American International University-Bangladesh (AIUB)

Faculty of Science & Technology (FST)

Assignment Title: CVPR Final Report

Supervised By

Dr. Debajyoti Karmaker

Submitted By (Group 3)

Semester: Spring 24-25		**Section: B**	**Contribution**
SN	Student Name	Student ID	
1	Nahid Hasan Nobil	22-46321-1	Introduction
2	Muntasir Maruf	22-46620-1	Methodology
3	Md. Mushfiqur Rahman Utsho	22-46602-1	Related Works
4	Md. Nazmul Islam Rahat	22-46323-1	Results & Discussion

Adaptive Monte-Carlo DropBlock for Improved Uncertainty Estimation in YOLOv8 Object Detection

Abstract:

Object detection models, while powerful, often produce overconfident predictions and lack the ability to quantify uncertainty, a critical requirement for deployment in safety-sensitive applications like autonomous driving. Monte-Carlo DropBlock (MC-DropBlock) has emerged as an effective technique to model epistemic uncertainty in convolutional neural networks by applying structured dropout at both training and test time. However, this method traditionally relies on a fixed, manually tuned dropout rate, which can be suboptimal for the complex training dynamics of modern detectors. A high rate can impede early learning, while a low rate may offer insufficient regularization in later stages. In this work, we propose Adaptive Monte-Carlo DropBlock, an enhancement which is applied with YOLOv8 architecture. Our approach replaces the static dropout probability with a dynamic rate that follows a cosine annealing schedule, gradually increasing from a small initial value to a larger final value as training progresses. This adaptive strategy allows the model to learn foundational features with minimal interference early on, while applying more aggressive regularization later to improve generalization and prevent overfitting. Which is backed by the comparison between image segmentation using custom food dataset without any dropout, with MC-DropBlock and Adaptive MC-DropBlock.

1. Introduction:

The rapid advancement of deep learning has positioned object detection as a cornerstone of modern computer vision, with models like YOLOv8 achieving remarkable speed and accuracy. These models are integral to numerous real-world systems, from automated surveillance to autonomous navigation. However, their deterministic nature presents a significant drawback. They are often incapable of expressing uncertainty. A standard detector may incorrectly classify an object with dangerously high confidence, posing a severe risk in applications where failures have critical consequences. Therefore, enabling these models to signal when they are uncertain is not just beneficial, but essential for their safe deployment.

Monte-Carlo Dropout (MC-Dropout) was introduced, demonstrating that applying standard dropout during both training and inference serves as a Bayesian approximation. For convolutional neural networks (CNNs), where standard dropout is less effective due to the spatial correlation of feature maps, DropBlock was developed to drop contiguous regions of features.

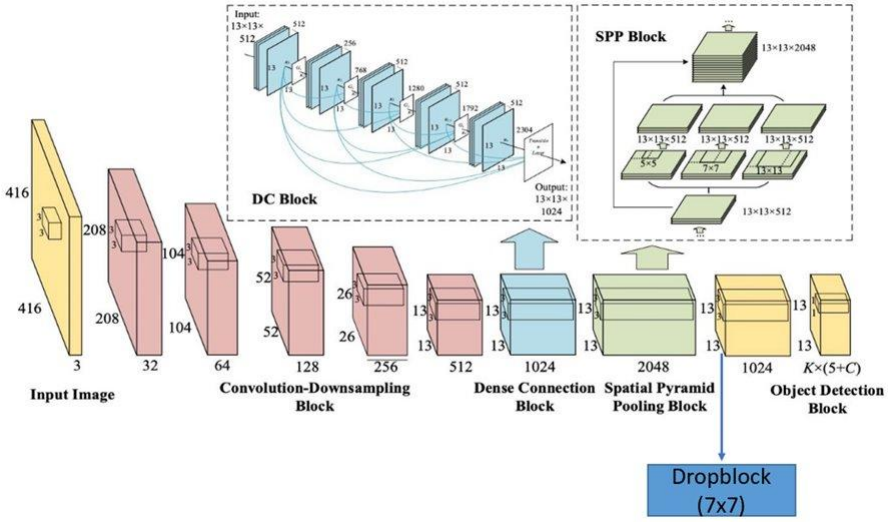

Figure 1: DropBlock Implemented in YOLOv4 [1]

Despite its success, the conventional MC-DropBlock approach suffers from a key limitation. Its reliance on a fixed dropout probability. This rate is a static hyperparameter that must be carefully tuned. If set too high, it can disrupt the learning process in the early epoch when the model is still learning fundamental features. If set too low, it may fail to provide adequate regularization as the model begins to overfit later in training. A scheduled increase in the dropout rate during training leads to better accuracy and more robust performance.

