Enhanced Deep Neural Network Model Pruning with Causal Inference

Anishka Ratnawat
CSA, IISc
SR no. 22777
anishkar@iisc.ac.in

Boul Chandra Garai
CSA, IISc
SR no. 23318
chandraboul@iisc.ac.in

Snigdha Shekhar
CSA, IISc
SR no. 22453
snigdhas@iisc.ac.in

Vaisakh P S
CSA, IISc
SR no. 23843
vaisakhp@iisc.ac.in

***Abstract*—Deep Neural Networks (DNNs) achieve high performance but can be computationally expensive to deploy. Pruning offers a way to reduce DNN size, but traditional methods based on correlations often struggle to identify which parameters can be removed without sacrificing performance. This can lead to reduced accuracy or unpredictable effects on the model's behaviour. This work investigates causal inference to enhance DNN pruning. We propose a causal graph that captures the relationships between pruning extent, model accuracy, amount of computations, and system resource usage (e.g., memory consumption). Using doWhy, we estimate the causal impact of pruning on accuracy and employ refutation tests to assess the robustness of these estimates. We further explore the mechanisms by which pruning affects accuracy. Our project aims to answer if and how pruning helps to run DNN/ML algorithms/models on edge devices under resource constraints. In this project, we evaluated pruning methods on the various architectures and analyzed the findings.**

Keywords: Deep neural networks (DNNs), Model Pruning, Causal Inference, doWhy, Refuter type.

I. Motivation and related work

Pruning in Deep Neural Networks (DNNs) is essential for reducing model complexity and size, enhancing computational efficiency, and preventing overfitting. By removing redundant parameters, pruning helps compress large models, leading to faster inference times, lower energy consumption, and improved generalization performance. Additionally, pruning induces sparsity in the network, a form of regularization, and promotes model interpretability. These benefits make pruning a crucial technique for deploying efficient and interpretable DNNs in real-world applications, particularly on hardware-constrained devices.

In the paper by Debbi et al. (2021) [1], a novel approach called CexCNN identifies the most crucial features in input images by assessing the responsibility, blame, and causality abstraction of each causal filter within the final convolution layer.

Pawlowski et al. (2020) [2] introduces a general framework for constructing Structural Causal Models (SCMs) by utilizing normalizing flows and variational inference to facilitate tractable inference of exogenous noise variables.

Research into Causal insights to methods such as Reinforcement Learning [3] [4] [5] has also revealed improvements. However, identifying the fundamental parameters (weights) or neurons remains challenging. Reliance on correlation-based heuristics in pruning can lead to sensitivity to noise and dataset bias [6]. These methods risk removing elements that correlate with performance but don't truly cause good performance, which may lead to unpredictable pruning outcomes.

Existing methods risk removing elements that correlate with performance but don't indeed cause good performance, leading to suboptimal and unpredictable pruning outcomes. Interpretability methods might not reveal the causal mechanisms through which pruning impacts accuracy. Understanding the causal reasons for a pruned model's performance is essential for developing effective pruning strategies, debugging potential issues, and ensuring trustworthiness. Hence, this project focused on using causal insights for model performance under various pruning configurations and system aspects. Our causal graph highlights the challenge of potential confounders. Careful consideration is required to understand the interactions among model complexity, edge device specifications, and pruning methods, especially when applying a specific degree of pruning to optimize performance in terms of device capabilities, inference time reduction, and model accuracy.

II. Methodology

We implemented a detailed algorithm for conducting causal inference on pruned models using the *doWhy* library. The doWhy package provides a principled framework for causal inference in Python [7]. doWhy library supports defining causal graphs, identifying causal effects, and explicitly modelling assumptions.

A. Methodology for a Data Gathering Workflow

Input: Various pre-trained neural network models and ImageNet and MNIST datasets.

Output: The output is a CSV file containing the experiments' results.

Pre-Trained Model Check: Load weights and biases if pre-trained models are available. If not, training is initiated on the dataset, and the resulting model parameters are saved to a file.

Pruning Experiment Loop: The process iterates over a range of pruning ratios. Pruning is applied to the network at the specified ratio using a chosen method. Network performance is evaluated and accuracy is calculated. Model performance statistics, such as memory consumption and FLOPs (Floating Point Operations), are collected. Average system statistics are computed throughout the experiment. All gathered information

is recorded in a data frame, indexed by the pruning ratio. The data frame is saved as a CSV file if it is the last iteration.

