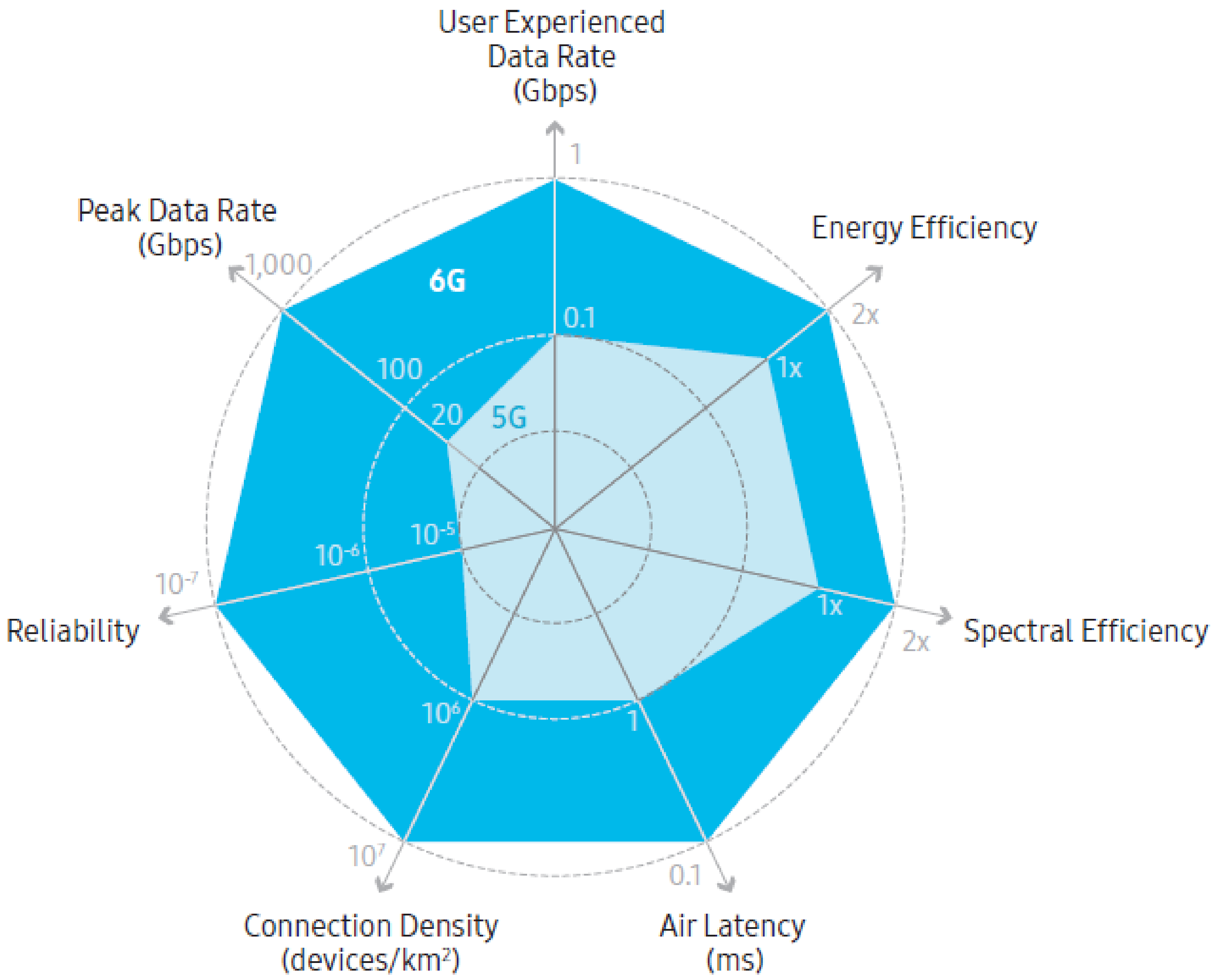
ITU-ML5G-PS-007
Lightning-Fast Modulation Classification With Hardware-Efficient Neural Networks

Aaronica Team

Mohammad Chegini [1] , Pouya Shiri [2] , Amirali Baniasadi [3]

International Telecommunication Union
Organized by ***Xilinx***

[1] Shahid Beheshti University. mohammad.chegini@hotmail.com
[2] University of Victoria. pouyashiri@uvic.ca
[3] University of Victoria. amiralib@uvic.ca



