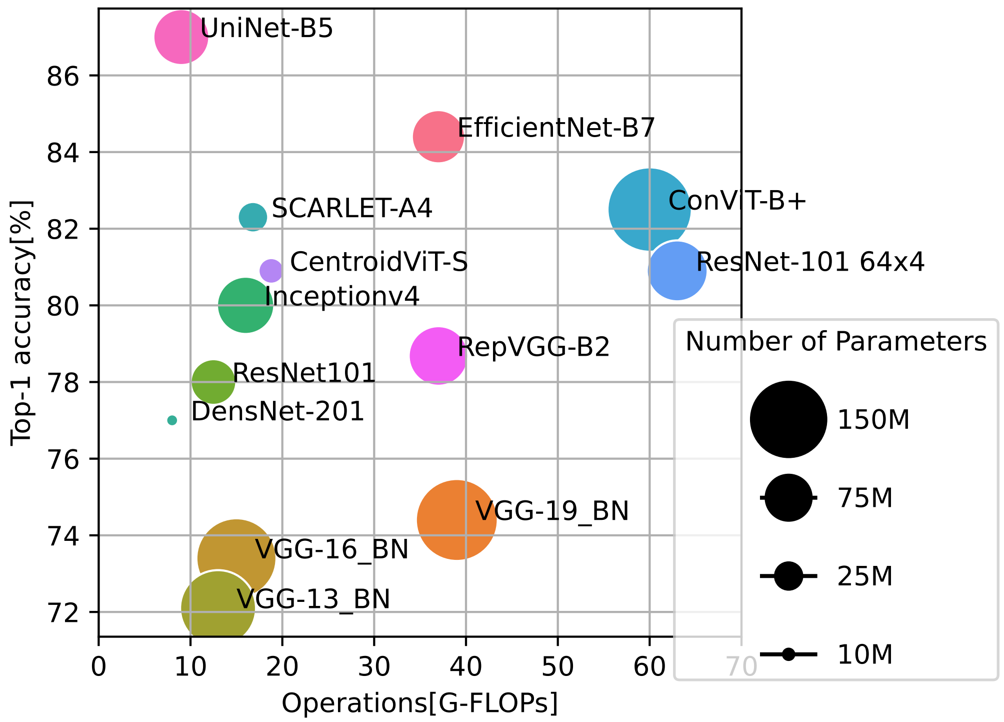
Robust and Tiny Binary Neural Networks using Gradient-based Explainability Methods

Muhammad Sabih[1], Mikail Yayla[2], Frank Hannig[1], Jürgen Teich[1], and Jian-Jia Chen[2]

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg, [2] Technische Universität Dortmund

