The 28th Meeting on Image Recognition and Understanding

Leveraging Gradient-Based Pruning for Efficient Neural Networks in Crop Disease Detection

Carlos PADEIRO[1,a)] Tse-Wei CHEN[1,b)] Takahiro KOMAMIZU[1,c)] Ichiro IDE[1,d)]

Abstract

Pruning is an essential technique for reducing the computational demands of Deep Neural Networks (DNNs) like Convolutional Neural Networks (CNNs) and Transformer, enabling deployment on resource-constrained devices. This particularly benefits crop disease detection in remote agricultural areas where expert diagnosis is costly and unreliable. We propose a novel pruning method based on gradient analysis, which identifies and removes less significant weight channels by measuring output variations from small perturbations in intermediate layers. The proposal method achieved good performance on models like ConvNeXt-T and SwinTransformer-T, outperforming benchmarks by gaining 0.88% and 0.68% mAP@50 for ConvNeXt-T and SwinTransformer-T backbones, respectively.

1. Introduction

Pruning is essential for decreasing the computational limitations of Deep Neural Networks (DNNs), allowing them to work on resource-limited devices. In agriculture, this is particularly useful for crop disease detection, which poses a significant challenge to global food security, causing substantial production and economic losses, estimated at $220 billion annually by the United Nations Food and Agriculture Organization [9]. These diseases result in insufficient human food supply and disrupt farmers' income-generating activities. Maize, which is a dominant food crop, is particularly susceptible to various diseases despite its high global yield of $1.2 billion in 2020, with a productivity of 6.0 t/ha [10].

Conventional crop disease diagnosis methods rely on visual inspection by experts, utilizing their in-depth knowledge of crop diseases and their symptoms [1]. This process is time-consuming, expensive, and prone to human error due to subjective perception. Despite revolutionizing precision agriculture, strong DNNs (esp., Convolutional Neural Networks (CNNs) and Transformers), for disease detection are computationally expensive for resource-limited farmers due to over-parameterization, a common characteristic of DNNs [7]. Moreover, Internet connectivity issues in remote cultivation areas, need to be considered.

To this end, we propose a lightweight object detection model to be deployed in low-resource devices in rural areas following our previous solutions [30]. Pruning [18] is the process of simplifying neural networks by removing unimportant and redundant weights, channels, filters, or neurons. It is usually divided into three types: unstructured [11, 32], semi-structured [12, 26, 29], and structured [8, 27, 35]. Unstructured pruning (a.k.a., non-structured pruning or weight-wise pruning) is the finest-grained case that can quickly achieve a higher compression ratio and accuracy on specialized hardware. Semi-structured pruning (a.k.a., pattern-based pruning [26]) achieves high accuracy and structural regularity simultaneously. Structured pruning removes entire filters [39], channels [2, 22], Transformer attention heads [27], or even layers [28] after pruning method does not require specialized hardware. The proposed method uses structured pruning due to its hardware compatibility and its practical benefits in terms of latency and memory improvements. This efficiency gain, however, may lead to a trade-off in accuracy compared to vanilla models.

In contrast to previous studies, we propose BlockWIPG (Block Weight and Image Perturbation-induced Gradients) which estimates channel importance; In summary, the main contributions of this paper are two fold:

(1) **BlockWIPG**: We propose a new pruning methodology that investigates the contribution of intermediate channels in layer or block to image variation through gradient-based analysis.
(2) **Sensitivity analysis**: We provide a systematic block sensitivity analysis and distribution prunability investigation in the model architecture based on CNN and Transformer-based models.

