Automatic mpMRI-based Prostate Lesions Assessment with Unsupervised Domain Adaptation

Jing Dai[1], Xiaomei Wang[1], Yingqi Li[1], Zhiyu Liu[1], Yui Lun Ng[1], Jiaren Xiao[1], James Lam[1], Qi Dou[1], Varut Vardhanabhuti[1], and Ka-Wai Kwok[1]

[1]Affiliation not available

August 18, 2022

Abstract

Multiparametric magnetic resonance imaging (mpMRI) has emerged as a valuable diagnostic tool in prostate lesion assessment, for which many reports conclude the advantageous use of convolutional neural networks (CNNs) in prostate lesion detection and classification (PLDC). However, the network training inevitably involves prostate magnetic resonance (MR) images from multiple sites/cohorts. There always exists variation in scanning protocol among the cohorts, inducing significant changes in data distribution between source and target domains. This challenge has greatly limited clinical adoption on a large scale. Recent domain adaptation (DA) models can alleviate the inherent cross-site heterogeneity. Some could leverage cross-domain knowledge transfer using whole-slide images, without prior knowledge of lesion regions. In this paper, we propose a *coarse mask-guided deep domain adaptation network* (CMD²A-Net) in order to develop a fully automated framework for PLDC using multi-cohort images. No category or mask label is required from the target domain. A coarse segmentation module is trained to cover the possible lesion-related regions, so that attention maps can be generated to dedicate the local feature extraction of lesions within those regions. As a result, the features of both prostate lesion and region can be fused to align the robust features between the source and target domains. Experiments have been performed on 512 mpMRI sets from datasets of PROSTATEx (with 330 sets) and two cohorts, A (with 74 sets) and B (with 108 sets). Using the ensemble learning, our CMD²A-Net accomplishes an AUC of 0.921 in cohort A and 0.913 in cohort B, demonstrating its transferability from a large-scale public dataset PROSTATEx to our small-scale target domains. Our results and ablation study also support the CMD²A-Net's effectiveness in lesion classification between benign or malignant, compared to the state-of-the-art models

Corresponding author(s) Email: *kwokkw@hku.hk (K.W. Kwok), varv@hku.hk (V. Vardhanabhuti)*

Results and Discussion

Datasets

Table 1. Characteristics of the five MRI datasets for prostate segmentation and PLDC.

Datasets	Total Cases	Positive Cases	Negative Cases	MRI Scanner	Diffusion *b-value* (s/mm^2)	In
I2CVB	646	N/A	N/A	Siemens, and General Electric	N/A	0.
P-x	330	76	254	Siemens Trio and Skyra	50, 400, and 800	0.
LC-A	74	51	23	Philips Achieva	0 and 1,400	0.
LC-B	108	14	94	Philips Achieva	1,000 and 1,400	0.
LC-C	29	11	18	Siemens Skyra	N/A	0.

Remarks: *PSeg – Prostate segmentation; #CHA – Cross-domain heterogeneity analysis.

Five datasets are utilized in this study, i.e. Initiative for Collaborative Computer Vision Benchmarking (I2CVB) [1], PROSTATEx[2] (P-x), and three datasets from Hong Kong hospital local cohorts, LC-A, LC-B, and LC-C. Note that, LC-A and LC-B were acquired from the same MR[3] imaging center. **Table 1** shows characteristics of these five datasets. Note that I2CVB is already available online, which has been widely investigated for prostate zone segmentation [4]. It contains 646 T2 images acquired from 36 patients. Fifteen patients were scanned by 3.0-T Siemens scanners and 21 patients by 1.5-T General Electric scanners. Given the segmentation labels on the prostate, central gland, peripheral zone, and lesion, only image slices covering the prostate are selected as our samples. A Mask R-CNN model is employed for prostate segmentation using this dataset. P-x, LC-A, LC-B, and LC-C are mpMRI-based datasets marked with point labels. The four datasets share the same set of category labels (i.e. csPCa and non-csPCa). Dataset, P-x, LC-A, and LC-B, are utilized to evaluate the PLDC performance of our CMD²A-Net, including 330 cases from P-x, 74 cases from LC-A, and 108 cases from LC-B. To avoid "overfit" caused by LC-C (29 cases) with its small size, it is only used for cross-site heterogeneity analysis.

Analysis of Cross-site Heterogeneity

We first evaluated prostate segmentation performance using mean intersection over union (IoU), in order to ensure that the prostate regions can be predicted accurately. The IoU indicates the intersection between the predicted prostate contour and the ground truth mask label, which was measured on the test split of I2CVB. The mean IoU of prostate region, central gland, and peripheral zone are 0.843, 0.781, and 0.516, respectively. These results are comparable with the work of Alkadi, Taher, El-Baz and Werghi [4] which attained an IoU of 0.673 and 0.599 for the central gland and peripheral zone, respectively. It implies that the training set containing MR[5] images from 36 patients is already sufficient for accurate prostate segmentation. Additionally, the segmentation results are found to be promising on the image obtained from either 1.5-T or 3.0-T MRI[6] machine, indicating that the IoU measuring is not sensitive to the scanner types (see details in **Supplementary Figure 1**).

Table 2. Comparisons of AUC using separate and joint learning approaches.

Datasets	T2	T2	T2	T2	ADC	ADC	ADC	ADC	hDWI	hDWI	hDWI	hDWI
	P-x	LC-A	LC-B	LC-C	P-x	LC-A	LC-B	LC-C	P-x	LC-A	LC-B	LC-C
P-x only	**0.91**	0.35	0.55	0.65	**0.67**	0.53	0.42	0.51	**0.81**	0.41	0.68	0.50
LC-A only	N/A	**0.61**	0.55	0.66	N/A	**0.69**	0.38	0.53	N/A	**0.70**	0.57	0.49
LC-B only	N/A	0.39	**0.61**	0.67	N/A	0.52	**0.61**	0.48	N/A	0.47	**0.88**	0.59
Joint P-x, LC-A	0.89	0.67	N/A	N/A	0.73	0.54	N/A	N/A	0.73	0.54	N/A	N/A
Joint P-x, LC-B	0.88	N/A	0.59	N/A	0.66	N/A	0.55	N/A	0.76	N/A	0.87	N/A
Joint LC-A, LC-B	N/A	0.63	0.53	N/A	N/A	0.74	0.61	N/A	N/A	0.72	0.91	N/A

Then, we analyzed cross-site heterogeneity on our multi-cohort datasets (P-x, LC-A, LC-B, and LC-C). We aim to verify whether the prior MR[7] image intensity normalization (e.g., Liu, et al. [8] is effective to reduce domain shift, when domain knowledge is not considered. Coarse Mask-guided Network (i.e. CM-Net, in **Figure 4**) was utilized for cross-site heterogeneity analysis. Here, training a model on an individual dataset is defined as "separate learning approach", while training a model using a combined dataset from multi-cohort samples is defined as "joint learning approach". As shown in **Table 2,** we trained CM-Net using the individual and combined datasets from P-x, LC-A, and LC-B. The three separate models were individually trained in these three domains. They are set as the baselines for comparisons with the joint models. During the testing phase, each separate model was tested on the four datasets. LC-C only acted as the hold-out testing set for domain shift analysis, as its small size (only 29) would cause overfitting in training, and biased prediction in testing. Note that, owing to the limited sample size of local cohorts (74 and 108 cases on LC-A and LC-B, respectively), separate models of LC-A and LC-B were pre-trained on the large-scale dataset P-x (330 cases), and then fine-tuned on the corresponding domain. Such a transfer learning[9] strategy would reduce overfitting caused by data scarcity. A common preprocessing method, *scaled*, was employed to normalize the image intensities within [0,1].

Table 3. Comparisons of AUC using six image preprocessing methods.

Preprocessing Methods	T2	T2	ADC	ADC	hDWI	hDWI
	P-x	LC-A	P-x	LC-A	P-x	LC-A
Scaled	0.89	0.67	**0.73**	0.54	0.73	0.54
Whitening	0.87	0.73	0.65	**0.72**	0.56	0.54
Scaled + BFC	0.90	0.71	0.67	0.65	0.76	**0.65**
Whitening + BFC	**0.91**	**0.80**	0.68	0.68	0.73	0.55
Scaled + NF	0.89	0.75	0.66	0.61	**0.80**	0.56
Whitening + NF	0.84	0.72	0.64	0.66	0.79	0.57

The results of separate models from P-x, LC-A, and LC-B are shown in **Table 2**. For the three sequences (i.e., T2, ADC, and hDWI), the AUCs of three separate models are relatively high when tested within their domains, but the AUCs sharply drop when directly tested in the unseen domains. Such results show the sensible cross-domain discrepancy (i.e. domain shift) among the four datasets. Note that, in terms of the T2 sequence, separate models of LC-A and LC-B accomplish the highest testing AUCs (0.66 and 0.67) in the unseen domain, LC-C, just marginally higher than the ones (0.61) within their corresponding domains. A potential reason for the biased predictions is the deficiency of testing samples (i.e. 29) on LC-C. When it comes to the joint models in the table, they cannot bring remarkable improvements in each sequence compared with the separate models, instead, even may lead to performance degradation due to cross-site heterogeneity.

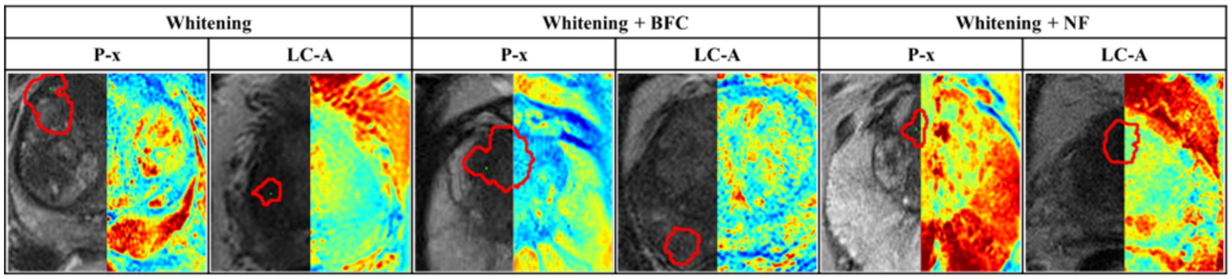


Figure 1. Image preprocessing examples (from P-x and LC-A) in quantitative analysis on inter-site heterogeneity. Among the six methods in **Table 3**, whitening, whitening + BFC, and whitening + NF act as representatives. Coarse lesion region is contoured (in red) on the randomly selected pre-cropped T2 images. Prior to the preprocessing (left half), the heterogeneity of intensity distribution can be observed obviously in the original samples, while the distributions are harmonized after the preprocessing (right half). All the jet color maps are same scale.

