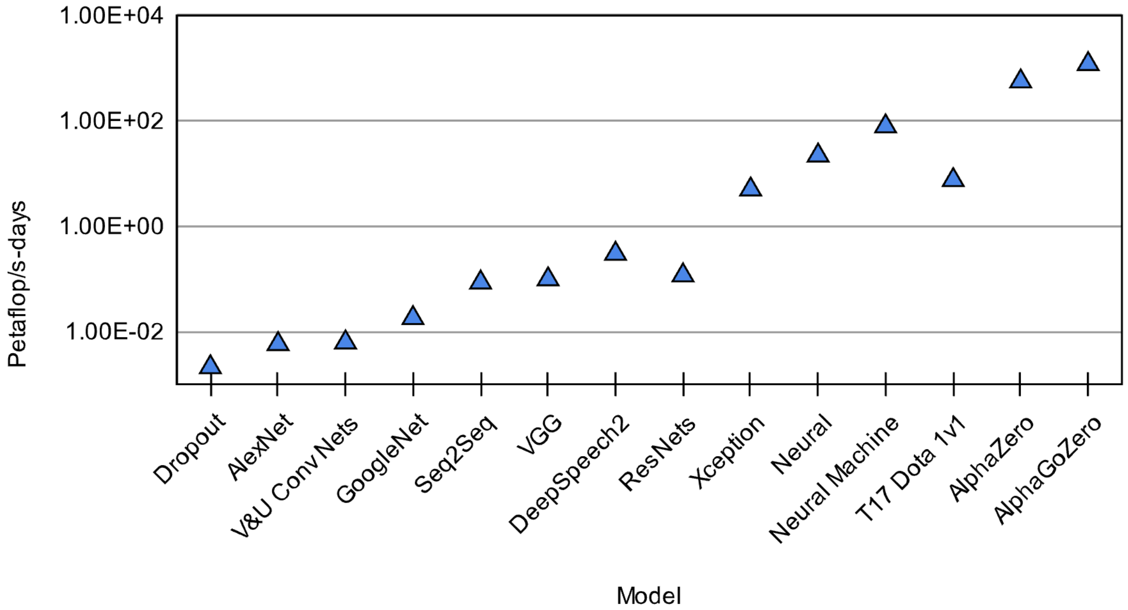
# NeSSA: Near-Storage Data Selection for Accelerated ML Training

**Neha Prakriya, Yu Yang, Baharan Mirzasoleiman, Cho-Jui Hsieh, Jason Cong**

University of California, Los Angeles

# Motivation

- **Training a GPT-3 on 45 TB of data:**
  - 💰 12 M
  - ⏰ 34 days on 1024 A100 GPUs
  - 🏠 17.5x the average yearly energy consumption of one American house.
  - 🚭 $CO_2$ release of a car driving 2x the distance between the Earth and the Moon.



