# PriRecT: Privacy-preserving Job Recommendation Tool for GPU Sharing

Aritra Ray
*Duke University*
Durham, NC, USA
aritra.ray@duke.edu

Zhaobo Zhang
*Futurewei Technologies*
Santa Clara, CA, USA
zzhang1@futurewei.com

Ying Xiong
*Futurewei Technologies*
Santa Clara, CA, USA
yxiong@futurewei.com

Krishnendu Chakrabarty
*Duke University*
Durham, NC, USA
krish@ee.duke.edu

***Abstract*—Machine Learning (ML) jobs significantly benefit when trained on abundant GPU resources. It leads to resource contention when several ML training jobs are scheduled concurrently on a single GPU in the compute cluster. A job's performance is susceptible to its competitor's task on a single GPU. We, in this paper, propose PriRecT, a novel ML job recommendation tool that preserves user privacy for scheduling ML training jobs in the GPU compute cluster. We perform workload characterization for several ML training scripts, and the Futurewei mini-ML Workload Dataset is released publicly [1]. We build a knowledge base of inter and intra-cluster task interference for GPU sharing through a clustering-based approach. For scheduling purposes, PriRecT blinds the user-sensitive information and assigns the job to an existing cluster. Based on clustering results, PriRecT recommends jobs that should run concurrently on a single GPU to minimize task interference and additionally assigns an uncertainty score to account for job variations in the recommendation.**

***Index Terms*—*Futurewei mini-ML Workload Dataset*, Clustering, GPU Sharing, Task Interference**

## I. Introduction

Deep Neural Network (DNN) training is resource-intensive and time-consuming, even on GPUs. Therefore, enterprises build large GPU compute clusters wherein these jobs are to be deployed onto hardware accelerators [2]. Given the costs of these computing clusters, enterprises encourage multi-tenancy. Concurrent scheduling of ML training jobs on the same GPU is mostly managed by traditional schedulers like Kubernetes [3], or through exploiting DNN job characteristics [4]. DNN cluster schedulers are mostly designed to improve one or multiple objectives regarding GPU scheduling. Gandiva [5] focuses on improving cluster utilization, Tiresias [6] addresses job completion time requirements to improve system throughput, while Themis [7] focuses on scheduling fairness. Some also explore performance heterogeneity to accentuate fulfilling compute cluster requirements [8] [9].

Sensitive user metadata like model architecture, the dataset used, and hyperparameters like epochs and batch size are sometimes considered influential in minimizing task interference for GPU sharing. Therefore in ML workload characterization, and thereby for its scheduling decision, sensitive metadata plays its part in intelligent decision-making. DNN schedulers for GPU compute cluster [4] [10] [5] [7] does not take into account user privacy while making scheduling decisions. We propose PriRecT, an intelligent privacy-preserving ML job recommendation tool that refrains from collecting sensitive user data for scheduling decisions while helping minimize task interference for GPU sharing.

PriRecT is based on the design principle of clustering ML workloads and building a knowledge base of inter and intra-cluster task interference. For workload characterization, we built the *Futurewei mini-ML Workload Dataset* which characterizes five ML training jobs for varied hyperparameters and model architectures across 66 different attributes. The dataset is released publicly for open access [1]. We propose ML techniques to expand the dataset for selected features through synthetic data generation. We build a knowledge base of intra- and inter-cluster task interference based on clustering results. Task interference is measured as *individual slowdown* and *packing saving* metrics. Individual slowdown refers to the percent of added time required to complete the job compared to when it was deployed on the same GPU individually. Packing saving refers to the percent time saved by executing two jobs concurrently compared to them being deployed sequentially. For a given user-defined ML job, the tool runs the job for one epoch in the compute cluster and assigns it to a cluster based on the privacy-preserving metadata. We also assign uncertainty scores to the recommendations on which two jobs be deployed concurrently based on the euclidean squared distance of the job from its cluster center. Our experiments show promising results as we can identify clusters that ensure high packing saving with the least individual slowdown. The key contributions of our work are as follows:

- We characterize ML workload in *Futurewei mini-ML Workload Dataset* for five ML training jobs over 66 metrics and released publicly. This dataset can have widespread utility ranging from designing intelligent GPU schedulers to monitoring network parameters and memory usage for ML jobs and several others.
- We propose a clustering-based approach to determine task interference for concurrent scheduling of ML training jobs in a single GPU in the compute cluster. The designed tool preserves user privacy by refraining from accessing sensitive metadata and recommends which jobs must be deployed concurrently in the GPU compute cluster along with an uncertainty score.

2022 IEEE Cloud Summit | 978-1-6654-5113-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/CLOUDSUMMIT54781.2022.00021

Section II delves deeper into design methodology, while the evaluation results are presented in Section III. Section IV concludes the paper.

## II. Design Methodology

Volumes of user-defined ML jobs are requested by users for training purposes in GPU compute cluster. These ML training jobs require high GPU utilization and are *GPU-affine*. As training ML jobs require several hours, users are always concerned about the *Job Completion Time* of the task. Multi-tenancy of user-defined ML training jobs is encouraged to acknowledge the costs of hardware accelerators. Deploying two or more jobs on the same GPU concurrently instead of sequentially saves time, which we refer to as packing saving. However, sharing a GPU-affine job with another, termed the *competitor job*, has its downfalls. When two GPU-affine tasks share a GPU concurrently, they cause interference, resulting in an individual slowdown of each job. Depending on the competing job, the slowdown and packing saving percentages vary. Minimization of slowdown and maximization of packing savings remains our sole objective.

