INFRARED DOMAIN ADAPTATION WITH ZERO-SHOT QUANTIZATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

BURAK SEVSAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF MULTIMEDIA INFORMATICS

SEPTEMBER 2024

Infrared Domain Adaptation with Zero-Shot Quantization

submitted by **BURAK SEVSAY** in partial fulfillment of the requirements for the degree of **Master of Science in Modeling and Simulation Department, Middle East Technical University** by,

Prof. Dr. Banu Günel Kılıç _____
Dean, **Graduate School of Informatics**

Assoc. Prof. Dr. Elif Sürer _____
Head of Department, **Modeling and Simulation**

Assoc. Prof. Dr. Erdem Akagündüz _____
Supervisor, **Modeling and Simulation, METU**

Examining Committee Members:

Prof. Dr. Alptekin Temizel
Modeling and Simulation, Middle East Technical University _____

Assoc. Prof. Dr. Erdem Akagündüz
Modeling and Simulation, Middle East Technical University _____

Assist. Prof. Dr. İrem Ülkü
Computer Engineering, Ankara University _____

Date: 03.09.2024

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Burak Sevsay

Signature :

ABSTRACT

INFRARED DOMAIN ADAPTATION WITH ZERO-SHOT QUANTIZATION

Sevsay, Burak
M.S., Department of Modeling and Simulation
Supervisor: Assoc. Prof. Dr. Erdem Akagündüz

September 2024, 54 pages

Object detection models are gaining popularity in daily life and industry, increasing the demand for real-time computation of these models. Model compression is an essential technique to achieve faster inference and smaller footprints. Quantization is a widely used model compression technique, reducing bit-width and enhancing efficiency at the cost of quantization error. Methods like post-training quantization and quantization-aware training require training data to minimize this error. However, training data may be inaccessible due to privacy concerns in applications such as surveillance and autonomous driving. In such cases, zero-shot quantization becomes necessary, as it applies quantization without the need for training data. Additionally, infrared imagery, which is more resilient to illumination and weather conditions, is often a part of these applications. To the best of our knowledge, zero-shot quantization in the infrared domain has not been investigated before. This thesis adapts batch normalization statistics-based zero-shot quantization for the infrared domain. This method aims to generate synthetic data by utilizing batch normalization statistics. We thoroughly investigated the data generation process to achieve optimal results for YOLOv8 and RetinaNet. For infrared adaptation, we fine-tuned models that were pretrained on RGB images using infrared images. The evaluation is based on comparing zero-shot quantization results with those from both full-precision models and post-training quantization. Additionally, we examined the effect of model size on zero-shot quantization. Our results show that batch normalization statistics-based zero-shot quantization is more effective in the infrared domain and is an essential method when training data is unavailable.

Keywords: Data-free Quantization, Infrared Domain, Object Detection

ÖZ

VERİ GEREKTİRMEYEN NİCELEMENİN KIZILÖTESİ ALANA UYARLAMASI

Sevsay, Burak
Yüksek Lisans, Modelleme ve Simülasyon Bölümü
Tez Yöneticisi: Doç. Dr. Erdem Akagündüz

Eylül 2024, 54 sayfa

Nesne algılama modelleri günlük yaşamda ve endüstride popülerlik kazanmakta ve bu modellerin gerçek zamanlı hesaplanmasına yönelik talep artmaktadır. Model sıkıştırma daha hızlı çıkarım ve daha küçük ayak izleri elde etmek için önemli bir tekniktir. Niceleme, bit genişliğini azaltan ve niceleme hatası pahasına verimliliği artıran yaygın olarak kullanılan bir model sıkıştırma tekniğidir. Eğitim sonrası niceleme ve niceleme farkındalıklı eğitim gibi yöntemler bu hatayı en aza indirmek için eğitim verilerine ihtiyaç duyar. Fakat, gözetim ve otonom sürüş gibi uygulamalarda gizlilik endişeleri nedeniyle eğitim verilerine erişim mümkün olmayabilir. Bu gibi durumlarda, eğitim verilerine ihtiyaç duymadan nicelemeyi uyguladığı için veri gerektirmeyen (sıfır atışlı) niceleme gerekli hale gelir. Ek olarak, aydınlatmaya ve hava koşullarına daha dayanıklı olan kızılötesi görüntüleme genellikle bu uygulamaların bir parçasıdır. Bildiğimiz kadarıyla, kızılötesi alanda veri gerektirmeyen niceleme daha önce araştırılmamıştır. Bu tez, kızılötesi alanı için toplu normalizasyon istatistiklerine dayalı veri gerektirmeyen nicelemeyi uygular. Bu yöntem, toplu normalizasyon istatistiklerini kullanarak sentetik veri üretmeyi amaçlar. YOLOv8 ve RetinaNet için en iyi sonuçları elde etmek amacıyla veri üretme sürecini kapsamlı bir şekilde araştırdık. Kızılötesi adaptasyon için, RGB görüntüyle önceden eğitilmiş modelleri kızılötesi görüntülerle tekrar eğittik. Değerlendirme, veri gerektirmeyen niceleme sonuçlarını hem tam hassasiyetli modellerden hem de eğitim sonrası nicelemeden elde edilen sonuçlarla karşılaştırmaya dayanmaktadır. Ek olarak, model boyutunun veri gerektirmeyen niceleme üzerindeki etkisini inceledik. Sonuçlarımız, toplu normalizasyon istatistiklerine dayalı veri gerektirmeyen nicelemenin kızılötesi alanda daha etkili ol-

duğunu ve eğitim verilerinin bulunmadığı durumlarda gerekli bir yöntem olduğunu göstermektedir.

Anahtar Kelimeler: Verisiz Niceleme, Kızılötesi, Nesne Algılama

To my dear family

ACKNOWLEDGMENTS

First of all, I want to express my gratitude to my advisor, Assoc. Prof. Dr. Erdem Akagündüz. His continuous support and invaluable guidance have been a driving force for me throughout this research. His encouragement and constructive feedback have shaped this thesis.

I am also sincerely thankful to the members of my thesis jury, Prof. Dr. Alptekin Temizel and Assist. Prof. Dr. İrem Ülkü. Their time, effort, and thoughtful critiques have greatly contributed to the refinement of this work.

I would like to express my thanks to the Kuartis family. Their resources and the supportive environment provided me with the motivation to pursue this research. I would also like to thank The Scientific and Technological Research Council of Turkey (TÜBİTAK) for supporting researchers and their grants for my M.S. studies.

Last but not least, I would like to thank my family for their unconditional love, patience, and understanding. Their unwavering support and belief in me have been my greatest source of strength throughout this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

PTQ	Post Training Quantization
QAT	Quantization Aware Training
ZSQ	Zero-Shot Quantization
BNS	Batch-Normalization Statistics
SSD	Single Shot MultiBox Detector
R-CNN	Region-based Convolutional Neural Network
ReLU	Rectified Linear Unit
STE	Straight-Through Estimator
YOLO	You Only Look Once
YOLOv8s	YOLO version 8 small model
YOLOv8m	YOLO version 8 medium model
YOLOv8l	YOLO version 8 large model
MS-COCO	Microsoft Common Objects in Context
IR	Infrared
ADAS	Advanced Driver-Assistance System

CHAPTER 1

INTRODUCTION

Image sensors have been widely used in industry and daily life, and their use is growing daily. Neural network models, which have proven their success in computer vision, have significantly increased the capabilities of systems using image sensors. This success of neural networks has contributed to the development of many systems, including smartphones, autonomous vehicles, and surveillance systems.

