Analyzing Deep-Learning Methods for Power Line Component Detection in Unmanned Aircraft System Imagery with Few Data

Guillaume Fourret[1,2], Marc Chaumont[1,3], Christophe Fiorio[1], Gérard Subsol[1], Samuel Brau[2]

***Abstract*— Due to the critical role of power lines in modern infrastructure, numerous automated methods have been developed for their inspection. Among these, Unmanned Aircraft Systems (UAS) have emerged as a valuable tool, offering rapid and precise inspections by capturing high-resolution aerial imagery of power lines. Drones enable access to hard-to-reach areas, reduce safety risks for workers near live wires, and significantly lower the time and cost associated with traditional inspection methods. In particular, deep learning techniques have been widely applied to automate the analysis of key components via the onboard camera. However, these methods typically rely on a first stage of detection based on large, annotated datasets focused on specific components, limiting their adaptability to new or unseen components. This paper investigates the application of two state-of-the-art algorithms of Few-Shot Object Detection (FSOD) for power line component detection: DeFRCN and CD-ViTO, alongside a modified Yolov8 detector of our own in which we integrated the modules of DeFRCN. We evaluate their performance using both public and proprietary datasets, analyzing unexpected outcomes and provide insights into the practical applicability of FSOD in real-world scenarios.**

I. Introduction

Power lines are critical to daily life, and outages can be costly. It is thus essential to inspect regularly the lines and their components to ensure they are functioning properly. Various inspection methods are used, including helicopter-assisted technician teams and robots that climb on the power lines [1]. Recently, UAS-based inspections have gained popularity due to their cost-effectiveness, wide coverage, and ability to capture optimal images of components using onboard cameras [2]. In parallel, automatic analysis of these images has been explored [3]. In particular, recent Deep-Learning based algorithms have been proposed to detect critical components, such as insulators, cables, shackles, or top caps [4] [5] and eventually detect their faults [6].

While these methods demonstrate promising results in detecting and analyzing components of power lines, they depend heavily on large labeled datasets for the specific components of interest. In fact, many components on an electric line are susceptible to defects that, if left unaddressed, could lead to grid failure. Furthermore, these methods are limited to detecting only the components they have been trained on and lack adaptability to new classes of components for which we do not have a lot of examples. Moreover, the appearance of components can vary significantly depending on their environment as shown in Fig. 1. For example, insulators near the sea often have supplementary protective disks to shield them from salt-induced corrosion. Additionally, the configuration of components can differ based on the power rating of the line; high-voltage lines typically use insulator chains with more disks than those found on low-voltage lines.

Few-Shot Object Detection (FSOD) offers a promising approach to deal with this difficulty by allowing a detector to quickly learn to recognize new component classes with only a few examples (typically between 1 and 30 labeled bounding boxes).

In this paper, we evaluate state-of-the-art FSOD algorithms on the domain of power line component detection using aerial imagery captured by UAS. First, we describe two state-of-the-art FSOD methods in Sec. II, along with our adaptation of the Yolov8 detector [8] for FSOD by using components from the DeFRCN [9] framework. Then, in Sec. III we provide a detailed description of the datasets used for this study and evaluate the performance of the presented methods, with a comparison to the standard Yolov8 model as

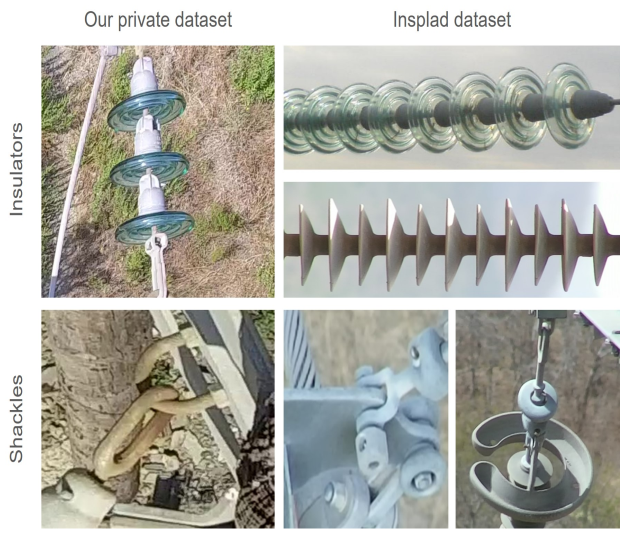

Fig. 1. Diversity of designs for two classes of component (insulators and shackles) based on power line context. The first column shows components from our private dataset, while the second shows those from the Insplad dataset [7], both of which will be detailed later.

This work is partially funded by the ANRT (Association Nationale de la Recherche et de la Technologie) through the CIFRE program, which supports collaboration between PhD students and enterprises.

