Supplementary Material
Foundation Ark: Accruing and Reusing Knowledge for Superior and Robust Performance

DongAo Ma[1], Jiaxuan Pang[1], Michael B. Gotway[2], and Jianming Liang[1]

[1] Arizona State University, Tempe, AZ 85281, USA
{dongaoma,jpang12,jianming.liang}@asu.edu
[2] Mayo Clinic, Scottsdale, AZ 85259, USA
Gotway.Michael@mayo.edu

Abstract. This supplementary material complements the paper titled "Foundation Ark: Accruing and Reusing Knowledge for Superior and Robust Performance". It is organized as follows. In Sec. A, we present a comprehensive list of diagnostic labels from different public datasets, revealing marked label heterogeneity across institutions. Sec. B offers a comparison between Ark and other existing works concerning the assembly of public datasets, emphasizing Ark's label-agnostic and task-scalable advantages. Section C includes ablation studies that demonstrate the necessity of the projector and consistency loss, along with the superiority of the teacher model. In Sections D and E, we present pseudocode for Ark's cyclic pretraining and elaborate on the experimental setups. Lastly, Section F contains acknowledgments for support.

Knowledge is power — Mac Flecknoe

Power comes not from knowledge kept but from knowledge shared — Bill Gates

A Heterogeneous Labels

Table 3. As listed in this table, datasets created at different institutions tend to be annotated differently even when addressing the same clinical issue. Our Ark aims to accrue and reuse expert knowledge from heterogeneous labels with numerous public datasets to pretrain generic source models that are more robust, generalizable, and transferable to application-specific target tasks, demonstrating superior and robust performance over the SOTA fully/self-supervised baselines (Table 2) and Google CXR-FM (Fig. 2). The challenge of learning from heterogeneous labels is addressed in Ark via multi-task heads and cyclic pretraining (Fig. 1).

Dataset	Inconsistencies in diagnostic labels associated with popular public X-rays datasets
1.CXPT 6.MMIC	No Finding, Enlarged Cardiomediastinum, Cardiomegaly, Lung Opacity, Lung Lesion, Edema, Consolidation, Pneumonia, Atelectasis, Pneumothorax, Pleural Effusion, Pleural Other, Fracture, Support Devices
2.NIHC	Atelectasis, Cardiomegaly, Effusion, Infiltration, Mass, Nodule, Pneumonia, Pneumothorax, Consolidation, Edema, Emphysema, Fibrosis, Pleural Thickening, Hernia
3.RSNA	Normal, No Lung Opacity/Not Normal, Lung Opacity
4.VINC	Pleural Effusion, Lung Tumor, Pneumonia, Tuberculosis, Other Diseases, No Finding
5.NIHS	Tuberculosis

B Other Works for Assembling Public Datasets

Table 4. Our Ark is dataset/task-agnostic as it does not require prior label "understanding" of public datasets. Unlike the listed example works that need to manually assemble the labels into a pre-defined list and train a dynamic controller/adapter as directives for different tasks, Ark is designed with pluggable multi-task heads and cyclic pretraining (Sec. 2) to offer flexibility and scalability for adding new tasks without manually consolidating heterogeneous labels or training task-specific controllers/adapters.

Related works	How to preprocess labels?	When a new task comes?
Label-Assemble[1]	Need a pre-defined label list	Update the label list and retrain the adapter if any labels aren't in the original list
DoDNet[2]	Need a pre-defined task list	Renew the task list and retrain the controller when adding new tasks
CLIP-diven[3]	Need manual designs of prompt to get CLIP embeddings	Re-generate the CLIP embedding for any new classes and retrain the controller
Ark	Task-agnostic, use all readily-accessible labels directly as they are	Plug in Ark a new head, independent from existing tasks, for the new task, no modification on the rest architecture

C Ablation study

Table 5. Our ablation studies on Ark-5 via linear probing show the projector and consistency loss are essential and the teacher significantly outperforms the student.

