Characterization of AI Model Configurations For Model Reuse

Anonymous ECCV submission

Paper ID 100

Abstract. With the widespread creation of artificial intelligence (AI) models in biosciences, researchers are reusing AI models trained for specific tasks. This work is motivated by the need to characterize AI models for reuse and dissemination based on metrics derived from optimization curves captured during model training. Such AI model characterization can aid future model accuracy refinement, inform users about model hyper-parameter sensitivity, and assist in model reuse according to multi-purpose objectives. The challenges lie in understanding relationships between AI model characteristics and optimization curves, defining and validating quantitative AI model metrics, and disseminating metrics with trained AI models. We approach these challenges by analyzing optimization curves generated for image segmentation and classification tasks with respect to AI model characteristics reused for many purposes.

Keywords: Optimization and learning methods; Efficient training and inference methods; Medical, biological, and cell microscopy

1 Introduction

The problems of reusing artificial intelligence (AI) models range from defining a standard AI model file format to sharing the code and AI models via repositories [8]. Multiple communities come together to define a standard file format, such as Open Neural Network Exchange (ONNX)[5], and agree on sharing application code, installation software dependencies, AI frameworks, and packaging via open framework projects (e.g., Conda, Colaboratory, PyTorch, TensorFlow), code and model repositories (e.g., GitHub, BitBucket, Model Zoo, Model Depot, TensorFlow Hub), and software packaging and distribution solutions (e.g., Docker, Apache Zookeeper, Apache Kafka) [8]. In the broad range of AI model reusability problems, our focus is on specific sub-problems related to characterizing AI models for the purpose of value added to parties reusing the models.

The need within the scientific imaging community for AI model characterization is driven by several factors. First, the scientific community strives for reproducible research results. Second, domain-specific applications with focus on special objects of interest acquired by unique imaging modalities struggle with insufficient training data (in comparison to typical imaging modalities and objects in computer vision datasets, e.g., ImageNet or Microsoft Common Objects in Context (COCO)) and privacy concerns, especially in the medical imaging field. Finally, the sciences struggle with a general lack of computational resources for AI model training compared to the resources available to large companies. We listed in Table 1several example tasks that are reusing AI models, utilizing task specific inputs, and benefiting from additional metrics. The terms "optimal configuration" and "explored configurations" in Table 1 refer to the one desirable configuration according to some optimization criteria and to the set of configurations that were evaluated during optimization. The columns labeled as "Reused" and "Task specific" refer to reused data and information generated by other parties, and to hardware and software artifacts that are specific to completing each task. The column "Needed metrics" highlights the key criteria that define a success of completing each task. For instance, reusing AI models from the TrojAI challenges with 1.7TB of AI models [10] to establish robustness of poisoned AI models to training data perturbations will benefit from sharing used error metric, uniformity of training data defining predicted classes of synthetic traffic signs superimposed on real background, and a model stability metric (e.g., percentage of AI models that did not converge when trojans were injected). Characterizing AI models improves the input metadata about AI models for reuse and reproducibility. Additionally, model reuse saves computational resources and time while providing higher final model accuracy.

Table 1. Example tasks reusing AI models and benefiting from additional metrics

Task	Reused	Task specific	Needed metrics
1. Inference on new hardware	Trained model	Hardware	GPU utilization and exec. time
2. Reproduce training	Training data Model architecture Optimal configuration	Hardware Training env.	Error GPU utilization and exec. time
3. Model refinement by param. optimization	Training data Trained model Explored configurations	Hyper-param.	Convergence GPU utilization and exec. time
4. Network architecture search	Training datasets	AI graphs Hyper-param.	Data-model representation
5. Transfer learning	Trained models Optimal sets of hyper-param.	Other training datasets	Gain from pretraining Data domain cross compatibility
6. Evaluate robustness to data, architecture and hyper-param. perturbations	Training data Model architecture Optimal configuration	Subsets of training datasets AI graphs Hyper-param.	Data uniformity Error Stability

Our problem space is illustrated in Figure 1 with a training configuration defined by training dataset, AI architecture, and hyper-parameters. The basic research question is: "what information can be derived from optimization and GPU utilization curves to guide a reuse of a trained AI model?" The objective of this work is to design computable metrics from optimization curves that would be included in model cards [15] and support decisions about when and how to reuse disseminated trained AI models. Our assumption is that data collected during training sessions are common to all modeling tasks including image classification and segmentation (tasks of our specific interest) and, therefore, the characteristics can be applied to a general set of AI models.

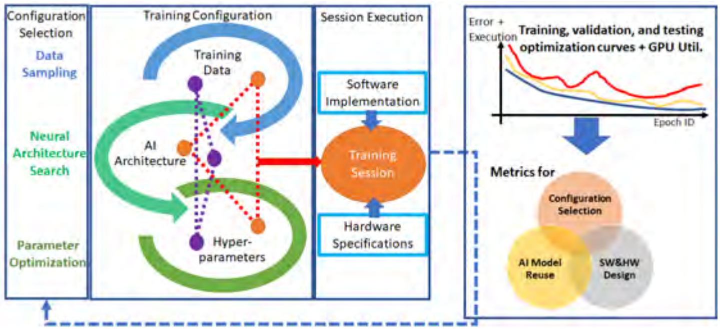


Fig. 1. A space of model training configuration space that is evaluated in each training session by analyzing training, validation, and testing optimization curves, and a set of metrics designed for configuration selection, model reuse, and software+hardware execution design.

