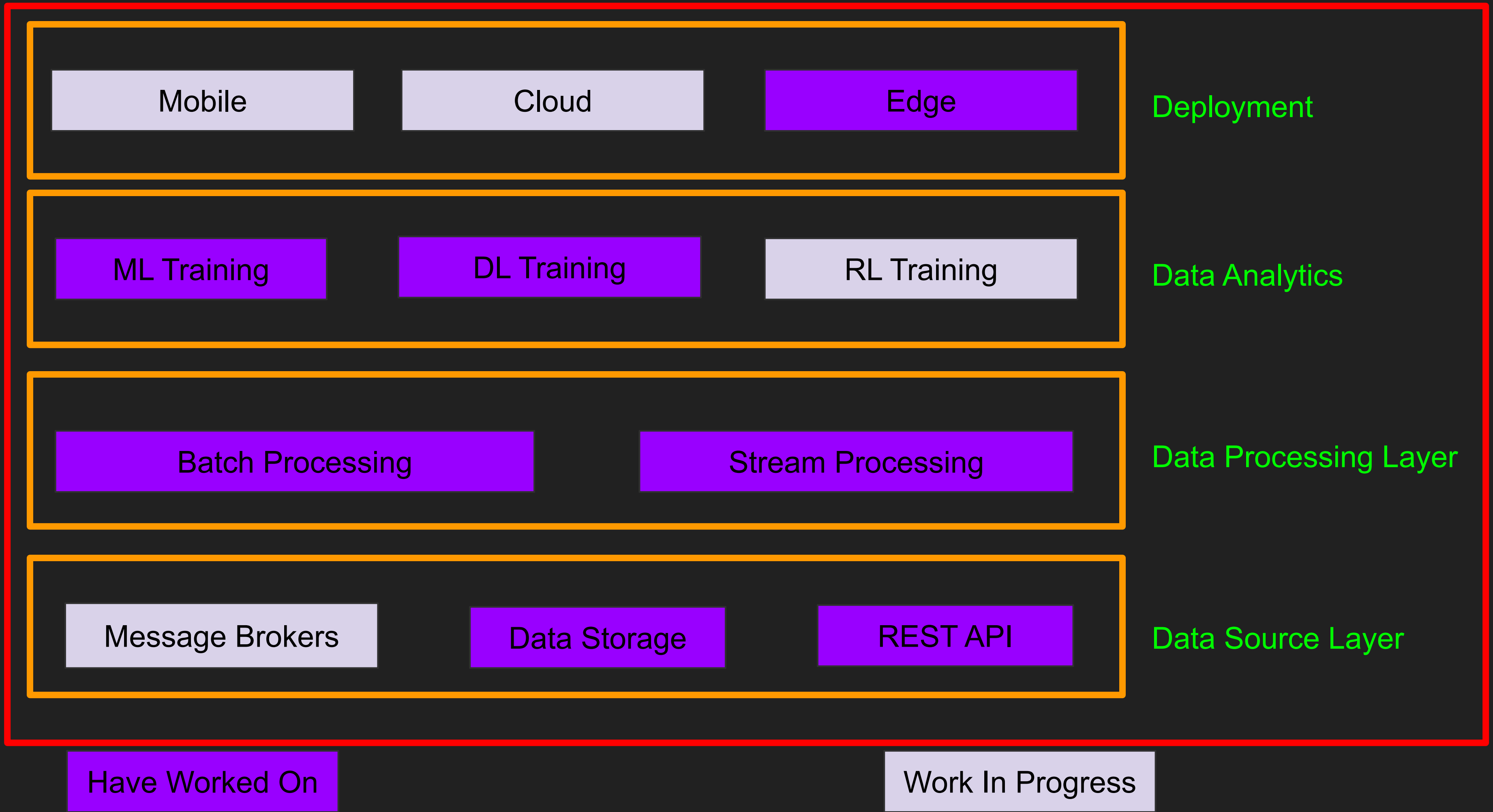
Qualifying Examination Presentation

Vibhatha Lakmal Abeykoon
Intelligent Systems Engineering
Luddy School of Informatics, Computing and Engineering

Academic Advisor: Geoffrey Fox

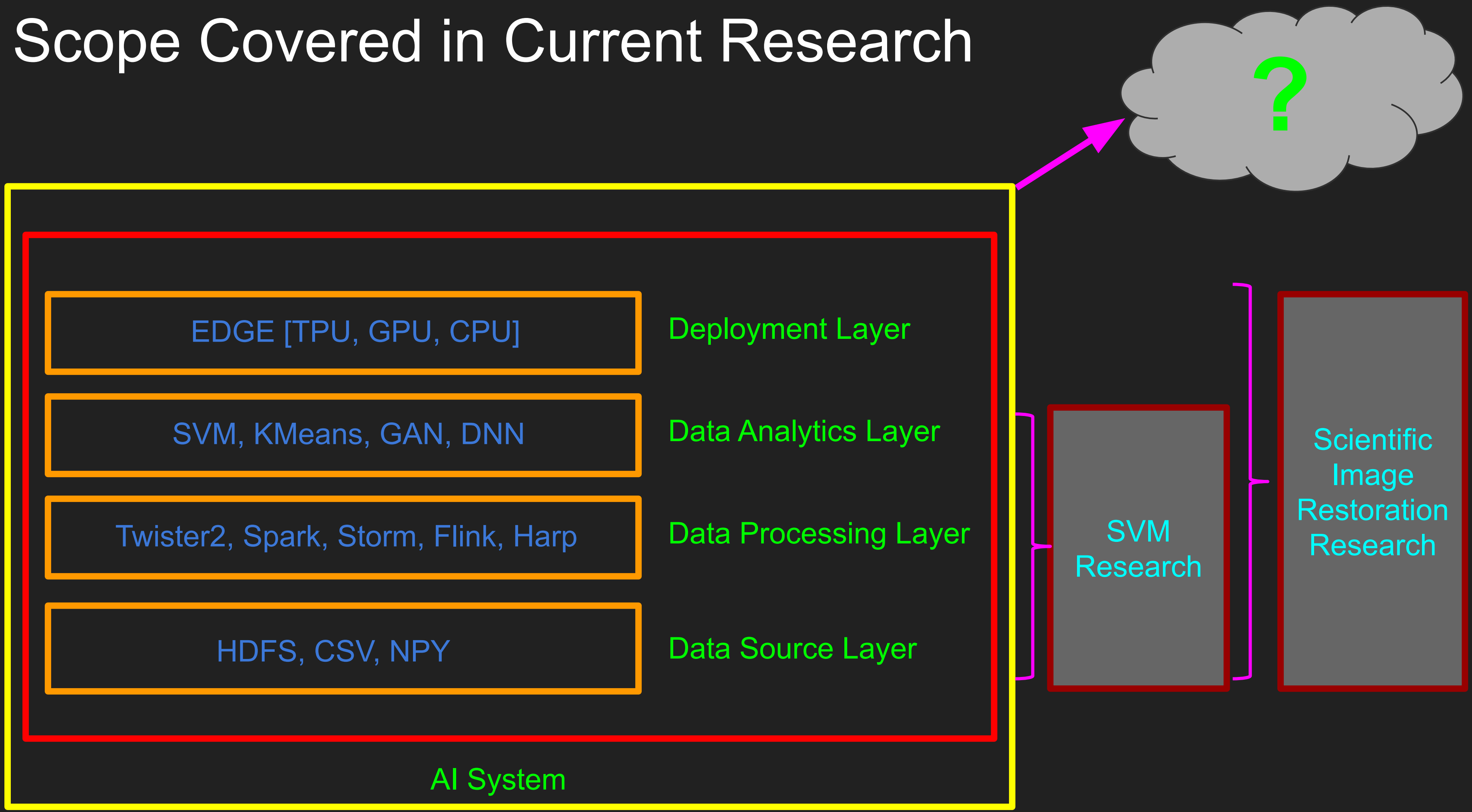
Advisory Committee: Geoffrey Fox, Judy Qiu, Minje Kim

AI System



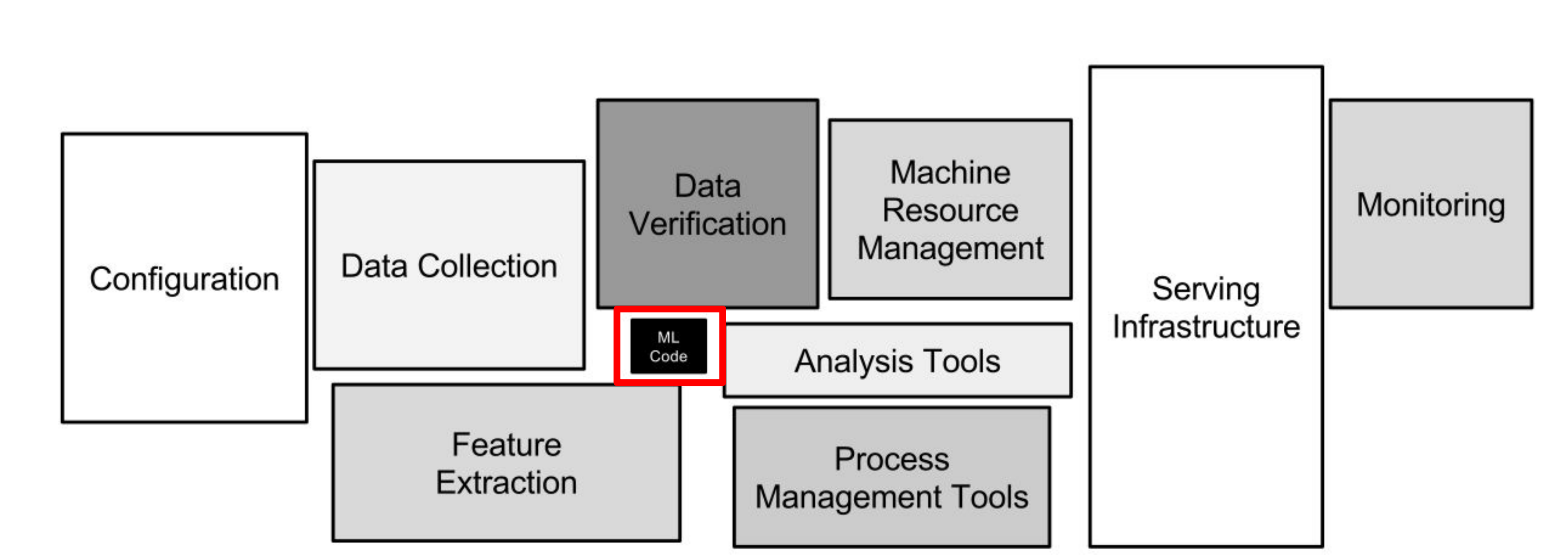
Mobile
Cloud
Edge
Deployment
ML Training
DL Training
RL Training
Data Analytics
Batch Processing
Stream Processing
Data Processing Layer
Message Brokers
Data Storage
REST API
Data Source Layer
Have Worked On
Work In Progress

Scope Covered in Current Research



Artificial Intelligence Systems

Architecture of a Deep Learning System



Motivation

Designing systems to solve problems on its own by learning through experience and memory.

