Rethinking Transfer Learning for Medical Image Classification

Le Peng, Hengyue Liang, Gaoxiang Luo, Taihui Li, Ju Sun

***Abstract*— Transfer learning (TL) from pretrained deep models is a standard practice in modern medical image classification (MIC). However, what levels of features to be reused are problem-dependent, and uniformly finetuning all layers of pretrained models may be suboptimal. This insight has partly motivated the recent *differential* TL strategies, such as TransFusion (TF) and layer-wise finetuning (LWFT), which treat the layers in the pretrained models differentially. In this paper, we add one more strategy into this family, called *TruncatedTL*, which reuses and finetunes appropriate bottom layers and directly discards the remaining layers. This yields not only superior MIC performance but also compact models for efficient inference, compared to other differential TL methods. We validate the performance and model efficiency of TruncatedTL on three MIC tasks covering both 2D and 3D images. For example, on the BIMCV COVID-19 classification dataset, we obtain improved performance with around $1/4$ model size and $2/3$ inference time compared to the standard full TL model. Code is available at `https://github.com/sun-umn/Transfer-Learning-in-Medical-Imaging`.**

***Index Terms*— transfer learning, image classification, deep learning, convolutional neural networks**

I. Introduction

TRANSFER learning (TL) is a common practice for medical image classification (MIC), especially when training data are limited. In typical TL pipelines for MIC, deep convolutional neural networks (DCNNs) pretrained on large-scale *source tasks* (e.g., object recognition on ImageNet [1]) are finetuned as backbone models for *target MIC tasks*; see, e.g., [2]–[7], for examples of prior successes.

The key to TL is feature reuse from the source to the target tasks, which leads to practical benefits such as fast convergence in training, and good test performance even if the target data are scarce [10]. Pretrained DCNNs extract increasingly more abstract visual features from bottom to top layers: from low-level corners and textures, to mid-level blobs and parts, and finally to high-level shapes and patterns [8]. While shapes and patterns are crucial for recognizing and segmenting generic visual objects (see Fig. 1 (left)), they are not necessarily the defining features for diseases: diseases can often take the form of abnormal textures and blobs, which correspond to low- to mid-level features (see Fig. 1 (right)). *So intuitively for MIC, we may only need to finetune a reasonable number of the bottom layers commensurate with the levels of features needed, and ignore the top layers.* However, standard TL practice for MIC retains all layers, and uses them as fixed feature extractors or finetunes them uniformly.

[11], [12] depart from the uniform TL approach and propose TL methods that treat top and bottom layers differently. Prioritizing high-level features and the classifier, [11] proposes

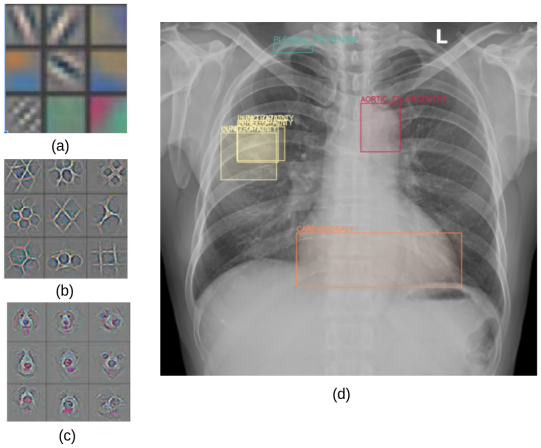


Fig. 1. (left) The feature hierarchy learned by typical DCNNs on computer vision (CV) datasets; (right) Examples of diseases in a chest x-ray. Only low-level (a) textures and/or mid-level (b) blobs are needed for detecting the diseases on the right (c). (the left visualized features adapted from [8]; the right chest x-ray adapted from [9]).

TABLE I
List of Common Acronyms (in alphabetic order)

AUROC	area under the receiver operating characteristic curve
AUPRC	area under the precision-recall curve
CV	computer vision
DCNN	deep convolutional neural networks
FTL	full transfer learning
LR	learning rate
LWFT	layer-wise finetuning
MACs	multiply–accumulate operation
MIC	medical image classification
MLP	multi-layer perceptron
SVCCA	singular value canonical correlation analysis
TL	transfer learning
TF	TransFusion
TTL	truncated transfer learning

Manuscript received xxx. This work was in part supported by Cisco Systems, Inc under the award SOW 1043496, and in part by National Science Foundation under the Grant CMMI 2038403. (Corresponding author: Ju Sun)

Le Peng, Gaoxiang Luo, Taihui Li, and Ju Sun are with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN USA (e-mails: {peng0347,luo00042, lixx5027, jusun}@umn.edu).

Hengyue Liang is with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN USA (e-mail: liang656@umn.edu).

layer-wise finetuning (LWFT) that finetunes an appropriate number of top layers and freezes the remaining bottom layers. In comparison, to improve training speed while preserving performance, [12] proposes *TransFusion* (TF) that finetunes bottom layers but retrains a coarsened version of top layers from scratch.

Neither of the above *differential* TL strategies clearly address the conundrum of why top layers are needed when only the features in bottom layers are to be reused. To bridge the gap, in this paper, we propose a novel, perhaps radical, TL strategy: remove top layers after an appropriate cutoff point, and finetune the truncated model left, dubbed *TruncatedTL* (TTL)—this is entirely consistent with our intuition about the feature hierarchy. Our main contributions include:

- **confirming the deficiency of full TL**. By experimenting with the full and differential TL strategies—including our TTL—on three MIC tasks and three popularly used DCNN models for MIC, we find that full TL (FTL) is almost always suboptimal in terms of classification performance, confirming the observation in [11].
- **proposing TruncatedTL (TTL) that leads to effective and compact models**. Our TTL outperforms other differential TL methods, while the resulting models are always smaller, sometimes substantially so. This leads to reduced computation and fast prediction during inference, and can be particularly valuable when dealing with 3D medical data such as CT and MRI images.
- **quantifying feature transferability in TL for MIC**. We use singular vector canonical correlation analysis (SVCCA) [13] to analyze feature transferability and confirm the importance of low- and mid-level features for MIC. The quantitative analysis also provides insights on how to choose high-quality truncation points to further optimize the model performance and efficiency of TTL.

II. Related Work

1) Deep TL: TL by reusing and finetuning visual features learned in DCNNs entered computer vision (CV) once DCNNs became the cornerstone of state-of-the-art (SOTA) object recognition models back to 2012. For example, [14], [15] propose using part of or full pretrained DCNNs as feature extractors for generic visual recognition. [10] studies the hierarchy of DCNN-based visual features, characterizes their transferability, and proposes finetuning pretrained features to boost performance on target tasks. Moreover, [8], [16] propose techniques to visualize visual features and their hierarchy. This popular line of TL techniques is among the broad family of TL methods for knowledge transfer from source to target tasks based on deep networks [17] and other learning models [18], and is the focus of this paper.

2) General TL stategies: While pretrained DCNNs can be either used as fixed feature extractors, or partially or fully finetuned on the target data, [10] argues that bottom layers learn generic features while top layers learn task-specific features, leading to folklore guidelines on how to choose appropriate TL strategies in various scenarios, as summarized in Fig. 2. Our intuition about the feature hierarchy is slightly

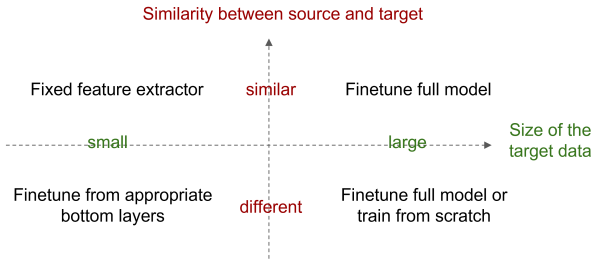


Fig. 2. Illustration of different DCNN-based TL scenarios and strategies

different: bottom layers learn low-level features that are spatially localized, and top layers learn high-level features that are spatially extensive (see Fig. 1). Although the two hierarchies may be aligned for most cases, they are distinguished by whether spatial scales are considered: task-specific features may be spatially localized, e.g., texture features to classify skin lesions [19], and general features may be spatially extensive, e.g., generic brain silhouettes in brain MRI images. Moreover, the spatial-scale hierarchy is built into DCNNs by design [20]. So we argue that the intuition about the low-high spatial feature hierarchy is more pertinent. We note that none of the popular strategies as summarized in Fig. 2 modify the pretrained DCNN models (except for the final multi-layer perceptron, MLP, classifiers)—in contrast to TF and our TTL.

3) Differential TL: Early work on TL for MIC [2]–[6], [21], [22] and segmentation [7], [23]–[25] parallels the relevant developments in CV, and mostly uses DCNNs pretrained on CV tasks as feature extractors or initializers (i.e., for finetuning). In fact, these two strategies remain dominant according to the very recent survey [26] on TL for MIC, which reviews around 120 relevant papers. But the former seems inappropriate as medical data are disparate from natural images dealt with in CV. From the bottom panel of Fig. 2, finetuning at least part of the DCNNs is probably more competitive even if the target data are limited. In this line, LWFT [11] finetunes top layers and freeze bottom layers, and incrementally allows finetuning more layers during model selection—this is recommended in [26] as a practical TL strategy for MIC that strikes a balance between training efficiency and performance. Similarly, TF [12] coarsens top layers which are then trained from scratch, and finetunes bottom layers from pretrained weights. Both LWFT and TF take inspiration from the general-specific feature hierarchy. In contrast, motivated by the low-high spatial feature hierarchy, our novel TTL method removes the top layers entirely and directly finetunes the truncated models. Our experiments in Section IV confirm that TTL surpasses LWFT and TF with improved performance and reduced inference cost.

4) Compact models for MIC: Both TF and our TTL lead to reduced models that can boost the inference efficiency, the first of its kind in TL for MIC, although the TF paper [12] does not stress this point. Compact models have been designed for specific MIC tasks, e.g., [12], [27], but our evaluation on the task of [27] in Section IV-D suggests that the differential TL strategies based on generic pretrained models, particularly our TTL, can outperform TL based on handcrafted models.

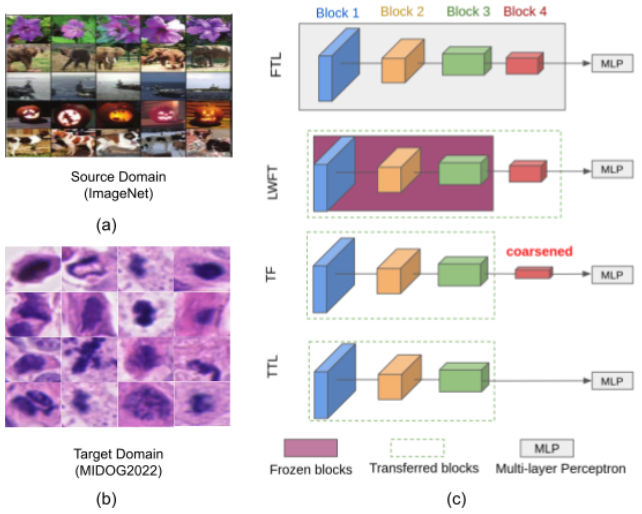


Fig. 3. Overview of typical TL setup, and the four TL methods that we focus on in this paper. (a) TL source domain: e.g., ImageNet object recognition; (b) TL target domain: e.g., mitotic cells classification; (c) Four TL methods: FTL, LWFT, TF, our TTL applied to ResNet50 pretrained on ImageNet.

Moreover, the growing set of methods for model quantization and compression [28]–[31] are equally applicable to both the original models and the reduced models.

III. Efficient Transfer Learning for MIC

Let $\mathcal{X}\times\mathcal{Y}$ denote any input-output (or feature-label) product space, and $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$ a distribution on $\mathcal{X}\times\mathcal{Y}$. TL considers a source task $\mathcal{D}_{\mathcal{X}_s\times\mathcal{Y}_s}\mapsto p_s$, where p_s is a desired predictor, and a target task $\mathcal{D}_{\mathcal{X}_t\times\mathcal{Y}_t}\mapsto p_t$. In typical TL, $\mathcal{X}_t\times\mathcal{Y}_t$ may be different from $\mathcal{X}_s\times\mathcal{Y}_s$, or at least $\mathcal{D}_{\mathcal{X}_t\times\mathcal{Y}_t}\neq\mathcal{D}_{\mathcal{X}_s\times\mathcal{Y}_s}$ even if $\mathcal{X}_t\times\mathcal{Y}_t=\mathcal{X}_s\times\mathcal{Y}_s$. The goal of TL is to transfer the knowledge from the source task $\mathcal{D}_{\mathcal{X}_s\times\mathcal{Y}_s}\mapsto p_s$ that is solved beforehand to the target task $\mathcal{D}_{\mathcal{X}_t\times\mathcal{Y}_t}\mapsto p_t$ [17], [32].

In this paper, we restrict TL to reusing and finetuning pretrained DCNNs for MIC. In this context, the source predictor $p_s=h_s\circ f_L\circ\cdots\circ f_1$ is pretrained on a large-scale source dataset $\{(x_i,y_i)\}\sim_{iid}\mathcal{D}_{\mathcal{X}_s\times\mathcal{Y}_s}$. Here, the f_i's are L convolulational layers, and h_s is the final MLP classifier. To perform TL, h_s is replaced by a new MLP predictor h_t with prediction heads matching the target task (i.e., with the desired number of outputs) to form the new model $p_t=h_t\circ f_L\circ\cdots\circ f_1$. Two dominant approaches of TL for MIC are: 1) *fixed feature extraction*: freeze the pretrained weights of $f_L\circ\cdots\circ f_1$, and optimize h_t from random initialization so that p_t fits the target data; 2) *full transfer learning (FTL)*: optimize all of $h_t\circ f_L\circ\cdots\circ f_1$, with h_t from random initialization whereas $f_L\circ\cdots\circ f_1$ from their pretrained weights so that p_t fits the target data.