Our approach to defining AI model characteristics from train, validation, and test optimization curves consists of three steps:

a) simulate and analyze relationships between AI model coefficients and optimization curves,
b) define mathematical functions used in analytical and statistical analyses that characterize a trained AI model from optimization curves, and
c) design a recommendation system based on extracted and validated quantitative characteristics of trained AI models.

The optimization curves are typically AI model accuracy or error metrics collected over many epochs from train, validation, and test datasets.

The overarching challenges lie in (a) limited information content in optimization curves that combine contributions from model architecture, training hyper-parameters, and training dataset, (b) limited a priori knowledge about relationships among parts of AI solutions that could be used for validation of quantitative AI model characteristics, and (c) computational resources needed to generate a large number of optimization curves for a variety of AI-based modeling tasks. Although the optimization curves contain limited information, they represent an information resource suitable for sharing with AI models to support legitimate reuse while protecting trade secrets (and pirated reuse) in commercial settings. To overcome the challenges, (a) we run simulations for a range of neural network configurations on 2D dot patterns to explore relationships between optimization curves and model configurations, (b) we use apriori information about datasets and modeling task complexity to validate metrics derived from optimization curves, and (c) we leverage optimization curves from a network architecture search database [22] and our model training sessions on five datasets and six architectures.

Relation to prior work: The concept of describing AI models has already been discussed in the past (Datasheets for datasets [6], The Dataset Nutrition Labels [9,21], Google AI Model Cards [15]). The published work on Datasheets for datasets and Dataset Nutrition Labels has been focused mainly on training datasets from the perspective of fairness. The fairness aspect is documented via data attributes, motivation for collection, data composition, collection process, and recommended uses in [6], as well as via design of ranking widgets in [9]. In contrast to [9,21,6], our work is focused on documenting lessons-learned from the optimization curves collected during training sessions. While a placeholder for model performance measures has been designated in the AI model cards [15] (i.e., under Metrics heading), the metrics have not been defined yet, which is the gap our work is trying to address. In addition, our work aims at utilizing the information that is not preserved with disseminated AI models for the multi-purpose reuse of the AI models right now, although multiple platforms for optimizing AI model configurations generate such information, such as TensorBoard [1], Optuna [2], Vizier [7], Autotune [11], or Experiment Manager [14]. Finally, our experiments are constrained by computational resources and therefore we leverage optimization values from a neural architecture search (NAS) database [22] with evaluated 5 million models and utilizing over 100 tensor processing unit (TPU) years of computation time.

Our contributions are (a) in exploring relationships between AI model coefficients and optimization curves, (b) in defining, implementing, and validating metrics of AI models from optimization curves for accompanying shared AI models, and (c) generating and leveraging optimization curves for image segmentation and classification tasks in a variety of reuse scenarios. The novelty of this work lies in introducing (a) computable metrics for model cards [15] that can provide cost savings for further reuse of AI models and (b) a recommendation system that can guide scientists in reusing AI models.

2 Methods

This section outlines three key components of using optimization curves for reuse of trained AI models: (1) relationships between AI model coefficients and optimization curves, (2) definitions of AI model metrics, and (3) design of a recommendation system.
Relationship between AI model coefficients and optimization curves: To address the basic research question "what information can be derived from optimization and GPU utilization curves to guide a reuse of a particular AI model?", we simulate epoch-dependent AI model losses (train and test) and analyze their relationship to epoch-dependent AI model coefficients. We approach the question by simulating training datasets, architectures, and hyper-parameters at a "playground" scale using the web-based neural network calculator [4]. An example of a simulation is shown in Figure 2.
Figure 2 (top) illustrates a labeled set with the classification rule described by a rule:

if $x^2 + y^2 > r^2$, then a point is outside of a circle else inside of a circle.

The AI model approximates this mathematical description by a rule:

if $2.5 * (\tanh(-0.2 * x^2 - 0.2 * y^2) + 1.7) < 0$ then a point is outside of a circle else it is inside of a circle.

Figure 2 (bottom) shows the train-test optimization curves (left) and AI model coefficient curves (right) as a function of epochs displayed on log scale. All coefficients go through a relatively large change of values around the epoch 7 with respect to their ranges of values. This change of values in coefficients is not reflected in the change of train-test loss measured via mean-squared error (MSE) since the relationship is a complex mapping from many model coefficients to one test MSE value. However, the absolute values of correlations ρ between MSE_{test} and model coefficients as a function of epochs for epochs ≥ 7 are close to 1 ($\rho(w1, MSE_{test}) = 0.91$, $\rho(w2, MSE_{test}) = 0.84$, $\rho(w3, MSE_{test}) = -0.95$, $\rho(bias, MSE_{test}) = -0.95$). In general, one can partition optimization curves to identify epoch intervals with and without oscillations (or jaggedness of a curve) and with below and above threshold error values. The epoch intervals without oscillations and below error values (e.g., $MSE_{test} < MSE_{test}(7) = 0.1$) can be modeled with curve fits, and the deviations from the curve model as indicators of trained AI model quality.