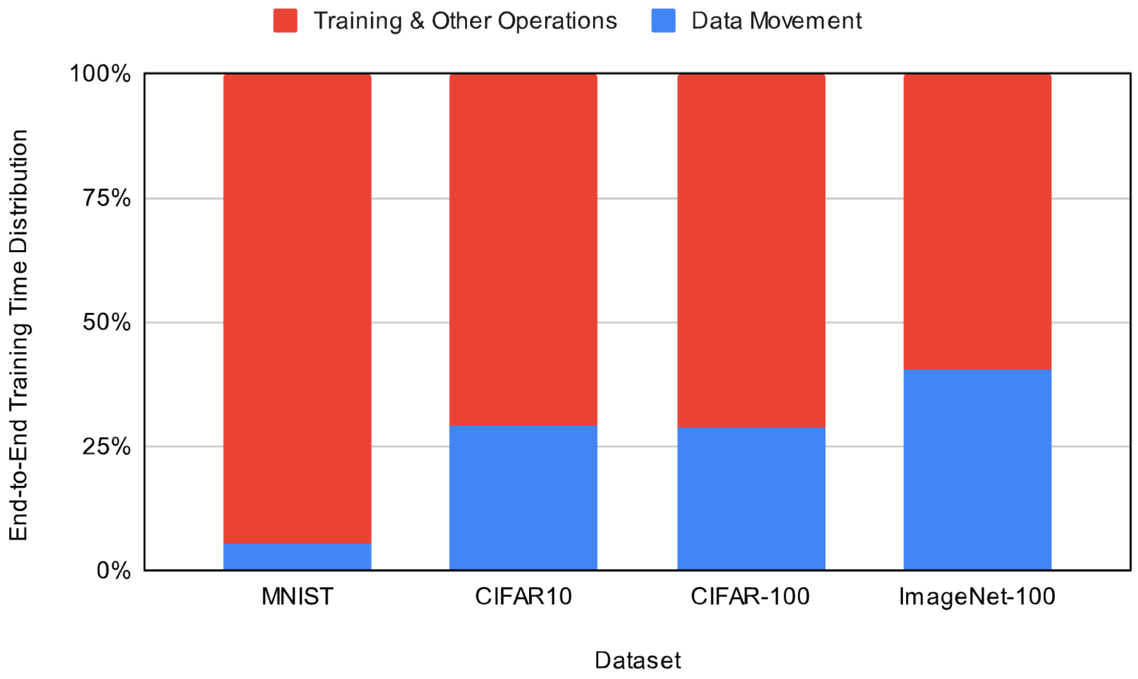

Training time of image classification models has been doubling every 3.4 months (OpenAi, 2018)

- **Grand challenges in ML (Asi & Duchi, 2019)**

# Contributors to Training Cost

- **Two main bottlenecks:**
  - Number of gradient computations
  - Data movement and I/O cost

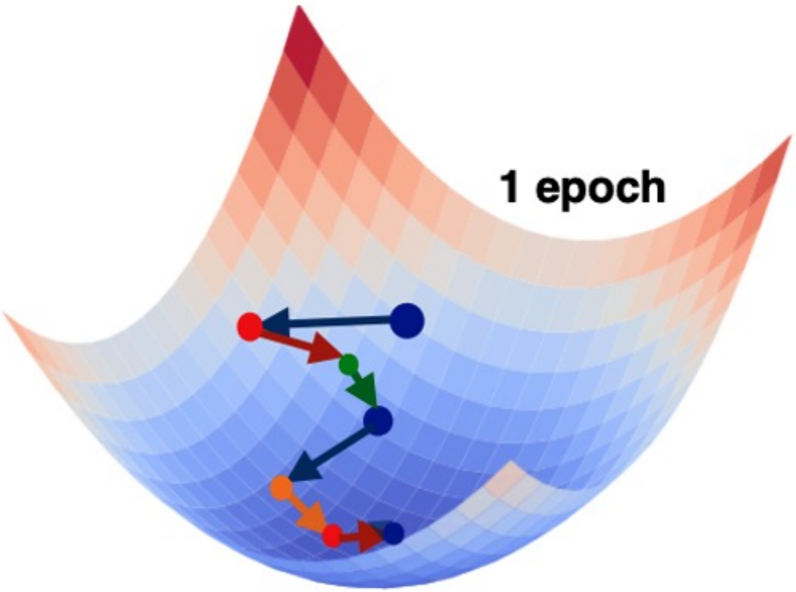
**Is it equally important to train on every data point?**


Distribution of training time for training a ResNet50 model using an NVIDIA V100 GPU.

# Subset Selection

- Training dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$
- The goal of training is to find optimal parameters $\theta$ of a model $\Psi(.\,;\theta)$ such that:
- $\theta * = min \frac{1}{N} \sum_{i=1}^{N} L(\Psi(x_i;\theta), y_i)$
- Goal: Find subset $S \subseteq D$ such that:
  - $S = \min|S|\ st.$
  - $max_\theta \parallel \sum_{i \in D} \nabla L_i(\theta) - \sum_{j \in S} \nabla L_j(\theta) \parallel \ \leq \epsilon$, where $\epsilon \geq 0$.




