

Edge AI System for Real-Time and Explainable Forest Fire Detection Using Compressed Deep Learning Models

Sidi Ahmed Mahmoudi^a, Maxime Gloesener, Mohamed Benkedadra^b and Jean-Sébastien Lerat

ILIA Lab, Faculty of Engineering, University of Mons, Mons, Belgium

{sidi.mahmoudi, maxime.gloesener, mohamed.benkedadra, jean-sébastien.lerat}@umons.ac.be

Keywords: Forest Fire Detection, Deep Learning, CNN Networks, Vision Transformers, Edge AI, XAI.

Abstract: Forests are vital natural resources but are highly vulnerable to disasters, both natural (e.g., lightning strikes) and human induced. Early and automated detection of forest fire and smoke is critical for mitigating damages. The main challenge of this kind of application is to provide accurate, explainable, real-time and lightweight solutions that can be easily deployable by and for users like firefighters. This paper presents an embedded and explainable artificial intelligence “Edge AI” system, for real-time forest fire, and smoke detection, using compressed Deep Learning (DL) models. Our model compression approach allowed to provide lightweight models for Edge AI deployment. Experimental evaluation on a preprocessed dataset composed of 1500 images demonstrated a test accuracy of 98% with a lightweight model running in real-time on a Jetson Xavier Edge AI resource. The compression methods preserved the same accuracy, while accelerating computation ($3\times$ to $18\times$ speedup), reducing memory consumption ($3.8\times$ to $10.6\times$), and reducing energy consumption ($3.5\times$ to $6.3\times$).


1 INTRODUCTION


Forests are crucial in combating climate change, absorbing 2.6 billion tons of carbon annually. (Madeira, 2023). However, in recent years, the increasing frequency and intensity of forest fires have posed a significant threat to these vital ecosystems, as well as to human lives and property. This underscores the urgent need for advanced technologies capable of detecting and mitigating these disasters in real-time. To address this critical challenge, this paper introduces a novel Embedded Edge AI system for real-time forest fire and smoke detection, enabling continuous monitoring and early warning. Leveraging cutting-edge techniques from the field of deep learning, we combine the strengths of Convolutional Neural Networks (CNNs) and Vision Transformers to create a highly accurate, efficient and explainable model. Despite its powerful capabilities, the model is optimized and compressed to operate within the resource constraints of edge devices such as the Jetson Nano, Jetson Xavier, Jetson Orin, and Jetson Orin Nano¹ ensuring its applicability in remote and challenging en-

vironments. The integration of artificial intelligence (AI) with edge computing has unlocked new possibilities for environmental monitoring and disaster prevention. Our system allows for the detection, localization and identification of forest fires and forest smoke. As a consequence, it can trigger accurate and timely alerts, enabling rapid response efforts that can save lives and reduce the ecological and economic impact of these disasters. By harnessing the power of Embedded AI at the edge, we present a scalable, cost-effective, and robust solution for real-time forest fire detection.

Despite the promising results of Deep Neural Networks (DNNs) for fire detection in the literature, no existing solution achieves an optimal balance between accuracy, explainability, computation time, and memory/energy efficiency. Our contribution addresses this gap through three innovations :

- Innovative development approach: best DNN models in terms of accuracy and explainability ;
- Innovative compression approach: compress DNN models to significantly accelerate inference while reducing memory and energy consumption. Our innovation lies in delivering the best trade-off across all these metrics within a single model.
- Innovative library “pytorch_bench”: comprehensive

^a  <https://orcid.org/0000-0002-1530-9524>

^b  <https://orcid.org/0000-0002-2590-8344>

¹Jetson Systems: <https://www.nvidia.com/fr-fr/autonomous-machines/embedded-systems/>

sive benchmarking library for deep neural networks, enabling precise evaluation across multiple metrics, including accuracy, loss, computation time, memory usage, and energy consumption.