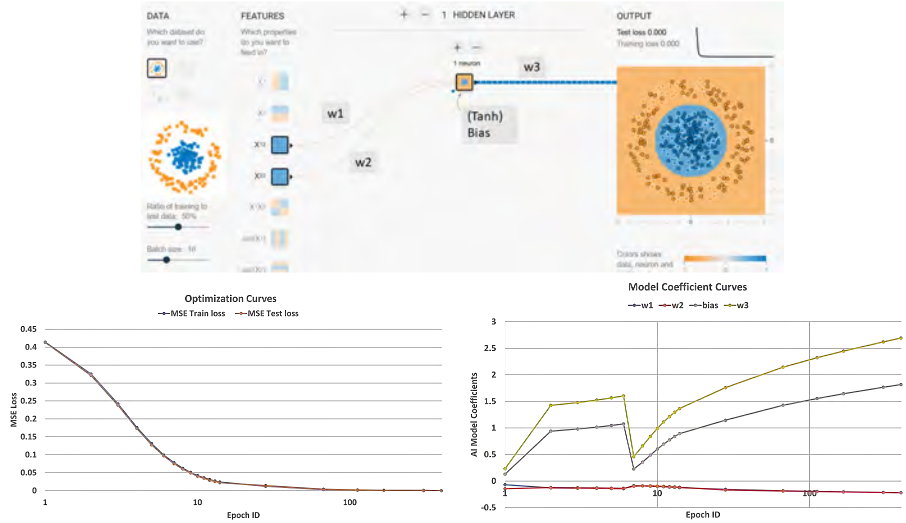


Fig. 2. Top - Simulation of a dot pattern with two clusters of labels separated by a circular boundary, (x^2, y^2) features, a single hidden layer with a single neuron model using $tanh$ activation, and a set of hyper-parameters (batch size$=10$, learning rate$=0.3$, train-test ratio $=50\%$) with w1, w2, w3, and bias coefficients. Bottom - The optimization curves (left) and the corresponding AI model coefficient curves (right).

In a similar vein, relationships between test and train curves have been used for assessing data sampling. Using the same web-based neural network calculator [4], one can analyze variable ratios of train:test random sampling for a complex spiral dot pattern. In such simulations, insufficient data sampling for training and insufficient model capacity lead to divergence of train and test curves. Such trends can be quantified, for instance, by a sum of areas under the curves (four-fold difference of the sums in our simulations over the first 14 epochs when comparing sufficient and insufficient data sampling).

These types of simulations suggest that one can derive several indicators (metrics) about convergence, stability, speed, impact of initialization, and uniformity of training and testing data from optimization curves.

Definitions of AI model metrics: The AI model characteristics are defined as sums, deltas, correlations, and extreme points as well as least-squared fits of power and exponential models to optimization curves from a varying number of data points. Equations 1-9 denote the index of each epoch as ep, number of epochs as EP, the epoch for which a model achieves the minimum error as $ep^*(M_{er})$, the window around $ep^*(M_{er})$ as $\pm\delta$, initializations as $rand$ (random) or $pretrain$ or a1/a2 graphs, execution time as T, correlation of two curves as ρ, and utilization of memory and processing power of a graphics processing unit as GPU^{mem} and GPU^{util}. In this work, we assume that the optimized error metric M_{er} per AI model is the cross entropy (CE) loss since it is widely used and supported by common AI libraries [18,1].

In Equation 5, the value of $H^{CE,fit}$ represents a predicted CE values from the first few epochs given power or exponential models for the least-squared fit approximation. In our analyses, we refer to the power model $a*x^b$ as PW and the exponential model $a*b^x$ as EXP for $a, b \in \mathbb{R}$ and $x = H^{CE}(ep)$. The metric $\Delta(fit)$ is a difference between predicted $H^{CE,fit}$ and measured H^{CE} cross entropy loss values. $\Delta(fit)$ is designed as an optimization cost function for finding the most accurate convergence prediction model ($\min \Delta(fit)$ constrained by the maximum number of measured epochs over two models $\{Model = PW, EXP\}$ and three sets of AI model optimization curves constrained by maximum of $\{10, 15, 20\}$ measured epochs.

While Equations 1 and 6 are commonly used in practice to assess model error and GPU requirements, other metric definitions are either not well-defined (e.g., stability [23]) or not mathematically defined at all. For instance, Equations 7,8, and 9 define D_{re}, D_{unif}, and D_{init} given the train and test curves as (a) representation power of an AI architecture with respect to the non-linear relationship between inputs and outputs defined by the training data D_{re}, (b) uniformity of training and testing data subsets D_{unif}, and (c) compatibility of training data and pretraining data or AI model graphs D_{init}. In this case, D_{re} in Equation 7 can be interpreted as the overall representation error of encoding training data in an AI architecture represented by a graph with a variety of nodes (modules) and node connectivity.

$$\begin{aligned} M_{er} &= \min_{ep}(H^{CE}_{test}(ep)) \\ ep^*(M_{er}) &= \underset{ep}{\text{argmin}} H^{CE}_{test}(ep) \end{aligned} \tag{1}$$

$$M_{stab} = \sum_{ep=ep^*(M_{er})-\delta}^{ep^*(M_{er})+\delta} (H^{CE}_{test}(ep) - M_{er}) \tag{2}$$

$$T(M_{er}) = ep^*(M_{er}) * \frac{1}{EP} * \sum_{i=1}^{EP} T_i \tag{3}$$

Table 2. Definition of AI model metrics

Metric name	Math symbol	Eq.
Model error	$M_{er}, ep^*(M_{er})$	1
Model stability	M_{stab}	2
Speed	$T(M_{er})$	3
Initialization gain	G_{init}	4
Predictability	$\Delta(fit)$	5
GPU utilization	$\text{GPU}^{MaxM}, \text{GPU}^{AvgU}$	6
Data-model representation	D_{re}	7
Train-test data uniformity	D_{unif}	8
Data compatibility	D_{init}	9