[1] Guillaume Fourret, Marc Chaumont, Christophe Fiorio, Gérard Subsol are with the ICAR research team, LIRMM (Laboratoire d'informatique, de robotique et de microélectronique de Montpellier), Univ Montpellier, CNRS, Montpellier, France (corresponding author: `guillaume.fourret@lirmm.fr`)

[2] Guillaume Fourret, Samuel Brau are with Drone Geofencing, Nîmes, France

[3] Marc Chaumont is with the University of Nîmes, France

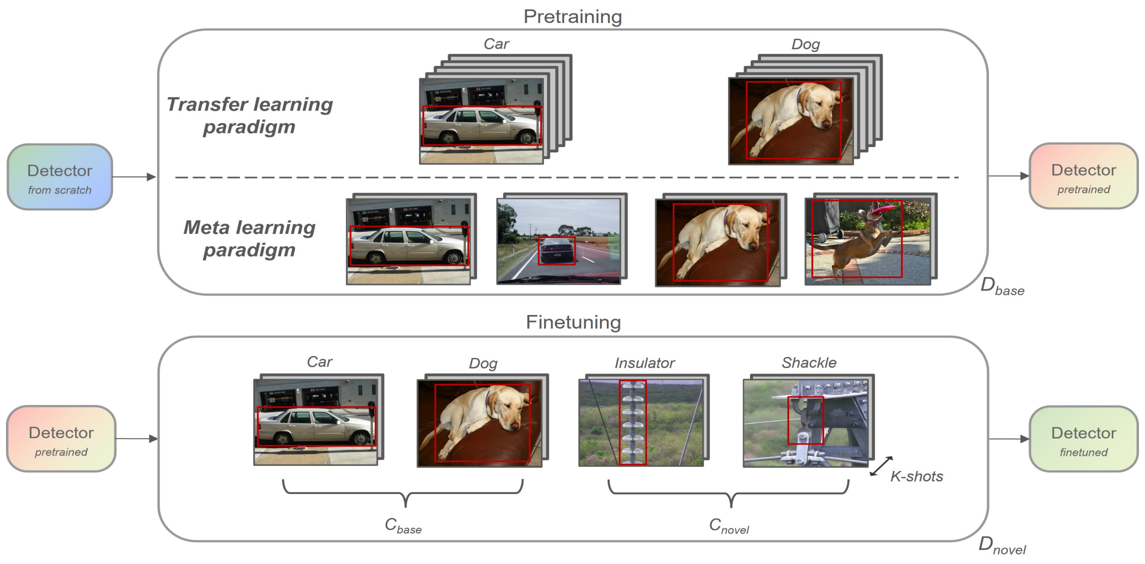

Fig. 2. Illustration of the principle of meta-learning and transfer-learning strategies for object detection.

a baseline. Finally, in Sec. IV, we discuss some unexpected results and derive insights regarding the practical application of FSOD in real-world scenarios of power line inspection with UAS.

II. Few-Shot Object Detection

A. Some notations

We will define the FSOD problem by following the notations given in [10], [11]. Let us assume two image datasets, D_{base} and D_{novel}, which contain objects belonging only to the base classes C_{base} and the novel classes C_{novel} respectively, such as $C_{base} \bigcap C_{novel} = \emptyset$. D_{base} is generally a very large dataset, whereas D_{novel} is a very small one, allowing for few-shot learning. A model can be trained first on D_{base} to detect objects belonging to C_{base}. Then, it can be fine-tuned on a subdataset of $C_{base} \bigcup C_{novel}$ which contains exactly K labeled samples of both the base and the N novel classes to detect (with $N \leq |C_{novel}|$). Notice that in the finetuning phase, K-shot are used for each class (base and novel) to prevent highly umbalanced dataset with all instances from base classes, and rapid overfitting with only the instances of novel classes.

This approach is called N-way K-shot FSOD and results are analyzed in reference benchmarks like MSCOCO [12] by providing performance metrics for both the base and the novel classes with respect to N and K. Note that a shot is an instance of the object (i.e. a bounding box) which means that a single image can contain more than one shot.

B. Two state-of-the-art Methods

There are two main strategies for FSOD (see Fig. 2): meta-learning [13] and transfer-learning [11]. The main difference lies in their pretraining approach where meta-learning simulates multiple FSOD tasks by sampling random K-shot from D_{base}, while transfer-learning focuses on learning high-quality features through standard supervised training.

Recently, transfer learning has appeared to be the more promising paradigm, as illustrated by the results on public benchmark of DeFRCN [9] and CD-ViTO [14]. We selected these two algorithms for our analysis.

