A NanoDet Model with Adaptively Weighted Loss for Real-time Railroad Inspection

Jiawei Guo[1], Sen Zhang[2], Yu Qian[3], Yi Wang[4*]

[1,4] *Department of Mechanical Engineering, University of South Carolina, Columbia, SC, 29208*
jiaweig@email.sc.edu
yiwang@cec.sc.edu (corresponding author)

[2]*Department of Computer Science and Engineering, University of South Carolina, Columbia, SC, 29208*

senz@email.sc.edu

[3] *Department of Civil and Environmental Engineering, University of South Carolina, Columbia, SC, 29208*
yuqian@sc.edu

ABSTRACT

Monitoring the railroad's components is crucial to maintaining the safety of railway operations. This article proposes a novel, compact computational vision system that works on edge devices, designed to provide precise, instantaneous assessments of rail tracks. This model reconfigures the teacher-student guidance system inherent in NanoDet [1] by incorporating an innovative adaptively weighted loss (AWL) in the learning phase. The AWL assesses the caliber of the teacher and student models, establishes the weightage of the student's loss, and dynamically adjusts their loss contributions, directing the learning procedure towards effective knowledge transfer and direction. In comparison with cutting-edge models, our AWL-NanoDet boasts a minuscule model size of less than 2 MB and a computational expense of 1.52 G FLOPs, delivering a processing time of less than 14 ms per frame (evaluated on Nvidia's AGX Orin). Compared to the original NanoDet, it also significantly enhances the model's accuracy by nearly 6.2%, facilitating extremely precise, instantaneous recognition of rail track elements.

1. INTRODUCTION

As reported by the Federal Railroad Administration's safety database [2], there were over 300 accidents in 2022 caused by missing track components, resulting in losses of more than $85 million. Rigorous inspections are essential to identify flaws in rail roads and ensure the safety of train's operations. However, most existing methods are manual and heavily rely heavily on operator's experience, making them costly and time-consuming. As such, the need for an automated, real-time, and cost-effective image-based system for accurate rail track inspections is significant.

Convolutional Neural Networks (CNN) have revolutionized computer vision and object detection recently. LeNet [3], AlexNet [4], Visual Geometry Group (VGG) [5], and ResNet [6] are a few notable CNN models that have contributed to the field's progress. LeCun et al. introduced CNN and demonstrated its superiority in classification tasks compared to traditional approaches. AlexNet enhanced the capabilities of CNN by delving into the extraction of more complex features through its relatively deep CNN layers (5 layers). In additional, VGG used a deeper CNN network (16 or 19 layers) for constructing more comprehensive features. However, the increasing depth of these networks resulted in higher probability of overfitting and deterioration in real detection tasks. He et al. addressed these problems by proposing a ResNet, which introduced shortcut connections to enable deeper CNN models while mitigating the issues above.

All these significant advances in CNN-based models have laid a strong foundation for developing highly accurate systems, inspiring a range of real-world applications. CNN-based object detection models can be broadly classified into two categories: anchor-based and anchor-free models. The former relies on preconfigured anchors to match corresponding objects, while the latter circumvents the use of those anchors. Noteworthy examples of anchor-based methods that have demonstrated remarkable performances in reported results include RCNN (proposed by Girshick et al.

[7]) and YOLO (proposed by Redmon et al. [8]). Despite the success of anchor-based models, a notable drawback arises from the anchor matching step, which could greatly slow down model inference. To address this limitation, the anchor-free method was introduced as an alternative. One notable example is CornerNet devised by Law et al. [9], which directly generates bounding boxes (BBs) without relying on anchor matching. Furthermore, Tian et al. [10] introduced the fully convolutional one-stage object detection (FCOS) method, which not only regresses the object's centroid but also utilizes parallel computation to calculate the distance between the centroid and the bounding box's boundary. Consequently, FCOS exhibits superior performance in various object detection tasks. No matter it is anchor-free or anchor-based, the utilization of deeper model structures has resulted in a great growth in computational cost. Consequently, deploying these complex models on small form-factor edge devices becomes challenging due to their slow runtime, which hinders real-time object detection capabilities. As a result, substantial research endeavors have been dedicated to developing lightweight models and approaches. These efforts aim to address the computational challenges and enable the efficient deployment of CNN models on edge devices, ensuring real-time object detection capabilities are achievable. Howard et al. [11] introduced MobileNet, a well-known lightweight model structure used as a backbone. One of its key innovations is the use of depth-wise separable convolutions, which greatly reduce the computational load for models. This design has made MobileNet an efficient solution for deploying CNN models on resource-constrained devices, striking a balance between model size and performance. In addition to MobileNet, another notable lightweight model structure is ShuffleNet, designed by Zhang et al. [12], which also utilizes depth-wise separable convolutions, but incorporates a creative shuffling of input feature information. This shuffling operation enhances the feature combination across different channels, leading to a more comprehensive feature representation and significantly enriching the overall feature information. Based on those excellent properties, RangiLyu et al. [1] leveraged the ShuffleNet to develop NanoDet, a lightweight object detection model. The integration of ShuffleNet into NanoDet translates into superior object detection performance, particularly in resource-limited scenarios.