A. Prior differential TL approaches

The two differential TL methods for MIC, i.e., LWFT [11] and TF [33], differ from the dominant TL approaches in that they treat the top and bottom layers differently, as illustrated in Fig. 3(c).

1) Layer-wise finetuning: LWFT does not distinguish MLP layers and convolutional layers. So slightly abusing our notation, assume that the pretrained DCNN is $f_N\circ f_{N-1}\circ\cdots\circ f_1$, where N is the total number of layers including both the MLP and convolutional layers. LWFT finetunes the top k layers $f_N\circ f_{N-1}\circ\cdots\circ f_{N-k+1}$, and freezes the bottom $N-k$ layers $f_{N-k}\circ\cdots\circ f_1$. The top layer f_N is finetuned with a base learning rate (LR) η, and the other $k-1$ layers with a LR $\eta/10$. To find an appropriate k, [11] proposes an incremental model selection procedure: start with $k=1$ layer, and include one more layer into finetuning if the previous set of layers does not achieve the desired level of performance[1]. Although LWFT was originally proposed for AlexNet [34], it can be easily generalized to work with advanced DCNN models such as ResNets and DenseNets that have block structures.

2) TransFusion: TF reuses bottom layers while slimming down top layers. Formally, for a cutoff index k, the pretrained model $p_s\circ f_L\circ\cdots\circ f_{k+1}\circ f_k\cdots\circ f_1$ is replaced by $p_t\circ f_L^{HV}\circ\cdots\circ f_{k+1}^{HV}\circ f_k\cdots\circ f_1$, where HV means halving the number of channels in the designated layer. TF then trains the coarsened model on the target data with the first half $f_k\cdots\circ f_1$ initialized by the pretrained weights (i.e., finetuning) and $p_t\circ f_L^{HV}\circ\cdots\circ f_{k+1}^{HV}$ initialized by random weights (i.e., training from scratch). The cutoff point k is the key hyperparameter in TF. TF was originally proposed to boost the finetuning speed, but we find that it often also boosts the classification performance compared to FTL; see Section IV.

B. Our truncated TL approach

Our method is radically simple: for an appropriate cutoff index k, we take the k bottom layers $f_k\circ\cdots\circ f_1$ from the pretrained DCNN model and then form and finetune the new predictor $h_t\circ f_k\circ\cdots\circ f_1$. Our TruncatedTL (TTL) method is illustrated in Fig. 3(c, last row).

The only crucial hyperparameter for TTL is the cutoff index k, which depends on the DCNN model and problem under consideration. For SOTA ResNet and DenseNet models that are popularly used in TL for MIC, there are always 4 convolutional blocks each consisting of repeated basic convolutional structures; see, e.g., the bottom of Fig. 4 for an illustration of ResNet50. So we propose a hierarchical search strategy: **(1) Stage 1: coarse block search**. Take the block cutoffs as candidate cutoff points, and report the best-performing one; **(2) Stage 2: fine-grained layer search**. Search over the neighboring layers of the cutoff from **Stage 1** to optimize the performance. Since k is also a crucial algorithm hyperparameter for TF and LWFT, we also adopt the same hierarchical search strategy for them when comparing the performance.

To quickly confirm the efficiency of TTL, we apply TTL and competing TL methods on an X-ray-based COVID-19 classification task on the BIMCV dataset [35]; more details about the setup can be found in Section IV-B. We pick the COVID example, as the salient radiological patterns in COVID X-rays such as multifocal and bilateral ground glass opacities and consolidations are low- to mid-level visual features [36]

[1]Finetuning starts with the original pretrained weights each time.

TABLE II
COVID-19 CLASSIFICATION WITH DIFFERENT TL STRATEGIES. THE BEST RESULT OF EACH COLUMN IS COLORED IN **RED**. ↑ INDICATES LARGER VALUE IS BETTER AND ↓ INDICATES LOWER VALUE IS BETTER. "-1" MEANS WITH THE BLOCK-WISE SEARCH ONLY, AND "-2" MEANS WITH THE TWO-STAGE BLOCK-LAYER HIERARCHICAL SEARCH.

Method	AUROC↑	AUPRC↑	Params(M)↓	MACs(G)↓	CPU(ms)↓	GPU(ms)↓
FTL	0.849 ± 0.001	0.857 ± 0.003	23.5	4.12	79.6	3.59
(l)1-7 TF-1	0.856 ± 0.011	0.863 ± 0.012	12.9	3.56	67.0	3.55
LWFT-1	0.848 ± 0.002	0.861 ± 0.004	23.5	4.12	76.9	3.59
TTL-1	0.851 ± 0.002	0.860 ± 0.002	8.55	3.31	59.7	3.19
TF-2	0.856 ± 0.011	0.863 ± 0.012	12.9	3.56	72.7	3.56
LWFT-2	0.853 ± 0.005	0.861 ± 0.001	23.5	4.12	79.7	3.56
TTL-2 (ours)	$\mathbf{0.861 \pm 0.013}$	$\mathbf{0.871 \pm 0.008}$	**6.31**	**2.87**	**53.1**	**2.97**

and hence we can easily see the benefit of differential TL methods including our TTL. We measure the classification performance by both AUROC (area under the receiver-operating-characteristic curve) and AUPRC (area under the precision-recall curve), and measure the inference complexity by Params (number of parameters in the model, M—millions), MACs (multiply-add operation counts [37][2], G—billion), CPU/GPU (wallclock run time on CPU and GPU by milliseconds[3]; details of our computing environment can be found in Section IV-A.).

Table II summarizes the results, and we observe that: (1) differential TL methods (TF, LWFT, and TTL) perform better or at least on par with FTL, and the layer-wise search in the second stage further boosts their performance. Also, TTL is the best-performing TL method with the two-stage hierarchical search; (2) TF and TTL that slim down the model lead to reduced model complexity and hence considerably less run time. TTL is a clear winner in terms of both performance and inference complexity.

C. Transferablity analysis

Besides the positive confirmation above, in this section, we provide quantitative corroboration for our claim that top layers might not be needed in TL for MIC. To this end, we need a variant of the classical canonical correlation analysis (CCA), singular-vector CCA (SVCCA) [13].

1) SVCCA for quantifying feature correlations: CCA is a classical statistical tool for measuring the linear correlation between random vectors. Suppose that $\boldsymbol{x} \in \mathbb{R}^p$ and $\boldsymbol{y} \in \mathbb{R}^q$ are two random vectors containing p and q features, respectively. CCA seeks the linear combinations $\boldsymbol{u}^\intercal \boldsymbol{x}$ and $\boldsymbol{v}^\intercal \boldsymbol{y}$ of the two sets of features with the largest covariance $\mathrm{cov}(\boldsymbol{u}^\intercal \boldsymbol{x}, \boldsymbol{v}^\intercal \boldsymbol{y})$. Assume $\mathbb{E}[\boldsymbol{x}] = \boldsymbol{0}$ and $\mathbb{E}[\boldsymbol{y}] = \boldsymbol{0}$. The problem can be formulated as

$$\max_{\boldsymbol{u},\boldsymbol{v}} \boldsymbol{u}^\intercal \boldsymbol{\Sigma}_{\boldsymbol{xy}} \boldsymbol{v} \quad \text{s.t. } \boldsymbol{u}^\intercal \boldsymbol{\Sigma}_{\boldsymbol{xx}} \boldsymbol{u} = 1,\ \boldsymbol{v}^\intercal \boldsymbol{\Sigma}_{\boldsymbol{yy}} \boldsymbol{v} = 1,$$

where $\boldsymbol{\Sigma}_{\boldsymbol{xy}} \doteq \mathbb{E}[\boldsymbol{x}\boldsymbol{y}^\intercal]$, $\boldsymbol{\Sigma}_{\boldsymbol{xx}} \doteq \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\intercal]$, and $\boldsymbol{\Sigma}_{\boldsymbol{yy}} \doteq \mathbb{E}[\boldsymbol{y}\boldsymbol{y}^\intercal]$,[4] and the constraints fix the scales of $\boldsymbol{u}$ and $\boldsymbol{v}$ so that the objective does not blow up. Once the $(\boldsymbol{u}, \boldsymbol{v})$ pair with the largest covariance is computed, subsequent pairs are computed iteratively in a similar fashion with the additional constraint that new linear combinations are statistically decorrelated with the ones already computed. The overall iterative process can be written compactly as

$$\max_{\boldsymbol{U} \in \mathbb{R}^{p \times k}, \boldsymbol{V} \in \mathbb{R}^{q \times k}} \mathrm{tr}\,(\boldsymbol{U}^\intercal \boldsymbol{\Sigma}_{\boldsymbol{xy}} \boldsymbol{V}) \quad \text{s.t. } \boldsymbol{U}^\intercal \boldsymbol{\Sigma}_{\boldsymbol{xx}} \boldsymbol{U} = \boldsymbol{I}_{k \times k},\ \boldsymbol{V}^\intercal \boldsymbol{\Sigma}_{\boldsymbol{yy}} \boldsymbol{V} = \boldsymbol{I}_{k \times k},$$

which computes the first k pairs of most correlated linear combinations. In practice, all the covariance matrices $\boldsymbol{\Sigma}_{\boldsymbol{xy}}$, $\boldsymbol{\Sigma}_{\boldsymbol{xx}}$, and $\boldsymbol{\Sigma}_{\boldsymbol{yy}}$ are replaced by their finite-sample approximations, and the covariances of the top k most correlated pairs are the top k singular values of $\boldsymbol{\Sigma}_{\boldsymbol{xx}}^{-1/2} \boldsymbol{\Sigma}_{\boldsymbol{xy}} \boldsymbol{\Sigma}_{\boldsymbol{yy}}^{-1/2}$ [38], all of which lie in $[0, 1]$. We call these singular values the *CCA coefficients*. Obviously, high values in these coefficients indicate high levels of correlation.

For our subsequent analyses, we typically need to find the correlation between the features of two data matrices $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ and $\boldsymbol{Y} \in \mathbb{R}^{n \times q}$, where n is the number of data points. SVCCA performs principal component analysis (PCA) separately on $\boldsymbol{X}$ and $\boldsymbol{Y}$ first, so that potential noise in the data is suppressed and the ensuing CCA analysis becomes more robust. We typically plot the CCA coefficients in descending order when analyzing two group of features.

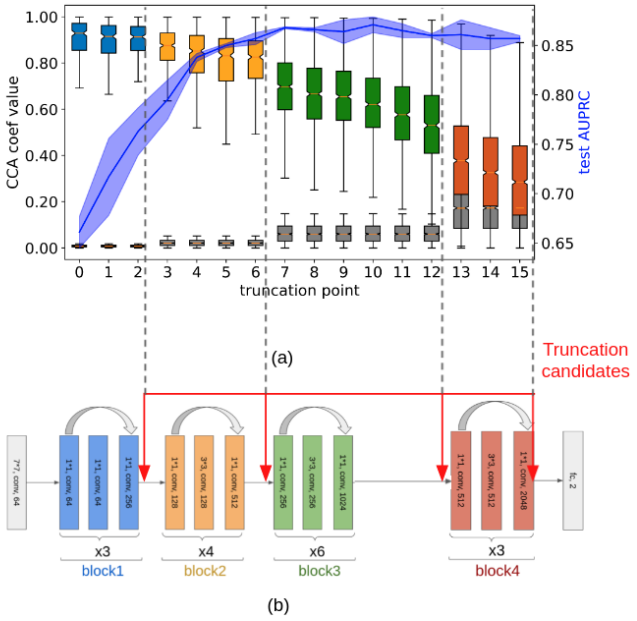


Fig. 4. Illustration of feature transferability and the performance of different levels of features on BIMCV. We take the a ResNet50 model pretrained on ImageNet, and perform a full TL on BIMCV. We consider **17** natural truncation/cutoff points that do not cut through the skip connections. (a) The top group of the bar plot shows the feature correlation before and after TL at different layers: each bar delineates the distribution of the CCA coefficients for that corresponding layer. The bottom group shows the level of correlation between random features of the same sizes for reference. The shaded blue curve represents the test AUPRC of TTL when performed at different truncation points—the shaded surrounding region indicates the level of standard deviation due to **3** independent runs; (b) The ResNet50 architecture and all possible truncation points.

[2] https://github.com/sovrasov/flops-counter.pytorch

[3] The speed on CPU is measured with the CPU-only version of PyTorch—the GPU version performs suboptimally when running on CPUs only; similarly for all the subsequent speed comparisons.

[4] $\doteq$ means "defined as".

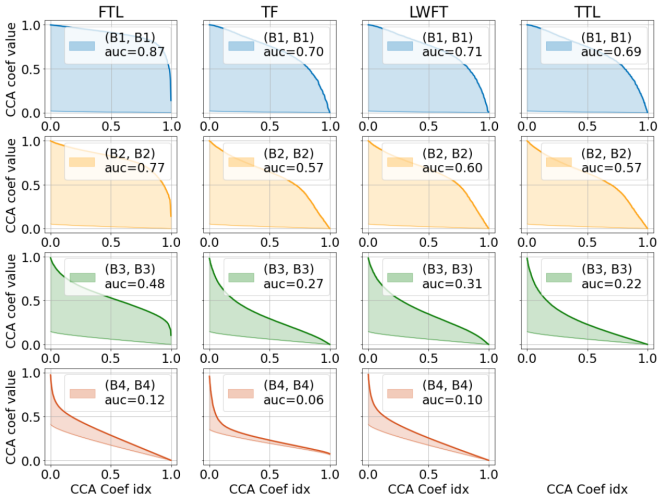


Fig. 5. The correlation of features at the same block before (i.e., the pretrained model) and after the various TL methods. In each plot, the bold curve indicates the CCA coefficients for the two blocks of features, and the light curve indicates the CCA coefficients for two sets of random features for reference. So the area (AUC) between the two curves is a quantitative measure of correlation between the said blocks. To ensure we can compare these AUC values vertically (i.e., across different blocks that may have different numbers of features), we normalize all the indices of the CCA coefficients to be $[0, 1]$. Our TTL does not include block4 as we remove it in the truncation.