With severe discrepancies among our datasets, we intend to validate whether the rigorous MR[10] image preprocessing methods can contribute to the joint models' classification performance. Similar to scaled, whitening is another common preprocessing method, capable of normalizing the pixel values with a mean of zero and a variance of unit. Taking the combined dataset, P-x and LC-A, as a representative for evaluation. In **Table 3**, scaled, whitening, and their combined function with bias field correction (BFC) or noise filtering (NF), 6 preprocessing methods in total (see details in **Supplementary Figure 3**) , were adopted as in [8]. The joint models using scaled and whitening acted as the two baselines for comparisons with the rigorous MR image preprocessing methods (i.e. BFC and NF). **Figure 1** depicts the image preprocessing examples of three methods (i.e. whitening, whitening + BFC, and whitening + NF). The left and right halves of each sample represent before and after preprocessing, respectively. Before preprocessing, we can observe noticeable intensity distribution discrepancies on the samples. The samples from LC-A are characterized by larger numbers of low-intensity grayscale pixels as compared with the images of P-x. Subsequently, the jet color maps were employed to highlight the intensity distribution between domains after preprocessing. All the color maps share the same intensity color scale. Similar intensity distributions can be found among the samples after preprocessing, demonstrating the effectiveness of the methods in image distribution harmonization.

In **Table 3**, for the T2 sequence, BFC with either scaled or whitening outperforms the baselines. Besides, BFC with whitening achieves best AUCs of 0.91 and 0.80 on P-x and LC-A, respectively. However, these findings are not consistent with the results in ADC and hDWI. In terms of ADC, the models preprocessed with BFC or NF underperform the baselines. Instead, the baseline models receive the highest AUCs, where scaled alone and whitening alone accomplish 0.73 and 0.72 on P-x and LC-A, respectively. When it comes to the sequence of hDWI, either BFC or NF attributes limited improvement over the baselines. On P-x, the AUC increases marginally from 0.73 (scaled only) to 0.80 (scaled with NF); on LC-A, only an AUC of 0.65 is achieved using scaled with BFC. The above results of the three sequences show that these pre-processing approaches could improve CM-Net's classification performance when combing our two datasets. However, none of the methods is capable of boosting the joint models' generalization considerably, as compared with the separate models of P-x and LC-A (in **Table 2**). This indicates that the preprocessing methods are probably insufficient to solve domain shift fundamentally. A possible reason is that the severe discrepancies do not come from the inter-site discrepancies (in **Table 1**), rather than the intensity distribution of the heterogeneous mpMRI[11] sequences only (see details in Supplementary **Figure 2**).

Cross-domain Malignancy Classification and Lesion Detection

We emphasize the importance of knowledge transfer from a large-scale publicly dataset to a small-scale target domain. The malignancy estimation performance of CMD**²**A-Net (the architecture is shown in **Figure 4** and described in detail in the Methods section) is evaluated. Dataset, P-x, is only regarded as the source domain. Either LC-A or LC-B is also set as the source domain for knowledge transfer between local cohorts. The scaled method was employed for image preprocessing. In general, available types of MR[11-12] sequences may vary in healthcare institutions. Thus, we employed ensemble learning[13] to handle multiple sequences, allowing the use of single and multiple sequence(s) in our framework. Three common metrics were adopted for classification performance evaluation, i.e. AUC, sensitivity (SEN), and specificity (SPE).

Table 4. Malignancy classification results in the target domains in four combinations of source-target domain.

Sequences Combinations	P-x - LC-A	P-x - LC-A	P-x - LC-A	P-x - LC-B	P-x - LC-B	P-x - LC-B	LC-A - LC-
	AUC	SEN	SPE	AUC	SEN	SPE	AUC
T2	0.87	0.79	0.80	0.84	0.75	0.79	0.65
ADC	0.79	0.76	0.66	0.75	0.74	0.73	0.64
hDWI	0.79	0.74	0.74	0.90	0.80	0.80	0.61
T2 + ADC	0.91	0.76	0.86	0.86	0.76	0.75	0.69

Sequences Combinations	P-x - LC-A	P-x - LC-A	P-x - LC-A	P-x - LC-B	P-x - LC-B	P-x - LC-B	LC-A - LC-I
T2 + hDWI	0.92	0.80	0.84	**0.92**	**0.94**	0.68	0.68
ADC + hDWI	0.84	0.79	0.74	0.90	0.83	0.73	0.68
T2 + ADC + hDWI	**0.92**	**0.83**	**0.90**	0.91	0.79	**0.82**	**0.74**

Table 4 illustrates the classification results (i.e. csPCa or non-csPCa). Seven sequence combinations are involved for comparisons. The former and the later domains in the table are denoted as the source and target domains, respectively. We define such pairs of domains/cohorts as DA settings. First, we compare CMD^{2}A-Net with the separate and joint models (in **Table 2**) in terms of AUC. Take the first DA setting (P-x - LC-A) as an example. In the T2 sequence, CMD2A achieves an AUC of 0.87 in the target domain (i.e. LC-A), outperforming both the separate model (AUC: 0.61) and the joint model (AUC: 0.67). Consistent findings can be observed in ADC and hDWI. When it comes to the other three DA settings, CMD2A-Net also demonstrates its advantage in resolving domain shift between two of our datasets. This validates our hypothesis that incorporating prostate lesion information in prior to the DA process can facilitate PCa classification.

Second, we analyze our model's PCa classification performance using a single sequence, i.e. T2, ADC, or hDWI. In most source-target DA settings, T2 is the most effective, while ADC receives the lowest AUC. Sequence, hDWI, shows unstable performance in the four DA settings. For example, it accomplishes the most superior performance (w.r.t. AUC, SEN, and SPE) in "P-x - LC-B", but underperforms T2 and ADC in "LC-A - LC-B". This could be caused by heterogeneous *b-values* among the domains. As shown in **Table 1** , *b-values* of 50, 400, and 800s/mm^2 were employed on P-x, while 0 and 1,400s/mm^2 were used in LC-A, 1,000 and 1,400 s/mm^2 were used in LC-B. Thus, we can conclude that the significant discrepancies in the acquisition parameters would result in the inconsistent performance of hDWI. Note that there had no widely accepted guidelines regarding *b-value* until the release of PI-RADS in 2019, which recommended a minimum value of 1,200s/mm^2.

We also investigate effect of ensemble learning using multiple sequences, which could provide references to choose appropriate sequences for PLDC. In each DA setting, the models using multiple sequences are always more effective than using any single sequence alone. Besides, although ADC or hDWI always leads to the worst classification results, T2 ensembled with one/both of them can explicitly enhance the model's performance. This finding is consistent with the clinical practice of using mpMRI for PCa diagnosis. Sequences ADC and hDWI are usually considered as secondary references by radiologists. It should be noted that the all-sequence-ensembled (i.e. ensemble of T2, ADC, and hDWI) models show significant predictions in most DA settings. Although ensemble of the three sequences could not lead to the best performance in the second DA setting (i.e. P-x - LC-A), the model of the second DA setting still attains a remarkable AUC of 0.91, which is only about 1% smaller than the highest AUC (0.92). It can be concluded that using more sequences would help multi-cohort MRI[14] harmonization, thus boosting the final classification performance. Moreover, with the same target domain (i.e. either LC-A or LC-B), the CMD2A-Net transferred from P-x attains a higher AUC than transferred from a local cohort domain in each sequence combination. This implies more source samples could enhance the model's cross-domain knowledge transferability, thus improving the model's generalization in the target domain. The superior performance also demonstrates CMD2A-Net's capability of transferring the knowledge of a public dataset to our local cohort domains.