Meanwhile, the smaller number of trainable parameters in lightweight models introduces a limitation of poorer learning capabilities, potentially restricting its ability to achieve high performance and adapt to complex patterns and variations in the data. Hinton et al. [13] proposed knowledge distillation (KD) to overcome the negative effects of over-simplification in lightweight models. KD uses a more complex teacher model to guide a simplified student model, facilitating knowledge transfer. Label Assignment Distillation (LAD) developed by Nguyen et al. [14] is based on KD. The teacher model in LAD, with a more complex structure, generates simplified labels for the student model, and enhances the training process. In the case of NanoDet, a similar approach of LAD is employed, where an Assign Guidance Module (AGM) is implemented as the teacher model, which assigns learning labels to the prediction head of the student model and facilitates learning convergence. By utilizing LAD, NanoDet enhances the learning capabilities and performance of the lightweight student model.

Some of the above-mentioned advances have also made significant contributions to the development of accurate and real-time models and processes for Prognostics and Health Management (PHM) in facilities and infrastructure.

Dais et al. [15] introduced a CNN-based system designed specifically for the classification and localization of cracks on masonry surfaces. Given that masonry structures constitute a significant portion of the global building stock, the development of a highly accurate and automated system is of paramount importance for ensuring public safety. Additionally, Zhang et al. [16] and Cha et al. [17], each developed CNN-based models for bridge inspection. Both models significantly enhance accuracy in detecting surface damages like cracks, spalling, and corrosion, thereby playing a crucial role in ensuring bridge safety and structural integrity. Moreover, Guo et al. [18], [19] extended YOLOv4 and YOLACT models for inspecting track components in railroads. Their work achieved satisfactory performance in classifying and localizing components like spikes and clips, improving the practice of maintenance and safety in the railroad industry. In another domain, Li et al. [20] introduced a CNN-based model tailored for inspecting damages on the surfaces of concrete structures. This innovative approach enables a visual assessment of the safety, durability, and serviceability of concrete structures.

While these influential studies demonstrated the potential of achieving real-time performance on powerful computing platforms, they are not very well-suited for field applications. And there is a notable scarcity of research endeavors focused on the development of lightweight models specifically tailored for edge devices, thereby enabling true mobile computing and field-deployable inspection.

Addressing the pressing demand for the real-time safety inspection, this paper presents an edge-device-compatible computer vision model based on the emerging NanoDet framework for real-time assessment of rail track components. Our proposed model redefines NanoDet's training process by introducing an adaptively weighted loss (AWL) and a dynamic algorithm online adjusting their loss metrics during the training phase. With this strategy, our model guarantees an effective and stable knowledge distillation process, achieving superior results than native NanoDet.

The subsequent sections of this paper are organized as follows: Section 2 will examine the framework of native NanoDet and discuss its existing issue. Section 3 is dedicated

to the delineation of the AWL pipeline. Section 4 will describe the experimental setup and results with interpretation. Section 5 will illustrate the conclusion of the proposed method.

2. PRELIMINARIES

This section will first describe the architecture of the native NanoDet, which forms the foundation of the current study. Subsequently, the challenges associated with NanoDet will be discussed.

2.1. NanoDet

The NanoDet is an extremely compact model, drawing its inspiration from the LAD. It incorporates a teacher model that, during the training phase, assists a student model in learning data, mitigating potential adverse effects caused by its lightweight nature. The architectural design of the native NanoDet is portrayed in Figure 1, comprised of three constituent modules:

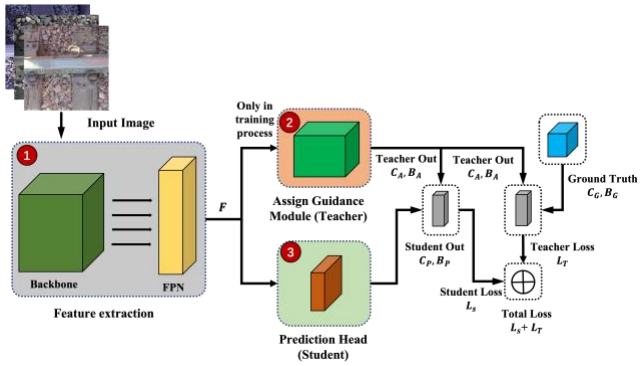


Figure 1. The structure of native NanoDet

1. The first part, crucial to the object detection task, comprises the Backbone and the Feature Pyramid Network (FPN), which together establish a feature extraction framework. This widely used structure is capable of producing salient quality of the feature maps denoted as F in Figure 1.
2. The Assign Guidance Module (AGM), or teacher model, processes feature maps F to predict bounding boxes (B_A) and their respective classes (C_A). The model's predictions are compared with the ground truths of the bounding box (B_G) and class (C_G), yielding a loss metric (L_T). Instead of B_G and C_G, these predictions (B_A and C_A) are used as the reassigned labels for student model training - a method known as Label Assignment Distillation (LAD).
3. The prediction head, often referred to as the student model, utilizes the same feature maps F as those in the teacher model for predicting the bounding box (B_P) and associated class labels (C_P). The predictions from the student model, however, are compared with the teacher model's reassigned labels (B_A and C_A), not the ground truth, to determine the student model's loss (L_S). This is a primary departure from conventional model training methods.