Our Goal

- Develop a **Deep Neural Network** (DNN)
  - Target: **Radio Frequency** (RF) applications
  - Extreme **Throughput**
  - Ultra-Low **Latency**
  - High **Energy Efficiency**
- Example RF Application
  - 6G Communication Networks
  - Modulation Classification

Modulation classification (MC)

- Modulation Classification
  - A well-known problem in RF domain
  - Requires **high throughput** and **low latency**
- MC Applications
  - Spectrum interference monitoring
  - Radio fault detection
  - Dynamic spectrum access
  - Numerous regulatory and defense applications

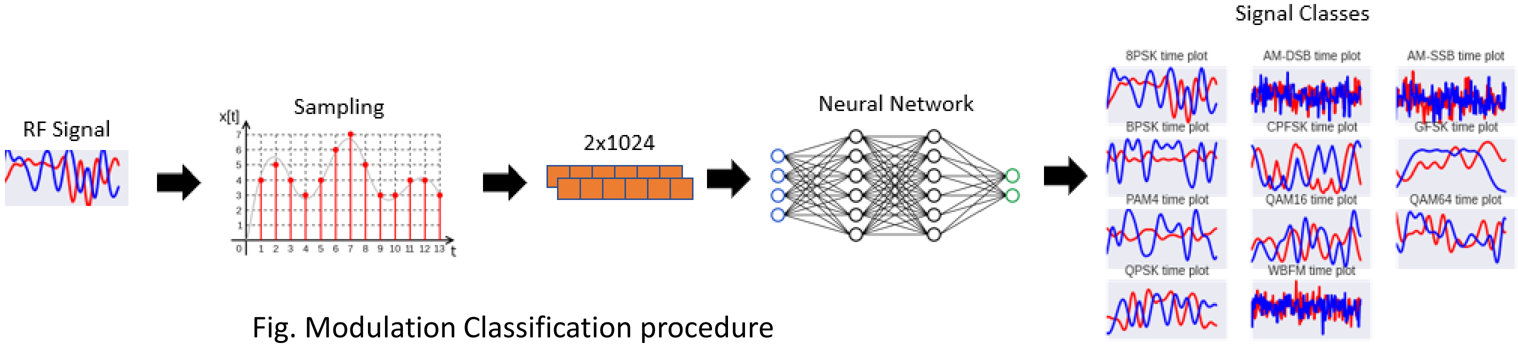


Fig. Modulation Classification procedure

DNN for Modulation Classification

- DNNs are promising tools for analyzing raw data
- Pros:
  - **Higher accuracy** w.r.t conventional methods
  - **Automatic** feature extraction
- Cons:
  - Design **complications**
  - **Computationally** expensive
  - **Resource demanding** (but we need high throughput and low-latency!)
- Our objective: design a DNN specialized for RF domain applications

