Enhancing EfficientNet-B3 for Military Aircraft Classification: A Study on Structured Pruning and Gaussian Noise Robustness

1st Lars Klunder
University of Twente, EEMCS
l.a.klunder@student.utwente.nl

2nd Stan Oostrik
University of Twente, EEMCS
s.h.oostrik@student.utwente.nl

3rd Vitalii Fishchuk
University of Twente, EEMCS
v.fishchuk@student.utwente.nl

4th Askar Sadykov
University of Twente, EEMCS
a.sadykov@student.utwente.nl

5th Misgana Chimsa
University of Twente, TNW
m.s.chimsa@student.utwente.nl

***Abstract*—This study investigates the impact of structured pruning on computational efficiency and robustness of EfficientNet-B3 on an Military Aircraft classification task. We evaluate effectiveness of filter pruning and find that pruning results in significantly lower inference latency with minimal performance impact. Especially a 30% pruning rate resulted in a average reduction of 2 ms per inference. We explored Gaussian noise injection during training, statistical analysis showed that it didn't improve or degrade our models performance.**

I. Introduction

Convolutional Neural Networks (CNNs) have become the backbone of many image classification tasks, demonstrating remarkable accuracy across diverse datasets. However, their deployment in real-world applications often come with high computational costs and are susceptible to performance degradation when exposed to noise and image corruption. This limitation is particularly relevant in domains such as military surveillance, where real-time classification of aircraft images necessitates both precision and computational efficiency to facilitate informed decision-making in dynamic environments.

This research investigates how introducing sparsity into CNN architectures impacts their accuracy and robustness under one type of noise, namely Gaussian noise. Sparse CNNs, which involve reducing the number of active parameters without significantly compromising performance, offer the benefits of computational efficiency, especially in resource-constrained environments. At the same time, noise injection during training has been shown to enhance model generalization and resilience to perturbations. Prior studies on sparsity's effect on models' robustness to perturbations provide limited insights and contradictory results [7].

EfficientNet-B3 was selected for this study due to its performance in image classification benchmarks. This choice is substantiated by its ranking among other top models and notebooks in Kaggle competitions. The EfficientNet family of models are introduced by Tan and Le [14] and are known for its compound scaling method, which balances network depth, with and resolution. This yields in a trade-off between computational efficiency and accuracy. While EfficientNet-B7 offers higher accuracy in the study, it does come with a increased complexity and higher memory requirements. As EfficientNet-B3 maintains a balance between accuracy and computational feasibility it is more optimal for our study on pruning and corruption.

The dataset employed in this study consists out of military aircraft images spanning around 70 distinct classes, it is around 10 GB in size. This scale can present a challenge for deep learning models as it is not an extensive dataset that can do exhaustive training. This dataset needs superior learning efficiency and generalization capabilities to get a high accuracy.

Therefore, our study will evaluate to what extent structured and unstructured pruning can be applied before the drop in performance for EfficientNetB3, when the test data is subjected to Gaussian noise. In addition, we will examine how noise injection during training will affect the performance of the sparse model, and if the potential performance degradation can be mitigated.

II. Literature review

The 2017 paper [9], Nazaré et al. studied the behaviour of deep convolutional neural networks when dealing with different types of image quality, by using images that where affected by salt-and-pepper noise or Gaussian

noise. The main finding of the study is that training networks with noisy images can improve the performance of models that must deal with images of varying quality. [7] studies the effect of structured and unstructured pruning on different models' robustness to different kinds of perturbation. The research points out limited insight in prior studies regarding this. The research indicates that weight and filter pruning improves the models' performance, or it remains unchanged. In contrast, channel pruning at 60 % deteriorates the performance for one of the considered models.

Image quality is important, a study done by Karam and Dodge [4] shows that neural networks are susceptible to quality distortions and recommends developing networks that are invariant to quality distortions as future work. A study done by da Costa et al [3], compares noises and highlights using Gaussian noise in training can improve the model's ability to handle varying image qualities. The Gaussian noise values used are often a zero mean with standard deviation (σ = 10, 20, 30, 40, 50). Similarly, Chang et al. in their work on stability training [20] applied Gaussian noise with σ = 0.01 until σ = 0.4, demonstrating that small levels of Gaussian noise during training enhance robustness and generalization without compromising image quality.