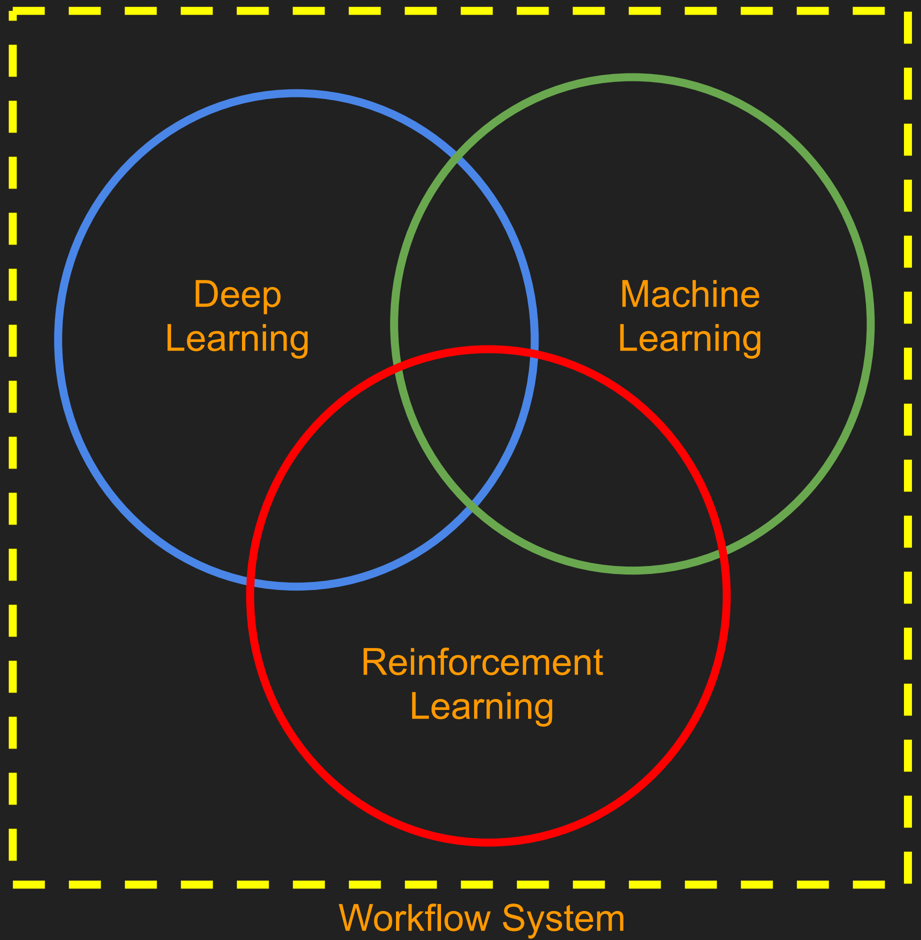
Objective

- Understand deep learning systems.
- How big data systems helps to drive deep learning systems?
- Analyze current trends in Academic and Industry research on deep learning systems.

Overview

- Introduction to AI Systems
- Big data systems design for specific AI use-cases
- Research on AI Systems
- Design Systems for Generic AI Systems
- Conclusion.

AI System



Deep Learning
Machine Learning
Reinforcement Learning
Workflow System

Data Organization

Identification

Kafka, RabitMQ, ZeroMQ, Redis, MongoDB,Oracle SQL, MySQL, etc

Preparation

PCA, T-Distributed Stochastic Neighbor Embedding, Deep Learning AutoEncoders

Ingestion

Spark Structured Streaming, Spark SQL, Apache Calcite, Apache Storm, Apache Flink, Twister2 Streaming.

Storage

Parquet, Apache Arrow, Apache Avro, Oracle SQL, MongoDB, etc.

AI Workflow

AI Training: Machine learning and deep learning systems training with organized data. For reinforcement learning evaluating inputs from the environments and deciding actions. [CPU, GPU, TPU]

AI Testing: Evaluating the AI algorithm against the expected goal.

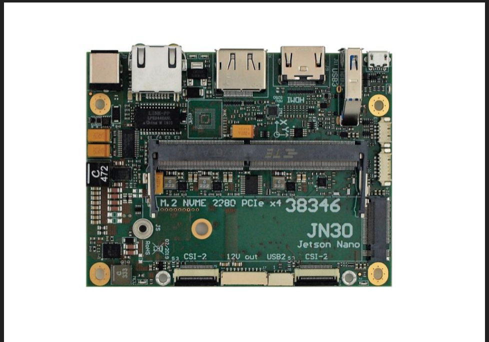
AI Deployment: Deploying trained system to electronic devices. [Android, IOS, Raspberry PI, Edge TPU, Jetson, etc]

AI Training and Evaluation

Table 1: AI Framework and Middleware

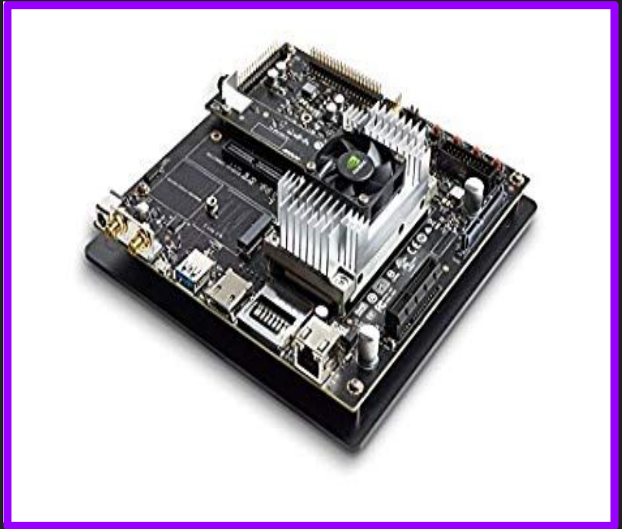
AI/BigData Framework	Middleware	AI Major	Status
Scikit-Learn	Dask	Machine Learning	Active
Tensorflow	Apache Spark, Google XLA	Deep Learning	Active
Pytorch	Apache Spark	Deep Learning	Active
MXNet	Horovod	Deep Learning	Active
MOA	Weka	Machine Learning	Active
Spark Mlib	Apache Spark	Machine Learning	Inactive
Apache Samoa	Apache Samoa	Streaming Machine Learning	Inactive
Keras	Tensorflow, Theano	Deep Learning	Active
Graphx	Apache Spark	Graph Computation	Active
Snap	C++ and Python	Graph Computation	Active
Petuum	Model and Data Parallel Native Core	Machine Learning	Active
H20	Multithreaded MapReduce	Machine Learning	Active
BigDL	Apache Spark	Deep Learning	Active
DeepDriveMD	RCT and EnTK	Protein Fold with Deep Learning	Active
Deep Graph Library	Native Python Libraries Amazon Sagemaker Apache MXNet	Graph Neural Network	Active

AI Deployment



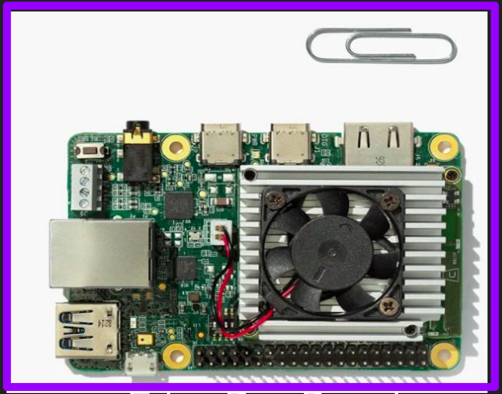
Jetson Nano

Raspberry PI



Jetson TX2

Edge TPU Accelerator



Edge Dev Board

So Far the Discussion

- Introduction to AI Systems
- Big data systems design for specific AI use-cases
- Research on AI Systems
- Conclusion.

Big Data System Design For AI

Training Systems

Predictive Systems

Transfer Learning Systems

Training Systems

- Data pipelining (training data stream or batch, testing data stream or batch, cross-validation data stream or batch)
- Batch and Stream processing on raw data
  - Apache Spark, Apache Storm, Apache Flink, Twister, Twister2, etc
- In memory and disk based data processing capability
  - Parquet, Apache Arrow, Apache Avro, etc
- Distributed Training Support and Collective Communication
  - MPI, Harp, Twister2
- Hybrid system design on HPC and Big Data frameworks.
  - Spark + MPI, Twister2

Predictive Systems

- Data pipelining (From storage or message brokers)
  - Streams (windowing for mini-batches)
  - Batches
- Storage
  - In memory-based or disk-based
- New Model update mechanisms
  - Manage model update over large number of devices
- Support Edge devices for low-latency inference
  - Edge TPU, Edge GPU, Edge CPU
- Collective Communication
  - Data gather, broadcast, reduce, shuffle, etc.

Transfer Learning

- Workflow connecting training and predictive systems
  - Model management for multiple experimentation configurations and keeping track on model specification (Supported by Azure ML)
- Existing model is re-used to train for a specified purpose supported by the existing knowledge on the model.
  - This requires the re-use of training system and predictive system.
- Workflow management is vital when thousands of models are being managed for different purposes. A well defined workflow system is highly influential in streamlined model deployment.

So far the discussion

- Introduction to AI Systems
- Big data systems design for specific AI use-cases
- Research on AI Systems
- Design Systems for a generic AI System
- Conclusion.

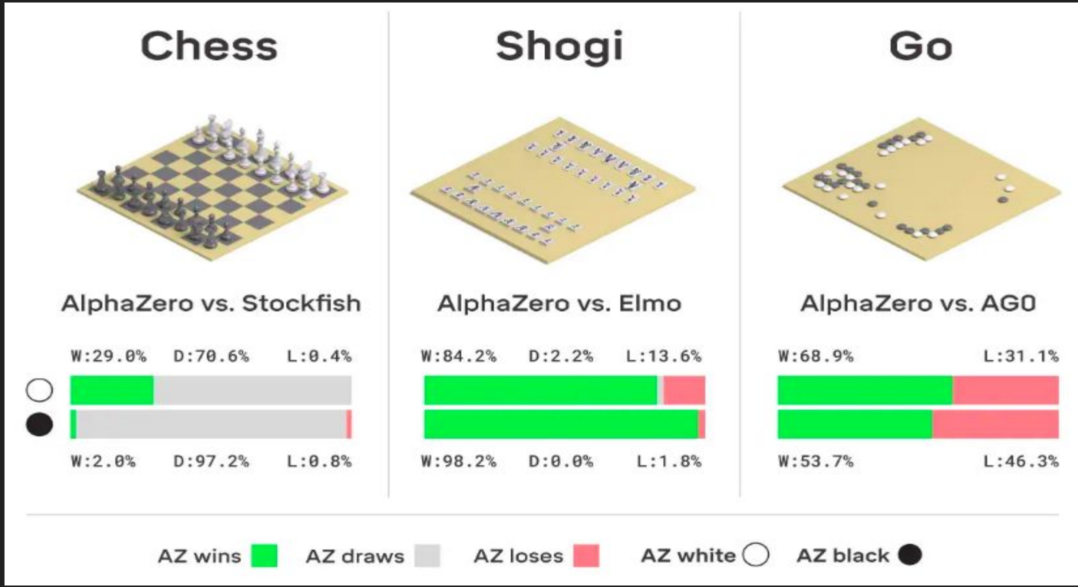
Research on AI Systems

Digital Advisors

Gaming

Gaming with AI

- Alpha Zero is the generalized version of AI game player to master more board games.
- It has played Sogi (Japanese Chess), Chess and Go.
  - Stockfish Engine for Chess
  - Elmo Engine for Sogi
  - Alpha Zero for all
- Facebook created an AI Poker agent beating professional poker players.
- OpenAI Dota



Digital Advisor's Capabilities

- A voice-enabled agent which performs defined tasks.
- Specific Capabilities
  - Audio generation
  - Text generation
  - Image generation
  - Search
- Continuous speech
- Autonomous answering
  - Capability to engage in a conversation with a human.
- Examples:
  - Alexa
  - Google Assistant
  - Cortana