The widespread use of neural networks for computer vision poses some challenges, including real-time computation and edge device usage. These challenges highlight the importance of efficient hardware deployment and model compression techniques. Quantization [1, 2, 3, 4, 5] is one of the most popular choices for model compression, but there are also other model compression techniques such as model pruning [6, 7, 8], knowledge distillation [9, 10, 11, 12, 13] and efficient neural model architecture design [14, 15, 16]. Additionally, these techniques can be combined to improve model compression [17, 18].

Quantization has become an important technique for deploying neural networks due to the increasing demand for faster computations and reduced memory usage. However, reducing precision to 8-bit or lower introduces significant quantization error, resulting in a noticeable decrease in accuracy. Post Training Quantization (PTQ) and Quantization Aware Training (QAT) are the two most common techniques used to apply quantization while minimizing accuracy loss. PTQ involves calibration with training data to minimize quantization error. The calibration process determines the clipping range for quantization, which is one of the key quantization parameters. On the other hand, QAT proposes retraining the quantized model with training data to recover the accuracy drop. However, in many cases, training data may not be available due to privacy concerns. PTQ and QAT are not feasible in these cases, so data-free quantization techniques are required.

An alternative method to quantization with training data is dynamic quantization [19, 20]. The method suggests determining the clipping range during the inference of each input data. However, it may not always be feasible for real-time requirements because of the computational overhead. On the other hand, static quantization determines the clipping range during the quantization process once. However, capturing the optimal clipping range in static quantization requires training data, which is essentially PTQ.

Zero-shot quantization [21, 22, 12, 23, 24] addresses this problem with data-free solutions. One of the most popular methods for zero-shot quantization is synthetic data generation to determine the optimal clipping range. Batch normalization statistics of a model can be used to generate synthetic data. Losses in this technique are based on the difference between statistics stored in the batch normalization layer and the statistics of the previous layer's output. The loss is propagated to the input image to make it match the internal statistics of the model. The resulting image can be used in calibration instead of training data.

Infrared domain applications often require zero-shot quantization due to the sensitive nature of datasets used in medical, security, and autonomous driving. We investigated the performance of zero-shot quantization in the infrared domain. First, we applied zero-shot quantization to object detection models pretrained on the MS-COCO dataset as a baseline experiment. We then fine-tuned these pretrained models using the FLIR ADAS dataset for the object detection task. The parameters of zero-shot quantization were adjusted to work with these fine-tuned models. Comparing the zero-shot quantization performance of the pretrained and fine-tuned models demonstrated the effectiveness of zero-shot quantization for thermal imagery.

Additionally, we compared the performance of zero-shot quantization with post-training quantization on RetinaNet and YOLOv8 models (small, medium, and large). Since YOLOv8 models include an additional image normalization processing step, we adapted the data generation process for effective zero-shot quantization of these models. We also examined the performance of using only the mean statistic versus both mean and standard deviation statistics in the loss function.

Our results showed that zero-shot quantization is more effective on models trained with infrared data since batch normalization layer channels are not diverse as in RGB-trained models. Also, our experiments showed that increasing model size decrease the performance of zero-shot quantization because of increasing number of batch normalization layers. We analyzed and demonstrated that generated data can effectively represent the training data for quantization performance. Thus, zero-shot quantization emerges as an important technique when training data is not available, especially in the infrared domain.

1.1 Problem Definition and Motivation

As the use of edge devices grows and model sizes increase to enhance performance, the need for neural network quantization becomes more critical. Zero-shot quantization, a significant branch of quantization, allows models to be quantized even in the absence of training data. Various methods have been proposed for zero-shot quantization. However, to the best of our knowledge, its performance in the infrared domain has not yet been explored.

Object detection is a widely used task in both industrial and everyday scenarios. Due to the prevalence of RGB cameras, most algorithms and datasets for object detection

are designed for three-channel RGB imagery. However, object detection in the infrared domain is necessary and rapidly expanding. Unlike RGB cameras, which are susceptible to illumination changes and may struggle in low-light or harsh weather conditions, infrared imagery provides more consistent results. Consequently, object detection algorithms must be adapted for infrared imagery. Our work addresses this need by adapting state-of-the-art object detection models to the infrared domain to apply and investigate zero-shot quantization.

Applications involving infrared imagery often deal with sensitive data, such as in surveillance and medical contexts. Additionally, these applications typically require edge device deployment with real-time processing capabilities. These factors highlight the importance of investigating zero-shot quantization in the infrared domain, which is the primary focus of our work.

1.2 Research Questions

Batch normalization statistics-based methods are popular in zero-shot quantization techniques. Are these methods applicable to the infrared domain?

Is there a relationship between model size and the performance of batch normalization statistics-based zero-shot quantization?

How can state-of-the-art YOLO models be effectively utilized for zero-shot quantization?

What are the differences between models trained on RGB imagery and those trained on infrared imagery in the context of zero-shot quantization?

1.3 Objectives of the Study

Our work aims to explore the potential of batch normalization statistics-based zero-shot quantization in the infrared domain. For this purpose, we propose a framework for zero-shot quantization of models trained on infrared imagery, which includes fine-tuning RGB-trained models with infrared imagery.

We experiment with different parameters of data distillation (synthetic data generation) to efficiently adapt models to the infrared domain. Additionally, YOLOv8 models include extra image normalization processing that significantly impacts the quantization performance of distilled data. Therefore, we investigate and propose a solution to adapt the YOLO series for zero-shot quantization.

Two main comparisons are planned in the experiments to demonstrate the effectiveness of zero-shot quantization in the infrared domain. The first comparison is between zero-shot quantized models and full-precision models. However, this comparison alone is not sufficient to reveal the full potential. Therefore, we also compare

zero-shot quantization with post-training quantization conducted using training data. This comparison is essential since the data distillation process is intended for situations where training data is unavailable. Lastly, model size is another important factor influencing zero-shot quantization performance, which we also investigate in the experiments.

1.4 Contributions of the Study

The contributions of this work can be summarized as follows:

- Zero-shot quantization in the infrared domain is investigated for the first time.
- The effect of model size on batch normalization statistics-based zero-shot quantization is explored.
- Previous zero-shot quantization works lack experimentation with current state-of-the-art object detection models like YOLOv8. Our work highlights the challenges of applying zero-shot quantization to YOLOv8 and offers solutions to address these issues.
- A comparison between PTQ and zero-shot quantization is thoroughly investigated to evaluate the true performance of zero-shot quantization.
- Previous batch normalization statistics-based zero-shot quantization methods utilized both mean and standard deviation for the loss function. The experiments in our work showed that the standard deviation has a negligible or negative impact on accuracy.

1.5 Organization of the Thesis

The organization of the thesis is as follows:

- Chapter 1: Provides a brief introduction to the conducted work, explaining the motivation and contribution of the study.
- Chapter 2: Covers the related background for neural network quantization, zero-shot quantization, and object detection in the infrared domain.
- Chapter 3: Describes the methodology of this study, including the basics of the applied quantization and the zero-shot quantization algorithm.
- Chapter 4: Details the experimental setup, including the utilized neural network models, the dataset, evaluation metrics, and the implementation details of the fine-tuning method and zero-shot quantization.