2 RELATED WORK

2.1 CV and AI for Fire Detection

Traditional fire and smoke detection relies on low-cost sensors, such as smoke, fire, and heat detectors. While effective at identifying the presence of fire, these sensors often lack capabilities for localization, incident analysis, and reliable data on the fire's origin and progression (Hamzah et al., 2022). Vision-based systems are proposed as a superior alternative, offering more precise and detailed information about the type, source, and severity of fire and smoke. This enhanced data supports improved resource allocation and more effective incident response.

Traditional fire detection and classification methods heavily relied on handcrafted features such as color, texture, and shape. These approaches required significant domain knowledge and often struggled to generalize across different fire scenarios. Recently, Deep Learning allowed for the automatic extraction of high-level features, which are difficult to obtain with traditional techniques.

Convolutional Neural Networks (CNNs) have demonstrated remarkable performance in fire classification tasks. (Zhao et al., 2018) proposed a 15-layer CNN architecture named 'Fire_Net,' inspired by AlexNet, which works as both a feature extractor and a classifier. Transfer learning has also emerged as a valuable technique in fire detection and classification, with many methods leveraging this approach. (Sousa et al., 2020) proposed a method using transfer learning by fine-tuning the weights of the Inceptionv3 model for fire classification tasks. Similarly, (Govil et al., 2020) proposed the 'ForestResNet' method, which is based on the ResNet50 model. (Park et al., 2021) explored various pre-trained models, including VGG16, ResNet50, and DenseNet121, for fire detection purposes. (Shamsoshoara et al., 2021) employed the Xception model for fire detection tasks.

Recent advancements have focused on model efficiency, leading to the development of architectures tailored for resource-constrained environments. For example, (Khan and Khan, 2022) and (Wu et al., 2020) utilized MobileNetV2 for real-time fire detection, achieving a balance between accuracy and computational efficiency.

2.2 XAI for Machine & Deep Learning

Deep learning models are seen as black boxes where the provided results cannot produce partial or complete explanations (Stassin et al., 2024) (Mahmoudi et al., 2022) (Englebert et al., 2023). In this context, post-hoc explainability methods can be applied to input images in order to describe the model reasoning. The post-hoc methods are represented by three categories: 1) Gradient-based methods (Simonyan et al., 2014) that track the neuron values propagation from the network output to the input, allowing to identify the importance of each input feature; 2) Perturbation-based methods (Zeiler and Fergus, 2014) (Petsiuk et al., 2018) that disturb the inputs and analyze the performance drop of evaluation metrics. A high drop of accuracy metrics means that the disturbed inputs are important and vice versa; 3) CAM methods that are mainly used to explain CNN networks by the use of convolutional layers activations in order to produce saliency maps showing the importance of each pixel in the model decision (Wang et al., 2020). Otherwise, although their high accuracies, vision transformers architectures (Dosovitskiy et al., 2020) are also hindered by their lack of explainability. A basic approach for explaining ViTs consists of the usage of attention weights as explanation. However, this approach remains inadequate for explaining transformer results since it considers the query and key elements of the self-attention, but not the value (Pruthi et al., 2020).

2.3 Optimization and Compression

Recently, a significant improvement of embedded AI Hardware has been noted where GPUs can be integrated in small Edge AI devices. This makes from these devices a promising solution for the deployment of machine and deep learning models. Although the provided power, several limitations (computation power, RAM memory, storage size, etc.) may be encountered when deploying complex neural architectures for processing massive data and more particularly HD images. Thus, we can propose restructuring and compression of deep neural architectures since the latter are generally over-parametrized where several weight and bias values can be retired without negative influence on models accuracy. We can cite 3 compression techniques: pruning (Blalock et al., 2020), quantization (Nagel et al., 2021) and knowledge distillation (Gou et al., 2021).