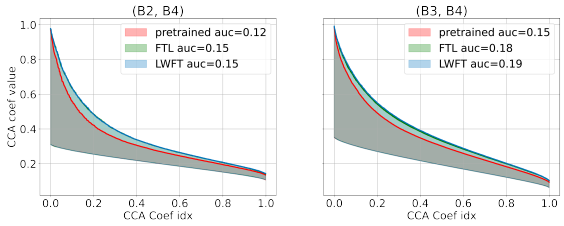


Fig. 6. The correlation between the block2 and block4 features (B2, B4) and of block3 and block4 features (B3, B4) in the pretrained model and the resulting models with different TL methods. In each plot, the bold curves indicate the CCA coefficients for the two blocks of features, and the light curve indicates the CCA coefficients for two sets of random features for reference. So the area (AUC) between the solid and light curves is a quantitative measure of correlation between the said blocks. TF is not included as it modifies the architecture of block4 that distorts the CCA coefficients, and our TTL is not included as we remove block4 in the truncation.

2) Layer transferability analysis: With SVCCA, we are now ready to present quantitative results to show that reusing and finetuning top layers may be unnecessary. We again take the BIMCV dataset for illustration.

a) Features in top layers change substantially, but the changes do not help improve the performance: In Fig. 4 we present the per-layer correlation of the intermediate features before and after FTL. First, the correlation monotonically decreases from bottom to top layers, suggesting increasingly dramatic feature finetuning/learning. For features residing in block1 through block3, the correlation levels are substantially higher than that of random features, suggesting considerable feature reuse together with the finetuning. However, for features in block4 of ResNet50 which contains the top layers, the correlation level approaches that of random features. So these high-level features are drastically changed during FTL and there is little reuse. The changes do not help and in fact hurt the performance: when we take intermediate features after the FTL and train a classifier based on each level of them, we find that the performance peaks at block3, and starts to degrade afterward. From Fig. 5 (only per-block feature correlations are computed to save space), we find similar patterns in TF, LWFT, and our TTL also: features in bottom blocks are substantially more correlated than those in top blocks, and features in block4—which we remove in our TTL—are almost re-learned as their correlation with the original features come close to that between random features.

b) Features in top layers become more correlated with bottom layers after TL: The "horizontal" analysis above says the features of the top layers are almost re-learned in FTL, LWFT, and TF, but it remains unclear what features are learned there. If we believe that high-level features are probably not useful for COVID classification [36], a reasonable hypothesis is that these top layers actually learn features that are more correlated with those of lower layers after TL. This seems indeed the case, as shown in Fig. 6: the correlation level between the block2 and block 4 features, as well as between block3 and block4 features, increases both visibly and quantitatively.

Given the above two sets of findings, our idea in TTL to remove the redundant top layers and keep the essential bottom layers is reasonable toward effective and compact models.

IV. Experiments

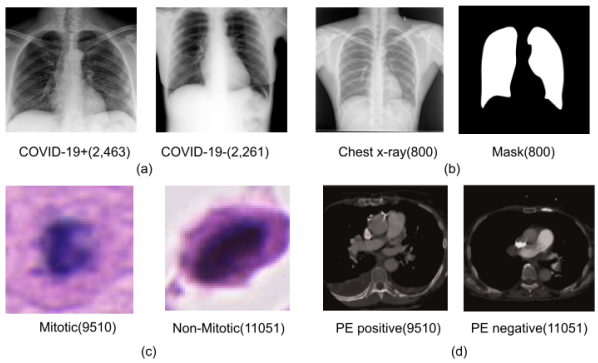


Fig. 7. Exemplary image of (a) BIMCV chest x-rays, (b) 2D lung segmentation data, (c) MIDOG2022 dataset, and (d) PE dataset. Numbers in parenthesis indicate the number of data points in each category.

A. Experiment setup

We systematically compare FTL, TF, LWFT, and our TTL on 3 MIC tasks covering both 2D and 3D image modalities (Sections IV-B to IV-D), and also explore a 2D lung segmentation task (Section IV-E). All 2D experiments follow the same basic protocol unless otherwise stated: **1)** the given dataset is split into 64% training, 16% validation, and 20% test; **2)** ResNet50 is the default model for all 2D MIC tasks, and ResNet34 for 2D segmentation; **3)** all 2D images are center-cropped and resized to 224×224; **4)** random cropping and slight random rotation are used for data augmentation during TL; **5)** the ADAM optimizer is used for all TL

methods, with an initial LR 10^{-4} and a batch size 64; The `ReduceLROnPlateau` scheduler in PyTorch is applied to adaptively adjust the LR: when the validation AURPC stagnates, the LR is decreased by $1/2$. Training is stopped when the LR drops below 10^{-7}; **6)** the best model is chosen based on the validation AUPRC. For the 3D MIC task in Section IV-D, we choose ResNeXt3D-101 as the default model, and compare it with PENet which is a handcrafted model in [39]. Both models are first pretrained on the kinetics-600 dataset [40], and then finetuned with the same setting as in [39]: 0.01 initial LR for randomly initialized weights and 0.1 for pretrained weights, SGD (momentum $= 0.9$) optimizer, cosine annealing LR scheduler, 100 epochs of training, and best model selected based on the validation AUROC. For experiments involving randomness, we repeat them 3 times and report the mean and standard deviation. Runtime analysis is performed on a system with Intel Core i9-9920X CPU and Quadro RTX 6000 GPU. We report AUROC/AUPRC for MIC, and Dice Coefficient/Jaccard Index for segmentation as performance metrics, and Params, MACs, CPU/GPU time as complexity metrics (see discussion below Table II for definitions).

B. COVID-19 chest x-ray image classification

We take the chest x-rays from the BIMCV-COVID19+ (containing COVID positives) and BIMCV-COVID19− (containing COVID negatives) datasets (iteration 1) [35][5]. We manually remove the small number of lateral views and outliers, leaving 2261 positives and 2463 negatives for our experiment; see Fig. 7 (a) for a couple of x-ray samples. We have demonstrated the plausibility and superiority of TTL based on this MIC task around Table II and Figs. 4 to 6, which we do not repeat here.

C. Mitotic cells classification

The density of mitotic cells undergoing division (i.e., mitotic figures) is known to be related to tumor proliferation and can be used for tumor prognosis [41]. Since cell division changes its morphology, we expect mid-level blob-like features to be the determinant here. For this experiment, we take the dataset from the mitotic domain generalization challenge (MIDOG2022) [41]. The challenge is about detection of mitotic figures, i.e., the training set consists of properly cropped 9501 mitotic figures and 11051 non-mitotic figures (see Fig. 7 for sample figures), and the task is to localize mitotic figures on large pathological slices during the test. We modify the task into a binary MIC (positives are mitotic figures, and negatives are non-mitotic figures): we take their training set and further divide it into training, validation, and test sets, following the ratios discussed in Section IV-A.

Table III summarizes the results obtained by the various methods. We observe that: **(1)** Our TTL-2 and TTL-1 beat all other TL methods, and also yield the most compact and inference-efficient models; **(2)** Both TTL-1/2 and TF find the best cutoffs at the transition of block3 and block4, implying that high-level features are possibly unnecessary and can even be hurtful for this task and confirming our tuition that mid-level visual features are likely be crucial for decision; **(3)** Of all methods, LWFT with block-wise search performs the worst; even after layer-wise search, its AUROC only matches that of the baseline FTL. The inferior performance of LWFT implies that only fine-tuning the top layers is not sufficient for this task, consistent with our observation in (2).

TABLE III
MITOTIC CELLS CLASSIFICATION WITH DIFFERENT TL STRATEGIES. THE BEST RESULT OF EACH COLUMN IS COLORED IN **RED**. ↑ INDICATES LARGER VALUE IS BETTER AND ↓ INDICATES LOWER VALUE IS BETTER. "-1" MEANS WITH THE BLOCK-WISE SEARCH ONLY, AND "-2" MEANS WITH THE TWO-STAGE BLOCK-LAYER HIERARCHICAL SEARCH.

Method	AUROC↑	AUPRC↑	Params(M)↓	MACs(G)↓	CPU(ms)↓	GPU(ms)↓
FTL	0.925 ± 0.003	0.917 ± 0.003	23.5	4.12	80.1	3.59
TF-1	0.925 ± 0.002	0.918 ± 0.002	12.9	3.56	65.3	3.50
LWFT-1	0.920 ± 0.003	0.913 ± 0.005	23.5	4.12	78.4	3.55
TTL-1	**0.928 ± 0.004**	**0.921 ± 0.003**	**8.55**	**3.31**	**69.6**	**2.99**
TF-2	0.925 ± 0.002	0.918 ± 0.002	12.9	3.56	65.3	3.50
LWFT-2	0.925 ± 0.001	0.919 ± 0.001	23.5	4.12	78.2	3.56
TTL-2 (ours)	**0.928 ± 0.004**	**0.921 ± 0.003**	**8.55**	**3.31**	**69.6**	**2.99**

TABLE IV
3D PULMONARY EMBOLISM CLASSIFICATION WITH DIFFERENT TL STRATEGIES. THE BEST RESULT OF EACH COLUMN IS COLORED IN **RED**. ↑ INDICATES LARGER VALUE IS BETTER AND ↓ INDICATES LOWER VALUE IS BETTER. "-1" MEANS WITH THE BLOCK-WISE SEARCH ONLY, AND "-2" MEANS WITH THE TWO-STAGE BLOCK-LAYER HIERARCHICAL SEARCH. NOTE THAT THE RUN TIME FOR THIS TABLE IS IN SECONDS, NOT MILLISECONDS.

Method	AUROC↑	AUPRC↑	Params(M)↓	MACs(G)↓	CPU(s)↓	GPU(s)↓
PENet	0.822 ± 0.010	0.855 ± 0.007	28.4	**51.7**	1.50	**1.59e-2**
FTL	0.821 ± 0.010	0.867 ± 0.006	47.5	66.3	1.44	1.96e-2
TF-1	0.849 ± 0.020	0.886 ± 0.017	36.1	64.9	1.41	1.93e-2
LWFT-1	0.817 ± 0.005	0.855 ± 0.003	47.5	66.3	1.44	1.96e-2
TTL-1	**0.854 ± 0.013**	**0.889 ± 0.015**	**26.11**	60.17	**1.32**	1.68e-2
TF-2	0.849 ± 0.020	0.886 ± 0.017	36.1	64.9	1.41	1.93e-2
LWFT-2	0.835 ± 0.038	0.870 ± 0.028	47.5	66.3	1.44	1.96e-2
TTL-2(ours)	**0.854 ± 0.013**	**0.889 ± 0.015**	**26.11**	60.17	**1.32**	1.68e-2

D. Pulmonary embolism CT image classification

Pulmonary Embolism (PE) is a blockage of the blood vessels connecting the lungs and the heart, and CT pulmonary angiography (CTPA) is the gold standard for its diagnosis [42]. To advance DCNN-based diagnosis of PE, [39] proposes the first public PE dataset consisting of 1797 CT images from 1773 patients, and a handcrafted classification model, PENet, that is based on 3D DCNNs. Applying the standard FTL (pretraining on kinetics-600 [40]) on PENet and competing models, [39] shows that PENet outperforms other SOTA 3D DCNN models, such as ResNet3D-50, ResNeXt3D-101, and DenseNet3D-121.

On CT images, PE often appears as localized blobs that map to mid-level visual features. So the suboptimal TL performance of the SOTA models is mostly likely due to the rigid FTL strategy. We confirm this on ResNeXt3D-101 pretrained

[5] https://bimcv.cipf.es/bimcv-projects/bimcv-covid19/#1590858128006-9e640421-6711.

on kinetics-600: after layer-wise search, all differential TL methods including TF, LWFT, and TTL outperform PENet by considerable margins in both AUROC and AUPRC, as shown in Table IV. Also, both TTL-1 and TF-1 find the best cutoff at the transition of block3 and block4, another confirmation of our intuition that probably only low- to mid-level features are needed here. We note that although the AUROC obtained via FTL is slightly lower than that of PENet, the AUPRC is actually higher—which [39] does not consider when drawing their conclusion. Our TTL-2 is a clear winner in performance, despite that PENet has been meticulously designed and optimized for the task! TTL is also highly competitive in inference speed, especially on CPUs.

TABLE V
2D LUNG SEGMENTATION WITH TTL. THE BEST RESULT OF EACH COLUMN IS COLORED IN **RED**. ↑ INDICATES LARGER VALUE IS BETTER AND ↓ INDICATES LOWER VALUE IS BETTER.

Method	Dice Coef ↑	Jaccard index ↑	Params(M) ↓	MACs(G) ↓	CPU(ms) ↓	GPU(ms) ↓
FTL	0.968 ± 0.029	0.940 ± 0.051	24.4	5.93	121	11.8
TTL_B1(Ours)	0.970 ± 0.029	0.941 ± 0.052	**21.3**	**1.68**	**47.8**	**7.05**
TTL_B2(Ours)	$\mathbf{0.972 \pm 0.027}$	$\mathbf{0.946 \pm 0.047}$	21.5	3.02	68.7	8.50
TTL_B3(Ours)	0.968 ± 0.029	0.939 ± 0.051	22.1	4.82	95.6	10.6

E. Chest X-ray lung segmentation

Despite our focus on MIC, we briefly explore the potential of TFL for segmentation also. To this end, we merge two popular public chest x-ray lung segmentation datasets: Montgomery Country XCR set (MC) and Shenzhen Hospital CXR Set (SH) [43], [44], both of which provide manual segmentation masks. MC consists of 58/80 tuberculosis/normal cases, and SH has 336/326 tuberculosis/normal cases; x-ray and mask samples can be found in Fig. 7(b).