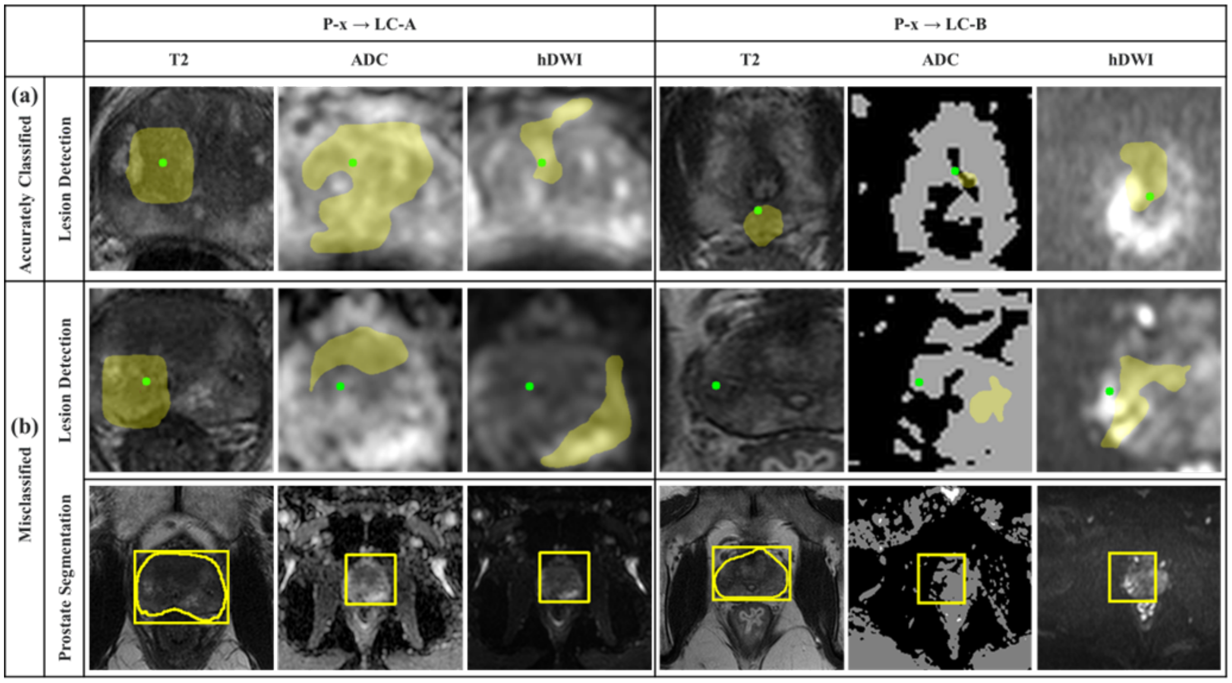


Figure 2. Coarse lesion detection results of (a) accurately classified and (b) misclassified examples in target domains, LC-A and LC-B, relative to the source, P-x. All-sequence-ensembled (T2, ADC, hDWI) approach is employed. In lesion detection results (1st and 2nd rows), The lesions (ground-truth) are pointed in green. The predicted coarse lesion regions are colored in yellow. Promising prediction of lesion region, i.e., containing the ground-truth in all sequences, can yield the higher correctness of classification as in **(a)**. Moreover, under-segmented prostate regions marked with yellow boxes/contours (i.e., the example of LC-B in the 3rd row) would also worse the classification outcome.

Figure 2 shows coarse lesion detection results of the accurately classified and misclassified examples. Two DA settings (i.e. P-x to LC-A, and P-x to LC-B) were selected as representatives for lesion detection evaluation. Results of the all-sequence-ensembled method are selected as representative for analysis. In the correctly classified examples, Coarse lesion contours could encircle the lesion ground-truth point in all sequences (as shown in **Figure 2a**). However, in the unclassified examples, the coarse lesion position could not be precisely detected in most sequences as shown in the third row. In the example of LC-A, the lesion on T2 is correctly detected, but the lesion contours on ADC and hDWI maps are falsely identified. The possible reason is that the coarse lesion masks applied as the training ground truth could not depict the actual lesion contours accurately. Therefore, we can observe that accurate detection on ADC and hDWI also play a role in enhancing the ensembled classification, although lesion detection generally heavily relies on T2 images. In the future, robust weak label processing methods (e.g., deep extreme level set evolution method [15]) are expected to be employed. For the example from LC-B, under-segmentation of the prostate region can be found on the T2 image, which could lead to failure lesion detection. As the prostate regions on ADC and hDWI were transformed using T2, under/over-segmentation of the prostate gland on T2 would deteriorate the lesion detection in the other two sequences. Despite the inaccurate lesion detection on ADC and hDWI, it should be noted that the models with multi-sequences input still outperform the models using T2 alone in lesion classification, accrediting to the re-use of prostate features from ADC and hDWI.

Comparisons with the State-of-the-art Methods

We compared our model with three state-of-the-art models using AUC, i.e. Resnet50 [16], DANN [17], and Deep Coral [18]. Dataset, P-x, was used as the source domain. Our local cohort datasets, LC-A and LC-B, acted as the target domains. The individual (i.e. T2, ADC, and hDWI) and the ensembled (i.e. T2 + ADC + hDWI) sequences were involved. The other ensembled sequences, T2 + ADC, T2 + hDWI, and ADC + hDWI were not involved here due to their inferior performance as discussed in Section 4.2. Detailed comparison results are summarized in **Table 5**.

Table 5. AUC comparisons on malignancy classification (i.e. csPCa or non-csPCa) with the three existing models.

Methods	P-x - LC-A	P-x - LC-A	P-x - LC-A	P-x - LC-A	P-x - LC-B	P-x - LC-B	P-x - LC-B	P-x
	T2	ADC	hDWI	Ensemble	T2	ADC	hDWI	En
Resnet50	0.65	0.70	0.55	0.70	0.66	0.71	0.68	0.7
DANN	0.70	0.72	0.66	0.79	0.68	0.66	0.75	0.8
Deep Coral	0.71	0.75	0.67	0.75	0.74	0.73	0.80	0.8
CMD^{2}A-Net (Ours)	**0.87**	**0.79**	**0.79**	**0.92**	**0.84**	**0.75**	**0.90**	**0.9**

Resnet50 is a common classification model. It was pre-trained from the source domain, then tuned and tested in the target domain, therefore, no DA was utilized. In **Table 5** , Resnet50 underperforms all other methods in all sequences. The possible reason may be the weak cross-domain knowledge transferability of the fine-tune strategy. This shows the advantage of domain adaptation methods over the fine-tune strategy. Another model, DANN, which is GAN-based and has been widely employed in lesion assessment. It can extract low-level features from the entire image. Moreover, Deep Coral was also introduced, which can leverage domain knowledge transfer by aligning the second-order statistics. Similar to DANN, it also adopts a common encoder for feature extraction from the input of a whole image slice. Differently, our model could fuse both lesion features and prostate features for effective DA, instead of extracting the prostate features. We "strengthened" the point labels to be coarse mask labels, such that features, particular lesion features, can be robustly aligned for DA using the mask labels. In **Table 5** , CMD^{2}A-Net outperforms the two UDA models in all the sequences in terms of AUC, indicating the effectiveness of our model in cross-domain feature harmonization and its advantage in prostate lesion classification. It is worth noting that all four models accomplish their highest AUCs using the ensembled sequence. The consistent conclusion can be found in Section Cross-domain Malignancy Classification and Lesion Detection, showing the benefits of the all-sequence-ensembled method again.

Visualization of Sample Distribution and Ablation Study

Apart from AUC, we also intend to visualize the sample distribution of source and target domains, in support to any improved performance of handling domain shift intuitively. Datasets, P-x and LC-A, were adopted to visualize the data distribution before and after the DA. Algorithm, t-SNE [19], was employed to visualize the data distributions of all sequences, i.e. T2, ADC, and hDWI. Fifty mpMRI cases from each dataset were randomly chosen. As shown in **Figure 3a-c**, obvious clustering can be observed before DA in each sequence, indicating severe domain shift between the two domains. After CMD^{2}A-Net training (i.e. DA), domain-invariant features were extracted by the well-trained model. After the DA, each sequence samples from the two cohorts are evenly distributed, proving that CMD^{2}A-Net could assure feature alignment on the heterogenous mpMRI sequences[20].

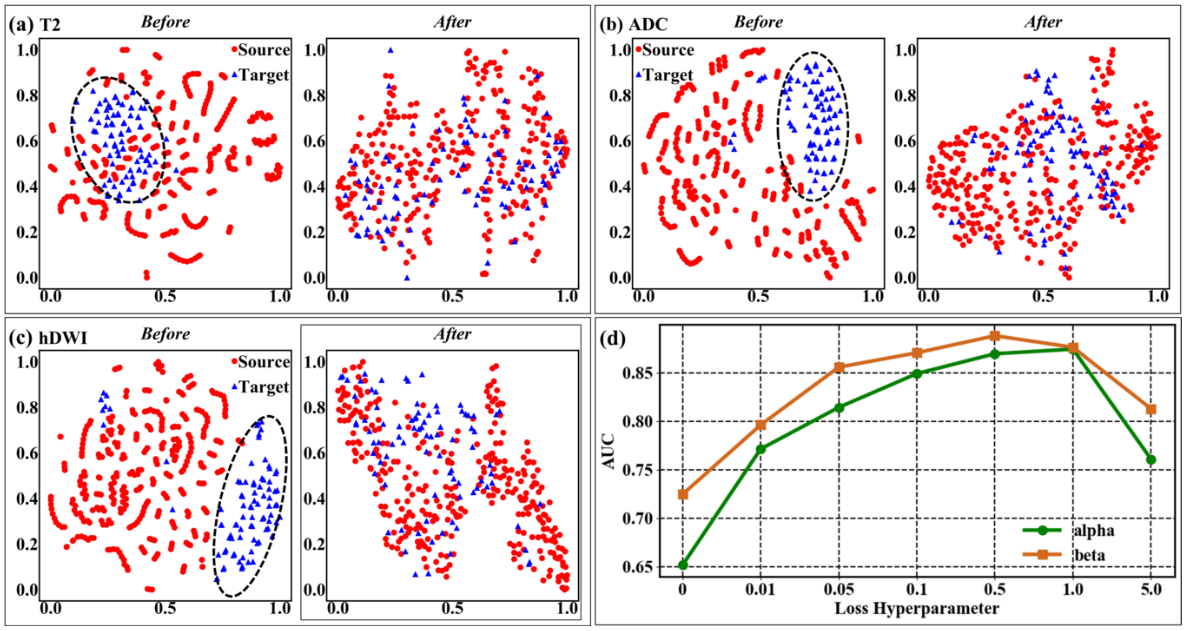


Figure 3. Sample distribution *before* and *after* DA for sequences (a) T2, (b) ADC, and (c) hDWI using t-SNE. Similar change of distributions can be observed in all the sequences. Before DA, sample distributions of the source (red dots) and target (blue triangles) are dispersed in separate clusters, indicating severe their domain shift. The mixed and even distribution after DA demonstrates the effectiveness of CMD^{2}A-Net in feature alignment. **(d)** indicates the impact of hyperparameters (i.e., α and β) in the loss sensitivity analysis.

To carry out the ablation study, we selected two key components, i.e. the coarse segmentation module and the domain transfer module, to analyze their contribution to lesion malignancy classification using T2 images. We compared our CMD^{2}A-Net with its two variants using AUC, i.e. (1) CMD^{2}A-Net *excluding* domain transfer module (i.e. CM-Net, shown in the black dashed box in **Figure 4**) and (2) CMD^{2}A-Net*excluding* the course segmentation modules (D^{2}A-Net). Since the CM-Net does not contain DA modules, it was trained in the source domain, then fine-tuned and tested in the target domain. Datasets, P-x and LC-A, were selected as the source and target domain, respectively. D^{2}A-Net obtains a lower AUC (0.65) compared with CMD^{2}A-Net (0.87). This suggests that the coarse segmentation module is essential for domain-invariant feature extraction between domains. This also supports our hypothesis that the coarse lesion maps would enhance the malignancy classification accuracy. CM-Net obtains an AUC of 0.67, also less than CMD^{2}A-Net. This indicates that the domain transfer module can substantially mitigate domain shift, thus enhancing CMD^{2}A-Net's PCa classification performance.