The study by Nazaré [9] highlights that noisy images in training can improve the models ability to handle varying image qualities. The key types of noise are Gaussian noise, Salt-and-Pepper noise, Speckle Noise, Poisson Noise [10]. Gaussian Noise involves adding random values from a Gaussian distribution to the image pixels (0-255, RGB). Salt-and-Pepper noise introduce random black and white pixels into the image. This emulates issues like faulty cameras or transmissions errors. Speckle Noise multiplies random pixel values by a random value. This creates a granular appearance. Poisson Noise is prevalent in low-light photography. As the aircraft images are coming from typical cameras and imaging sensors, instead of infrared and UV, a Gaussian noise will more closely mimic real-world distortions compared to speckle and salt-and-pepper. The extensive use of Gaussian noise in CNN training for image classification is justified by its prevalence in real-world imaging scenarios and the alignment of its statistical properties with the assumptions of many machine learning models. An important caveat we accept in developing is that no single type of noise is universally best across all domains. As mentioned in another paper [1], Image classification tasks involving RGB images are to be sufficient using only one of the noise types. In another paper by Zhang et al [19], varying standard deviations are used, specifically σ = 15, 25, 50, representing low, moderate and high noise.

Real-time applications deployed with CNN often need careful optimization. This is to reduce computational overhead without significantly compromising the performance. Two approaches have emerged to mitigate the computational costs for large model sizes and high computational requirements. This is Model Pruning and Optimizer Selection.

Pruning techniques are often categorized into structured and unstructured approaches. With structured pruning, we often remove entire layers, filters, or channels. [2]. This results in a smaller network. For unstructured pruning, we eliminate individual weights based on magnitude. This results in a sparser matrix that often requires more special hardware. The runtime during prediction is dominated by the evaluation of convolutional layers. A study by Molchavo et al [8] found that CNNs can be successfully pruned by iteratively removing the least important layers. Pruning is often performed iteratively: prune a certain fraction of parameters, then fine-tune, and repeat. Doing this tends to preserve accuracy better than a One-Shot prune [12]. Another study finds three categories for pruning methods [15]. These are methods that are magnitude-based. Methods that use clustering to identify redundancy. And methods that use sensitivity analysis to see the effect of pruning. The paper shows a large extent of possible pruning-methods that fall under these categories.

A. Unstructured Pruning methods

- Magnitude-Based Pruning: The idea of magnitude-based pruning is removing individual weights with the smallest absolute values. A study by Gale et al [5], compares various pruning strategies and found Magnitude pruning to be the highly competitive.
- Gradient-Based Pruning: Its core idea is to evaluate weight importance via gradient magnitude and remove the weights that affect loss minimally.
- Random Pruning: Randomly prune weights to a certain sparsity level. This often serves as a control to see how intelligent the pruning algorithm is against random.
- Movement Pruning: The approach of movement pruning is where the pruning mask follows the movement of the weights [13].

While unstructured pruning can offer very high compression ratios, the resulting weight matrices are often irregularly sparse, requiring specialized hardware or libraries to fully leverage the speedups [5].

As unstructured pruning produces irregular sparsity and will require specialized software we estimate this will be a time constraint. Structured pruning will directly reduce

the number of operations. To get results we will use some method of structured pruning.

B. Structured Pruning Methods

Possible choises for structured pruning are:

- Filter Pruning (Kernel Pruning): This method removes entire convolutional layers based on a criteria, such as contribution to output. Filters with the least significance often measured using L1-norm or SDG are pruned [8].
- Channel pruning: This method removes convolutional layers based on their activation values. Where Filter pruning removes the entire filter, channel pruning only removes specific parts of the input/output connections. [6].
- Layer Pruning: A layer is a collection of operations that is applied to the input data. By removing layers we significantly reduce network depth. A study from Wang et al [16] found that pruning layers outperforms pruning filters.
- Block pruning: This prunes groups of layers and is effective for models that have repeating patterns or blocks.
- Structured Sparsity Learning: This regularizes filter, channel, filter shape and depth structures [17].