We construct the TTL model in a manner similar to what we have done for the MIC tasks. We take the U-Net architecture with ResNet34 and pretrained on ImageNet as our backbone model. To ensure the skip-connection structure can be preserved after truncation, we symmetrically truncate the backbone and segmentation head simultaneously. We do not include comparison with LWFT and TF, as the original papers do not discuss how to extend them to segmentation and our MIC tasks above already show the superiority of our TTL.

For simplicity, we only perform block-wise truncation, and our experimental results are shown in Table V. TTL achieves the best segmentation performance at block 2 (TTL_B2) and outperforms FTL by 0.6% in terms of Dice Coefficient and Jaccard index (both standard metrics for evaluating segmentation performance). More importantly, TTL_B2 is more efficient than FTL and reduces the model size by 11.5% and inference time by over 50%.

F. Ablation study

1) Impact of network architecture: To study the impact of network architecture on the result of TTL, we compare three network models: ResNet50, DenseNet121, and EfficientNet-b0, which represent large-, mid-, and small-size models, respectively, on COVID-19 classification. The results are summarized in Table VI. It is clear that our TTL uniformly improves the model performance and reduces the model size by at least several fold, compared to FTL. In particular, even for the most lightweight model EfficientNet-b0, TTL still pushes up the performance by about 5% in both AUROC and AUPRC, and reduces the model size by about 4-fold.

TABLE VI
IMPACT OF NETWORK ARCHITECTURE ON TTL PERFORMANCE. THE BEST RESULTS IN EACH GROUP ARE COLORED IN **RED**.

	Method	AUROC	AUPRC	Params(M)	MACs(G)	Speed(s) CPU	Speed(s) GPU
RN50[1]	FTL	0.853 ± 0.004	0.861 ± 0.002	23.5	4.12	80.0	6.72
	TTL	$\mathbf{0.865 \pm 0.001}$	$\mathbf{0.870 \pm 0.007}$	**8.55**	**3.31**	**60.9**	**5.83**
DN201[2]	FTL	0.852 ± 0.002	0.856 ± 0.004	18.2	3.31	107	24.0
	TTL	$\mathbf{0.866 \pm 0.006}$	$\mathbf{0.871 \pm 0.012}$	**1.53**	**2.11**	**46.2**	**5.94**
ENb0[3]	TL	0.795 ± 0.004	0.794 ± 0.002	3.63	0.378	30.0	8.33
	TTL	$\mathbf{0.842 \pm 0.001}$	$\mathbf{0.843 \pm 0.007}$	**0.896**	**0.255**	**24.0**	**6.46**

Abbreviation: [1] ResNet50, [2]:DenseNet201, [3] EfficientNet B0

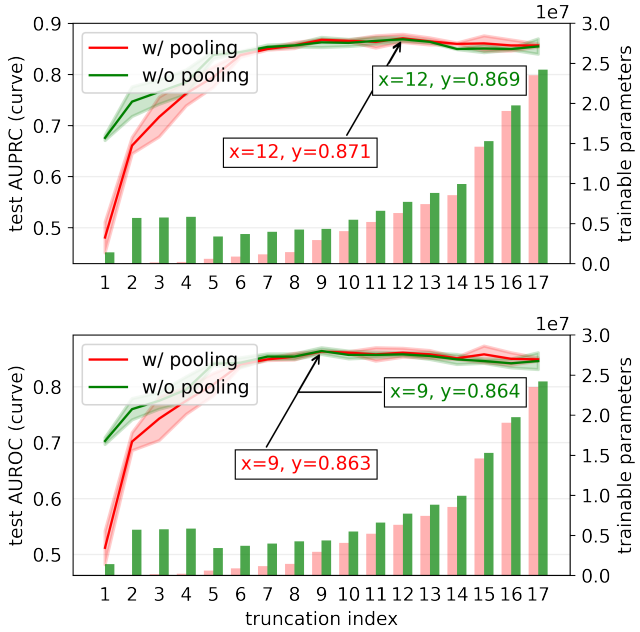


Fig. 8. ResNet50 transferred from ImageNet to BIMCV using our TTL. We report the performance in both AURPC (top) and AUROC (bottom). The red and green curves represent TTL performance with and without the extra adaptive pooling layer between the final convolution layer and the MLP classifier. Arrows point to the peak performance.

2) Impact of pooling: After truncation, we can either directly pass the full set of features or downsample them before feeding them into the MLP classifier. For deep learning models, we can easily perform subsampling by inserting a pooling layer. For this purpose, we use PyTorch's built-in function `AdaptiveAvgPool2d` to downsample the feature map to 1×1 in the spatial dimension, which produces the most compact spatial features. We compare models with vs. without this extra pooling layer on COVID-19 classification, and present the results in Fig. 8. We observe no significant gap between the two settings in terms of peak performance measured by both AUPRC and AUROC. However, the setting with pooling induces a lower-dimensional input to the MLP classifier, and hence contains much fewer trainable parameters

compared to that without.

G. Exploratory study

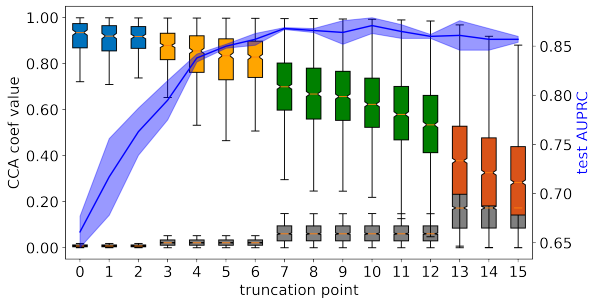


Fig. 9. SVCCA on ResNet50 transferred from ImageNet to BIMCV. The plot is inherited from Fig. 4. The gray boxes and the associated bars represent the distributions of CCA coefficients between random features.

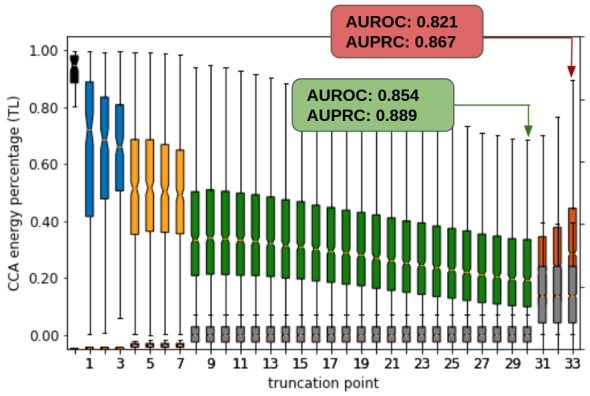


Fig. 10. SVCCA on ResNeXt3D-101 transferred from Knetic-600 to CTPA. Truncating at layer 30 (marked in green) significantly outperforms FTL (marked in red).

1) Detecting the near-best truncation point: Our two-stage search strategy for good truncation points (see Section III-B) is effective as verified above, but can be costly due to the need for multiple rounds of training, each for one candidate truncation point. In this subsection, we explore an alternative strategy to cut down computation: the idea is to detect the transition point where feature reuse becomes negligible. For this, we recall the SVCCA analysis in Fig. 4: we use the distribution of the SVCCA coefficients to quantify the correlation of features before and after FTL. To determine when the correlation becomes sufficiently small, we compare the correlation with that between random features.

We quickly test the new strategy on both COVID-19 and CT-based PE classification, shown in Fig. 9 and Fig. 10, respectively. From Fig. 9, we can see that for COVID-19 classification, feature reuse becomes limited after the 12-th candidate truncation point: from 13-th onward, feature correlation becomes very close to that between random features, suggesting that the features are almost uncorrelated. As expected, the 12-th truncation point also achieves near-optimal performance as measured by AUPRC. Similarly, for CT-based PE classification in Fig. 10, this strategy suggests the 30-th truncation point, which yields the same result as that we have obtained using the two-stage search strategy as reported in Table IV, in both AUPRC and AUROC. Note that in Fig. 10, the SVCCA coefficients slightly increase after the 30-th truncation point due to dimensionality: the channel sizes are doubled there compared to the previous blocks, and the raised dimension induces higher correlation coefficients. But to decide if feature reuse becomes trivial, we only need to check if the two SVCCA-coefficient distributions are sufficiently close, not their absolute scalings.

Thus, this new strategy seems promising as a low-cost alternative to our default two-stage search strategy. We summarize the key steps as follows.

1) **Step 1**: perform FTL on the target dataset;
2) **Step 2**: compute SVCCA coefficients between the features before and after FTL, at all candidate truncation points;
3) **Step 3**: compute SVCCA coefficients between random features, at all candidate truncation points;
4) **Step 4**: locate the truncation point where the distribution of the two groups of SVCCA coefficients from **Step 2** and **Step 3** becomes substantially overlapped, and take it as the truncation point;
5) **Step 5**: fine-tune the model with the truncation point selected from **Step 4**.

Compared to our two-stage strategy that entails numerous rounds of model finetuning, the new strategy only requires two rounds of finetuning: one from **Step 1** on the whole model, and the other from **Step 5** on the truncated model.

TABLE VII
SYMPTOM OF LUNG DISEASE IN CHEXPERT

Diseases	Shape	Edge	Contrast	level of features
No Finding	✗	✗	✗	None
Enlarged Cardiom.	✓	✗	✓	low and high
Cardiomegaly	✓	✗	✓	low and high
Lung Opacity	✗	✗	✓	low
Lung Lesion	✓	✗	✓	low and high
Edema	✗	✗	✓	low and high
Consolidation	✗	✗	✓	low
Pneumonia	✗	✗	✓	low
Atelectasis	✓	✗	✓	low and high
Pneumonthorax	✗	✓	✓	low
Pleural Effusion	✗	✗	✓	low
Pleural Other	✗	✗	✓	low
Fracture	✓	✓	✓	low and high
Support device	✓	✓	✓	low and high

2) Inter-domain TL: All we have discussed above are intra-domain TL where the source domain is distinct from the target domain. It is tempting to think inter-domain TL might be more effective. To test this, we take the x-ray-based COVID-19 classification task again, but now finetune the ResNet50 model pretrained on CheXpert, a massive-scale x-ray dataset ($\sim 220K$ images) for disease prediction covering 13 types

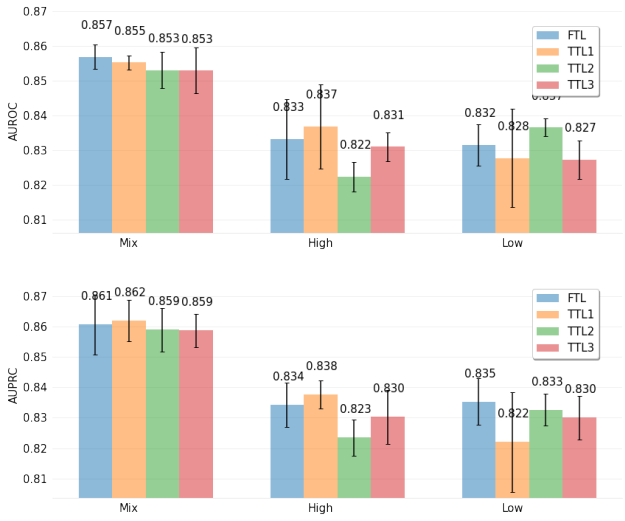


Fig. 11. ResNet50 transferred from CheXpert to BIMCV. (top) test AUROC score; (bottom) test AUPRC score. "Mix", "High", and "Low" mean finetuning from pretrained models on the original CheXpert, CheXpert-high, and CheXpert-low, respectively. TTLx means the truncation point is chosen at the transition point between block x and block x+1.

of diseases (see Table VII) [45]. These diseases correspond to varying levels of visual features. Hence, we categorize them into two groups: **CheXpert-low** includes diseases that need low-level features only, and **CheXpert-high** covers those needing both low- and high-level features. A summary of the categorization can be found in Table VII[6]. We pretrain ResNet50 on 3 variants of the dataset, respectively: full CheXpert, CheXpert-low, and CheXpert-high, and then compare FTL and our TTL on the three resulting models. For TTL, we only perform a coarse-scale search and pick the 3 transition layers between the 4 blocks in ResNet50 as truncation points. As always, all experiments are repeated three times.

From the results summarized in Fig. 11, we find that: **(1)** Our TTL always achieves comparable or superior performance compared to FTL, reaffirming our conclusion above; **(2)** Transferring from the model pretrained on the original dataset substantially outperforms transferring from those pretrained on the CheXpert-high and CheXpert-low subsets. This can be explained by the diversity of feature levels learned during pretraining: on the subsets more specialized features are learned; in particular, on CheXpert-low, perhaps only relatively low-level features are learned; and **(3)** Notably, our results show that medical-to-medical TL does not do better than TL from natural images, when we compare the results in Fig. 11 with those in Table II. We suspect this is because feature diversity is the most crucial quality required on the pretrained models in TL, and models pretrained on natural images perhaps already learn sufficiently diverse visual features.

V. Conclusion

In this paper, we systematically examine TL via feature reuse in the context of deep learning for MIC. Considering that most MIC tasks do not need high-level visual features, we propose an effective TTL method that almost always outperforms FTL and produces substantially smaller models. We propose a two-stage search strategy for finding reasonable truncation points (Section III-B), complemented by an experimental lightweight alternative (Section IV-G). Our preliminary result on medical image segmentation (Section IV-E) shows similar advantages of our method.

