Optimizing Accuracy/Score Tradeoff: Pruning & Quantization methods

Mehdi BERRADA & Selim ZIGHED

05/03/2025

Overview

Baseline Model

- The baseline model used in this project is the ***ResNet-18***.
- *Model Hyperparameters* :
  - *Epochs* : 200
  - *Batch Size* : 128
  - *Loss Function* : Cross-entropy
  - *Optimizer* : SGD
  - *Learning Rate* : 0.01, *Momentum* : 0.9,
  - *Weight Decay* : 5.10^{-4}, *Scheduler* : CosineAnnealingLR

Baseline Model

- The baseline model used in this project is the ***ResNet-18***.
- ***Model Hyperparameters*** :
  - *Epochs* : 200
  - *Batch Size* : 128
  - *Loss Function* : Cross-entropy
  - *Optimizer* : SGD
  - *Learning Rate* : 0.01, *Momentum* : 0.9,
  - *Weight Decay* : 5.10^{-4}, *Scheduler* : CosineAnnealingLR

Baseline Model's Performance

- ***Accuracy*** : Our ResNet-18 model achieves an accuracy of 93.60%
- ***Total Parameters*** : 11173962 params
- ***FLOPs*** : 556651520 operations
- ***Average Inference Latency*** : 0.002321 sec
- ***Score to be optimized :*** 3.98778

Motivation

$$\text{score} = \underbrace{\frac{\left[1-(p_s+p_u)\right]\frac{q_w}{32}w}{5.6\cdot 10^6}}_{\text{param}} + \underbrace{\frac{(1-p_s)\frac{\max(q_w,q_a)}{32}f}{2.8\cdot 10^8}}_{\text{ops}}$$

Although it provides a solid initial accuracy, the model is relatively large in terms of size , which motivates exploring a compromise between accuracy and efficiency through pruning and quantization techniques (max accuracy for min score).

Optimising Strategy

- **Structured Pruning** → 4 models (*different pruning amounts*).
- **Unstructured Pruning** → Applied to each structured pruned model, generating **16 models**.
- **Quantization** → Applied **FP16** and **INT8** to each pruned model.
- **Evaluation** → Measured performance (**accuracy** and **score**) at each stage.

Structured Pruning

- Structured pruning involves removing entire units, such as convolutional filters, layers, or blocks, to reduce the model size while preserving its overall structure.
- 4 different pruning amounts $\rightarrow$ 4 distinct models to be analyzed

The 4 Pruned Models

Model	**Number of Parameters**
0.1x Pruned Model	$10584756 \approx 10M$
0.3x Pruned Model	$8903011 \approx 9M$
0.5x Pruned Model	$6758914 \approx 7M$
0.7x Pruned Model	$4199988 \approx 4M$

Structured Pruning → Fine-tuning

After pruning, each model undergoes fine-tuning to recover accuracy and adapt to the reduced architecture.