As we highly value practicality and effectiveness in inference time and memory usage, we think Filter Pruning or channel pruning will be best for our project. A common way to deploy filter pruning is done with the l1-norm of filter weights. Its also a straightforward metric to rank filter importance.

C. L1-Norm Based Pruning

L1-norm based pruning is a structured pruning technique where filters (kernels) in a convolutional neural network (CNN) are pruned based on the L1-norm of their weights. Filters with the smallest L1-norm values are considered less important and are pruned.

The L1-norm of a filter is defined as the sum of the absolute values of all its weights. For a filter W with dimensions $C \times K \times K$:

- C: Number of input channels.
- K: Kernel size (e.g., 3×3).

The L1-norm of filter W is computed as:

$$\|W\|_1 = \sum_{c=1}^{C} \sum_{i=1}^{K} \sum_{j=1}^{K} |w_{c,i,j}|$$

Where $w_{c,i,j}$ represents the weight at position (i, j) in the c-th channel of the filter.

1) Steps for L1-Norm based pruning:

1) Compute L1-Norm for each filter: For every filter in a convolutional layer, calculate its L1-norm using the formula above.
2) Rank filters: Sort the filters in ascending order of their L1-norm values. Filters with smaller norms are considered less significant.
3) Prune filters: Remove a fraction of filters (e.g., 20%) with the smallest L1-norm values.
4) Fine-Tune the model: Retrain the pruned model to recover performance lost during pruning.

2) Practical Example: For a convolutional layer with 64 filters, each of size 3×3 with 3 input channels ($C = 3$):

1) Compute the L1-norm for each filter:

$$\|W_f\|_1 = \sum_{c=1}^{3} \sum_{i=1}^{3} \sum_{j=1}^{3} |w_{c,i,j}|$$

2) Rank all 64 filters based on their L1-norm values.
3) Prune the filters with the lowest L1-norm values (e.g., the bottom 20%).

3) Fine-tuning the pruned model: Once filters are pruned, fine-tuning the model is crucial to recover any lost performance. The steps include:

- Retrain the pruned model using the same dataset and a reduced learning rate.
- Use a standard optimizer such as Stochastic Gradient Descent (SGD) with Momentum.
- Apply regularization techniques like weight decay to prevent overfitting during fine-tuning.
- Monitor validation metrics during training and apply early stopping if the validation loss stops improving.

D. Additional Model Optimization Techniques

After pruning there are additional methods one can apply to further fine-tune and optimize the model.

1) Optimizers: Stochastic Gradient Descent Variants performed best according to paper [11]. The SGD with momentum remains a popular choice for many pruning techniques. Adaptive Methods (e.g., Adam, RMSProp) can speed up training, but sometimes lead to suboptimal generalization, particularly in pruned or quantized models, the paper by Wilson et al [18] observe that that adaptive methods generalize worse then SGD and should be reconsidered for use.

2) Quantization: Reduces the remaining parameters to a lower bit floating-point. This reduces the storage use per operation.
3) Knowledge Distillation: A smaller "student" network learns to mimic a larger, pre-trained "teacher" network. Distillation is orthogonal to pruning; a pruned model can be further distilled or vice versa
4) Neural Architecture Search (NAS): Automated search algorithms (e.g., reinforcement learning or evolutionary algorithms) to find architectures that are both accurate and efficient [21]. While NAS is computationally heavy, it has been used in tandem with pruning to yield specialized architectures that require fewer FLOPs.

III. Methodology

The study's overall goal is to improve the image-detection convolutional neural network's (CNN) computational performance to allow for the effective utilization of the aforementioned models in local deployment and edge-computing scenarios.

To achieve the goal, we determine whether a drop in performance inherent to model optimization can be offset by maximizing the efficiency of the leftover components through the training stage data augmentation. The literature review above pointed toward Gaussian image corruption as an all-rounded option for data augmentation while suggesting structured and unstructured pruning as popular model optimization strategies. Based on the above, the following research question encapsulates the target of this study:
"How can we decrease the model computational performance through optimization while offsetting the loss in performance by increasing the efficiency of the leftover model components by inducing Gaussian image corruption during the training stage?".
To answer the above question, the following sub-questions are answered:

1) How can we improve the predictive performance of image-detection models based on CNN architecture by inducing Gaussian image corruption during the model training stage?
2) How can we reduce the model's computational cost through structured and unstructured pruning, and what is the tradeoff between computational cost and predictive ability?
3) How can we reduce the model's computational cost while retaining most of the performance by maximizing the model's efficiency through training data augmentation with Gaussian corruption?