System Design for a Generic AI System

Unified Data Analytics

Deep Learning System Support

Optimizing Systems for Scaling

Breaching the Language Boundary

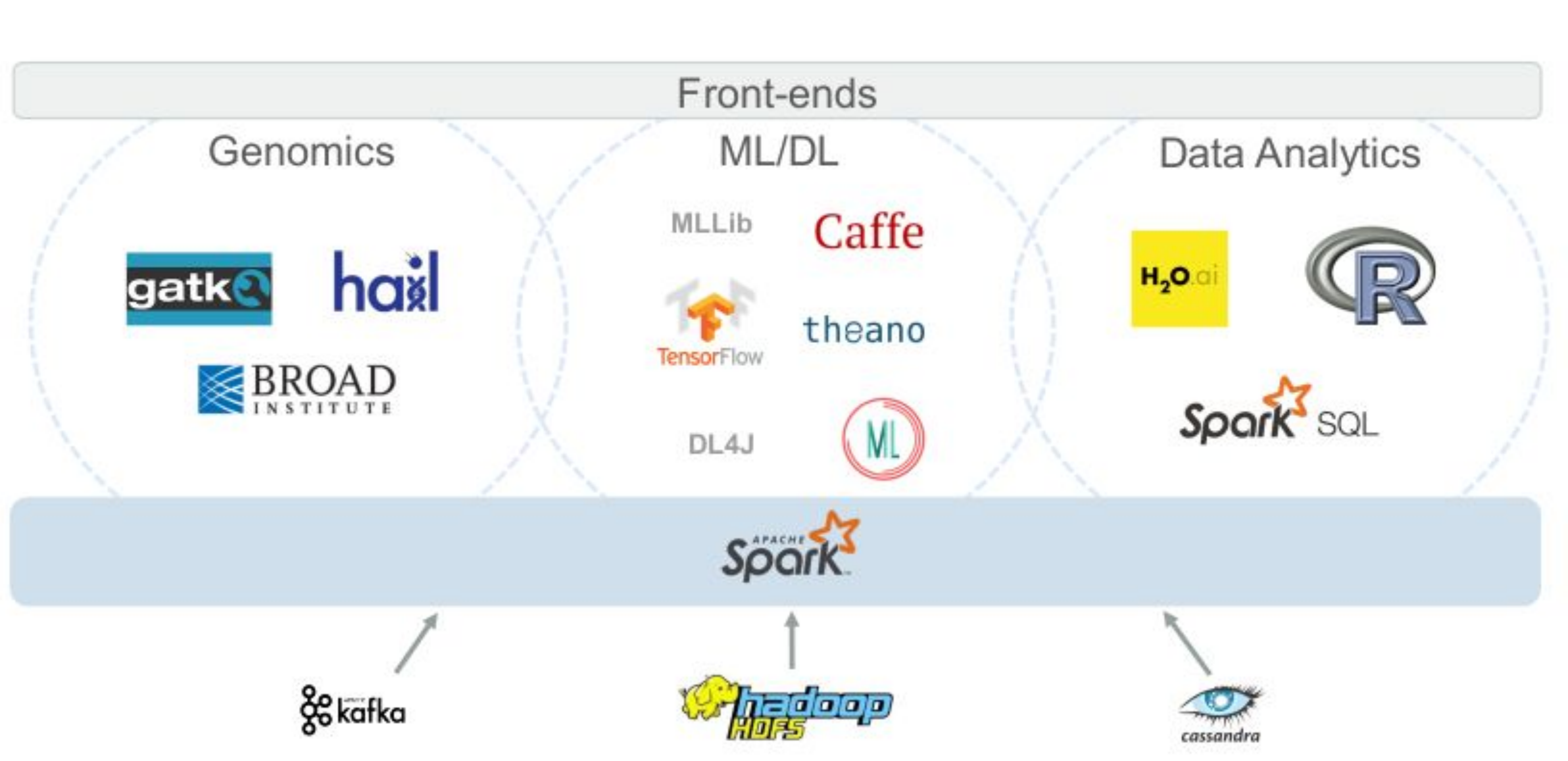
Unified Data Analytics

- Unified data processing engine for Batch and Stream processing
- Google Dataflow is powered by PCollections and Other libraries in Apache Beam
- Supports SQL, Large batch jobs and long running stream jobs
  - Special support for Session based log processing at scale
- Apache Flink, Apache Spark also provides runners to run their jobs using Beam.
- Twister2 is a batch and stream processing unified framework running MPI, TCP and UCX modes with high scalability and performance.
- Harp provides a collective communication API on Java for application developers to run efficient code
  - Supports Intel DAAL and enables Deep Learning and Machine Learning on Intel Hardware

Deep Learning Support

- Dynamic Graph Structure support in Pytorch provides distributed training on both model and data parallelism at scale.
    - Uses MPI, Gloo, TPC modes to run distributed training on CPU, GPU and TPU.
- Tesla Hydra nets supported with Distributed Training systems designed on MPI backends with PyTorch.
- Apache Spark provides support for Tensorflow, Pytorch, MXNet with Data preprocessing and distributed training.
    - Horovod from Uber is one such implementation using MPI for distributed training and PySpark for data pre-processing at scale.
- MPI-oriented research with HiDL (D.K Panda’s et.al) provides highly scalable training support for Pytorch and Tensorflow.

Optimizing Systems for Scaling: Spark



Optimizing Systems for Scaling: Containers

- K8s or Kubernetes works on large scale application scaling with containers.
- K3s is a lightweight Kubernetes version which supports light weight devices likes Edge TPUs, Edge GPUs and Raspberry PIs.

Breaching Language Boundary

- Ultimate goal is to solve problems easily.
  - One wouldn't prefer writing more code on environment design, rather interested in data analytics and simulation design.
- Python is the straightforward choice from the research community.
  - Dask
  - Scikit-Learn
  - Pytorch
  - Tensorflow
  - Parsl
  - EPython
- Core system design is on principles of the state of the art high performance big data systems.
- APIs are more readable and easy to code.

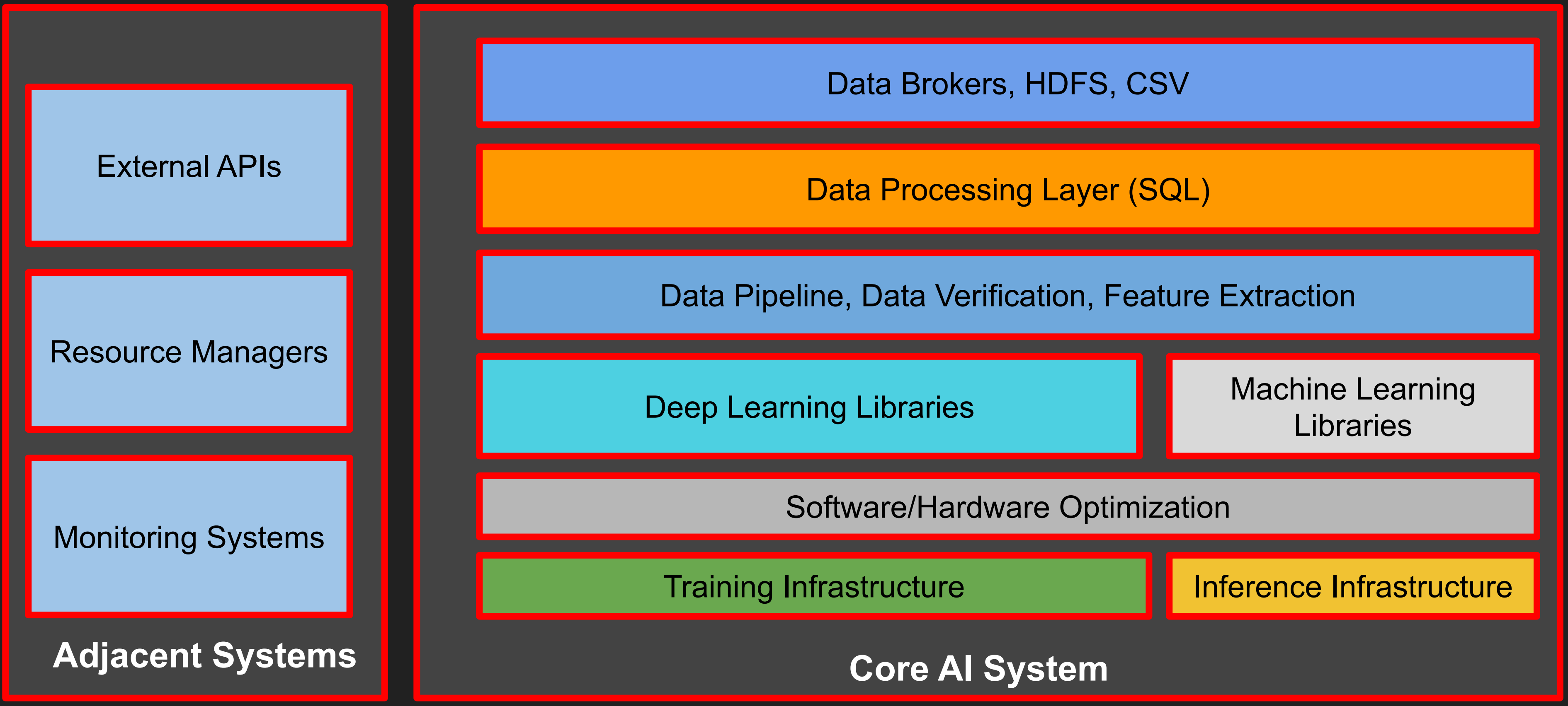
Where does research focus on?

- Improving systems at scaling
  - Distributed training and testing
  - Edge computing and fast inference on mobile devices
- Design surrogate AI systems to solve existing mathematical and physics models with higher efficiency
- Agent training for exceeding human function
  - Currently in Gaming
  - Translation
  - Voice recognition
  - Image recognition
- **Efficient Training** and **Efficient Inference** with decentralization of resources

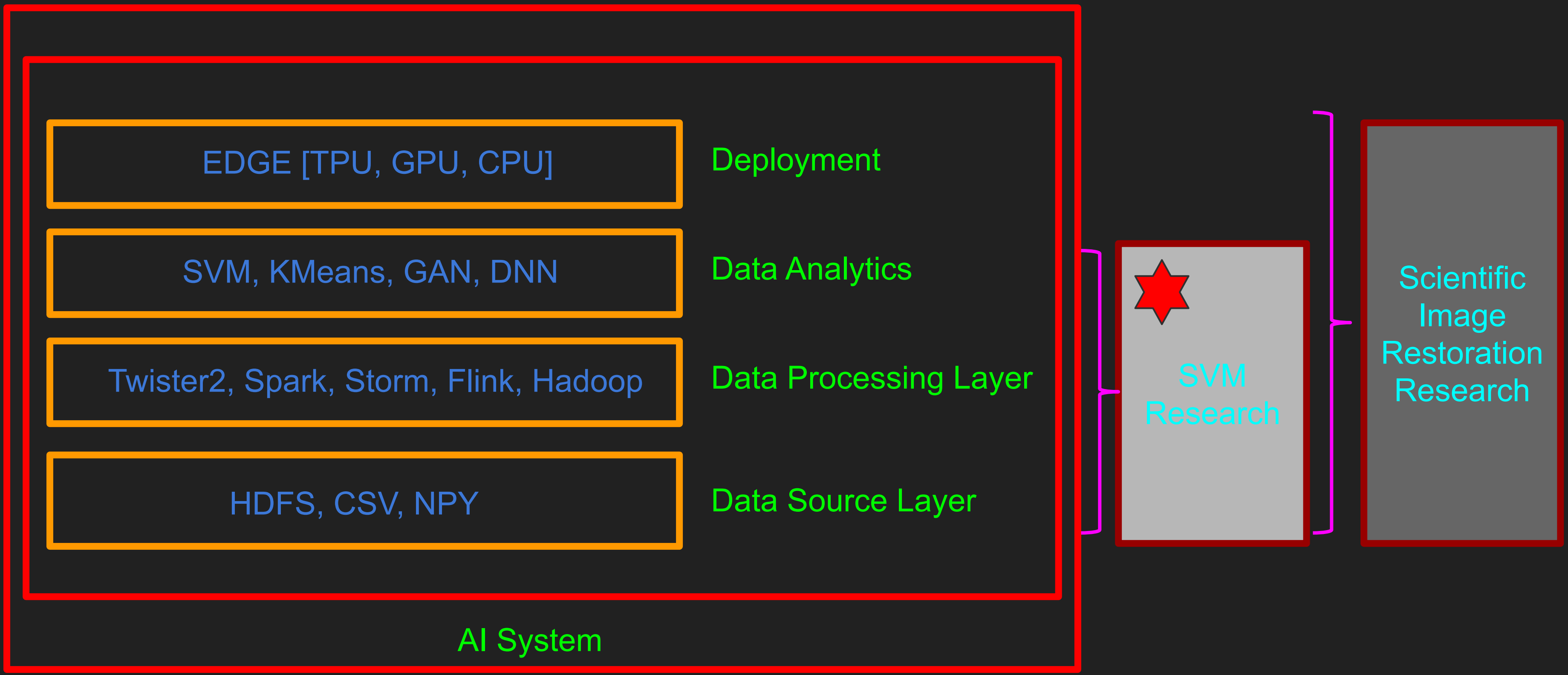
Conclusion

- AI Systems needs a major support from **high performance big data systems** (HPBS) for data organization, algorithm training in parallel mode and provide services in both cloud and edge devices for deploying AI models.
- A high performance unified data analytics framework is necessary to design intelligent systems with a streamlined workflow when dealing with multiple data processing disciplines associated with vivid use cases.
- Supporting various AI libraries from different vendors is vital when managing a larger community of researchers and engineers working on their specialized disciplines in various areas of the workflow.
- Integrating overall AI workflow must be done using HPBS to scale research, development and deployment process.