The loss parameters sensitivity is also analyzed. CMD^{2}A-Net was trained using P-x (source domain) and LC-A (target domain). Hyperparameters α and β (i.e. weighting parameters of the total loss) would influence the model's generalization ability essentially. **Figure 3d** shows the AUC of our model with various values of α and β , both of which ranged in {0, 0.1, 0.05, 0.1, 0.5, 1.0, 5.0}. We can observe that CMD^{2}A-Net demonstrates superior classification performance with α within [0.1, 1.0] and β within [0.05, 1.0]. It should be noted that our model receives the lowest AUC when either α or β is set to 0, showing that the coarse segmentation module and the domain transfer module could enhance cross-domain knowledge transferability positively, thus improving lesion classification accuracy.

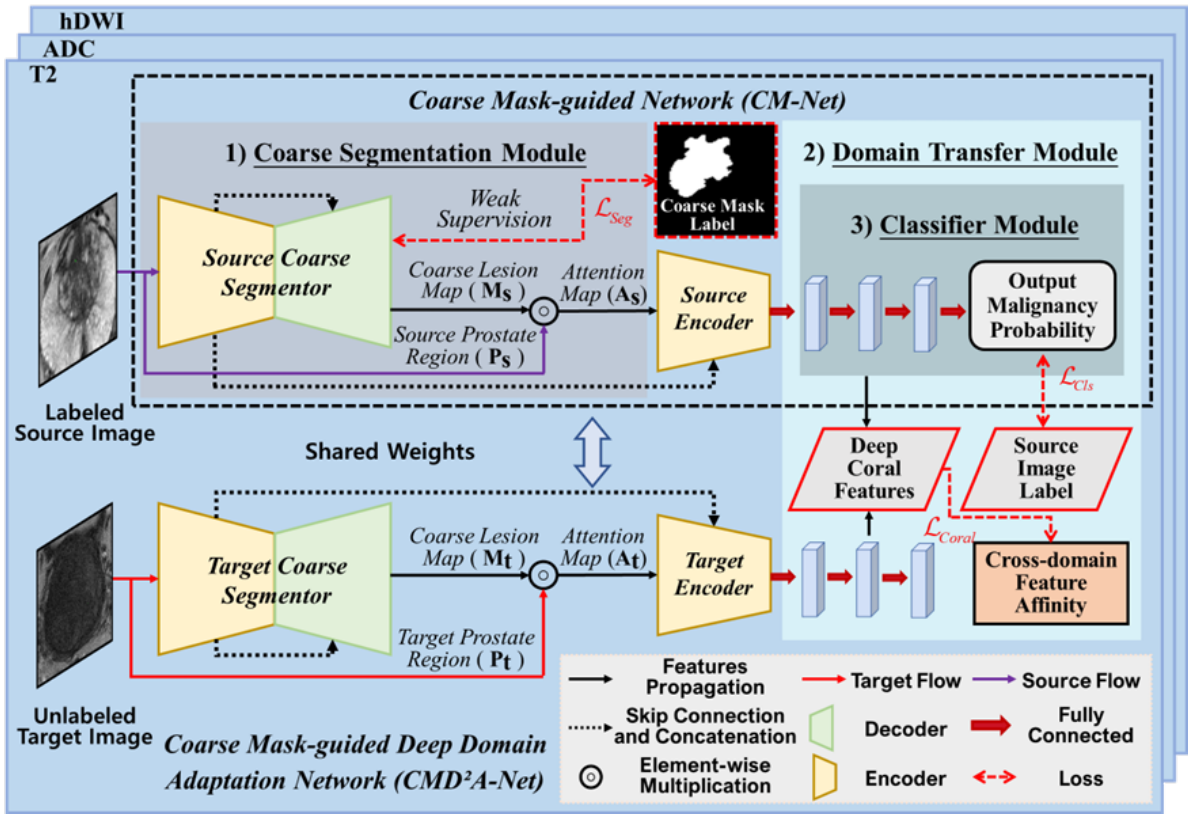
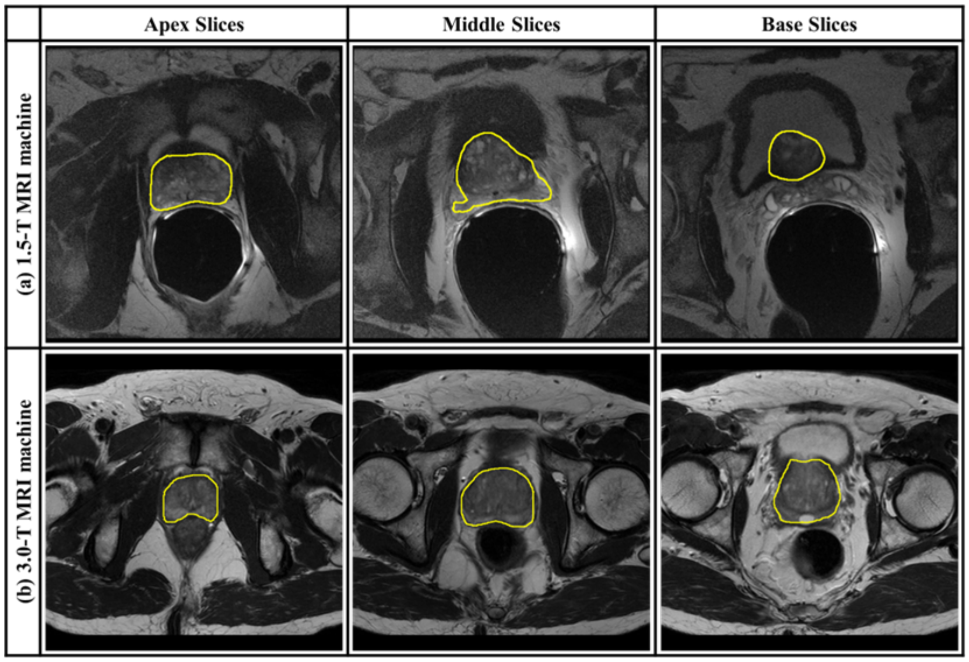


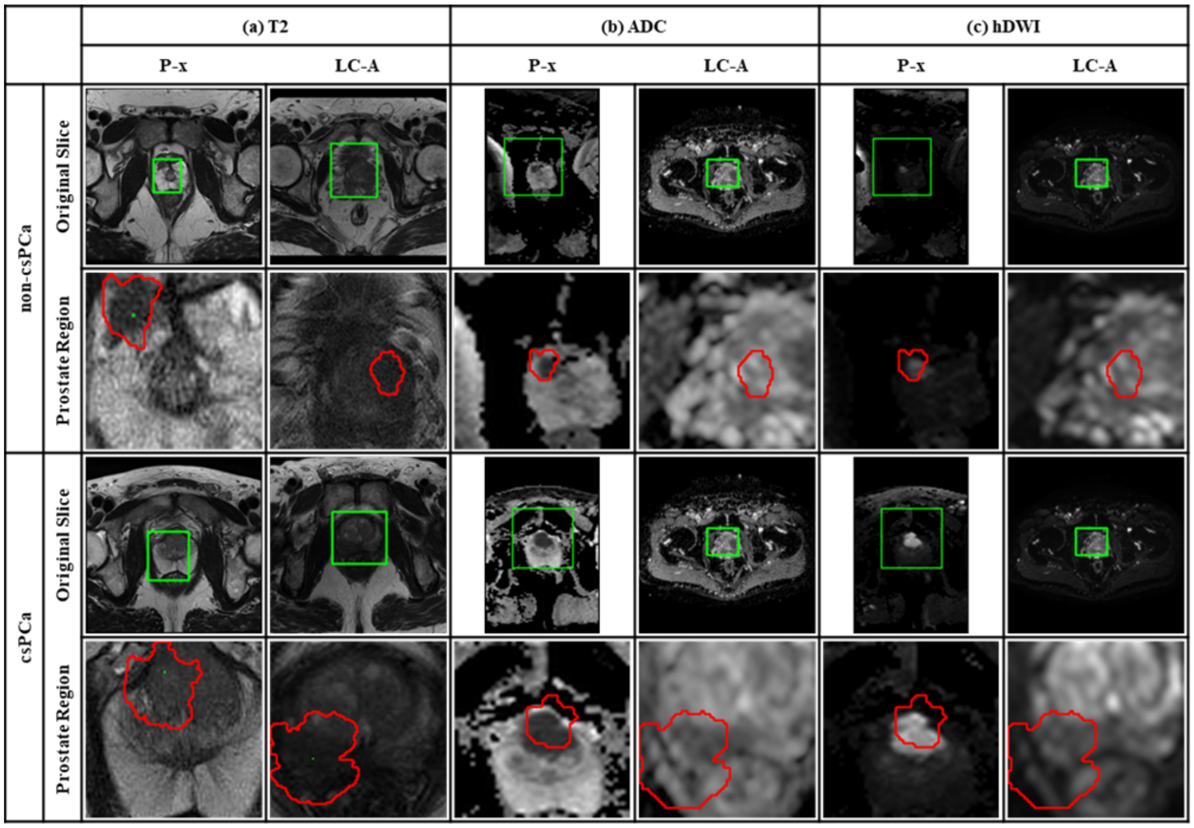
Figure 4. Overview of the proposed CMD²A-Net using T2, ADC, and hDWI image inputs. Each image sequence network is featured with two parallel branches with respect to the source and target domains. Three main modules in the source one: 1) coarse segmentation module for coarse lesion detection and feature alignment enhancement, 2) domain transfer module for knowledge transfer between domains, and 3) classifier module for malignancy classification.