There are numerous open directions naturally following: **(1) Alternative deep architectures**. Transformers have been an emerging family of deep models for numerous vision tasks. So far, we have not seen much work reporting successfully transferring pretrained Transformer models to MIC [46], [47], presumably due to the lack of effective finetuning methods: Transformer models are well known to be much more data-hungry and pretraining-hungry than convolutional models; **(2) General medical imaging tasks**. It will be interesting to systematically study our method for other major types of medical imaging tasks, such as segmentation, registration, reconstruction; **(3) Other mismatches between the source and target**. We address the different feature levels between the source and target in TL. But there can be many other types of mismatches, e.g., different tasks (classification to segmentation) and different imbalance ratios. How to derive the optimal TL to address these scenarios? **(4) Compatibility with emerging learning frameworks**. Federated learning and imbalanced learning [48], [49] are emerging learning paradigms to address central issues in medical imaging tasks. New developments in TL should ensure the compatibility with these new learning methods.

[6]Disease symptoms information obtained from Mayo clinic: https://www.mayoclinic.org/symptom-checker/select-symptom/itt-20009075

References

[1] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[2] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[3] G. van Tulder and M. de Bruijne, "Combining generative and discriminative representation learning for lung ct analysis with convolutional restricted boltzmann machines," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1262–1272, 2016.

[4] A. Rehman, S. Naz, M. I. Razzak, F. Akram, and M. Imran, "A deep learning-based framework for automatic brain tumors classification using transfer learning," *Circuits, Systems, and Signal Processing*, vol. 39, no. 2, pp. 757–775, 2020.

[5] B. Q. Huynh, H. Li, and M. L. Giger, "Digital mammographic tumor classification using transfer learning from deep convolutional neural networks," *Journal of Medical Imaging*, vol. 3, no. 3, p. 034501, 2016.

[6] N. Antropova, B. Q. Huynh, and M. L. Giger, "A deep feature fusion methodology for breast cancer diagnosis demonstrated on three imaging modality datasets," *Medical physics*, vol. 44, no. 10, pp. 5162–5171, 2017.

[7] M. Ghafoorian, A. Mehrtash, T. Kapur, N. Karssemeijer, E. Marchiori, M. Pesteie, C. R. Guttmann, F.-E. de Leeuw, C. M. Tempany, B. Van Ginneken *et al.*, "Transfer learning for domain adaptation in mri: Application in brain lesion segmentation," in *International conference on medical image computing and computer-assisted intervention*. Springer, 2017, pp. 516–524.

[8] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[9] H. Q. Nguyen, H. H. Pham, N. T. Nguyen, D. B. Nguyen, M. Dao, V. Vu, K. Lam, and L. T. Le, "Vinbigdata chest x-ray abnormalities detection," url=https://www.kaggle.com/c/vinbigdata-chest-xray-abnormalities-detection, 2021.

[10] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, 2014, pp. 3320–3328.

[11] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[12] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio, "Transfusion: Understanding transfer learning for medical imaging," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019, pp. 3347–3357.

[13] M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein, "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability," *Advances in neural information processing systems*, vol. 30, 2017.

[14] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International conference on machine learning*. PMLR, 2014, pp. 647–655.

[15] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806–813.

[16] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[17] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*. Springer, 2018, pp. 270–279.

[18] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[19] P. Tschandl, C. Rosendahl, and H. Kittler, "The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Scientific data*, vol. 5, no. 1, pp. 1–9, 2018.

[20] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *arXiv preprint arXiv:2104.13478*, 2021.

[21] K. M. Hosny, M. A. Kassem, and M. M. Foaud, "Skin cancer classification using deep learning and transfer learning," in *2018 9th Cairo international biomedical engineering conference (CIBEC)*. IEEE, 2018, pp. 90–93.

[22] J. Sun, L. Peng, T. Li, D. Adila, Z. Zaiman, G. B. Melton, N. Ingraham, E. Murray, D. Boley, S. Switzer *et al.*, "A prospective observational study to investigate performance of a chest x-ray artificial intelligence diagnostic support tool across 12 us hospitals," *arXiv preprint arXiv:2106.02118*, 2021.

[23] A. Van Opbroek, M. A. Ikram, M. W. Vernooij, and M. De Bruijne, "Transfer learning improves supervised image segmentation across imaging protocols," *IEEE transactions on medical imaging*, vol. 34, no. 5, pp. 1018–1030, 2014.

[24] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[25] Z. Jiang, H. Zhang, Y. Wang, and S.-B. Ko, "Retinal blood vessel segmentation using fully convolutional network with transfer learning," *Computerized Medical Imaging and Graphics*, vol. 68, pp. 1–15, 2018.

[26] H. E. Kim, A. Cosa-Linan, N. Santhanam, M. Jannesari, M. E. Maros, and T. Ganslandt, "Transfer learning for medical image classification: a literature review," *BMC Medical Imaging*, vol. 22, no. 1, p. 69, Apr 2022. [Online]. Available: https://doi.org/10.1186/s12880-022-00793-7

[27] S.-C. Huang, T. Kothari, I. Banerjee, C. Chute, R. L. Ball, N. Borus, A. Huang, B. N. Patel, P. Rajpurkar, J. Irvin *et al.*, "Penet—a scalable deep-learning model for automated diagnosis of pulmonary embolism using volumetric ct imaging," *NPJ digital medicine*, vol. 3, no. 1, pp. 1–9, 2020.

[28] P. Wang, Q. Chen, X. He, and J. Cheng, "Towards accurate post-training network quantization via bit-split and stitching," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9847–9856.

[29] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.

[30] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," *arXiv preprint arXiv:1802.05668*, 2018.

[31] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[32] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[33] M. Raghu, K. Blumer, R. Sayres, Z. Obermeyer, B. Kleinberg, S. Mullainathan, and J. Kleinberg, "Direct uncertainty prediction for medical second opinions," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5281–5290.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[35] M. d. l. I. Vayá, J. M. Saborit, J. A. Montell, A. Pertusa, A. Bustos, M. Cazorla, J. Galant, X. Barber, D. Orozco-Beltrán, F. García-García *et al.*, "Bimcv covid-19+: a large annotated dataset of rx and ct images from covid-19 patients," *arXiv preprint arXiv:2006.01174*, 2020.

[36] H. Shi, X. Han, N. Jiang, Y. Cao, O. Alwalid, J. Gu, Y. Fan, and C. Zheng, "Radiological findings from 81 patients with covid-19 pneumonia in wuhan, china: a descriptive study," *The Lancet infectious diseases*, vol. 20, no. 4, pp. 425–434, 2020.

[37] V. Sovrasov. (2019) Flops counter for convolutional networks in pytorch framework. [Online]. Available: https://github.com/sovrasov/flops-counter.pytorch/

[38] V. Uurtio, J. M. Monteiro, J. Kandola, J. Shawe-Taylor, D. Fernandez-Reyes, and J. Rousu, "A tutorial on canonical correlation methods," *ACM Computing Surveys*, vol. 50, no. 6, pp. 1–33, nov 2018.

[39] S.-C. Huang, T. Kothari, I. Banerjee, C. Chute, R. L. Ball, N. Borus, A. Huang, B. N. Patel, P. Rajpurkar, J. Irvin, J. Dunnmon, J. Bledsoe, K. Shpanskaya, A. Dhaliwal, R. Zamanian, A. Y. Ng, and M. P. Lungren, "Penet—a scalable deep-learning model for automated diagnosis of pulmonary embolism using volumetric ct imaging," *npj Digital Medicine*, vol. 3, no. 1, p. 61, Apr 2020. [Online]. Available: https://doi.org/10.1038/s41746-020-0266-y

[40] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman, "A short note about kinetics-600," *arXiv preprint arXiv:1808.01340*, 2018.

[41] M. Aubreville, C. Bertram, M. Veta, K. Breininger, S. Jabari, and N. Stathonikos, "Mitosis domain generalization challenge 2022," 2022.

[42] C. Wittram, M. M. Maher, A. J. Yoo, M. K. Kalra, J.-A. O. Shepard, and T. C. McLoud, "CT angiography of pulmonary embolism: Diagnostic criteria and causes of misdiagnosis," *RadioGraphics*, vol. 24, no. 5, pp. 1219–1238, sep 2004.

[43] S. Jaeger, A. Karargyris, S. Candemir, L. Folio, J. Siegelman, F. Callaghan, Z. Xue, K. Palaniappan, R. K. Singh, S. Antani *et al.*, "Automatic tuberculosis screening using chest radiographs," *IEEE transactions on medical imaging*, vol. 33, no. 2, pp. 233–245, 2013.

[44] S. Candemir, S. Jaeger, K. Palaniappan, J. P. Musco, R. K. Singh, Z. Xue, A. Karargyris, S. Antani, G. Thoma, and C. J. McDonald, "Lung segmentation in chest radiographs using anatomical atlases with nonrigid registration," *IEEE transactions on medical imaging*, vol. 33, no. 2, pp. 577–590, 2013.

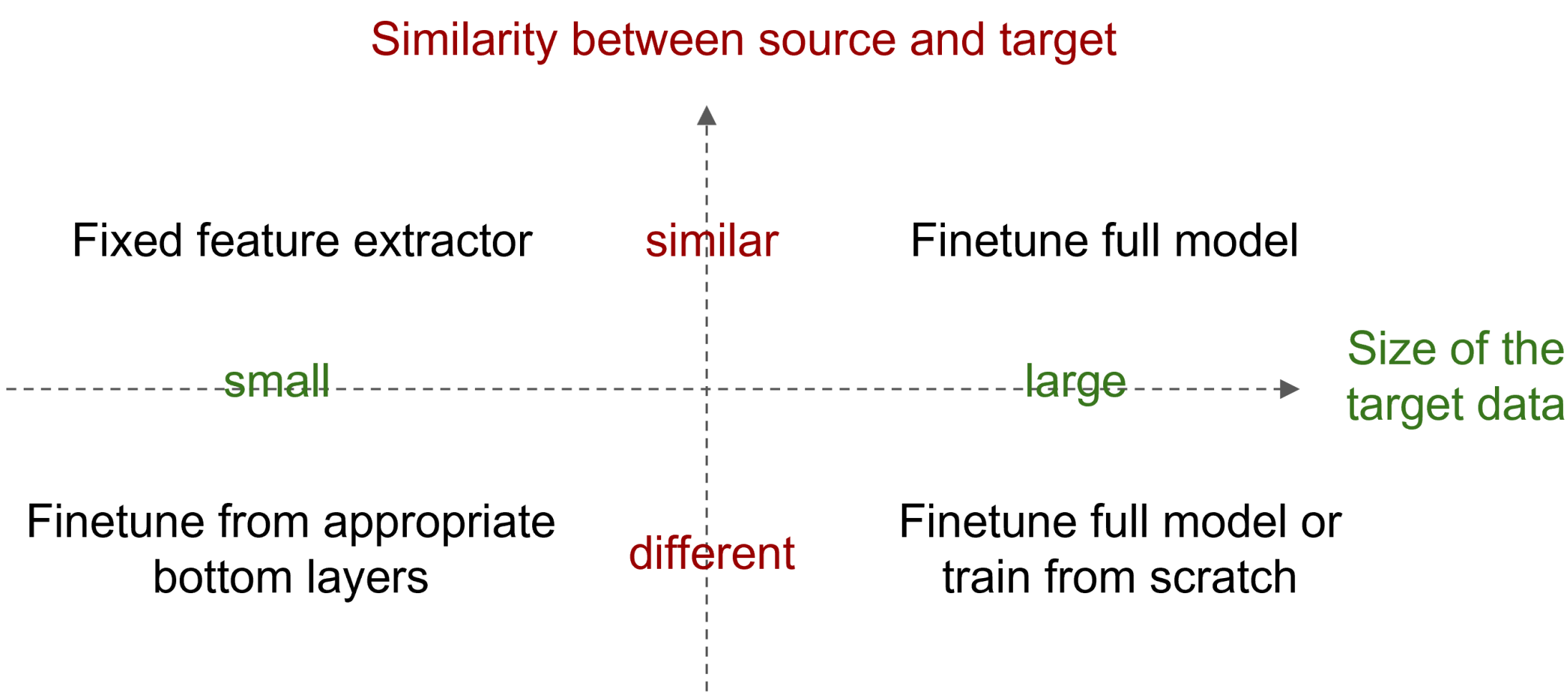
[45] J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya *et al.*, "Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 590–597.