2. Related Work

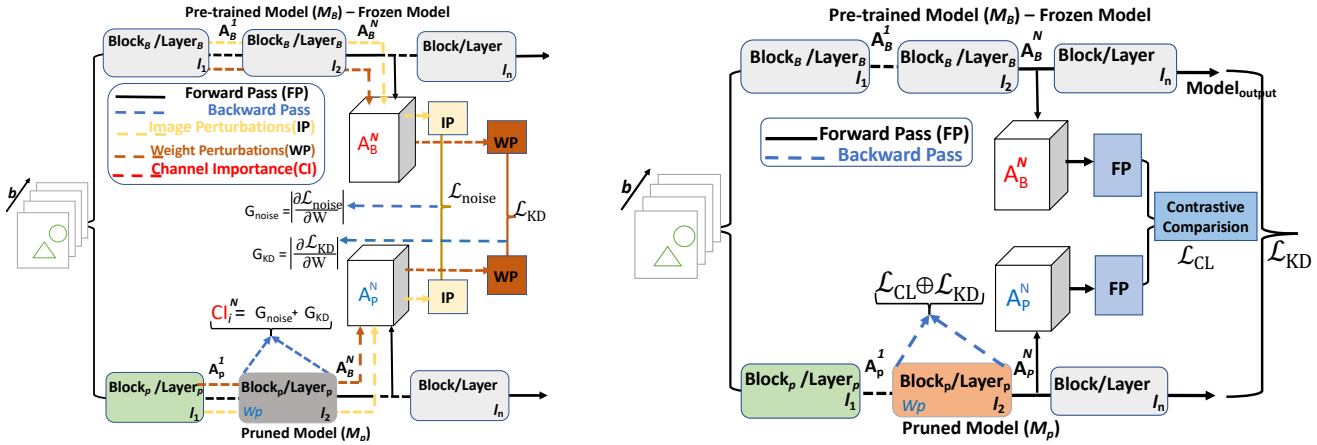
The proposed BlockWIPG adopts structured pruning due to its alignment with general-purpose hardware and its practical advantages in reducing latency and memory usage without requiring specialized infrastructure. Here, pruning is formalized as:

$$\arg\min_{W_P} L(x; W_p) \quad \text{s.t.} \quad \|W_p\|_0 < N, \tag{1}$$

where L denotes the objective function for neural network training, x is the input, W_p are pruned weights, $\|W_p\|_0$ represents the number of none-zero weights in W_p, and N is the target number of none-zero weights in W_p. We remove the weights tensor in channel pruning.

The channel output size $c_{\text{op}} = c_o(1 - s)$ and input size $c_{\text{ip}} = c_i(1 - s)$ after pruning are used to calculate the channel to be kept, where c_o and c_i denote the original channel output and input sizes, and s is the sparsity pruning ratio ranging between 0

[1] Nagoya University, Graduate School of Informatics
[a)] padeiroc@cs.is.i.nagoya-u.ac.jp
[b)] twchen@ieee.org
[c)] taka-coma@acm.org
[d)] ide@i.nagoya-u.ac.jp



(a) Channel Importance (CI) estimation

(b) Fine-tuning

Fig. 1: Overview of the proposed method Block Weight and Image Perturbation-induced Gradients(BlockWIPG). which investigates image perturbations and small gradient-based (G_{noise}, G_{KD}) weight perturbations. Bold grey layers/blocks are pruned. CI_i^N is the channel's importance of the absolute value by measuring changes in the output of intermediate layers or blocks in (a). Fine-tuning is applied to layers or blocks, which can use data with or without label or annotations to compute (L_{CL}, L_{KD}). Red block's are fine-tuned in (b).

and 1.

Recent pruning methods have focused on evaluating the Channel Importance (CI) in neural networks, selectively removing those considered less significant based on various criteria [14, 16, 38]. Li et al. [21] propose using the sum of absolute weight values to assess CI. In contrast, Liu et al. [24] aim to minimize the reconstruction error between pruned and original models. Different methods, such as those by He et al. [17] and Tang et al. [33], emphasize the role of input data in determining channel relevance, advocating for input-specific subnetworks model. Although evaluating the impact of removing individual channels on final loss is theoretically sound, it becomes computationally infeasible in deep networks due to the diminishing influence of shallow layers.