08.05.2023

Outline



Complexity and Over-parameterization

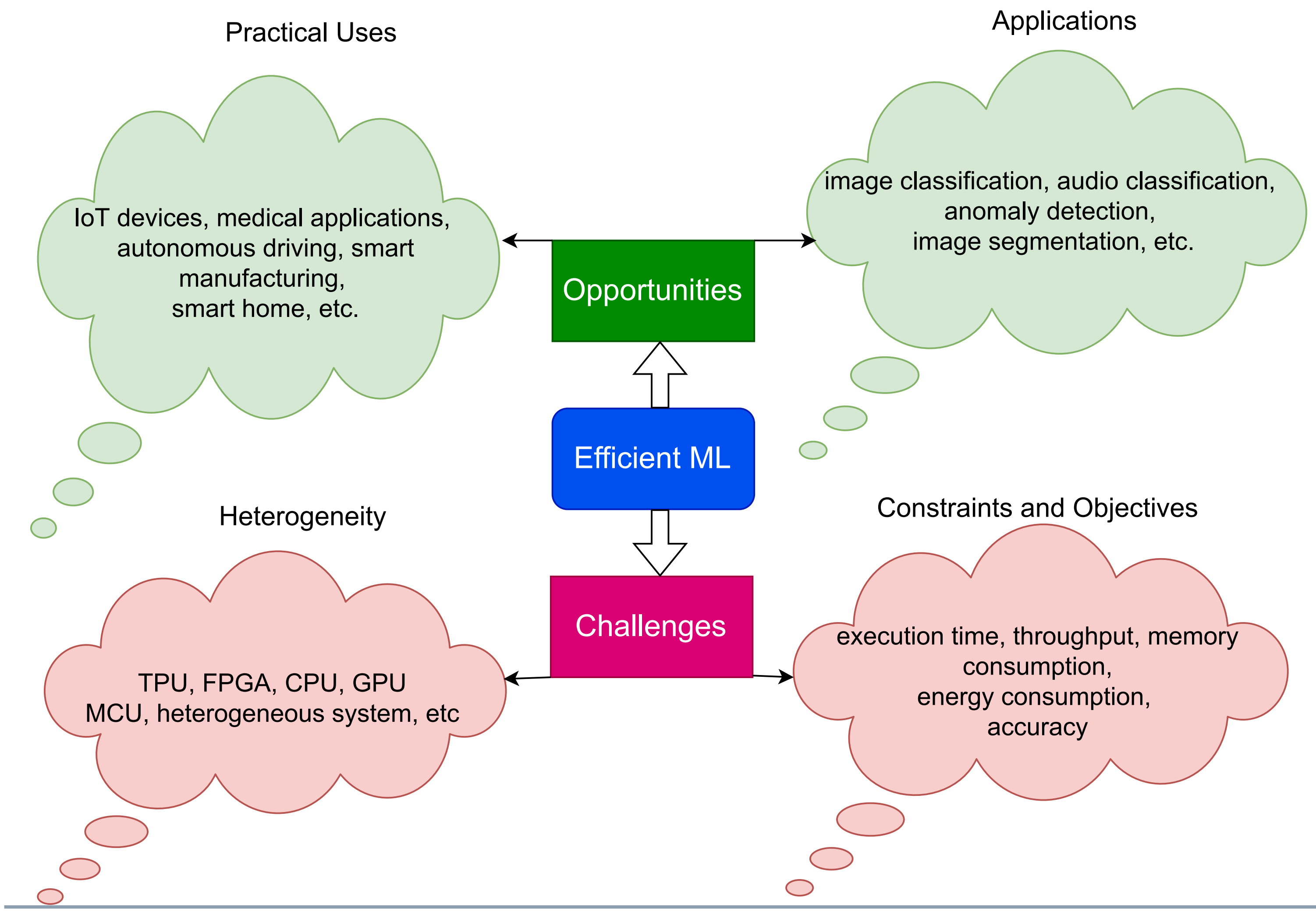
- Deep Learning applications have high computational complexity.
- Most DL methods are typically *over-parameterized*.

Solution

- *Efficient ML* is needed to overcome complexity.
- Overcoming complexity is necessary for successful application of ML and DL.

Efficient Machine Learning

Some of the Applications, Challenges, and Methods



Efficient ML

"Efficient ML" can be defined as an area in machine learning focussed on reducing the resources needed for deploying ML applications on a target.

Some Efficient ML approaches

- **Approximate Computing**
- **Pruning**
- Quantization
- NAS
- Efficient Compilation

BNN

- A special type of NNs that only use binarized weights and activations.
- Theoretically, using a BNN should yield 32x speedup. Practically, speedups upto 23x have been achieved [Hub+16].
- Training of BNNs is practical for simpler applications like image classification.
- BNNs are more robust than their counterparts.
- Robustness of BNNs allows them to be implemented on approximate memory systems providing various benefits.
- Multiplications of binary weights with inputs/activations can be done with XORs.

Approximate Memory Systems

- Voltage may be adjusted to save energy.
- Yields high bit-error rates.

DNN Pruning

Desired characteristics for our Pruning Method

DNN Pruning

DNN Pruning refers to removing or zeroing undesired weights of a DNN.