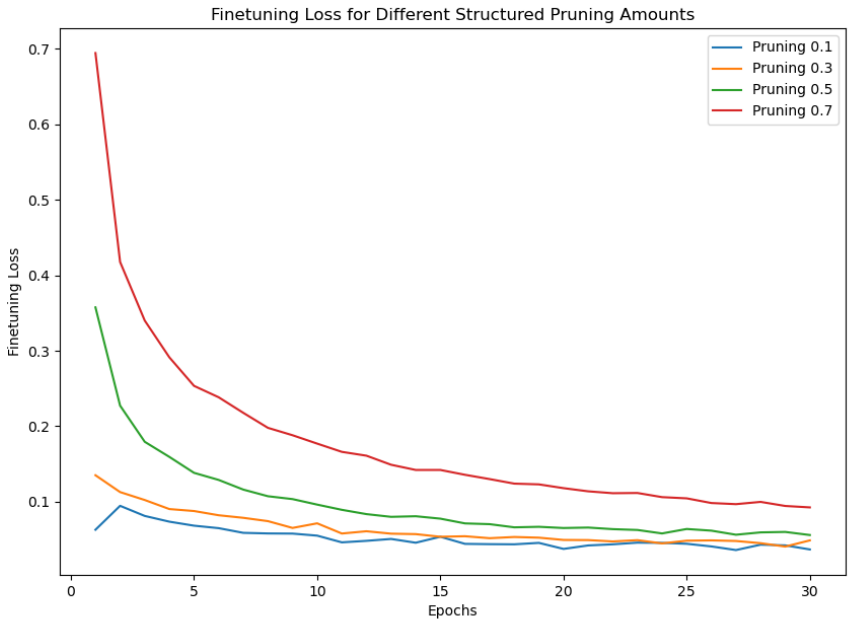


Figure – Fine-tuning loss Evolution over Epochs

Structured Pruning

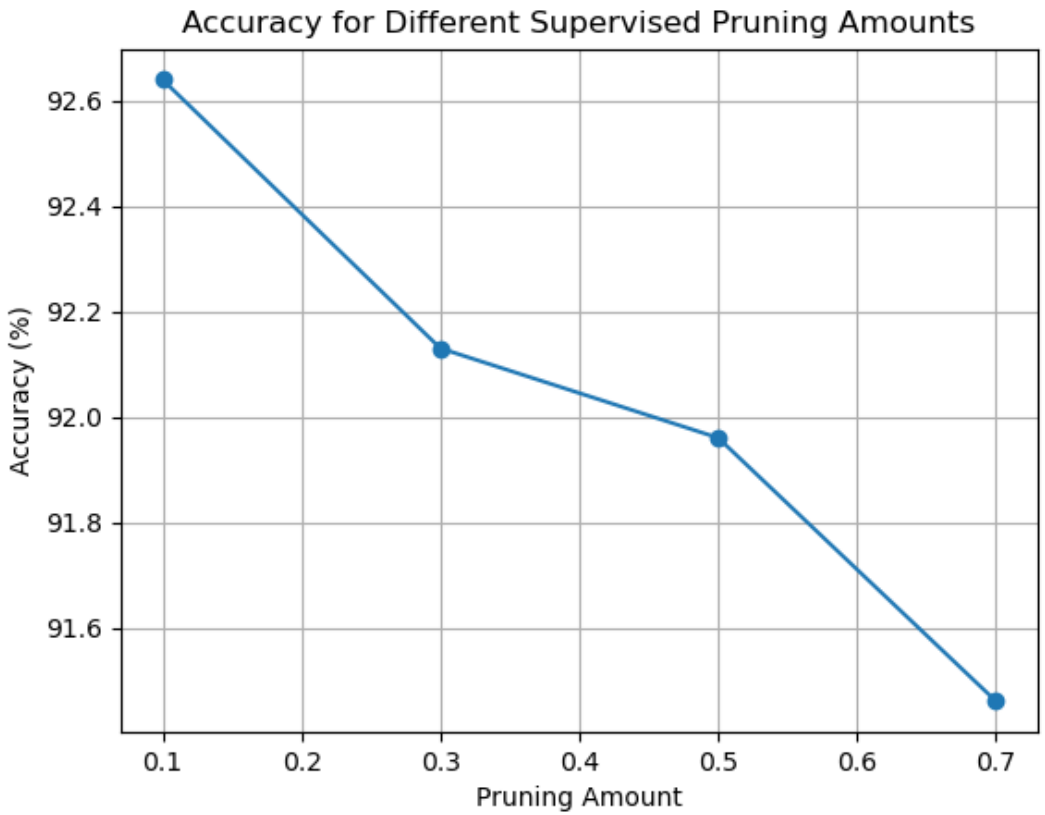


Figure – Accuracy with different pruning amounts