Supplementary Figure 1



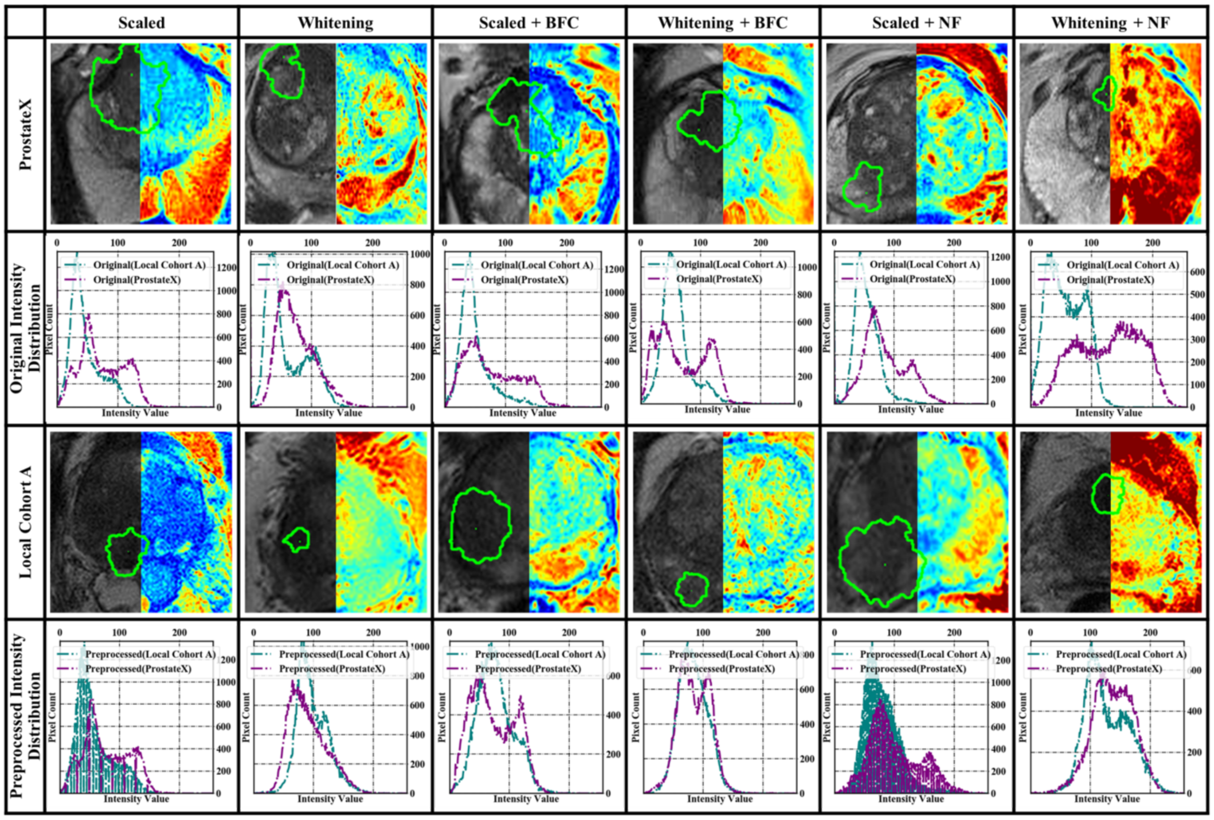
Supplementary Figure 1. Prostate segmentation results of T2 images scanned from (a) 1.5-T MRI scanner and (b) 3.0-T MRI scanner. Prostate region is contoured (in yellow) on the randomly selected apex, middle, and base slices.

Supplementary Figure 2



Supplementary Figure 2. non-csPCa and csPCa mpMRI examples (from P-x and LC-A) of (a) T2, (b) ADC, and (c) hDWI. The prostate gland is contoured with rectangles (in green) on the original slices (1st and 3rd row). The coarse lesion is annotated (in red) on the cropped prostate regions (2nd and 4th row) using the level set method, showing lesion morphological discrepancy of the benign and malignant samples. Apparent inter-site heterogeneity (e.g., FoV, image intensity distribution) of the samples demonstrates domain shift between P-x and LC-A.

Supplementary Figure 3



Supplementary Figure 3. Six preprocessing methods were adopted for inter-site heterogeneity analysis. Six randomly selected T2 images from two cohorts (i.e. ProstateX, Local Cohort A) are preprocessed with the same intensity color scale, whereas the green contour shows the lesion region. The second and fourth rows show the image intensity distribution from corresponding cohorts before and after preprocessing.

Introduction to our open-sourced PLDC system

This open-sourced deep-learning-based model acts as an end-to-end system, input from prostate mpMRI sequences (i.e. T2, ADC, and hDWI), output to prediction results (i.e. prostate segmentation, coarse lesion detection, and malignancy estimation). The system supports multi-format inputs, including DICOM, jpeg, png, and jpg files. It is emphasized that no manual prostate segmentation or annotation is required.

1. To realize automatic PLDC with multi-cohort mpMRIs (i.e., T2, ADC, and hDWI), please download the executable software first from the Github repository. In addition, you should also download the PLDC_software.zip from the Github repository. Unzip the zip file, and put the downloaded executable software in the folder "./PLDC_software/ ".

2. Install required packages for mpMRI-based PLDC testing.

- python=3.6.5
- Keras
- Tensorflow = 1.15
- Opencv-python
- Pydicom
- Numpy
- Pillow
- Scikit-image
- SimpleITK

3. In order to perform PLDC using your local cohort samples, you should train a domain adaptation (DA) model first (see details in the next Section " Prostate lesion assessment using your local cohort mpMRI "). Put all of your well-trained model weights in the folder "./PLDC_software/doc/weights/ ".

4. Begin to test the target mpMRIs from your local cohort using the open-sourced system. Open the executable software. Start your testing via the "Main menu" button, and then click "Start testing". The predicted results will be saved in the folder "./media/output/ ", including prostate segmentation, prostate lesion detection, and lesion malignancy results. You can download the following prostate_exe.mp4 to learn the details if necessary.

Hosted file

`prostate_exe.mp4` available at https://authorea.com/users/499033/articles/579780-automatic-mpmri-based-prostate-lesions-assessment-with-unsupervised-domain-adaptation

Note: 1) Please do not delete any existing files/folders under the folder "./PLDC_software/ ", such as image, and doc folders.

2) For image format input, make sure your input T2 image is a 3-channel image. Only the T2 image is allowed, as ADC and hDWI require metadata of T2 DICOM for prostate region registration.

3) For DICOM format input, T2 is necessary, while ADC and hDWI are optional.

4) We provide several samples (under the folder "./test_samples") from PROSTATEx to demonstrate the software functions, while the local cohort samples are not open-sourced here due to patient privacy and ethical issues. Our weights trained with PROSTATEx (source domain) and our local target dataset are also provided for demonstration. Our weights can be downloaded from the same Github repository. The total size of this software is around 6GB, including the executable software (~699MB) and model weights (~3.8G).

Prostate lesion assessment using your local cohort mpMRI

Prepare your multi-cohort mpMRI (T2, ADC, and hDWI) datasets for model training. mpMRI dicoms are required, as the metadata has to be extracted for prostate region segmentation on ADC and hDWI. The datasets include a source domain dataset (e.g., the public dataset PROSTATEx), and your mpMRI-based local cohort dataset (i.e., target domain). Our models (i.e. prostate segmentation model and CMD^{2}A-Net) are trained with API Keras. You will train and test the models for PLDC in the following steps. We also offer our executable codes and files online available via the same Github repository, so as to allow any work extension or application by others.