1. **Pruning:** Identify and remove unnecessary neurons and connections, retaining only the most relevant components within a deep neural network. While pruning reduces model size, to produce less

parametrized (sparse) architectures, the resulting sparse architecture often remain computationally inefficient as a consequence of the unchanged operation counts. To address this, the block pruning method (Lagunas et al., 2021) produces dense, pruned architectures that achieve reduced memory consumption and computation time.

2. **Quantization:** An approximation process is applied to neural networks to represent weights and/or activations with lower precision values. Hence, minimizing memory usage and computation time without compromising on model accuracy (Nagel et al., 2021).
3. **Knowledge Distillation:** Consists of training a small “Student Network” that can learn during the training from a large “Teacher Network”. The objective is to have a small, simplified neural network that can make decisions similar to those of the larger and more complex one (Gou et al., 2021).

3 MODEL DEVELOPMENT APPROACH

3.1 DL Models Selection

In this step, transfer learning is applied to pre-trained classification models initialized with ImageNet weights, to benefit from previously acquired learning weights for solving our classification problem, consisting of fire detection using three classes (fire, smoke, no fire). We utilized deep neural architectures suited for image classification like convolutional Neural Networks (CNN), Vision Transformers (Vit) and even models that combine the usage of convolutional layers and attention mechanism like ConvNeXt (Liu et al., 2022). Vision transformers belong to the family of transformers that are represented by encoder–decoder architectures using multihead self-attention. The Vision transformer (ViT) architectures are mainly proposed for images classification tasks (recently for object detection, and image segmentation as well), where the embeddings vector is directly used for classification. With the encoder part, the ViT represents images as sequences and classes where each input image is divided to a sequence of patches. Every patch is flattened to a vector containing all pixel channels. Then, a linear projection is applied to provide the desired input dimension. The decoder part exploits the outputs of the encoder and uses a Multilayer Perceptron architecture for image classification (Dosovitskiy et al., 2020). This study

selects the most accurate and generalizable architectures based on their performance with ImageNet pre-trained weights ².

- **CNN Architectures:** VGG family (VGG11, VGG13, VGG16), ResNet family (ResNet18, ResNet34, Resnet50, ResNet101, ResNet152) EfficientNet family (EfficientNet_V2_S, EfficientNet_V2_M, EfficientNet_V2_L)
- **Vision Transformers:** Vit_B_16, Vit_B_32
- **Combined Architectures:** ConvNeXt_Small, ConvNeXt_Base, ConvNeXt_Large.

3.2 Data Preprocessing

To improve the quality and efficacy of our dataset, we implemented a preprocessing pipeline using the following steps:

1. **Data Collection:** we have collected several public videos that represent different situations related to fire or/and smoke in forest. From each video, we have extracted 3 frames per second in order to generate a large variety of images representing normal situations, fire and smoke situations. At the end of this step, we got 4000 images (fire: 1500 images, smoke: 1500 images, no_fire: 1000 images)
2. **Removal of Duplicated Images:** this step consists on removing similar images after calculating the difference between images. Once the similar images eliminated, we applied data augmentation in order to increase the size of our dataset from the selected unique images.
3. **Extreme Luminance Filtering:** in this phase, we removed images with extreme luminance values, as they provide limited useful information for our specific use case.
4. **Blur Detection and Removal:** we used a blur detection algorithm to identify and remove severely blurred images. The top 3% images with the highest blur metrics were eliminated from the dataset.
5. **Label Verification:** an inspection process was conducted to identify and remove images with incorrect labels, ensuring the integrity of our ground truth data. If an image is presenting both fire and smoke, we propose “fire” label since the priority is prevent fire.
6. **Near-Duplicate Detection:** to address the issue of near-duplicate images, particularly those originating from video frame extraction, we employed