$$G_{init} = M_{er}^{rand} - M_{er}^{pretrain} \tag{4}$$

$$\Delta(fit) = \sum_{ep=1}^{EP} (H_{test}^{CE}(ep) - H_{test}^{CE,fit}(ep)) \tag{5}$$

$$\begin{aligned} \text{GPU}^{MaxM} &= \max_{ep}(\text{GPU}^{mem}(ep)) \\ \text{GPU}^{AvgU} &= \frac{1}{EP} * \sum_{ep=1}^{EP} \text{GPU}^{util}(ep) \end{aligned} \tag{6}$$

$$D_{re} = \sum_{ep=1}^{EP} (H_{train}^{CE}(ep) + H_{test}^{CE}(ep)) \tag{7}$$

$$D_{unif} = \rho(H_{train}^{CE}(ep), H_{test}^{CE}(ep)) \tag{8}$$

$$\begin{aligned} D_{init}(data) &= \sum_{ep=1}^{EP} (H_{test}^{CE,rand}(ep) - H_{test}^{CE,pretrain}(ep)) \\ D_{init}(graph) &= \sum_{ep=1}^{EP} (H_{test}^{CE,a1}(ep) - H_{test}^{CE,a2}(ep)) \end{aligned} \tag{9}$$

Design of a recommendation system: Given a set of example tasks shown in Table 1 and a set of derived metrics from optimization curves in Table 2, one needs to rank and recommend AI models, training data, and hyper-parameter configurations to complete a specific task. The optimization curves for deriving characteristics of AI models can be generated by in-house scripts or tools that automate sweeping a range of AI model parameters [2], [7], [11], [14], while following random, grid, simulated annealing, genetic algorithm, or Bayesian search strategies. In our study, we generated pairs of train and test optimization curves using in-house scripts (see Tables 3, 4, and 5) and leveraged NAS-Bench-101 database [22] that contains information about 5 million trained AI model architectures.

Figure 3 (right) shows an example of two pairs of optimization curves with fluctuations due to the complexity of a high-dimensional loss surface traversed during optimization using the training images illustrated in Figure 3 (left). The optimization curves illustrate the complexity of the cross entropy (CE) loss surface with respect to all contributing variables (i.e., the space is discontinuous and/or frequently flat without expected extrema, evaluations fail due to sparse discrete objective space formed by integer and categorical variables and/or numerical difficulties and hardware failures [19]). Thus, metrics characterizing optimization curves must be presented as a vector and ranked by each vector element interactively. In our recommender design, we chose a parallel coordinate graph to convey ranking and support decisions. We also focus on validating the metrics based on additional information about training datasets.

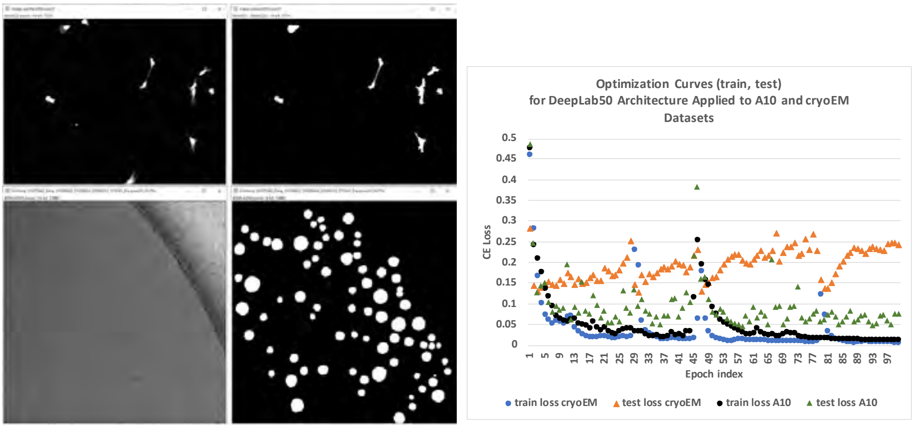


Fig. 3. Left - Examples of training image pairs (intensity, segmentation mask) for A10 dataset (top row), and cryoEM dataset (bottom row). Right - Optimization curves (train, test) for DeepLab50 AI architecture trained on A10 and cryoEM datasets.

3 Experimental Results

We divided the experimental work into (1) generating optimization curves over a range of AI model configurations, (2) validating the designed metrics based on a couple of datasets and their *prior* characterization of segmentation difficulties, and (3) describing recommendations for a reuse of trained AI models driven by use cases listed in Table 1 and applied to image segmentation and classification tasks.

Generation of optimization curves: In order to generate optimization curves for varying training datasets, AI model architectures, and hyper-parameters, we gathered five segmentation training images acquired by multiple imaging modalities (see Table 4), implemented six AI segmentation architectures by leveraging the PyTorch library [18] (see Table 5), and varied a couple of hyper-parameters (see Table 3). The model initialization using pre-trained coefficients was based on the COCO dataset [13] for object segmentation (1.5 million object instances).