$$C_A, B_A = T_i(F)$$
$$C_P, B_P = S_i(F) \quad (1)$$

In Eq. (1), "i" signifies the models at the i^{th} training epoch. The teacher model, T_i, provides class output C_A and bounding box output B_A, while S_i, the student model, delivers classification output C_P and bounding box output B_P. The teacher loss (L_T), a combination of binary cross entropy classification loss (BCE) and Generalized Intersection over Union bounding box loss (GIOU), is then computed as Eq. (2) below. The reassigned labels C_A and B_A are used to train S_i in the learning process.

$$L_T = BCE(C_A, C_G) + GIOU(B_A, B_G) \quad (2)$$

Moreover, as depicted in Eq. (3), NanoDet's student loss (L_S) is derived by comparing S_i 's outputs (C_P, B_P) with the reassigned labels (C_A, B_A).

$$L_S = BCE(C_P, C_A) + GIOU(B_P, B_A) \quad (3)$$

This equation signifies that S_i relies on these reassigned labels, rather than the ground truth, for optimized learning when T_i is in operation. The quality of T_i profoundly influences NanoDet's efficacy. Thus, the cumulative loss for training NanoDet is $L_S + L_T$.

Once the training is completed, the teacher model is removed, leaving the student model to make predictions.

2.2. Issues in NanoDet

The native NanoDet's teacher model employs the COCO dataset [21] for training. Yet, our research utilizes a distinctive dataset sourced from railroad inspections, as shown in Figure 2, which is considerably different from COCO. This disparity necessitates the retraining of the teacher model, which introduces new issues.

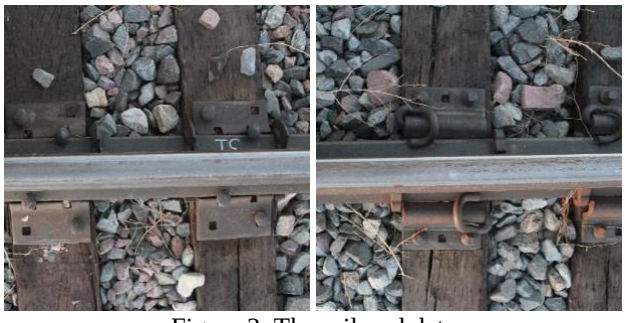


Figure 2. The railroad data

The NanoDet's teacher model has a compact structure composed of only four 3x3 convolutional kernels, could be trained insufficiently at the beginning of the training process. This can lead to possible mislabeling of data intended for guiding the student model. Given the inherent structure of

Label Assignment Distillation (LAD), the quality of the student model's training is intrinsically dependent on the teacher model's performance. If the teacher model's output distribution diverges drastically from the actual distribution, it could induce a wrong direction for early-stage training, which might degrade the student model's quality and impede the backpropagation of loss, thereby slowing down the training process.

Nevertheless, if the teacher model is meticulously trained to a high qualify from the very beginning and is capable of generating accurate labels, it will certainty improve the efficacy of the student model's training. This immaculate level of instruction from an impeccably trained teacher model not only boosts the student model's ability to grasp true data patterns but also accelerates the learning process, hence enhancing its overall performance.

3. PROPOSED ADAPTIVELY WEIGHTED LOSS (AWL)

To remedy this issue, we introduce an innovative loss algorithm, viz., the adaptively weighted loss (AWL), which evaluates and dynamically adjusts the weight of the student loss based on the quality of the teacher and student models in each training epoch. At the beginning of training, the teacher loss is given priority to facilitate rapid teacher learning and prevent generation of incorrect labels. Once the teacher model is mature, the emphasis shifts toward the student model.

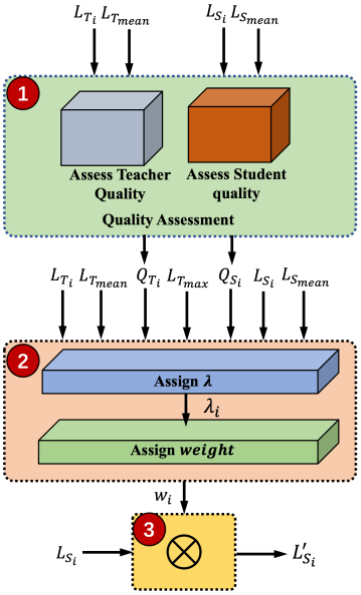


Figure 3. The pipeline of the adaptively weighted loss (AWL)

Figure 3 illustrates our proposed pipeline, consisting of three critical components: firstly, the assessment of the teacher (Q_{T_i}) and student (Q_{S_i}) model qualities in the i^{th} training epoch; secondly, the assignment of a quantitative score to the teacher model in the i^{th} epoch (λ_i), and the determination of a weight (w_i) for the student loss based on current Q_{T_i}, Q_{S_i} and λ_i; finally, the updating of student loss L_{S_i} using weight w_i, forming a weighted student loss L'_{S_i} in response to various training situations.