3. Methodology: BlockWIPG

In DNNs, especially as their depth increases, it becomes increasingly complex to determine and eliminate channels that contribute slightly to the model's inference capacity. To address this, we propose an intermediate computational measure that evaluates and selects less significant channels for pruning, named Block Weight and Image Perturbation-induced Gradients(BlockWIPG) thereby enhancing the efficiency and accuracy of the pruning process. It was inspired by works by Chung et al. [6] and Ilhan et al. [22]. The overall pipeline of BlockWIPG is illustrated in Figure 1.

3.1 Weight Perturbation (WP)

Building on the effectiveness of Taylor expansion-based weight importance estimation, we define the importance of each channel by quantifying the impact of slight weight perturbations within the channel group on the output of intermediate layers or blocks. The output of the pre-trained model, $M_{\theta_n}(y \mid x)$, changes when the weight θ_n is perturbed to $\theta_n + \beta$, where β is a slight weight perturbation. This change is approximated by the Kullback-Leibler (KL) divergence $D_{\text{KL}}(M_{\theta_n}(y \mid x) \| M_{\theta_n+\beta}(y \mid x))$ between the two output distributions. The expected KL divergence is:

$$\mathbb{E}_x(D_{\text{KL}}(M_{\theta_n}(y \mid x) \| M_{\theta_n+\beta}(y \mid x))) \approx \beta^2 H_{\theta_n} + O(\beta^3) \quad (2)$$

where x is the input data or features, y is the output features or label, and H_{θ_n} is the n-th diagonal value of the Fisher information matrix, defined as:

$$H_{\theta_n} = \mathbb{E}_{x \sim M(x)} \left[\mathbb{E}_{y \sim M(y|x)} \left(\frac{\partial \log M_{\theta_n}(y \mid x)}{\partial \theta_n} \right)^2 \right] \quad (3)$$

As evident from Eq. 2 and Eq. 3, the Fisher information quantifies the influence of the perturbed weight on the output of the model. To reduce computational complexity, the following approximation is applied for the expectation [34], where the importance score I_p for the weight channel c is given by:

$$I_p = \sum_{\theta_n \in Q_c^m} \left(\frac{\partial \log M_{\theta_n}(D_s)}{\partial \theta_n} \right)^2, \quad (4)$$

where Q_c^m is the m-th weight within the channel group and $\frac{\partial \log M_{\theta_n}(D_s)}{\partial \theta_n}$ is the mean derivative of the pre-trained model output concerning θ_n over D_s. It is used to reduce the misalignment between all perturbed weights and weights of the pre-trained model.

3.2 Image Perturbation (IP)

BlockWIPG needs to analyze the sensitivity of the model's weights within each channel group to data variations. We hypothesise that the pre-trained model's response to such variations could identify the weights most critical for its predictions. Thus, preserving these channels can improve the performance and generalisation capability of the pruned model. Given an image x and a sampled noise vector γ (such as Gaussian noise vector), we compute the difference between two images $M(x)$ and $M(x+\gamma)$ performed by the pre-trained model $M(\cdot)$ using the L_1 distance as:

$$\mathcal{L}_{\text{noise}} = |M(x) - M(x+\gamma)|, \quad x \in \mathbb{R}^{h\times w\times 3}, \tag{5}$$

where $|\cdot|$ denotes the absolute value, and w and h represent the width and height of the images, respectively. We can compute the gradients of loss generated by diference between the image noising and normal image by applying back-propagating in Eq. 5. These gradients are represented as:

$$G_{\text{noise}} = \left|\frac{\partial \mathcal{L}_{\text{noise}}}{\partial W}\right|, \quad G_{\text{noise}} \in \mathbb{R}^{c_i\times c_o\times k_1\times k_2}, \tag{6}$$

where c_i and c_o represent the number of input and output channels, and k_1 and k_2 the kernel size of W, respectively. To mitigate the impact of noisy outlier gradients caused by image perturbations from Eq. 6, we employ an average-based scoring mechanism. Specifically, for each image perturbations, we calculate the average gradient magnitudes across N perturbation and use these values to adjust the induced gradients for the learnable weights within each channel group. Thus, the importance score $I_n(c) \in \mathbb{R}^1$ of a channel c based on image perturbations can be defined as:

$$I_n(c) = \left\|\mathbb{E}_z(c)\mathbb{E}_d(c)\left[(G_{\text{noise}} - \mathbb{E}_d[G_{\text{noise}}])^2\right]_c\right\|_1, \tag{7}$$

where $\mathbb{E}_d[G_{\text{noise}}]$ denotes the gradient compensated for the original image vector x, and $[\cdot]^2$ denotes element-wise square. Given this, the overall importance score for each channel can be determined by performing an element-wise addition of the terms in Eq. 4 and Eq. 6. Therefore, the final channel importance can be defined as:

$$\mathcal{I}_m(c) = \gamma I_n(c) + \beta I_p(c). \tag{8}$$

Where, γ and β represent each channel's regularized channel importance score. We noted that setting γ to 1.0 and β to 0.2 yielded the best results through all the experiments. Thus, all the pruning experiments are under these settings. As a result, we identify the channel groups to prune by ranking the channels according to their importance scores (Eq. 8).

4. Experimental Evaluation

This section comprehensively evaluates the proposed Block-WIPG pruning method, comparing its performance against state-of-the-art methods on a maize disease detection dataset. It also includes ablation studies across various sparsity levels to analyze its effectiveness and underlying contributions.

4.1 Experimental Setup

4.1.1 Dataset

The dataset used in this experiment comprises annotated maize leaf images from three disease classes: *Northern Corn Leaf Blight* (NLB), *Fall ArmyWorm* (FAW), and *Maize Streak Virus* (MSV) collected from four public sources, including 18,222 images with 105,735 NLB lesions from the USA [36], and 15,468 images with FAW and MSV annotations from Ghana [6], Uganda [3], and Namibia [4]. All images were downsampled to 512×512 pixels resolution and split into training, validation, and test sets in a $70 : 20 : 10$ ratio, with the training and validation sets used for model training and the validation and test sets for evaluating the pruned models.

4.1.2 Metrics

We consider five key metrics: *Inference Time*, *Multiply-And-Accumulate (MAC)* operations, *FLoating point OPerations (FLOPs)*, *Parameters (Param)* count, and *Accuracy* measured by mean Average Precision (mAP). These metrics collectively reflect the computational cost, energy efficiency, memory footprint, and predictive performance of the model, comprehensively assessing its suitability for deployment in resource-constrained device.

4.1.3 Implementation Setup

We performed inferences for a large Red, Green, and Blue (RGB) image with the size of $4,032\times 3,024$ pixels on an NVIDIA RTX A6000 GPU and an Intel Core i9 CPU (2.3GHz 8-core). We tested this method on object detectors: Faster R-CNN [31] under ConvNeXt-T [25] and SwinTransformer-T [23] backbones, implemented by PyTorch. After pruning, the model is reconstructed block-by-block to recover the accuracy. Batch sizes 32 and 90 epochs, Adam optimizer [19], and the initial learning rate 10^{-3} were used.

4.2 Block Sensitivity Analysis

In this paper, we perform block sensitivity analysis by pruning one block at a time to observe the resulting accuracy degradation between the model before and after one specific block pruning. The block which causes a high accuracy drop is considered more sensitive, otherwise, it is less sensitive. Unlike prior works [15] which focus on layer-level sensitivity analysis, the propsed Block-WIPG method concentrates specifically on block-level sensitivity since we compute channel importance estimation for pruning based on gradient analysis within the model block. We evaluate block sensitivity within distinct model architecture groups to determine the sparsity pruning ratio.

