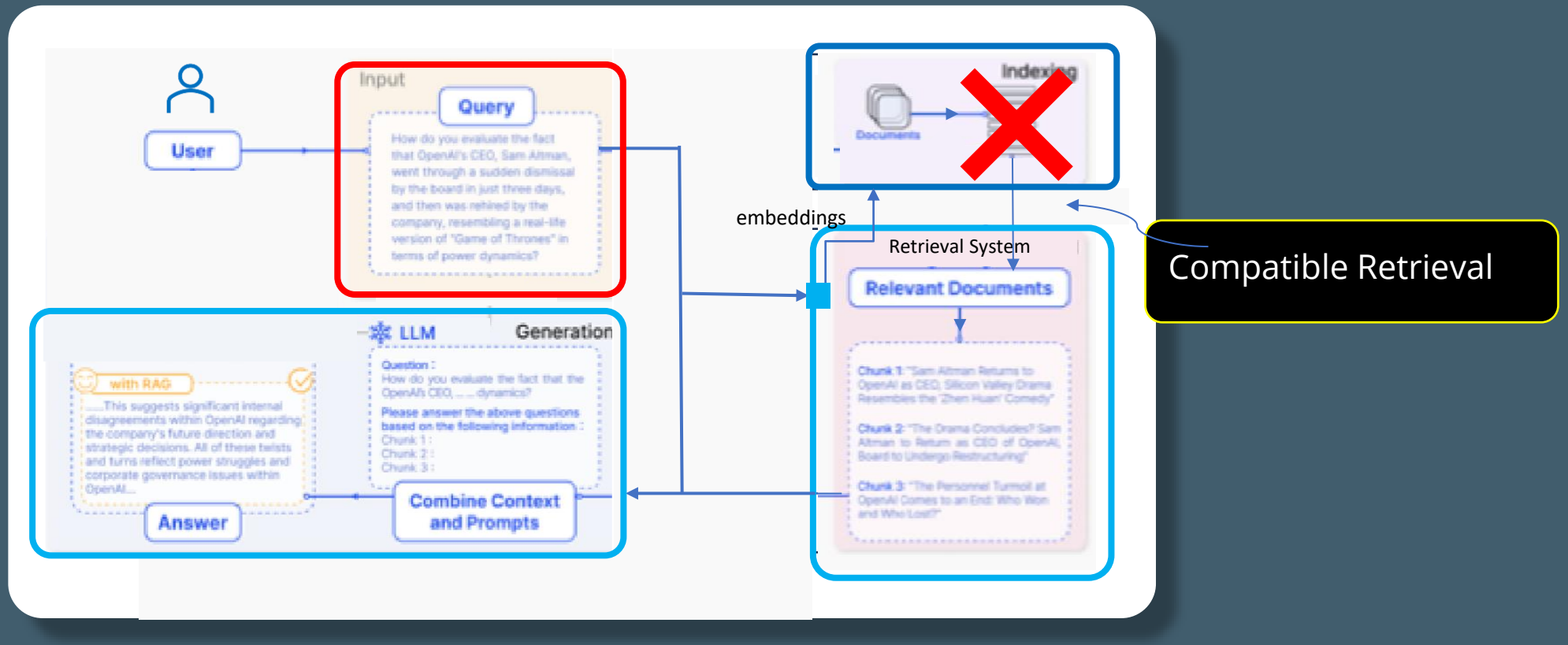


3 steps:

1) **Indexing** Documents are split into chunks, encoded into vectors, and stored in a database
2) **Retrieval** Retrieve the top k most relevant chunks based on semantic similarity.
3) **Generation** Input the original question and the retrieved chunks to generate the final answer