Desired characteristics of a DNN pruning method

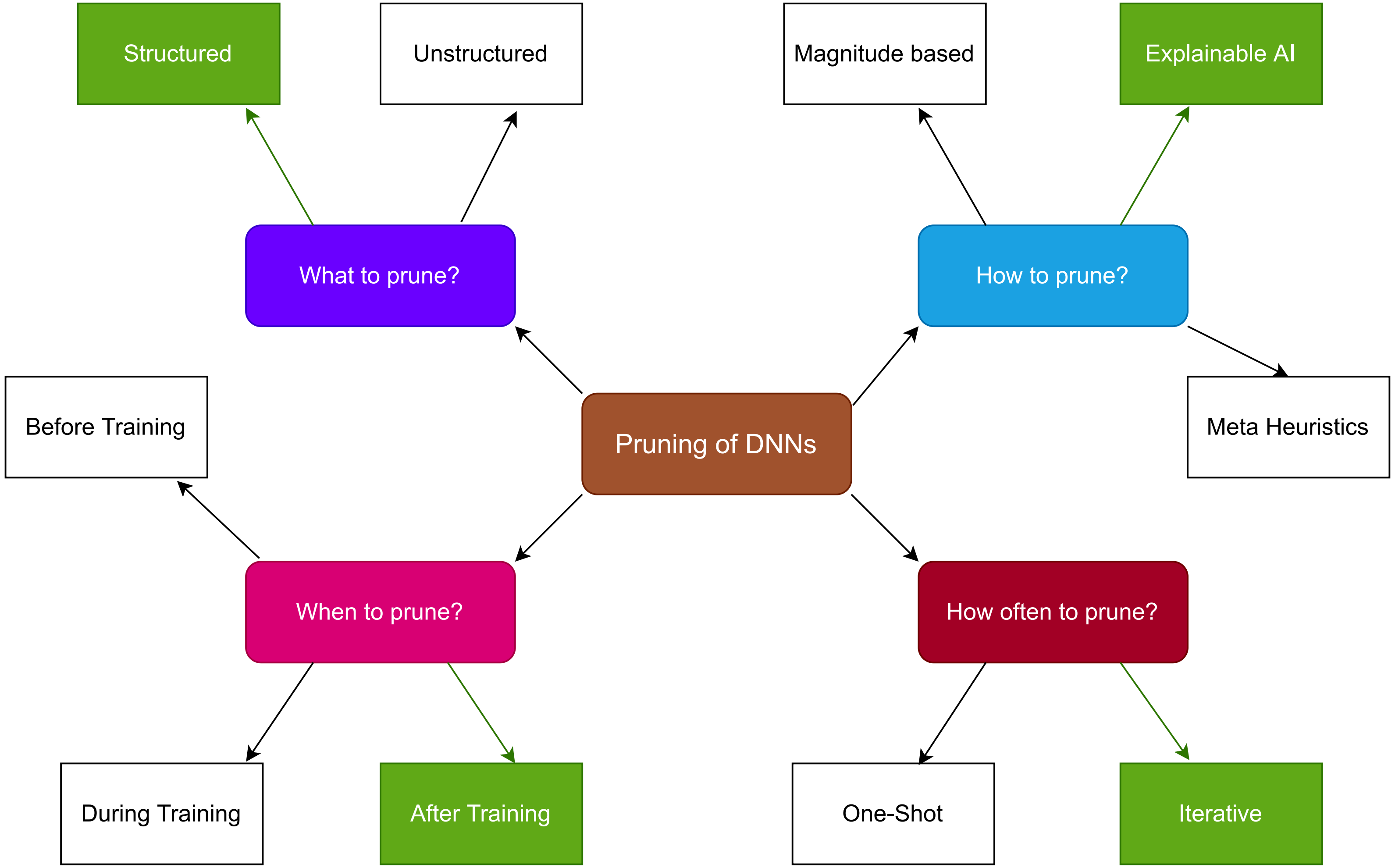
- Can support diverse hardware platforms.
- Can optimize for multiple target objectives.
- Has a reasonable search time.
- Utilizes explainability or model characteristics.
- Has robustness and generalizability.

An Overview of DNN Pruning

Constituents of a DNN Pruning Algorithm

technische universität dortmund

FAU



What to Prune?