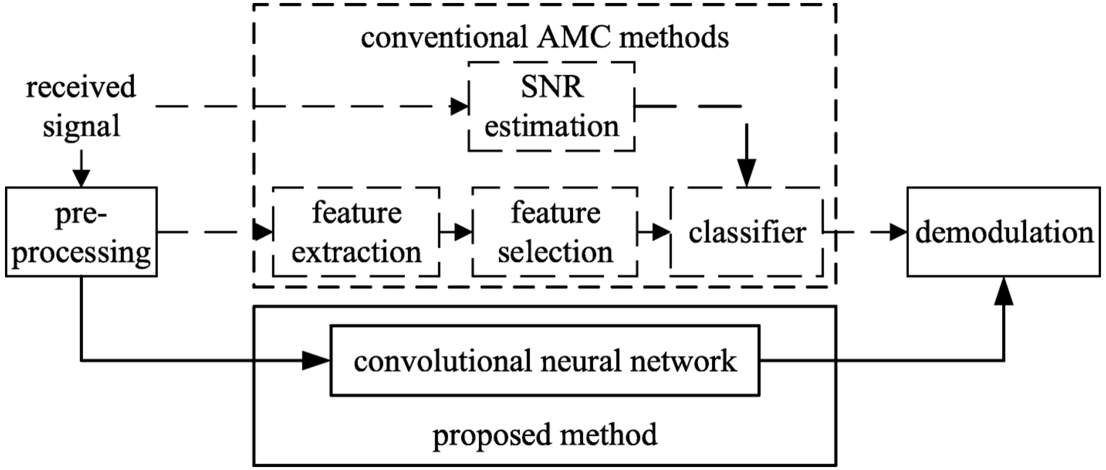


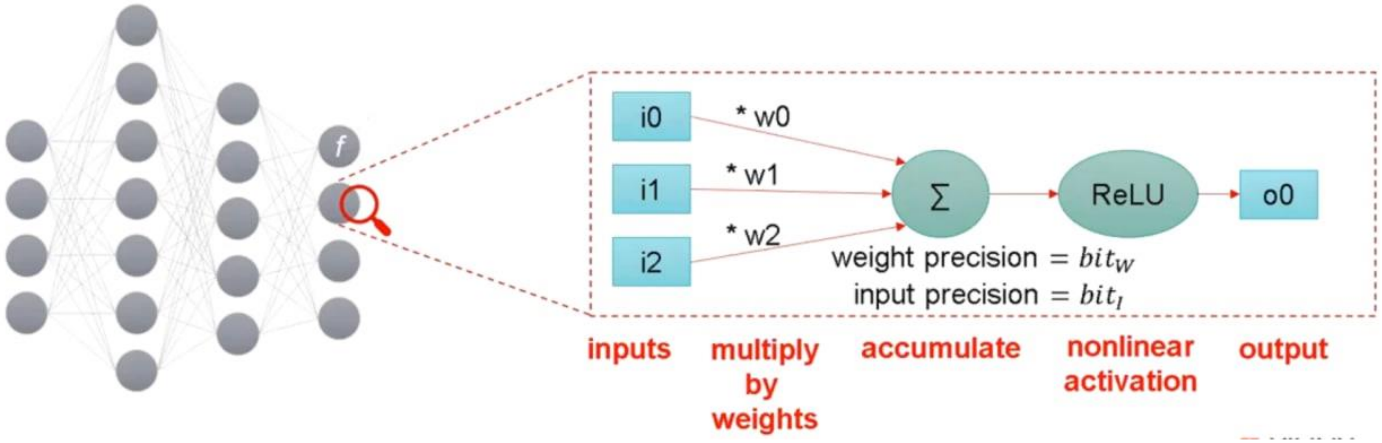
Fig. Comparison of conventional and proposed method

Evaluation metrics

In RF-Domain, we generally require **hardware implementation**.

H/W performance indices:

- FLOPs
- MACs



Computation cost $bit\_ops = \sum^{layers} (\sum^{MACs} bit_W \cdot bit_I)$

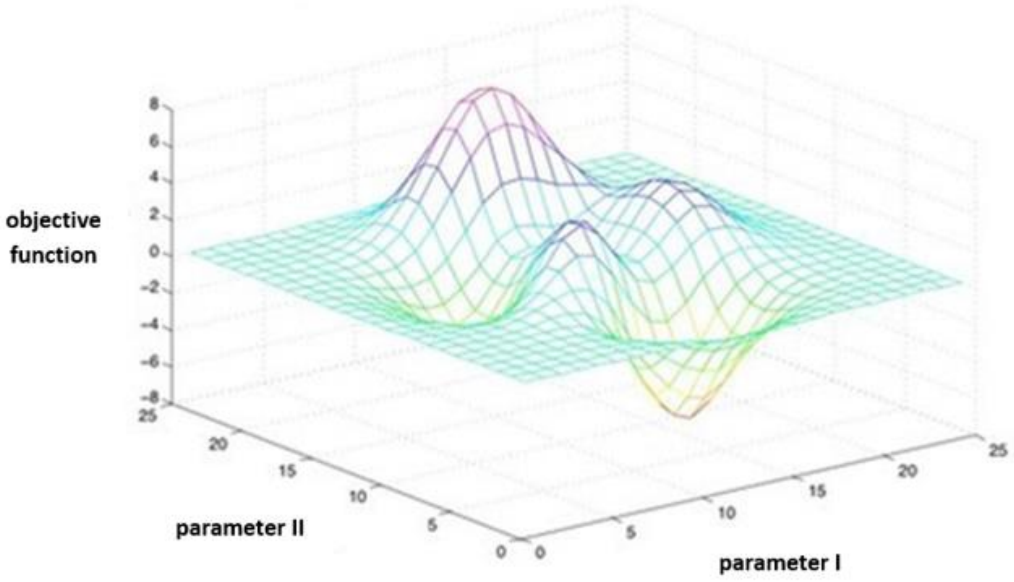
Memory cost $bit\_mem = \sum^{layers} (\sum^{weights} bit_W)$

Normalized score (lower is better)

$$score = 0.5 \cdot \frac{bit\_ops}{baseline\_bit\_ops} + 0.5 \cdot \frac{bit\_mem}{baseline\_bit\_mem}$$

Design Complications

- Domain-specific DNN Design
  - Choice of network architecture
  - Network modifications:
    - **Type/Number** of layers/filters
    - Network **compression** techniques (Quantization, etc.)