Loss function

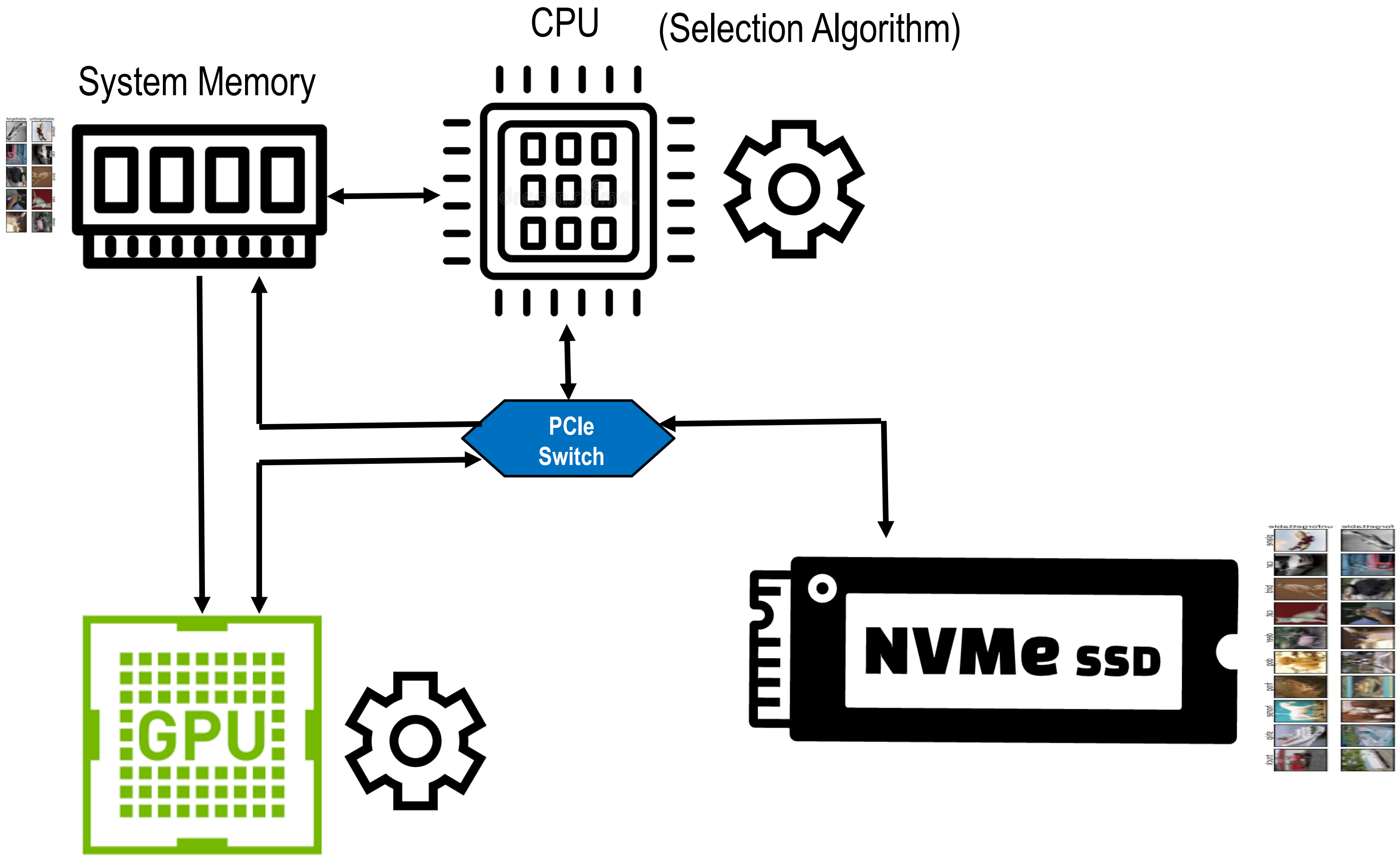
# Subset Selection – Assigning Importance

| Selection Method | Key Idea | Pros | Cons | Examples |
|---|---|---|---|---|
| Trained models | Infer importance post-training | High accuracy | Incurs more gradient computations than model trained on all data samples. | Toneva, ICLR'19<br>Zhang, NeurIPS'19<br>Coleman, ICLR'20<br>Zhao, ICLR'21<br>…. |
| Training dynamics | Infer importance during training – loss values, clustering | Low cost solution | Accuracy degradation | Sener, ICLR'18<br>Katharopoulos, ICLR'18<br>Mirzasoleiman, ICML'20<br>…. |

Different methods of assigning importance.