Prostate region segmentation

To train the prostate segmentation model using T2 images, you can train your prostate segmentation model with the codes under ./maskrcnn_model. Apart from your local cohort datasets, the public dataset I2CVB is also available online for training. The input shape of the model is set to 512 $\times$ 512 pixels. Adam optimizer is applied. During the training process, the model with the highest dice coefficient score is retained (i.e., weight1.h5). The prostate on T2 images can be segmented in the following code.

```
<# process a raw T2 png, and save the prostate region, rostate with contour in './media/output/'

def predict(image_path):

    MODEL_DIR = './doc/weights/'
    Prostate_WEIGHTS_PATH = "weight1.h5"  # TODO: update this path

    class InferenceConfig(ProstateConfig):
        # Run detection on one image at a time
        GPU_COUNT = 1
        IMAGES_PER_GPU = 1

    config = InferenceConfig()
    config.display()

    DEVICE = "/cpu:0"  # /cpu:0 or /gpu:0 or /gpu:1

    with tf.device(DEVICE):
        model = MaskRCNN(mode="inference", model_dir=MODEL_DIR,
                                   config=config)

    weights_path = MODEL_DIR + Prostate_WEIGHTS_PATH

    model.load_weights(weights_path, by_name=True)

    class_names = ['BG', 'pz', 'cg', 'prostate', 'cap']
    import skimage
    image = skimage.io.imread(image_path)
    if image.ndim != 3:
        image = skimage.color.gray2rgb(image)
    # If has an alpha channel, remove it for consistency
    if image.shape[-1] == 4:
        image = image[..., :3]

    h,w=image.shape[0],image.shape[1]
    image, window, scale, padding, crop = resize_image(
        image,
        min_dim=config.IMAGE_MIN_DIM,
        min_scale=config.IMAGE_MIN_SCALE,
        max_dim=config.IMAGE_MAX_DIM,
        mode=config.IMAGE_RESIZE_MODE)

    results = model.detect([image], verbose=1)

    # Display results

    r = results[0]
    index_pz = None
```

```
index_cg = None
index_prostate = None
index_cap = None
if len(results[0]['rois']) > 1:
    for i in range(len(results[0]['class_ids'])):
        if r['class_ids'][i] == 1:
            if index_pz == None:
                index_pz = i
            else:
                continue

        if r['class_ids'][i] == 2:
            if index_cg == None:
                index_cg = i
            else:
                continue

        if r['class_ids'][i] == 3:
            if index_prostate == None:
                index_prostate = i
            else:
                continue

        if r['class_ids'][i] == 4:
            if index_cap == None:
                index_cap = i
            else:
                continue

    list_index = []
    for index in [index_pz, index_cg, index_prostate, index_cap]:
        if index != None:
            list_index.append(index)

    if len(list_index) > 1:
        rois = results[0]['rois'][list_index[0]].reshape(1, -1)
        masks = results[0]['masks'][:, :, list_index[0]].reshape(512, 512, 1)
        class_ids = results[0]['class_ids'][list_index[0]].reshape(1, -1)
        scores = results[0]['scores'][list_index[0]].reshape(1, -1)

        for i in range(len(list_index) - 1):
            rois = np.concatenate((rois, results[0]['rois'][list_index[i + 1]].reshape(1, -1)), axis
                -1, 4)
            masks = np.concatenate((masks, results[0]['masks'][:, :, list_index[i + 1]].reshape(512
                                   axis=2).reshape(512, 512, -1)
            class_ids = np.concatenate((class_ids, results[0]['class_ids'][list_index[i + 1]].resha
                                       axis=0).reshape(-1, 1)
            scores = np.concatenate((scores, results[0]['scores'][list_index[i + 1]].reshape(1, -1)
                                    axis=0).reshape(-1, 1)

        r['rois'] = rois
        r['masks'] = masks
        r['class_ids'] = class_ids.reshape(-1)
        r['scores'] = scores.reshape(-1)

    else:
        r['rois'] = results[0]['rois'][0].reshape(-1, 4)
        r['masks'] = results[0]['masks'][:, :, 0].reshape(512, 512, 1)
        r['class_ids'] = results[0]['class_ids'][0].reshape(-1)
        r['scores'] = results[0]['scores'][0].reshape(-1)

x1, y1, x2, y2 = 0, 0, 0, 0
save_path = './media/output/'

if len(r['rois']) != 0:

    for object in range(len(r['rois'])):
        roi = r['rois'][object]
        mask = r['masks'][:, :, object]
        mask = mask + 0
        mask = mask * 255
        mask = np.array(mask, dtype='int8')

        class_id = r['class_ids'][object]

        from PIL import Image

        if class_id==3:
            Image.fromarray(mask, mode='L').resize((w, h)).save(
                save_path + str(class_names[class_id])+'.png')

            mask_image=Image.open(save_path + str(class_names[class_id])+'.png')
            mask=np.array(mask_image)
            raw_image=Image.open(image_path).convert('L')

            contour = measure.find_contours(mask, 100)

            image_with_contour = cv2.cvtColor(np.array(raw_image), cv2.COLOR_GRAY2BGR)
            color = (0, 255, 255)

            prostate_contour = []

            ratio = w / 512

            #cv2.circle(image_with_contour,(283,287),20,(0,255,255),thickness=-1)
            #cv2.circle(image,(int(283/ratio),int(287/ratio)),20,(0,255,255),thickness=-1)

            for c in contour[0]:
                c = np.around(c).astype(np.int)
                prostate_contour.append(np.array([c[1], c[0]]))

            cv2.drawContours(image_with_contour, [np.array(prostate_contour)], -1, color, thickness
            raw_image.save(save_path +’raw.png’)

            cv2.imwrite(save_path + ’contour.png’, image_with_contour)
            prostate_roi=image[roi[0]:roi[2],roi[1]:roi[3]]
            #prostate_roi=image_with_contour[roi[0]:roi[2],roi[1]:roi[3]]

            cv2.imwrite(save_path+’prostate_roi.png’,prostate_roi)

            ratio = w / 512
            x1 = int(str(roi[0])) * ratio
            y1 = int(str(roi[1])) * ratio
            x2 = int(str(roi[2])) * ratio
            y2 = int(str(roi[3])) * ratio

del model
K.clear_session()
gc.collect()
# print(’done!’)
return x1, y1, x2, y2>
```