The training image datasets represent optical florescent, optical bright-field, electron, cryogenic electron, and neutron imaging modalities, and are characterized in terms of the number of predicted classes (#Classes), the number of pixels (#Pixels), and the average coefficient of variation ($\overline{CV}$) over all training images as defined in Equation 10.

$$\overline{CV} = \frac{1}{N} \sum_{i=1}^{N} \frac{\sigma_i}{\mu_i} \tag{10}$$

where μ_i and σ_i are the mean and standard deviation of each intensity image in the training collection of size N images. The A10 dataset denotes fluorescently labeled optical microscopy images of A10 cells [17]. The concrete dataset came from electron microscopy of concrete samples [3]. The cryoEM dataset was prepared by the authors using cryogenic electron microscopy of lipid nanoparticles. The infer14 dataset was prepared by the authors using data-driven simulations of porous concrete samples from measured neutron images [16]. The rpe2d dataset denotes time-lapse bright-field optical microscopy images of retinal pigment epithelial (RPE) cells published in [20].

For the training runs, we chose to train each model configuration for $EP = 100$ epochs. In general, this value will vary during hyper-parameter optimization runs depending on available computational resources, the definition of model convergence error, or the use of early stopping criterion (an increment observed in CE loss values over consecutive epochs is smaller than ϵ). We also set the value $\delta = 5$ epochs in Equation 2. All computations were performed on a compute node with a Quadro Ray Tracing Texel eXtreme (RTX) 4000 GPU card and Compute Unified Device Architecture (CUDA) 11.6.

Validation of AI characteristics: We selected two training image datasets labeled as A10 and cryoEM in Table 4 for validation. Examples of training image pairs are shown in Figure 3. The A10 dataset has a high contrast ($\overline{CV} = 1.34$) while the cryoEM dataset has a low contrast ($\overline{CV} = 0.06$) and a large heterogeneity in sizes and textures. The datasets were chosen based on the assumption that segmenting images with low contrast is a much harder task than segmenting images with high contrast.

Given the assumption about segmentation difficulty, the complexity of an input-output function for cryoEM dataset is larger than the complexity of such a function for A10 dataset and hence the model utilization or the error must be higher for cryoEM. We observe the worst error over all AI models for A10 ($M_{er} = 0.0568$) to be at least twice smaller than the best error over all AI models for cryoEM ($M_{er} = 0.127$). This implies that any of the explored model capacities could not increase model utilization to accommodate the cryoEM input-output function and hence optimization errors are much higher for cryoEM. Furthermore, the heterogeneity of segments in sizes and textures in cryoEM versus A10 poses challenges on sampling for train-test subsets. Since the sampling is completely random, it is very unlikely that segments with varying sizes and textures from cryoEM will be equally represented in train-test subsets. This implies that $D_{unif} \in [0.226, 0.836]$ for cryoEM is expected to be smaller on average than $D_{unif} \in [-0.359, 0.310]$ for A10 due to the train-test gap. Ideally, the correlation of train and test CE loss curves D_{unif} should be close to one.

One can now validate AI model characteristics against expected inequalities to be satisfied by the values derived from these two datasets. The expected inequalities include model error M_{er}, uniformity of training and testing data D_{unif}, and convergence predictability $\Delta(fit)$ as shown in Equation 11. These inequalities are validated by comparing the values of M_{er} and D_{unif} in Table 5. Figure 5 shows the values in parallel coordinate plots including the values of $\Delta(fit, \text{A10})$ and $\Delta(fit, \text{CryoEM})$ on the right most vertical line denoted as $\min P(PW\_20)$. The values of $\min P(PW\_20)$ were calculated using the power model fit from the first 20 epochs. The sum of train and test optimization curves D_{re}, as well as the convergence predictability $P(PW\_20)$, quantify the sensitivity of model training to hyper-parameters (i.e., learning rate and initialization). In both A10 and cryoEM datasets, D_{re} and $P(PW\_20)$ values for 2-3 architecture types indicate epoch-specific optimization divergence (as illustrated by the scattered black points in Figure 4(right) for the A10 dataset).

$$
\begin{aligned}
M_{er}(\text{A10}) &< M_{er}(\text{CryoEM}) \\
D_{unif}(\text{A10}) &> D_{unif}(\text{CryoEM}) \\
\Delta(fit, \text{A10}) &< \Delta(fit, \text{CryoEM})
\end{aligned}
\tag{11}
$$

Table 3. Explored hyper-parameters in AI model configurations

Hyper-parameters	Values
Initialization	Random COCO pre-trained
Learning Rate	10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}
Optimizer	Adam
Optimization criterion	Cross entropy loss
Epochs	100
Batch size	2
Class balance method	Weighting by class proportion
Augmentation	None
Train-Test split	80 : 20

Reuse of trained AI models for image segmentation: When the task focus is on convergence predictability $\Delta(fit)$, an AI model configuration in Figure 4 (left) shows better convergence for power model with 20 epochs than any configuration in Figure 4 (right). A large divergence from the predicted optimization curves typically indicates that (a) it is not sufficient to predict the model training convergence using a few initial epochs (10, 15, or 20 epochs), (b) training and testing subsets might not have been drawn from the same distribution, and (c) the COCO dataset used for pretraining the AI model might not be compatible with the domain training dataset, and, hence, the test CE loss values vary a lot during the first few epochs. This is undesirable for researchers

Table 4. Training datasets. OF - optical fluorescent, EM - electron microscopy, OB - optical bright field, NI - neutron imaging

Dataset	Modality	#Classes	#Pixels [MPix]	$\overline{CV}$
A10	OF	2	5.79	1.34
concrete	EM	4	71.7	0.31
cryoEM	EM	2	117.44	0.06
infer14	NI	9	125.9	0.24
rpe2d	OB	2	53.22	0.84

Table 5. Summary of AI model characteristics per model architecture and per dataset where the models were optimized over the learning rates and pretraining options listed in Table 3.