We embark on investigating task interference for GPU sharing through characterization of ML workloads for an array of various categories of tasks, which is one of our key contributions. Next, we delve into exploratory data analytics (EDA) to assess the importance of the attributes over our two target variables, namely, *Maximum GPU utilization percent*, and *Maximum GPU memory allocation percent*. To preserve the user-privacy, we do away with sensitive metadata like dataset, the number of epochs, and batch size. Before moving on to ML techniques to analyze task interference for GPU sharing, we perform a design exploration of synthetic data generation techniques to upscale the dataset concerning the predictor and outcome variables. We also analyze the quality of the generated synthetic samples and quantify uncertainties in the synthetic data. Following on, clustering helps group the tasks. We hypothesize that owing to inter-cluster dissimilarity and intra-cluster similarity, competing tasks from any two clusters will show similar behavior.

Interference analysis of clustered tasks leads us to the design of our privacy-preserving AI job recommendation tool. For any submitted job, the tool can execute the job for one epoch to assess the predictor attributes and assign it to a pre-existing cluster. Given the knowledge base for inter and intra-cluster task interference, the tool can recommend a competitor job that would help minimize individual job slowdown and maximize packing saving. While addressing this min-max optimization problem, the tool can also come up with an uncertainty score, accounting for the distance of the task from its assigned cluster's center. We discuss the same in detail as follows:

- **AI Workload Characterization**: Every ML model training job has specific characteristics over GPU and CPU utilization requirements, network parameters, memory/disk usage parameters, and impacts several system parameters according to the user-defined task. To capture these variations for standard ML architectures over various datasets and across varied hyper-parameters, we run several ML scripts and document our findings in *Futurewei mini-ML Workload Dataset* [1]. In brief, we characterize ML training jobs from five categories: image classification, image segmentation, generative adversarial networks, reinforcement learning, and sentiment analysis from the natural language processing domain. We characterize over 66 different attributes for 49 training instances.
- **Exploratory Data Analysis**: To determine the most critical features on task interference for GPU sharing, we perform feature selection on the *Futurewei mini-ML Workload Dataset*. We delve into the statistical correlation of predictors to target attributes using the coefficient of determination. We also explore random forests, ridge regression, and lasso regression as feature selection tools.
- **Synthetic Dataset Generation**: Resorting to ML-based task interference prediction for GPU sharing would require the dataset to be sufficiently large. To bolster the size of the dataset, we generate synthetic data from our collected ground truth. We generated synthetic data only for selected predictor(s) and outcome variables. For quality assessment of the generated synthetic data, we re-generate the collected data while having the generated synthetic data as pseudo-ground truth. We explore different interpolation methods, namely linear, barycentric, krogh, pchip, cubicspline, and spline of order two, for the same according to varying batch sizes.

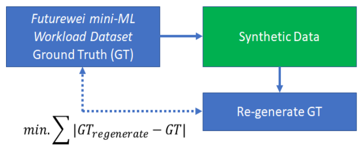
Futurewei mini-ML Workload Dataset Ground Truth (GT)
Synthetic Data
Re-generate GT
$min. \sum |GT_{regenerate} - GT|$

Fig. 1. Synthetic Data Generation Methodology

- **Predicting Task Interference and Recommending AI Jobs for GPU Sharing**: We perform clustering based on the privacy-preserving metadata to group similar tasks. Then, we build a knowledge base of individual slowdown and packing saving for inter and intra-cluster tasks deployed concurrently. For a given user-defined task, the tool executes it for one epoch, collects the required metadata for clustering, and recommends a competitor job based on the knowledge base, to address the min-max optimization. The tool keeps a tab on the distance of the task from the cluster center as an indicative measure of the uncertainty introduced in the recommendation system.

## III. Evaluations

### A. Data Collection

This section documents the *Futurewei mini-ML Workload Dataset* developed to bolster ML workload characterization and garner insights into task interference for GPU sharing purposes. The dataset consists of 49 instances, capturing 66 parameters, as detailed in Table I. The ML scripts include