To address this deficiency, an Adaptive Monte-Carlo DropBlock is introduced. Which is a novel enhancement that allows the regularization strength to adapt to the training phase, starting low and smoothly increasing as training progresses. We hypothesize that this scheduled approach will foster more stable learning, lead to a better-calibrated model, and provide more meaningful uncertainty estimates.

The main contributions of this work are:

- The proposal of Adaptive Monte-Carlo DropBlock (AM-DropBlock), which integrates a dynamic, scheduled dropout rate into the DropBlock mechanism.
- A methodology for dynamically adjusting the dropout probability during training using a cosine annealing schedule to optimize regularization.

2. Related Work:

This paper proposes MC-DropBlock, a technique that applies DropBlock at both training and inference time to improve uncertainty modeling in deep learning models, especially YOLO-based object detectors. Unlike standard MC Dropout, which performs poorly on convolutional layers, MC-DropBlock drops contiguous regions of features, making it more effective. It captures both epistemic and aleatoric uncertainty (via Gaussian likelihood) and improves generalization, calibration, and OOD (out-of-distribution) robustness. Experiments on object detection, segmentation, and classification tasks (YOLOv4/v5, YOLACT, ResNet) show MC-DropBlock outperforms MC-Dropout and baseline models in uncertainty-aware vision tasks [1].

This paper introduces Box Stability Score (BoS), a novel, unsupervised metric to estimate object detector performance (mAP) on unlabeled test datasets. The key insight is that stable bounding boxes under feature dropout indicate better generalization. Using MC dropout, the authors perturb feature maps during inference and calculate BoS as the average IoU between matched original and perturbed bounding boxes. They demonstrate a strong correlation ($R^2 > 0.94$) between BoS and actual mAP across varied environments and detector types (e.g., RetinaNet, Faster R-CNN). BoS enables reliable label-free evaluation and outperforms existing confidence-based AutoEval methods in both vehicle and pedestrian detection tasks [2].

The paper investigates how to quantify uncertainty in deep learning models by applying Monte Carlo (MC) dropout to skip connection-based CNNs. The authors focus on a high-dimensional fluid flow regression problem in oil/gas reservoir modeling, using over 376,000 samples. MC dropout is used during both training and testing to estimate epistemic uncertainty while maintaining high accuracy. Results show strong performance with R^2 up to 0.9584 and low standard deviation (SD) values, proving MC dropout is both efficient and reliable for uncertainty quantification in deep learning models [3].

3. Methodology:

This implementation builds upon the MC-DropBlock framework. DropBlock is a form of structured dropout that zeroes out contiguous regions in a feature map rather than individual, random neurons. This is more effective for CNNs because it removes semantic information more robustly, forcing the network to learn more distributed and resilient features.