1) DeFRCN, a Performant Faster R-CNN Method: During the training of a Faster R-CNN [15], the network enhances localization with class-invariant features and classification with class-specific features, creating contradictory optimization objectives. This tradeoff is manageable with many examples but problematic in few-shot settings. While DCFS [16] employs a form of gradient decoupling by using separate classifiers to address the issue of missing labels in FSOD, DeFRCN introduces the Gradient Decoupling Layer (GDL) to separate localization from classification during gradient backpropagation. This design enables the backbone network to continue learning during fine-tuning, rather than being frozen. Two decoupling layers are used: one between the Region Proposal Network (RPN) and the backbone, and another between the R-CNN head and the backbone. The backpropagation becomes then:

$$\theta_b \leftarrow \theta_b - \gamma(\lambda_{rpn} \frac{\partial \mathcal{L}_{rpn}}{\partial \theta_b} + \lambda_{rcnn} \frac{\partial \mathcal{L}_{rcnn}}{\partial \theta_b}). \quad (1)$$

Where θ_b is the parameters of the backbone, $\partial \mathcal{L}_{rpn}$ and $\partial \mathcal{L}_{rcnn}$ respectively the loss from RPN (localization loss) and RCNN (classification loss), γ the learning rate, and λ_{rpn} and λ_{rcnn} the gradient scaling value for each task. Then, during the forward pass, GDL applies an affine transformation (dense layer) to adapt the feature to the decoupled branches, with learned parameters W and b for weights and biases:

$$GDL(x) = Wx + b. \quad (2)$$

DeFRCN also introduces the Prototypical Calibration Block (PCB), inspired by few-shot classification [17], to enhance classification scores during inference. PCB compares predictions with precalculated class representations

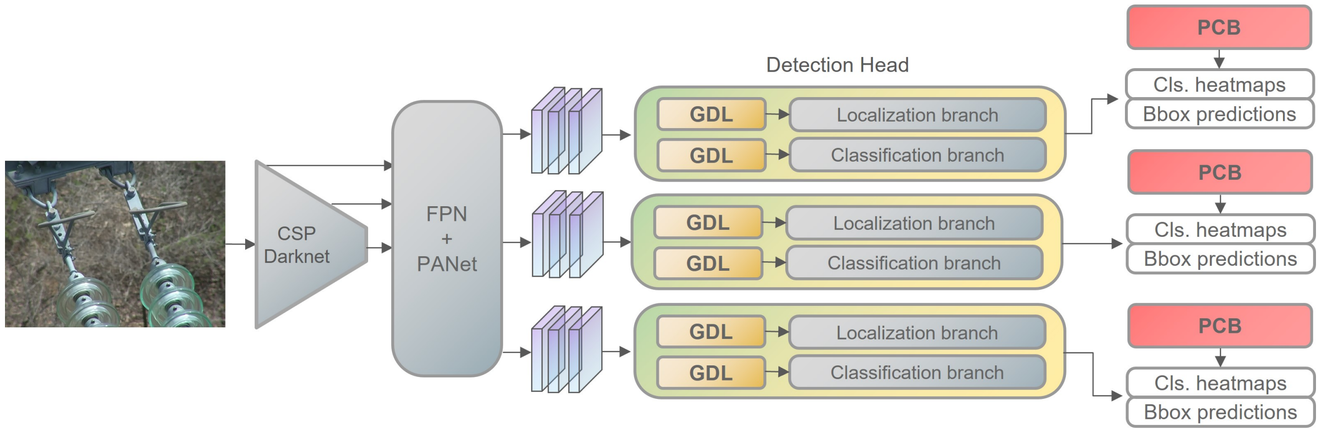

Fig. 3. Illustration of our proposed modification of Yolov8 that we named Yolov8_DeFRCN. GDL adjusts the amplitude of the gradients backpropagated into the backbone by applying scaling factors, λ_{loc} for the localization branch and λ_{cls} for the classification branch. PCB applies its classification score on the heatmaps produced by the classification branch.

(prototypes). These prototypes are generated by passing K-shot of each class through a pretrained feature extractor to obtain vector representations. PCB then averages these vectors for each class to create a prototypes bank $P = \{p_c\}$.

Then during inference on novel images, PCB extracts features f of the query image by using the same pretrained features extractor used for creating P. Next, to obtain the same features size for a bounding box proposed by the RPN f_{bbox} and the prototypes p_c, it uses an RoI Align (i.e: $p_c, f_{bbox} \in \mathbb{R}^{1000}$).

Afterwards, for a bounding box, it calculates cosine similarity between its representation and all the prototypes to obtain the classification score for the PCB module:

$$PCB_{bbox} = \frac{f_{bbox} \cdot p_c}{||f_{bbox}|| ||p_c||}, p_c \in P. \tag{3}$$

Finally, it reports the scores of PCB for all bounding boxes with a weighted sum (parameterized by a value α) with the ones of the RCNN head to obtain the final classification score C_{final} of the model:

$$C_{final} = \alpha \cdot RCNN + (1 - \alpha) \cdot PCB. \tag{4}$$

2) CD-ViTO, Using Self-Supervised Pretrained Transformers for FSOD: Transfer learning object detection has then profited from vision transformers that have proven to be particularly well-suited for self-supervised learning (SSL). Specifically, SSL-pretrained transformers offer better generalization capabilities, which is essential in scenarios with limited data, such as FSOD [18] [19].

Recently, DE-ViT [20] utilizes the capabilities of Dinov2 [21] [22] for FSOD by generating prototypes for each class based on support examples A frozen RPN is used to generate bounding boxes, which are subsequently refined by the propagation network. Then, a dot product is calculated between the prototypes and the features of the query image extracted by Dinov2, similar to the PCB approach in DeFRCN.