Structured vs. Unstructured Pruning

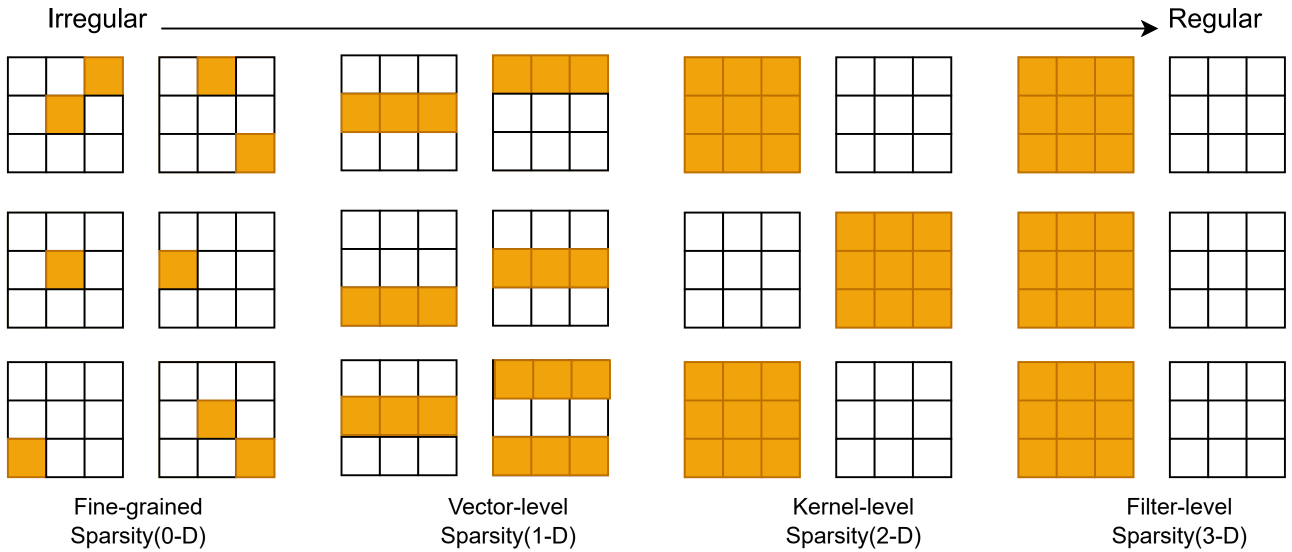


Figure: Colored boxes indicate pruned weights

Pros and Cons of Structured Pruning

- Structured Pruning can be easily accelerated on most or all target platforms with little or no overhead.
- Results in DNN accuracy degradation, therefore *prunability* is less.

Pros and Cons of Unstructured Pruning

- Obtaining gains from Unstructured Pruning may require special target hardware, device-specific code, and overhead.
- Typically preserves DNN accuracy better than structured pruning.

What is Explainable AI (XAI)?

Explainable AI means to explain why a neural network makes a particular decision depending upon the input it gets and the output it produces.

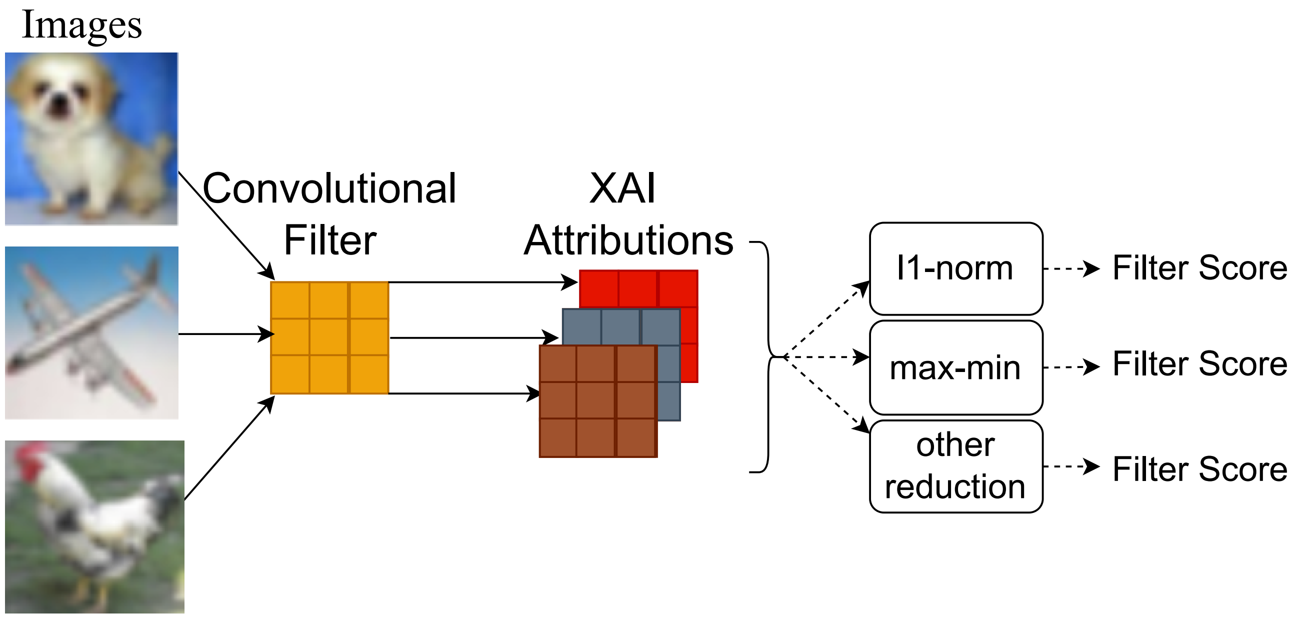
XAI as ranking criterion for DNN Pruning

XAI methods that can be used to obtain a measure of the importance of filters, channels, or weights, are used as ranking heuristic in DNN Pruning.