TABLE I
ATTRIBUTE DESCRIPTION OF *Futurewei mini-ML Workload Dataset*

| Attribute ID | Attribute | Attribute Description |
|---|---|---|
| 1 | *ID* | Task ID |
| 2 | *Model* | Architecture of ML model used |
| 3 | *Dataset* | Dataset used for the task |
| 4 | *Epoch* | Number of iterations for training the ML model |
| 5 | *BS* | Batch Size |
| 6 | *Run Env* | Runtime Environment |
| 7 | *CPU Util % ran* | CPU Utilization percent range |
| 8 | *Max CPU Util %* | Maximum CPU Utilization percent |
| 9 | *Min CPU Util %* | Minimum CPU Utilization percent |
| 10 | *GPU Util % ran* | GPU Utilization percent range |
| 11 | *MGUP* | Maximum GPU Utilization percent |
| 12 | *Min GPU Util %* | Minimum GPU Utilization percent |
| 13 | *Sys Mem Util % ran* | System Memory Utilization percent range |
| 14 | *Max Sys Mem Util %* | Maximum System Memory Utilization percent |
| 15 | *Min Sys Mem Util %* | Minimum System Memory Utilization percent |
| 16 | *MaPMiU(non swap)* | Maximum Process Memory in Use (non swap) |
| 17 | *PMiU(non swap) %* | Process Memory in Use (non swap) percent |
| 18 | *CPU Thds* | CPU Threads |
| 19 | *GPU Temp ran* | GPU Temperature range |
| 20 | *Max GPU Temp* | Maximum GPU Temperature |
| 21 | *Min GPU Temp* | Minimum GPU Temperature |
| 22 | *GTSAMPR* | GPU Time spent accessing memory percent range |
| 23 | *MaGTSAMP* | Maximum GPU Time spent accessing memory percent |
| 24 | *MiGTSAMP* | Minimum GPU Time spent accessing memory percent |
| 25 | *MGMAP* | Maximum GPU Memory Allocated percent |
| 26 | *GPUPR* | GPU Power Usage percent range |
| 27 | *MaGPUPR* | Maximum GPU Power Usage percent |
| 28 | *MiGPUPR* | Minimum GPU Power Usage percent |
| 29 | *SCT user* | In user mode, execution time of normal processes |
| 30 | *SCT nice* | In user mode, execution time of priority processes |
| 31 | *SCT sys* | In kernel mode, execution time of processes |
| 32 | *SCT idle* | System idle time |
| 33 | *SCT iowait* | Input-output completion time (not accounted in idle time counter) |
| 34 | *SCT irq* | Time to service hardware interrupts |
| 35 | *SCT softirq* | Time to service software interrupts |
| 36 | *SCT steal* | Time consumed by operating systems (OS) running in virtualized environment |
| 37 | *SCT guest* | Time consumed to run a virtual CPU for guest OS under the control of the Linux kernel |
| 38 | *SCT guest nice* | Time consumed running a virtual CPU for guest OS while executing priority processes executing in user mode |
| 39 | *Cor in Sys* | Number of Cores in the System |
| 40 | *CS ctx switches* | Number of context switches since boot |
| 41 | *CS interrupts* | Number of interrupts since boot |
| 42 | *CS soft interrupts* | Number of software interrupts since boot |
| 43 | *CS syscalls* | Number of system calls since boot. Always set to 0 in Ubuntu. |
| 44 | *SMU total* | Total available physical memory (excluding swap) |
| 45 | *SMU available* | Memory that can be assigned to processes without any system swap |
| 46 | *SMU percent* | Percent of memory used |
| 47 | *SMU used* | Memory used |
| 48 | *SMU free* | Available memory |
| 49 | *SMU active* | Memory currently in use (or very recently used) |
| 50 | *SMU inactive* | Memory marked as unused |
| 51 | *SMU buffers* | Cache data like file system, storing metadata |
| 52 | *SMU cached* | Cached data |
| 53 | *SMU shared* | Memory that may be accessed by multiple processes |
| 54 | *SMU slab* | In-kernel data structures cache |
| 55 | *DU total* | Total disk space |
| 56 | *DU used* | Used disk space |
| 57 | *DU free* | Free disk space |
| 58 | *DU percent* | Disk usage percent |
| 59 | *NIOB sent* | Number of bytes sent |
| 60 | *NIOB received* | Number of bytes received |
| 61 | *NIOP sent* | Number of packets sent |
| 62 | *NIOP received* | Number of packets received |
| 63 | *NIO errin* | Total number of errors while receiving I/O signals |
| 64 | *NIO errout* | Total number of errors while sending I/O signals |
| 65 | *NIO dropin* | Total number of dropped incoming packets |
| 66 | *NIO dropout* | Number of dropped outgoing packets |