Architecture	A10 M_{er}	A10 D_{unif}	cryoEM M_{er}	CryoEM D_{unif}
DeepLab101	0.0528	0.3513	0.1271	-0.0093
DeepLab50	0.0451	0.8356	0.1284	-0.2515
LR-ASPP	0.04	0.7978	0.1435	-0.3585
MobileNetV3	0.0568	0.5794	0.1602	0.3092
ResNet101	0.042	0.2256	0.1379	-0.0867
ResNet50	0.045	0.391	0.1369	-0.2269

who would like to predict how many more epochs to run on the existing model while targeting a low test CE loss value. On the other hand, the configuration in Figure 4(right) achieves lower CE error M_{er} (vertically lowest black point) than the configuration in Figure 4(left) for the same dataset A10 and the same DeepLab50 AI architecture.

When the task focus is on gain from pretraining G_{init}, the values are less than zero for all datasets except the concrete dataset listed in Table 4. These values indicate that the objects in the COCO dataset are significantly different from the objects annotated in the five scientific microscopy datasets, and the pre-training on COCO does not yield better model accuracy.

When the task focus is on model stability M_{stab}, Figure 6 illustrates how stable each optimized AI model is over the configurations listed in Tables 3, 4, and 5. If the test CE loss curve is close to constant within the neighborhood of $\delta = 5$ epochs, then the value of M_{stab}, as defined in Equation 2, is small indicating model stability. Based on Figure 6, all model architectures for the rpe2d dataset yielded highly stable, trained models, while the stability of trained models for the infer14 dataset was low and varied depending on a model architecture.

Reuse of trained AI models for image classification: We analyzed train, validation, and test accuracies obtained using CIFAR-10 training dataset [12] and a network architecture search (NAS) published in NAS-Bench-101 database [22]. NAS-Bench 101 contains information about final and halfway accuracies obtained by searching over ResNet and Inception like architectures for 5 million graphs. Accuracies over three repeated training runs were averaged and plotted for the Inception-like and ResNet-like architectures in Figure 7 at halfway and final epochs and for four epoch budgets $\{4, 12, 36, 108\}$. The standard deviation of average values over all accuracies was 0.038 and 0.042. Since two points in each optimization curve were not very informative, we

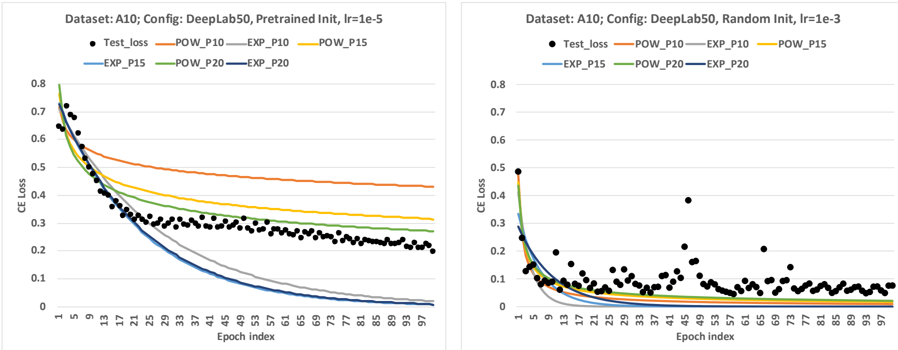
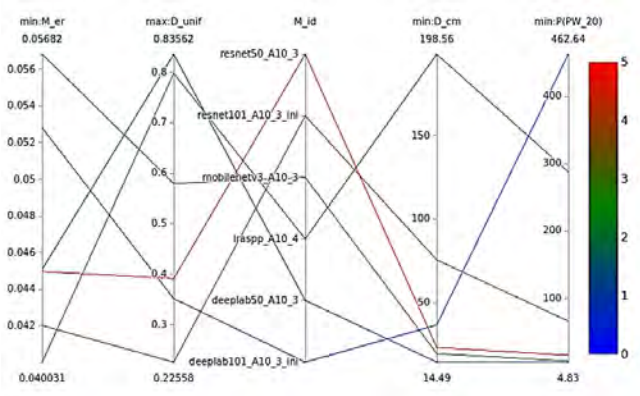


Fig. 4. Predictions of training model convergence from A10 dataset for two configurations. Left configuration: (DeepLab50, COCO pre-trained initialization, learning rate: 1e-5). Right configuration: (DeepLab50, random initialization, learning rate: 1e-3). Black dots are the measured test CE loss values. Color-coded curves are predictions for a set of fitted model parameters.