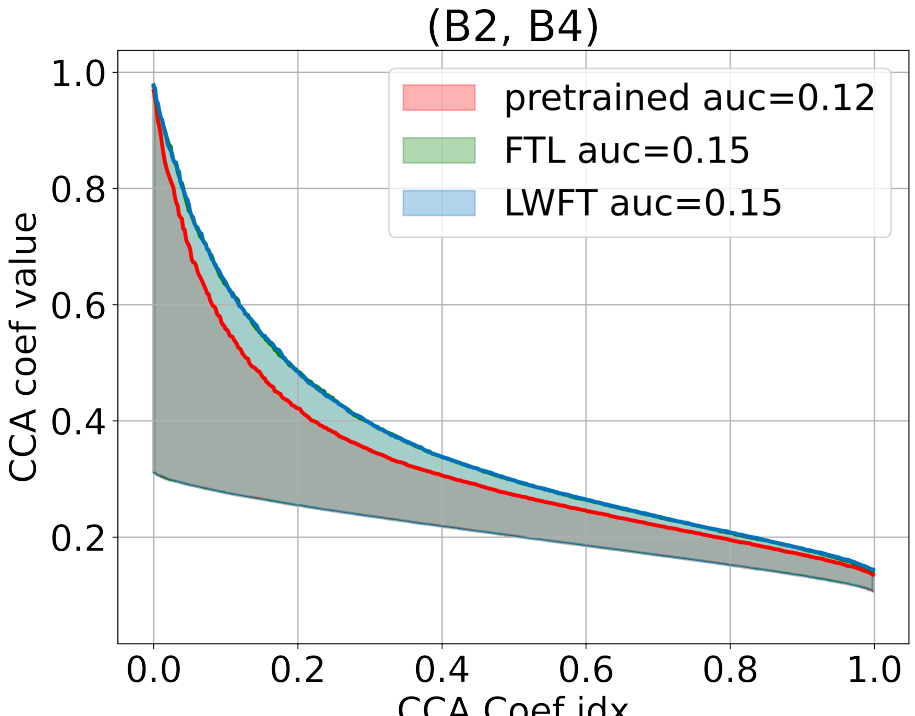
[46] C.-F. R. Chen, Q. Fan, and R. Panda, "Crossvit: Cross-attention multi-scale vision transformer for image classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 357–366.

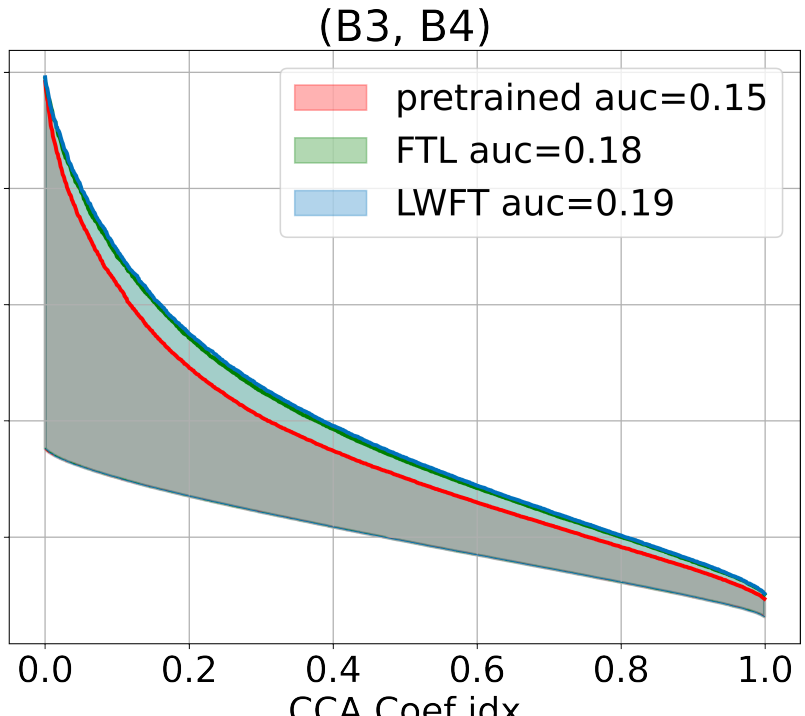
[47] J. M. J. Valanarasu, P. Oza, I. Hacihaliloglu, and V. M. Patel, "Medical transformer: Gated axial-attention for medical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2021, pp. 36–46.

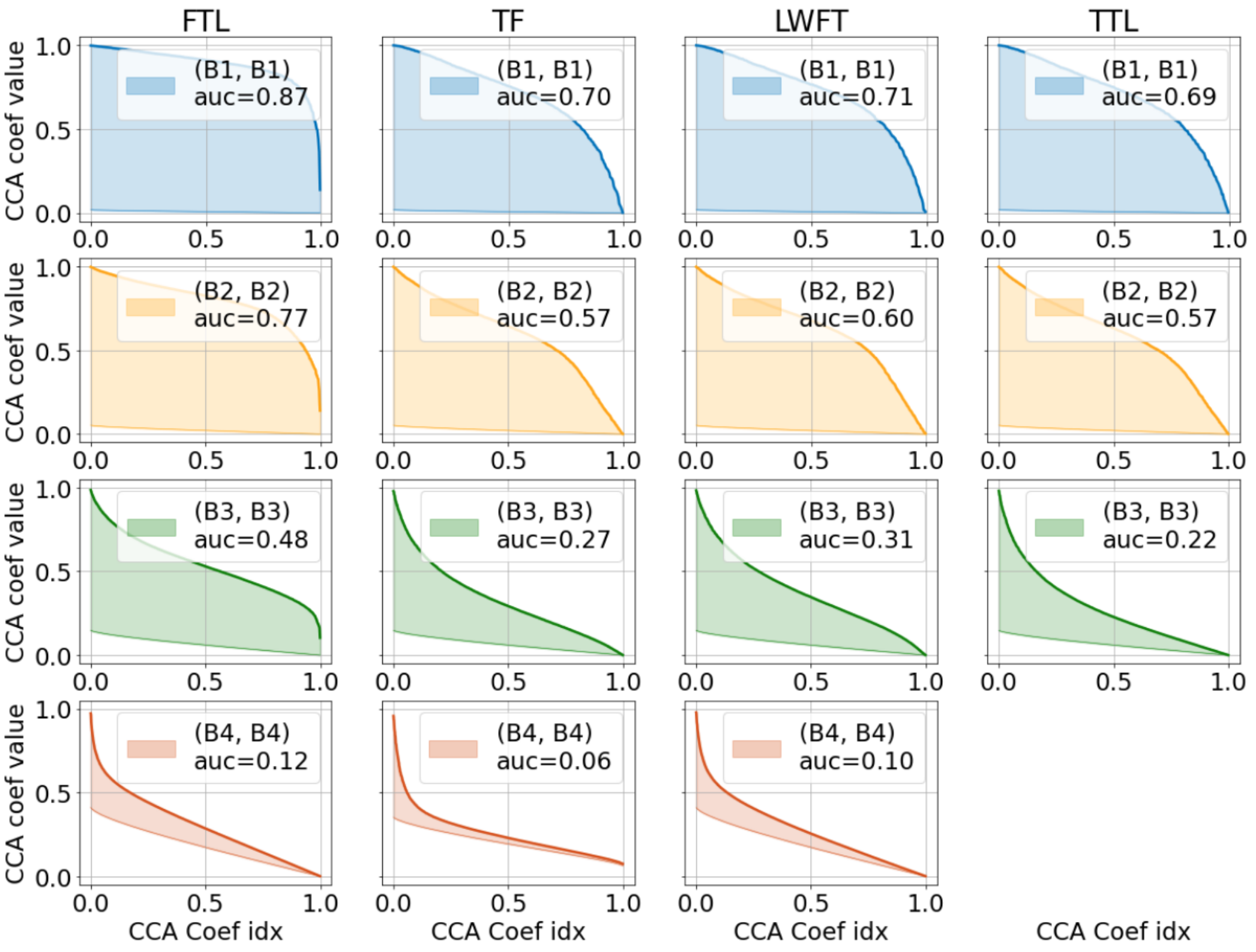
[48] L. Peng, G. Luo, A. Walker, Z. Zaiman, E. K. Jones, H. Gupta, K. Kersten, J. L. Burns, C. A. Harle, T. Magoc *et al.*, "Evaluation of federated learning variations for covid-19 diagnosis using chest radiographs from 42 us and european hospitals," *Journal of the American Medical Informatics Association*, 2022.

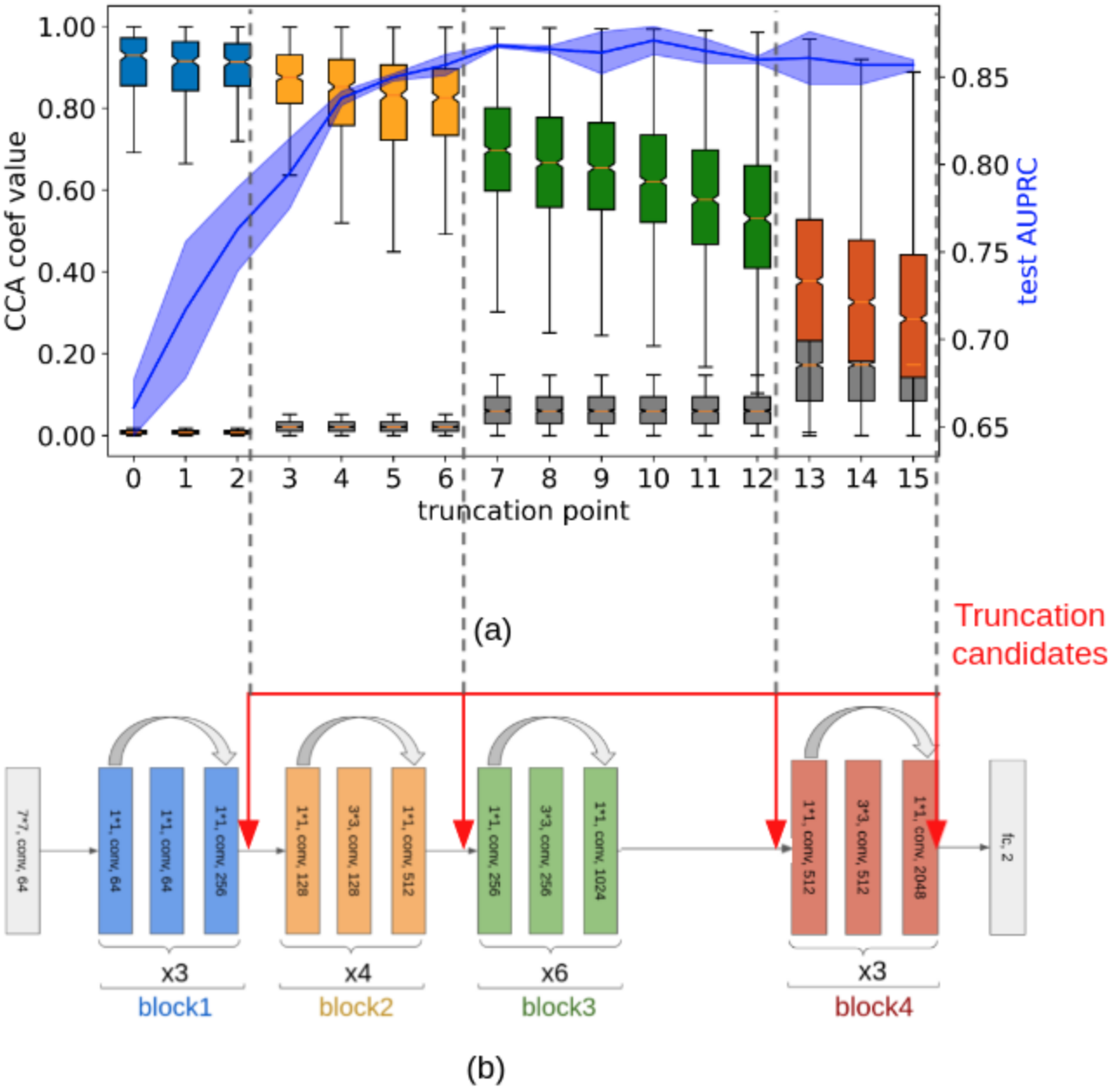
[49] L. Peng, Y. Travadi, R. Zhang, Y. Cui, and J. Sun, "Imbalanced classification in medical imaging via regrouping," *arXiv preprint arXiv:2210.12234*, 2022.

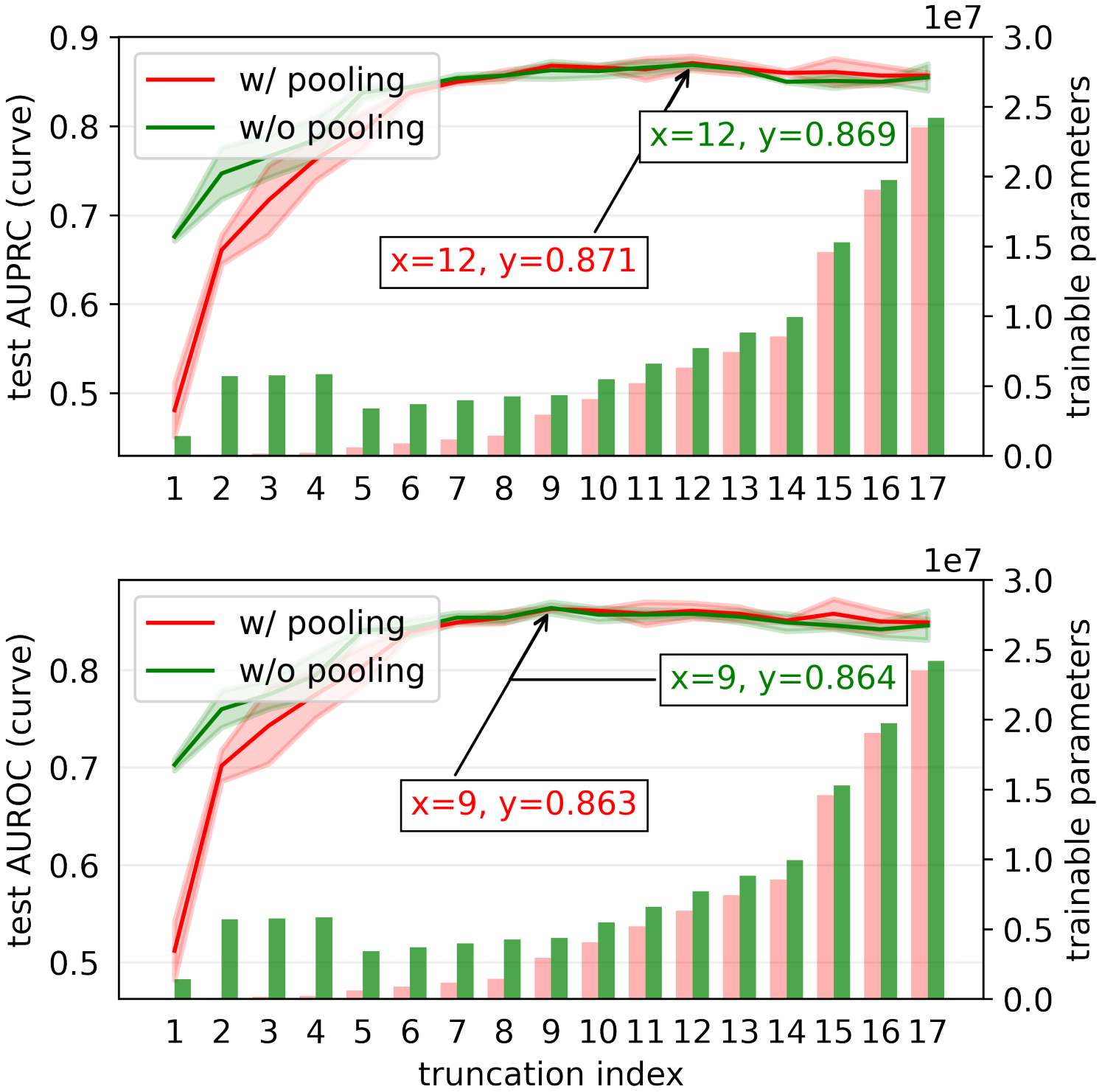

Similarity between source and target
Fixed feature extractor
similar
Finetune full model
small
large
Size of the target data
Finetune from appropriate bottom layers
different
Finetune full model or train from scratch