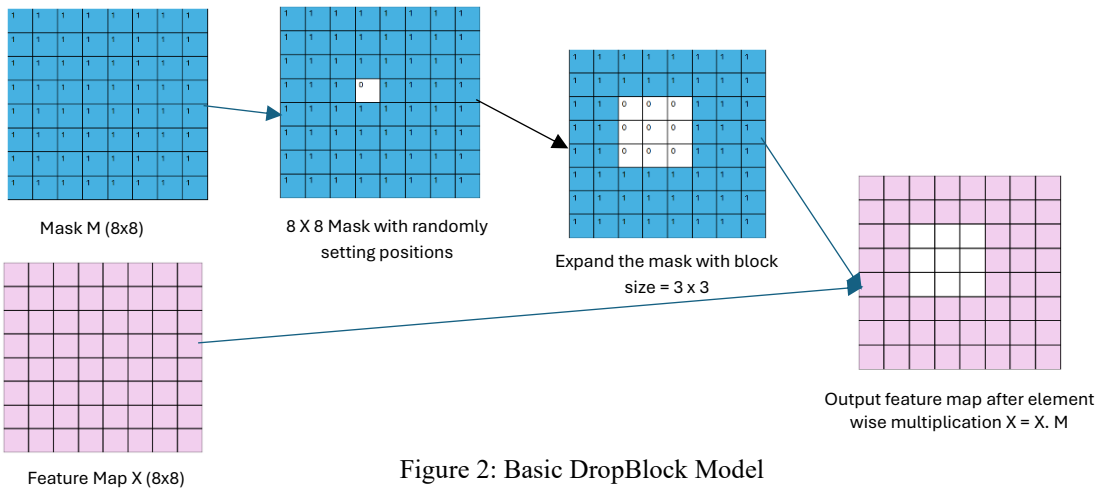

Figure 2: Basic DropBlock Model

This can be done by fixed dropout rate, which in turn used to calculate the gamma, which is a crucial calculated probability that controls how often a pixel is chosen as the center of a block that will be dropped. The factors like Feature Height (H), Weight (W) and Block Size (B) also used to determine the gamma.

```
B = self.block_size
H, W = x.shape[2], x.shape[3]
gamma = self.drop_prob * (H * W) / ((H - B + 1) * (W - B + 1)) * (1 / (B * B))
```

Figure 3: Setting the DropBlock probability

Proposed Improvements: Adaptive DropBlock with Cosine Scheduling

To enhance the performance a introduce a dynamic dropout schedule. Instead of keeping drop probability constant, we vary it over the epochs according to a cosine annealing schedule. This provides a smooth, gradual transition from a low initial dropout rate to a higher final rate.

The dropout probability for a given epoch is calculated using the following schedule function:

```
def cosine_schedule(epoch, total_epochs, start, end):
    cos_inner = math.pi * epoch / total_epochs
    return end - (end - start) * 0.5 * (1 + math.cos(cos_inner))
```

Figure 4: Setting the Adaptive Dropout schedule

The cosine_schedule function gradually transitions value from start to end over a set number of epochs using a cosine function. First, it converts the current epoch into a radian angle ranging from 0 to π, then applies the cosine function to produce a value that decreases smoothly from 1 to -1. This output is normalized to a (0-1 scale) using "0.5 * (1 + cos (1 + math. cos(cos_inner))", so it starts at 1 and ends at 0. Finally, this normalized value is scaled to the desired range (end - start) and subtracted from end to yield a smooth, non-linear transition from start to end over the training process.

Results:

This approach is used in a food segmentation task on a custom dataset, which is done via YOLOV8. The first run is implemented using the base model without any dropout. Then the experiment was repeated once using fixed drop probability and another time using adaptive drop probability.