Algorithm 1 Pruning Experiment on Neural Network

Input: Model architecture, dataset
Output: CSV file with experiment results

```
Choose and configure hardware accelerator for execution
Define the model architecture
Instantiate test dataset loaders
Define the dataset transformation pipeline
Load pre-trained weight and bias, if available
Otherwise, Instantiate the training dataset, perform training, and save model parameters to file
for each pruning ratio in the experiment do
    Apply pruning with the specified pruning ratio to the network
    Train the new network
    Evaluate the network's performance and calculate accuracy
    Collect model performance statistics such as memory consumption and FLOPs
    Calculate average system statistics over the experiment duration
    Record all information against the pruning ratio to a DataFrame
    if last iteration then
        Save the DataFrame to a CSV file
```

Applying doWhy to pruning offers the potential for a more rigorous and interpretable analysis of the pruning process. Using doWhy, we can carefully construct a causal graph that hypothesizes the relationships between pruning, model parameters, system resource usage, and accuracy.

In phase 2 of the project, the main focus was to establish and prove the working of the Causal Model to a DNN model pruning problem. A workflow, as shown in Fig-1, was setup on a PC which incorporated a profiling block for the pruned model, as listed in Algorithm-1, and the Causal Model generation, as shown in Algorithm-2, from the data generated by the Algorithm-1. Arbitrary threshold levels were selected to understand and prove causal modelling for the values of X, Y, and Z mentioned Algorithm-2.

In the final phase of the project, we experimented by executing the vgg16 and GoogLeNet models on different devices. We collected the data, to carry out the causal analysis to characterize our model assumptions.

B. Methodology for a Validation Workflow

As indicated in Fig 1, a causal model generation involves four significant steps. The first is modelling the causal mechanism, wherein a causal graph, specifically a Directed Acyclic Graph(DAG), is created based on domain knowledge input. The second step is to identify the target estimand, wherein a correct estimand is formulated based on the causal model. The third step of estimating the causal effect includes using suitable methods to assess the impact. The fourth and final step before obtaining a Causal Estimate involves refuting estimates. In this step, the generated model is checked for robustness regarding its estimations to assumption violations. This step provides reasonable confidence in the causal model that is produced.

This produced Causal Model helps perform causal analysis tasks such as estimating effects in case some environ-

Algorithm 2 Causal Inference using doWhy

Input: CSV file containing experiment data
Output: Causal effect estimate of pruning on accuracy loss and its refutation tests

```
Import the required libraries (pandas, doWhy, etc.)
Read the Algorithm-1 CSV into a DataFrame
begin Preprocess the data:
    Add a column 'High_Accuracy_Loss' indicating whether accuracy is below a threshold X  Add a column 'High_Pruning' indicating whether the pruning ratio is above a threshold Y  Add a column 'High_Memory_Consumption' indicating whether memory usage percentage is above a threshold Z
Prepare a causal graph hypothesizing relationships
Instantiate a causal model against added columns
Identify the causal effect
Estimate the causal effect
begin Perform refutation tests to check robustness of the causal estimate:
    Check estimate variation for random changes  Check estimate consistency against a data subset  Replace the treatment alternative and observe
Output results
```

TABLE I
DATA POINTS CAPTURED IN EXPERIMENT

Data-point Type	Data points
CPU: Perf-stat	Instruction, cycles, cache, branch statistics etc.
CPU: System statistics	CPU Load averages Pressure Stall Information, GPU Load, Swap and Main Memory statistics, Disk IO Statistics etc.
HW Configuration	DDR Product details CPU Operating Frequencies, Board information etc.
GPU Instruction statistics	Instruction per warp, Issued/executed Instruction, Global/Local Hit rates etc.
GPU Memory statistics	System Memory utilization, Stall statistics, Load/Store statistics, DRAM write/read throughput and utilization etc.
Floating point statistics	Instruction count, FLOPS count , etc.

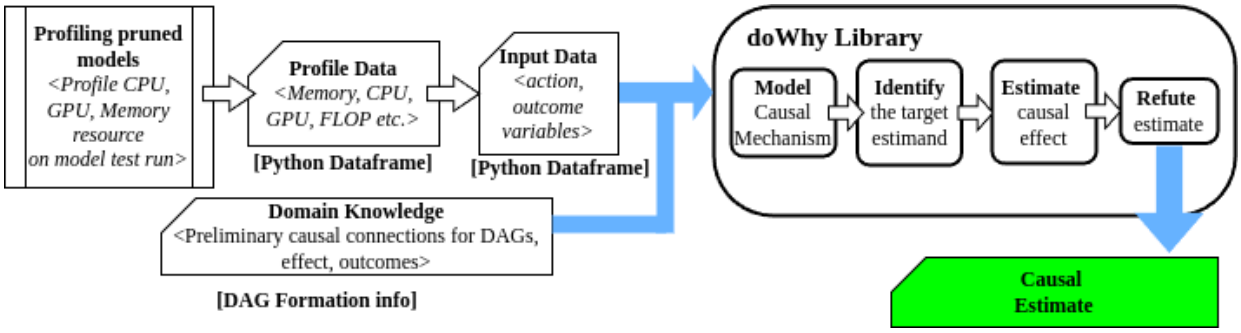

Fig. 1. Overview of model pruning with causal inference derivation using doWhy [7] library