CD-ViTO [14] builds on DE-ViT to enhance its performance, particularly in domains significantly different from the MSCOCO benchmark. To better distinguish between closely related precomputed class prototypes in downstream tasks, CD-ViTO converts them into learnable instance features. Instead of averaging the vectors of the same class, it passes them through a multi-layer perceptron (MLP) to highlight high-quality shot while reducing the weight of less relevant ones. Additionally, CD-ViTO incorporates domain adaptation techniques by applying contrastive loss between different prototypes, combined with randomly generated virtual domains. These improvements not only boost DE-ViT's performance on cross-domain FSOD tasks but also improve metrics on traditional FSOD benchmarks like MSCOCO.

C. An adaptation of Yolov8 for fast FSOD

Yolov8 [8] is one of the latest iteration from Ultralytics in the YOLO family [23]. It excels in object detection with high inference speed while having a lightweight architecture making it well-suited for embedded devices, such as UAS applications [24], [25].

Unlike its predecessor, it employs an anchor-free detection head, enabling dense predictions on its feature maps. Yolov8 extracts features on three different scales via Feature Pyramid Network [26] (FPN) and a Path Aggregation Network [27] (PANet), which are then fed into three separate detection heads, one for each scale. These detection heads consist of two parallel branches: one for classification, generating a heatmap for each class, and the other for localization, predicting bounding boxes. Even if Yolov8 uses differents "modules" for these tasks as in faster RCNN, it differs because both are independents and can be calculated in parallel whereas the RCNN head relies on the output of the RPN.

While Yolov8 achieves great performances in object detection when trained with abudant data, it was not originaly designed for FSOD. However, we observed that its new architecture makes it compatible with the GDL and PCB principles presented in DeFRCN as illustrated in Fig. 3, allowing its use in FSOD while maintaining a fast inference

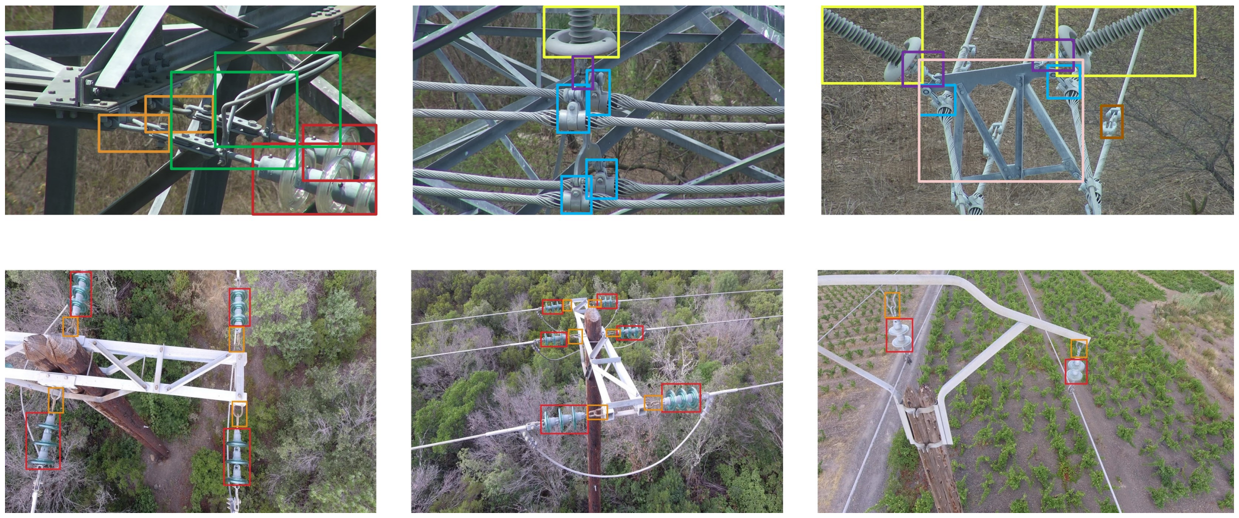
Fig. 4. Samples images from the Insplad dataset [7] in the first row vs our private dataset in the second row.

speed. First, the Gradient Decoupling Layer can be incorporated at the beginning of both branches in the detection head to control the flow of classification and localization gradients during backpropagation into the backbone. GDL, being an affine layer, adds almost no computational overhead during inference. Secondly, the Prototypical Calibration Block can calculate its classification score and modify with a weighted-sum the heatmap output by Yolov8 classification branch. PCB adds some computation for the pretrained feature extractor's inference, but this can run in parallel with YOLO's detection, minimizing overhead. In our experiments, we refer to this modified model as **Yolov8_DeFRCN**.

III. Experiments

For the experiments, we use two databases: one public, Insplad [7], while the second is a private benchmark consisting of UAS images captured in a more realistic setting—specifically, with the drone flying above power lines without focusing precisely on the components, as illustrated in Fig. 4.