Deep Learning System Layered Architecture



Scope Covered in Current Research



Distributed SGD-based SVM

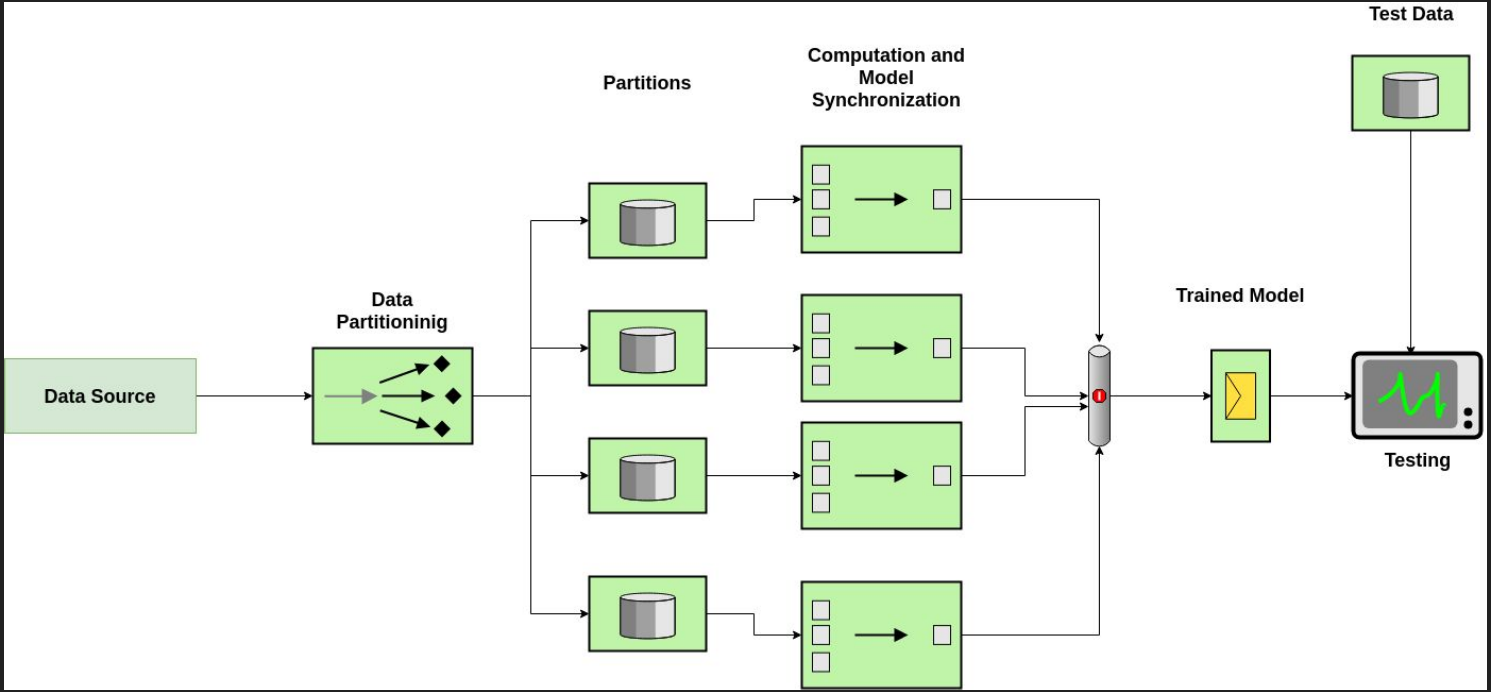
Related Work

- [Pegasos SVM]
- [DC-SVM]
- [pPackSVM]
- [Parallel SGD]
- [Parallel SGD For High Level Architectures]

Objective

- Effect of mini-batch based model synchronization on SGD based SVM algorithm convergence.
- Evaluate efficiency of the training model based on execution time and testing accuracy upon batch size.

System Architecture



Anatomy of Datasets

DataSet	Training Data (60% / 80%)	Cross-Validation Data (60% / 80%)	Testing Data (60%, 80%)	Sparsity(%)	Features
Ijcnn1	21,000 / 28,000	7,000 / 3,500	7,000 / 3,500	40.91	22
Webspam	210,000 / 280,000	70,000 / 35,000	70,000 / 35,000	99.9	254
Epsilon	240,000 / 320,000	80,000 / 40,000	80,000 / 40,000	44.9	2000

Objective Function and Equations

$$J^t = \min_{w \in R^d} \frac{1}{2} ||w||^2 + C \sum_{x,y \in S} g(w;(x,y))$$

$$g(w;(x,y)) = \max(0, 1 - y\langle w|x \rangle)$$

$$w = w - \alpha \nabla J^t, \ \alpha = \frac{1}{1+t}$$

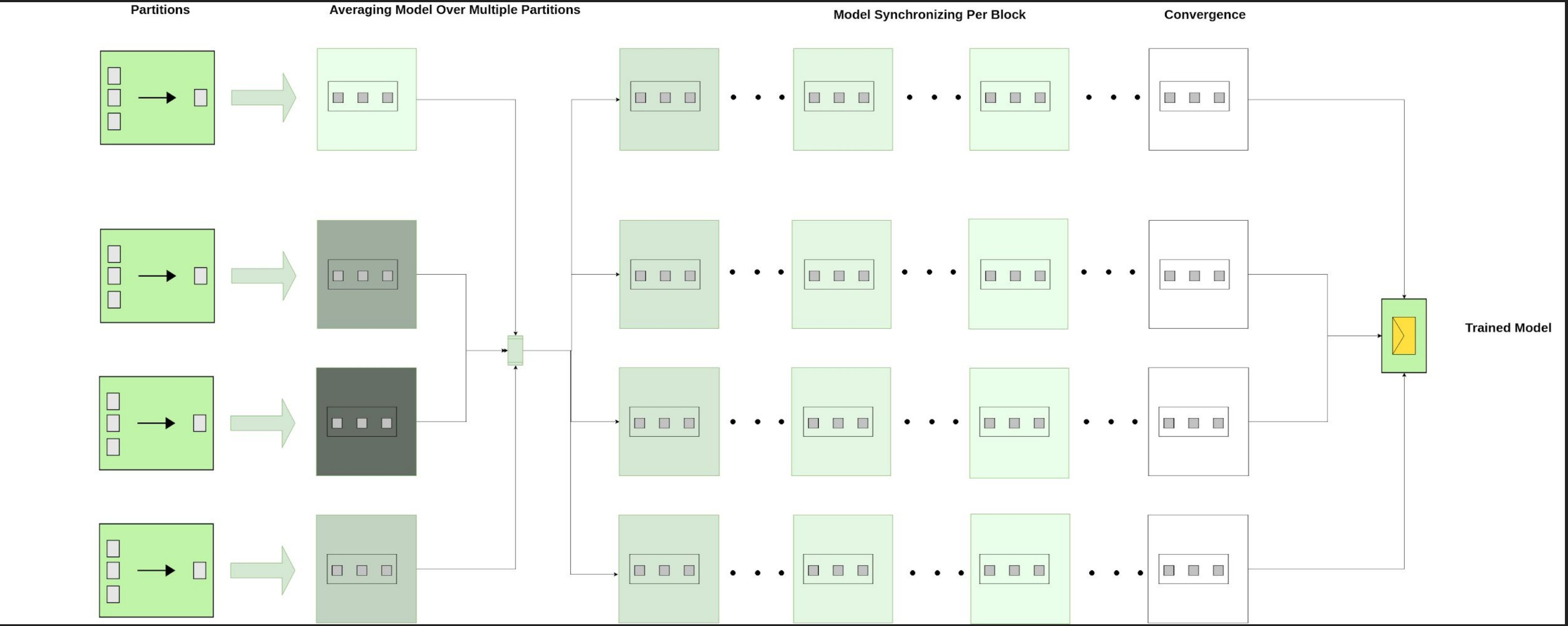
$$\nabla J^t = \begin{cases} w & \text{if } \max(0, 1 - y\langle w|x \rangle) = 0 \\ w - Cx_i y_i & \text{Otherwise} \end{cases}$$

$$y\langle w|x \rangle = y_i w^\top x_i$$

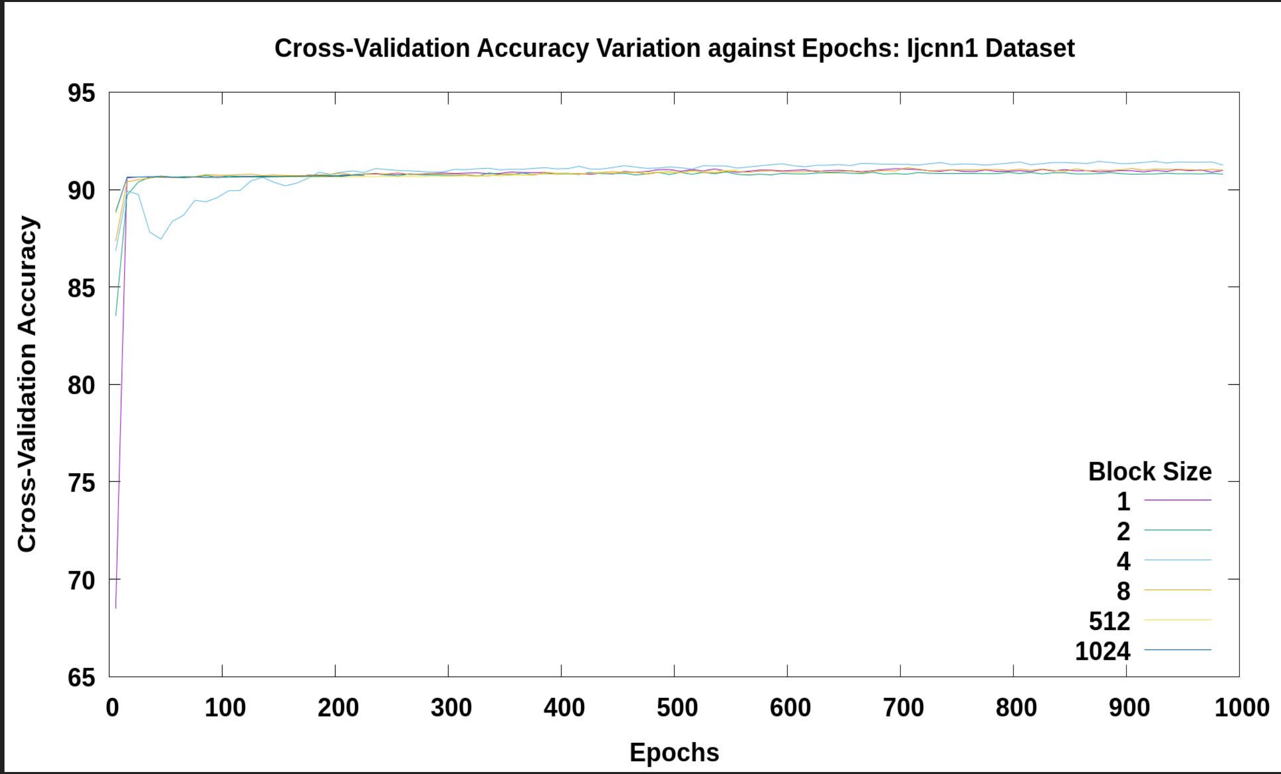
Algorithm Implementation

- We used OpenMPI 3.0.0 (C++)
- AllReduce collective was used to do model synchronization and later averaging was done over each process.
- Learning rate is an adaptive diminishing function.
    - Function of number of epochs