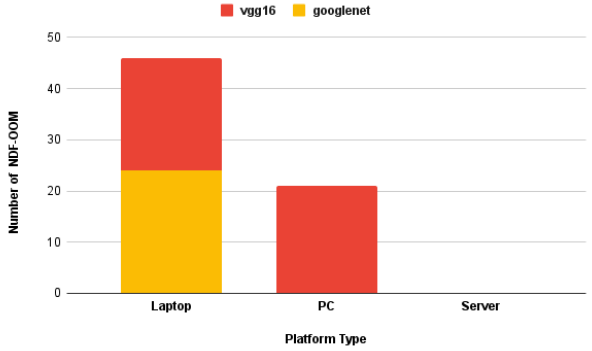

Fig. 2. OOM incidents for different platform types, with specification in Table-II

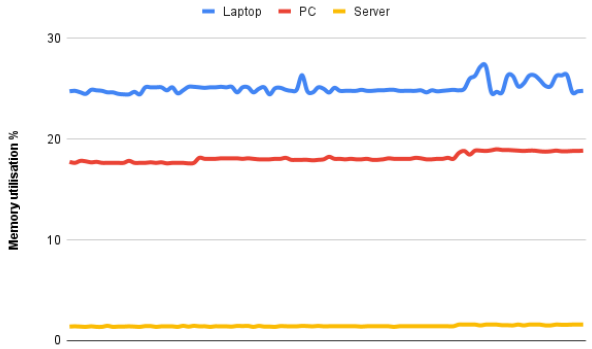

Fig. 3. Memory utilization % vs platform type

ment changes occur, root-causing and explaining observed behaviour, etc. This root cause analysis and explanation can assist anomaly attribution, quantifying causal contributions and feature relevance.

Table-I summarizes the various data points captured during model profiling across different hardware platforms.

III. Results

Pruning and model performance were done on three separate hardware platforms as shown in Table-II. A fully automated data-gathering script was developed to evaluate model performance before and after pruning and whether it was executed successfully or not. An unsuccessful execution can happen due to Out-Of-Memory(OOM) or other factors in the evaluation script logs.

TABLE II
Details of hardware used for experiment

Type	CPU	GPU	Memory
Server	Intel Xeon(R) 6140 72-Core @2.30GHz	NVIDIA Tesla P40 23GB GDDR	376 GB
PC	AMD Ryzen 7 3700X 8-Core Processor @ 4.40GHz	NVIDIA GeForce RTX 3060 12GB GDDR	46 GB
Laptop	Intel Core(TM) i7-9750H 8-Core @ 2.60GHz	NVIDIA GeForce GTX 1650 4GB GDDR	16GB

Fig 3 depicts that memory consumption greatly depends on the class of device we have chosen to run the models on. This trend suggests that more advanced hardware configurations are better optimized for memory management as they enable more efficient utilization of available resources. We have also, therefore, observed a positive correlation between the out-of-memory issues and the device's memory capacity as depicted in Fig 2. Consequently, we also observed a similar correlation between batch size and the OOM issue. As batch size increases, so does the memory footprint of the model during inference. This correlation highlights the importance of considering batch size as a confounding variable in our causal model.

The causal model obtained from the experiment setup, as shown in Fig 1, and Algorithm-1 and Algorithm-2, observed a p-value of 1.0 (Random common cause), 0.97(Subset of data), and 0.64 (Placebo Treatment) refuters as summarized in Table-III.

IV. Discussion on Results

Experiments on setting up causal estimation enabled workflow as in Fig-1, revealed deficiencies in learning frameworks in Python such as PyTorch [8], wherein accuracy drop after pruning were observed, yet, corresponding reduction in FLOPs and memory footprint of the pruned model were not reported [9]. Hence, we profiled the metrics for evaluation using *perf* and *nvprof* facilities to incorporate relevant systems statistics.