# Prior Work – Limitations

- **Limitation 1: High data movement.**
- **Traditional subset selection:**
  - Load data from disk to CPU memory
  - Run selection algorithm to assign importance.
  - Pass selected data samples to the GPU.
  - Train on the selected data samples.
  - Repeat every epoch



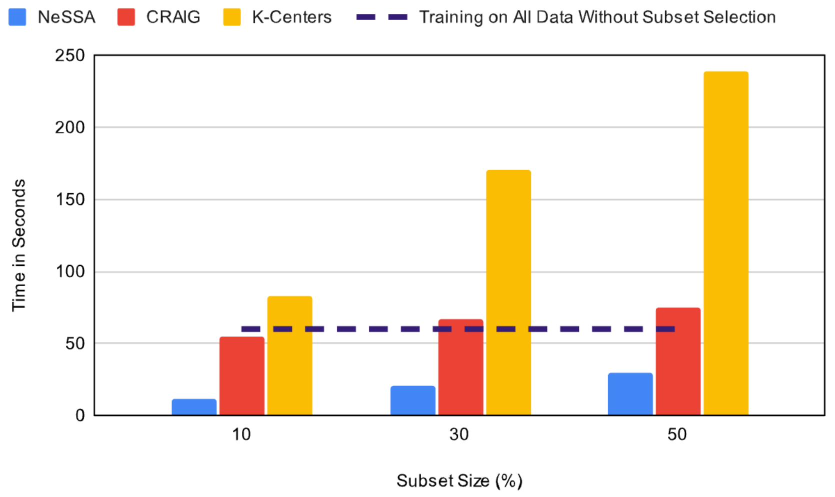

Steps involved in traditional subset selection

# Prior Work – Limitations

- **Traditional subset selection using training dynamics:**
  - Limitation 2: CPU-based selection – High selection time
  - Limitation 3: Limited information - Accuracy degradation

| Subset (%) | CRAIG | K-Center | NeSSA | Goal |
|---|---|---|---|---|
| 10 | 87.07 | 65.72 | **87.75** | 92.44 |
| 30 | 89.12 | 88.49 | **90.68** | 92.44 |
| 50 | 90.32 | 90.14 | **91.92** | 92.44 |

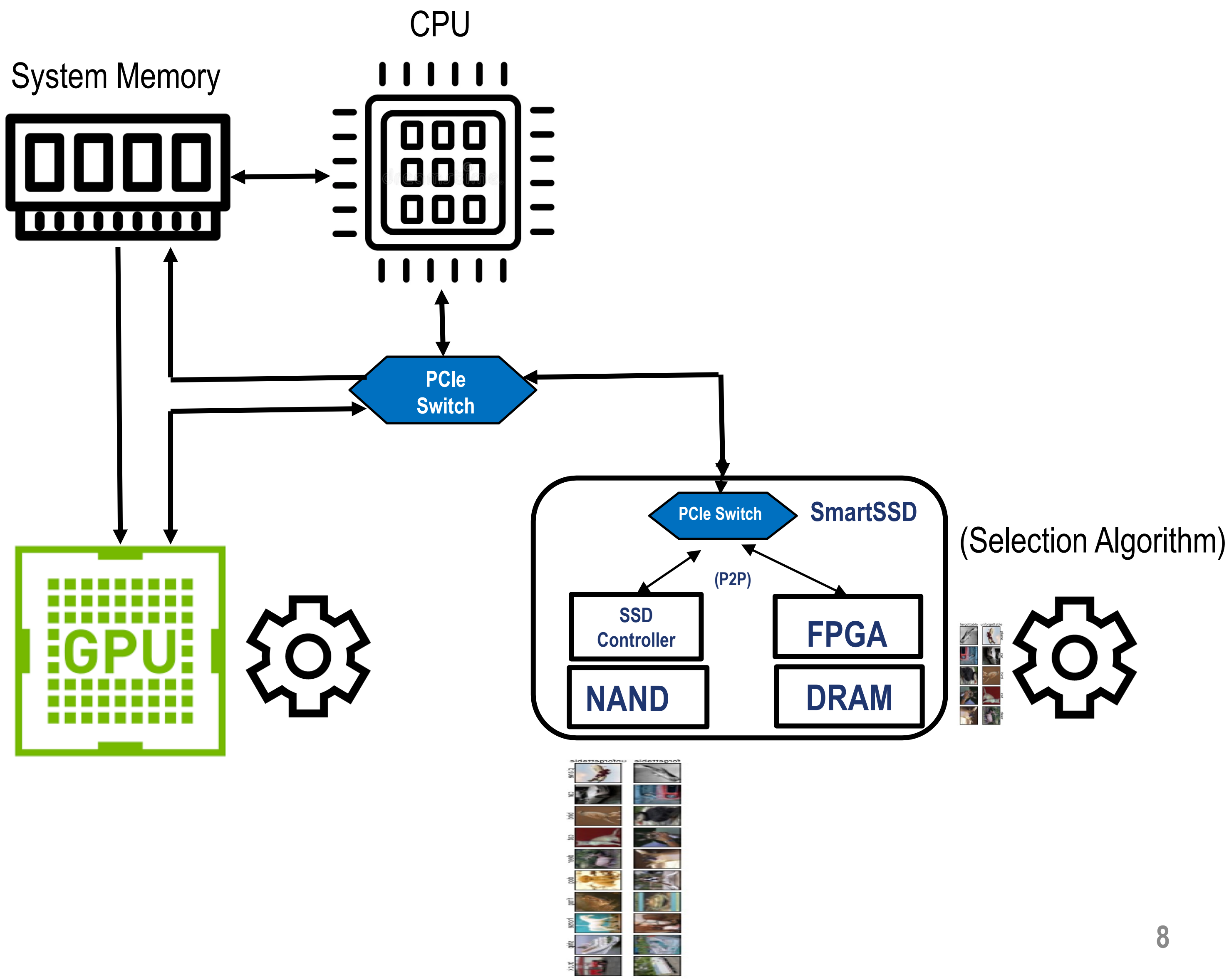
Accuracy when trained on different subset sizes on the CIFAR10 dataset using a ResNet20 model.