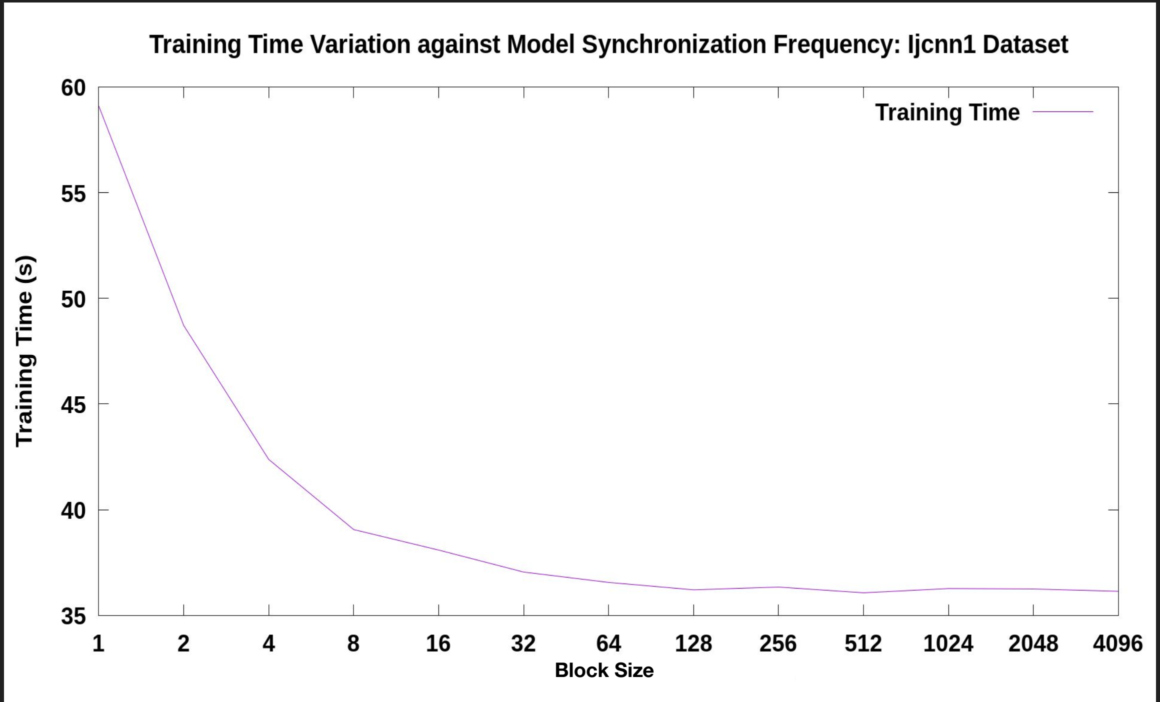
Model Synchronization



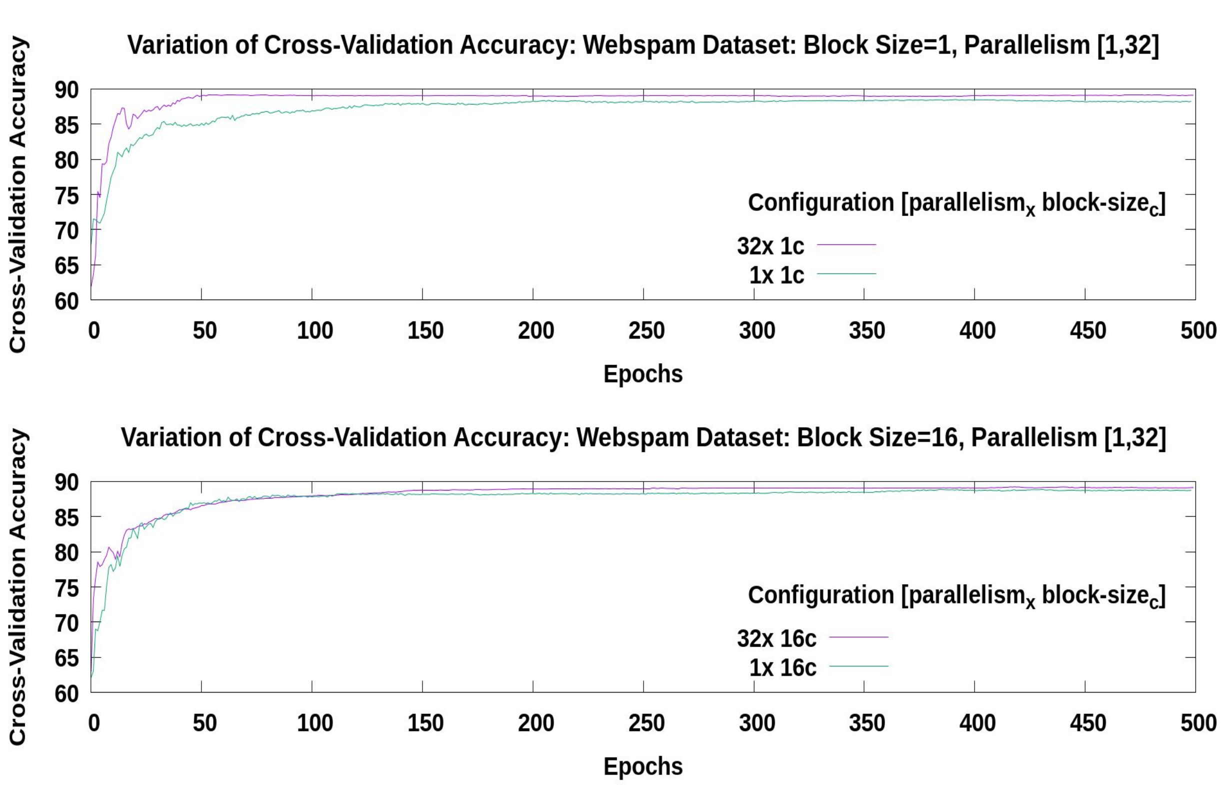
Cross Validation Accuracy Variation [Sequential Mode] - Ijcnn1 Dataset



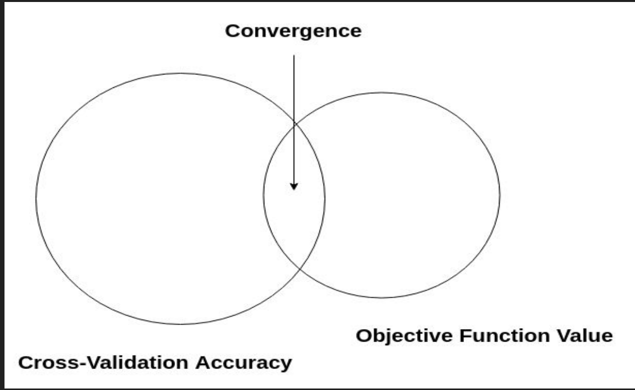
Training Time Variation [Sequential Mode] - Ijcnn1 Dataset

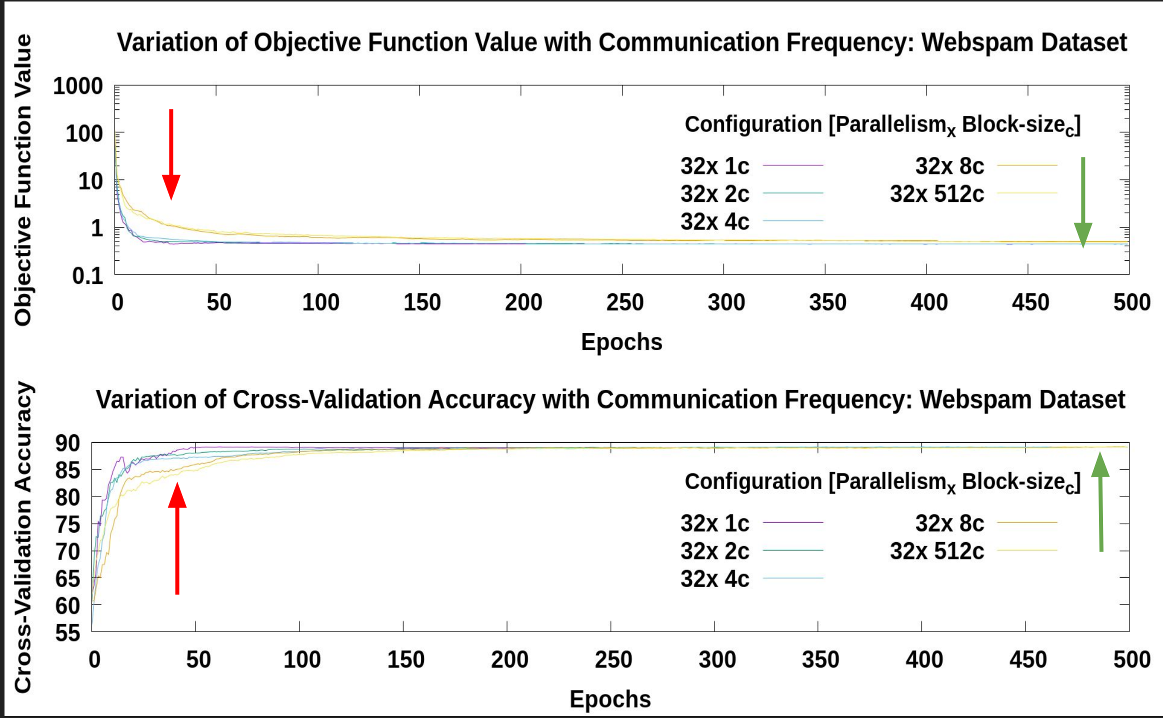


Cross-Validation Accuracy Variation Against Parallelism - Webspam Dataset



Convergence with Parallelism



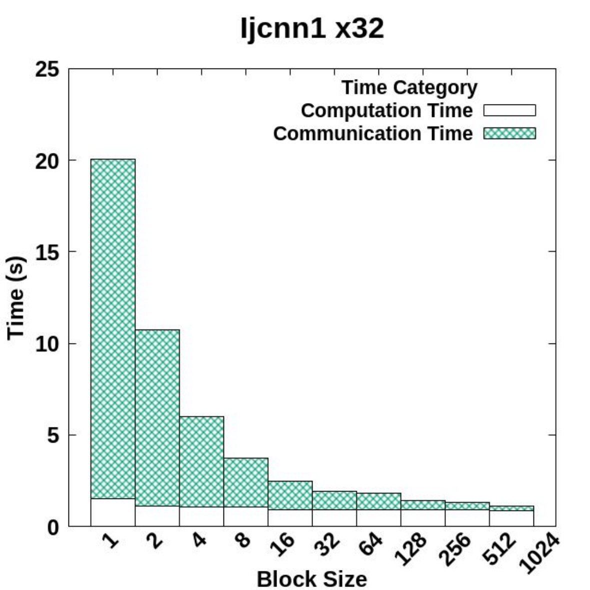


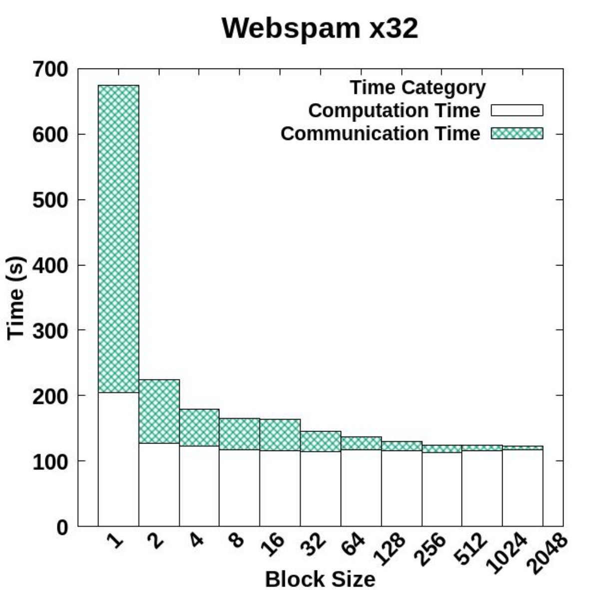
Convergence Point

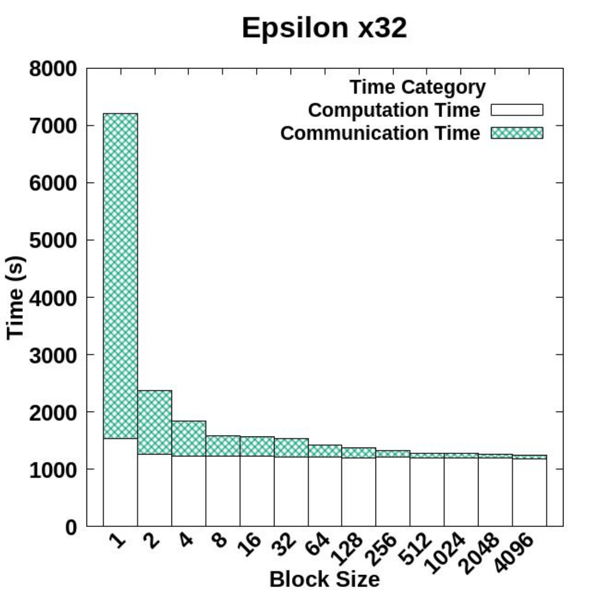
Understanding Performance

- Understanding the performance of the algorithm in terms of parallelism level and block size, in terms of times.
    - Time to update one point (0.5625 us/epoch - epsilon x32 b=1)
    - Time to check for convergence (0.375 us/epoch - epsilon x32 b=1) (objective function evaluation)
    - Time for MPI collective (3.5625 us/epoch - epsilon x32 b=1) (model synchronization, i.e allreduce)