²Models and pre-trained weights. <https://pytorch.org/vision/main/models.html>

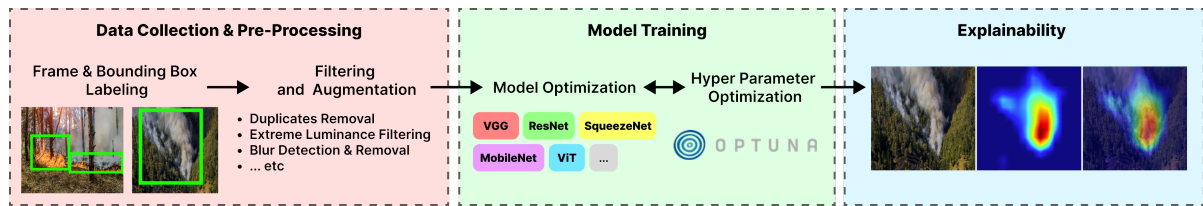


Figure 1: Model Development Pipeline from Data Collection and Model Selection to Explainability.

a similarity threshold approach. Images with a Structural Similarity Index (SSIM) score above 92% were grouped together and from each group of near-duplicates, only one representative image was retained, while the others were discarded.

Fig. 2 illustrates some image examples that have been removed due to their excessive darkness, brightness or blurring. This preprocessing pipeline sig-



Figure 2: A. dark images removal ; b. bright images removal ; c. high blurred images removal.

nificantly improved the quality and diversity of our dataset. The final size of our dataset is represented by 1500 images divided in a balanced way between the 3 classes (500 images for each class). The last step consists of splitting the data into three (03) subsets: train (70%), validation (15%) and test (15%). This allowed to ensure a fair validation during and after the training process.

3.3 Models Training and Hyperparameters Optimization

This step focuses on the optimization of hyperparameters for each model. For this aim, we used Optuna framework (Akiba et al., 2019) that allowed to explore in an efficient way the hyperparameters space using grid search algorithms such as as Bayesian optimization within Tree-structured Parzen Estimator (TPE). In our case, the objective function is represented by the "loss_val" and the researched hyperparameters are represented by the values of: optimizer (gradient descent method, batch size, learning rate, etc.), regularization (L1, L2, batch-normalization, dropout, etc.) and data augmentation. Notice that we used an early stopping process in order to stop the training at the best moment and avoid overfitting.

3.4 Model Explainability

We have applied an explainability method among those proposed in Section 2.2. For this aim, we selected Grad-CAM (Selvaraju et al., 2020) technique due to its high efficiency and also its compatibility with CNN, ViT and combined (CNN-ViT) architectures. This explainability, applied within the inference phase (post-hoc), allowed to identify the responsible pixels of each model decision and then detect eventual bias in the model or/and in the training datasets (Fig. 1). As shown in this figure, we can note that the model focuses on the good fire are for detection which means that our model is not biased.

3.5 Evaluation Metrics

Before analyzing the computational, memory, and energy performance of our models, we first evaluate their accuracy using the following metrics: accuracy, loss and confusion matrix. Note that the accuracy and loss metrics are calculated for the training, validation, and test datasets.

4 MODEL COMPRESSION APPROACH

In order to deploy these models in production, we use Edge AI resources. For that, we propose and study model compression for producing light weight accurate models.

4.1 Edge AI System

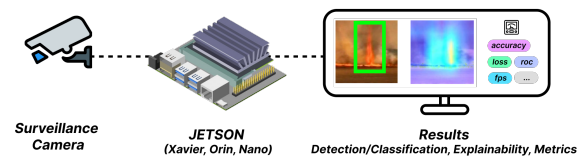


Figure 3: Edge Surveillance System.

We propose to use a system composed of (Fig.3):

- **HD Camera:** that can be connected with USB port and collect 2 images in real-time ;
- **Jetson Nvidia Device:** including an embedded GPU allowing to apply model inference on video frames. Our solution is compatible with several Jetson devices proposing variable computational capacities depending on the device price ³.
- **Portable Monitor:** to visualize the processed video showing the detection results and notifications all evaluation metrics.