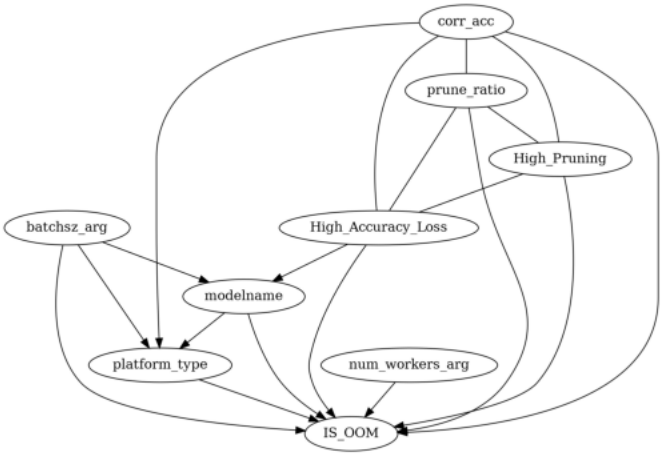

Fig. 4. Causal Model generated using doWhy [7] library and data obtained from experiment.

TABLE III
ESTIMATOR REFUTER TEST RESULTS

Refuter Type	p-value
Add a random common cause	1.0
Use a subset of data	0.97
Use a Placebo Treatment	0.64

The favourable refuter test results shown in Table-III give a high confidence in the causal model, as a p-value < 0.05 in refuter tests indicates a problem with the estimator [10]. Though further extensive testing and validation are required, these numbers provide confidence in the direction towards a well-performing causal model. Fig-4 illustrates the causal model derived by doWhy from the data capture across the experiment setup.

Since the captured data consists of instances on any test platform wherein a specific batch size and number of data runners are configured against various pruning approaches and ration configurations, it can generate relationships between multiple data points. For instance, IS_OOM is a boolean field generated from test results wherein the execution results in an Out-of-Memory(OOM) condition. It can be observed that the Causal Model was indeed about identifying the relationship between the configuration entries within the captured data frame against an outcome variable.

It can be seen that Causal Analysis packages like doWhy can assist novice or expert developers in complex fields like Systems for Machine Learning and Artificial Intelligence in deciphering complex relationships between various data points and outcomes with the help of generating a Causal Graph. The number of lines of code once the experiment runs provides a data-frame is made available is less than 25 lines of code. Such simplicity can help any developer to understand other causal relationships in different aspects of the underlying system and fine-tune its operation to an optimal point as desired.

V. CONCLUSION

As a part of this project, we have utilised the techniques of causal inference to evaluate the performance of various models by incorporating system statistics to deepen our understanding of the impact of different pruning methods and the execution environment of these models. The results from the experiment help improve the quality of insights and enhance the model to provide counterfactual inferences that allow us to estimate, in a new environment, how a specific configuration would perform.

While the data-gathering of benchmark and performance statistics was performed on Server, Personal Computer (PC) and laptop configurations with Intel and AMD chipsets, there is a scope for extending this work to other ARM-based architectures, mobile devices etc.

Such an analysis can also be extended to more complex models like Bert, GPT etc that would benefit even more from this exercise, thereby targeting the development and deployment of optimal custom pruning strategies for different platforms. Yang [11] provides an extensive repository of pre-trained and pruned models which can serve this purpose.

The complete source code of automated pruning and profiling, along with Causal Model forming is available on GitHub [12].

REFERENCES

[1] H. Debbi, "Causal explanation of convolutional neural networks," *Oliver, N., Pérez-Cruz, F., Kramer, S., Read, J., Lozano, J.A. (eds) Machine Learning and Knowledge Discovery in Databases. Research Track. ECML PKDD 2021. Lecture Notes in Computer Science(), vol 12976. Springer, Cham.*, 2021.

[2] N. Pawlowski, D. C. Castro, and B. Glocker, "Deep structural causal models for tractable counterfactual inference," 2020.

[3] Z. Deng, J. Jiang, G. Long, and C. Zhang, "Causal reinforcement learning: A survey," 2023.

[4] T. He, J. Gajcin, and I. Dusparic, "Causal counterfactuals for improving the robustness of reinforcement learning," 2023.

[5] M. Seitzer, B. Schölkopf, and G. Martius, "Causal influence detection for improving efficiency in reinforcement learning," 2021.

[6] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2019.

[7] A. Sharma and E. Kiciman, "Dowhy: An end-to-end library for causal inference," 2020.

[8] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[9] (2020) Add prune tutorial discussion. [Online]. Available: https://github.com/pytorch/tutorials/pull/605#issuecomment-585994076

[10] (2022) Basic example for calculating the causal effect. [Online]. Available: https://www.pywhy.org/dowhy/v0.9.1/example_notebooks/dowhy_simple_example.html

[11] Y. He and L. Xiao, "Structured pruning for deep convolutional neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–20, 2023.

[12] A. R. B. C. G. Snigdha Shekhar, Vaisakh P S, "E0-294_project-causalmodellingpruning," https://github.com/vaisakh-sudheesh/E0-294$_{Project - CausalModellingPruning}$, 2024.