3.1. Quality Assessment

Our method begins with the determination of the quality metrics for the teacher model T_i and the student model S_i at the i^{th} epoch. These quality measures are instrumental in defining the procedures for assigning the weight w_i and adjusting the student loss L_{S_i}. The final loss for backpropagation during training incorporates both the updated student loss and teacher loss, as given by $L'_{S_i}+L_{T_i}$. Further details of each component will be elaborated in the subsequent sections. An essential prerequisite of AWL is to have a pre-training process employing the native NanoDet model without AWL, which generates the pre-trained loss at each training epoch ($L_{T_i}^{pre}$ and $L_{S_i}^{pre}$). Then its mean and the maximum teacher loss ($L_{T_{mean}}$ and $L_{T_{max}}$) can be extracted as shown in Eq. (4)

$$\begin{aligned} L_{T_{mean}} &= \frac{1}{M^{pre}}\sum_{i=1}^{M^{pre}} L_{T_i}^{pre} \\ L_{S_{mean}} &= \frac{1}{M^{pre}}\sum_{i=1}^{M^{pre}} L_{S_i}^{pre} \\ L_{T_{max}} &= Max\left(L_{T_i}^{pre}\right) \\ L_{S_{max}} &= Max\left(L_{S_i}^{pre}\right) \\ i &= 1, \dots, M^{pre} \end{aligned} \tag{4}$$

The superscript *pre* denotes quantities associated with the pre-training phase, with M^{pre} being the total number of training epochs in that phase. Once the constants $L_{T_{mean}}$, $L_{S_{mean}}$, $L_{T_{max}}$, and $L_{S_{max}}$ are determined, they function as benchmark values for evaluating the performance of the teacher and the student models during the AWL-integrated training process.

Eq. (5) establishes that the quality of the teacher model Q_{T_i} and the student model Q_{S_i} at the i^{th} epoch based on their respective losses L_{T_i} and L_{S_i}, compared to the mean losses $L_{T_{mean}}$ and $L_{S_{mean}}$ in the pre-training

$$\begin{aligned} Q_{T_i} &= \begin{cases} 0, & L_{T_i} \le L_{T_{mean}} \\ 1, & L_{T_i} > L_{T_{mean}} \end{cases} \\ Q_{S_i} &= \begin{cases} 0, & L_{S_i} \le L_{S_{mean}} \\ 1, & L_{S_i} > L_{S_{mean}} \end{cases} \end{aligned} \tag{5}$$

$Q_{T_i} = 0$ resulting from L_{T_i} lesser than $L_{T_{mean}}$, signifies a relatively low loss and high qualify of the teacher model, that is, the teacher is qualified for guiding the student model. In contrast, if $L_{T_i} > L_{T_{mean}}$, it yields $Q_{T_i} = 1$, indicating poor

teacher model's quality. A similar criterion can be applied to the student model to assess its quality.

3.2. Score/Weight Assignment

The rationale behind the aforementioned quality evaluation lies in accounting for the different scenarios involving various combinations of T_i and S_i qualities, and, accordingly, assigning suitable scores and weights to the student loss during model training. In Eq. (5), we have four possible combinations of T_i and S_i qualities. The methodologies for assigning scores and weights in each combination are explained in the subsequent sections.

1. $Q_{T_i} = 0$ and $Q_{S_i} = 0$:

In this first scenario, where both the teacher and student models are qualified, we propose to set w_i using Eq. (6), where w_i remains a constant value of 1, and the adjusted student loss L'_{S_i} will equal to L_{S_i}. The rationale is that when both T_i and S_i are qualified, T_i can accurately predict the ground truth and provide high-quality labels to instruct S_i, and S_i is also qualified to learn from the labels reassigned by T_i. Then, there is no need for weight adjustments in the loss, which can be used directly for backpropagation.

$$\begin{aligned} w_i &= 1 \\ L'_{S_i} &= w_i * L_{S_i} \end{aligned} \tag{6}$$

2. $Q_{T_i} = 1$ and $Q_{S_i} = 1$:

In the scenario where both T_i and S_i are unqualified, the weight assignment, w_i, follows Eq. (7). Here, λ_i, defined in the first row of Eq. (7), measures the degree to which T_i deviates from being qualified. A larger λ_i indicates a significant deviation from qualification for T_i. Given that, for an unqualified teacher, L_{T_i}-$L_{T_{mean}}$ will always be positive, normalizing by $L_{T_{max}}$ ensures λ_i ranges from 0 to 1. This calculated λ_i is utilized as a penalty coefficient, multiplied with the teacher's loss L_{T_i} in the second row of Eq. (7), along with a hyperparameter, c, acting as a gain factor for λ_i*L_{T_i} to compute w_i. Noticeably, w_i is confined between 0 and 1. Eq. (7) means that an increase in L_{T_i} due to the teacher's poor alignment with the ground truth leads to a corresponding rise in λ_i and a decrease in w_i. This implies that when T_i is significantly unqualified, guiding S_i becomes futile, prompting a smaller w_i to downscale L_{S_i}, thereby reducing its contribution to the total loss. This allows prioritizing training of the teacher model to enhance its qualifications.