TABLE II
ILLUSTRATION EVALUATION OF SOME SYNTHETIC DATA GENERATION TECHNIQUES

| ML Model/ Task | Dataset | Number of Synthetic Data Points Generated | Interpolation Technique | MGUP | | MGMAP | | MaGTSAMP | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | RMSE | MAE | RMSE | MAE | RMSE | MAE |
| MobileNet | GTSRB | 16 | Linear | 2.25 | 1.99 | 5.20 | 3.73 | 2.18 | 1.95 |
| NasnetMobile | | | | 4.64 | 4.16 | 8.48 | 6.28 | 3.79 | 3.25 |
| EfficientNetV2 | | | | 7.18 | 6.25 | 10.69 | 8.39 | 8.24 | 7.40 |
| ResNet50 | | | | 3.65 | 3.25 | 4.42 | 2.65 | 2.97 | 2.47 |
| InceptionV3 | | | | 1.98 | 1.82 | 4.43 | 2.65 | 3.17 | 2.44 |
| Image Segmentation | Oxford-IIIT | | | 3.98 | 3.65 | 16.27 | 11.71 | 3.038 | 2.66 |
| Reinforcement Learning | Cart-Pole | | | 0.59 | 0.48 | 0.66 | 0.001 | 0.0 | 0.0 |
| DCGAN | MNIST | 17 | | 4.77 | 4.4 | 1.47 | 1.24 | 1.59 | 1.40 |
| MobileNet | GTSRB | 16 | Barycentric | 1.68 | 1.06 | 1.89 | 1.2 | 1.39 | 0.88 |
| NasnetMobile | | | | 2.73 | 1.73 | 7.95 | 5.03 | 2.99 | 1.89 |
| EfficientNetV2 | | | | 6.53 | 4.13 | 7.95 | 5.03 | 5.23 | 3.30 |
| ResNet50 | | | | 2.35 | 1.49 | 1.5E-11 | 9.5E12 | 1.30 | 0.82 |
| InceptionV3 | | | | 0.63 | 0.4 | 7.4E-11 | 3.8E-11 | 1.88 | 1.84 |
| Image Segmentation | Oxford-IIIT | | | 3.20 | 10.28 | 15.91 | 10.06 | 2.73 | 1.73 |
| Reinforcement Learning | Cart-Pole | | | 0.29 | 0.18 | 7.7E-13 | 3.8E-13 | 0.0 | 0.0 |
| DCGAN | MNIST | 17 | | 5.9E-10 | 4.1E-10 | 9.5E-11 | 6.7E-11 | 2.5E-10 | 1.8E-10 |
| MobileNet | GTSRB | 16 | Krogh | 1.68 | 1.06 | 1.89 | 1.2 | 1.39 | 0.88 |
| NasnetMobile | | | | 2.73 | 1.73 | 7.95 | 5.03 | 2.99 | 1.89 |
| EfficientNetV2 | | | | 6.53 | 4.13 | 7.95 | 5.03 | 5.23 | 3.30 |
| ResNet50 | | | | 2.35 | 1.49 | 1.7E-11 | 8.07E-12 | 1.30 | 0.82 |
| InceptionV3 | | | | 0.63 | 0.4 | 3.2E-11 | 1.7E-11 | 2.90 | 1.84 |
| Image Segmentation | Oxford-IIIT | | | 3.20 | 2.02 | 15.91 | 10.06 | 2.73 | 1.73 |
| Reinforcement Learning | Cart-Pole | | | 0.29 | 0.18 | 1.59E-12 | 1.01E-12 | 0.0 | 0.0 |
| DCGAN | MNIST | 17 | | 1.9E-10 | 1.3E-10 | 9.9E-11 | 7.0E-11 | 7.7E-11 | 5.4E-11 |
| MobileNet | GTSRB | 16 | Pchip | 1.64 | 1.05 | 2.04 | 1.42 | 1.40 | 0.92 |
| NasnetMobile | | | | 2.79 | 1.80 | 7.97 | 5.08 | 3.03 | 1.92 |
| EfficientNetV2 | | | | 6.39 | 4.31 | 8.11 | 5.25 | 5.04 | 3.69 |
| ResNet50 | | | | 2.41 | 1.54 | 0.48 | 0.30 | 1.34 | 0.88 |
| InceptionV3 | | | | 0.69 | 0.48 | 0.48 | 0.30 | 2.94 | 1.88 |
| Image Segmentation | Oxford-IIIT | | | 3.21 | 2.10 | 15.76 | 9.98 | 2.73 | 1.77 |
| Reinforcement Learning | Cart-Pole | | | 0.31 | 0.20 | 0.0003 | 0.0002 | 0.0 | 0.0 |
| DCGAN | MNIST | 17 | | 0.46 | 0.44 | 0.13 | 0.10 | 0.10 | 0.09 |
| MobileNet | GTSRB | 16 | Cubicspline | 1.64 | 1.05 | 1.90 | 1.20 | 1.39 | 0.87 |
| NasnetMobile | | | | 2.74 | 1.73 | 7.95 | 5.03 | 3.00 | 1.89 |
| EfficientNetV2 | | | | 6.52 | 4.12 | 7.96 | 5.03 | 5.19 | 3.28 |
| ResNet50 | | | | 2.36 | 1.49 | 0.01 | 0.007 | 1.31 | 0.82 |
| InceptionV3 | | | | 0.63 | 0.40 | 0.01 | 0.007 | 2.91 | 1.84 |
| Image Segmentation | Oxford-IIIT | | | 3.20 | 2.02 | 15.90 | 10.05 | 2.73 | 1.72 |
| Reinforcement Learning | Cart-Pole | | | 0.29 | 0.18 | 1.1E-5 | 5.1E-6 | 0.0 | 0.0 |
| DCGAN | MNIST | 17 | | 4.4E-14 | 3.3E-14 | 8.05E-15 | 6.4E-15 | 3.7E-15 | 2.7E-15 |
| MobileNet | GTSRB | 16 | Spline, order=2 | 1.55 | 1.03 | 4.62 | 3.66 | 2.53 | 2.41 |
| NasnetMobile | | | | 4.61 | 4.12 | 8.66 | 5.89 | 4.09 | 3.17 |
| EfficientNetV2 | | | | 5.38 | 4.57 | 10.16 | 7.43 | 3.94 | 3.26 |
| ResNet50 | | | | 4.08 | 3.50 | 2.72 | 2.23 | 2.98 | 3.12 |
| InceptionV3 | | | | 2.82 | 2.65 | 2.72 | 2.23 | 18.633 | 2.64 |
| Image Segmentation | Oxford-IIIT | | | 4.35 | 4.09 | 15.44 | 11.48 | 3.30 | 3.08 |
| Reinforcement Learning | Cart-Pole | | | 0.66 | 0.49 | 0.004 | 0.004 | 0.0 | 0.0 |
| DCGAN | MNIST | 17 | | 2.93 | 2.90 | 2.40 | 2.38 | 2.52 | 2.50 |

image classification, image segmentation, generative adversarial networks, natural language processing, and Reinforcement Learning (RL). Referring to Table I, Attribute ID 1 to 6 refers to the metadata of the ML scripts. Attribute ID 7 to 28 is collected while monitoring the script's system memory utilization, GPU memory utilization, CPU threads, and several others through an MLOps platform, Wandb [11]. Wandb provided a resolution of 30 secs to monitor these attributes. Attribute ID 29 to 66 is collected using a Python library, psutil, which tracks various resource utilization parameters in the system, ranging from memory usage, interrupts, system calls, context switches, network traffic statistics, and several others.