First Trade-off example

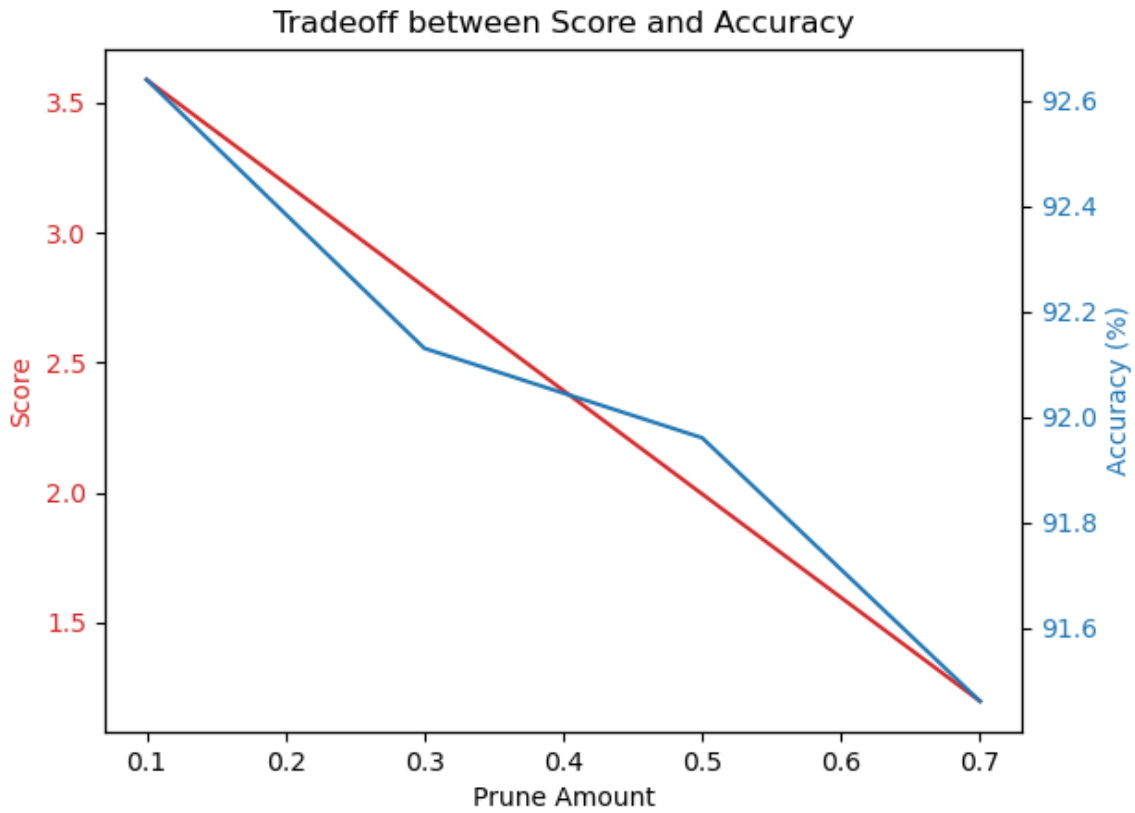


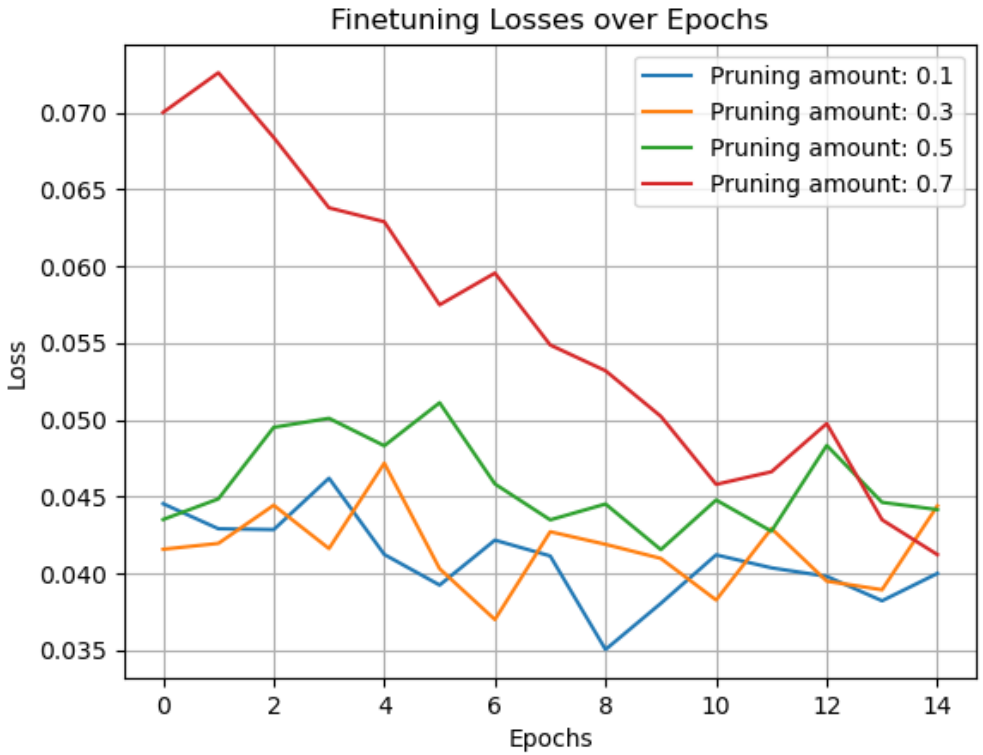
Figure – Score and Accuracy Trade-off after Structured Pruning

