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. In this initial phase of the project, we have evaluated pruning methods on the LeNet-5 architecture and tabulated the preliminary 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. Using counterfactual reasoning, CexCNN effectively identifies salient regions within input images.

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. Validated on synthetic datasets generated from MNIST to create a dataset with a known causal structure and access to the actual process of generating counterfactuals.

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.

II. Methodology

We implemented a detailed algorithm for conducting causal inference on the pruned model using LeNet-5 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.

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.

As indicated in Fig-1, a causal model generation involves four significant steps. The first is the Model 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

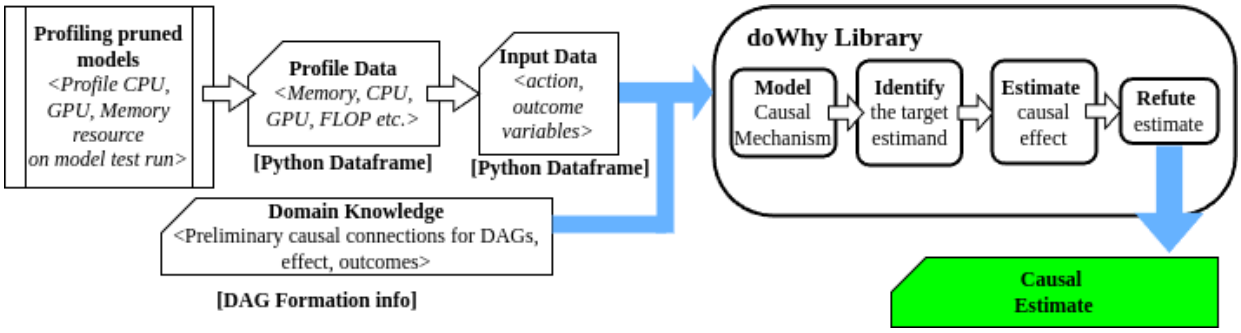

Fig. 1. Overview of model pruning with causal inference derivation using doWhy [7] library

Algorithm 1: Pruning Experiment on LeNet-5 Neural Network

Input: LeNet5 architecture, MNIST dataset
Output: CSV file with experiment results

Choose and configure hardware accelerator for execution
Define the LeNet5 Neural 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

Algorithm 2: Causal Inference using the 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

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 environment 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.

III. Preliminary Results

Evaluation of CexCNN [1] showed consistent results as reported in the publication. Specifically, we pruned (13/20) filters from the first convolutional layer and (28/50) filters for the second convolutional layer, resulting in a total reduction of 55.92% in parameters. In [2], we ran the three models and observed that enabling conditional image generation enhanced the incorporation of a causal dependency between thickness and intensity. Specifically, the independent model lacked any relationship between the image and thickness/intensity, thus preserving the image under the given intervention. The conditional model did not capture any dependency of intensity on

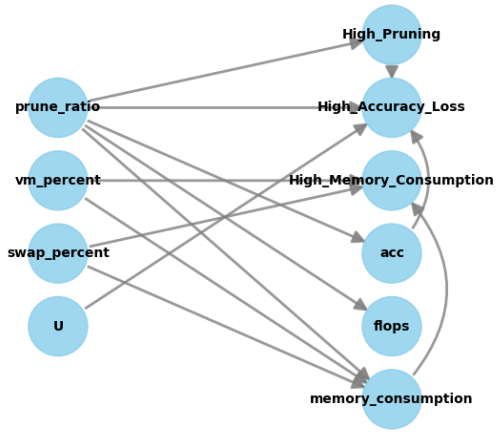

Fig. 2. Causal Model generated using doWhy [7] library and data obtained from experiment

TABLE I
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

thickness, resulting in counterfactuals with varying thickness but constant intensity.

The causal model obtained from the experiment setup, as shown in Figure-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-I.

IV. Discussion on Results

The numbers stated by state-of-the-art [1] [2] were confirmed, revealing complexity and challenges that required deeper insights that Causal analysis could provide in terms of runtime environment wherein models are to be deployed.

Experiments on setting up causal estimation enabled workflow as in Figure-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, for evaluation, we will use models that are trained and pruned for various ratios and FLOP goals to a more considerable extent to avoid misconfiguration. Yang [10] provides an extensive repository of pre-trained and pruned models which can serve this purpose.

The favourable refuter test results shown in Table-I give a high confidence in the causal model, as a p-value < 0.05 in refuter tests indicates a problem with the estimator [11]. Though further extensive testing and validation are required, these numbers provide confidence in the direction towards a well-performing causal model. Fig-2 illustrates the causal model used in refuting tests.

V. Future Plans

Causal inference will be extended to more complex models in our research, potentially incorporating additional performance-evaluating parameters to deepen our understanding of the impact of various pruning methods and the execution environment of these models. A few of these parameters will be low-level CPU statistics such as L1/L2 caches, CPU frequencies, swap performance statistics, Primary Memory type/frequency, GPU memory/load statistics etc. This will help in improving 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.

Exploratory work will be performed on multiple aspects of the causal model by searching the space of refutation tests. Different frameworks will also be explored for the pruning technique implementation of LeNet-5 along with other pre-trained models, such as AlexNet [12], Inception [13] to observe a reduction in FLOPS and memory-related counters.

Data-gathering of benchmark and performance statistics will be done on Personal Computer (PC) and laptop configurations with Intel and AMD chipsets. Limited experiments will be performed on edge devices depending on availability.

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] 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.

[11] (2022) Basic example for calculating the causal effect. [Online]. Available: https://www.pywhy.org/dowhy/v0.9.1/example_notebooks/dowhy_simple_example.html

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015.