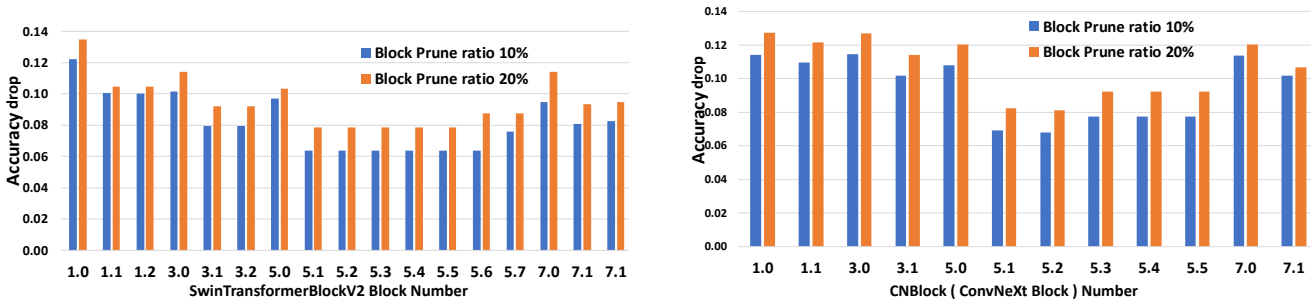
For instance, in group 1, Block 1.0 exhibits higher sensitivity (larger mAP drops), whereas in group 3, Blocks 3.1 and 3.2 demonstrate lower sensitivity (smaller mAP drops) in Fig. 2b. This differentiation allows targeted pruning strategies based on block-level criticality. The overall results are shown in Fig. 2. For instance, SwinTransformerBlockV2 block numbers (1.0 ; 3.0 ; 5.0 ; 7.0) are more sensitive to pruning in Fig. 2a and Fig. 2b. Meanwhile, ConvNeXt CNBlock block number (1.1 ; 1.2 ; 3.1 ; 3.2 ; 5.2..5.5 ; 7.1) and SwinTransformerBlockV2 block numbers (1.1 ; 1.2 ; 3.1 ; 3.2 ; 5.2..5.7 ; 7.1 ; 7.2) are less sensitive to pruning in Fig. 2a and Fig. 2b, respectively. To address this, we propose Eq. 9 and Eq. 10 to calculate the sparsity pruning ratio for more sensitive and less sensitive blocks, respectively.

$$p_r^m = p_r + \text{round}(p_r \cdot \tau, 2) \tag{9}$$

$$p_r^l = \text{round}(p_r \cdot \tau, 2) \tag{10}$$

Here, p_r is the initially desired pruning ratio of the block, and p_r^m and p_r^l are new pruning ratios of the more sensitive and less sensitive block, respectively, and $\tau (0 \leq \tau < 1)$ is the threshold value to control the number of parameters removed and the accuracy drop. The rounding function $\text{round}(\cdot, 2)$ rounds a given value in two decimal points. We set τ to 0.20 for results across all experiments.

Furthermore, SwinTransformer-T [56] and ConvNeXt-T [57]



(a) SwinTransformer-T [23] (b) ConvNeXt-T [25]

Fig. 2: Block sensitivity in mAP scores on ImageNet-ILSVRC2012 [20] classification dataset with 10% and 20% of parameters removed within a block, and without fine-tuning the pruned model.

Table 1: Benchmark of the proposed BlockWIPG method across various method's on crop disease detection test set. The best score in each column is bold-faced.

Detector	Backbone	Method	Top-1 mAP@50[%] (↑)			#Parm[M] (↓)		GFLOPs[G] (↓)		
			Vanilla	Pruned	Δ(↓)	Vanilla	Pruned	Vanilla	Pruned	Δ(↑)
Faster R-CNN [31]	ConvNeXt-T [25]	ViT-Slim [5]	**45.78**	41.49	4.49	**149.54**	93.75	**233.12**	153.11	80.01
		Isomorphic [13]	**45.78**	41.79	3.99	**149.54**	**93.72**	**233.12**	151.60	81.52
		BlockWIPG (proposed)	**45.78**	**42.67**	**3.11**	**149.54**	**93.72**	**233.12**	**149.20**	**83.92**
	SwinTransformer-T [23]	ViT-Slim [5]	**43.15**	41.47	1.68	**149.54**	96.51	**233.12**	169.93	63.19
		Isomorphic [13]	**43.15**	41.89	1.26	**149.54**	**96.48**	**233.12**	169.23	63.89
		BlockWIPG (proposed)	**43.15**	**42.57**	**0.58**	**149.54**	**96.48**	**233.12**	**168.22**	**64.90**