In regards to image classification, standard ML architectures were used, like ResNet-50 [12], EfficientNetv2 [13], NasnetMobile [14], MobileNet [15], and Inceptionv3 [16] for classifying traffic signs in German Traffic Sign Recognition Benchmark (GTSRB) dataset [17]. GTSRB dataset consists of 39209 RGB training images spread across 43 classes, with 12630 testing images. For image segmentation, data points were collected across various hyper-parameters while segmenting the Oxford-IIIT pet dataset [18] (version-3.2.0). It consists of 37 different categories of pets with approximately 200 RGB images per class. The standard architecture of Deep Convolutional Generative Adversarial Network (DCGAN) was

TABLE III
INTER CLUSTER TASK INTERFERENCE PREDICTION

| **Task** | **Cluster ID** | **Distance from Centroid** | **Time for Completion (real) (Secs)** | **Individual Slow Down** | **Packing Saving** |
|---|---|---|---|---|---|
| Inceptionv3 (*BS*=20, *Epoch*=5) + Small BERT L2-H128-A2 (*BS*=256, *Epoch*=1) | 0,1 | 115.14, 150.17 | 338.8, 294.5 | 7.14%, 15.99% | 40.5% |
| Inceptionv3 (*BS*=20, *Epoch*=5) + RL (*BS*=1024, *Epoch*=5000) | 0,1 | 115.14, 211.28 | 336.8, 306.1 | 6.51%, 25.7% | 39.8% |
| NasnetMobile (*BS*=562, *Epoch*=5) + Small BERT L2-H128-A2 (*BS*=256, *Epoch*=1) | 0,1 | 123.82, 150.17 | 277.6, 274.7 | 6.2%, 8.19% | 46.11% |
| NasnetMobile (*BS*=562, *Epoch*=5) + RL (*BS*=1024, *Epoch*=5000) | 0,1 | 123.82, 211.28 | 272.5, 271.3 | 4.2%, 11.4% | 46.01% |
| ResNet-50 (*BS*=20, *Epoch*=5) + DCGAN (*BS*=256, *Epoch*=5) | 0,1 | 109.95, 129.13 | 341.9, 75.0 | 2.7%, 8.2% | 14.9% |
| MobileNet (*BS*=205, *Epoch*=5) + DCGAN (*BS*=230, *Epoch*=5) | 0,1 | 114.8, 131.43 | 255.7, 78.3 | 1.7%, 6.2% | 21.34% |
| Inceptionv3 (*BS*=20, *Epoch*=5) + ResNet50 (*BS*=205, *Epoch*=5) | 0,2 | 115.14, 139.96 | 470.1,342.6 | 48.67%, 28.7% | 19.2% |
| Inceptionv3 (*BS*=20, *Epoch*=5) + Small BERT L4-H512-A8 (*BS*=256, *Epoch*=1) | 0,2 | 115.14, 204.2 | 364.6,341.8 | 9.6%, 16.1% | 43.22% |
| NasnetMobile (*BS*=562, *Epoch*=5) + ResNet50 (*BS*=205, *Epoch*=5) | 0,2 | 123.82, 139.96 | 290.5,343.0 | 11.11%, 28.9% | 34.9% |
| NasnetMobile (*BS*=562, *Epoch*=5) + Small BERT L4-H512-A8 (*BS*=256, *Epoch*=1) | 0,2 | 123.82,204.2 | 265.6,317.5 | 1.64%, 7.8% | 42.8% |
| MobileNet (*BS*=205, *Epoch*=5) + Image Segmentation (*BS*=256, *Epoch*=5) | 0,2 | 114.8, 122.4 | 253.7, 35.5 | 0.9%, 3.4% | 11.19% |
| RL (*BS*=1024, *Epoch*=5000) + Small BERT L4-H512-A8 (*BS*=256, *Epoch*=1) | 1,2 | 211.28, 204.2 | 260.19, 307.33 | 6.8%, 4.4% | 42.8% |
| Small BERT L2-H128-A2 (*BS*=256, *Epoch*=1) + ResNet50 (*BS*=205, *Epoch*=5) | 1,2 | 150.17, 139.96 | 305.6, 276.5 | 20.3%, 3.9% | 41.2% |
| Small BERT L2-H128-A2 (*BS*=256, *Epoch*=1) + Small BERT L4-H512-A8 (*BS*=256, *Epoch*=1) | 1,2 | 150.17, 204.2 | 267.2, 303 | 5.2%, 2.9% | 44.7% |
| RL (*BS*=1024, *Epoch*=5000) + ResNet50 (*BS*=205, *Epoch*=5) | 1,2 | 211.28, 139.96 | 303.1, 273.8 | 24.4%, 2.9% | 40.5% |
| DCGAN (*BS*=230, *Epoch*=5) + Image Segmentation (*BS*=256, *Epoch*=5) | 1,2 | 131.43, 122.4 | 81.1, 46 | 10%, 34.11% | 24.9% |

opted for re-generating grayscale 28x28 MNIST images. Moving on to RL, the Cart-Pole problem in the OpenAI Gym suite environment was implemented using Deep Q-Networks (DQN). Cart-pole problem, also known as the inverted pendulum, refers to balancing an unstable pole placed atop the center of mass of a cart by moving the cart in either direction of a 1D plane. The DQN-based RL algorithm considers the cart's position and velocity, the pole's inclination angle to the cart, and its angular momentum to maintain the pole's stability. Moving onto workload characterization in regards to natural language processing, it revolved around fine-tuning seven different pre-trained BERT models [19] for sentiment analysis in the IMDB movie review dataset.

The scripts were executed in the Google Colaboratory Pro platform, on a Tesla P100 GPU, with all implementations in the TensorFlow environment. The dataset is publicly released through GitHub [1], along with the scripts used for *Futurewei mini-ML Workload Dataset* generation.