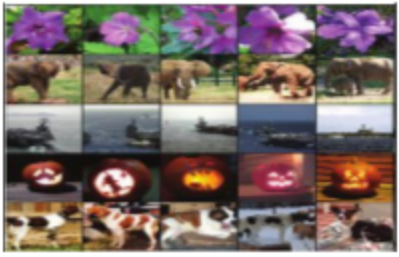

(B2, B4)
pretrained auc=0.12
FTL auc=0.15
LWFT auc=0.15
CCA coef value
CCA Coef idx
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0


(B3, B4)
pretrained auc=0.15
FTL auc=0.18
LWFT auc=0.19
CCA Coef idx
0.0
0.2
0.4
0.6
0.8
1.0


FTL
TF
LWFT
TTL
CCA coef value
CCA Coef idx
(B1, B1)
auc=0.87
(B1, B1)
auc=0.70
(B1, B1)
auc=0.71
(B1, B1)
auc=0.69
(B2, B2)
auc=0.77
(B2, B2)
auc=0.57
(B2, B2)
auc=0.60
(B2, B2)
auc=0.57
(B3, B3)
auc=0.48
(B3, B3)
auc=0.27
(B3, B3)
auc=0.31
(B3, B3)
auc=0.22
(B4, B4)
auc=0.12
(B4, B4)
auc=0.06
(B4, B4)
auc=0.10
0.0
0.5
1.0

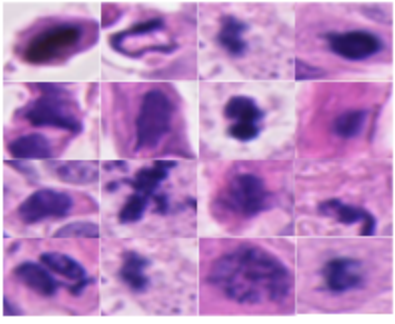

CCA coef value
test AUPRC
truncation point
(a)
Truncation candidates
7*7, conv, 64
1*1, conv, 64
1*1, conv, 64
1*1, conv, 256
1*1, conv, 128
3*3, conv, 128
1*1, conv, 512
1*1, conv, 256
3*3, conv, 256
1*1, conv, 1024
1*1, conv, 512
3*3, conv, 512
1*1, conv, 2048
fc, 2
x3
block1
x4
block2
x6
block3
x3
block4
(b)


1e7
test AUPRC (curve)
trainable parameters
w/ pooling
w/o pooling
x=12, y=0.869
x=12, y=0.871
0.9
0.8
0.7
0.6
0.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
1e7
test AUROC (curve)
trainable parameters
w/ pooling
w/o pooling
x=9, y=0.864
x=9, y=0.863
0.8
0.7
0.6
0.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
truncation index



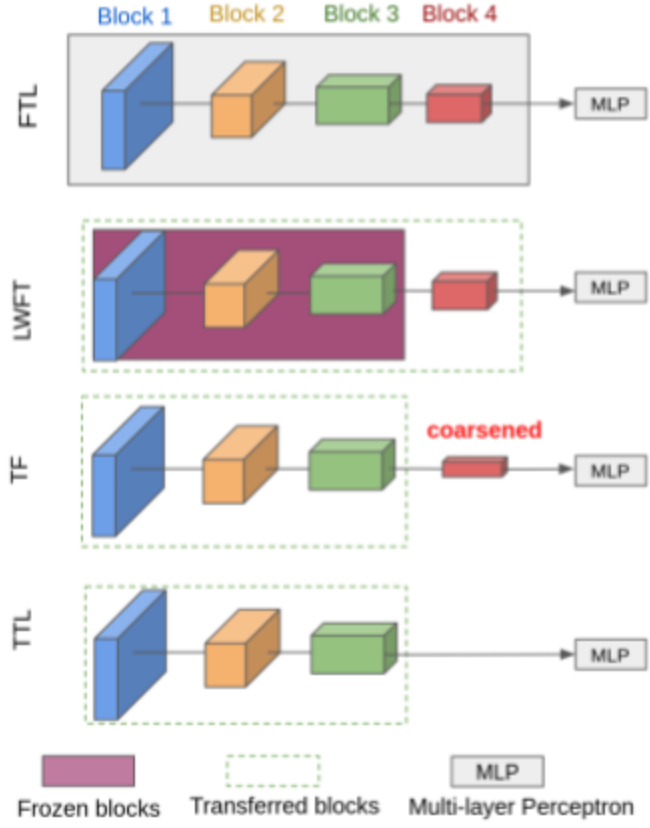
Source Domain
(ImageNet)

(a)

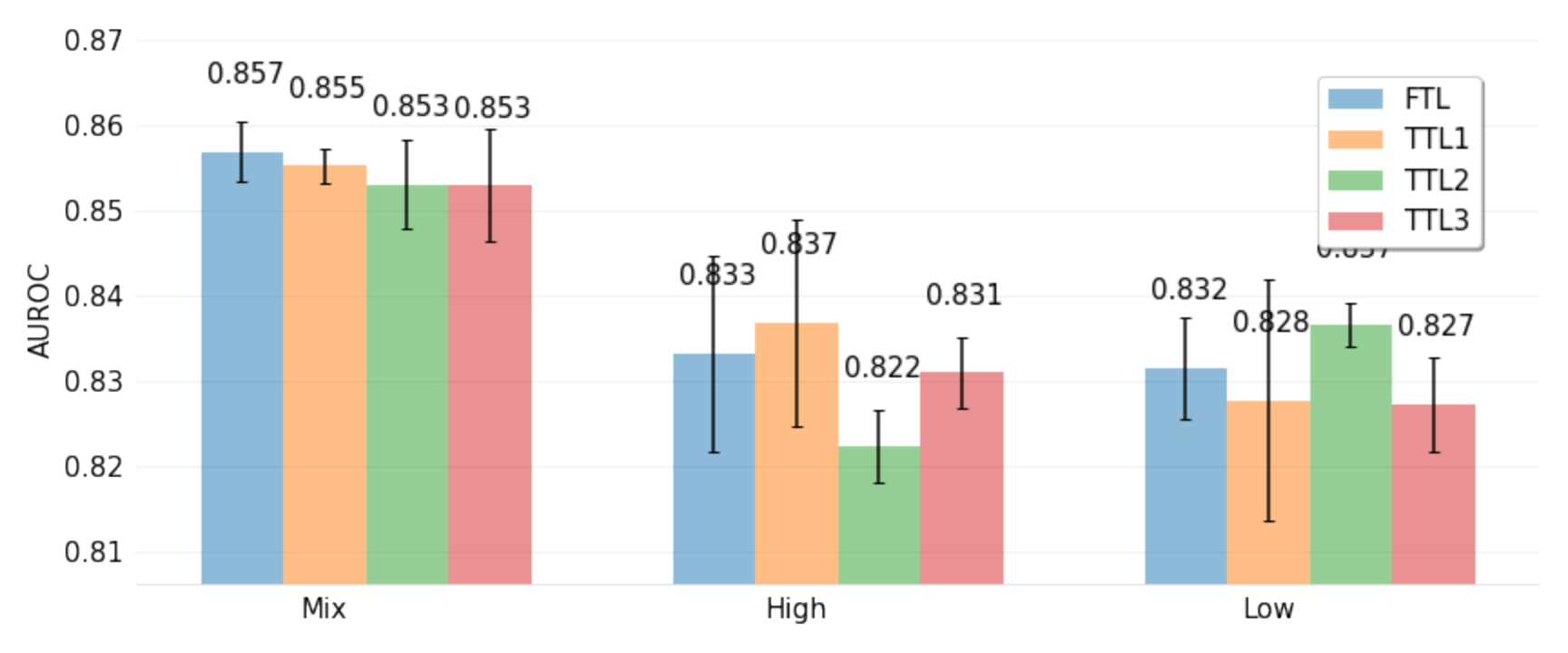


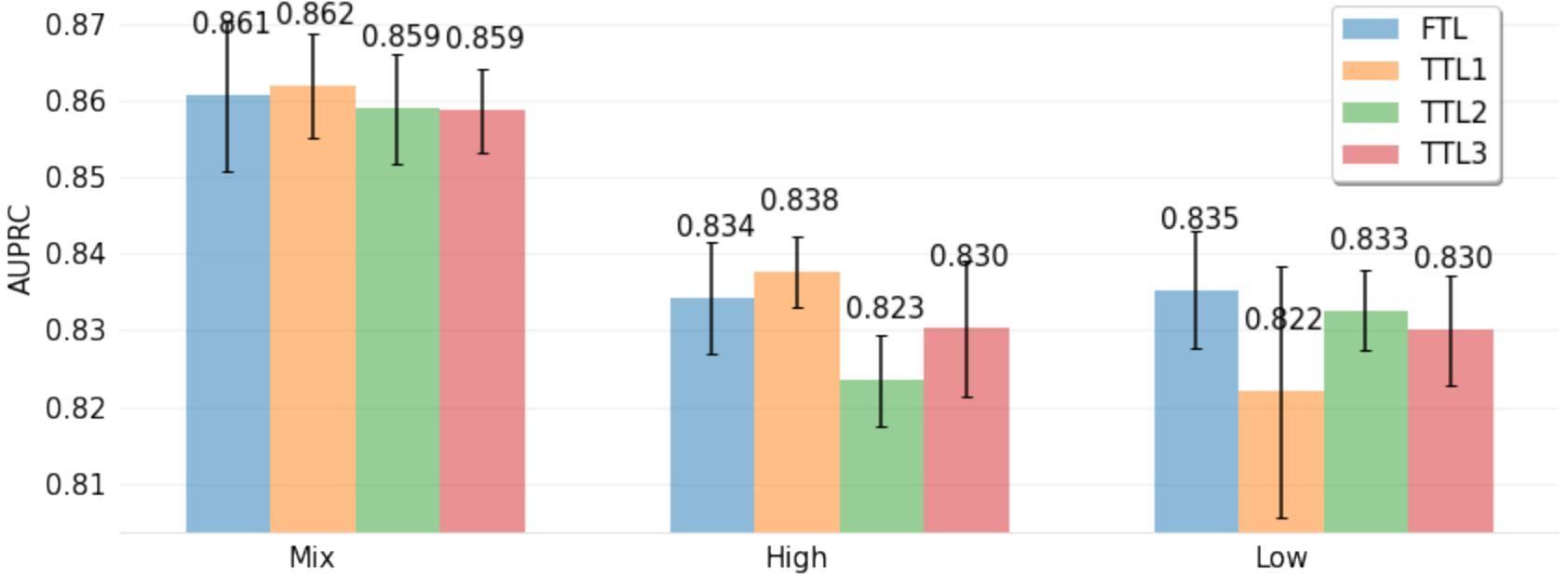
Target Domain
(MIDOG2022)

(b)



(c)

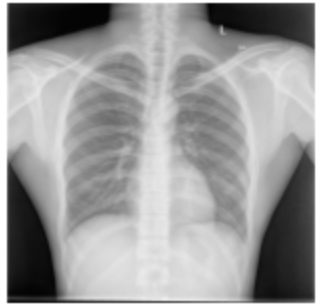

0.87
0.86
0.85
0.84
0.83
0.82
0.81
AUROC
0.857
0.855
0.853
0.853
0.833
0.837
0.822
0.831
0.832
0.828
0.827
FTL
TTL1
TTL2
TTL3
Mix
High
Low


0.87
0.86
0.85
0.84
0.83
0.82
0.81
AUPRC
0.861
0.862
0.859
0.859
0.834
0.838
0.823
0.830
0.835
0.822
0.833
0.830
Mix
High
Low
FTL
TTL1
TTL2
TTL3

COVID-19+(2,463)

COVID-19-(2,261)

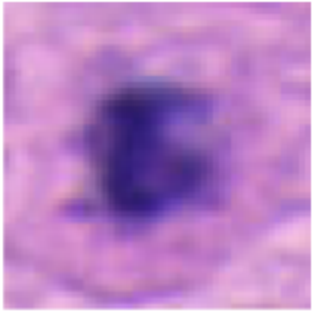
(a)



Chest x-ray(800)