- First option: Exhaustive search
  - Grid Search
  - Evolutionary Algorithms (Genetic Algorithm, PSO, …)
  - Neural Architecture Search (Using Reinforcement Learning, …)
  - **Time-Consuming** and **Resource Hungry** for large Datasets (highly unpractical)

Problem statement

Minimize **inference cost** on the challenging and well known **RadioML 2018** dataset

Minimum final accuracy: 56%

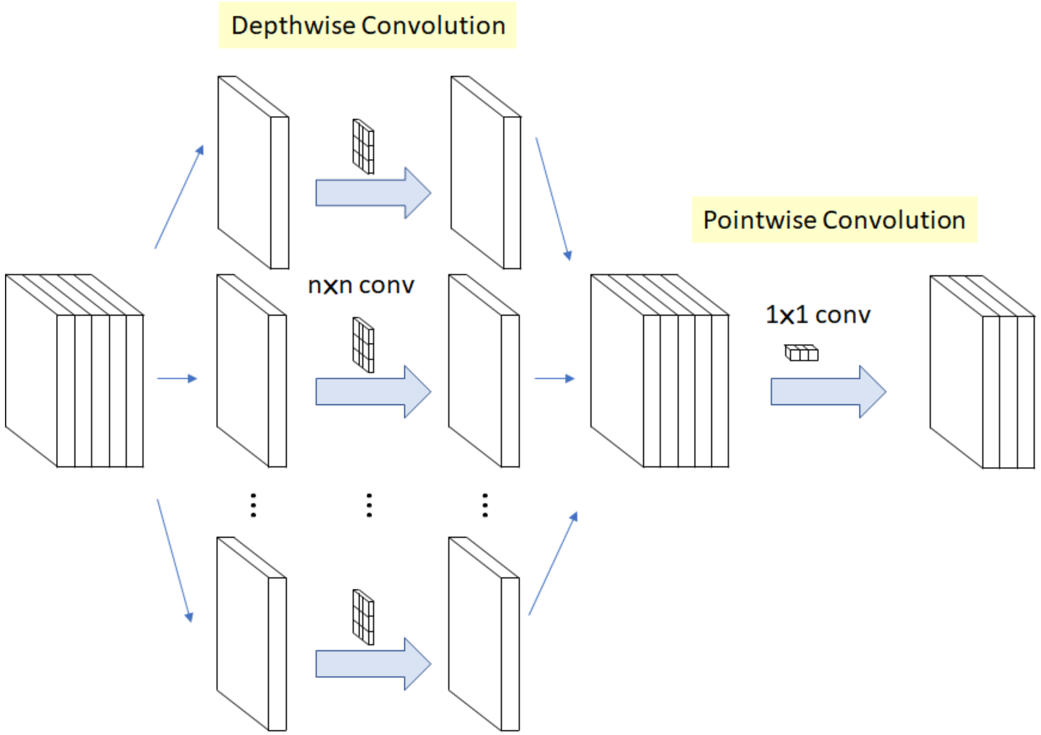
- Solutions of **Aaronica** Team
- Initial round: Pruning **MobileNet**
  - Got 3rd place
- Final round: **AaronNet** Network
  - Designed a specialized DNN after the initial round

Possible Architectures

- VGG, Inception, ResNet, MobileNet, ShuffleNet
  - Each networks has innovations to stand out

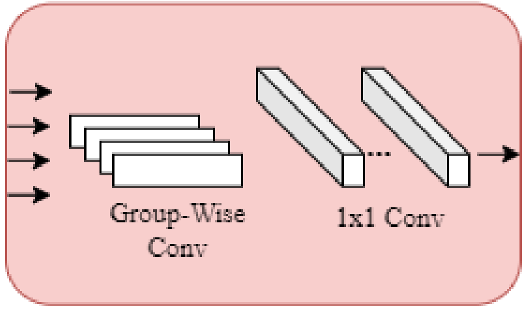
MobileNet

- Pointwise and depth-wise convolution
  - Reducing the number of multiplications
  - Higher **throughput** with lower **computation cost**