blocks show particular sensitivity behaviour to pruning. To support this, we leverage block sensitivity distribution analysis, illustrated in Fig. 2a and Fig. 2b. We evaluate block sensitivity distribution within distinct model architecture groups. We noticed that the first block within the model architecture group is more sensitive than those within the same model architecture group. For instance, in Fig. 2a and Fig. 2b, the model's architecture group 1, Block 1.0, exhibits the highest sensitivity, whereas Block 1.1 shows the lowest sensitivity. A similar trend continues in the models' architecture groups 3, 5, and 7. In the models' architecture group 5, Block 5.0 displays the highest sensitivity, while Block 5.2 indicates the lowest sensitivity. Furthermore, we found that along the blocks in the model architecture, the behaviour follows the first blocks, which are more sensitive than the intermediate ones, and the intermediate ones are less sensitive than the final blocks. This behaviour along the blocks is aligned with findings by Yang et al. [37].

4.3 Evaluation Results

We extensively compare the proposed BlockWIPG method across various pruning ratios, and notably, it consistently outperformed the other methods. Using the Faster R-CNN object [31] detector with ConvNeXt-T [25] and SwinTransformer-T [23] as the backbones, we benchmark different pruning methods for crop disease detection in Table 1.

In terms of the ConvNeXt-T backbone, among the evaluated methods, the BlockWIPG method achieved the highest Top-1 mAP@50 42.67%, outperforming ViT-Slim [5] (41.49%) and Isomorphic [13] (41.79%), respectively. Regarding computational efficiency, the BlockWIPG also achieved the lowest GFLOPs of 149.20 after pruning, indicating a more efficient model. The reduction in FLOPs from 233.12G to 149.20G represents a significant gain in computation efficiency, compared to Slim [5] (153.11G) and Isomorphic [13] (151.60G), respectively.

Meanwhile, in terms of the SwinTransformer-T backbone, similar to ConvNeXt-T, after pruning, BlockWIPG achieved the highest Top-1 mAP@50 of 42.57%, outperforming ViT-Slim (41.47%) and Isomorphic (41.89%). Concerning computational cost. BlockWIPG furthermore achieved the lowest FLOPs at 168.22G, compared to ViT-Slim (169.93G) and Isomorphic (169.23G). BlockWIPG thus demonstrated a superior balance between accuracy and efficiency.

5. Conclusion

We proposed a new model pruning method BlockWIPG, for detecting crop diseases in distinct complex object detection network architectures. It realized pruning by applying image and weight perturbations. The proposed method significantly improved the pruning performance. Thanks to it's intermediate gradient-based analysis, in contrast, to existing methods which focusing on magnitude value or final loss variation, which is unsuitable for deeper and tiny models in complex tasks such as detection. As future work, we will analyze the effect of FLOPs, power, and Inference on different devices, especially on mobile devices.

Acknowledgment

This work was partly supported by JSPS KAKENHI 25K00161.

References

[1] Ahmad, A., Saraswat, D. and El Gamal, A.: A survey on using deep learning techniques for plant disease diagnosis and recommendations for development of appropriate tools, *Smart Agricultural Technology*, Vol. 3, No. 100083, pp. 1–13 (2023).

[2] Ashkboos, S., Croci, M. L., do Nascimento, M. G., Hoefler, T. and Hensman, J.: SliceGPT: Compress Large Language Models by Delet-

ing Rows and Columns, *Proc. 12th International Conference on Learning Representations* (2024).

[3] Babirye, C., Nakatumba-Nabende, J., Namanya, G., Mutebi, C., Ebellu, M., Murungi, J., Tobius, S., Ssemwogerere, J., Nakayima, A., Nabagereka, D., Asasira, J. and Kanyesigye, R.: Makerere University Maize Image Dataset (2022). https://doi.org/10.7910/DVN/LPGHKK (Visited: 11-June-2025).