4.2 Model Compression

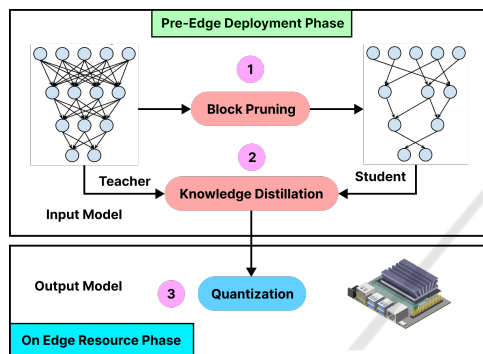


Figure 4: Two Steps Edge Optimized Model Deployment Pipeline: (1) None Edge Resource based Pruning + Distillation (2) On Edge Resource Quantization and Deployment

In order to propose a compression method that allows to reduce both computation, memory and energy consumption, we propose to apply a hybrid method combining the techniques of pruning, quantization and knowledge distillation. Therefore, our compression algorithm consists of three steps: block pruning, knowledge distillation and int8 quantization.

- **Block Pruning:** this method reduces model complexity by eliminating less important blocks from the neural network, resulting dense and compressed architectures. This approach is more advantageous than the unstructured method, which removes less important weights by converting them to zero values. While the unstructured method reduces model size, it does not decrease computational complexity and leads to sparse neural networks. Notably, block pruning is applied during the training process to ensure that the highest possible accuracy is achieved.
- **Knowledge Distillation:** used with the teacher model (base model) and the student model (pruned model) to help the pruned model recover

³Jetson Nvidia Comparison:
<https://developer.nvidia.com/embedded/jetson-modules>

its accuracy. The base model guides the learning process, enabling the pruned model to regain performance despite its reduced complexity.

- **Int8 Quantization:** allows to store both weights and activation (static quantization) values using a lower precision format (int8) instead of the original format (float 32). We propose to apply the quantization after training (on Edge device) and not during the training (quantization aware training) in order to identify the format according to the target Edge resource. Notice that static quantization provided better results than dynamic one which justify its selection for this work. The quantization format of int8 was selected empirically after experimentation within several configurations.

4.3 Evaluation Metrics

To ensure a fair and comprehensive evaluation of our models, we analyze key metrics that account for accuracy, computation time, memory and storage usage, energy consumption, and model complexity. For this purpose, we developed a library called "PyTorch_Bench" to compute the following metrics: Number of model parameters (weights & biases), Model size (in MB), Number of MACs (complexity evaluation), Maximum memory usage (in MB, Current memory usage (in MB), FPS (frames processed per second), Estimated energy emissions (in gCO₂), Total energy consumption (in mWh).

5 EXPERIMENTAL RESULTS

In this section, we evaluate and compare CNN, VIT and combined architecture within metrics related to accuracy, loss, explainability, computation performance, memory size and energy consumption. Our experimental results are presented in four subsections: experimental setup and database, models evaluation before compression, models evaluation after compression, discussion.

5.1 Experimental Setup and Database

Experimentations were conducted using the following hardware:

- CPU Processor of training server: Intel I9-13900K, 5.8 GHZ, 24 Cores, 64G RAM DDR5
- GPU Processor of training server: RTX 4090 24GB, 16000 CUDA cores

- Edge AI device: Jetson Xavier, 512 CUDA cores, 16 GB of memory

In the following parts, the deep learning models are trained and evaluated using the framework Pytorch, known by its computation efficiency (Lerat et al., 2023) compared to other deep learning frameworks. As shown in Section 3.2, the models are trained within a 3 classes balanced dataset composed of 1500 images where 70% are used for training, 15% for validation and 15% for test.