A. A Public Power Line Dataset: Insplad

Insplad [7] contains 18 object classes[1] typically found on power lines, with 10,563 HD RGB images (1920x1080), split into 7,936 for training and 2,627 for testing.

Originally designed for classical supervised learning, we adapted the dataset for FSOD by creating two splits and evaluating methods in 5-shot and 10-shot. Unlike FSOD benchmarks such as MSCOCO, where all classes in the fine-tuning stage have the exact same number of training shot, our setup deviates slightly. In practical scenarios, some classes appear only in conjunction with others (e.g., insulators and shackles), making it challenging to have the exact same number of shots in each classes. Our splits[2], detailed in Tab. I, are designed to be as balanced as possible.

[1]Insplad also includes 5 classes for defect classification, which are not utilized in our study since our focus is on the initial detection phase.

[2]https://github.com/XiphosF/Insplad_FSOD

TABLE I
Number of instances per class for our 5-shot and 10-shot splits for the Insplad dataset.

Asset category	5-shot split Inst. Count	10-shot split Inst. Count
Damper - Spiral	5	12
Damper - Stockbridge	5	10
Glass Insulator	5	10
Glass Insulator Big Shackle	5	11
Glass Insulator Small Shackle	6	10
Glass Insulator Tower Shackle	6	10
Lightning Rod Shackle	5	11
Lightning Rod Suspension	5	10
Tower ID Plate	5	10
Polymer Insulator	5	10
Pol. Insulator Lower Shackle	5	10
Pol. Insulator Upper Shackle	5	10
Pol. Insulator Tower Shackle	5	10
Spacer	5	11
Vari-grip	5	10
Yoke	4	6
Yoke Suspension	5	10
Sphere	5	6

B. Our Private Benchmark

Our FSOD benchmark focuses on detecting two object classes similar to those in the Insplad dataset but in a more challenging context: glass insulators and the shackles that connect them to power poles. The glass insulators in this case differ slightly from those in the Insplad dataset, as they consist of a chain of three disks, which is more common in medium and low voltage power lines. For this benchmark, we used a 12-shot setup for both classes, drawing from 3 images as training examples. The detectors were evaluated on 122 instances of glass insulators and 53 instances of shackles with a total number of images of 36 images. The images were 4K (4096x2160) RGB captures from real-world inspections conducted by a technician using a UAS.

C. Protocol and Results

We tested DeFRCN (97 million parameters), CD-ViTO (331 million parameters), our custom Yolov8_DeFRCN, and

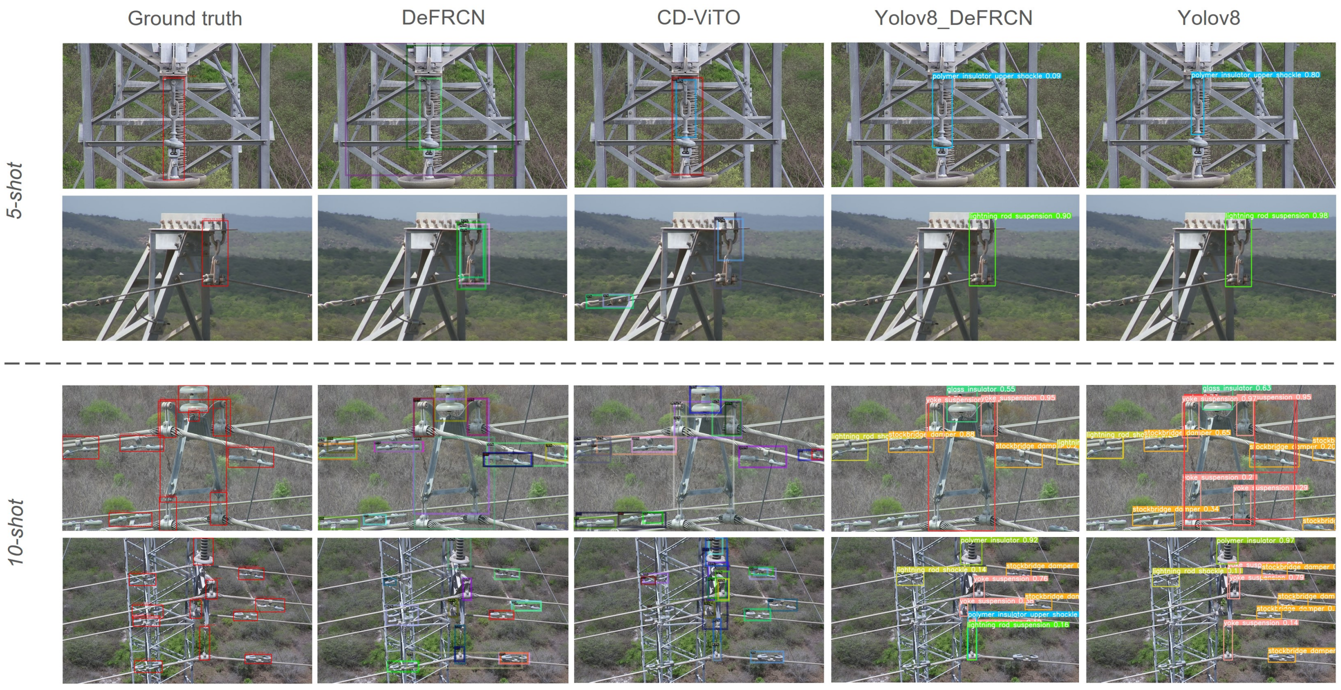

Fig. 5. Detection results from the tested methods on the Insplad dataset [7] for the 5-shot and 10-shot splits.