Mask(800)

(b)

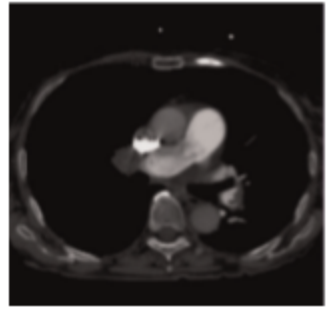


Mitotic(9510)

Non-Mitotic(11051)

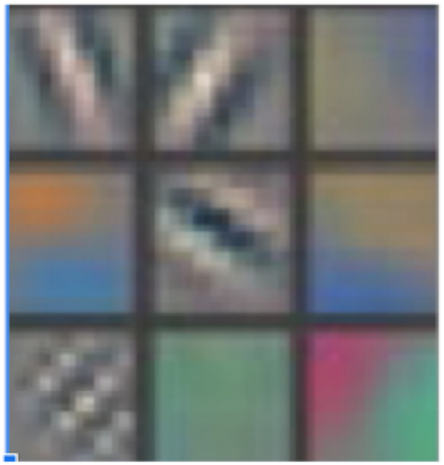
(c)

PE positive(9510)



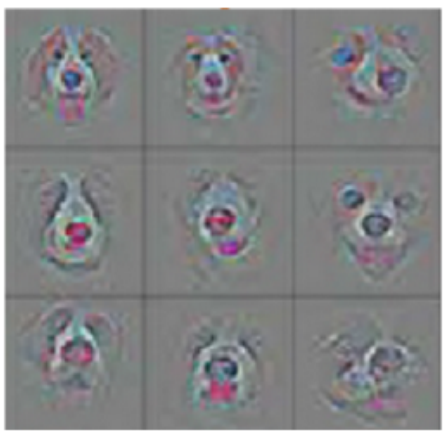
PE negative(11051)

(d)

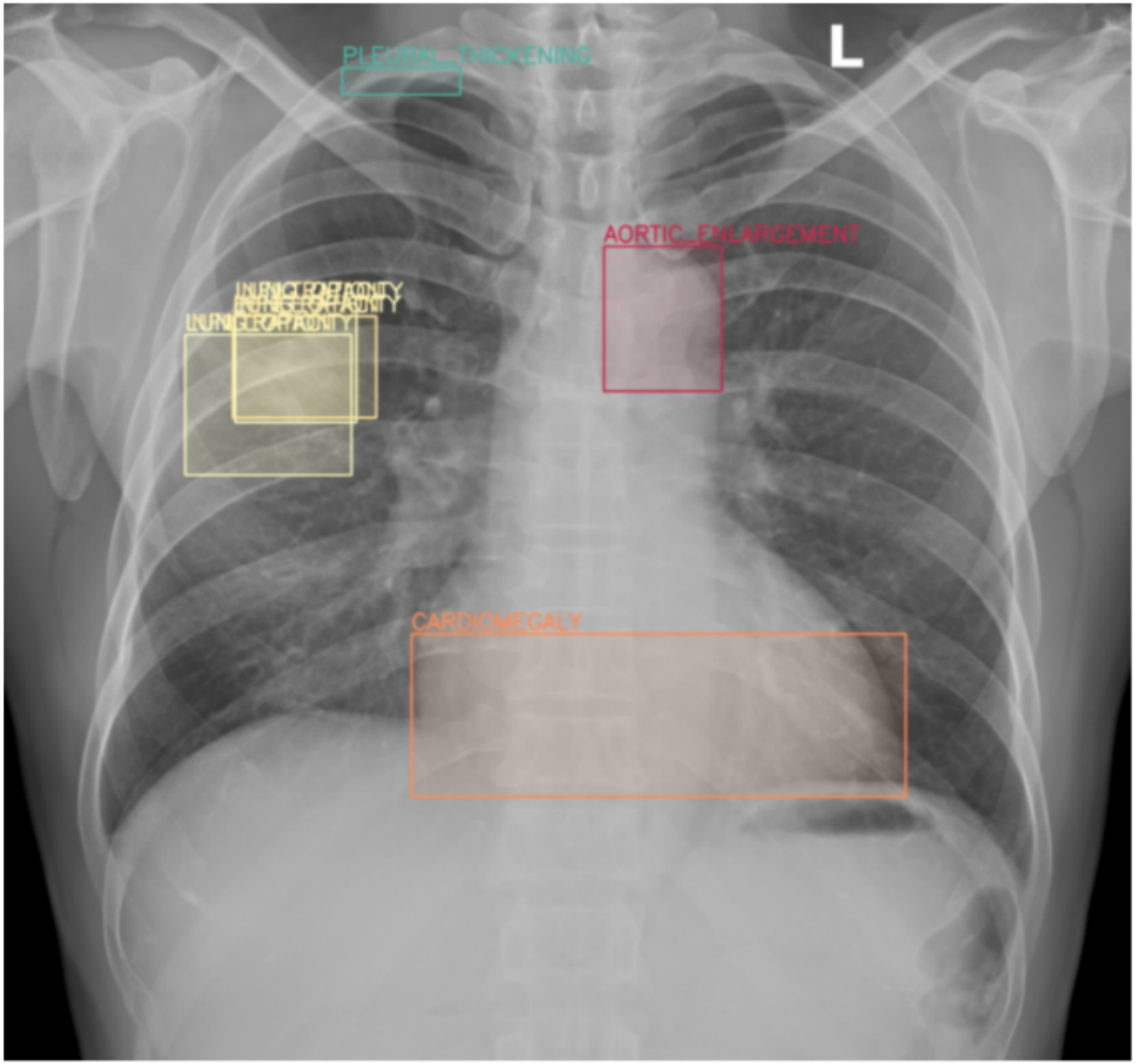


(a)

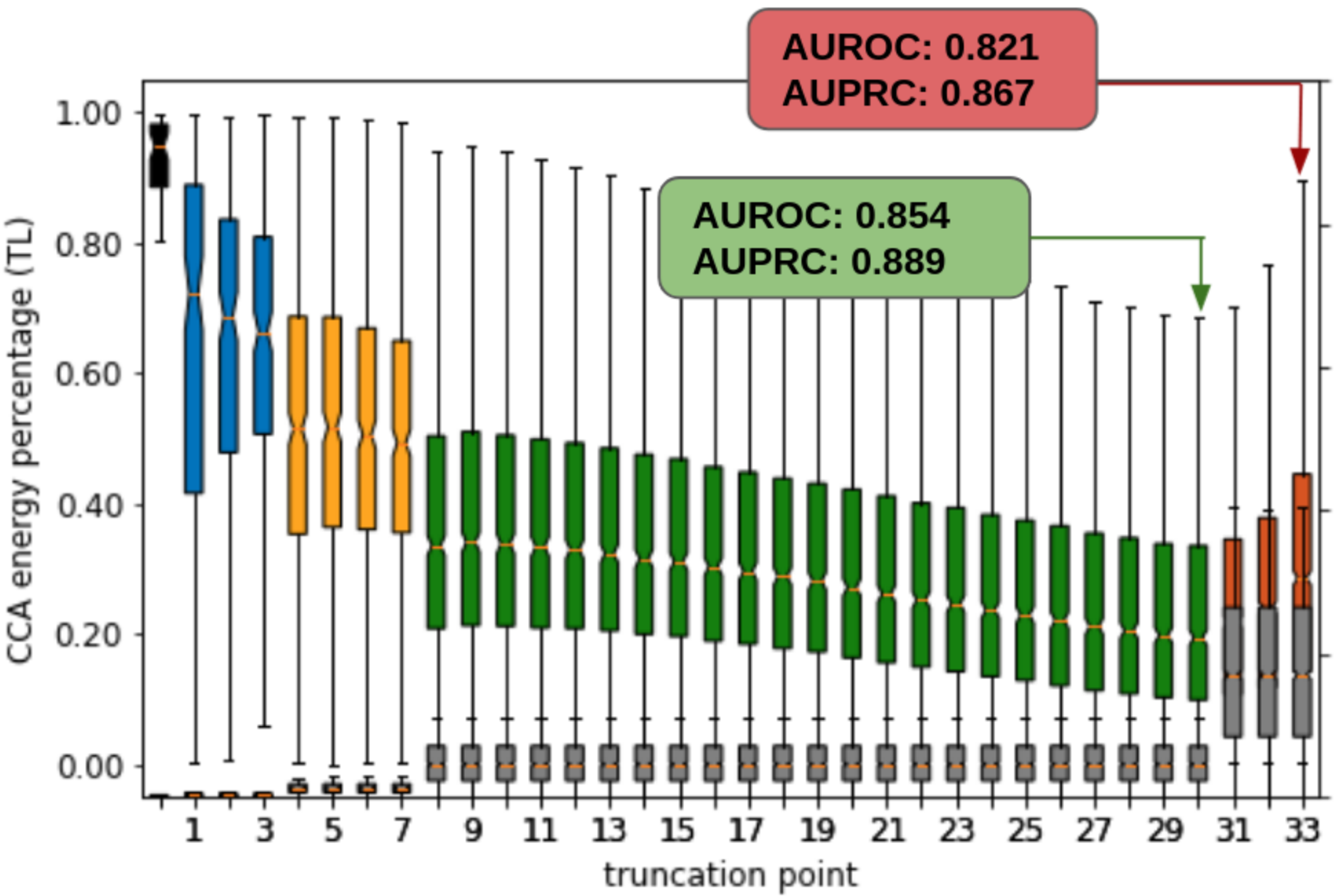
(b)

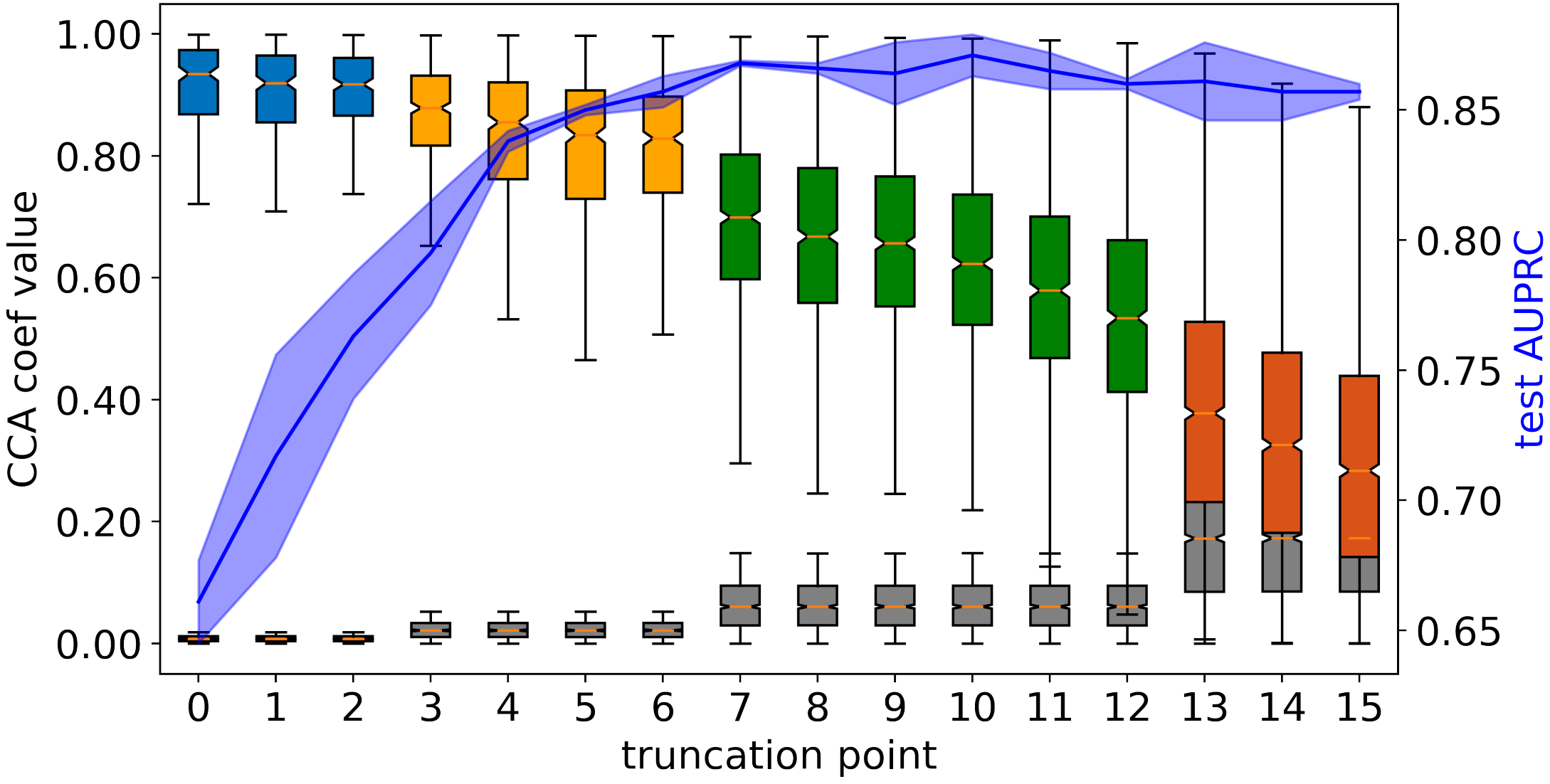


(c)



(d)


AUROC: 0.821
AUPRC: 0.867
AUROC: 0.854
AUPRC: 0.889
CCA energy percentage (TL)
1.00
0.80
0.60
0.40
0.20
0.00
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
truncation point


1.00
0.80
0.60
0.40
0.20
0.00
CCA coef value
0.85
0.80
0.75
0.70
0.65
test AUPRC
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
truncation point