Model	Projector	$\mathcal{L}_{consist}$	2.NIHC	3.RSNA	4.VINC	5.NIHS
Teacher	×	×	81.09±0.08	74.21±0.42	94.89±0.07	98.81±0.25
Teacher	×	✓	81.19±0.05	74.42±0.25	95.24±0.08	99.01±0.08
Student	✓	✓	81.34±0.04	74.12±0.11	94.85±0.07	99.17±0.07
Teacher	✓	✓	**81.39±0.02**	**74.74±0.19**	**95.35±0.04**	**99.41±0.03**

[1] Zhu *et al.*(2022), Assembling Existing Labels from Public Datasets to Diagnose Novel Diseases: COVID-19 in Late 2019

[2] Zhang *et al.*(2020), DoDNet: Learning to segment multi-organ and tumors from multiple partially labeled datasets

[3] Liu *et al.*(2023), CLIP-Driven Universal Model for Organ Segmentation and Tumor Detection

D Pseudocode for Ark's cyclic pretraining

As illustrated in Fig. 1 and described in Algorithm 1, Ark is built on a teacher-student model, whose student is augmented with multi-task heads (each corresponding to one task) and trained via cyclic pretraining. Cyclic pretraining is an iterative process. At each iteration, the student aims to accrue knowledge from every expert annotation through its corresponding task head by sequentially scanning all datasets (tasks) one by one for one epoch. At the end of each task, the accrued knowledge is accumulated into the teacher (via EMA) and reused to help accrue more knowledge from the expert annotations associated with the next dataset. To reinforce the feedback loop between the student and teacher, after their encoders, a projector is introduced to map the representations to the same feature space via the consistency loss, also serving as the embedding for linear probing in our evaluation. After pretraining, the accumulated knowledge in the teacher is reused and transferred to the application-specific target tasks.

Algorithm 1: A round of Ark's cyclic pretraining

Data: Datasets: $\mathcal{D} = \{D_1, D_2, ..., D_n\}$; Sample: image-label pair $(x, y) \in \mathcal{D}_i$
Functions: Data augmentation: $\tau_1(\cdot)$, $\tau_2(\cdot)$; Dataset/task-specific losses: $\{\mathcal{L}_{D1}(\cdot,\cdot), \mathcal{L}_{D2}(\cdot,\cdot), ..., \mathcal{L}_{Dn}(\cdot,\cdot)\}$; Consistency loss: $\mathcal{L}_{const}(\cdot,\cdot)$; Loss update by SGD optimizer: $Update_{sgd}(\cdot,\cdot)$
Trainable Parameters: Student's encoder and projector: e_s, p_s; Multi-task heads $\mathcal{H} = \{h_1, h_2, ..., h_n\}$
Stop Gradient: Teacher's encoder and projector: e_t, p_t
Hyperparameters: Momentum: λ

$\{e_t, p_t\} \leftarrow \{e_s, p_s\}$ `// initialize teacher with student's parameters`
for D_i **in** $D_1, D_2, ..., D_n$ **do**
  `/* train student for one epoch */`
  for (x, y) **in** D_i **do**
    $x' = \tau_1(x)$
    $x'' = \tau_2(x')$
    $emb_t, emb_s = p_t(e_t(x')), p_s(e_s(x''))$
    $pred = h_i(emb_s)$
    $Loss = \mathcal{L}_{Di}(pred, y) + \mathcal{L}_{const}(emb_t, emb_s)$
    $Update(\{e_s, p_s, h_i\}, Loss)$
  `/* Update teacher by student's parameters via epoch-wise EMA */`
  $\{e_t, p_t\} \leftarrow \lambda\{e_t, p_t\} + (1 - \lambda)\{e_s, p_s\}$

E Experiment details

Pretraining: We have trained Ark-5/6 with 335,484/704,363 chest X-rays from the first 5/6 datasets in Table 1 collected by 5/6 different institutions around the world and annotated by their experts. We use their originally-provided labels (Table 3), showing marked differences across institutions. To avoid test-image leaks, all validation and test data are excluded from the Ark pretraining. We employ the base version of the Swin transformer with an input resolution of 224×224 as the backbone. The encoders in teacher and student are initialized with the officially released weights trained on ImageNet[1], and the projectors and the multi-task heads are randomly initialized. The task-specific (classification) loss is associate with each dataset based on its labels. We use binary cross-entropy for the binary/multi-label classification tasks (Dataset 1-2, 4-6) and cross-entropy for the multi-class classification task (Dataset 3). Besides, we use mean-squared error for the consistency loss. We optimize the student model using SGD optimizer with an initial learning rate of 0.3, and a batch size of 200 distributed across 4 Nvidia V100 GPUs with a memory of 32 GB per-card; we apply a *stop-gradient* operator on the teacher and update it using *epoch-wise EMA* of the student parameters at the end of each task with an initial momentum of 0.9. The image augmentation function $\tau_1(.)$ includes random cropping and rotation, and $\tau_2(.)$ includes randomly changing brightness, contrast, and Gamma distribution of an image.