Data preprocessing on multi-cohort mpMRI

With the well-segmented prostate on T2 images, the prostate regions on all the ADC and hDWI can be pre-cropped in the following codes.

```
<
# preprocess a T2 dicom to raw png, and saved in "./media/convert_img/T2"
def preprocessing(file_path,file_name):
    dicom = pydicom.read_file(file_path)
    try:
        pixel = dicom.pixel_array
    except:
        image = sitk.ReadImage(file_path)
        image_array = sitk.GetArrayFromImage(image)
        pixel = np.int16(image_array)

    # window_center = dicom.get(’WindowCenter’)
    # window_width = dicom.get(’WindowWidth’)
    #
    # pixel[pixel > window_center + window_width / 2] = window_center + window_width / 2
    # pixel[pixel < window_center - window_width / 2] = window_center - window_width / 2

    #pixel=np.array(pixel, dtype=np.uint8)
    # pixel=np.array(pixel,dtype=np.float32)

    pixel = 255 * ((pixel - np.amin(pixel)) / (np.amax(pixel) - np.amin(pixel)))

    pixel = np.array(pixel, dtype=np.uint8)
    if len(pixel.shape)==3:
        pixel=pixel[0]
    image = Image.fromarray(pixel).convert('L')
    save_path = os.path.join("./media/convert_img/T2", file_name.replace('dcm','png'))
    image.save(os.path.join(save_path))
    return save_path


# process a raw ADC dicom and save a ADC png in './media/output/'
def read_adc_prostate(ADC_path):
    ADC_dicom = pydicom.dcmread(ADC_path)
    try:
        ADC_array = ADC_dicom.pixel_array
    except:
        image = sitk.ReadImage(ADC_path)
        image_array = sitk.GetArrayFromImage(image)
        ADC_array = np.int16(image_array)

    ADC_array = 255 * ((ADC_array - np.amin(ADC_array)) / (np.amax(ADC_array) - np.amin(ADC_array)))

    if len(ADC_array.shape)==3:
        ADC_array=ADC_array[0]

    save_path = './media/output/'
    Image.fromarray(ADC_array).convert('L').save(save_path + 'adc.png')
    return

# process a raw BVAL dicom and save a BVAL png in './media/output/'
def read_bval_prostate(BVAL_path):
    BVAL_dicom = pydicom.dcmread(BVAL_path)
    try:
        BVAL_array = BVAL_dicom.pixel_array
    except:
        image = sitk.ReadImage(BVAL_path)
        image_array = sitk.GetArrayFromImage(image)
        BVAL_array = np.int16(image_array)

    BVAL_array = 255 * ((BVAL_array - np.amin(BVAL_array)) / (np.amax(BVAL_array) - np.amin(BVAL_array)

    if len(BVAL_array.shape)==3:
        BVAL_array=BVAL_array[0]

    save_path = './media/output/'
    Image.fromarray(BVAL_array).convert('L').save(save_path + 'bval.png')
    return

# process a ADC png, and save its contour, prostate region in './media/output/'
def adc_transform(ADC_path,T2_path,x1_T2,y1_T2,x2_T2,y2_T2):

    y1_inraw = int(y1_T2)
    x1_inraw = int(x1_T2)
    y2_inraw = int(y2_T2)
    x2_inraw = int(x2_T2)
```

```
T2_dicom = pydicom.dcmread(T2_path)

ImageOrientationPatient = T2_dicom.ImageOrientationPatient
ImagePositionPatient = T2_dicom.ImagePositionPatient
PixelSpacing = T2_dicom.PixelSpacing

a = np.array([[ImageOrientationPatient[0] * PixelSpacing[0],
               ImageOrientationPatient[3] * PixelSpacing[1],
               ImagePositionPatient[0]],
              [ImageOrientationPatient[1] * PixelSpacing[0],
               ImageOrientationPatient[4] * PixelSpacing[1],
               ImagePositionPatient[1]],
              [ImageOrientationPatient[2] * PixelSpacing[0],
               ImageOrientationPatient[5] * PixelSpacing[1],
               ImagePositionPatient[2]]
              ])

threeD_x1y1 = np.dot(a, (np.array([x1_inraw, y1_inraw, 1])))
threeD_x1y2 = np.dot(a, (np.array([x1_inraw, y2_inraw, 1])))
threeD_x2y1 = np.dot(a, (np.array([x2_inraw, y1_inraw, 1])))
threeD_x2y2 = np.dot(a, (np.array([x2_inraw, y2_inraw, 1])))

ADC_dicom = pydicom.dcmread(ADC_path)

ImageOrientationPatient = ADC_dicom.ImageOrientationPatient
ImagePositionPatient = ADC_dicom.ImagePositionPatient
PixelSpacing = ADC_dicom.PixelSpacing

a_adc = np.array([[ImageOrientationPatient[0] * PixelSpacing[0],
                   ImageOrientationPatient[3] * PixelSpacing[1],
                   ImagePositionPatient[0]],
                  [ImageOrientationPatient[1] * PixelSpacing[0],
                   ImageOrientationPatient[4] * PixelSpacing[1],
                   ImagePositionPatient[1]],
                  [ImageOrientationPatient[2] * PixelSpacing[0],
                   ImageOrientationPatient[5] * PixelSpacing[1],
                   ImagePositionPatient[2]]
                  ])

result_x1y1 = np.linalg.solve(a_adc, threeD_x1y1)
result_x1y1_x = result_x1y1[0]
result_x1y1_y = result_x1y1[1]

result_x1y2 = np.linalg.solve(a_adc, threeD_x1y2)
result_x1y2_x = result_x1y2[0]
result_x1y2_y = result_x1y2[1]

result_x2y1 = np.linalg.solve(a_adc, threeD_x2y1)
result_x2y1_x = result_x2y1[0]
result_x2y1_y = result_x2y1[1]
```

```
    result_x2y2 = np.linalg.solve(a_adc, threeD_x2y2)
    result_x2y2_x = result_x2y2[0]
    result_x2y2_y = result_x2y2[1]

    x1_raw_ADC = int(min(result_x1y1_x, result_x1y2_x, result_x2y1_x, result_x2y2_x))
    x2_raw_ADC = int(max(result_x1y1_x, result_x1y2_x, result_x2y1_x, result_x2y2_x))
    y1_raw_ADC = int(min(result_x1y1_y, result_x1y2_y, result_x2y1_y, result_x2y2_y))
    y2_raw_ADC = int(max(result_x1y1_y, result_x1y2_y, result_x2y1_y, result_x2y2_y))

    try:
        ADC_array = ADC_dicom.pixel_array
    except:
        image = sitk.ReadImage(ADC_path)
        image_array = sitk.GetArrayFromImage(image)
        ADC_array = np.int16(image_array)

    ADC_array = 255 * ((ADC_array - np.amin(ADC_array)) / (np.amax(ADC_array) - np.amin(ADC_array)))

    if len(ADC_array.shape)==3:
        ADC_array=ADC_array[0]
    try:
        if y2_raw_ADC - y1_raw_ADC < 50:
            y1_raw_ADC = int(y1_raw_ADC - (50 - (y2_raw_ADC - y1_raw_ADC)) / 2)
            y2_raw_ADC = int(y2_raw_ADC + (50 - (y2_raw_ADC - y1_raw_ADC)) / 2)
        if x2_raw_ADC - x1_raw_ADC < 50:
            x1_raw_ADC = int(x1_raw_ADC - (50 - (x2_raw_ADC - x1_raw_ADC)) / 2)
            x2_raw_ADC = int(x2_raw_ADC + (50 - (x2_raw_ADC - x1_raw_ADC)) / 2)

        crop_img = ADC_array[y1_raw_ADC:y2_raw_ADC, x1_raw_ADC:x2_raw_ADC]

        save_path = './media/output/'
        Image.fromarray(ADC_array).convert('L').save(save_path+'adc.png')
        Image.fromarray(crop_img).convert('L').resize((224,224)).save(save_path+'adc_crop.png')

        im=Image.fromarray(ADC_array).convert('RGB')

        draw=ImageDraw.Draw(im)
        draw.rectangle((x1_raw_ADC,y1_raw_ADC,x2_raw_ADC,y2_raw_ADC),outline=(255,255,0),width=3)
        im.save(save_path+'adc_contour.png')
    except:
        print('ADC error!')
    return

# process a BVAL png, and save its contour, prostate region in './media/output/'
def bval_transform(bval_path,T2_path,x1_T2,y1_T2,x2_T2,y2_T2):
    y1_inraw = int(y1_T2)
    x1_inraw = int(x1_T2)
    y2_inraw = int(y2_T2)
    x2_inraw = int(x2_T2)
```

```
T2_dicom = pydicom.dcmread(T2_path)

ImageOrientationPatient = T2_dicom.ImageOrientationPatient
ImagePositionPatient = T2_dicom.ImagePositionPatient
PixelSpacing = T2_dicom.PixelSpacing

a = np.array([[ImageOrientationPatient[0] * PixelSpacing[0],
               ImageOrientationPatient[3] * PixelSpacing[1],
               ImagePositionPatient[0]],
              [ImageOrientationPatient[1] * PixelSpacing[0],
               ImageOrientationPatient[4] * PixelSpacing[1],
               ImagePositionPatient[1]],
              [ImageOrientationPatient[2] * PixelSpacing[0],
               ImageOrientationPatient[5] * PixelSpacing[1],
               ImagePositionPatient[2]]
              ])

threeD_x1y1 = np.dot(a, (np.array([x1_inraw, y1_inraw, 1])))
threeD_x1y2 = np.dot(a, (np.array([x1_inraw, y2_inraw, 1])))
threeD_x2y1 = np.dot(a, (np.array([x2_inraw, y1_inraw, 1])))
threeD_x2y2 = np.dot(a, (np.array([x2_inraw, y2_inraw, 1])))

bval_dicom = pydicom.dcmread(bval_path)

ImageOrientationPatient = bval_dicom.ImageOrientationPatient
ImagePositionPatient = bval_dicom.ImagePositionPatient
PixelSpacing = bval_dicom.PixelSpacing

a_bval = np.array([[ImageOrientationPatient[0] * PixelSpacing[0],
                   ImageOrientationPatient[3] * PixelSpacing[1],
                   ImagePositionPatient[0]],
                  [ImageOrientationPatient[1] * PixelSpacing[0],
                   ImageOrientationPatient[4] * PixelSpacing[1],
                   ImagePositionPatient[1]],
                  [ImageOrientationPatient[2] * PixelSpacing[0],
                   ImageOrientationPatient[5] * PixelSpacing[1],
                   ImagePositionPatient[2]]
                  ])

result_x1y1 = np.linalg.solve(a_bval, threeD_x1y1)
result_x1y1_x = result_x1y1[0]
result_x1y1_y = result_x1y1[1]

result_x1y2 = np.linalg.solve(a_bval, threeD_x1y2)
result_x1y2_x = result_x1y2[0]
result_x1y2_y = result_x1y2[1]

result_x2y1 = np.linalg.solve(a_bval, threeD_x2y1)
result_x2y1_x = result_x2y1[0]
result_x2y1_y = result_x2y1[1]
```

```
    result_x2y2 = np.linalg.solve(a_bval, threeD_x2y2)
    result_x2y2_x = result_x2y2[0]
    result_x2y2_y = result_x2y2[1]

    x1_raw_bval = int(min(result_x1y1_x, result_x1y2_x, result_x2y1_x, result_x2y2_x))
    x2_raw_bval = int(max(result_x1y1_x, result_x1y2_x, result_x2y1_x, result_x2y2_x))
    y1_raw_bval = int(min(result_x1y1_y, result_x1y2_y, result_x2y1_y, result_x2y2_y))
    y2_raw_bval = int(max(result_x1y1_y, result_x1y2_y, result_x2y1_y, result_x2y2_y))

    try:
        bval_array = bval_dicom.pixel_array
    except:
        image = sitk.ReadImage(bval_path)
        image_array = sitk.GetArrayFromImage(image)
        bval_array = np.int16(image_array)

    bval_array = 255 * ((bval_array - np.amin(bval_array)) / (np.amax(bval_array) - np.amin(bval_array))

    if len(bval_array.shape)==3:
        bval_array=bval_array[0]
    try:
        if y2_raw_bval - y1_raw_bval < 50:
            y1_raw_bval = int(y1_raw_bval - (50 - (y2_raw_bval - y1_raw_bval)) / 2)
            y2_raw_bval = int(y2_raw_bval + (50 - (y2_raw_bval - y1_raw_bval)) / 2)
        if x2_raw_bval - x1_raw_bval < 50:
            x1_raw_bval = int(x1_raw_bval - (50 - (x2_raw_bval - x1_raw_bval)) / 2)
            x2_raw_bval = int(x2_raw_bval + (50 - (x2_raw_bval - x1_raw_bval)) / 2)
        crop_img = bval_array[y1_raw_bval:y2_raw_bval, x1_raw_bval:x2_raw_bval]
        save_path = './media/output/'
        Image.fromarray(bval_array).convert('L').save(save_path + 'bval.png')
        Image.fromarray(crop_img).convert('L').resize((224,224)).save(save_path + 'bval_crop.png')

        im = Image.fromarray(bval_array).convert('RGB')

        draw = ImageDraw.Draw(im)
        draw.rectangle((x1_raw_bval, y1_raw_bval, x2_raw_bval, y2_raw_bval), outline=(255, 255, 0), wid
        im.save(save_path + 'bval_contour.png')
    except:
        print('bval error!')
    return

>
```