Compatibility with RAG

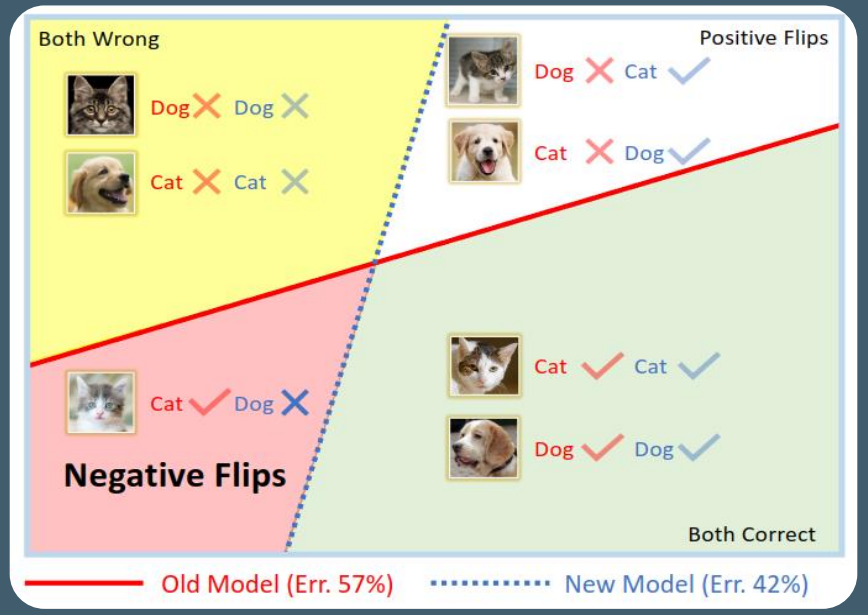
Compatibility **avoids re-indexing the vector database** when the large model is updated



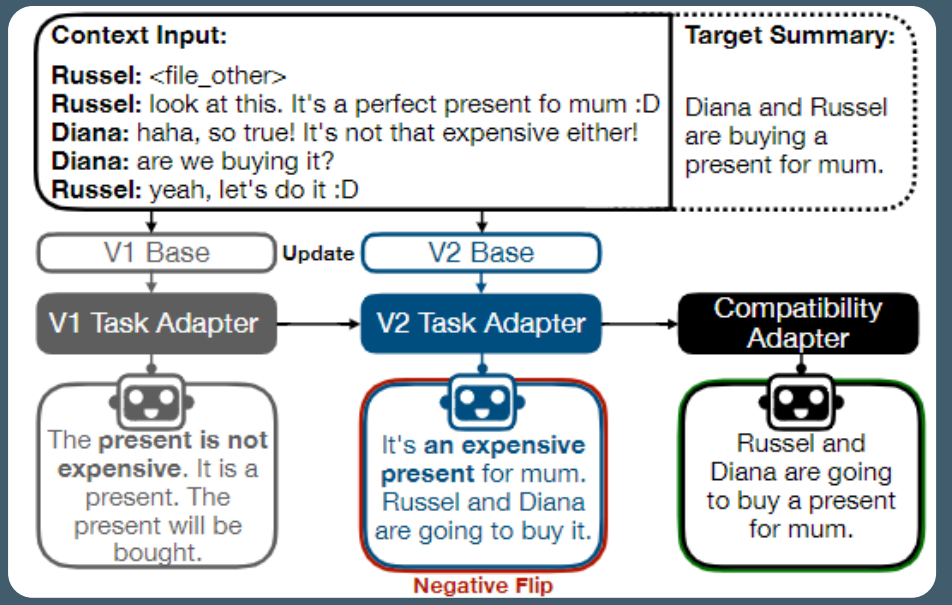
Third Party-side Compatibility

Upon updating the model should maintain consistency wrt the previous versions to minimize the disruption of the user experience

Compatibility should aim at minimizing the number of **Negative Flips** i.e. previously correct instances that are predicted incorrectly in the new version of the model