A. Metrics

To answer the research questions, we define the following metrics as cornerstones of this study:

- Accuracy - all-rounded model performance metric.
- Number of disagreements between models as a performance indicator suitable for hypotheses testing (see table I). We cannot directly conduct hypothesis testing for accuracy as a metric, but we can use a common workaround of focusing on points where models disagree on the same observation. This approach utilizes the fact that instances, when both models agree, do not add any information to the comparison of their accuracy, but the cases where the models disagree do. We use this metric to test for statistical significance of accuracy differences between the models.
- Inference latency - latency of the model during inference.

Inference memory footprint is not considered in this study due to JupyterLab and Python constraints. In the current project architecture, accurately recording memory usage is infeasible. Although memory usage is expected to drop due to optimization, we cannot reliably verify this.

B. Dataset and model

This study will use a military aircraft dataset with 70 classes. The tested model will be EfficientNet-B3.

C. Image corruption

In this section, we answer the first research sub-question: How can we improve the predictive performance of image-detection models based on CNN architecture by inducing Gaussian image corruption during the model training stage?. We test two models: original - without data augmentation and alternative - trained on the augmented dataset (through expansion) with a constant corruption strength of 10%, determined from the literature review above and prior testing. We will utilize McNemar contingency table test for the first sub-research question. The McNemar test procedure allows to test for predictive performance across a sample of paired observations as an upgrade to chi-square. Due to the nature of this study, each model is tested on the same data sample, hence, the McNemar test is suitable as it satisfies the pairwise dependency requirement and prevents bias that would result from a chi-square test. The McNemar test is typically two-sided, so we adapt it to a one-sided scenario to better suit the aforementioned research questions. Another alternative would be a binomial sign test on the difference between disagreements, but the results would be similar due to the sufficiently high sample size. The McNemar test is non-parametric and requires an assumption of independence between different pairs of observations. If we consider whether a model is correct as a dependent variable and observed pairs of predictions

for each data point in the test set as independent variable, the assumption is satisfied and the test has merit.

TABLE I
ORIGINAL VS ALTERNATIVE MODEL PREDICTIONS

	Alternative Model	
Original Model	**Correct**	**Incorrect**
Correct	n_{11}	n_{10}
Incorrect	n_{01}	n_{00}

1) McNemar Test: Let each test example be drawn from the same data distribution. For any such instance, we denote the probability that the original model is *correct* on a given instance as p_O, and the probability that the alternative model is *correct* on the same instance as p_A.

We want to test whether the alternative model has a significantly higher probability of being correct in the long run. Sequentially, we can define the following hypotheses:

$H_0 : p_O = p_A$ (No difference in correctness probability),
$H_1 : p_O < p_A$ (Alternative model is more accurate).

Now, we define n_{01} as the number of instances in which the original model is *incorrect* and the alternative model is *correct* (see table I). Likewise, we define n_{10} as the number of instances where the original model is *correct* and the alternative model is *incorrect* (see table I).

Under H_0, the disagreement between models is equally likely to lie in both n_{01} and n_{10}, hence n_{01} follows binomial distribution with number of trials: n_{01} + n_{10} and probability of success 0.5. Consequently, binomial-normal approximation allows us to define the following test statistics (McNemar statistics):

$$\chi^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}}.$$

Under H_0, $\chi^2 \sim \chi^2_{(1)}$.
2) Decision rule: If we define p-value as:

$$p\text{-value} = P\big(\chi^2_{(1)} \geq \chi^2_{\text{obs}}\big),$$

The decision rule becomes:

$$\text{Reject } H_0 \text{ if } \big(n_{01} > n_{10}\big) \text{ and } \frac{\text{p-value}}{2} < \alpha,$$

$$\text{Fail to reject } H_0 \quad \text{otherwise.}$$

This is an adaptation of a two-tailed McNemar test to support unidirectional hypothesis testing. First, we confirm that the direction of the disagreement satisfies our hypotheses. Second, we test the halved p-value to account only for the unidirectional hypothesis.