Training time averaged across epochs for NeSSA, prior work, and a model trained on the full dataset.

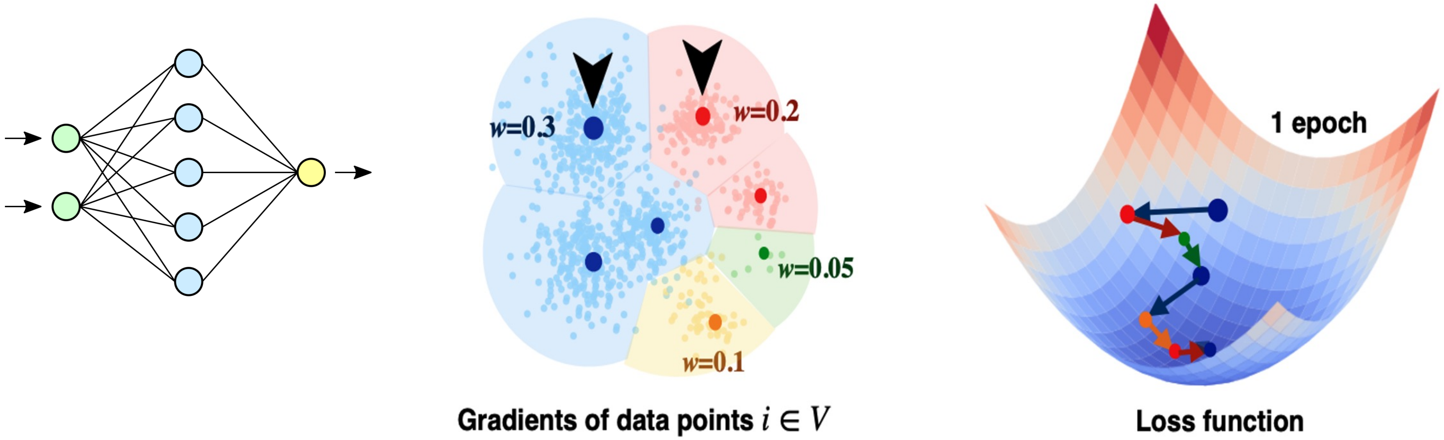
# NeSSA System Design

- **Subset selection using FPGA-based near-storage acceleration:**
  - Reduces data movement by |D|/|S|
  - High-speed selection compared to CPU-based selection
  - Energy efficient compared to GPU-based selection
  - Reconfigurable and scalable for different models and datasets compared to ASIC-based selection



# Selection Algorithm – High Accuracy, Low-Cost

- **Goal: Find subset $S \subseteq D$ such that:**
- $S = \min|S|\ st.$
  - $max_\theta \parallel \sum_{i \in D} \nabla L_i(\theta) - \sum_{j \in S} \nabla L_j(\theta) \parallel \ \leq \epsilon$, where $\epsilon \geq 0$.
- **Upper bound:**
  - $min_{S \subseteq V} \parallel \sum_{i \in C} \nabla L_i(\theta) - \sum_{j \in S} \nabla L_j(\theta) \parallel \leq \sum_{i \in D} min_{j \in S} \parallel \nabla L_i(\theta) - \nabla L_j(\theta) \parallel$
  - RHS: k-medoids problem
  - S is the set of medoids!