Unstructured Pruning

- Unlike structured pruning, which removes entire units, unstructured pruning eliminates individual weights based on their importance, it is called ***Magnitude Based Pruning***.
- 16 models in total obtained by applying unstructured pruning (with the same distinct pruning amounts) on each of the 4 previous models.
- In the following are stated the results obtained after adding the Unstructured Pruning technique with the 0.1x Pruned Model as a Baseline (Best accuracy after structured pruning).

Unstructured Pruned Model

As in the structured pruning, each model has to undergo fine-tuning to recover accuracy.



Accuracy/Score Trade-off

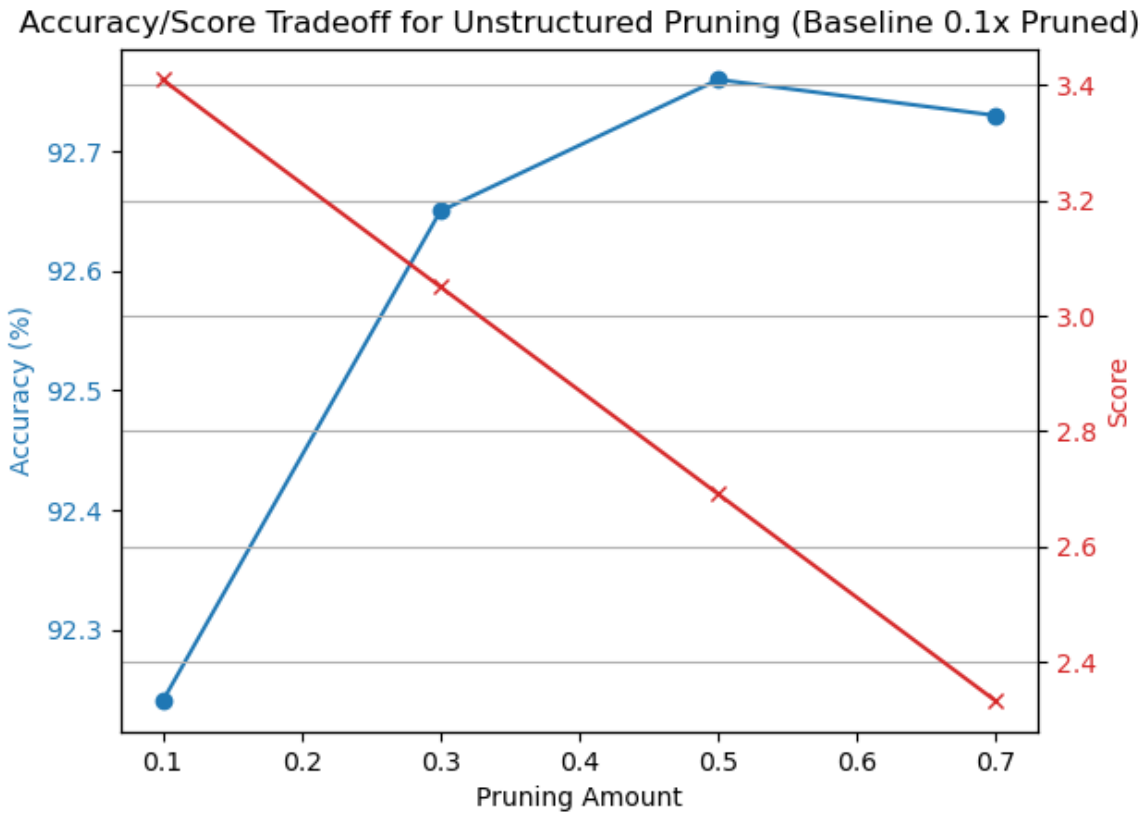
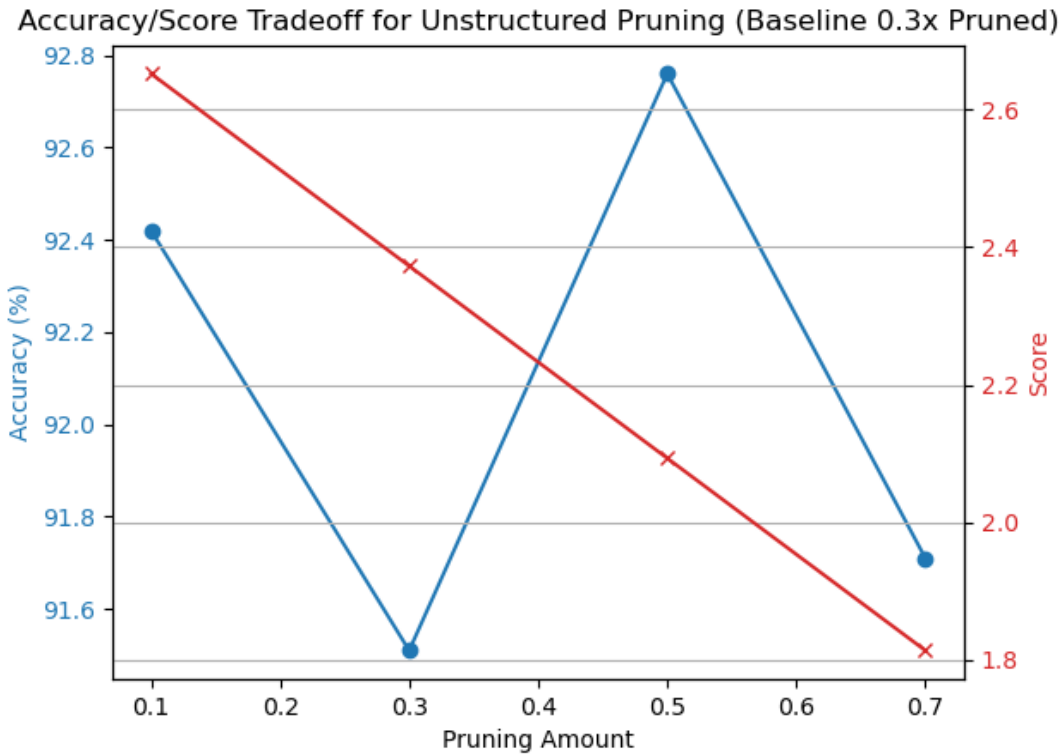


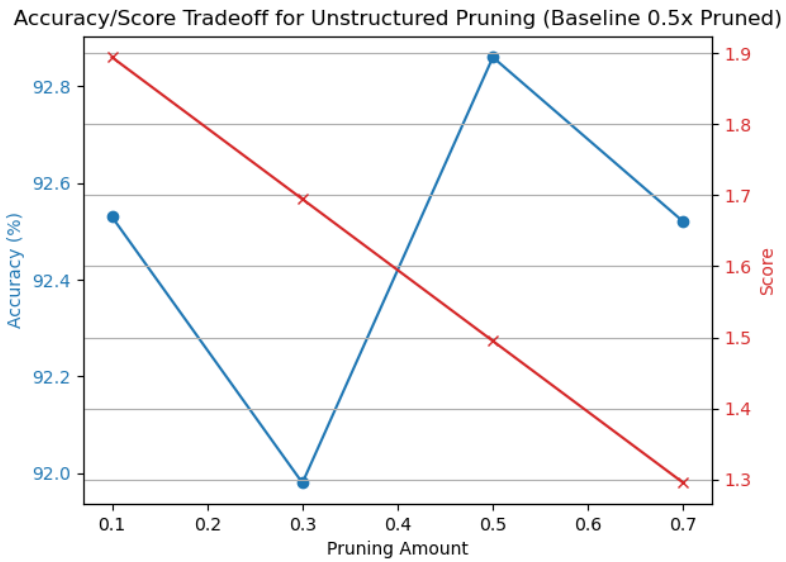
Figure – Accuracy/Score Trade-off after 0.1x Unstructured Pruning