the baseline Yolov8 models on these two datasets. Note that, for both Yolov8, the medium model (25 million parameters) was used as a reasonable size for embedded device. The training of both Yolov8 used the Ultralytics framework with 100 epochs and a batch size of 128 on two GPUs RTXA6000. Both DeFRCN and CD-ViTO are based on the Detectron2 framework [28] and are used as off-the-shelf models with their default configuration for 5 and 10 shot.

As discussed in Sec. II, these models utilize Transfer Learning techniques. They are first pretrained on large datasets and then fine-tuned using K-shot samples to adapt to new classes. In the FSOD-adapted MSCOCO dataset [11], used for pretraining DeFRCN, Yolov8_DeFRCN, and Yolov8, standard benchmark consists to use 60 of the 80 classes as base classes for pretraining, while the remaining 20 are used for K-shot finetuning. For CD-ViTO, all learnable parameters were pretrained on the FSOD-adapted MSCOCO, except for its frozen backbone, Dinov2, which was pretrained on the private LVD-142M dataset [21].

Finally, to place ourselves in a more realistic case of FSOD, we do not use a validation dataset during the fine-tuning stage. Indeed, given that we only have K-shot for training, we cannot determine the optimal stopping point for training to achieve the best performance for the models. We therefore trained the models for 100 epochs with a learning rate one-tenth of the pretraining learning rate. The metrics used are mean Average Precision (mAP), split into base AP (bAP) for pretraining classes and novel AP (nAP) for new classes.

The performance of all four selected models on the 5-shot and 10-shot splits of Insplad is summarized in Tab. II, with qualitative results illustrated in Fig. 5.

TABLE II
AVERAGE PRECISION IN OUR 5-SHOT AND 10-SHOT SPLITS OF THE INSPLAD DATASET [7], AVERAGED OVER ALL NEW CLASSES.

Method	**Shot**	**nAP50**	**nAP50-95**
Yolov8 [8]	5-shot	**52.5**	**38.6**
Yolov8_DeFRCN	5-shot	45.6	33.0
DeFRCN [9]	5-shot	34.2	20.5
CD-VITO [14]	5-shot	50.5	32.4
Yolov8 [8]	10-shot	**62.6**	**45.7**
Yolov8_DeFRCN	10-shot	59.6	44.1
DeFRCN [9]	10-shot	53.5	31.0
CD-VITO [14]	10-shot	59.0	37.6

TABLE III
AVERAGE PRECISION AFTER 12-SHOT FINETUNING ON INSULATORS AND SHACKLES OF OUR PRIVATE DATASET.

12-shot	**Insulator**		**Shackle**	
	nAP50	**nAP50-95**	**nAP50**	**nAP50-95**
Yolov8 [8]	**89.8**	64.7	41.4	**20.7**
Yolov8_DeFRCN	89.2	**67.6**	**42.8**	19.6
DeFRCN [9]	84.8	57.6	17.0	6.0
CD-ViTO [14]	84.9	56.8	29.5	8.3

We then conducted the 12-shot finetuning on our private benchmark which results are presented in Tab. III.

IV. ANALYSIS OF RESULTS

Both datasets lead to the same unexpected conclusion: FSOD-specialized methods do not achieve the best performance on this task. One might assume that the specific architecture of Yolov8 is particularly well-suited for these types of objects, but even our adapted Yolov8_DeFRCN model performs worse than the baseline. This is especially

TABLE IV

PERFORMANCE METRICS ON FSOD-ADAPTED MSCOCO [11] OF YOLOV8_DEFRCN. AVERAGE PRECISION IS REPORTED FOR BOTH BASE (BAP) AND NOVEL (NAP) CLASSES SEPARATELY.