Some XAI methods used as a ranking criterion for DNN Pruning

- Layer-wise Relevance Propagation
- Taylor Expansion
- DeepLIFT
- Layer Conductance
- Integrated Gradients

How Ranking is Obtained?



Ranking scores from XAI methods

1. XAI methods that we considered give us *"attributions"* having same dimensions as activation maps.
2. A subset of the validation set is used to obtain attributions.
3. Various reductions can be used to obtain a single score for a filter.
4. Multiple reductions can also be combined in a meta-heuristic.

Gradient-Based Methods for Tiny BNNs

Recipe for **Our** Iterative Pruning Approach

1. Pruning amount in every iteration

Application-specific

2. Layer-wise pruning amount in each iteration

Sensitivity Analysis using real measurements from the target hardware for respective objectives followed by a small exploration step. This distributes the number of filters to prune in respective layers.

3. Ranking filters

Explainable AI-based ranking criteria that ranks filters in every layer and removes them.

4. Training recipe

Application-specific

5. When to stop

Application-specific

Guided FAT

- Fault-Aware Training (FAT) is used for Robust BNNs.
- Our idea is to guide FAT to inject lesser faults to important neurons.
- Serves to **verify** the Gradient-Based methods.
- Also serves to provide better robustness for some BER regions.

Evaluation Setup

Hardware

1. GPU
   - NVIDIA Titan RTX with 24 GB of memory
2. CPU
   - Intel i7-9700 processor

Software

1. PyTorch
2. Captum
3. Intel Distiller
4. Torch Pruning

Metrics

1. Accuracy
2. MACs
3. Robustness.

Models and Datasets

Datasets

1. CIFAR10
   - Popular Image Classification datasets with 10 classes. Each image has dimensions of $3 \times 32 \times 32$.
2. FashionMNIST
   - An Image Classification dataset with 10 classes and images of dimension $1 \times 28 \times 28$.
3. Google Speech Commands v2
   - A Key Word Spotting (KWS) dataset consisting of 35 spoken classes like “down”, “up”, “right”, “left”, etc.
   - We sample 10 classes from the dataset and use feature extraction to obtain $1 \times 32 \times 32$ images.