$$\begin{aligned} \lambda_i &= \frac{L_{T_i} - L_{T_{mean}}}{L_{T_{max}}} \\ w_i &= 1 - \tanh\left(c * \lambda_i * L_{T_i}\right) \\ L'_{S_i} &= w_i * L_{S_i} \end{aligned} \tag{7}$$

3. $Q_{T_i} = 0$ and $Q_{S_i} = 1$:

In the third scenario, where T_i is qualified but S_i is unqualified, the assignment of w_i follows Eq. (8). As suggested by Eq. (5), when T_i is qualified, it means L_{T_i} is less than $L_{T_{mean}}$, resulting in a negative value for L_{T_i}-$L_{T_{mean}}$. Applying the operation of absolute value ensures λ_i remains positive. The first two rows of Eq. (8) imply that when T_i is highly qualified (i.e., L_{T_i} is very low), and hence capable of generating accurate labels for S_i, both the score λ_i and weight w_i become large. Here, w_i ranges from 1 to 2, thereby scaling up the student loss L_{S_i} and increasing its contribution to the total loss. Consequently, the training process for S_i is expedited, enabling S_i to learn faster towards T_i.

$$\begin{aligned} \lambda_i &= \frac{\left|L_{T_i} - L_{T_{mean}}\right|}{L_{T_{max}}} \\ w_i &= 1 + \tanh\left(c * \frac{\lambda_i}{L_{T_i}}\right) \\ L'_{S_i} &= w_i * L_{S_i} \end{aligned} \tag{8}$$

4. $Q_{T_i} = 1$ and $Q_{S_i} = 0$:

In the final scenario, where T_i is unqualified while S_i is qualified, the score and weight assignment is the same as the formula in Eq. (7). That is, that, when T_i is unqualified and generates erroneous label reassignments, the student loss becomes insignificant. Consequently, its contribution to the total loss should be diminished to expedite the training of the teacher model, prioritizing its convergence for qualification.

3.3. Loss Update

The total training loss, derived from the different scenarios, is given by L'_{S_i} + L_{T_i}. The non-linear correlation between L_{T_i} and w_i forms the core of our AWL method and is visualized in Figure 4.

From this diagram, it's clear that the weight reduces non-linearly with the increasing teacher loss. When the teacher model is well-qualified, a large weight is assigned to the loss of the unqualified student (blue curve) to speed up learning of the latter. When both the student and the teacher are qualified (green curve), the weight is constant 1, leading to behavior similar to the native NanoDet. For unqualified teacher models (Orange line) where teacher loss exceeds 0.8 (the threshold for qualifying a teacher model.), the student loss becomes less important regardless of the student's qualifications. Therefore, a weight lower than 1 is assigned to the student loss, prioritizing the improvement of the teacher model's training process.

Our method, illustrated in Figure 3, can be succinctly represented by Eq. (9). T_i and S_i denote the teacher and student models at the i^{th} epoch, respectively. The Score

function determines the accurate level of these models, consequently generating the weight w_i. The student loss, L_{S_i}, is updated by multiplying it with w_i, mitigating learning from an inadequate teacher model.

$$w_i = Score(T_i, S_i)$$
$$L'_{S_i} = w_i * L_{S_i} \quad (9)$$
$$i = 1, \dots, M^{tr}$$

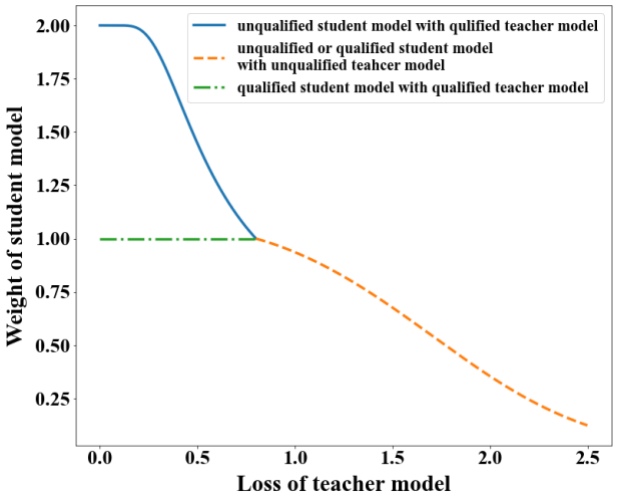


Figure 4. Teacher loss vs. student weight

4. EXPERIMENTAL SETUP AND RESULT

The AWL-NanoDet model and other baseline models were trained on a comprehensive dataset of 2,000 railroad inspection images, leveraging the computational power of the NVIDIA RTX A5000 GPU, equipped with 8,192 CUDA cores and capable of 27.77 TFLOPS. Those models were later tested on Nvidia's AGX Orin, an edge computing device of modest specifications, including 2,048 CUDA cores and 5.3 TFLOPS, which poses a challenge for real-time detection.