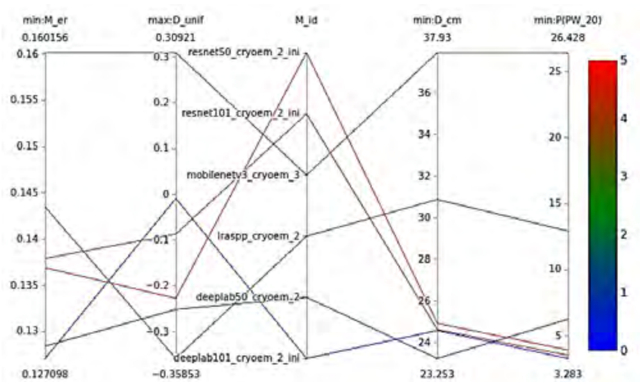
analyzed all points combined from the epoch budgets per Inception-like and ResNet-like architecture graphs. The correlations of train, validation, and test contours D_{unif} were larger than 0.999 indicating uniformity of train-test splits of CIFAR-10 dataset. Regarding convergence $\Delta(fit)$, a power model is more accurate than an exponential model. We observed the following inequality $\Delta(fit, ResNet) < \Delta(fit, Inception)$ suggesting that one could predict convergence of ResNet-like architecture more accurately than convergence of Inception-like architecture. On the other hand, the Inception-like architecture reaches higher accuracy values faster with epoch numbers than the ResNet-like architecture as it can be documented with a positive sum of deltas $D_{init}(\text{graph}) = 1.45$, where $a1 =$Inception and $a2 =$ResNet in Equation 9.

4 Discussion

The practical value of each metric is task dependent for the use cases listed in Table 1. For instance, if the task is model portability to a new hardware or model-training reproducibility, then knowing maximum required GPU memory and GPU utilization would be very valuable. It is frequent in biology to apply transfer learning and reuse trained AI models due to a limited size of annotated image datasets in scientific experiments. Knowing what architectures, datasets, and hyper-parameters were explored can save computational time when cell type, tissue preparation, or imaging modalities differ between trained AI models and a reuse application. We recommend minimizing the number of variables between original trained model and reuse configurations if possible because (a) AI-based modeling entangles all variables in non-linear way and (b) optimization curves provide a limited information content about the changes inside of a model (as stated in the list of challenges in Section 1). Nevertheless, the metrics are useful if one has some apriori knowledge about the training datasets, architecture types, and the complexity of predicting image segmentation and the reuse of trained AI models is applied in the configuration proximity of shared AI models.



(a)



(b)

Fig. 5. Parallel coordinate plots for A10 (top) and cryoEM (bottom) datasets. The plots are intended to support decisions about which AI model architecture is the most accurate for the image segmentation tasks.

From the perspective of defining metrics, our goal is not to invent them from scratch but rather to define them in consistent mathematical and computational ways as opposed to the current variations of subsets of presented metrics in practice. Consistent metric definitions enable their use in model cards and improved reuse of shared AI models. Nonetheless, more metrics will need to be designed and defined to address a broad range of AI model reuses.

5 Summary

This work presented the problem of learning from a set of optimization curves, defining metrics derived from the curves for model cards [15], and reusing trained AI models by leveraging accompanying metrics. The output quantitative AI model metrics serve

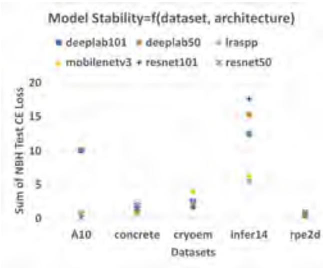


Fig. 6. Model stability of the most accurate AI model per architecture and per dataset

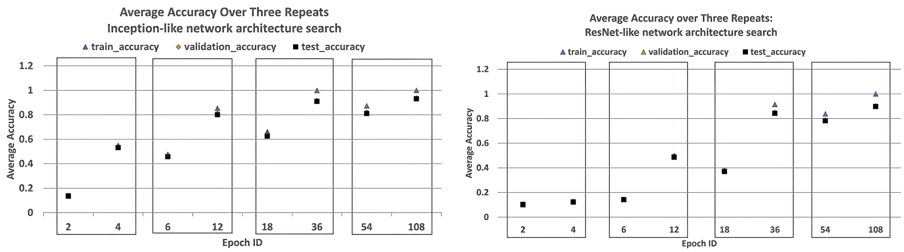


Fig. 7. Accuracies extracted from NAS-Bench101 database for Inception like (left) and ResNet like (right) architectures

multiple purposes: as entries under *Metrics* in the AI model card definition [15] and as inputs to ranking of AI models according to a variety of objectives (e.g., model accuracy refinement, model architecture recommendation).

The designed metrics were evaluated on image segmentation tasks applied to image datasets selected based on their estimated segmentation level of difficulty. Our main results demonstrated use cases of scientists reusing pre-trained AI models for the purposes of (1) improving model accuracy by further training/optimization of model hyper-parameters constrained by computational resources (convergence predictability), (2) selecting from pretrained models (data initialization gain), (3) finding stable models (model stability), and (4) choosing optimal graphs for training data (graph initialization gain).