[4] Blessing, S., Gamundani, A., Iyawa, G., Matsa, L. S., Koruhama, A. K., Pasipanodya, J. T., Kasaona, B., Kadhikwa, A. and Amadhilla, H. N.: Namibia University of Science and Technology Maize Dataset (2022). https://doi.org/10.7910/DVN/6R78HR (Visited: 11-June-2025).

[5] Chavan, A., Shen, Z., Liu, Z., Liu, Z., Cheng, K.-T. and Xing, E. P.: Vision Transformer slimming: Multi-dimension searching in continuous optimization space, *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4931–4941 (2022).

[6] Darlington, A., Issah, S., Christabel, A., Michael, A., Emmanuel, A., Frank, E.-C., Jerry, B., Seloame, Nyaku, T., Agyeiwaa, M., Hodasi, B., Darko, K. G., Owusu, A. Y. and Asare, C. D.: The KaraAgro AI Maize Dataset (2022). https://doi.org/10.7910/DVN/CXUMDS (Visited: 11-June-2025).

[7] Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y. and Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation, *Advances in Neural Information Processing Systems*, Vol. 27, p. 1269–1277 (2014).

[8] Fang, G., Ma, X. and Wang, X.: Structural pruning for diffusion models, *Advances in Neural Information Processing Systems*, Vol. 36, pp. 16716–16728 (2023).

[9] Food and Agriculture Organization of the United Nations: New standards to curb the global spread of plant pests and diseases (2021). https://www.fao.org/news/story/en/item/1187738/icode/ (Visited: 11-June-2025).

[10] Food and Agriculture Organization of the United Nations: Agricultural production statistics. 2000–2020. FAOSTAT Analytical Brief Series No. 41 (2022).

[11] Frankle, J. and Carbin, M.: The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks, *Proc. 7th International Conference on Learning Representations* (2019).

[12] Frantar, E. and Alistarh, D.: SparseGPT: Massive language models can be accurately pruned in one-shot, *Proc. 15th International Conference on Machine Learning*, pp. 10323–10337 (2023).

[13] Gongfan, F., Xinyin, M., Bi, M. M. and Xinchao, W.: Isomorphic pruning for vision models, *Proc. 18th European Conference on Computer Vision*, Vol. 62, pp. 232–250 (2024).

[14] Han, S., Pool, J., Tran, J. and Dally, W. J.: Learning both weights and connections for efficient neural networks, *Advances in Neural Information Processing Systems*, Vol. 28, p. 1135–1143 (2015).

[15] Han, S., Pool, J., Tran, J. and Dally, W. J.: Learning both weights and connections for efficient neural networks, *Proceedings of the 29th International Conference on Neural Information Processing Systems*, NIPS'15, Vol. 1, p. 1135–1143 (2015).

[16] He, K., Zhang, X., Ren, S. and Sun, J.: Deep residual learning for image recognition, *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778 (2016).

[17] He, Y., Zhang, X. and Sun, J.: Channel pruning for accelerating very deep neural networks, *Proc. 16th IEEE International Conference on Computer Vision*, pp. 1398–1406 (2017).

[18] Hu, H., Peng, R., Tai, Y. and Tang, C.: Network trimming: A data-driven neuron pruning approach towards efficient deep architectures, *Computing Research Repository arXiv Preprints*, Vol. arXiv:1608.08710 (2016).

[19] Kingma, D. P. and Ba, J.: Adam: A method for stochastic optimization, *Computing Research Repository arXiv Preprints*, Vol. arXiv:1412.6980 (2017).

[20] Krizhevsky, A., Sutskever, I. and Hinton, G. E.: ImageNet classification with deep convolutional neural networks, *Advances in Neural Information Processing Systems*, Vol. 25, pp. 1106–1114 (2012).

[21] Li, H., Kadav, A., Durdanovic, I., Samet, H. and Graf, H. P.: Pruning filters for efficient ConvNets, *Computing Research Repository arXiv Preprints*, Vol. arXiv:1608.08710 (2017).