Training Time Breakdown

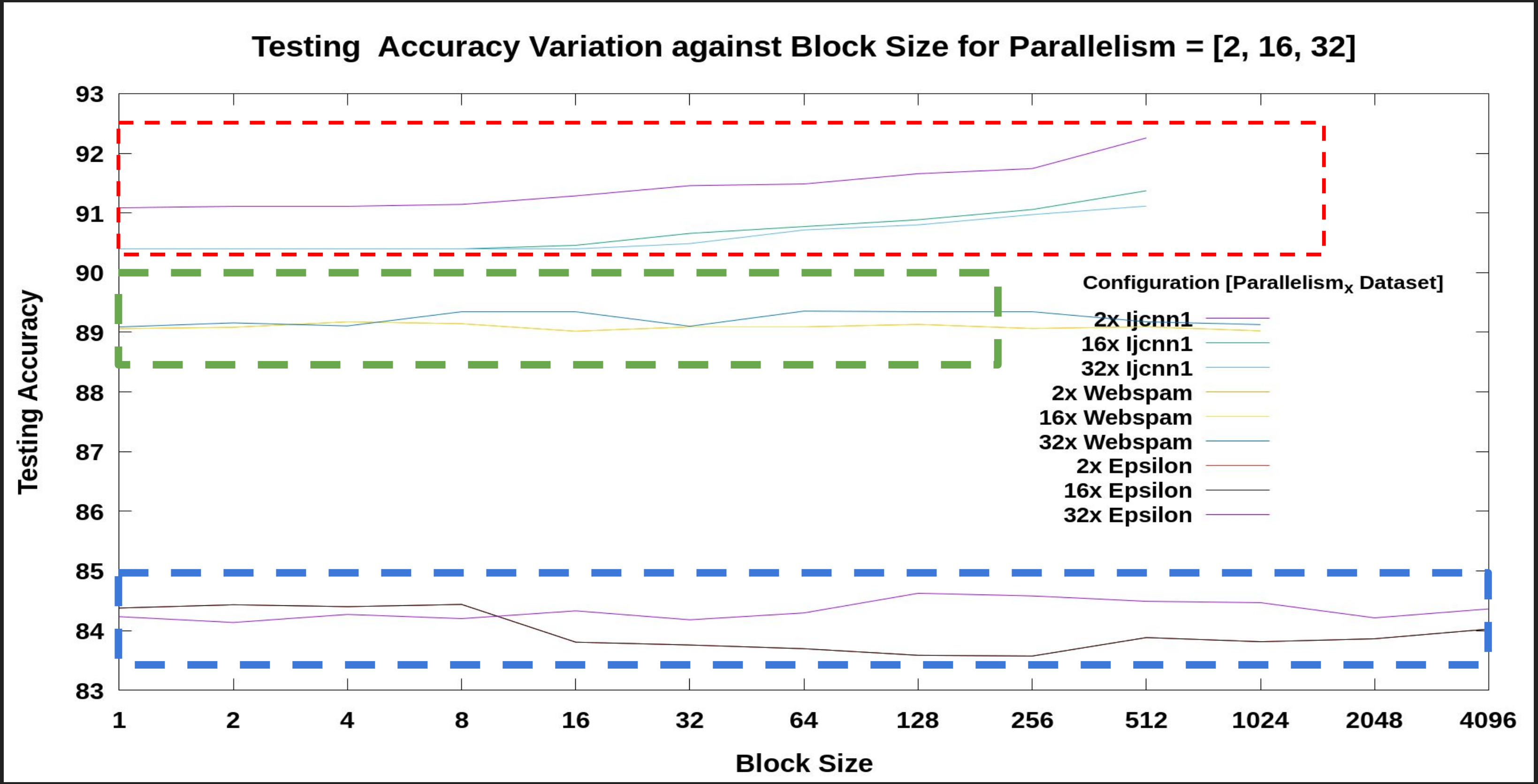






IJCNN1 WEBSPAM EPSILON

Testing Accuracy Variation



Summary of Experimental Results

DataSet	Sequential Timing (seconds)	Parallel Timing (seconds)	Speed Up (x1 vs x32)
Ijcnn1	22.19	1.37	16.2
Webspam	2946.49	120.02	24.55
Epsilon	20037.5	968.782	21.12

Experiment Environment

- For this we used Juliet Cluster which is a part of the Future Systems cloud environment of Digital Science Center in Indiana University Bloomington
- Configuration of a Node in the Cluster
    - Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz
    - Cores Per Socket = 18
    - Sockets = 2
    - Threads Per Core = 2

Extension of Research

- Providing support in both HPC and Dataflow-like computation models.
- Twister2 SVM (Batch and Streaming (Published Stream-ML, IEEE Big Data Dec/2019)) https://twister2.gitbook.io/twister2/examples/ml/svm
- Available With Twister2 0.2.0 release. [Twister2 is a framework developed by Indiana University Bloomington as a Big Data Hosting Environment: A composable framework for high-performance data analytics]
- Twister2 TSet: High Performance Iterative Dataflow (paper published on May 10th, 2019) uses this SVM model as an application.
- Currently working on BLAS level and language level optimizations on distributed SVM on Java and C++ implementations.

Conclusion

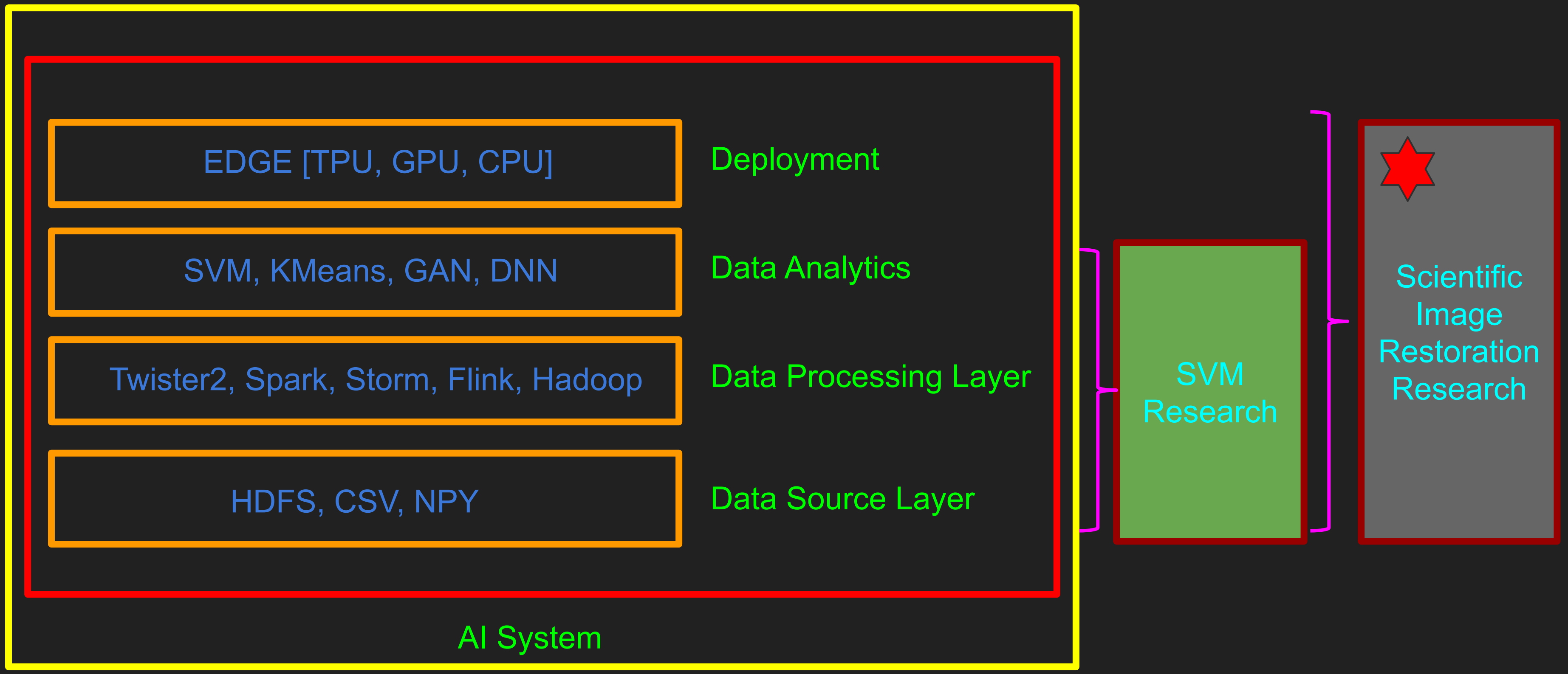
- Designing a highly scalable SVM algorithm with respect existing implementations.
- Understood the convergence of a distributed machine learning algorithm in depth with micro-benchmarking on accuracy and execution time.

Implementation Type	Optimization Model	Scalability	Implementations
SMO	Lagrangian Optimization	Low	LibSVM (Sequential)
PSVM	Matrix Decomposition	Moderate	PSVM (Google)
SGD-SVM	Gradient Descent Variations	High	Pegasos SVM
PSGD-SVM	Parallel SGD	Very high	pPackSVM
Our HPC Implementation	Parallel Pegasos SGD (supports BLAS(Java,C++))	Very high	PSGDDSVMC, PSGDSVM
Our Hybrid Implementation	Parallel SGD (Twister2 [MPI Backend])	Very high	Twister-SVM

Summary

- Code
  - OpenMPI C++: https://github.com/vibhatha/PSGDSVMC [Used in Paper]
  - OpenMPI Java: https://github.com/vibhatha/PSGDSVM
  - OpenMPI Python: https://github.com/vibhatha/PSGDSVMPY
  - Twister2: https://twister2.org/docs/examples/ml/svm/svm
- Paper
  - Pre-print: https://arxiv.org/abs/1905.01219

Scope Covered in Current Research



Scientific Image Restoration Anywhere

Image Restoration with High Efficiency

- Deep Learning Models are used in most of the scientific experimental facilities. The intensity of the usage has highly increased in the recent years.
- Low Latency inference is one of the highly demanding requests by scientists.
- Once the deep learning models are trained they must be portable such that it can be used anywhere with less installation and configuration overheads.