## *B. Feature Selection*

With 66 features at our disposal, we manually trim a few non-relevant features like attribute ID 1, which acts as the task identifier, referring to Table I, at the very outset. We also let go of some other attributes like Attribute ID 44, 45, 55, 56, and 57. These refer to the total availability of memory and

TABLE IV
INTRA CLUSTER TASK INTERFERENCE PREDICTION

| Task | Cluster ID | Distance from Centroid | Time for Completion (real) (Secs) | Individual Slow Down | Packing Saving |
|---|---|---|---|---|---|
| Inceptionv3 (*BS*=20, *Epoch*=5) + NasnetMobile (*BS*=562, *Epoch*=5) | 0,0 | 115.14, 123.82 | 394.5, 272.17 | 24.7%, 4.1% | 31.6% |
| Inceptionv3 (*BS*=20, *Epoch*=5) + MobileNet(*BS*=205, *Epoch*=5) | 0,0 | 115.14, 114.8 | 364.24, 269.7 | 15.2%, 7.2% | 35.8% |
| NasnetMobile (*BS*=562, *Epoch*=5) + MobileNet(*BS*=205, *Epoch*=5) | 0,0 | 123.82, 114.8 | 269.53, 255.08 | 3.1%, 1.4% | 47.4% |
| NasnetMobile (*BS*=562, *Epoch*=5) + ResNet-50 (*BS*=20, *Epoch*=5) | 0,0 | 123.82, 109.95 | 275.15, 420.13 | 5.3%, 26.3% | 29.2% |
| MobileNet (*BS*=205, *Epoch*=5) + ResNet-50 (*BS*=20, *Epoch*=5) | 0,0 | 114.8, 109.95 | 270.44, 392.40 | 7.5%, 17.9% | 32.8% |
| Small BERT L2-H128-A2 (*BS*=256, *Epoch*=1) + DCGAN (*BS*=230, *Epoch*=5) | 1,1 | 150.17, 131.43 | 276.49, 77.51 | 8.8%, 5.1% | 15.6% |
| Small BERT L2-H128-A2 (*BS*=256, *Epoch*=1) + DCGAN (*BS*=256, *Epoch*=5) | 1,1 | 150.17, 129.13 | 278.69, 69.4 | 9.7%, 0.1% | 13.7% |
| RL (*BS*=1024, *Epoch*=5000) + DCGAN (*BS*=230, *Epoch*=5) | 1,1 | 211.28, 131.43 | 249.36, 75.19 | 2.4%, 2.0% | 21.3% |
| RL (*BS*=1024, *Epoch*=5000) + DCGAN (*BS*=256, *Epoch*=5) | 1,1 | 211.28, 129.13 | 247.50, 69.87 | 1.6%, 0.8% | 20.8% |
| DCGAN (*BS*=230, *Epoch*=5) + DCGAN (*BS*=256, *Epoch*=5) | 1,1 | 131.43, 129.13 | 88.45, 83.63 | 20.0%, 20.6% | 38.1% |
| ResNet-50 (*BS*=205, *Epoch*=5) + Small BERT L4-H512-A8(*BS*=256, *Epoch*=1) | 2,2 | 136.96, 204.2 | 300.97, 357.77 | 13.1%, 21.5% | 36.1% |
| ResNet-50 (*BS*=205, *Epoch*=5) + Image Segmentation (*BS*=256, *Epoch*=5) | 2,2 | 136.96, 122.4 | 268.90, 35.33 | 1.0%, 3.0% | 10.4% |
| ResNet-50 (*BS*=205, *Epoch*=5) + Inceptionv3(*BS*=205, *Epoch*=5) | 2,2 | 136.96, 117.25 | 359.10, 298.06 | 34.9%, 12.6% | 32.3% |
| Small BERT L4-H512-A8(*BS*=256, *Epoch*=1) + Image Segmentation (*BS*=256, *Epoch*=5) | 2,2 | 204.2, 122.4 | 303.27, 35.08 | 3%, 2.2% | 7.7% |
| Small BERT L4-H512-A8(*BS*=256, *Epoch*=1) + Inceptionv3(*BS*=205, *Epoch*=5) | 2,2 | 204.2, 117.25 | 332.95, 283.29 | 13.1%, 7.1% | 40.4% |
| Image Segmentation (*BS*=256, *Epoch*=5) + Inceptionv3(*BS*=205, *Epoch*=5) | 2,2 | 122.4, 117.25 | 35.66, 272.34 | 3.9%, 2.9% | 8.8% |

disk space. We do so as redundantly available memory or disk space would not affect task interference.

We explore four feature extraction techniques for our two target variables, attribute ID 11 *Maximum GPU utilization percent*, and attribute ID 25 *Maximum GPU Memory Allocated percent*. We measure the variation or relationship of all the attributes for our two regressor variables through Pearson, Spearman, and Kendall's coefficient of determination. We observe that maximum GPU time spent accessing memory percent holds the highest correlation for Maximum GPU Utilization percent. However, our other target variable, Maximum GPU memory allocated percent, demonstrates a weak correlation to mostly all the predictors. The correlation of all attributes for the predictor variables is visualized in our public GitHub repository [1].