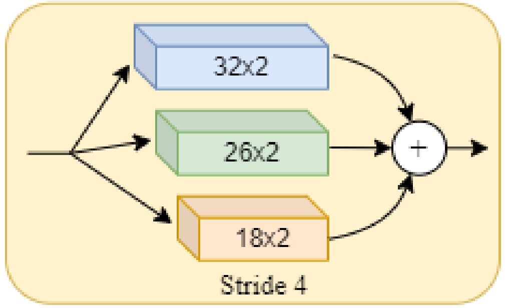


AaronNet

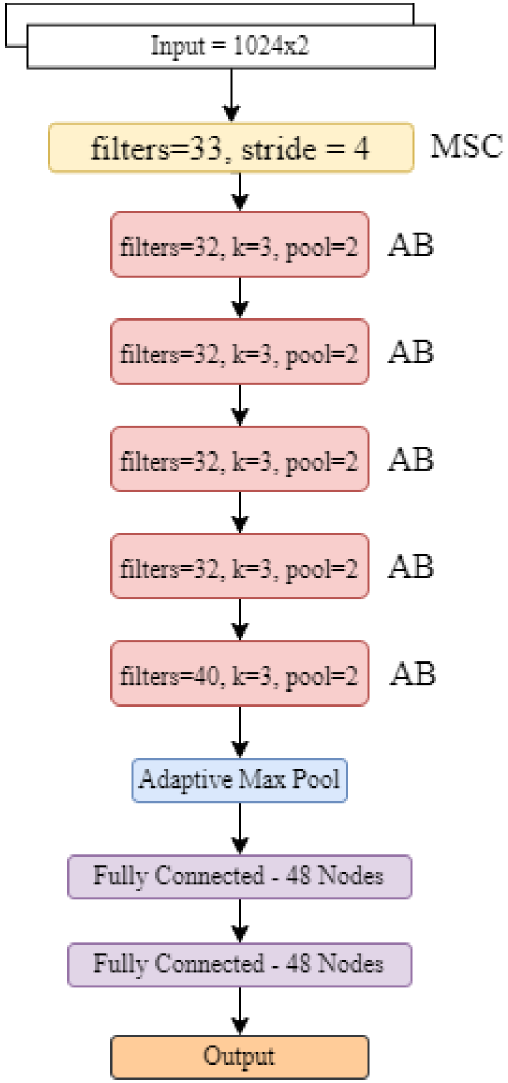
- Specialized for RF-domain
- Group-Wise convolutions
- Introducing **Multi-Scale Convolutional (MSC)** layer
- Adaptive Max Pooling
- **Variations**
  - AaronNet32: 32 filters
  - AaronNet48: 48 filters



AaronBlock (AB)



Multi-Scale Convolutional Layer (MSC)



AaronNet Architecture

Optimizing AaronNet

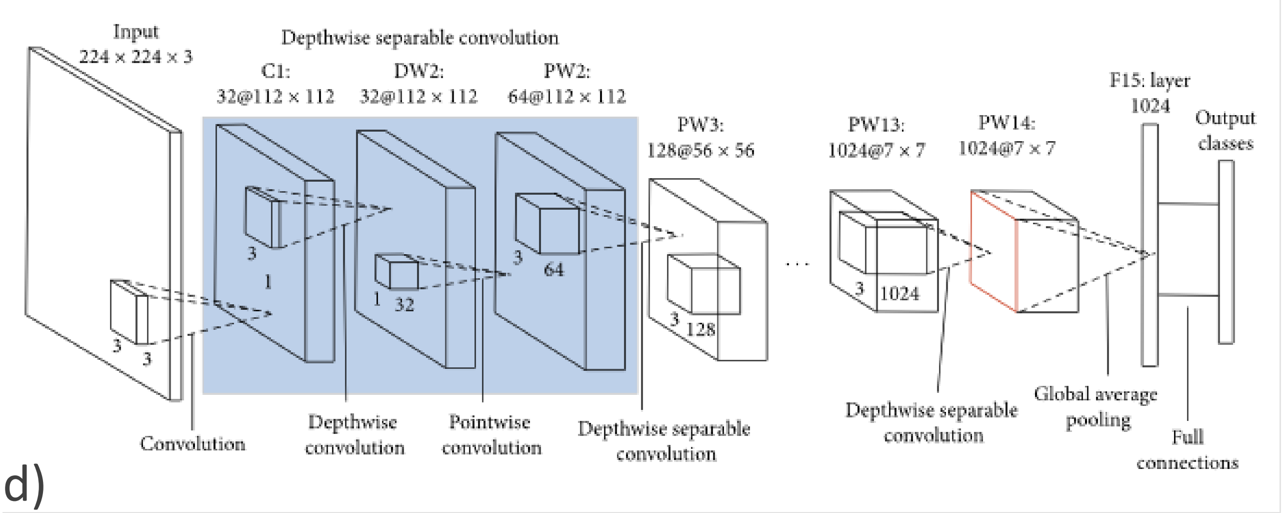
- Significant results using the following methods:
  - Layer-wise quantization and pruning
  - Using Self-attention mechanism SE block
  - Unstructured pruning (More energy efficiency)
  - Structured pruning (More hardware friendly)