4.1. Dataset collection & pre-process

For data collection, four high-speed cameras, generating videos with a resolution of 416*416 pixels at the rate of 60 fps, were mounted under a Hi-rail truck, as Figure 5 shows, each targeting a different trackside.

Our study focused on detecting two key railroad components: clips and spikes, as shown in Figure 6. We collected 2000 images, and the images are split at an 80% (1600 images) to 20% (400 images) ratio for model training and testing, respectively. Given the fixed perspective angles between the camera and the rail, extensive data augmentation is not necessary. We only implement brightness augmentation to handle variable lighting conditions during the inspection, as Figure 7 shows.

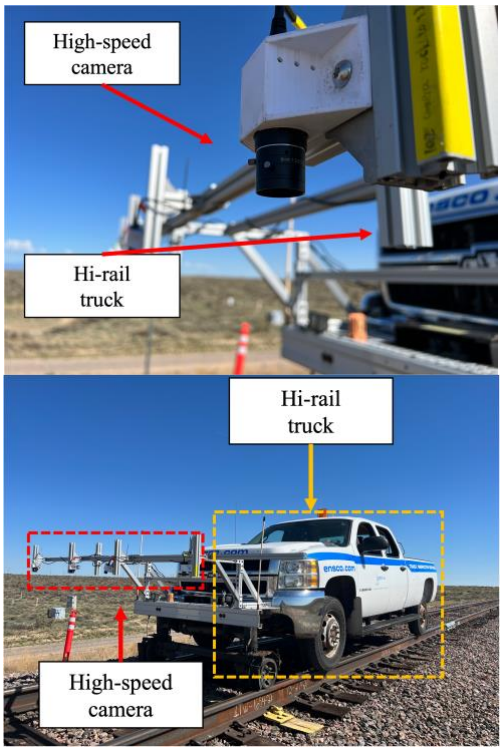


Figure 5. Data collection system.

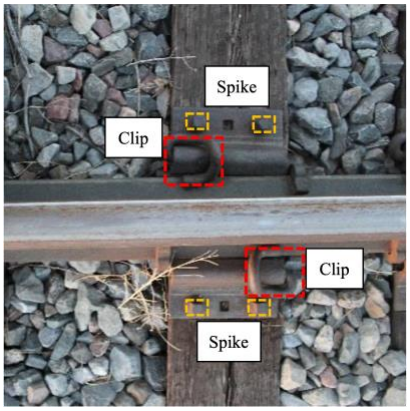
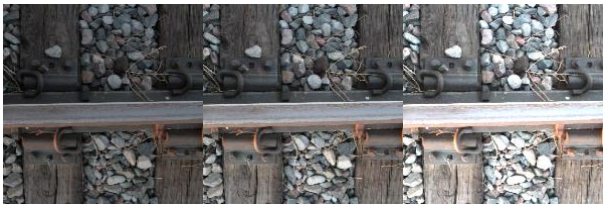


Figure 6. Collected image data



(a) -20% brightness (b) 0% (c) +20% brightness

Figure 7. Data augmentation.

4.2. Performance Metrics

For assessing our proposed method, standard performance metrics, including precision, recall, and mean Average

Precision (mAP), along with computational efficiency measures of FLOPs (Floating Point Operations) and inference time, are utilized.

Precision and recall in Eq. (10), indicate the proportion of correctly detected objects among all detected ones, and the ratio of correctly identified objects to all objects in the ground truth, respectively. True positive, false positive, and false negative are, respectively, denoted by TP, FP, and FN in the equation.

$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN} \quad (10)$$

Moreover, the mAP, in Eq. (11) , serves as an overarching measure of the model's performance across all object classes, with each class having its own Average Precision (AP) value that represents the area under the Precision-Recall (P-R) curve. We specifically use mAP@0.5 and mAP@0.5:0.95 to evaluate the accuracy of Intersection over Union (IoU) values in bounding box predictions, which are shown in Eq. (11). The Eq. (11) involves two classes: clips and spikes. For the evaluation metric mAP@0.5, bounding boxes with an Intersection over Union (IoU) greater than 0.5 are treated as positive predictions, indicating a significant overlap between the predicted and actual object location.

$$mAP = \frac{1}{C} * \sum_{i}^{C} AP_i \quad (11)$$

For mAP@0.5:0.95, as shown in Eq. (12), we average mAP values calculated at an increment of 0.05 within the IoU range of 0.5-0.95

$$mAP@0.5{:}\,0.95 = \frac{1}{N} * \sum_{i}^{0.95} mAP@i \quad (12)$$
$$(i = 0.5, 0.55 \ldots 0.95, N = 10)$$

4.3. Results

In the following section, the performance outcomes of the proposed AWL-NanoDet model will be analyzed and compared with other real-time, lightweight benchmark models These include the native NanoDet, along with YOLOv8-n and YOLOv8-s [22].