[22] Liu, L., Zhang, S., Kuang, Z., Zhou, A., Xue, J.-H., Wang, X., Chen, Y., Yang, W., Liao, Q. and Zhang, W.: Group Fisher pruning for practical network compression, *Proc. 38th International Conference on Machine Learning*, pp. 7021–7032 (2021).

[23] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. and Guo, B.: Swin Transformer: Hierarchical vision transformer using shifted windows, *Proc. 18th IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022 (2021).

[24] Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S. and Zhang, C.: Learning efficient convolutional networks through network slimming, *Proc. 16th IEEE International Conference on Computer Vision*, pp. 2755–2763 (2017).

[25] Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T. and Xie, S.: A ConvNet for the 2020s, *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986 (2022).

[26] Ma, X., Niu, W., Zhang, T., Liu, S., Lin, S., Li, H., Wen, W., Chen, X., Tang, J., Ma, K., Ren, B. and Wang, Y.: An Image enhancing pattern-based sparsity for real-time inference on mobile devices, *Proc. 16th European Conference on Computer Vision*, pp. 629–645 (2020).

[27] Ma, X., Fang, G. and Wang, X.: LLM-Pruner: On the structural Pruning of Large Language Models, *Advances in Neural Information Processing Systems*, Vol. 36, pp. 21702–21720 (2023).

[28] Men, X., Xu, M., Zhang, Q., Wang, B., Lin, H., Lu, Y., Han, X. and Chen, W.: ShortGPT: Layers in large language models are more redundant Than You Expect, *Computing Research Repository arXiv Preprints*, Vol. arXiv:2403.03853 (2024).

[29] Meng, F., Cheng, H., Li, K., Luo, H., Guo, X., Lu, G. and Sun, X.: Pruning filter in filter, *Advances in Neural Information Processing Systems*, Vol. 33, pp. 17629–17640 (2020).

[30] Padeiro, C. V., Chen, T. W., Komamizu, T. and Ide, I.: SPQ: Similarity-Preserving posttraining Quantization for portable crop disease detection, *IEEE Transactions on AgriFood Electronics*, p. 13 (2025).

[31] Ren, S., He, K., Girshick, R. and Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks, *Advances in Neural Information Processing Systems*, Vol. 28, pp. 91–99 (2015).

[32] Tanaka, H., Kunin, D., Yamins, D. L. and Ganguli, S.: Pruning neural networks without any data by iteratively conserving synaptic flow, *Advances in Neural Information Processing Systems*, Vol. 33, pp. 6377–6389 (2020).

[33] Tang, Y., Wang, Y., Xu, Y., Deng, Y., Xu, C., Tao, D. and Xu, C.: Manifold regularized dynamic network pruning, *Proc. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5018–5028 (2021).

[34] Theis, L., Korshunova, I., Tejani, A. and Huszár, F.: Faster gaze prediction with dense networks and Fisher pruning, *Computing Research Repository arXiv Preprints*, Vol. arXiv:1801.05787 (2018).

[35] Wang, H. and Fu, Y.: Trainability preserving neural pruning, *Proc. 11th International Conference on Learning Representations* (2023).

[36] Wiesner-Hanks, T. and Brahimi, M.: Image set for deep learning: Field images of maize annotated with disease Symptoms (2019). https://osf.io/p67rz/ (Visited: 11-June-2025).

[37] Yang, H., Yin, H., Shen, M., Molchanov, P., Li, H. and Kautz, J.: Global Vision Transformer Pruning With Hessian-Aware Saliency, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18547–18557 (2023).

[38] Yang, N., Jang, Y., Lee, H., Jeong, S. and Jung, K.: Task-specific compression for multi-task language models using attribution-based pruning, *Proc. 17th Conference of the European Chapter*, pp. 594–604 (2023).

[39] You, Z., Yan, K., Ye, J., Ma, M. and Wang, P.: Gate Decorator: Global filter pruning method for accelerating deep Convolutional Neural Networks, *Computing Research Repository arXiv Preprints*, Vol. arXiv:1909.08174 (2019).