Discriminative task

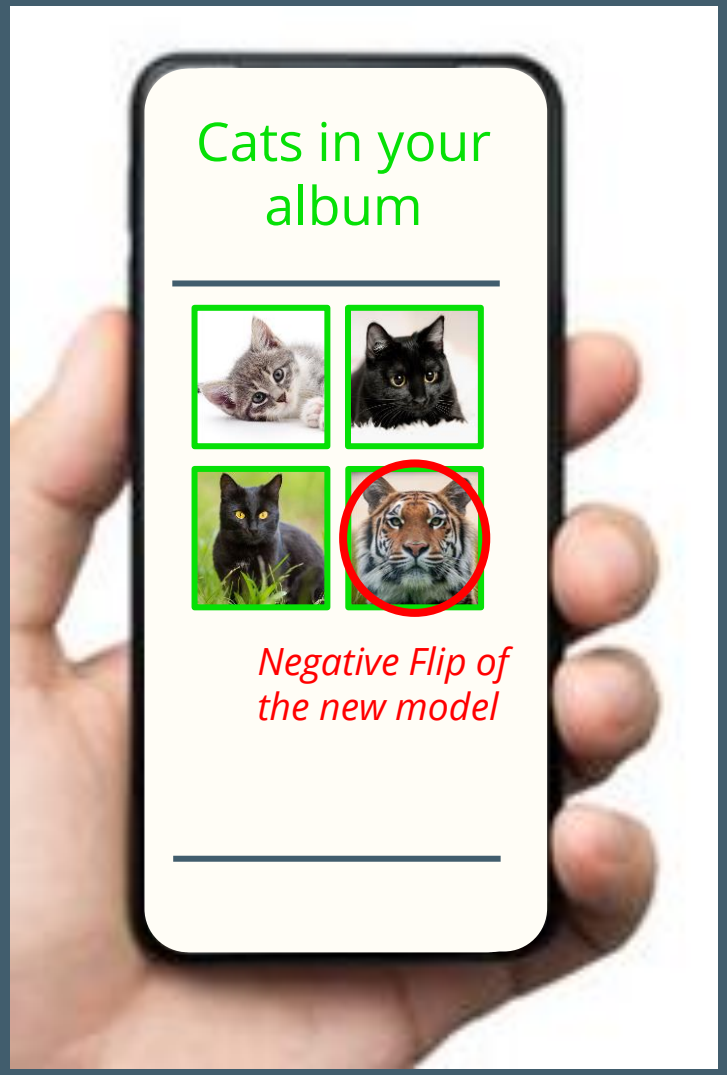


Generative task

Negative Flips in Discriminative Tasks

Goal of Compatibility training should be minimization of both the **Error Rate** and the **Negative Flip Rate**

$$\mathrm{NFR} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(\hat{y}_i^{\mathrm{new}} \neq y_i, \hat{y}_i^{\mathrm{old}} = y_i)$$



Discriminative task

Positive Congruent Training [*]

Negative Flipping Reduction

Positive-Congruent training aims to **minimize both the Error Rate and the Negative Flip Rate** as **separate and independent goals**

$$\min_{w} \underbrace{\mathcal{L}_{\text{CE}}(\phi^{\text{new}}, w)}_{\text{Error rate}} + \lambda \underbrace{\mathcal{L}_{\text{PC}}(\phi^{\text{new}}, w; \phi^{\text{old}})}_{\text{Negative flips}}$$

where

$$\lambda \mathcal{L}_{\text{PC}}(\phi^{\text{new}}, w; \phi^{\text{old}}) = \mathcal{F}(x_i)\mathcal{D}(\phi_w^{\text{new}}(x_i), q(x_i))$$

Approach	$\mathcal{F}$	$\mathcal{D}$	q
Naive Basline	$\mathbf{1}(\hat{y}^{\text{old}}(x_i) = y_i)$	Cross Entropy	$\vec{y}_i$
Focal Distillation - KL (FD-KL)	$\alpha + \beta \cdot \mathbf{1}(\hat{y}^{\text{old}}(x_i) = y_i)$	τ-scaled KL-Divergence	$p^{\text{old}}(y\|x_i)$
Focal Distillation - Logit Matching (FD-LM)	$\alpha + \beta \cdot \mathbf{1}(\hat{y}^{\text{old}}(x_i) = y_i)$	ℓ_2 distance	$\phi_w^{\text{old}}(x_i)$