D. Model optimization

In this section, we will test the impact of model optimization on inference times, adhering to the second research question. Based on the literature review above, we will perform filter-based pruning with an L1-norm ranking vector. The resulting model will then be re-trained with a reduced learning rate and an SGDM (Stochastic Gradient Descent with Momentum) optimizer.

The trade-off between inference latency and model accuracy will be presented as two side-by-side graphs. The graphs will display validation and test set results, but we will only use results from the validation set to select pruning strength for the last research question and a test set to answer this section's research question. This will allow us to prevent "selection bias," as our test set remains constant between experiments.

- Average inference latency across test samples with respect to filtering strength
- Average accuracy with respect to filtering strength

After analyzing the trade-off, we will select the model with the most promising accuracy and inference times and test for statistical significance. This section and results will be based on the test set but not carried forward to the combination experiment, which will be based purely on validation results from this section.
1) Wilcoxon Signed-Rank Test: We test whether there is statistically significant evidence favouring the claim that the pruned model has lower inference times than the baseline. Let $t_i^{(\text{baseline})}$ and $t_i^{(\text{pruned})}$ denote the inference times of the baseline and pruned models for the i-th sample $\big(i = 1, 2, \ldots, n\big)$ respectively. Then, if we define the paired differences as

$$d_i = T_i^{(\text{baseline})} \; - \; T_i^{(\text{pruned})}$$

, we can test if the pruned model has lower inference times by testing whether the mean of the difference μ is bigger than 0. Formally, the hypotheses become:

$$H_0 : \mu = 0,$$
$$H_1 : \mu > 0.$$

We use the Wilcoxon Signed-Rank Test, a nonparametric procedure for paired data, to test the above hypotheses:

1) **Rank the Absolute Differences** $|d_i|$ (excluding any $d_i = 0$), assigning a rank r_i from smallest to largest.
2) **Sum the Positive Ranks**:
$$W^+ \; = \sum_{\{i : d_i > 0\}} r_i.$$
3) **Obtain the p-Value** by comparing W^+ against its reference distribution under H_0.

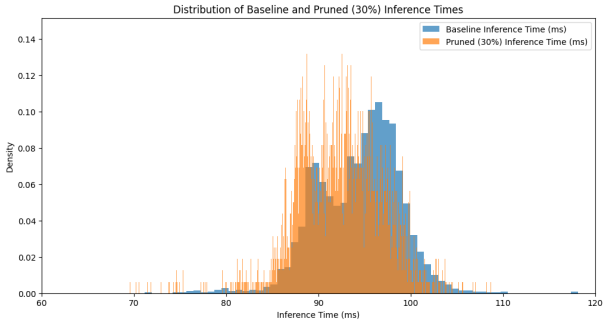

Fig. 1. Inference time of the selected models

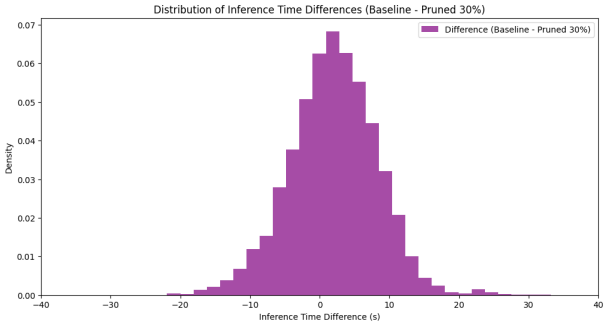

Fig. 2. Difference in inference time

4) **Decision**: Reject H_0 in favor of H_1 if $p \leq \alpha$. Fail to reject otherwise.

2) McNemar Test: We will also check for the difference in accuracy by repeating the McNemar procedure defined above.

E. Model optimization and corruption

In this section, we test whether image corruption can enhance model pruning, leading to lower inference times and similar or better prediction performance. We train two models, the baseline and the 30% pruned trained (and fine-tuned), on an extended dataset with original images and 10% corrupted images. We record both inference times and predictions and perform the above-stated Wilcoxon test for the difference in inference times and the McNemar test for the difference in predictive performance. Furthermore, we test for the difference in predictive performance between the 30% pruned with 10% corruption and the 30% model using a two-tailed McNemar test exploring whether the corruption on top of pruning impacts predictive performance.