- Chapter 5: Presents the results of the study.
- Chapter 6: Concludes the thesis and suggests possible future work.

CHAPTER 2

RELATED WORK

In this chapter, the background of the thesis and related works are provided. Firstly, quantization fundamentals are introduced. All parameters for basic quantization and advanced techniques to improve quantization accuracy are discussed. Then, zero-shot quantization techniques and object detection in the infrared domain are explained.

2.1 Neural Network Quantization

2.1.1 Hardware

Besides the accuracy performance of a neural network, deployment performance factors such as memory consumption, inference speed, and power efficiency are important parameters for effective and feasible usage of neural networks.

The performance of neural networks largely depends on the deployment hardware. Because operations like convolution and matrix multiplication can be parallelized, GPUs are commonly used for deploying neural networks. Additionally, specialized hardware such as TPUs (Tensor Processing Units) or NPUs (Neural Processing Units) can be utilized for efficient deployment. Many of these hardware options support 8-bit integer (int8) operations, allowing tensor operations to be up to 16 times faster compared to 32-bit floating point (fp32) operations. Using int8 precision also reduces memory bandwidth usage and improves cache utilization [25]. Therefore, quantizing neural networks to the lower bit precision can significantly enhance efficiency if the hardware is compatible.

2.1.2 Model

Efficient deployment of a neural network model requires a small memory footprint, which depends on the number of parameters. Increasing the number of model parameters may enhance the learning capability of the model. However, learning capability also depends on other factors like model architecture and training parameters. Object detection is the main focus of this thesis, so popular object detection models are compared and benchmarked according to the number of parameters.

Object detection is a computer vision task that aims to identify and locate objects within an image. Several models have been developed and tested on common datasets like COCO, comparing their accuracy, speed, and efficiency. Two-stage object detection models, such as Fast R-CNN [26] and Faster R-CNN [27], are one of the the early architectures in the field. These models have one component that generates region proposals and another that predicts the object class for each proposal and refines the bounding box coordinates. Fast R-CNN introduced RoI popling to share computation across different region proposals within a single forward pass. Then, Faster R-CNN introduced the Region Proposal Network (RPN) to share convolutional features with the detection network and predicts region proposals simultaneously. Two-stage models are inherently slow, even though models like Faster R-CNN propose faster computation methods.

Single-stage object detection models became popular due to their faster computation capabilities compared to two-stage detectors. The Single Shot MultiBox Detector (SSD) [28] is one of the early single-stage detectors, offering much faster performance than two-stage detectors, but with lower accuracy. RetinaNet [29] introduced Focal Loss to address class imbalance, focusing on hard samples by down-weighting the loss assigned to well-classified examples. RetinaNet achieves accuracy close to that of Fast R-CNN while providing better speed performance.

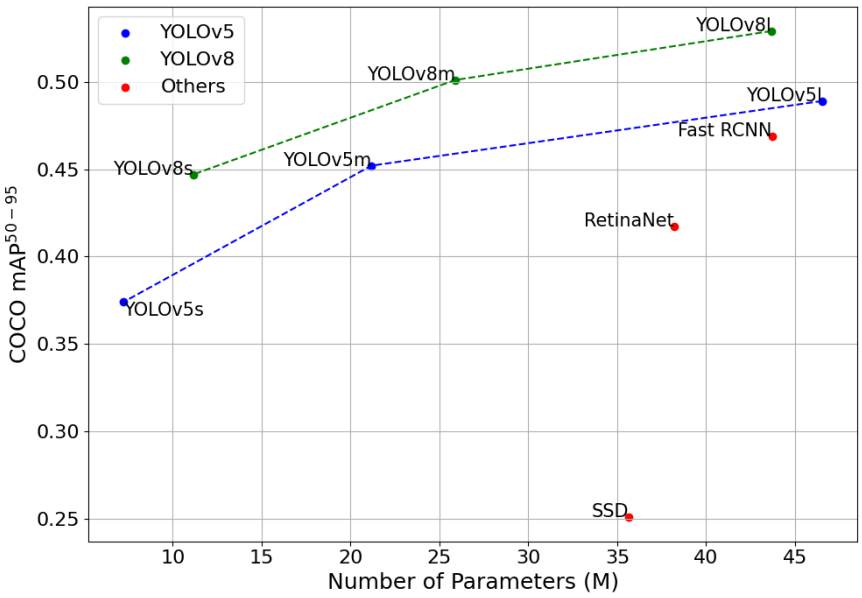
The YOLO (You Only Look Once) series represents state-of-the-art model architectures. The first version of the series [30] was published in 2016. Two years later, YOLOv3 [31] utilized a Feature Pyramid Network with residual blocks, which increased performance, including a higher capability for small object detection. Each version of YOLO enhanced performance through changes in architecture, training strategies, and augmentation techniques. YOLOv5 [32] and YOLOv8 [33], published by Ultralytics, further increased performance. They became popular due to their simplicity in training and deployment and their higher accuracy results. YOLOv5 introduced automated hyper-parameter optimization and architectural changes, including Enhanced PANet, which aims to enhance feature fusion and propagation for better multi-scale detection. In YOLOv8, besides an enhanced architecture and augmentation techniques, support for various computer vision tasks, including detection, segmentation, and classification, was introduced.

The number of model parameters is directly related to model size. Increasing the number of parameters can lead to longer inference time and potentially better accuracy because of increasing learning capability. However, accuracy and inference time do not always increase with the number of parameters. Model architecture is also one of the main parameters for both accuracy and inference time.

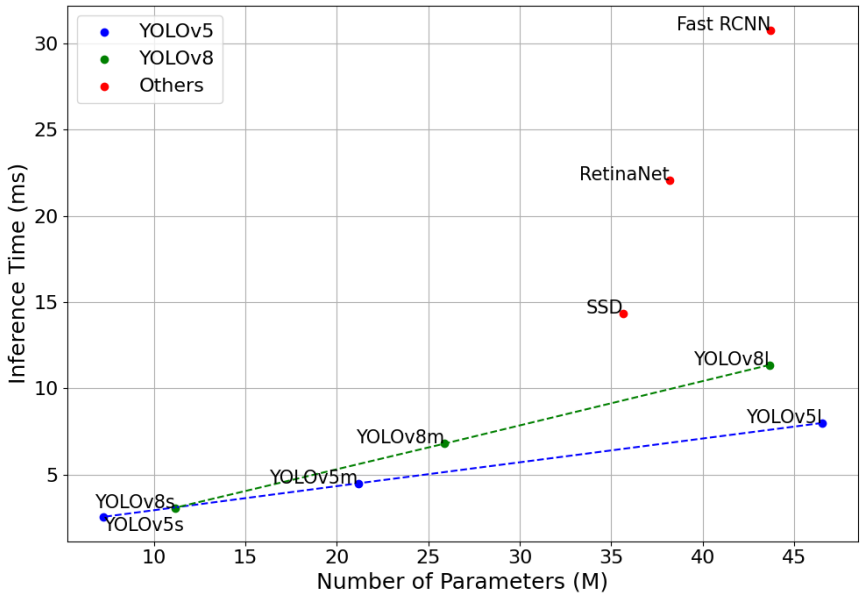
To gain insights into the impact of the number of parameters on accuracy and speed performance, pretrained models including RetinaNet (v2 with ResNet-50 backbone), SSD (with VGG16 backbone), Faster-RCNN (v2 with ResNet-50 backbone) from PyTorch [34], and the PyTorch implementations of YOLOv5 and YOLOv8 from Ultralytics are benchmarked. The benchmark is illustrated in Figure 1. An Nvidia RTX 3090Ti GPU is utilized for this benchmark, with a batch size of 64.