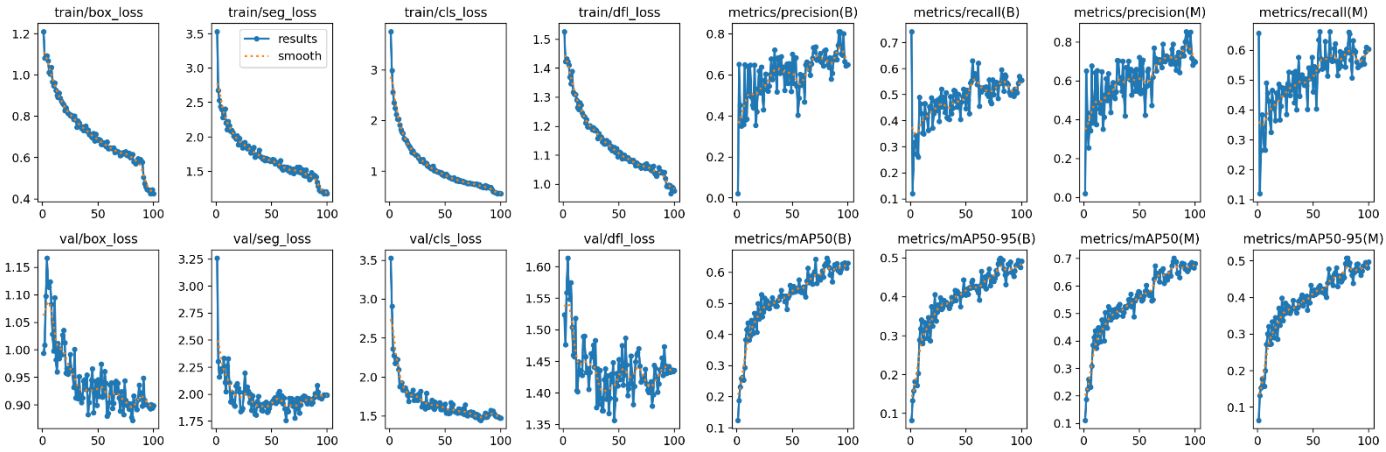

Figure 5: Performance metrics without any dropout (Training).

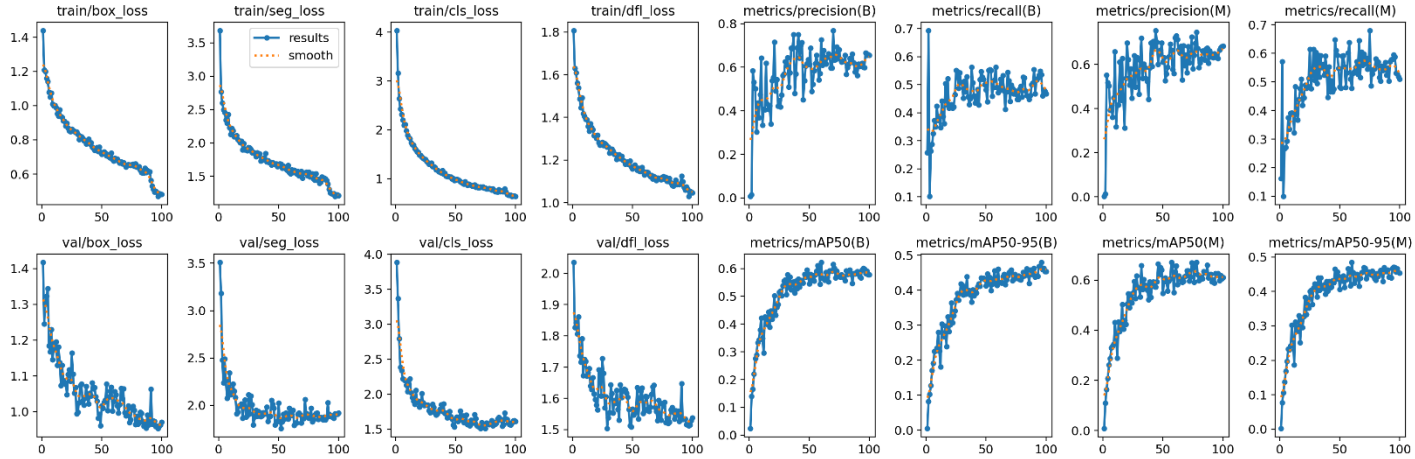

Figure 6: Performance metrics with fixed drop probability (Validation).