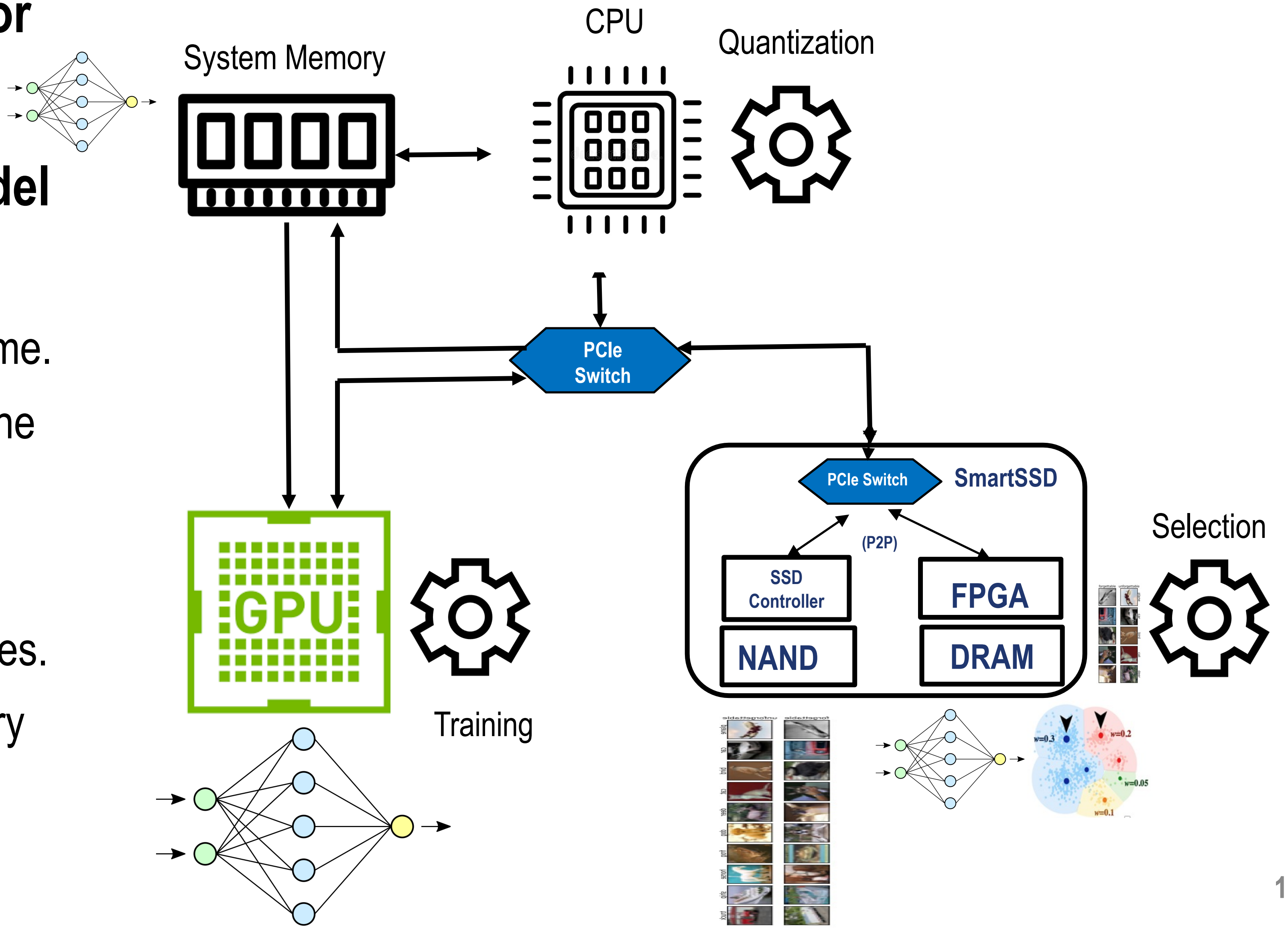



1

Overview of the selection algorithm

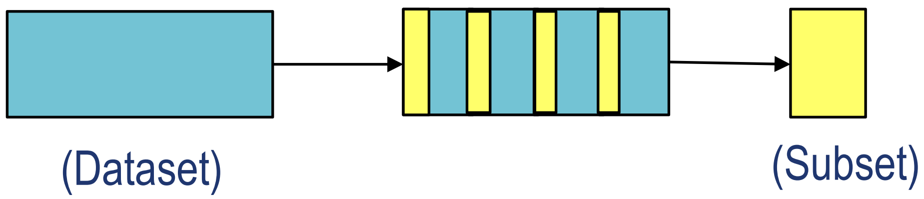
# Software Optimizations – High Accuracy, Speed, Minimum Subset Size

- **Quantize model on FPGA for inference.**
- **Feedback of quantized model weights:**
  - Improve selection model over time.
  - Select only those points which the model needs in that epoch.
- **Subset biasing:**
  - Selecting from unlearned samples.
  - Drop samples with low loss every 20 epochs.



# Hardware Optimizations – High-Speed Selection, Low-Cost

- **Quantization of model weights:**
  - 1-bit weights
  - 2-bit activations
  - 4-bit residuals
  - 8-bit first / last layer weights
- **Dataset partitioning:**
  - Randomly partition dataset into several chunks and select a smaller subset from each chunk.
  - No need to fit gradients of an entire class onto on-chip memory.
  - Example:
    - ❖Mini-batch size m, subset size k, dataset size N
    - ❖Partition dataset into k/m random chunks
    - ❖Select m examples from each chunk

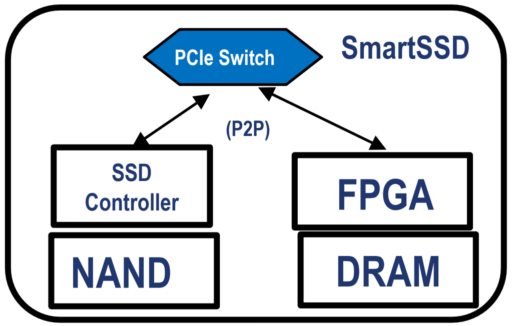


# Evaluation Setup

- **Datasets and models evaluated:**