Examples with other Baseline Models

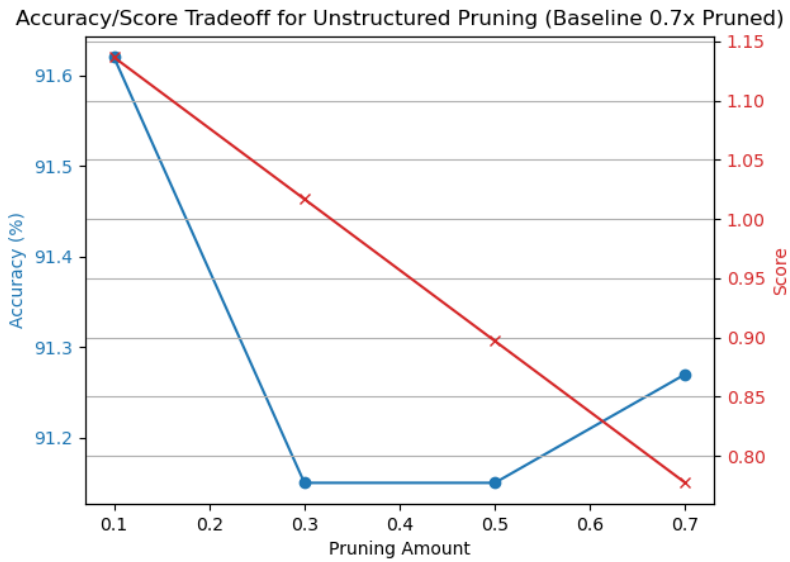
To further see Trade-off plots, we now take as Baseline Models the 3 other Pruned Models (0.3x, 0.5x and 0.7x).



Examples with other Baseline Models



With 0.5x Pruned Model



With 0.7x Pruned Model

Quantization

- Quantization reduces model size and speeds up inference by representing weights and activations with lower precision data types (i.e int8 instead of float32). This improves efficiency, especially on edge devices, with minimal impact on accuracy.

Post Training Quantization

- Reduces model size and improves inference efficiency without requiring model retraining.
- Converts weights and activations to lower precision after the model has been trained.
- Also converts inputs to corresponding versions for the quantization type.
- Simple to apply and widely used for deploying models on resource-constrained devices.
- Post Training Quantization typically ***does not*** require fine-tuning.

Quantization to half precision

- We first tried to quantize into ***fp16*** our first model
- *The Quantized Model's Performance*
  - ***Accuracy*** : 93.62%, which is slightly better than the initial model (not expected but still can be explained).
  - ***Non-Zero Params*** : 11135955 params (some have been set to zero after quantization)
  - ***Score*** : 1.993890

Quantization after Pruning Examples

- For this part, we are going to apply fp16 quantization on the 4 best models after pruning in terms of accuracy.