5.2 Evaluation Before Compression

This subsection is related to present the results of the model development approach outlined in Section 3 where several steps (DL models selection, data preprocessing, models training and hyperparameters optimization, models explainability and evaluation metrics) have been followed to provide the best results within CNN architectures, Vit and ConvNeXt architectures. Notice that the compression is not yet applied in this subsection. Table 1 illustrates the obtained results within the different deep neural architectures, where different metrics are calculated.

We note a high test accuracy (from 94% to 98%) achieved by these models, thanks to our data preprocessing and hyperparameters optimization process (the latter allowed us to improve models precision with around 10% of test accuracy). We also note that CNN architectures provided the best results (lightly) compared to vision transformers or even architectures that combine convolutions and attention mechanisms such as ConvNeXt, which are more complex and consuming in time. This can be interpreted by the nature of our images and classes where fire and smoke areas are generally present within connected zones that do not require necessarily the usage of attention mechanisms adapted for taking into account the correlation between different zones (tokens). Notice also that best results are provided by more complex architectures such as EfficientNet_V2_L that uses more parameters and hence more computation time. This motivates our proposition of developing compressed deep neural architectures.

5.3 Evaluation after Compression

Based on the previous subsections, we selected the most accurate model for each DNN category. We then applied our compression approach, which combines block pruning, knowledge distillation, and quantization (Section 4.2), and evaluated the models on the Jetson Xavier to simulate real-world scenarios. As shown in Table 2, the combination of pruning and

knowledge distillation allowed to accelerate and reduce the complexity of our different models (CNN, Vit and ConvNeXt) with the maintain of a high model accuracy. Notice that pruning has mainly contributed on reducing the model computational complexity while knowledge distillation helped recover the accuracy lost as a result of the pruning step. Thus, the combination between the processes of pruning and knowledge distillation provided excellent balance for all metrics. On the other hand, quantization applied on the Edge AI resource allowed to improve significantly the computation performance. As a result, our compression approach enabled the following:

- Reduce the complexity of our models thanks to the block pruning: with a ratio of 50%
- Accelerate the computation: with a speedup ranging from $3\times$ to $18\times$
- Reduce the memory consumption: with a factor ranging from $3.8\times$ to $10.6\times$
- Reduce the model sizes: with a factor ranging from $1.8\times$ to $7.6\times$
- Reduce the energy consumption: with a factor ranging $3.5\times$ to $6.3\times$

Notice that we have also analyzed the explainability results where the compression has not affected the calculation of attention heatmap that are always focusing on the detection classes zones (Fig. 1). We note that the pruning was applied with a rate of 50% which means that models complexity is reduced to the half. We also note that the proposed Edge AI system can run during several hours in constrained environments since the deployed models are highly compressed and do not cause overrun problems.

5.4 Discussion

Experimental results demonstrated the interest of using CNN or/and Vision Transformer based neural architectures for fire detection, where the accuracy can reach 98% with a high varied test dataset. We also note that our steps of data preprocessing and hyperparameters optimization allowed to increase the test accuracy with a range of 10%. The application of regularization techniques (L1, L2, Dropout, Batch-normalization and early stopping) allowed to avoid the problem of overfitting where no variance is noted between train, validation and test accuracy values. On the other hand, experimentations allowed to validate our compression approach since we were able to reduce significantly the models consumption in terms of memory, storage, energy and computation with the maintain of a high accuracy with a good explanation

Table 1: Performance comparison of CNN and Transformer-based models for fire detection: before compression.