| Dataset | Classes | Number of training samples | Network |
|---|---|---|---|
| CIFAR-10 | 10 | 50K | ResNet-20 |
| SVHN | 10 | 73K | ResNet-18 |
| CINIC-10 | 10 | 90K | ResNet-18 |
| CIFAR-100 | 100 | 50K | ResNet-18 |
| TinyImageNet | 200 | 100K | ResNet-18 |
| ImageNet-100 | 100 | 130K | ResNet-50 |

- **GPU used: NVIDIA A100**
- **SmartSSD v1.0:**
  - 3.84TB NAND
  - Xilinx Kintex UltraScale+ KU15P FPGA
  - 4GB DDR4 SDRAM



# Performance – Accuracy, Convergence Speed-Up

| Dataset | All data (%) | NeSSA (%) | Subset (%) |
|---|---|---|---|
| CIFAR-10 | 92.02 | 90.17 | 28 |
| SVHN | 95.81 | 95.18 | 15 |
| CINIC-10 | 81.49 | 80.26 | 30 |
| CIFAR-100 | 70.98 | 69.23 | 38 |
| TinyImageNet | 63.40 | 63.66 | 34 |
| ImageNet-100 | 84.60 | 83.76 | 28 |

Accuracy comparison between NeSSA and training on the full data.

# Impact of Each Optimization

- **Vanilla: Medoid-based selection without any optimizations.**
- **SB: medoid-based selection with subset biasing.**
- **PA: medoid-based selection with dataset partitioning.**
- **SB+PA: Medoid-based selection with both optimizations.**
- **Goal: Accuracy when trained on the full dataset.**