Model	Struc. Prun.	Unstruc. Prun.	Accuracy	Score
0.5,0.5	0.5	0.25	92.86%	1.495052
0.1,0.5	0.1	0.45	92.76%	2.691094
0.1,0.7	0.1	0.63	92.76%	2.331931
0.3,0.5	0.3	0.35	92.73%	2.093073
0.1,0.3	0.1	0.27	92.65%	3.050257
0.5,0.1	0.5	0.05	92.53%	1.894123

Post Quantization Performances

Table – Models Performances after fp16 Quantization

Model	Accuracy	New Accuracy	Score	New Score
0.5,0.5	92.86%	92.87%	1.495052	0.747526
0.1,0.5	92.76%	92.78%	2.691094	1.345547
0.1,0.7	92.76%	92.72%	2.331931	1.165966
0.3,0.5	92.73%	92.76%	2.093073	1.046537

Trade off Plot

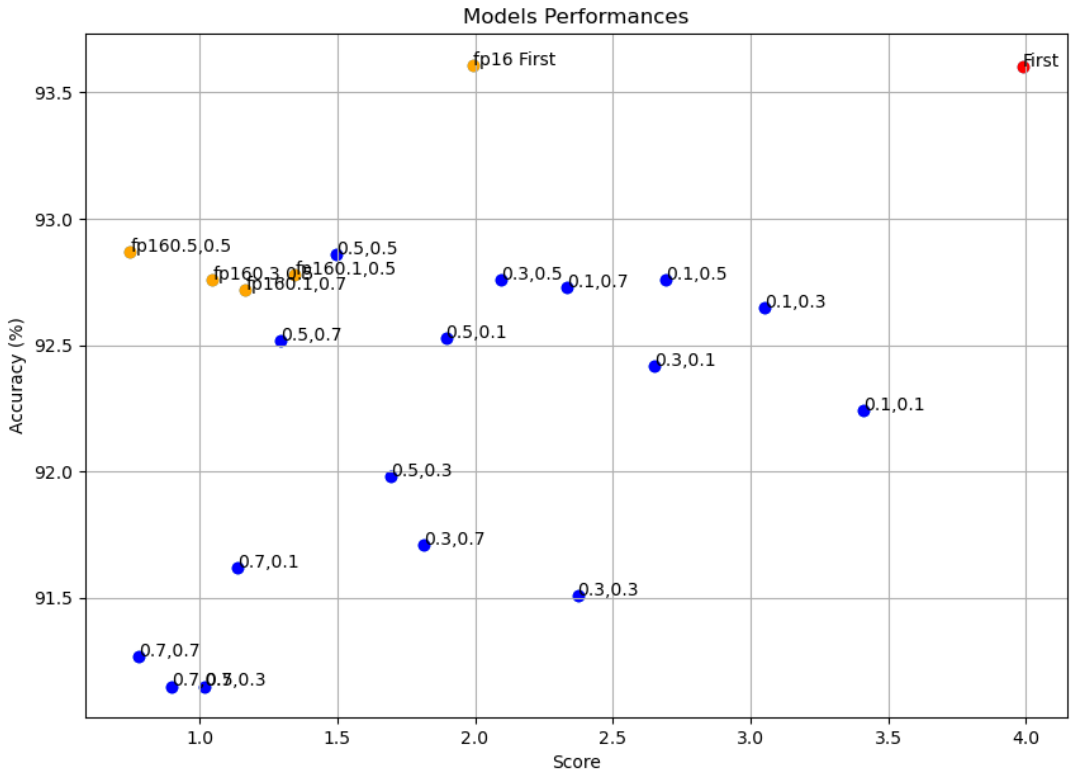


Figure – Trade Off Plot

Pruning Filters for Efficient ConvNets

The method proposed in the paper **"Pruning Filters for Efficient ConvNets"** is based on the **structured removal** of convolutional filters to improve the efficiency of neural networks.

- Filters are evaluated based on their **importance** using the **L1 norm** (sum of the absolute values of the weights).

$$||F||_1 = \sum_{i,j,k} |w_{i,j,k}|$$

Pruning Filters for Efficient ConvNets

- The filters with the lowest importance are **removed**, which directly reduces the model's complexity.
- This approach produces **lighter** and **faster** models while maintaining accuracy close to the original model.
- A **fine-tuning** phase follows the pruning process to recover some of the lost accuracy, as we did with all the pruning steps earlier.

Paper Results Recap

Pruning Amount	Accuracy	Before FT	Score
0.3	92.93%	54.3%	2.506003
0.5	92.48%	10.17%	1.596360