Comparing YOLOv5 and YOLOv8 models separately, it can be observed that increasing the number of parameters increases model accuracy and inference time due to their identical model architecture. However, older models, labeled as "other models" in Figure 1, perform with lower accuracy and higher inference times compared to YOLO models. Meanwhile, comparing older models among themsleves validates the impact of the number of parameters on accuracy and speed.

Even though newer models provide higher accuracy and speed, there is still a trade-off between accuracy and speed for similar model architectures. To achieve higher accuracy while meeting real-time requirements, one of the most effective methods is using lower precision, known as quantization.



(a) Parameters vs COCO mAP$^{50-95}$



(b) Parameters vs Inference Time

Figure 1: Performance Analysis of Different Neural Network Models

2.2 Quantization Fundamentals

Quantization is a technique used to reduce the size and computational requirements of neural networks by converting the weights and activations from higher precision

(FP32, FP16) to lower precision (Int8, Int4). Int8 quantization is commonly used due to its balance between performance and accuracy, while Int4 offers further compression at the potential cost of some accuracy. Also, instead of reducing the bit precision of each parameter separately, vector and product quantization can be applied by grouping weights into vectors and approximating them to reduce precision collectively [35]. In addition to vector quantization, product quantization partitions the weight matrix and quantize each subspace separately, providing finer control over precision and memory usage. However, both vector and product quantization degrades the accuracy performance because of the grouping approximation.

The basic quantization formula is given in Equation 1, where q is the quantized integer value and x is the original floating-point value. Weights and activations are scaled and shifted to fit within a specified bit-width. There are three parameters for performing quantization: the scale factor s, the zero-point z, and the bit-width k. The formula for the scale factor is given in Equation 2. The scale factor maps the parameters to $(2^k - 1)$ intervals in the range of [a,c] for k-bit representation. The interval length varies according to the choice of uniform or non-uniform quantization.

$$q = \left\lfloor \frac{x}{s} \right\rfloor + z \tag{1}$$

$$s = \frac{a - c}{2^k - 1} \tag{2}$$

Non-uniform quantization adapts the intervals to be smaller in dense parameter regions and larger in sparse regions, offering higher representational accuracy. However, it requires an additional look-up table for non-uniformity, which brings considerable overhead. On the other hand, uniform quantization uses uniform intervals to quantize the parameters. It is a simple and effective method to apply. Therefore, uniform quantization proves to be more resource-efficient in terms of memory footprint and computation speed. The scaling factor formula for uniform quantization is shown in Equation 2.

Clipping range is a core part of the scaling factor. The basic clipping range can be determined between the minimum and maximum of the parameters. In min/max clipping, clipping range can be represented as [a,c] where r is real-valued tensor, a = min(r), c = max(r). Another method is the use of percentiles. Instead of the minimum and maximum values, the i-th maximum and minimum values are used to determine the clipping range. Min/max clipping may increase the quantization interval length because of outlier parameters. To better representation for most of the parameters, the outliers can be clipped. In that case, percentile clipping results in better representation. Figure 2 represents the comparison of the min/max clipping and percentile clipping for the second convolution layer of pretrained RetinaNet. Since there are a small percentage of outlier parameters, narrow clipping range results in higher representation capacity with lower bit-width. In addition to these methods, there are other methods like entropy calculation, which minimize the error between

real values and quantized values. Because of simplicity of percentile calculation, it is an effective way to calculate clipping range which is referred to as calibration.

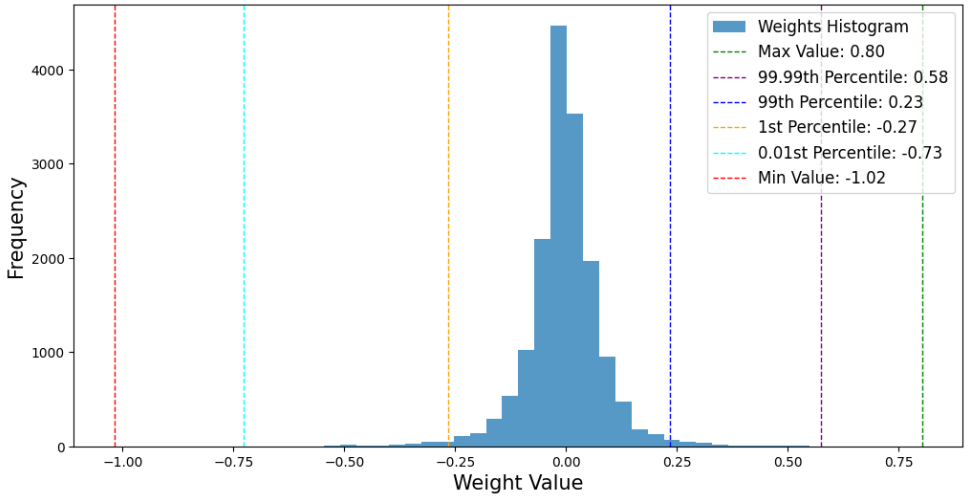


Figure 2: Histogram of The Second Convolutional Layer Weights

Another choice for quantization is the use of a symmetric or asymmetric quantization scheme. Asymmetric quantization can improve accuracy for data that is not zero-centered. In symmetric quantization, the zero-point z is fixed to zero. This simplifies the quantization process but performs worse in representing the data range.

Granularity is also an important aspect of quantization. The clipping range and scaling factor can be determined layer-wise, group-wise, or channel-wise, which is referred to as the granularity choice. Smaller groups for finding quantization parameters result in better representation while increasing computation overhead. Therefore, granularity should be determined wisely to balance fast computation and better representation.

After determination of the basic quantization parameters, another important choice is calculation time of clipping range. Clipping range can be determined dynamically in the inference time or statically during quantization. Dynamic quantization results in higher accuracy than static quantization but computing the clipping range for each input is not always a feasible solution for real-time requirements. Dynamic clipping range computation slows down the inference process, which limits the main benefit of quantization. On the other hand, the clipping range is determined only once in static quantization. It is a better option to reduce inference time and memory footprint. However, training data is required to capture the most suitable clipping range in static quantization.

Quantization limits can be pushed with binary neural networks [36] and ternary neural networks [37] which may be beneficial for efficient hardware like FPGA deployment of neural networks. However, binary or ternary quantization mostly need change in architectural design and training parameters including loss function.

Quantization of weights are simpler than quantization of activation functions because of non-linearity and dynamic range of activation functions. Some activation functions are easier to quantize because of the bounded output range. Sigmoid function with 0 to 1 range, and Tanh function with -1 to 1 range make easy to determine quantization parameters. However, activation functions with unbounded output like ReLU function results in high dynamic range which is hard to quantize without performance loss. While ReLU function has comparable benefit from quantization since it zeros out negative values, activation functions like leaky and parametric ReLU requires careful consideration during quantization because of highly dynamic range of output. There are some techniques like PACT (Parameterized Clipping Activation Function) [38] which optimize the clipping parameter during training to quantize activation functions more effectively.