DNN Model Category	DNN Model Family	DNN Model Name	# Param (M)	Val Acc	Val Loss	Test Acc	Test Loss
CNN Networks	VGG	VGG11	132.9	96.44	0.186	95.11	0.092
		VGG13	133.0	95.56	0.176	96.00	0.150
		VGG16	138.4	96.89	0.200	96.44	0.160
		VGG19	143.7	97.33	0.088	97.33	0.095
	ResNet	ResNet18	11.7	96.00	0.100	97.78	0.060
		ResNet34	21.8	96.00	0.110	96.44	0.083
		ResNet50	25.6	96.89	0.090	97.33	0.093
		ResNet101	44.5	96.89	0.072	95.56	0.095
		ResNet152	60.2	97.33	0.088	95.56	0.100
	EfficientNet	EfficientNet_V2_S	21.5	96.00	0.074	96.00	0.081
		EfficientNet_V2_M	54.1	97.78	0.072	96.00	0.110
EfficientNet_V2_L		118.5	96.89	0.059	98.22	0.071	
Vision Transformers	ViT	Vit_B_16	86.6	96.89	0.110	97.33	0.057
		Vit_B_32	88.2	95.55	0.150	96.44	0.076
Combined	Conv + Attention Mechanism	ConvNeXt-Small	50.2	96.44	0.074	96.00	0.131
		ConvNeXt-Base	88.6	96.88	0.104	94.66	0.122
		ConvNeXt-Large	197.8	97.78	0.089	96.00	0.092

Table 2: Performance comparison of various DNN Models (CNNs and Vision Transformers) under different optimization scenarios (Base, Pruned & Distilled, and Pruned & Distilled & Quantized).

Model	Base Model				Pruned & Distilled Model				Pruned & Distilled & Quantized Model			
	FPS	Mem (MB)	Energy (mWh)	Test acc	FPS	Mem (MB)	Energy (mWh)	Test acc	FPS	Mem (MB)	Energy (mWh)	Test acc
VGG19	18.2	2128	0.082	97.33	51.2	835	0.063	98.66	326.6	547	0.013	98.66
ResNet18	196.9	195	0.016	97.78	256.9	94	0.014	97.78	606.5	80	0.004	96.88
EfficientNet_V2_L	18.3	737	0.148	98.22	20.2	601	0.142	96.44	98.1	255	0.033	96.44
Vit_B_16	21.9	611	0.099	97.33	34.6	479	0.089	97.33	102.4	121	0.028	96.88
ConvNeXt-Large	9.2	834	0.322	96.00	17.1	421	0.177	96.66	78.9	223	0.055	96.66

that confirms the absence of model and data biases. It is also worth noting that, in some cases, the compressed model achieves better accuracy than the original model. This is due to the fact that pruning acts as a form of regularization, enhancing the model's performance. We also note that our compression approach has provided these improvements across the different DNN architectures (CNN, Vision transformers and combined architectures) which is very promising for a widespread use of our solution.

6 CONCLUSION

In this paper, we proposed an innovative approach providing accurate and explainable deep neural architectures for forest fire detection. The proposed approach is based on an efficient preprocessing step followed by the selection of CNN or/and transformer

based architectures with an optimization of their hyperparameters. Once a high accuracy obtained, we focused on the compression of the models with a hybrid approach combining the processes of block pruning, knowledge distillation and quantization. As result, the generated models were improved in terms of accuracy and performance (memory, storage, size, complexity) with the maintain of a good explainability of results. These improvements enabled easy deployment of our models on Edge AI devices, allowing real-time processing. As future work, we plan to apply our compression approach on other DNN architectures such as those used for image detection or segmentation and even those using Large vision models (Shi et al., 2025). We plan also to accelerate the training process using parallel and distributed learning (Lerat and Mahmoudi, 2024).