Next, we delve into feature selection by random forests, ridge regression, and lasso regression. For random forests, we perform a feature space exploration by varying decision tree depth over 3, 4, 6, 8, 10, 15, and 20; with exploration in the number of considered features for constructing the decision trees as 3, 4, 6, 8, 10, 15, and 20; alongside model estimators of 50, 100, and 150. In regards to ridge and lasso regression, we perform a exploration over 0.001, 0.01, 0.1, 1, 10, 100, and 1000 for the best fit *L2* and *L1* multipliers. Fig. 2 demonstrates the relative ranking of all the attributes for our target variable, Maximum GPU Utilization percent.

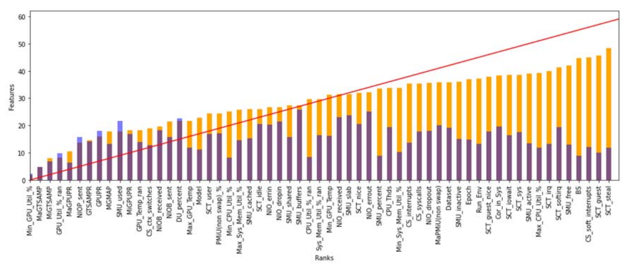

Fig. 2. Relative ranking of all the attributes for our target variable, Maximum GPU Utilization percent. Orange bars indicate the mean rank, with blue bars indicating the standard deviation of the attributes over all the four feature importance techniques used. The red line refers to the mean average rank, which grows by unity with added features.

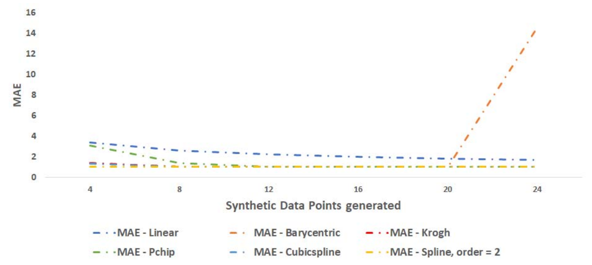

Fig. 3. Uncertainty Quantification for Volume of Data Generated

## *C. Synthetic Data Generation*

Through Table II, we visualize the synthetic data generation results for Maximum GPU utilization percent using all the interpolation methods to varying batch sizes, for image classification on the GTSRB dataset using MobileNet architecture. We demonstrate the quality assessment results, as demonstrated in Fig. 1 of the generated synthetic samples using Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) metrics. We infer that synthetic data yield quality differs depending on the ML architecture.

Next, we quantify the uncertainty in synthetic samples for quality assessment over the volume of generated synthetic data. Roughly speaking, with a small quantity of synthetic data, regenerating the ground truth accurately is improbable. On the other hand, with more synthetic data, the regenerated ground truth would be wayward. Fig. 3 shows a bell-shaped curve for krogh interpolation technique for regenerating ground truth from the interpolated synthetic data for image classification on the GTSRB dataset using MobileNet architecture.

## *D. Clustering*

We perform K-means clustering for our two target variables, maximum GPU Utilization percent and maximum GPU memory allocation percent, alongside maximum time spent accessing memory percent. The clustering is performed on the three attributes over the collected and synthetic instances. We refrain from harnessing attributes like model architecture, dataset, and sensitive hyper-parameters like epochs and batch size to preserve user privacy. With the aid of the elbow method and distortion score calculation, we find the number of clusters to be three. Fig. 4 shows the three clusters and their cluster centers.

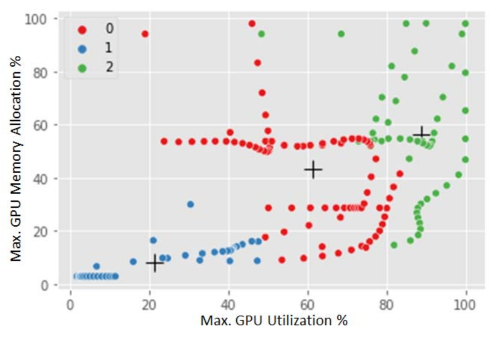

Fig. 4. Cluster Visualization

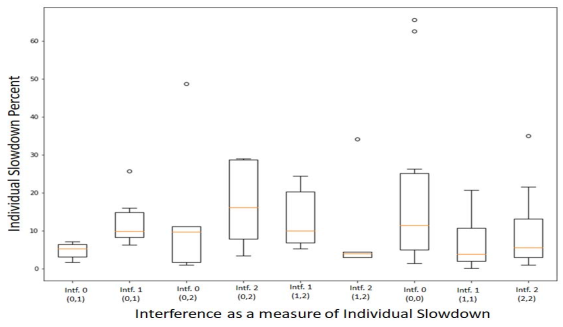

Fig. 5. Cluster-wise Individual Slowdown Visualization. (Intf. 0 (0,1) refers to individual slowdown of for tasks from Cluster 0 when concurrently run with Cluster 1.)

## *E. Evaluating Task Cluster Interference*

We assess the task interference for GPU sharing in our *Futurewei GPU compute cluster*, consisting of NVIDIA v100 GPUs. We determine the execution time of some randomly selected jobs when implemented individually on the GPU cluster. We also measure the euclidean square distance of the job from the cluster center. The relative measure of the jobs from the cluster center induces a proportional measure of the uncertainty. The table is presented in our GitHub repository [1] and omitted from this paper owing to presentation constraints.