	1-shot				***2-shot***			
	bAP50	bAP50-95	nAP50	nAP50-95	bAP50	bAP50-95	nAP50	nAP50-95
Yolov8 [8]	13.9	9.1	3.9	2.3	12.9	8.9	6.3	3.7
Yolov8_DeFRCN	**54.5**	**38.8**	10.5	**7**	49.0	**34.6**	12.7	7.8
DeFRCN [9]	48.7	31.8	**10.9**	6.5	**49.7**	32.5	**20.6**	**11.7**
	3-shot				***5-shot***			
	bAP50	bAP50-95	nAP50	nAP50-95	bAP50	bAP50-95	nAP50	nAP50-95
Yolov8 [8]	19.1	13.4	9.3	5.6	19.3	13.6	11.7	7.4
Yolov8_DeFRCN	**55.9**	**40.4**	15.9	10.8	**53.0**	**38.1**	23.0	15.0
DeFRCN [9]	49.8	32.5	**24.2**	**13.4**	50.6	33.1	**28.4**	**15.3**
	10-shot				***30-shot***			
	bAP50	bAP50-95	nAP50	nAP50-95	bAP50	bAP50-95	nAP50	nAP50-95
Yolov8 [8]	19.3	13.6	15.5	10.0	23.6	16.9	23.2	15.7
Yolov8_DeFRCN	49.6	**34.7**	27.9	17.9	51.7	**36.2**	33.4	22.1
DeFRCN [9]	**53.1**	34.5	**34.5**	**18.5**	**52.8**	34.6	**39.9**	**22.4**

TABLE V

GAINS ACHIEVED BY THE PCB MODULE IN YOLOV8_DEFRCN IN THE 5-SHOT AND 10-SHOT SPLITS FROM FSOD-ADAPTED MSCOCO [11] AND INSPLAD, SHOWING THE INFLUENCE OF THE PRETRAINED FEATURE EXTRACTOR.

PCB gain	**MSCOCO**		**Insplad**	
Splits	5-shot	10-shot	5-shot	10-shot
nAP50	+1.7	+1.7	-3.4	-3.1
nAP50-95	+1.0	+1.0	-0.1	-0.2

surprising when comparing the results of vanilla Yolov8 and Yolov8_DeFRCN on the standard FSOD benchmark MSCOCO. Our best-performing Yolov8_DeFRCN model was achieved with $\lambda_{loc} = 0.25$ and $\lambda_{cls} = 0.75$ during pre-training and $\lambda_{loc} = 0.1$ and $\lambda_{cls} = 0.1$ during K-shot finetuning for the GDL, with $\alpha = 0.5$ for PCB. When evaluated on the MSCOCO dataset, this model showed improvements over the baseline Yolov8 and achieved performance levels approaching those of DeFRCN as shown in Tab. IV, while being faster and more lightweight (25M vs 97M parameters).

However, we argue that some FSOD methods that perform well on datasets like MSCOCO may struggle in other domains with different types of objects. As discussed in Sec. II, the PCB module relies on a pretrained feature extractor to enhance performance. The feature extractor used in both DeFRCN and our Yolov8_DeFRCN is ResNet101 [29], pretrained on ImageNet [30]—a dataset closely aligned with MSCOCO. However, it is likely that the prototypes learned from this feature extractor are less relevant for industrial objects like those in our dataset, leading to reduced performance. This is evident when comparing the impact of the PCB module in our Yolov8_DeFRCN on FSOD-adapted MSCOCO versus Insplad, as shown in Tab. V.

V. CONCLUSION

In this article, we presented the performance of several state-of-the-art FSOD-specialized methods, along with our new FSOD-adapted architecture based on Yolov8, for detecting power line components in real-world applications. We evaluated these models on two datasets that contain similar object classes but differ in UAS capture contexts. The results were surprising: the baseline detector, Yolov8, outperformed the FSOD methods. This highlights that, in practical use cases, FSOD methods remain challenging to implement effectively, partly due to biases in the algorithms and limitations in traditional FSOD benchmarks. Moreover, addressing this issue could enable a highly data-efficient pipeline for unsupervised anomaly detection methods to identify faults in power line components [31], which currently rely on extensively trained detectors for their analysis. Finally, future research could investigate the integration of Vision Langugage Model (VLM) that use internet-scale pre-training data to obtain general knowledge from both text and images for zero-shot object detection [32], [33]. However, adopting this paradigm for FSOD would necessitate rethinking current evaluation methodologies [34], particularly in light of potential data contamination issues [35].

REFERENCES

[1] L. Yang, J. Fan, Y. Liu, E. Li, J. Peng, and Z. Liang, "A review on state-of-the-art power line inspection techniques," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 12, pp. 9350–9365, 2020.

[2] Z. Wang, Q. Gao, J. Xu, and D. Li, "A review of uav power line inspection," in *Advances in Guidance, Navigation and Control: Proceedings of 2020 International Conference on Guidance, Navigation and Control, ICGNC 2020, Tianjin, China, October 23–25, 2020*. Springer, 2022, pp. 3147–3159.

[3] V. N. Nguyen, R. Jenssen, and D. Roverso, "Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning," *International Journal of Electrical Power and Energy Systems*, vol. 99, pp. 107–120, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0142061517324444

[4] X. Liu, X. Miao, H. Jiang, and J. Chen, "Data analysis in visual power line inspection: An in-depth review of deep learning for component detection and fault diagnosis," *Annual Reviews in Control*, vol. 50, pp. 253–277, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578820300596

[5] R. Jenssen, D. Roverso, *et al.*, "Intelligent monitoring and inspection of power line components powered by uavs and deep learning," *IEEE Power and energy technology systems journal*, vol. 6, no. 1, pp. 11–21, 2019.