Objectives

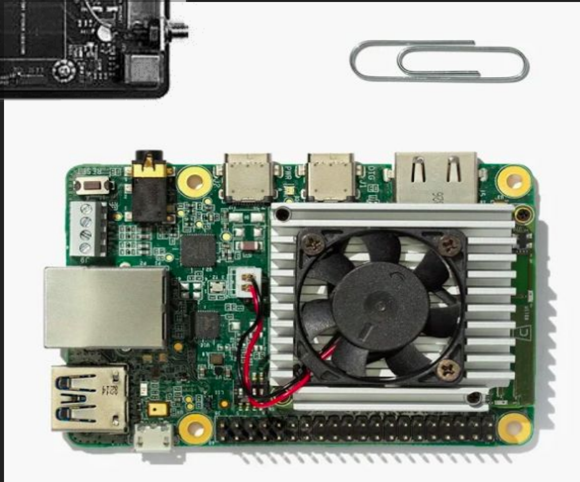
- Faster Inference
- Low Cost Medium for Inference
- Availability for Scientific Applications

Technology Usage and Data Source

- Tensorflow 1.14
- TensorflowLite (Edge Compatible M
- TPU Accelerator
- TPU Dev Board
- GPU Accelerator: NVIDIA Jetson
- Low-Dose X-ray images from APS (Argonne Photon Source)

Jetson Tx2

TPU Accelerator



TPU Dev Board

Edge TPU Specs

CPU	NXP i.MX 8M SoC (quad Cortex-A53, Cortex-M4F)
GPU	Integrated GC7000 Lite Graphics
ML accelerator	Google Edge TPU coprocessor
RAM	1 GB LPDDR4
Flash memory	8 GB eMMC
Wireless	Wi-Fi 2x2 MIMO (802.11b/g/n/ac 2.4/5GHz) and Bluetooth 4.2
Dimensions	48mm x 40mm x 5mm

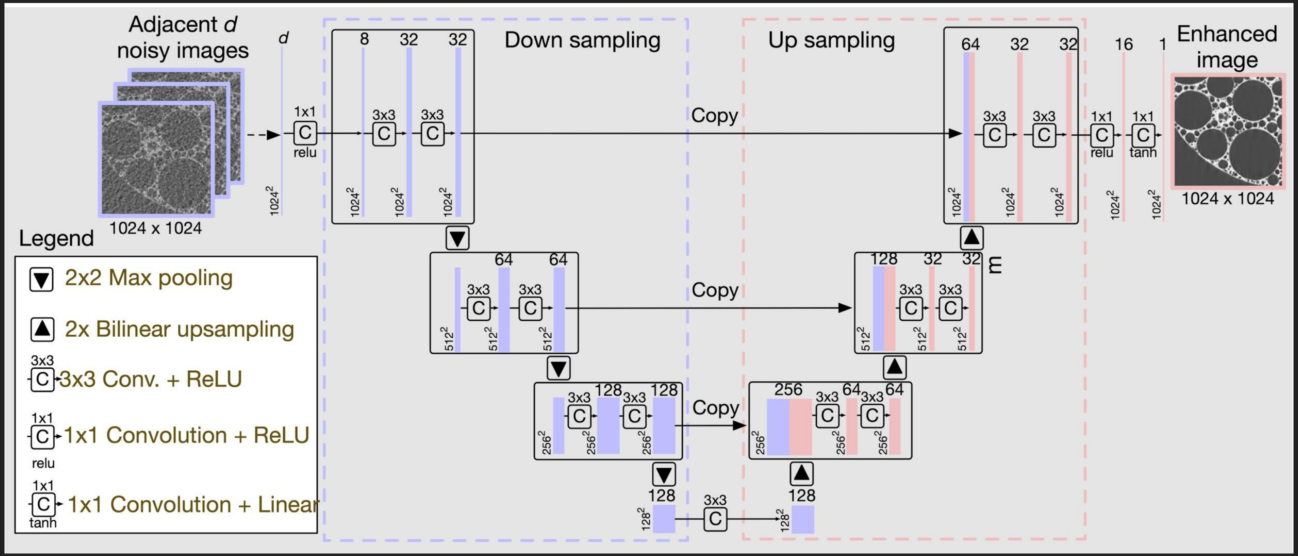
(5V USB Type-C)

Edge TPU Dev Board

ML accelerator	Google Edge TPU coprocessor
Connector	USB 3.0 Type-C* (data/power)
Dimensions	65 mm x 30 mm

Edge TPU Accelerator

TomoGAN Generator Architecture

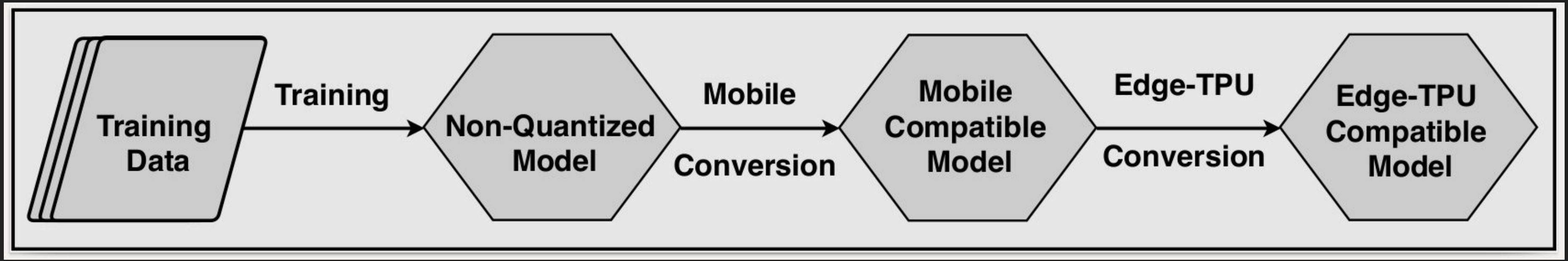


For inference, TomoGAN needs 301 Billion floating-point operation to denoise an image with 1024x1024 pixels.

Ref.: Z. Liu et al. TomoGAN. arXiv:1902.07582

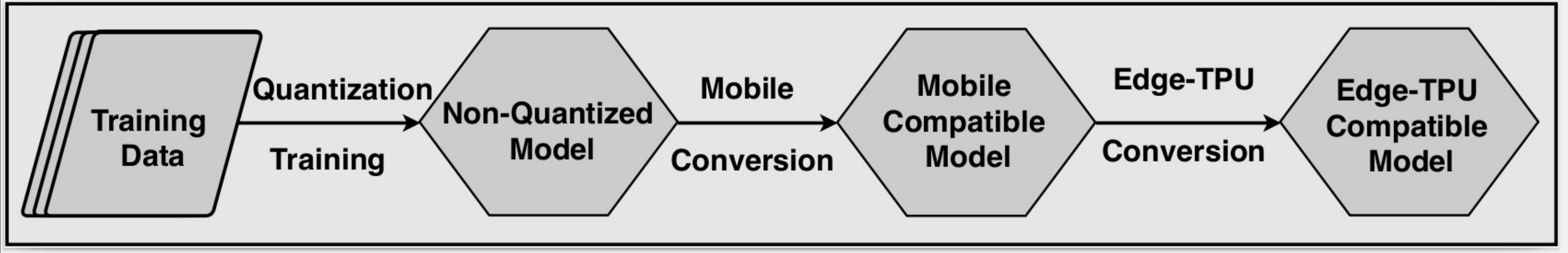
Operations	Layers
1x1 conv2D + Relu	3
3x3 Conv2D + Relu	13
Bilinear upsampling	3
Max Pooling	3
Concatenation (Channel Axis)	3

Post Quantization of Training Model [USED IN EXPERIMENTS]



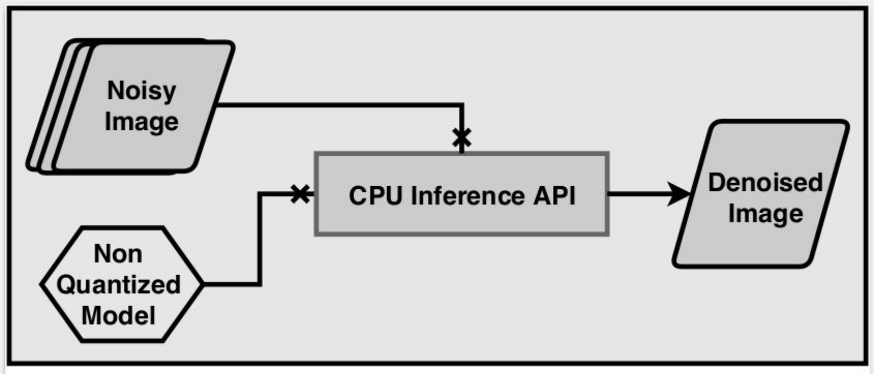
- Fast
- Model is modified after training (for quantization purpose)
- Works well with smaller and large models
- Trained for 40K epochs (24 hours training time)

Quantization-Aware Training Model



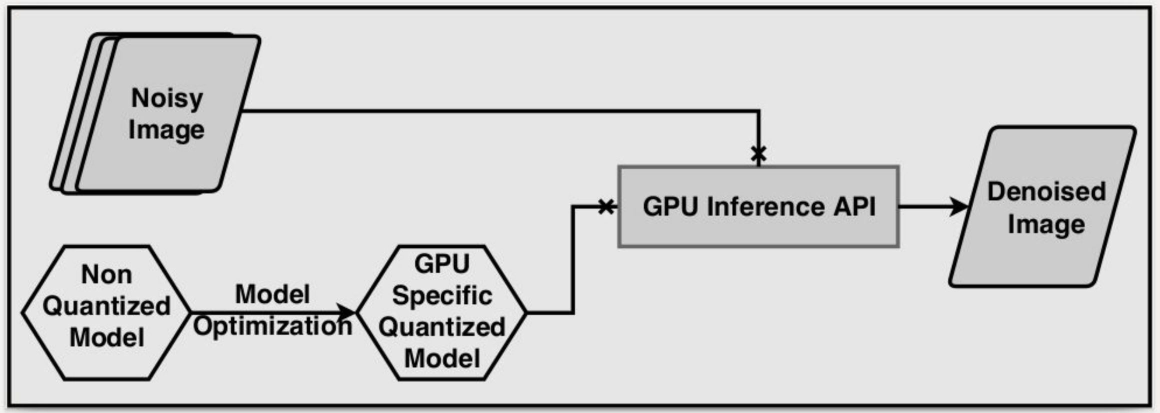
- Very slow
- Model configured with fake quantization layers
- Good for small models (observation with current experiments on GPU training)
- 24 hours to train for 1K epochs (low image quality @Testing)

Workflow 1: Inference with CPU



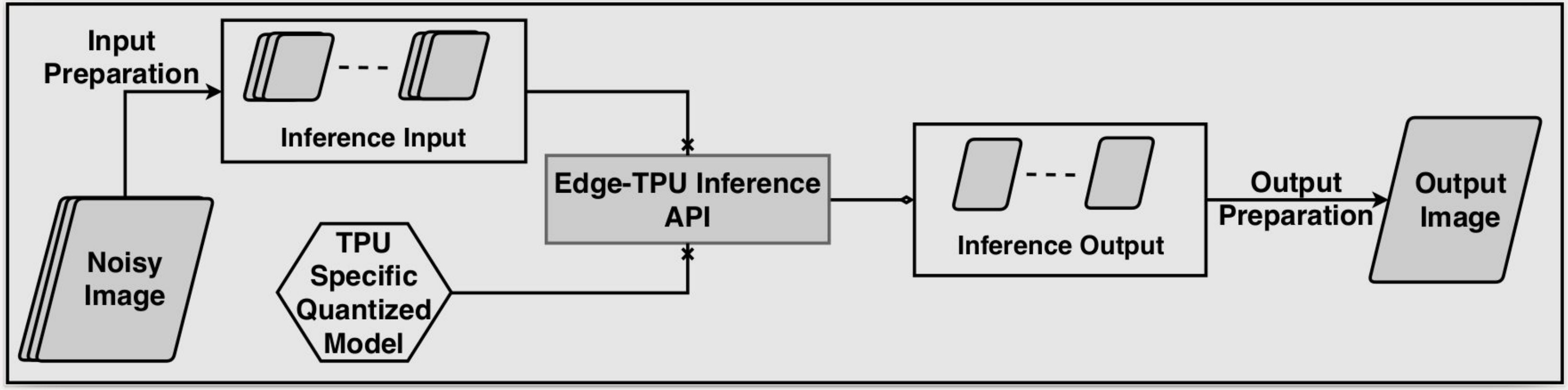
- Use Non-quantized model and do inference on CPU
- Records timing and quality stats
- Uses 1024 x 1024 input image and outputs 1024 x 1024 image