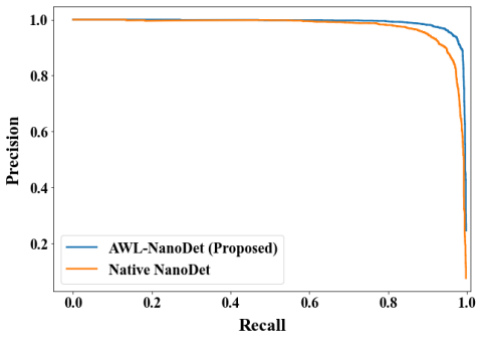
NanoDet, with its size under 2 Mb and computational cost of 1.52G FLOPs, is a compact model. In comparison, YOLOv8-n is larger in size (3.2 Mb) and more computationally intensive with 8.7G FLOPs. Meanwhile, YOLOv8-s is even larger with 28.6G FLOPs and 11.2M size. However, the scale of FLOPs across NanoDet, YOLOv8-n, and YOLOv8-s, is comparable in general, hence confirming the relevance of their comparative analysis.

The tabulated results in Table 1 provide a quantitative comparison in precision, recall, mAP, FLOPs, and inference time across various classes. The data indicates that our proposed AWL-NanoDet model outperforms the other models in most performance measures.

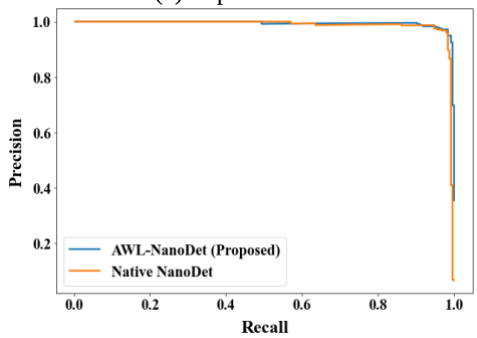
In evaluating the spike class, the native NanoDet model shows worse results, only reaching a recall rate of 92.2% and a precision of 93.0%. Although YOLOv8n achieves better results with a recall rate of 94.1% and a precision of 95.8 for the spike class, our AWL-NanoDet demonstrates superior detection featuring a recall rate of 95.2% and a precision of 96.1%. This corresponds to a notable 3% improvement over the native NanoDet. Although the recall rate of YOLOv8 is almost the same as our AWL-NanoDet, YOLOv8s has a FLOPs nearly 19 times larger than our AWL-NanoDet.

Furthermore, our model surpasses the native NanoDet by a significant 5.6% in AP@0.5:0.9, as shown in Table 1. When compared with the larger models like YOLOv8n and YOLOv8s, our model maintains its performance at the high level in the mAP@0.5 metrics and matches larger model in mAP@0.5:0.9. Considering that our AWL-NanoDet and the native NanoDet share the same model structure, this remarkable performance enhancement in both object classes is solely due to the AWL algorithm proposed in this study.

Figure 8 depicts Precision-Recall (P-R) curves for both native NanoDet and AWL-NanoDet, colored in orange and blue curves, respectively. The integration of AWL during training noticeably enhances the performance compared to the native NanoDet.



(a) Spike P-R curve

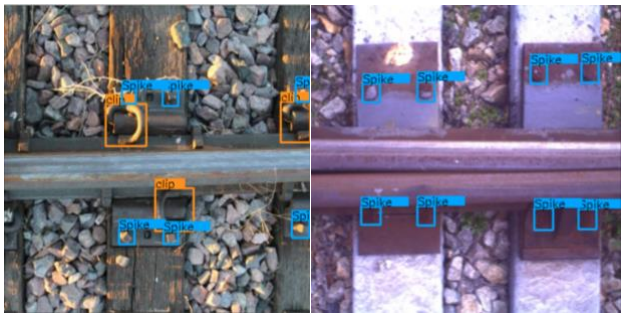


(b) Clip P-R curve

Figure 8. Precision-Recall (P-R) curve

In Figure 8 (a), the P-R curve of the spike class indicates a dramatic drop in precision for native NanoDet as recall increases. On the contrary, the drop in precision of AWL-NanoDet is much mild, maintaining a precision above 0.25 even at a recall close to 1, whereas native NanoDet is close to 0. This robust performance confirms AWL's effectiveness for detecting challenging object classes. Figure 8 (b), shows the P-R curve of the clip class, which similarly demonstrates that AWL-NanoDet slightly surpasses NanoDet. Similarly, our model's precision decays more slowly as recall increases, keeping precision above 0.35 even when the recall is near 1, which is notably higher than that of native NanoDet at the same recall level.

Figure 9. displays the image output from our model, employing the AWL algorithm under complex conditions, such as cluttered background or component blurring.



(a) (b)

Figure 9. AWL-NanoDet results

In Figure 9 (a), even though the arrangement of objects is dense, the model is still successful in detecting all the components. Similarly, in Figure 9 (b), despite the rusty spikes embedded in the background, the model continues to effectively identify all components.