IV. Results

A. Image Corruption

We observed that the model trained on corrupted images had slightly higher accuracy and a lower number of false disagreements II. However, with $p = 0.19$, we do not reject the null hypothesis for the standard $\alpha = 0.05$. There is no evidence that image corruption results in a statistically significant increase in the accuracy of the model in question.

B. Model optimization

We have found that model optimization through filtering reduces average inference times without noticeable accuracy cost (see Figure: 3). Notably, we found the decrease to be, on average, 2 ms with a standard deviation of 14 ms. Based on the test graph, we found the filtering strength of 30% to result in the most noticeable drop in inference times at no substantial performance cost. We use this model to illustrate and analyze the results of pruning. Coincidentally, the validation set gave the same result, so we will also use the 30% model for the next section, but based on the validation set, not the test set. The difference in inference times can be viewed on the Figure 1 and 2. With $p = 5.478e^{-75}$ we reject the null hypothesis for the standard $\alpha = 0.05$. Hence, the pruned model at 30% consistently has lower inference times. Furthermore, McNemar test on the prediction differences $p = 0.0006$ showing that the pruned model had a statistically significant increase in accuracy over the baseline model.

TABLE II
ORIGINAL VS ALTERNATIVE MODEL PREDICTIONS

	Alternative Model	
Original Model	**Correct**	**Incorrect**
Correct	2792	123
Incorrect	138	175

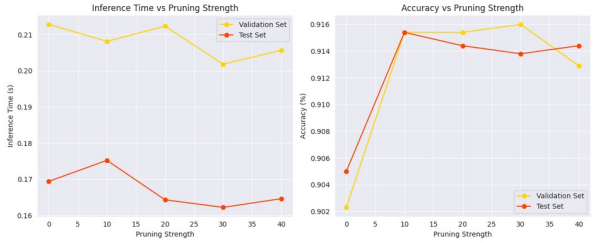

Fig. 3. Optimization strength vs Inference time and Accuracy

C. Pruning and Image corruption

We observe the same decrease in inference time with the mean change of 2 ms in favour of the pruned and corrupted model. However, the standard deviation almost doubled to 24 ms. The results are available in Figures ???4 and ???5. The hypothesis testing also confirmed the same trend as observed in the previous section, with

optimized and corrupted models having both slightly higher accuracy and lower inference times (both statistically significant with extremely low p-values). However, when we tested the predictive performance against the pruned model without corruption, we found the two-tailed McNemar test p to be around 0.95, confirming the results of the previous image corruption test that there is no statistically significant improvement in predictive performance from image corruption. The improvement in both inference times and accuracy stems solely from pruning rather than image corruption.

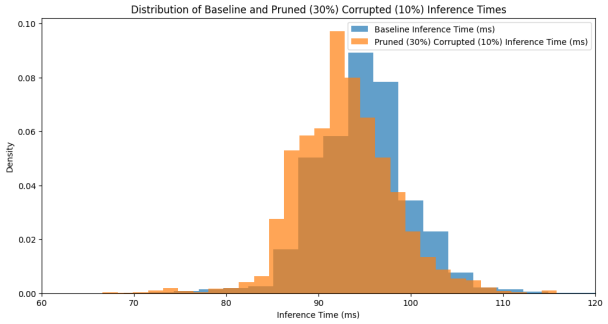

Fig. 4. Inference times for pruned-corrupted model and the baseline

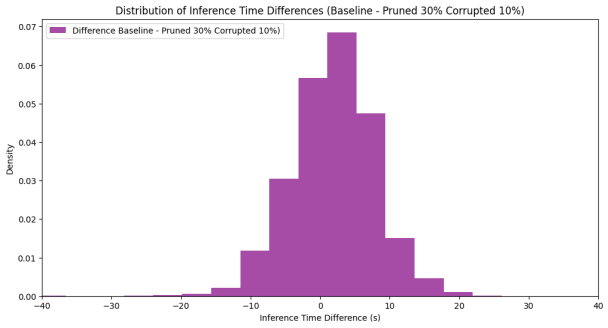

Fig. 5. Difference in inference times for pruned-corrupted model and the baseline

V. Conclusion??

We found that pruning effectively reduces inference times while slightly increasing predictive performance in image classification tasks utilizing CNNs and, specifically - EfficientNet-B3. We also observed that Gaussian image corruption did not produce the expected outcomes and presented no improvement in predictive performance, whether used both with pruning or as a standalone. Consequently, the use of image corruption in scenarios with sufficiently large datasets and high predictive performance of the barebone CNN models is questionable and should be carefully considered.