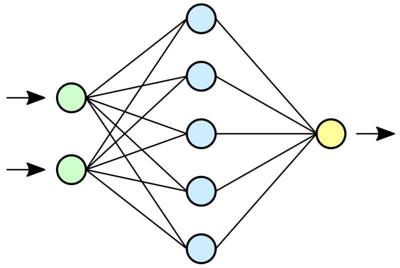
| Subset (%) | Vanilla (%) | SB (%) | PA (%) | SB+PA(%) | Goal (%) |
|---|---|---|---|---|---|
| 10 | 82.76 | 87.61 | 83.75 | 87.75 | 92.44 |
| 30 | 89.51 | 90.42 | 90.68 | 90.42 | 92.44 |
| 50 | 90.59 | 91.81 | 91.91 | 91.92 | 92.44 |

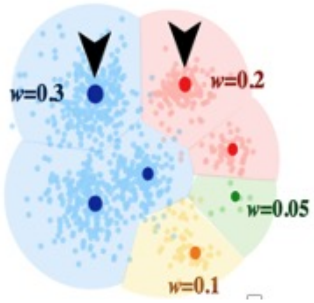
Impact of each optimization when training a ResNet20 model on the CIFAR-10 dataset.

# Accelerator Design for Selection

- **Inference accelerator generated using FINN compiler:**
  - Deep neural network inference for FPGAs
  - Dataflow-style quantized neural networks
  - Takes as input ONNX model trained in Brevitas:
    - ❖Pytorch library for quantization-aware training.

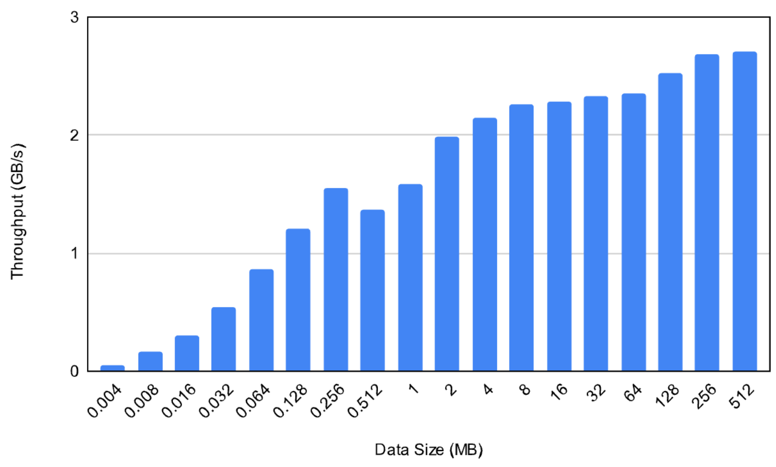
| Resource | Available | Utilization (%) |
|---|---|---|
| LUT | 432K | 67.53 |
| FF | 919K | 23.14 |
| BRAM | 738 | 50.30 |
| DSP | 1962 | 42.67 |





# Benefits of Using FPGA-Based Near-Storage Acceleration

- **4.3x faster than CPU-based selection.**
- **Without P2P between SSD and FPGA:**
  - Achievable bandwidth reduces from 3GBps to 1.4GBps.
- **Overall reduction of data movement over host-drive interconnect by an average of 3.5x.**
- **Effects of increasing dataset size:**
  - ❖CIFAR-10: 0.003MB/image, throughput: 1.46GBps
  - ❖ImageNet-100: 0.126MB/image, throughput: 2.28GBps.
- **As dataset size increases, storage-assisted training becomes more effective and essential.**
- **Overall end-to-end training speed-up of 5.4x.**




Data transfer throughput between FPGA and on-board SSD on SmartSSD

# Conclusion

- **Motivation:**
  - Significantly reduce model training costs without affecting final model accuracy.
- **Key Ideas:**
  - Use FPGA-based near-storage data selection to reduce training & data movement costs.
  - Use feedback from target model to improve selection.
  - Automatically reduce subset size over time.
  - Quantize selection model to improve speed.
- **Key Results:**
  - Data movement reduction of 3.5x.
  - Training speed-up of 5.4x.

# Acknowledgements

This research was supported by PRISM Award CC OTH 00541340 2022 TR and CRISP Award GI18518.156870. We would like to thank Samsung

Electronics Co. for providing access to the SmartSSD devices. This work was supported in part by the Semiconductor Research Corporation (SRC) and DARPA.

BM was supported by the National Science Foundation CAREER Award 2146492. CJH was supported by the National Science Foundation Award IIS-2008173, IIS-2048280 and Sony Research Award.

Repository:

Link: https://github.com/nehaprakriya/Near-SSD-Data-Selection

# Thank you!
# Questions?