Models

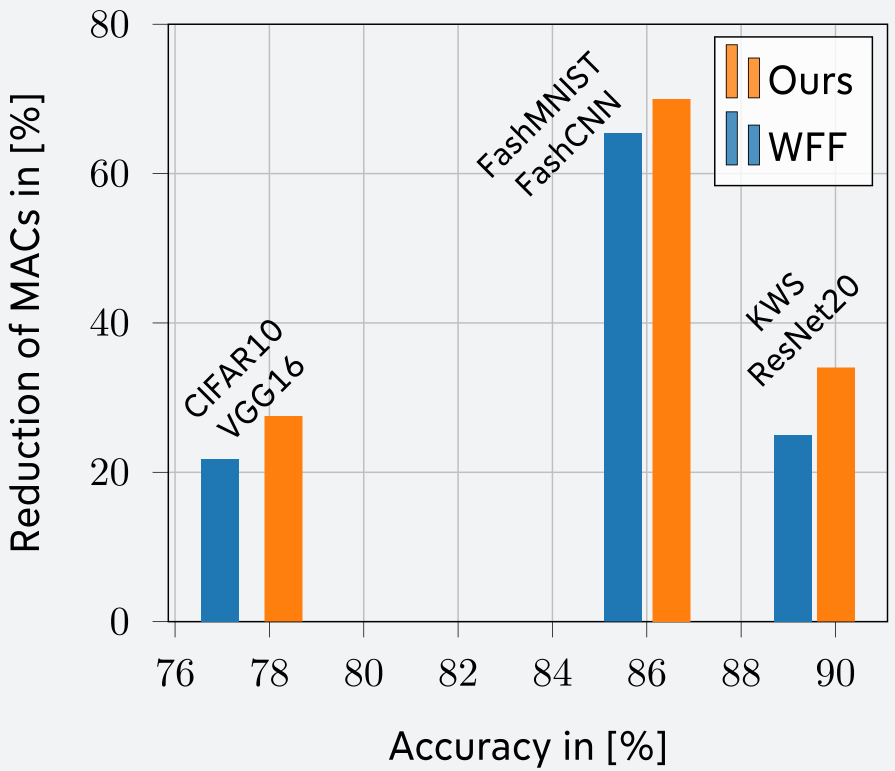
1. VGG16
2. ResNet20
3. FashCNN

Evaluating Pruning Criteria for BNNs

WFF (Weight Flipping Frequency) criteria

WFF measures the importance of neurons by observing the flipping frequency of a weight [LR20].

Ours vs. WFF (Weight Flipping Frequency)

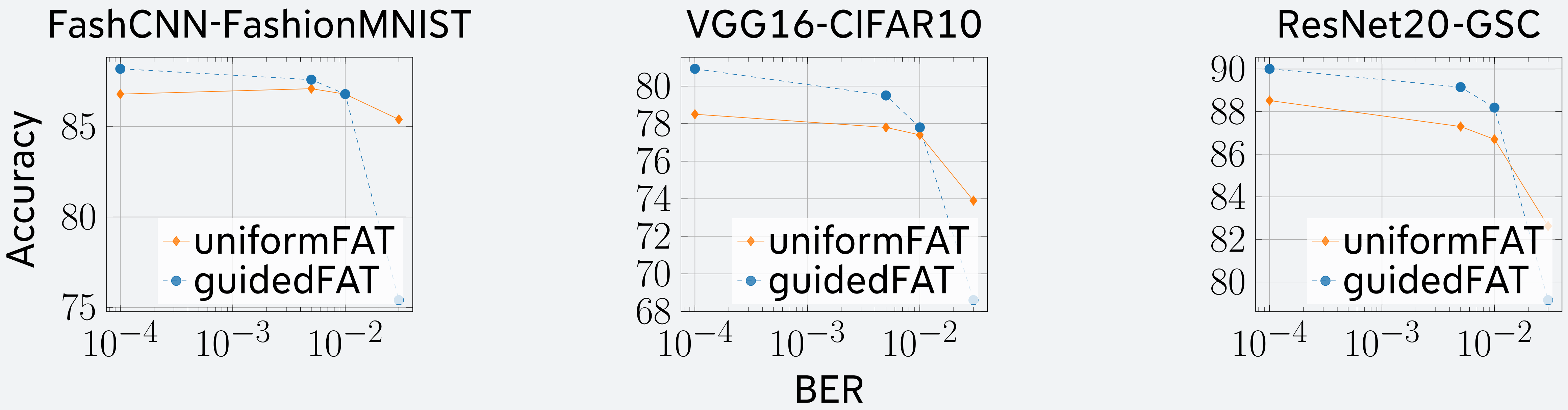


Evaluating guidedFAT

Setup

Both uniformFAT and guidedFAT are trained with a BER of 10^{-2} and tested using uniform error injections.

GuidedFAT vs. uniformFAT



1. Introduction and Motivation

2. Background Work

3. Proposed Approach

4. Evaluation

5. Conclusion and Future Work

6. Questions

Conclusion

Conclusion

- We utilize XAI Gradient-Based methods for:
  - Pruning BNNs.
  - Guiding Fault-Aware-Training (FAT).
- We compare our approach with previous works and demonstrate the benefits.

Future Work

- Implementation of of tiny and robust BNNs on target with approximate memory.

1. Introduction and Motivation
2. Background Work
3. Proposed Approach
4. Evaluation
5. Conclusion and Future Work
6. **Questions**

[Hub+16] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. “Binarized Neural Networks”. In: *Proc. of the Conference on Advances in Neural Information Processing Systems*. 2016, pp. 4107–4115.

[LR20] Y. Li and F. Ren. “BNN Pruning: Pruning Binary Neural Network Guided by Weight Flipping Frequency”. In: *Proc. of the International Symposium on Quality Electronic Design*. 2020, pp. 306–311. DOI: 10.1109/ISQED48828.2020.9136977.