A. Discussion

We contribute to research by showing the effectiveness of structured pruning without sacrificing accuracy.

An area of interest is the potential trade-off between computational efficiency and interpretability. Pruning reduces model complexity and improves inference speed, it may also affect feature representation within the CNN. Future research could investigate how different pruning levels impact interpretability with metrics such as activation map visualization and explainability frameworks.

Our study has several limitations. The dataset size (10GB) imposes constraints on model generalization, as a larger and more diverse dataset could yield different outcomes regarding pruning efficiency and noise robustness.

We limited our evaluation to Gaussian noise, whereas real-world applications may involve multiple forms of noise and distortions or combinations and different scales, requiring a broader investigation. Also as it did not prove significant in model robustness it might suggest that our dataset characteristics don't work well with the model.

Future work should explore whether similar pruning techniques hold across deeper architectures such as EfficientNet-B7. Lastly, the study focused on inference time and classification accuracy as primary metrics; however, additional factors such as memory footprint and energy consumption should be examined for practical deployment scenarios.

Appendix

Appendix I: AI tools

During the preparation of this work the author(s) used ChatGPT or similar Generative AI in order to make this project. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the work. AI has been used to gather research papers and in analytical coding. It has been used to check the paper for textual errors and grammar mistakes.

Appendix II: Github coding

Dataset URL

Appendix III: Kaggle Military Aircraft Dataset

Dataset URL

References

[1] M. E. Akbiyik. Data augmentation in training cnns: Injecting noise to images, 2023.

[2] S. Anwar, K. Hwang, and W. Sung. Structured pruning of deep convolutional neural networks, 2015.

[3] G. B. P. da Costa, W. A. Contato, T. S. Nazare, J. E. S. B. Neto, and M. Ponti. An empirical study on the effects of different types of noise in image classification tasks, 2016.

[4] S. Dodge and L. Karam. Understanding how image quality affects deep neural networks. In *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, 2016.

[5] T. Gale, E. Elsen, and S. Hooker. The state of sparsity in deep neural networks, 2019.

[6] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks, 2017.

[7] P. Mitra, G. Schwalbe, and N. Klein. Investigating calibration and corruption robustness of post-hoc pruned perception cnns: An image classification benchmark study. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3542–3552, 2024.

[8] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference, 2017.

[9] T. S. Nazaré, G. B. P. da Costa, W. A. Contato, and M. Ponti. Deep convolutional neural networks and noisy images. In M. Mendoza and S. Velastín, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 416–424, Cham, 2018. Springer International Publishing.

[10] S. S. Pawan Patidar. Article:image de-noising by various filters for different noise. *International Journal of Computer Applications*, 9(4):45–50, November 2010.

[11] R. Poojary and A. Pai. Comparative study of model optimization techniques in fine-tuned cnn models. In *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pages 1–4, 2019.

[12] A. Renda, J. Frankle, and M. Carbin. Comparing rewinding and fine-tuning in neural network pruning, 2020.

[13] V. Sanh, T. Wolf, and A. Rush. Movement pruning: Adaptive sparsity by fine-tuning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20378–20389. Curran Associates, Inc., 2020.

[14] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[15] S. Vadera and S. Ameen. Methods for pruning deep neural networks. *IEEE Access*, 10:63280–63300, 2022.

[16] Z. Wang, C. Li, and X. Wang. Convolutional neural network pruning with structural redundancy reduction, 2021.

[17] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks, 2016.

[18] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning, 2018.

[19] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[20] S. Zheng, Y. Song, T. Leung, and I. Goodfellow. Improving the robustness of deep neural networks via stability training, 2016.

[21] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning, 2017.