Quantization error may vary across different layers. Therefore, forcing some layers which generate high quantization error to stay in high precision can increase accuracy performance of the quantization. The degree of how quantization of a layer affects the accuracy performance can be named as quantization sensitivity. There are quantization sensitivity analysis techniques like in [39, 21] to measure quantization sensitivity of each layer and optimize the trade-off between quantization error and model size. This analysis enables usage of mixed precision if the hardware is compatible. In that case, some layers operates at higher bit-precision to higher accuracy performance. [40] claims that the optimal quantization policy of a model in a specific hardware may not be optimal for the same model on the another hardware. Therefore, they simulate the target hardware and utilize a reinforcement learning agent that get feedback signal from the hardware to optimize the bit-width of each layer.

There are different needs and approaches to simulate the quantization. In the first approach, model parameters are stored in low precision to decrease the memory footprint but the operations are calculated in higher precision [41]. This approach requires dequantization operator before each calculation. It does not benefit quantization, and also add extra computation load for dequantization just for lower memory footprint. Another approach is simulating quantization if the hardware does not support the target bit precision. [42] introduce that simulating low bit precision by restring the range of integer values can reduce power consumption in neural network inference.

2.3 Fine-Tuning Methods

Quantizing neural network weights and activations to a lower precision introduces quantization error. There are fine-tuning methods to minimize this error for better accuracy. Post-Training Quantization (PTQ) is one of the most popular techniques because it doesn't require retraining. Calibration, which is explained in section 2.2, is the core part of PTQ. Calibration uses training data to estimate the range of the weights and activations for quantization. Labels are not needed in the calibration because retraining is not required.

Quantization Aware Training (QAT) is another popular technique that simulates the quantization effect while retraining the model. To simulate quantization effect, weights and activations are quantized to lower precision and then converted back to the original precision during forward and backward passes. This is achieved by adding a fake quantization block that performs immediate quantization and de-quantization operations. Fake quantization introduces the quantization error in retraining phase, allowing the model parameters to be updated to reduce the quantization error. However, the fake quantization also presents a challenge in backpropagation since the quantization is a non-differentiable operation, making the gradient zero almost everywhere [43]. A common solution to this problem is the usage of Straight-Through Estimator (STE). STE approximates the gradient as an identity function during backpropagation while applying the fake quantization as usual in the forward pass. This process is illustrated in Figure 3 where the fake quantization is simplified as a rounding operation.

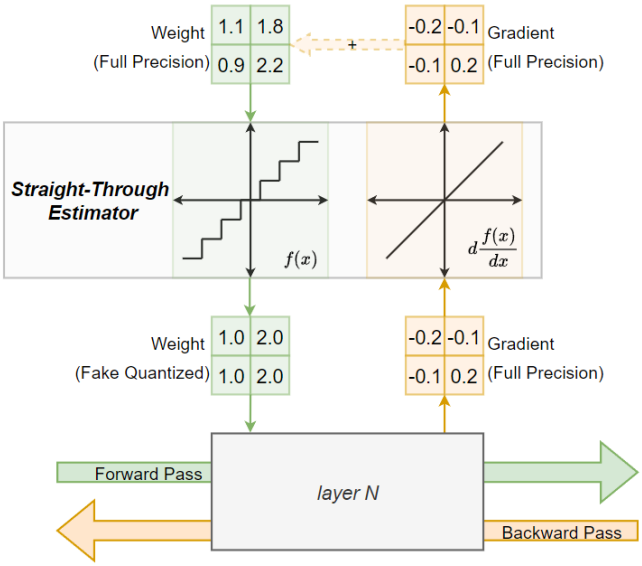


Figure 3: Illustration of the Straight-Through Estimator (STE) during the forward and backward pass

QAT can be enhanced by an iterative stochastic quantization scheme. QuantNoise [44] proposes a method that quantizes different random subsets of weights at each forward pass of QAT. The aim is to control the amount of quantization error through iterative quantization. Additionally, quantized weights can lead to gradients with high bias. Since not all weights are quantized simultaneously in QuantNoise, the bias introduced by quantization errors is not cumulative across all weights. Instead, it is averaged out over time, leading to more stable and accurate weight updates.

Weight equalization and bias correction are methods to reduce quantization error in neural networks. Weight equalization aims to make the weight distributions across different layers more uniform, which makes the network less sensitive to the changes introduced by quantization. [45] use scaling equivalence property of activation functions like ReLU to adjust weight ranges. Also, [45] claims that quantization can

produce a biased error in the activations, which can be computed and corrected using batch normalization parameters.

Distillation and pruning can be applied to improve quantization performance. [46] proposed a method that combines quantization and knowledge distillation. This method applies knowledge distillation between a full precision teacher network and a quantized low-bit student network. They assign Kullback–Leibler (KL) divergence between the teacher and student network distributions and train both networks to make the teacher network quantization-friendly. [18] combines quantization with both distillation and pruning to increase the compression rate and improve performance. They firstly combine iterative pruning with quantization-aware training to prune and quantize at each iteration. Then, they use pruned weights to construct the teacher model, and apply knowledge distillation between the non-pruned full precision teacher network and the pruned low-precision student network to improve performance.

2.4 Zero-Shot Quantization

Training data is not always accessible due to privacy and security concerns. Lack of training data restricts the availability of quantization methods. PTQ requires training data for calibration to capture the best clipping range. PTQ can be applied without training data only by dynamic quantization which is described in the section 2.2. However, dynamic quantization brings calibration overhead to each inference, so it limits the potential of the quantization. Moreover, QAT become unavailable since it requires training data for the re-training phase. Since the two most popular quantization methods are not available due to lack of training data, data-free quantization methods are developed which is called zero-shot quantization.

Weight equalization and bias correction method [45] described in the section 2.3 is one of the data-free quantization methods. Since this method relies on the scale-equivariance property of (piece-wise) linear activation functions, it may be less effective for neural networks with non-linear activations [41], such as YOLOv5, YOLOv8, or EfficientDet [47] with SiLU [48].

SQuant [49] proposes a Hessian-based zero-shot quantization method. The Hessian matrix represents the second-order partial derivatives of the loss function with respect to the model weights. It provides information about the curvature of the loss function, indicating how sensitive the loss is to changes in the weights. By using this information, SQuant can determine which weights are more sensitive to quantization errors. The objective is to minimize the absolute sum of errors introduced by quantizing the model weights. This results in mixed-precision quantization without any data.

Both methods proposed in [45] and [49] are good examples of data-free quantization. However, these methods can cause substantial performance degradation when quantizing models to ultra-low precision [50].

2.4.1 Synthetic Data Generation

One of the early work in synthetic data generation with neural networks is DeepDream algorithm [51]. This method involves feeding a random noise image into the network, asking the network to enhance certain features it detects, and iteratively amplifying these features. The result is a highly detailed and often surreal image that highlights the features the network has learned to recognize.

ZeroQ [21] is one of the pioneering works in zero-shot quantization. The proposed method generates a synthetic dataset, referred to as Distilled Data, which is engineered to match the batch normalization (BN) statistics (mean and standard deviation) of the original training data. They start with random Gaussian data with zero mean and unit variance to feed the neural network model. In each feed forward pass, the difference between the BN statistics of the input of the batch normalization layer and the stored BN statistics is used to construct the loss function. In backpropagation, the input image is updated to fit the BN statistics. The resulting image is expected to mimic the statistical properties of the original training data. The generated dataset is used in PTQ, which results in accuracy close to PTQ with the original training dataset.