Workflow 2: Inference with GPU



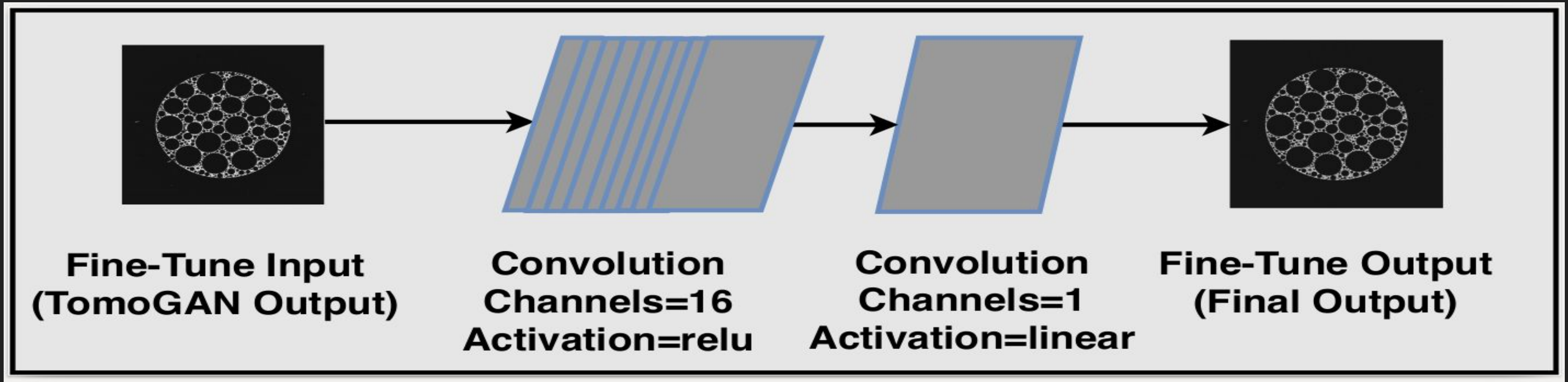
- Non-quantized model converted to GPU compatible quantization model
- Records timing and quality stats
- Uses 1024 x 1024 input image and outputs 1024 x 1024 image
- TensorRT support to run on Edge GPU [Tx2]

Workflow 3: Inference with TPU



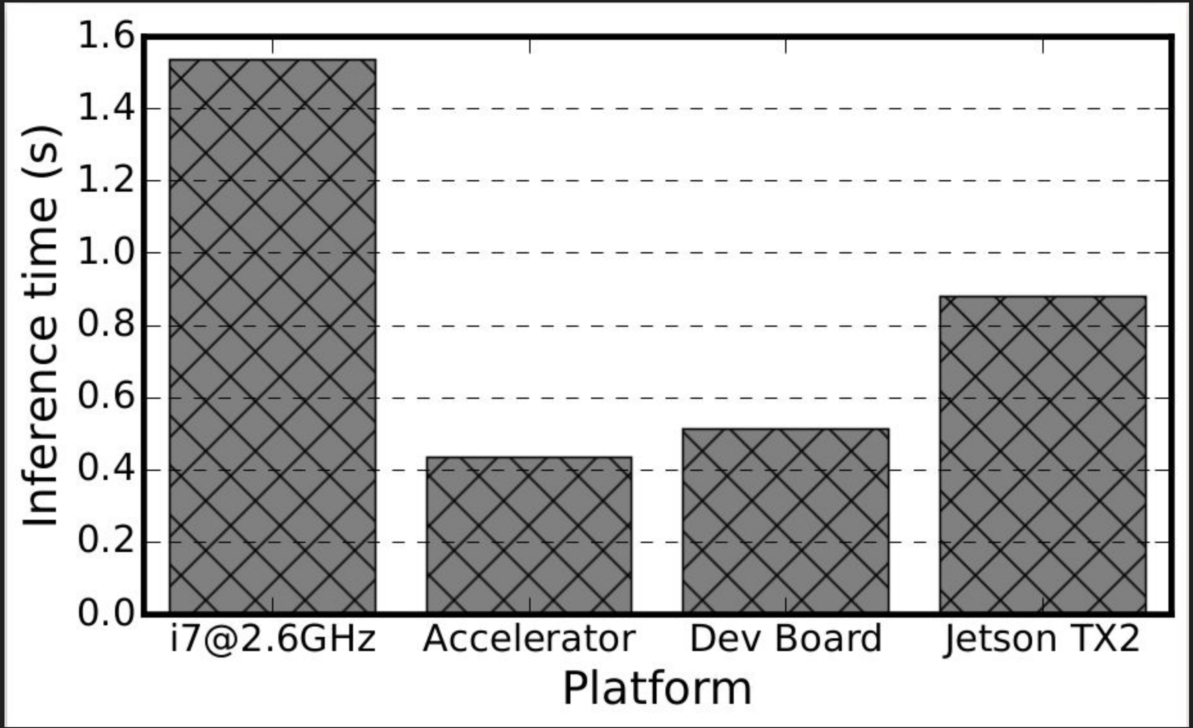
- 1024 x 1024 image => 64 x 64 x 256 (256 slices of 64 x 64 images)
- Model Quantization to TPU Compatibility requires a sample dataset.
- The sample dataset governs the quantization range
- Wrote custom wrapper (Specially for image to image translation) for BasicEngine (Tensoflow API modification)(currently not supported as an API in Tensorflow)
- Use Interpreter API (Extended API was designed for Image Restoration)

Fine Tune Layer



- Improves the Image Quality
- Shallow CNN Used
- Quantized for Edge-TPU Compatibility

Performance Evaluation on Inference



- TPU Accelerator is much faster than CPU
- TPU Accelerator performs faster than GPU Accelerator.
- Time taken to restore 1024x1014 image
- Tested for 1024 image set an average timing is recorded

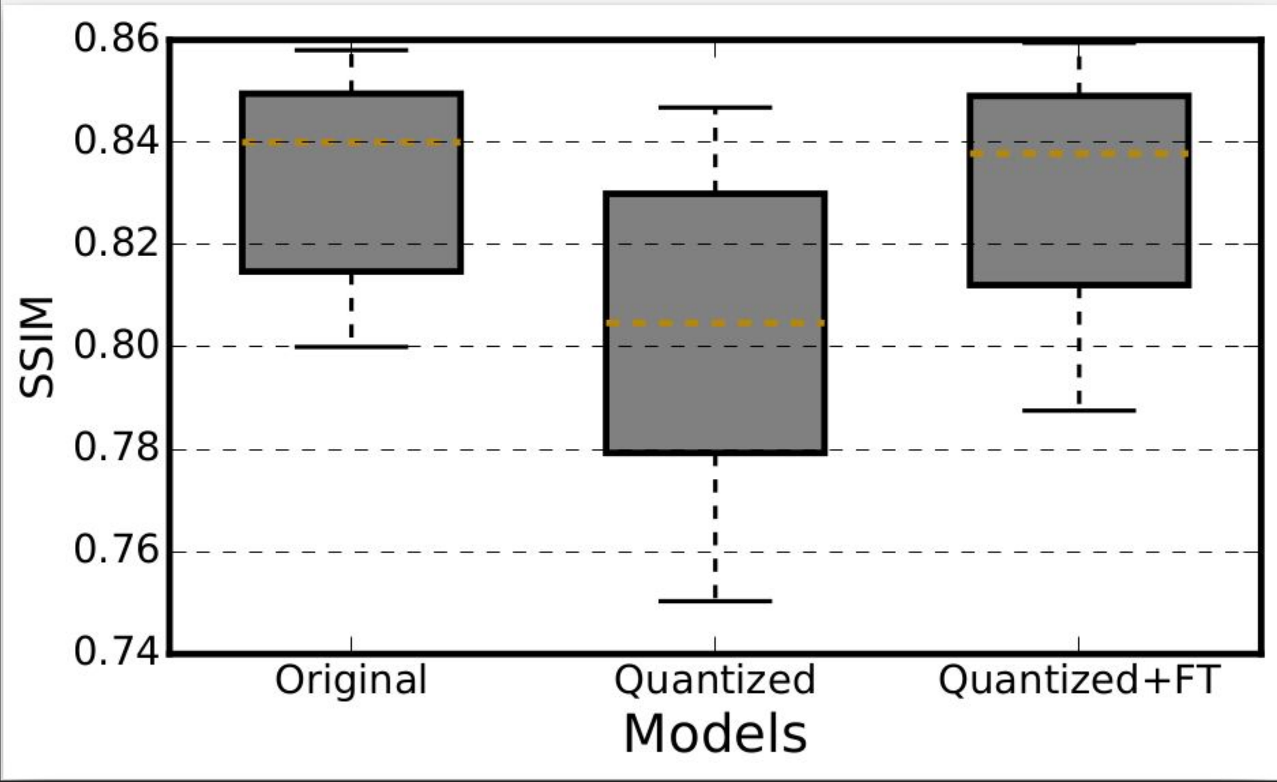
Accelerator was mounted on host machine[i7@2.6GHz]

Accelerator runs on High frequency mode >> Dev board on low frequency mode

Performance Time Breakdown

	Accelerator	Dev Board	Jetson Tx2
Quantized TomoGAN (s)	**0.435**	0.512	0.880
Stitching (s)	**0.049**	0.120	-
Fine-Tune (s)	**0.070**	0.166	-
Total (s)	**0.554**	0.798	0.880
Power Consumption (w)	2	2	7.5
Peak Performance	4 TOPS	4 TOPS	1.3 TFLOPS

Image Quality Evaluation with SSIM

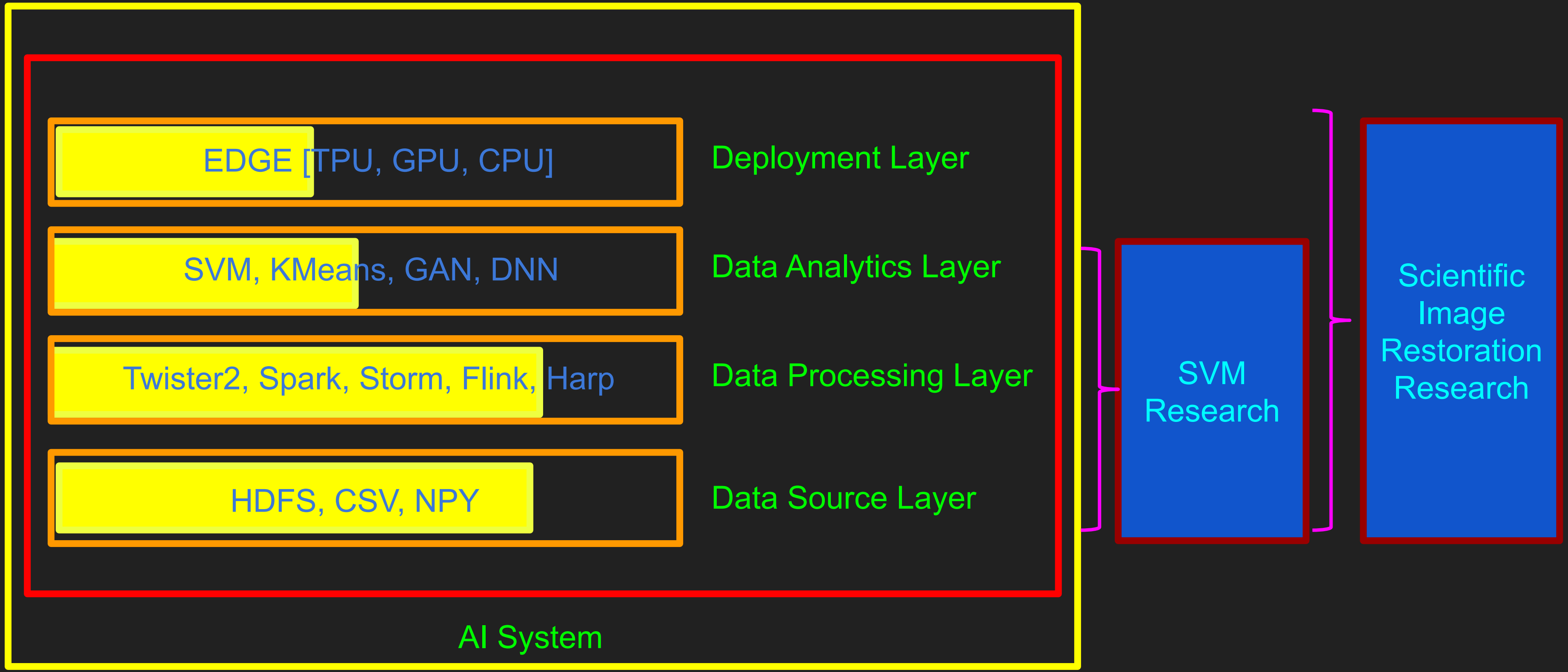


- Structural Similarity Index is used.
- Fine Tune Layer improves the Quantized model up to the image quality of a non-quantized model.
- Original model doesn't use edge an device. It is a non-quantized model.

Conclusion

- We design a faster inference workflow to restore scientific images (Edge TPU and Edge GPU).
- Enabled the inference workflow to generate high quality images even after quantization.
- Low cost solution with higher efficiency.
- High accessibility to scientific application developers.
- Designed a portable system which can run anywhere.
- First research approach on image translation on Edge TPUs

Overall Summary



Thank You