- The best results we had so far: **Unstructured Pruning**
  - **AaronNet+**: Slightly pruned version of AaronNet while having the same accuracy
  - **AaronNet++**: Pruned to it's limit, while meeting minimum accuracy of competitio

- Finalizing our paper for submission

Results: Initial Round

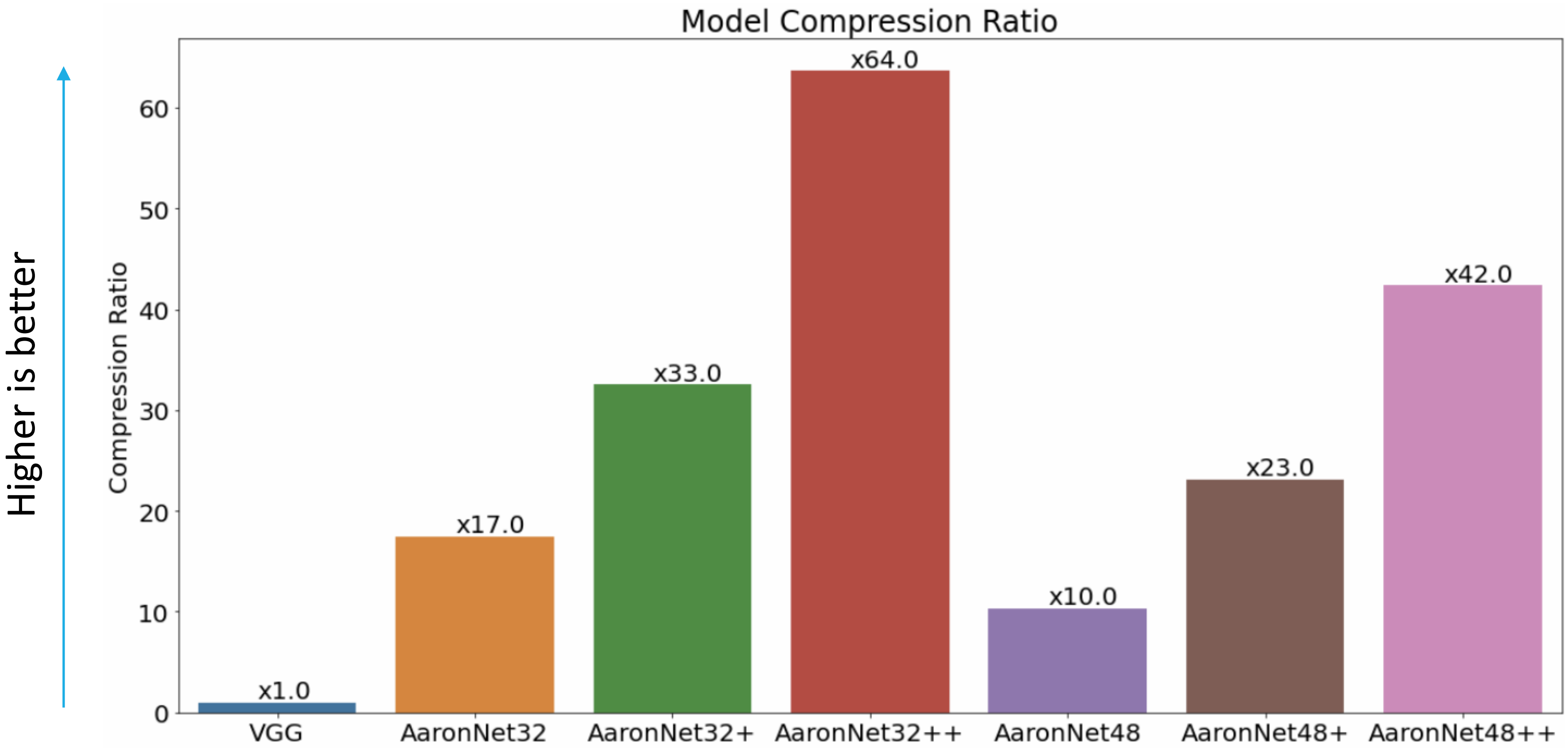
- Baseline for inference cost: VGG network
  - Inference cost = **1**
  - Accuracy = **59.46%**