[52] discusses that fitting BN statistics does not fully imitate the training data, and each generated synthetic data sample is quite similar because they are optimized by the same objective. They add some randomness to BN statistics to allow more variation in the synthetic data. Additionally, they use a loss function that reinforces the BN statistics of a specific layer for each sample to increase sample-level diversity.

[50] also discusses that synthetic images generated by BN statistics are homogeneous and fail to capture the diverse nature of real data within each class. They propose methods to use intra-class heterogeneity to solve this homogeneity problem. They define a loss as a combination of three distinct loss functions. The first one is BN statistics alignment loss to ensure that the synthetic data matches the statistical properties of the original training data. The second loss, marginal distance constraint loss, ensures that the feature vectors of synthetic images are not too similar (encouraging diversity) and not too dissimilar (maintaining class coherence). The third loss is soft inception loss, which introduces soft labels instead of hard one-hot labels to prevent overfitting to fixed objects.

[53] introduces generative models for synthetic data generation. In addition to the BN statistics, they utilize class label information for the loss. The generator is trained to produce synthetic data that the pretrained model classifies correctly. [54] is another method that utilizes both BN stats and class labels. They construct more complex label generators by ensembling multiple compressed models. [55] utilizes neural architecture search to discover an optimized generator architecture for reconstructing training data from a pretrained model.

[56] points out that BN statistics-based generation leads to poor performance on hard samples. Therefore, they define a method to measure the difficulty of samples and add this difficulty measurement into the loss function to prioritize hard samples dur-

ing synthesis. They combine BN statistics alignment loss and hard-sample-enhanced inception loss. Their results show that the lack of hard samples is a critical factor in the performance gap between models trained with synthetic data and those trained with real data. However, their hard sample measurement method covers only classification problems.

All of the methods mentioned above improve the performance of data generation based on BN statistics. However, they are tailored for image classification problems. Therefore, their usage is limited without adaptation or rework for other computer vision problems, including object detection.

ZAQ [22] utilizes a generator in the context of GANs to create synthetic data in an adversarial manner. They measures the discrepancy between a full-precision model and its quantized counterpart at both output and intermediate feature levels. They experimented with the proposed method in classification, object detection and segmentation problems. Therefore, ZAQ remains an important alternative to ZeroQ.

GENIE [57] proposes another generative model based on BN statistics to generate synthetic data. Instead of using BN statistics directly, they use these statistics to optimize latent vectors. By optimizing the latent vectors, the generator can produce a variety of synthetic images that match the statistical properties of the training data. This ensures that the synthetic data is not only similar in distribution but also diverse enough to represent the complexity of the training dataset.

Even though the experimental results presented in ZAQ and GENIE are better than ZeroQ results, there is considerable computational overhead with these methods.

2.5 Object Detection in Infrared Domain

2.5.1 Infrared Imagery

Object detection in the infrared (IR) domain is an active research area. There are many application fields, including surveillance, autonomous driving, medical applications, and rescue operations. For thermal cameras, there are two main types of IR detectors, thermal and photon detectors. Thermal detectors are low-cost sensors operating at room temperature; however, they have slow response times, low sensitivity, and low resolution. Photon detectors, while offering higher sensitivity and faster operation, require cooling and have a limited IR spectrum range. The images constructed from both sensors are represented as greyscale images with a depth of 8 to 16 bits per pixel [58].

Infrared sub-bands are categorized by their wavelength ranges and unique properties. List of IR sub-bands are provided in Table 1. The Visible, Near-Infrared (NIR), and Short-Wave Infrared (SWIR) light offer high clarity and resolution due to their alignment with peak solar illumination and high atmospheric transmission. However, SWIR systems struggle to image objects below 300K without external light. Mid-

Wave Infrared (MWIR) captures emitted radiation from objects, making it useful for passive imaging and long-range detection. Long-Wave Infrared (LWIR) is effective for imaging through smoke or particles and is preferred for surveillance. Far Infrared (FIR) covers a broad spectrum but is limited by atmospheric absorption, making it mainly useful for astronomical observations outside Earth's atmosphere [59].

Table 1: Infrared Sub-Bands and Their Wavelength Regions

Wavelength Region	**Infrared Band**
0.7 μm to 1.4 μm	Near-Infrared (NIR)
1.4 μm to 3 μm	Short-Wave Infrared (SWIR)
3 μm to 8 μm	Mid-Wave Infrared (MWIR)
8 μm to 15 μm	Long-Wave Infrared (LWIR)
15 μm to 1 mm	Far-Infrared (FIR)

2.5.2 Object Detection

The need for robust object detection techniques that can operate under adverse environmental conditions (e.g., low/excessive lighting, fog, and rain) and handle low object resolution has grown due to the expanding use of computer vision in civilian and military applications. Traditional object detection methods using visible images often struggle in these conditions, as objects may lack details due to poor lighting, occlusions from reflections, weather conditions, and other obstructions. IR images, with their ability to capture thermal signatures, can mitigate some of these challenges [60].

Working with IR images presents some other challenges. IR images often suffer from low resolution and a lack of visual details. Fusing features from visual and thermal data is a method to overcome this issue. However, the misalignment of features poses a significant challenge. [61] proposes enhancing overall feature quality and balancing the complementary and consistent nature of fused features, leading to improved object detection performance. However, obtaining pairs of RGB and IR images is not always feasible as it requires calibrated and synchronized sensors to capture the same scene simultaneously. To address this issue, GAN-based methods have been developed to translate thermal images into pseudo-RGB images, enabling the fusion of features from IR images and pseudo-RGB images for better performance [62]. However, pseudo-RGB images also have low resolution like their IR counterparts, which limits the potential of feature fusion. Additionally, the method is highly dependent on the performance of GANs, which requires extra training and development processes.

Moreover, the unique characteristics of IR imagery, such as varying thermal signatures depending on object material, temperature, and emissivity, introduce further complexity in object detection tasks. Unlike visible light, which has relatively consistent reflection patterns, thermal signatures can change rapidly with environmental conditions, leading to significant variations in IR image data. This variability makes it difficult for standard object detection algorithms to generalize well across different

scenarios. To address these challenges, there are adaptive algorithms that can dynamically adjust to the changing thermal landscape, utilizing techniques such as adaptive thresholding, multi-spectral feature extraction, and thermal-invariant feature learning [63]. These advancements are important for improving the robustness and accuracy of object detection systems that rely on IR imagery, particularly in mission-critical applications where consistent performance is essential.

Another challenge for object detection on IR imagery is dataset availability. While there are many large datasets available for visible imagery, there are few for IR imagery. This makes training a model from scratch for IR imagery challenging. To overcome this issue, a popular solution is domain adaptation, which involves fine-tuning pretrained (RGB imagery) networks with IR imagery [64].

CHAPTER 3

METHODOLOGY

3.1 Overview

The infrared domain adaptation of zero-shot quantization is experimented on in this thesis. Pretrained neural network models are fine-tuned with an infrared dataset, FLIR ADAS. Batch normalization statistics are utilized to distill data from the model for quantization calibration, making it zero-shot quantization. Additionally, training data is used for post-training quantization (PTQ) to compare the performance of zero-shot quantization. The main structure of the methodology and experiments is represented in Figure 4.