The impact of sharing AI models with presented metrics is significant for principal investigators limited by their grant budgets and small research labs limited by their own computational resources or the cost of cloud resources. A higher reuse of shared AI models can save not only cost and time to researchers but also advance their scientific goals more efficiently. The cost of achieving a higher reuse of AI models is the extra summarization of optimization sessions using transparent metrics and sharing them in AI model cards. In the future, we plan to explore how to replace relative with absolute comparisons of AI model metrics when recommending trained AI models.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015). https://doi.org/10.5281/zenodo.5898685, https://www.tensorflow.org/, software available from tensorflow.org
2. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework (2019)
3. Bajcsy, P., Feldman, S., Majurski, M., Snyder, K., Brady, M.: Approaches to training ai-based multi-class semantic image segmentation. Journal of Microscopy **279(2)**, 98–113 (2020). https://doi.org/http://dx.doi.org/10.1111/jmi.12906, https://pubmed.ncbi.nlm.nih.gov/32406521/
4. Bajcsy, P., Schaub, N.J., Majurski, M.: Designing trojan detectors in neural networks using interactive simulations. Applied Sciences **11**(4) (2021). https://doi.org/10.3390/app11041865, https://www.mdpi.com/2076-3417/11/4/1865
5. Community: Open neural network exchange (ONNX). https://onnx.ai/ (2022), https://onnx.ai/
6. Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J.W., Wallach, H., III, H.D., Crawford, K.: Datasheets for datasets. arXiv 1803.09010 (2021)
7. Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J.E., Sculley, D. (eds.): Google Vizier: A Service for Black-Box Optimization (2017), http://www.kdd.org/kdd2017/papers/view/google-vizier-a-service-for-black-box-optimization
8. Haibe-Kains, B., Adam, G., Hosny, A., Khodakarami, F., Waldron, L., Wang, B., McIntosh, C., Goldenberg, A., Kundaje, A., Greene, C., Broderick, T., Hoffman, M., Leek, J., Korthauer, K., Huber, W., Brazma, A., Pineau, J., Tibshirani, R., Hastie, T., Ioannidis, J., Quackenbush, J., Aerts, H.: Transparency and reproducibility in artificial intelligence. Nature **586(7829)**, E14–E16 (2020). https://doi.org/10.1038/s41586-020-2766-y, https://aclanthology.org/Q18-1041
9. Holland, S., Hosny, A., Newman, S., Joseph, J., Chmielinski, K.: The dataset nutrition label: A framework to drive higher data quality standards. arXiv 1805.03677 (2018)
10. IARPA: Trojans in Artificial Intelligence (TrojAI) (2020), https://pages.nist.gov/trojai/, https://www.iarpa.gov/index.php/research-programs/trojai
11. Koch, P., Golovidov, O., Gardner, S., Wujek, B., Griffin, J., Xu, Y.: Autotune. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Jul 2018). https://doi.org/10.1145/3219819.3219837, http://dx.doi.org/10.1145/3219819.3219837
12. Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 (canadian institute for advanced research) http://www.cs.toronto.edu/~kriz/cifar.html
13. Lin, T.Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft COCO: Common objects in context. arXiv 1405.0312 (2014), http://arxiv.org/abs/1405.0312
14. Long, J., Shelhamer, E., Darrell, T.: Experiment manager. https://www.mathworks.com/help/deeplearning/ref/experimentmanager-app.html (2022)

15. Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I.D., Gebru, T.: Model cards for model reporting. Proceedings of the Conference on Fairness, Accountability, and Transparency (Jan 2019). https://doi.org/10.1145/3287560.3287596, http://dx.doi.org/10.1145/3287560.3287596
16. NIST: Data-driven simulations of measured neutron interferometric microscopy images (2022), https://www.nist.gov/programs-projects/interferometry-infer-neutron-interferometric-microscopy-small-forces-and
17. NIST: Fluorescent microscopy images of A-10 rat smooth muscle cells and NIH-3T3 mouse fibro-blasts. https://isg.nist.gov/deepzoomweb/data/dissemination (2022), https://isg.nist.gov/deepzoomweb/data/dissemination
18. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
19. Ranjit, M., Ganapathy, G., Sridhar, K., Arumugham, V.: Efficient deep learning hyperparameter tuning using cloud infrastructure: Intelligent distributed hyperparameter tuning with bayesian optimization in the cloud. In: 2019 IEEE 12th International Conference on Cloud Computing (CLOUD). pp. 520–522. IEEE Computer Society, Los Alamitos, CA, USA (jul 2019). https://doi.org/10.1109/CLOUD.2019.00097, https://doi.ieeecomputersociety.org/10.1109/CLOUD.2019.00097
20. Schaub, N., Hotaling, N., Manescu, P., Padi, S., Wan, Q., Sharma, R., George, A., Chalfoun, J., Simon, M., Ouladi, M., Simon, C.J., Bajcsy, P., K., B.: Deep learning predicts function of live retinal pigment epithelium from quantitative microscopy. J Clin Invest. **130(2)**, 1010–1023 (2020). https://doi.org/10.1172/JCI131187, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6994191/
21. Yang, K., Stoyanovich, J., Asudeh, A., Howe, B., Jagadish, H., Miklau, G.: A nutritional label for rankings. Proceedings of the 2018 International Conference on Management of Data (May 2018). https://doi.org/10.1145/3183713.3193568, http://dx.doi.org/10.1145/3183713.3193568
22. Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., Hutter, F.: NAS-bench-101: Towards reproducible neural architecture search. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 7105–7114. PMLR, Long Beach, California, USA (09–15 Jun 2019), http://proceedings.mlr.press/v97/ying19a.html
23. You, S., Zhao, Y., Mandich, M., Cui, Y., Li, H., Xiao, H., Fabus, S., Su, Y., Liu, Y., Yuan, H., Jiang, H., Tan, J., Zhang, Y.: A review on artificial intelligence for grid stability assessment. In: 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm). pp. 1–6 (2020). https://doi.org/10.1109/SmartGridComm47815.2020.9302990