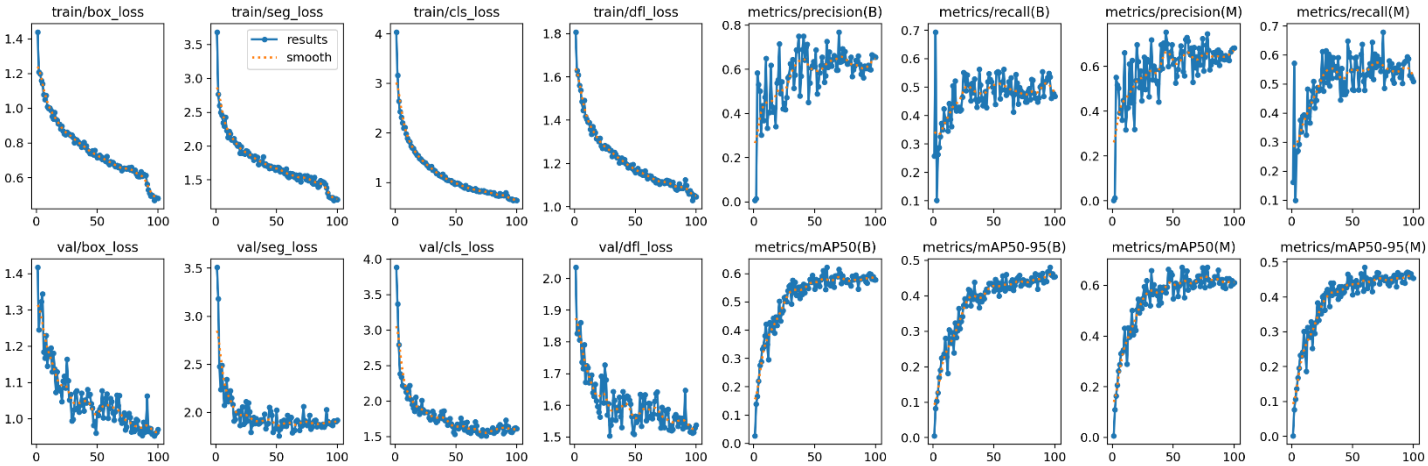

Figure 7: Performance metrics with adaptive drop probability (Validation).

Discussion and Comparison:

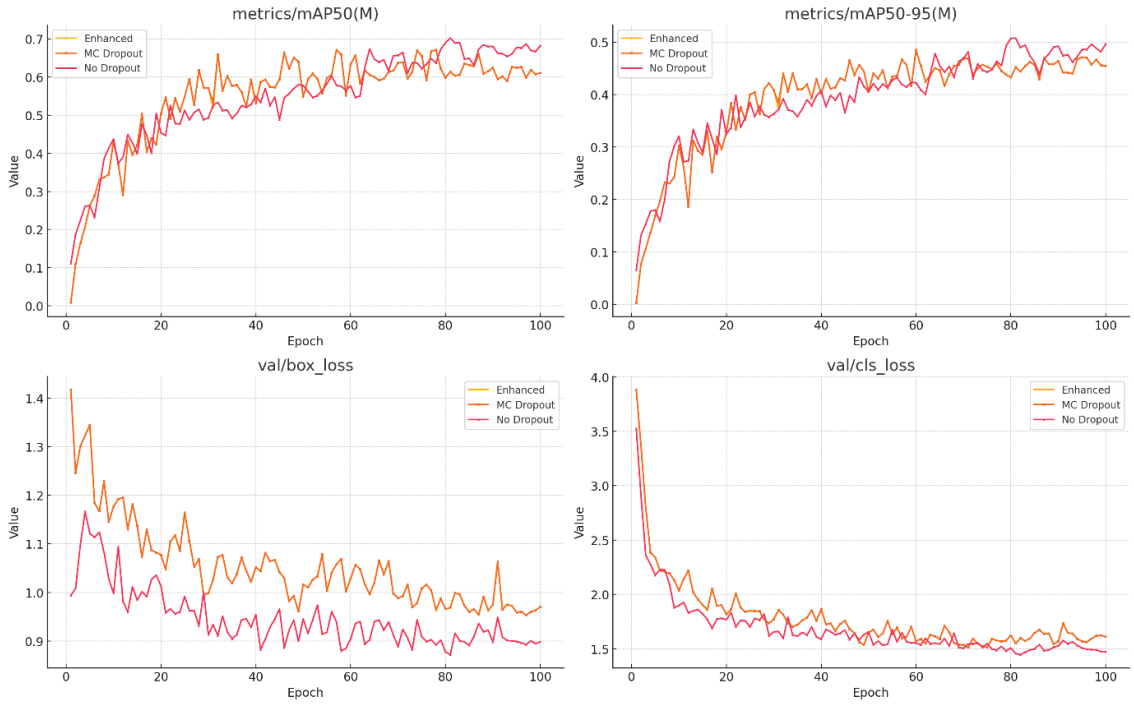

Figure 8: Performance metrics

According to the performance metrics it can be illustrated that mAP (mean Average Precision) scores are fluctuating, but a general upward trend can be noticed for the Adaptive (Enhanced) Dropout approach. But in case of validation losses the performance is not so impressive.