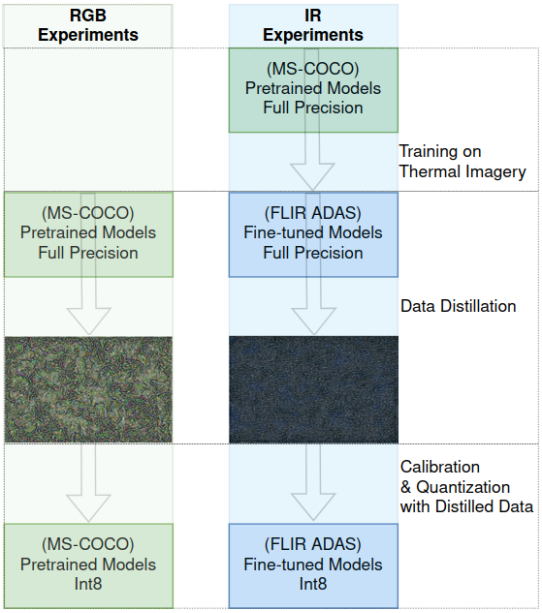


Figure 4: The proposed zero-shot quantization framework

3.2 Fine-tuning with Infrared Dateset

Training a neural network involves optimizing the model's parameters to accurately perform a specific task, such as classification, segmentation, or object detection. The thesis utilizes and focuses on supervised object detection tasks.

3.2.1 Dataset Selection

Dataset selection or creation is the core part of the training. High-quality, diverse, and well-prepared datasets enable models to learn effectively and generalize well to unseen data. Increasing the amount of data helps the model learn and generalize better. However, certain parameters must be considered when increasing dataset size. It is essential to have a balanced number of class labels to prevent overfitting or underfitting. Also, data variety is important for better generalization. Including data with different backgrounds, lighting conditions, etc., is important for robust learning.

Augmentation techniques such as rotation, flipping, scaling, and cropping can also increase both the variety and size of the dataset, increasing robustness and preventing over-fitting. While increasing robustness, correct labeling is important because incorrect labels can mislead the training process. Moreover, data corruption, noise, or artifacts can decrease model performance, so the dataset should be cleaned of such issues.

After preparing the dataset, it needs to be split into training, validation, and test sets. The training set typically contains %70-80 of the data. The validation set can be utilized for hyperparameter tuning and model selection. The test set should be different from the training set so that model performance can be evaluated correctly. Maintaining equal and consistent distribution of labels across each set helps to prevent bias in training and evaluation.

Preprocessing is another important aspect of dataset usage. There are many preprocessing techniques to improve training stability. It is important to apply the same preprocessing both in training and inference. Neural networks typically require inputs of fixed size. Varying image sizes can cause issues in batch processing and convolutional layer computations, so resizing is a widely used preprocessing technique. Aspect-ratio should be preserved during resizing to prevent distortion in the image which can cause loss of meaningful features. Another important preprocessing technique is normalization. Neural networks perform better with standardized input ranges. It helps in faster convergence and stable training. Subtracting the mean and dividing by the standard deviation is the general usage of normalization.

3.2.2 Training Loop

Training a neural network model can be summarized as a forward pass, a backward pass, and a parameter update. In the forward pass, training data is fed to the network, allowing each layer to apply its specific transformation to the input. Generally, the final layer generates the network's predictions. A loss function is then utilized to measure the difference between the predictions and the true labels. The main objective of the training is to minimize this loss to make the model learn the relationship between input and output.

The backward pass involves computing the gradients of the loss with respect to each parameter in the network. Partial derivatives of the loss with respect to each parameter are calculated, and the chain rule is applied to propagate the gradients backward through the network. An optimizer function is then utilized to update each parameter according to the gradients.

The training loop is repeated until the model converges, i.e., when the loss stops decreasing significantly. Training is a computationally intensive process that requires careful tuning of hyperparameters like learning rate, batch size, and the number of epochs. Each hyperparameter is important for achieving optimal accuracy and performance. The learning rate controls the step size during optimization, with values that are too high causing divergence and values that are too low leading to slow convergence. Batch size affects the stability and speed of training, with larger sizes providing more stable gradient estimates at the cost of higher memory usage, and smaller sizes introducing noise but helping escape local minima. Epochs determine how long the model trains, with too few causing under-fitting and too many potentially leading to over-fitting. The optimizer method dictates how the model's weights are updated during training, influencing convergence speed and model accuracy. Loss functions combine localization and classification losses to guide the model's learning, and regularization techniques like dropout and weight decay prevent over-fitting by penalizing complex models.

3.2.3 Fine-Tuning

Training a neural network from scratch involves initializing the model weights randomly and training the model on the target dataset from the beginning. However, this method requires a large amount of labeled data and significant computational resources. On the other hand, fine-tuning involves taking a pretrained model and adapting it to a specific task or dataset. This process leverages the knowledge acquired during the initial training on a large dataset and applies it to a new, often smaller, dataset. Fine-tuning is particularly useful when working with specialized datasets, such as infrared images, which may not have extensive labeled data available for training from scratch. The pretrained model has already learned features that can be useful for many tasks.

During fine-tuning, the model weights are slightly adjusted to better fit the target dataset. One efficient way for fine-tuning is freezing the early layers of the pretrained model to retain the general feature extraction capabilities. Additionally, using a smaller learning rate may be necessary to prevent drastic updates to the pretrained weights, helping to make gradual adjustments specific to the fine-tuning task or dataset. Fine-tuning requires fewer computational resources compared to training from scratch, making it a more practical approach for many applications.

3.2.4 Infrared Dataset Utilization

Infrared datasets often differ significantly from RGB datasets, as they capture thermal information rather than visible light. Fine-tuning pretrained models on infrared datasets allows the models to adapt to these differences without needing to learn basic visual features from scratch. Also, publicly available infrared datasets are much smaller in number and size compared to publicly available RGB datasets. This makes it difficult to train a model from scratch with infrared data, so fine-tuning is an efficient way to train a model with infrared data.

RGB datasets consist of images in the visible spectrum, capturing color information in three channels (red, green, and blue). Infrared datasets, on the other hand, capture thermal information, which is often represented in a single channel or with different intensity values corresponding to temperature variations. To handle this difference, The architecture of the model might need slight modifications to process the single-channel input of infrared images instead of the three-channel input of RGB images. However, changes in the model architecture can prevent fine-tuning. Therefore, a common approach to handling single-channel infrared data is to duplicate the data and feed it as three-channel data to maintain the pretrained model architecture.

In the experiments, we utilized RetinaNet and YOLOv8 (small, medium, large) models. Starting from pretrained weights on MS-COCO, we fine-tuned the models with the FLIR ADAS dataset, which is an infrared dataset. We fed the infrared images as three-channel images by duplication. Both fine-tuning details and dataset explanations are given in Chapter 4.

3.3 Data Distillation

In the absence of training data, zero-shot quantization is an emerging data-free quantization technique that focuses on generating synthetic images for the calibration process of quantization. In our work, we utilized batch normalization statistics to generate synthetic images. The applied data distillation methodology is based on ZeroQ [21]. The data distillation process is visualized in Figure 5. Data distillation is applied to both full-precision pretrained and fine-tuned models in our experiments. The only architectural difference between the pretrained and fine-tuned versions of the models is the output number of the classification head because of the difference between the

number of labeled classes in the MS-COCO and FLIR ADAS datasets. Therefore, the same distillation process is applied to both versions of the models, and a three-channel image is distilled.