REFERENCES

- Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., and Gutttag, J. (2020). What is the state of neural network pruning? In Dhillion, I., Papailiopoulos, D., and Sze, V., editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 129–146.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR 2021 · The Ninth International Conference on Learning Representations*.
- Englebort, A., Stassin, S., Nanfack, G., Mahmoudi, S. A., Siebert, X., Cornu, O., and De Vleeschouwer, C. (2023). Explaining through transformer input sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 806–815.
- Gou, J., Yu, B., Maybank, S. J., and Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.
- Govil, K., Welch, M. L., Ball, J. T., and Pennypacker, C. R. (2020). Preliminary results from a wildfire detection system using deep learning on remote camera images. *Remote Sensing*, 12(1).
- Hamzah, S. A., Dalimin, M. N., Som, M. M., Zainal, M. S., Khairun, Ramli, N., Utomo, W. M., and Yusoff, N. A. (2022). High accuracy sensor nodes for a peat swamp forest fire detection using esp32 camera.
- Khan, S. and Khan, A. (2022). Ffirenet: Deep learning based forest fire classification and detection in smart cities. *Symmetry*, 14(10).
- Lagunas, F., Charlaix, E., Sanh, V., and Rush, A. M. (2021). Block pruning for faster transformers. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629.
- Lerat, J.-S. and Mahmoudi, S. A. (2024). Scalable deep learning for industry 4.0: Speedup with distributed deep learning and environmental sustainability considerations. In *Artificial Intelligence and High Performance Computing in the Cloud*, pages 182–204, Cham. Springer Nature Switzerland.
- Lerat, J.-S., Mahmoudi, S. A., and Mahmoudi, S. (2023). Single node deep learning frameworks: Comparative study and cpu/gpu performance analysis. *Concurrency and Computation: Practice and Experience*, 35(14):e6730.
- Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. (2022). A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986.
- Madeira, E. M. (2023). The crucial role of forests in combating climate change. *Trends in Biotechnology and Plant Sciences*.
- Mahmoudi, S. A., Stassin, S., Daho, M. E. H., Lessage, X., and Mahmoudi, S. (2022). *Explainable Deep Learning for Covid-19 Detection Using Chest X-ray and CT-Scan Images*, pages 311–336. Springer International Publishing, Cham.
- Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., Van Baalen, M., and Blankevoort, T. (2021). A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*.
- Park, M., Tran, D. Q., Lee, S., and Park, S. (2021). Multilabel image classification with deep transfer learning for decision support on wildfire response. *Remote Sensing*, 13(19).
- Petsiuk, V., Das, A., and Saenko, K. (2018). Rise: Randomized input sampling for explanation of black-box models. In *BMC*.
- Pruthi, D., Gupta, M., Dhingra, B., Neubig, G., and Lipton, Z. C. (2020). Learning to deceive with attention-based explanations. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.
- Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2020). Grad-cam: visual explanations from deep networks via gradient-based localization. *International journal of computer vision*, 128:336–359.
- Shamsoshoara, A., Afghah, F., Razi, A., Zheng, L., Fulé, P. Z., and Blasch, E. (2021). Aerial imagery pile burn detection using deep learning: The flame dataset. *Computer Networks*, 193:108001.
- Shi, B., Wu, Z., Mao, M., Wang, X., and Darrell, T. (2025). When do we not need larger vision models? In *European Conference on Computer Vision*, pages 444–462. Springer.
- Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. *ICLR 2014 · International Conference on Learning Representations. Workshop Poster*.
- Sousa, M. J., Moutinho, A., and Almeida, M. (2020). Wildfire detection using transfer learning on augmented datasets. *Expert Systems with Applications*, 142:112975.
- Stassin, S., Corduant, V., Mahmoudi, S. A., and Siebert, X. (2024). Explainability and evaluation of vision transformers: An in-depth experimental study. *Electronics*, 13(1).
- Wang, H., Wang, Z., Du, M., Yang, F., Zhang, Z., Ding, S., Mardziel, P., and Hu, X. (2020). Score-cam: Score-weighted visual explanations for convolutional neural networks. In *CVPR Workshop on TCN*.
- Wu, H., Li, H., Shamsoshoara, A., Razi, A., and Afghah, F. (2020). Transfer learning for wildfire identification in uav imagery. In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6.
- Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.
- Zhao, Y., Ma, J., Li, X., and Zhang, J. (2018). Saliency detection and deep learning-based wildfire identification in uav imagery. *Sensors*, 18(3).