The PC loss serves to bias training towards maximal positive congruence

- ✓ F is a filter function that applies a weight for each training sample based on the model outputs
- ✓ D is a distance function that measures the difference of the new model's outputs to a target vector $q(x_i)$ conditioned on x_i

[*] S. Yan et al., CVPR 2021

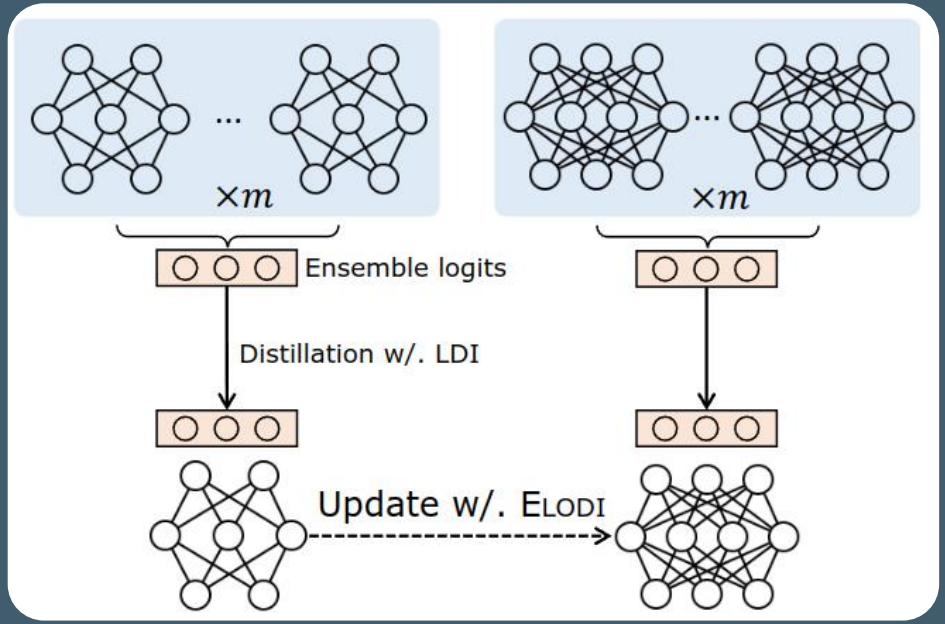
ELODI: Ensemble Logit Difference Inhibition for Positive-Congruent Training[*]

Negative Flipping Reduction

Deep **ensembles reduce NFR** by remedying potential flip samples that have large variations in the logits space of different single models

To reduce the cost of models ensemble the Logit Difference Inhibition (LDI) distillation loss is introduced

- ✓ LDI penalizes the logit difference between the ensemble and a single model on a subset of classes with the highest logit values
- ✓ The subset of classes with high logit values is more prone to prediction flipping



$$\mathcal{L}_{\mathrm{LDI}}(x) = \sum_{k \in \mathbb{K}(x)} \left(\| \phi_k(x) - \phi^{(\mathrm{ens})}(x) \|_p \right)^p$$

[*] Y. Zhao et al., IEEE TPAMI 2024

Negative Flips in Generative Tasks [*]

The NFR definition has been recently extended to the case of generative tasks (e.g. translation or summarization) with LLMs

PFR: positive flips rate

NFR: negative flips rate

$$\widetilde{\text{PFR}} \triangleq \frac{1}{N}\sum_{i}^{N} \mathbf{1}[D(x_i) > 0]$$

$$\widetilde{\text{NFR}} \triangleq \frac{1}{N}\sum_{i}^{N} \mathbf{1}[D(x_i) < 0]$$

where $D(x_i) \triangleq S(\mathcal{M}_{v2}(x_i), y_i) - S(\mathcal{M}_{v1}(x_i), y_i)$

is the distance between the two model outputs with respect to the grand truth according to a similarity metric *S*