Domain adaptation model training for mpMRI-based PLDC

For DA model (i.e., CMD^{2}A-Net) training, the prostate regions from both the source and target domains are scaled to 224 × 224 pixels. Random rotation is applied for data augmentation. Adam optimizer is chosen. We initialized our model with the pre-trained CM-Net, in order to facilitate its convergence. To be specific, we trained both the coarse segmentation module and classifier of CM-Net first using codes under ./joint_-model, with the combined samples from both domains, the best models are saved (i.e., T2: weight2_1.h5, ADC: weight3_1.h5, hDWI: weight4_1.h5). Then, we optimized the total loss of CMD^{2}A-Net with the source and target samples. By co-training all the modules, the models with the highest accuracy (i.e., T2: weight2_2.h5, ADC: weight3_2.h5, hDWI: weight4_2.h5) are saved for testing. The PLDC testing in the target samples from local cohort can be achieved by the following codes:

```
<
def main_testing(dicom_T2_path):

    output_path = './media/output/'

    # remove all the exsiting files for current case testing
    for file in os.listdir(output_path):
        os.remove(os.path.join(output_path,file))

    # preprocess a T2 dicom to raw png, and save the png in "./media/convert_img/T2"
    image_path = preprocessing(dicom_T2_path,dicom_T2_path.split('/')[-1]) #image_path: raw T2 png path

    # save T2 png, prostate_roi.png, T2_with contour.png under output folder
    x1, y1, x2, y2 = predict(image_path)

    # return malignacy and save './media/output/T2_lesion.png'
    prostate_roi_T2_path='./media/output/prostate_roi.png'
    malignancy_T2 = predict_malignancy_T2(prostate_roi_T2_path,image_path)

    malignancy_ADC=malignancy_T2
    malignancy_BVAL=malignancy_T2

    adc_dicom = os.listdir('./media/upload/ADC/')
    if len(adc_dicom)==1:
        if os.path.isfile(os.path.join('./media/upload/ADC/',adc_dicom[0])):
            if adc_dicom[0].endswith("dcm"):
                dicom_ADC_path = os.path.join('./media/upload/ADC/', adc_dicom[0])

                # save raw png, raw png with contour,and cropped prostate png under './media/output/'
                adc_transform(dicom_ADC_path, dicom_T2_path, x1, y1, x2, y2)

                prostate_roi_ADC_path = './media/output/adc_crop.png'
                prostate_ADC_path = './media/output/adc.png'

                if os.path.exists(prostate_roi_ADC_path):
                    malignancy_ADC = predict_malignancy_ADC(prostate_roi_ADC_path)
                else:
                    malignancy_ADC = predict_malignancy_ADC(prostate_ADC_path)

    bval_dicom = os.listdir('./media/upload/BVAL/')
    if len(bval_dicom)==1:
        if os.path.isfile(os.path.join('./media/upload/BVAL/',bval_dicom[0])):
            if bval_dicom[0].endswith("dcm"):
                dicom_BVAL_path = os.path.join('./media/upload/BVAL/', bval_dicom[0])

                # save raw png, raw png with contour,and cropped prostate png under './media/output/'
                bval_transform(dicom_BVAL_path, dicom_T2_path, x1, y1, x2, y2)
                prostate_roi_BVAL_path = './media/output/bval_crop.png'
                prostate_BVAL_path = './media/output/bval.png'

                if os.path.exists(prostate_roi_BVAL_path):
                    malignancy_BVAL = predict_malignancy_BVAL(prostate_roi_BVAL_path)
                else:
                    malignancy_BVAL = predict_malignancy_BVAL(prostate_BVAL_path)

    malignancy = np.argmax(malignancy_T2 + malignancy_ADC + malignancy_BVAL)
    # print('malignancy:', malignancy)
    return malignancy>

def predict_malignancy_T2(prostate_roi,raw_image_path):

    if os.path.exists(prostate_roi):

        image=Image.open(prostate_roi).convert('L')
        image = image.resize((224, 224))
        img=np.array(image)/255

        model = sinnet_share('./doc/weights/weight2_1.h5')
        model.load_weights('./doc/weights/weight2_2.h5')

        aa = img.reshape(1, 224, 224, 1)
        out_0_2 = model.predict([aa,aa])

        del model
        K.clear_session()
        gc.collect()

        raw_image_RGB = np.array(image.convert('RGB'))
        pred_mask = Image.fromarray(255*out_0_2[0][0][:,:,0]).convert("L")
        pred_mask = np.array(pred_mask)

        ret, binary = cv2.threshold(pred_mask, 40, 255, cv2.THRESH_BINARY)  # second para is threhold
        contours, hierarchy = cv2.findContours(binary, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

        if np.size(contours) != 0:
            #method 1
            contour = max(contours, key = cv2.contourArea)
            image_with_contour = cv2.drawContours(raw_image_RGB,[contour],-1, color=(0, 255, 255),lineT
            #method 2
            #image_with_contour = cv2.drawContours(raw_image_RGB, [contours[0]], -1, color=(0, 0, 255),

            # draw pred mask
            image_with_contour = cv2.addWeighted(raw_image_RGB,1,image_with_contour,0.3,0)
            # save final label and pred result
            cv2.imwrite('./media/output/T2_lesion.png',image_with_contour)
        else:
            # print("np.size(contours)= 0")
            pass
        print('done predict_malignancy_T2!')

        return out_0_2[1][0]
    else:

        image=Image.open(raw_image_path).convert('L')
        image = image.resize((224, 224))
        img=np.array(image)/255

        model = sinnet_share('./doc/weights/weight2_1.h5')
        model.load_weights('./doc/weights/weight2_2.h5')

        aa = img.reshape(1, 224, 224, 1)
        out_0_2 = model.predict([aa,aa])

        del model
        K.clear_session()
        gc.collect()

        return out_0_2[1][0]

def predict_malignancy_ADC(prostate_roi):

    image=Image.open(prostate_roi).convert('L')
    image = image.resize((224, 224))
    img=np.array(image)/255

    model = sinnet_share('./doc/weights/weight3_1.h5')
    model.load_weights('./doc/weights/weight3_2.h5')

    aa = img.reshape(1, 224, 224, 1)
    out_0_2 = model.predict([aa,aa])

    del model
    K.clear_session()
    gc.collect()

    # print('ADC done!')
    return out_0_2[1][0]

def predict_malignancy_BVAL(prostate_roi):

    image=Image.open(prostate_roi).convert('L')
    image = image.resize((224, 224))
    img=np.array(image)/255

    model = sinnet_share('./doc/weights/weight4_1.h5')
    model.load_weights('./doc/weights/weight4_2.h5')

    aa = img.reshape(1, 224, 224, 1)
    out_0_2 = model.predict([aa,aa])

    del model
    K.clear_session()
    gc.collect()

    # print('BVAL done!')
    return out_0_2[1][0]
```

Acknowledgements

This work is partially supported by the Research Grants Council (RGC) of Hong Kong (Ref. No.: 17209021, 17207020, 17205919, 17206818, T42-409/18-R), Innovation and Technology Commission (ITC) (Project No.: MRP/029/20X), and Multi-scale Medical Robotics Center Limited.

K.W. Kwok and V. Vardhanabhuti are co-corresponding authors.

J.D., X.W., Z.L., and J.X. conceived and designed the study supervised by K.W.K., V.V., and Q.D.; V.V. was responsible for collecting data from the Hong Kong hospital local cohorts as well as conducting manual annotations; Z.L. conducted data-cleaning, weak label generation, and image processing; J.D. conducted deep learning algorithm implementations, experimental design, data analysis, and drafted the paper with the assistance and feedback of all other co-authors; Y.L. developed the open-sourced software; K.W.K., X.W., Y.L., and Y.L.N., revised the paper draft, and V.V. and Q.D. provided constructive feedback on the developed models; All authors read and approved the final manuscript.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The public dataset I2CVB can be available online. Raw images and image-level labels of the public dataset PROSTATEx are also accessed online. The other local cohort datasets used in this study are not publicly available at this time, as the data from the third party contain patients' confidential information and are not authorized to be shared openly at this stage. Qualified researchers with reasonable requests for access to the data should contact the corresponding authors after permission from the local hospitals. Any data use will be restricted to non-commercial research purposes.

The codes to train our models are based on Keras using Tensorflow (v1.15) as backend. The models were implemented using Python (v3.6.5).

References

[1] G. Lemaître, R. Martí, J. Freixenet, J. C. Vilanova, P. M. Walker, F. Meriaudeau, medicine, *Computers in biology* **2015**, 60, 8.

[2] Z. Liu, W. Jiang, K.-H. Lee, Y.-L. Lo, Y.-L. Ng, Q. Dou, V. Vardhanabhuti, K.-W. Kwok, presented at *Workshop on Artificial Intelligence in Radiation Therapy*, **2019**.

[3] Y. Chen, W. Wang, E. J. Schmidt, K.-W. Kwok, A. N. Viswanathan, R. Cormack, Z. T. H. Tse, *IEEE/ASME Transactions on Mechatronics* **2015**, 21, 956.

[4] R. Alkadi, F. Taher, A. El-Baz, N. Werghi, *Journal of digital imaging* **2019**, 32, 793.

[5] C.-L. Cheung, J. D.-L. Ho, V. Vardhanabhuti, H.-C. Chang, K.-W. Kwok, *IEEE/ASME Transactions on Mechatronics* **2020**, 25, 1016.

[6] V. Vitiello, K.-W. Kwok, G.-Z. Yang, in *Medical Robotics*, Elsevier, **2012**.

[7] Z. Dong, Z. Guo, K.-H. Lee, G. Fang, W. L. Tang, H.-C. Chang, D. T. M. Chan, K.-W. Kwok, *IEEE Robotics Automation Letters* **2019**, 4, 1964.

[8] Q. Liu, Q. Dou, L. Yu, P. A. Heng, *IEEE transactions on medical imaging* **2020**, 39, 2713.

[9] X. Wang, Y. Li, K.-W. Kwok, AI, *Frontiers in Robotics* **2021**, 280.

[10] G. Fang, M. C. Chow, J. D. Ho, Z. He, K. Wang, T. Ng, J. K. Tsoi, P.-L. Chan, H.-C. Chang, D. T.-M. Chan, Y.-h. Liu, F. Holsinger, J. Y.-K. Chan, K.-W. Kwok, *Science Robotics* **2021**, 6, eabg5575.

[11] Y. Chen, J. Ge, K.-W. Kwok, K. R. Nilsson, M. Fok, Z. T. Tse, *Journal of cardiovascular magnetic resonance* **2014**, 16, 1.

[12] W. Jiang, Z. Liu, K.-H. Lee, S. Chen, Y.-L. Ng, Q. Dou, H.-C. Chang, K.-W. Kwok, *arXiv preprint arXiv:.09745* **2019**.

[13] a)K. Wang, C. H. Mak, J. D. Ho, Z. Liu, K. Y. Sze, K. K. Wong, K. Althoefer, Y. Liu, T. Fukuda, K.-W. Kwok, *Advanced Intelligent Systems* **2021**, 3, 2100089; b)Y.-L. Ng, M. C. Lo, K.-H. Lee, X. Xie, T. N. Kwong, M. Ip, L. Zhang, J. Yu, J. J. Sung, W. K. Wu, S. H. Wong, K.-W. Kwok, *Advanced Intelligent Systems* **2021**, 3, 2000188.

[14] a)Z. He, Z. Dong, G. Fang, J. D.-L. Ho, C.-L. Cheung, H.-C. Chang, C. C.-N. Chong, J. Y.-K. Chan, D. T. M. Chan, K.-W. Kwok, *IEEE Robotics Automation Letters* **2020**, 5, 2100; b)K.-W. Kwok, K.-H. Lee, Y. Chen, W. Wang, Y. Hu, G. C. Chow, H. S. Zhang, W. G. Stevenson, R. Y. Kwong, W. Luk, *Circulation* **2014**, 130, A18568.

[15] Z. Wang, D. Acuna, H. Ling, A. Kar, S. Fidler, presented at *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, **2019**.

[16] K. He, X. Zhang, S. Ren, J. Sun, presented at *Proceedings of the IEEE conference on computer vision and pattern recognition*, **2016**.

[17] Y. Ganin, V. Lempitsky, presented at *International conference on machine learning*, **2015**.

[18] B. Sun, K. Saenko, presented at *European conference on computer vision*, **2016**.

[19] L. Van der Maaten, G. Hinton, *Journal of machine learning research* **2008**, 9.

[20] Z. Dong, X. Wang, G. Fang, Z. He, J. D.-L. Ho, C.-L. Cheung, W. L. Tang, X. Xie, L. Liang, H.-C. Chang, C. K. Ching, K.-W. Kwok, **2022**, IEEE Transactions on Robotics.