- Unstructured Pruning of MobileNet (initial round)
  - Using **P100 GPU** on Kaggle
  - Pruned 30% of the weights
  - Accuracy = **56.15%**
  - **21x** inference cost improved (**0.046**)

MobileNet Architecture

Results: Final Round

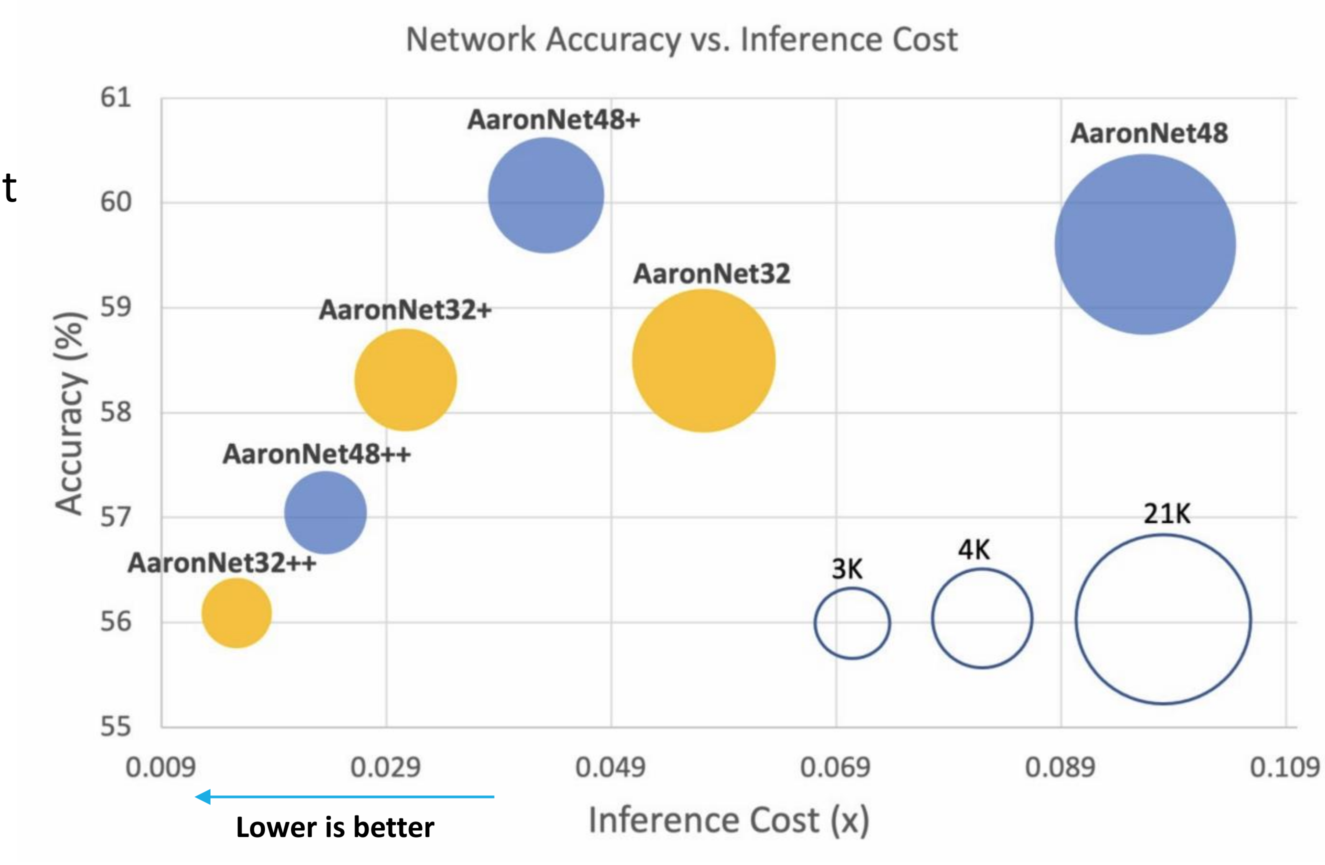


Results: Final Round

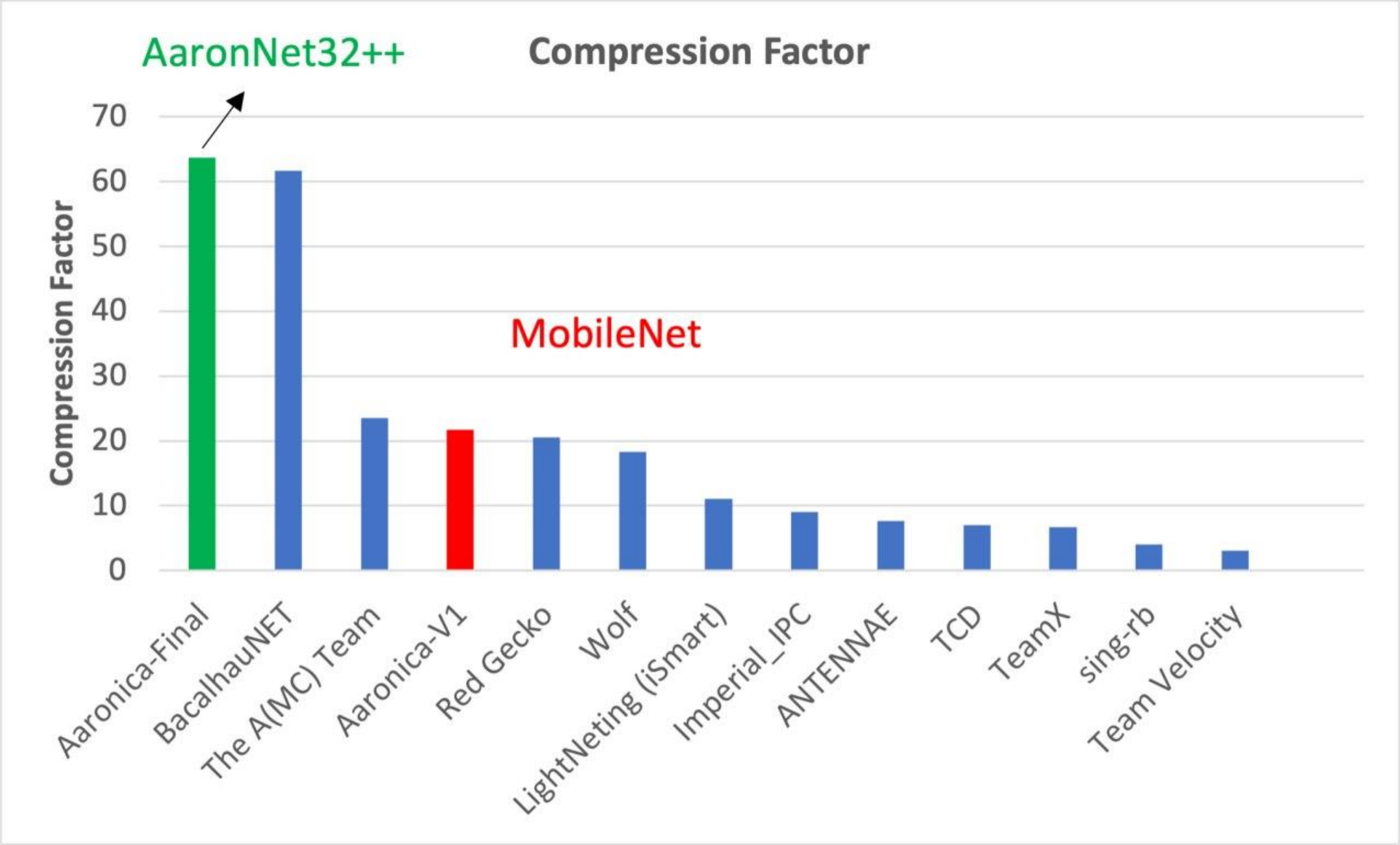
- Using **A6000 RTX GPU**
    - Trained and pruned variations of AaronNet

- Highest Accuracy: AaronNet48+
    - Accuracy = **60.07%**
    - **23x** inference cost (**0.04320**)

- Best Cost: AaronNet32++
    - Acceptable accuracy **56.07%**
    - **64x** inference cost (**0.01539**)



Results: Final Standing



Questions