[*] J. Echterhoff et al., 2024

MUSCLE: A Model Update Strategy for Compatible LLM Evolution [*]

Negative Flipping Reduction

Compatibility (in term of NFR reduction) is achieved by fine-tuning the local adapter of the new LLM with a distillation procedure:

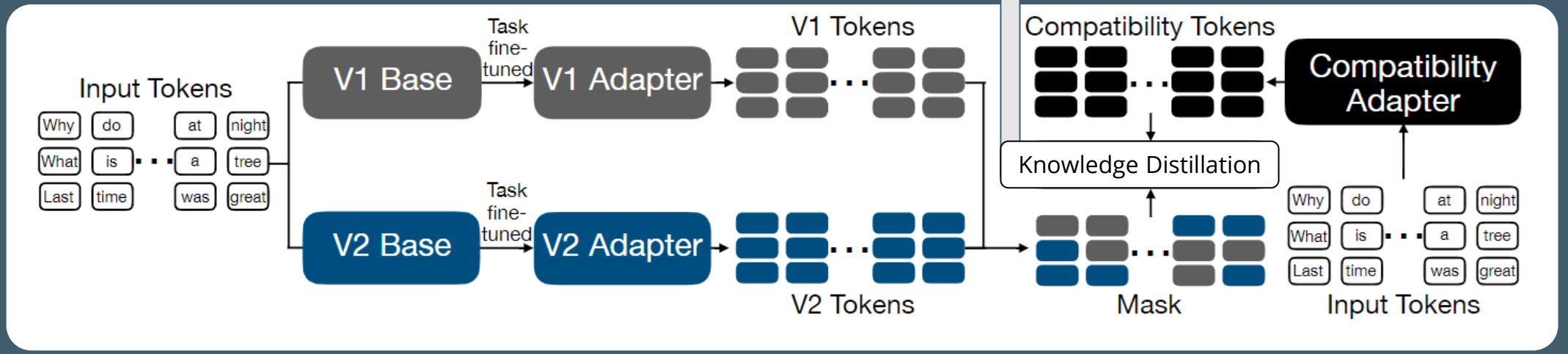
- ✓ if the compatible adapter predict the correct token it is aligned with the new model (improving performance
- ✓ otherwise it is aligned with the old adapter to obtain consistency (reducing the negative flip)

$$m_i = \mathbf{1}[\text{argmax}\ \sigma(z_{\mathcal{M}^C_{v2},i}) \neq y_i]$$

$$a_{\mathcal{M}_{v1}} = KL(\sigma(z_{\mathcal{M}_{v1},i}/T) \| \sigma(z_{\mathcal{M}^C_{v2},i}/T))$$

$$a_{\mathcal{M}_{v2}} = KL(\sigma(z_{\mathcal{M}_{v2},i}/T) \| \sigma(z_{\mathcal{M}^C_{v2},i}/T))$$