Evaluation: We have evaluated Ark-5 and Ark-6 via transfer learning and compared them with SOTA fully-supervised and self-supervised models (Table 2). For fair comparisons, we follow the SoTA[2] and apply the same augmentations for all methods. We measure the performance of binary/multi-label classification by AUC (area under the ROC curve), multi-class classification by accuracy, and segmentation by Dice. We perform at least 10 trials, report the mean and standard deviation of the performance metrics, and further present statistical analysis based on an independent two-sample t-test.

To provide a more comprehensive evaluation, we have conducted linear probing (Fig. 2) and analyzed gender biases (Fig. 3) on the Ark models in comparison with Google CXR-FM. We pre-generated the embeddings for all images in the target tasks from Ark-5, Ark-6 and Google CXR-FM[3], and then train a simple linear classifier for each target task.

For the gender biases analysis, we follow the train/test splits in the GenderBias_CheXNet repository[4] to ensure a balanced number of cases per class in the 20 male-only and 20 female-only folds, where the labels "No Finding" and "Support Device" are excluded. We train 40 linear classifiers on male-only and female-only splits using embeddings from Ark-6 and CXR-FM to evaluate their gender biases. We then evaluate these classifiers on the corresponding

[1] GitHub.com/SwinTransformer/storage/releases/download/v1.0.0/swin_base_patch4_window7_224_22kto1k.pth

[2] GitHub.com/JLiangLab/BenchmarkTransformers

[3] GitHub.com/Google-Health/imaging-research/tree/master/cxr-foundation

[4] GitHub.com/N-Nieto/GenderBias_CheXNet

Table 5. Experimental configuration details.

Ark pretraining setup	
Backbone	Swin Transformer Base (input resolution: 224 × 224)
Initialization	Encoders: officially released ImageNet weights Projectors and Multi-task heads: random weights
Loss function	Task-specific loss: binary cross-entropy (BCE) for Dataset 1-2, 4-6 cross-entropy (CE) for Dataset 3 Consistency loss: mean-squared error (MSE)
Optimization	Student: SGD optimizer, learning rate of 0.3, Cosine scheduler Teacher: Stop gradient, EMA update, momentum of 0.9
Pretraining	200 rounds (iterates through all datasets 200 times)
Augmentation	$\tau_1(.)$: Random cropping and rotation $\tau_2(.)$: Random changing of image brightness, contrast, and Gamma distribution
Devices	4 Nvidia V100 GPUs (32GB)
Ark evaluation setup	
Tranferred model	Teacher's encoder
Metrics	Binary/Multi-label classification: Area under the ROC curve (AUC) Multi-class classification: Accuracy (ACC) Segmentation: Dice similarity coefficient (Dice)
Performance	Mean and Standard Deviation of the metrics for 10 trials
Significance test	Independent two-sample t-test (p-value < 0.05)

male/female-only test splits and report the average performance over the 20 folds.

Table 5 lists the key setups used in Ark's pretraining and evaluation protocols.

F Acknowledgements

This research has been supported in part by ASU and Mayo Clinic through a Seed Grant and an Innovation Grant, and in part by the NIH under Award Number R01HL128785. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH. This work has utilized the GPUs provided in part by the ASU Research Computing and in part by the Bridges-2 at Pittsburgh Supercomputing Center through allocation BCS190015 and the Anvil at Purdue University through allocation MED220025 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296. We also acknowledge Google for granting us access to CXR Foundation API, which enabled us to generate the embeddings for the target datasets. The content of this paper is covered by patents pending.