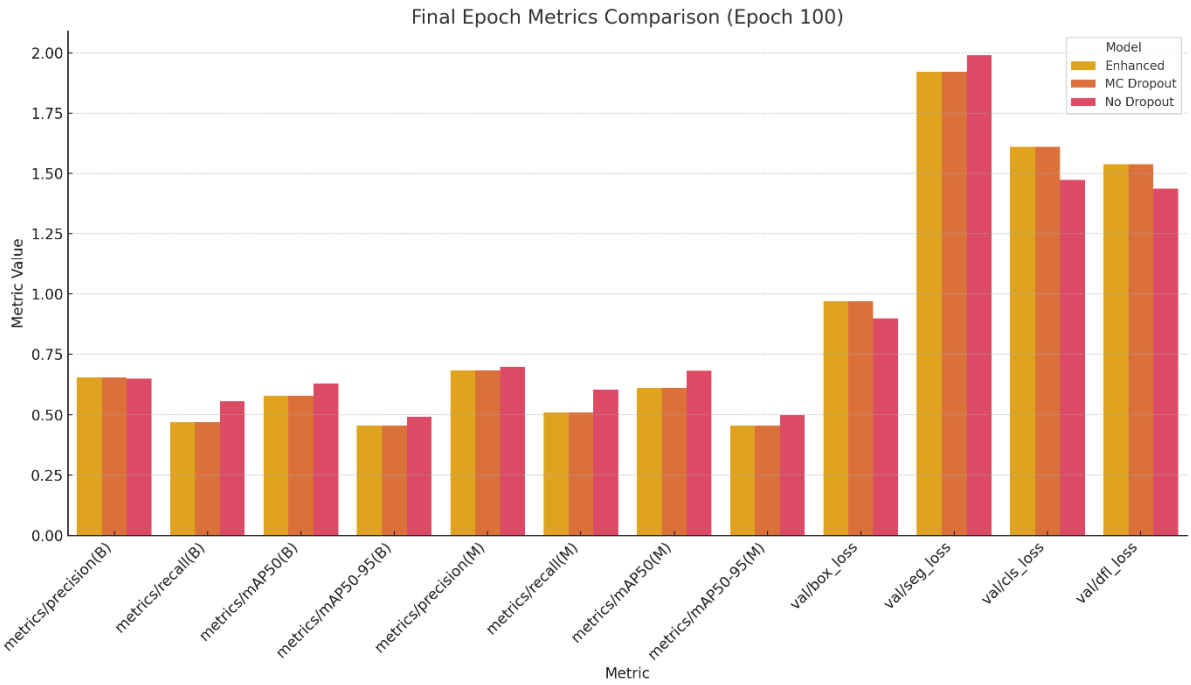

Figure 9: Metric values of the final Epoch

Here the model with no dropout slightly outperformed the dropout and enhanced model. This could be because of the base model's already super strong regularization techniques like batch normalization. Applying dropout might have been disrupting learning by disabled parts of the network at random during training, which harms performance when there is no overfitting. Additionally, MC Dropout adds noise during inference time to estimate uncertainty, which can degrade prediction sharpness and slightly reduce accuracy metrics. Overall, dropout probably added unnecessary complexity without conferring advantages under this well-regularized setup. And the small size of the custom data might have negatively impacted the performance.

Conclusion and Future work:

The results indicate that the base YOLOv8 model with no dropout achieved a slightly better results than MC Dropout as well as the enhanced DropBlock model, likely due to the fact that the base model already included robust regularization techniques such as batch normalization and residual connections. In this case, additional stochastic regularization may have disrupted learning by dropping parts of the network randomly, especially as the training dataset was fairly modest in size. Moreover, MC Dropout's noise at inference time, while helpful for uncertainty estimation, can reduce prediction sharpness and possibly harm accuracy. Future work would need to examine the benefit of these methods on larger or noisier datasets, use uncertainty in downstream decisions, and experiment with attention-guided or layer-wise adaptive dropout for targeting regularization where it is most beneficial.

Reference:

1. Monte Carlo DropBlock for Modelling Uncertainty in Object Detection
2. Bounding Box Stability against Feature Dropout Reflects Detector Generalization across Environments
3. Applying Monte Carlo Dropout to Quantify the Uncertainty of Skip Connection-Based Convolutional Neural Networks Optimized by Big Data