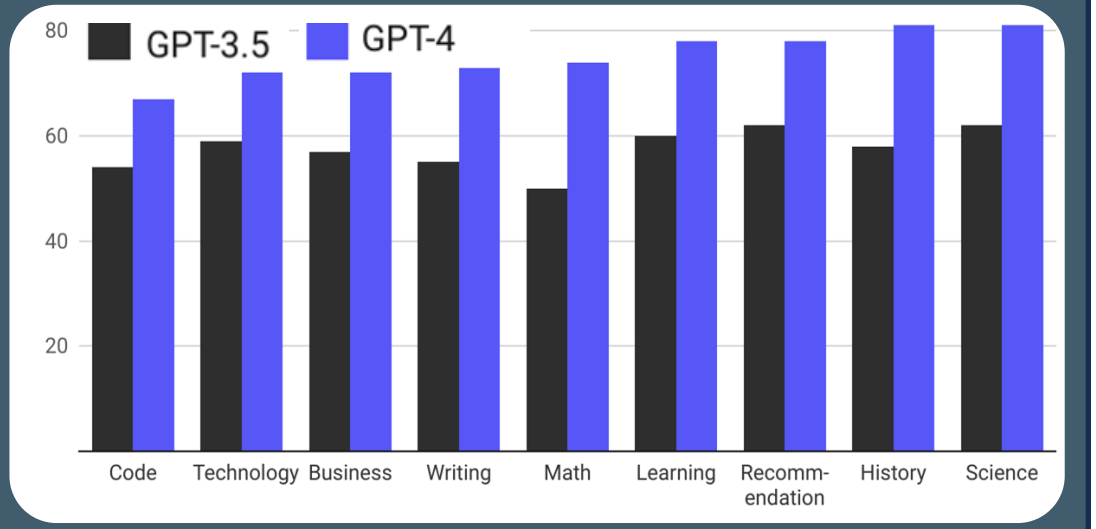
$$\mathcal{L}^m_{comp} = \frac{1}{n}\sum_{i=1}^{n} m_i \cdot a_{\mathcal{M}_{v1}} + (1 - m_i) \cdot a_{\mathcal{M}_{v2}}$$