Model	Class	Precision (%)	Recall (%)	mAP@0.5 (%)	mAP@0.5:0.95 (%)	FLOPs (G)	Inference time (ms)
AWL-Nanodet	All	**96.7**	**96.8**	**98.7**	71.5	**1.52**	**4.2**
	Clip	**97.2**	**98.4**	**99.0(AP)**	75.9(AP)		
	Spike	96.1	95.2	**98.4(AP)**	67.0(AP)		
Native-NanoDet	All	94.9	95.1	98.1	65.9	1.52	4.2
	Clip	96.8	98.0	98.2(AP)	69.7(AP)		
	Spike	93.0	92.2	97.9(AP)	62.1(AP)		
YOLOv8n	All	95.0	95.5	98.2	73.2	8.7	19.5
	Clip	94.2	97.0	98.6(AP)	78.9(AP)		
	Spike	95.8	94.1	97.8(AP)	67.0(AP)		
YOLOv8s	All	96.2	96.5	98.6	76.3	28.6	60.2
	Clip	95.1	97.1	98.9	81.7		
	Spike	97.2	95.9	98.3	70.8		

Table 1. Comparison of each models

5. CONCLUSION

This paper introduces AWL-NanoDet, a real-time rail track inspection model suited for edge devices. Its innovation lies in adapting the teacher-student guidance during training, altering the weight of the student loss based on the teacher and student model qualities, and eventually balancing the teacher and student losses. The model is tested on railroad inspection data and compared with other models in precision, recall, mAP, and FLOPs for performance benchmarking, and shows superior results. Future improvements will focus on refining the scoring and weight strategies, evaluating AWL-NanoDet on real-world platforms, and expanding its applications.

ACKNOWLEDGEMENT

This study, partly funded by the Federal Railroad Administration (FRA) under Contract No. 693JJ621C000011, utilized images from FRA's Track Component Imaging System. Thank Mr. Cameron Stuart from FRA for his valuable guidance. The views presented

herein are the authors' alone and do not reflect those of the funding agency.

REFERENCES

[1] RangiLyu., "NanoDet ," *https://github.com/RangiLyu/nanodet*.

[2] FRA, "Train accidents by cause form (form FRA F 6180.54)." https://safetydata.fra.dot.gov/OfficeofSafety/publicsite/Query/inccaus.aspx.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 1998.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun ACM*, vol. 60, pp. 84–90, 2017.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint*, 2014.

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

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[9] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *European conference on computer vision*, 2018, pp. 734–750.

[10] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *IEEE/CVF international conference on computer visio*, 2019, pp. 9627–9636.

[11] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint*, 2017.

[12] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[13] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint*, 2015.

[14] C. H. Nguyen, T. C. Nguyen, T. N. Tang, and N. L. Phan, "Improving object detection by label assignment distillation," in *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 1005–1014.

[15] D. Dais, I. E. Bal, E. Smyrou, and V. Sarhosis, "Automatic crack classification and segmentation on masonry surfaces using convolutional neural networks and transfer learning," *Autom Constr*, vol. 125, p. 103606, 2021.

[16] J. Zhang, S. Qian, and C. Tan, "Automated bridge crack detection method based on lightweight vision models," *Complex & Intelligent Systems*, pp. 1–14, 2022.

[17] Y.-J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, and O. Büyüköztürk, "Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, pp. 731–747, 2018.

[18] F. Guo, Y. Qian, and Y. Shi, "Real-time railroad track components inspection based on the improved YOLOv4 framework," *Autom Constr*, vol. 125, p. 103596, 2021.

[19] F. Guo, Y. Qian, Y. Wu, Z. Leng, and H. Yu, "Automatic railroad track components inspection using real-time instance segmentation," *Computer-Aided Civil and Infrastructure Engineering*, vol. 36, pp. 362–377, 2021.

[20] S. Li, X. Zhao, and G. Zhou, "Automatic pixel-level multiple damage detection of concrete structure using fully convolutional network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, pp. 616–634, 2019.

[21] https://cocodataset.org/#home.

[22] https://docs.ultralytics.com.

BIOGRAPHIES

Jiawei Guo earned his B.S. from Tianjin University of Technology and Education (2019), Tianjin, China, and his M.S. from the University of Southern California, CA, USA. He is now pursuing his Ph.D. of mechanical engineering with the University of South Carolina. His research interests are in computer vision and machine learning for engineering applications.

Sen Zhang has been pursuing his B.S. in computer science with the University of South Carolina since 2019. His research interests include deep learning and machine learning algorithms and its application to various engineering fields.

Yu Qian Yu Qian holds dual B.S. degrees from Huazhong University of Science and Technology and Wuhan University (2008), an M.S. from University of Kansas (2009), another M.S. and a Ph.D. from the University of Illinois at Urbana-Champaign (2013, 2014). He is currently an Associate Professor at the University of South Carolina. His research focuses on heavy haul and transit track structures, artificial intelligence, and infrastructure 4.0.

Yi Wang earned his B.S. and M.S. from Shanghai Jiao Tong University (1998, 2000), and his Ph.D. from Carnegie Mellon University (2005). He worked in R&D at CFD Research Corporation from 2005-2017. He is now an Associate Professor at the University of South Carolina. His research focuses on computational and data-enabled science and engineering, multi-fidelity surrogate modeling and autonomous systems.