Now, we execute the selected jobs parallel to other jobs from the same or different clusters. We evaluate the individual slowdown and packing saving of the same and build a knowledge base for task interference of inter and intra-cluster jobs, as demonstrated in Table III and Table IV, respectively. Fig. 5 demonstrates the individual slowdown for inter and intra-cluster jobs. We can infer from the box-and-whisker plot that jobs of cluster 1 when run in parallel with jobs from cluster 2, show the least individual slowdown. Fig. 6 demonstrates the packing saving for inter and intra-cluster jobs. We can

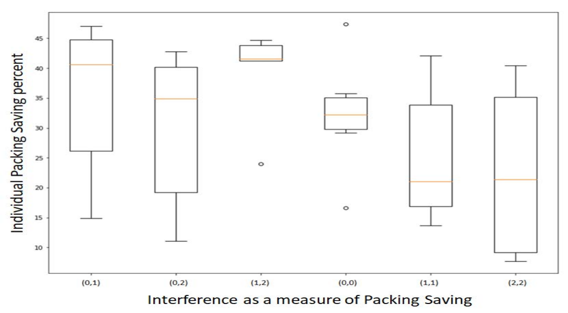

Fig. 6. Cluster-wise Packing Saving Visualization. ((0,1) refers to packing saving for tasks from Cluster 0 when concurrently run with Cluster 1.)

infer that jobs of cluster 1 when run in parallel with jobs from cluster 2, gives the maximum packing saving.

For any given ML training job, we can run the job for one epoch and note the three attributes: maximum GPU utilization, maximum GPU time spent accessing memory percent, and maximum GPU memory allocation percent, and be able to assign it to one of the three clusters. The tool can recommend which two jobs should be executed concurrently from our knowledge base of individual slowdown and packing saving for the jobs from different cluster centers when run concurrently in a single GPU. Also, given how far away the job is from the cluster center, the tool can proportionally assign an uncertainty score to the recommendations.

## IV. Conclusion

To summarize, we have characterized a wide variety of ML training workloads over 66 attributes and released it publicly [1]. Through our knowledge base of inter and intra-cluster task interference, for any user-defined ML training job, our designed tool can recommend a competitor job that can minimize individual job slowdown while maximizing packing saving, with an assigned uncertainty score. PriRecT refrains from collecting user-sensitive metadata like the number of epochs, batch size, model architecture, and the dataset used in its scheduling decision. Our proposed privacy-preserving job recommendation tool for GPU sharing, PriRecT, is thereby successful to meet our proposed objectives. The validation results complements the same. As a part of our future work, we aim to delve deeper into building our knowledge base for a wide variety of workload characterization and having an online reinforcement learning-based design as a upgradation to our designed PriRecT tool.

## References

[1] 2022. [Online]. Available: https://github.com/CentaurusInfra/alnair/tree/main/autonomous-scheduler/job-recommendation-service

[2] M. Jeon, S. Venkataraman, A. Phanishayee, J. Qian, W. Xiao, and F. Yang, "Analysis of Large-Scale Multi-Tenant GPU clusters for DNN training workloads," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. Renton, WA: USENIX Association, Jul. 2019, pp. 947–960.

[3] D. K. Rensin, *Kubernetes - Scheduling the Future at Cloud Scale*, 1005 Gravenstein Highway North Sebastopol, CA 95472, 2015. [Online]. Available: http://www.oreilly.com/webops-perf/free/kubernetes.csp

[4] S. Chaudhary, R. Ramjee, M. Sivathanu, N. Kwatra, and S. Viswanatha, "Balancing efficiency and fairness in heterogeneous gpu clusters for deep learning," ser. EuroSys '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3342195.3387555

[5] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang, F. Yang, and L. Zhou, "Gandiva: Introspective cluster scheduling for deep learning," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. Carlsbad, CA: USENIX Association, Oct. 2018, pp. 595–610. [Online]. Available: https://www.usenix.org/conference/osdi18/presentation/xiao

[6] J. Gu, M. Chowdhury, K. G. Shin, Y. Zhu, M. Jeon, J. Qian, H. Liu, and C. Guo, "Tiresias: A GPU cluster manager for distributed deep learning," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. Boston, MA: USENIX Association, Feb. 2019, pp. 485–500. [Online]. Available: https://www.usenix.org/conference/nsdi19/presentation/gu

[7] K. Mahajan, A. Balasubramanian, A. Singhvi, S. Venkataraman, A. Akella, A. Phanishayee, and S. Chawla, "Themis: Fair and efficient GPU cluster scheduling," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 289–304. [Online]. Available: https://www.usenix.org/conference/nsdi20/presentation/mahajan

[8] T. N. Le, X. Sun, M. Chowdhury, and Z. Liu, "Allox: Compute allocation in hybrid clusters," in *Proceedings of the Fifteenth European Conference on Computer Systems*, ser. EuroSys '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3342195.3387547

[9] D. Narayanan, K. Santhanam, F. Kazhamiaka, A. Phanishayee, and M. Zaharia, "Heterogeneity-Aware cluster scheduling policies for deep learning workloads," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, Nov. 2020, pp. 481–498. [Online]. Available: https://www.usenix.org/conference/osdi20/presentation/narayanan-deepak

[10] Y. Peng, Y. Bao, Y. Chen, C. Wu, and C. Guo, "Optimus: An efficient dynamic resource scheduler for deep learning clusters," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3190508.3190517

[11] L. Biewald, "Experiment tracking with weights and biases," 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[13] M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training," *CoRR*, vol. abs/2104.00298, 2021. [Online]. Available: https://arxiv.org/abs/2104.00298

[14] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," *CoRR*, vol. abs/1707.07012, 2017. [Online]. Available: http://arxiv.org/abs/1707.07012

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015. [Online]. Available: http://arxiv.org/abs/1512.00567

[17] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, vol. 32, pp. 323–332, 2012, selected Papers from IJCNN 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608012000457

[18] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[19] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805