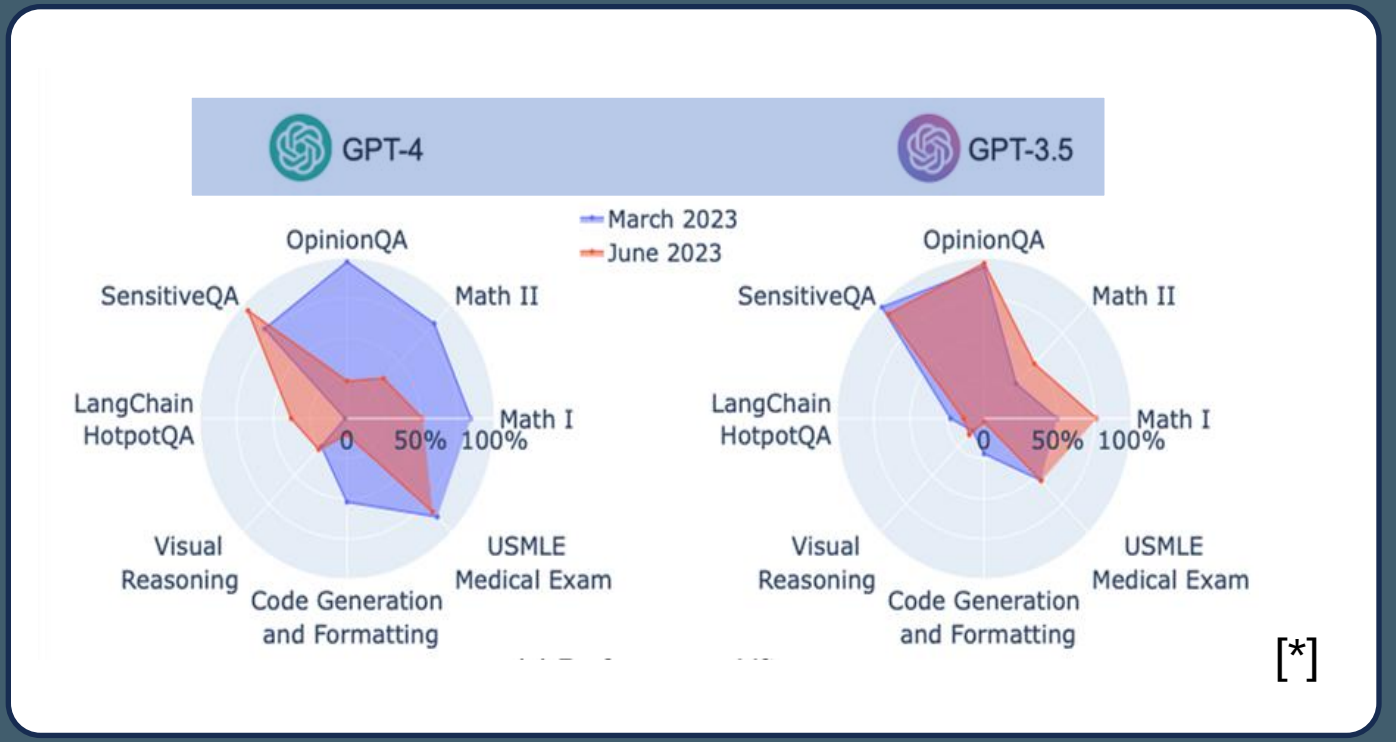
[*] J. Echterhoff et al., 2024

Relevance of NFR-based Compatibility

In GPT-3.5 and GPT-4 the performance and behavior between versions have gotten **substantially worse** on some tasks while have improved on others despite of the **overall performance increase**



Generative task



[*]

[*] L. Chen et al. HDSR 2024

Check it out!
Repo with slide and code!

github.com/NiccoBiondi/compatibility-tutorial

Niccolò Biondi
niccolo.biondi@unifi.it

Alberto Del Bimbo
alberto.delbimbo@unifi.it

References

N. Biondi et al., *Cl2r: Compatible lifelong learning representations*, ACM TOMM2023

N. Biondi et al., *Cores: Compatible representations via stationarity*, IEEE TPAMI2023

N. Biondi et al., *Stationary representations: Optimally approximating compatibility and implications for improved model replacements*, CVPR2024

N. Biondi et al., *Stationary (and therefore compatible) representation is all you need,* IEEE TPAMI 2024 (submitted)

R. Bommasani et al. , *On the opportunities and risks of foundation models*, ArXiv2022

K. Chen et al., *R3 adversarial network for cross model face recognition*, CVPR2019

M. de Lange et al.., *A continual learning survey: defining forgetting in classification tasks*, IEEE TPAMI 2021

J. Echterhof et al., *MUSCLE: A Model Update Strategy for Compatible LLM Evolution,* 2024

E. Hoffer et al., *Fix your classifier: the marginal value of training the last weight layer*, ICLR2018

A. Iscen et al., *Memory-efficient incremental learning through feature adaptation*, CVPR2020

Z. Li & D. Hoiem, *Learning without forgetting*, IEEE TPAMI2017

D. Mc Candless, blog: *https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt*

Q. Meng et al., *Learning compatible embeddings*, ICCV2021

References

R. Merrit, *What is a transformer model?*, NVIDIA blog

F. Pernici et al., *Regular polytope networks*, TNNLS2021

F. Pernici et al., *Class-incremental learning with pre-allocated fixed classifiers*, ICPR2020

F. Pernici et al., *Maximally Compact and Separated Features with Regular Polytope Networks*, CVPRW2019

A. Radford et al., *Learning transferable visual models from natural language supervision,* ICML2021

S. A Rebuffi et al., *iCARL incremental classifier and representation learning*, CVPR2017

Y. Shen et al. *Towards backward-compatible representation learning*, CVPR2020

G. van de Ven et al., *Three types of incremental learning*, Nature Machine Intelligence 2022

A. Vaswani et al., *Attention is all you need,* NeurIPS2017

CY. Wang et al., *Unified Representation Learning for Cross Model Compatibility*, BMVC2020

TST. Wan et al., *Continual learning for visual search with backward consistent feature embedding*, CVPR2022

B. Zhang et al., *Towards Universal Backward-Compatible Representation Learning*, IJCAI 2022

B. Zhao et al., *Continual representation learning for biometric identification*, WACV 2021

Y. Zhao et al., *ELODI: Ensemble Logit Difference Inhibition for Positive-Congruent Training*, IEEE TPAMI2024