[6] Y. Zhang, S. Cao, L. Mu, X. Xue, J. Xin, and Y. Zhang, "A fault detection method for power transmission lines using aerial images," in *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2024, pp. 1247–1252.

[7] F. P. M. S. Andre Luiz Buarque Vieira e Silva, Heitor de Castro Felix, "Insplad: A dataset and benchmark for power line asset inspection in uav images," *International Journal of Remote Sensing*, vol. 44, no. 23, pp. 1–27, 2023. [Online]. Available: https://doi.org/10.1080/01431161.2023.2283900

[8] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[9] L. Qiao, Y. Zhao, Z. Li, X. Qiu, J. Wu, and C. Zhang, "Defrcn: Decoupled faster r-cnn for few-shot object detection," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8681–8690, 2021.

[10] M. Köhler, M. Eisenbach, and H.-M. Gross, "Few-shot object detection: a comprehensive survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[11] X. Wang, T. Huang, J. Gonzalez, T. Darrell, and F. Yu, "Frustratingly simple few-shot object detection," *Proceedings of the 37th International Conference on Machine Learning*, vol. 119, pp. 9919–9928, 4 2020. [Online]. Available: https://proceedings.mlr.press/v119/wang20j.html

[12] Michael, B. Serge, H. James, P. Pietro, R. Deva, D. Piotr, Z. C. L. L. Tsung-Yi, and Maire, "Microsoft coco: Common objects in context," *Computer Vision – ECCV 2014*, pp. 740–755, 2014.

[13] X. Wu, D. Sahoo, and S. Hoi, "Meta-rcnn: Meta learning for few-shot object detection," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1679–1687.

[14] Y. Fu, Y. Wang, Y. Pan, L. Huai, X. Qiu, Z. Shangguan, T. Liu, L. Kong, Y. Fu, L. Van Gool, *et al.*, "Cross-domain few-shot object detection via enhanced open-set object detector," *arXiv preprint arXiv:2402.03094*, 2024.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[16] B.-B. Gao, X. Chen, Z. Huang, C. Nie, J. Liu, J. Lai, G. Jiang, X. Wang, and C. Wang, "Decoupling classifier for boosting few-shot object detection and instance segmentation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 18 640–18 652, 2022.

[17] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[18] F. Liu, X. Zhang, Z. Peng, Z. Guo, F. Wan, X. Ji, and Q. Ye, "Integrally migrating pre-trained transformer encoder-decoders for visual object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 6825–6834.

[19] A. Bar, X. Wang, V. Kantorov, C. J. Reed, R. Herzig, G. Chechik, A. Rohrbach, T. Darrell, and A. Globerson, "Detreg: Unsupervised pretraining with region priors for object detection," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14 605–14 615, 2022.

[20] X. Zhang, Y. Wang, and A. Boularias, "Detect every thing with few examples," *arXiv preprint arXiv:2309.12969*, 2023.

[21] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," *arXiv:2304.07193*, 2023.

[22] T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski, "Vision transformers need registers," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=2dnO3LLiJ1

[23] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, 2023.

[24] M. A. Farooq, W. Shariff, and P. Corcoran, "Evaluation of thermal imaging on embedded gpu platforms for application in vehicular assistance systems," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1130–1144, 2023.

[25] C. Fite, M. Yedroudj, M. Chaumont, P. Serres-Noïtaky, F. Comby, F. Forget, G. Subsol, L. Mannocci, M. Capello, M. Tolotti, and L. Dagorn, "Some considerations to design an autonomous buoy system to enumerate pelagic sharks at Fish Aggregating Devices," LIRMM (UM, CNRS), Tech. Rep. 1, Aug. 2024. [Online]. Available: https://hal-lirmm.ccsd.cnrs.fr/lirmm-04895187

[26] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[27] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[28] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[31] G. Fourret, C. Fiorio, G. Subsol, M. Chaumont, and S. Brau, "Anomaly detection in drone videos for preventive maintenance of power lines," in *IGARSS 2025 - 2025 IEEE International Geoscience and Remote Sensing Symposium*, 2025.

[32] M. Minderer, A. Gritsenko, and N. Houlsby, "Scaling open-vocabulary object detection," *Advances in Neural Information Processing Systems*, vol. 36, pp. 72 983–73 007, 2023.

[33] A. Wang, L. Liu, H. Chen, Z. Lin, J. Han, and G. Ding, "Yoloe: Real-time seeing anything," 2025. [Online]. Available: https://arxiv.org/abs/2503.07465

[34] A. Madan, N. Peri, S. Kong, and D. Ramanan, "Revisiting few-shot object detection with vision-language models," *arXiv preprint arXiv:2312.14494*, 2023.

[35] V. Udandarao, A. Prabhu, A. Ghosh, Y. Sharma, P. Torr, A. Bibi, S. Albanie, and M. Bethge, "No" zero-shot" without exponential data: Pretraining concept frequency determines multimodal model performance," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.