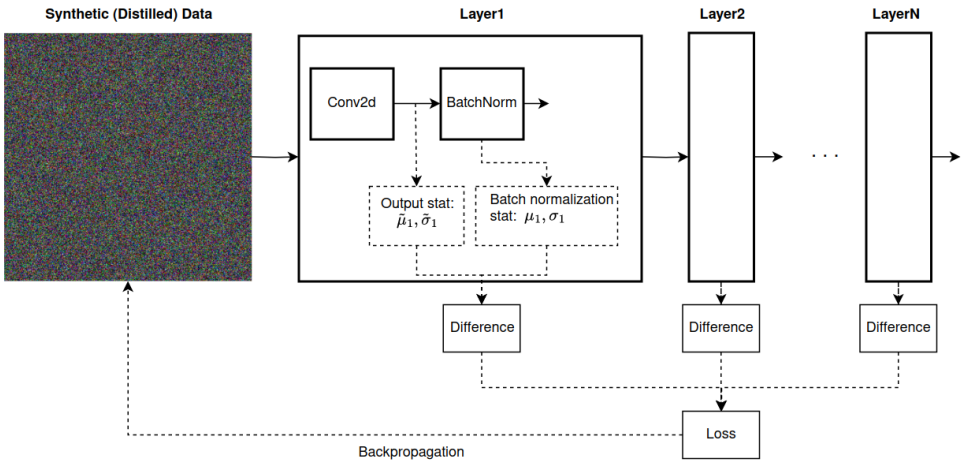


Figure 5: Data Distillation Process.

The data distillation process described in ZeroQ requires batch normalization layers since the loss is constructed from the difference between statistics (mean and standard deviation) stored in batch normalization and the statistics of the output of the preceding layer. The loss is backpropagated to the input image, which is initialized randomly from a Gaussian distribution with zero mean and unit variance. The main objective is to optimize the input data to match statistical distributions. The optimization problem is given in Formula 3 where x^d is the input data to be distilled, $\tilde{\mu}_i^d/\tilde{\sigma}_i^d$ are mean/standard deviation of the distribution of distilled data at $i-th$ batch normalization layer, and μ_i/σ_i are the mean/standard deviation parameters stored in the $i-th$ batch normalization layer.

$$\min_{\mathbf{x}^d} \frac{1}{L} \sum_{i=0}^{L} \left(\|\tilde{\mu}_i^d - \mu_i\|_2^2 + \|\tilde{\sigma}_i^d - \sigma_i\|_2^2 \right) \tag{3}$$

3.4 Quantization

We use uniform asymmetric quantization (aka the uniform affine quantization) in our implementation. Asymmetric quantization generally captures a tighter clipping range since the distributions of activation and weight values are not typically centered around zero. The first part of asymmetric quantization is determining the clipping range. There are many methods [65, 66] to determine the clipping range, such as min/max, percentile and entropy methods. We use the percentile method with a 99.99% threshold for the clipping range because it performed better than the others in our experiments for all of the utilized models.

Starting from the real-valued tensor r, we used the clipping range $[a, c]$, where a and c are determined by the 99.99th percentile of r. Specifically, a is the value below which 0.01% of the elements of r fall, and c is the value above which 0.01% of the elements of r fall. Then, we apply quantization to discretize the value distribution into even intervals. There are three parameters to perform quantization, *the scale factor s, the zero-point z, and the bit-width k*. For k-bit representation, quantization maps the parameters to $[-2^{k-1}, 2^{k-1} - 1]$. Since we use uniform quantization, the clipping range [a,c] is divided into $2^k - 1$ uniform intervals. The interval length is represented by the scaling factor, which can be calculated as $s = \frac{c-a}{2^k-1}$. Since the method is asymmetric, we need a zero-point to map real value zero to integer value as formulated in 4. With these parameters, quantization operation $Q(.)$ is formulated in 5. Rounding operation $round(.)$ rounds the input value to the nearest integer. Also, the resulting value is clamped between $[-2^{k-1}, 2^{k-1} - 1]$.

$$z = -\text{round}\left(\frac{a}{s}\right) - 2^{k-1} \tag{4}$$

$$Q(x) = \text{clamp}(\text{round}\left(\frac{x}{\text{s}}\right) + \text{z}) \tag{5}$$

For the quantization of the weights, the collection of parameter distribution is trivial. Figure 2 shows an example of clipping range determination for a convolutional layer. We utilized layer-wise quantization in our experiments. However, the quantization of activation functions is not the same due to the unbounded nature of activation functions like ReLU and its variants. To determine the clipping range for activation functions in post-training quantization (PTQ), a small portion of the training data is fed to the models, and the resulting activation values are collected for each activation function. The saved activation values result in a distribution to which we applied the same clipping range determination explained above.

The main difference between PTQ and the zero-shot quantization (ZSQ) technique applied in our experiments is the data utilized for calibration. While PTQ uses training data to capture the best clipping range, ZSQ uses synthetic (distilled) data. To make a fair comparison between PTQ and ZSQ, the same quantization technique is applied in both PTQ and ZSQ.

3.5 Evaluation Metrics

Evaluation of object detection models measures the performance and effectiveness of models. It helps to compare model performances and understand the strengths and weaknesses of models. There are different metrics for the evaluation of object detection models.

While correctly detected objects are counted as true positives, incorrectly detected objects are counted as false positives. Additionally, there are missed objects that are not detected at all, known as false negatives.

Precision: It measures the accuracy of the model's positive predictions.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{6}$$

Recall: It measures the model's ability to find all relevant objects in the dataset.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{7}$$

Average Precision (AP): AP is a widely used metric that summarizes the precision-recall curve into a single value. The precision-recall curve is plotted with precision on the y-axis and recall on the x-axis, using different confidence thresholds. AP is calculated as the area under the precision-recall curve.

Mean Average Precision (mAP): To provide an overall performance measure for object detection models, mAP extends the concept of AP to multiple classes by averaging the AP values for all classes.

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i \tag{8}$$

Intersection over Union (IoU): IoU measures the overlap between the predicted bounding box and the ground truth bounding box. It is used as a threshold to determine true positives, false positives, and false negatives.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{9}$$

COCO Evaluation Metrics: We utilized a standard COCO evaluation metric described below in the experiments for a fair comparison of the models.

mAP:0.5:0.05:0.95: This metric averages the AP values over multiple IoU thresholds (from 0.5 to 0.95 with a step size of 0.05).

CHAPTER 4

EXPERIMENTS

One of the primary goals of this thesis is to adapt zero-shot quantization to the infrared domain. To achieve this, we fine-tuned RGB image-trained (pretrained) models with infrared (IR) images and then conducted the same experiments on both the pretrained and fine-tuned models. In addition to the RGB vs. IR comparison, we compared the accuracy of full-precision models, zero-shot quantized models, and post-training quantized models to reveal the true potential of zero-shot quantization. We measured model size before and after quantization to emphasize the importance of quantization. Additionally, we adapted the data distillation process for YOLOv8 models to achieve better performance. Finally, we demonstrated the impact of model size on the performance of zero-shot quantization.

4.1 Setup

A server equipped with six NVIDIA A100 GPUs was utilized for the training phase of the experiments, consuming approximately 100 GPU hours. A computer equipped with an NVIDIA RTX 3090 Ti GPU was utilized for the rest of the experiments.

4.1.1 Dataset

FLIR ADAS [67], as a thermal dataset, and MS-COCO [68], as an RGB dataset, are utilized in the experiments. MS-COCO is used only for evaluation. The FLIR ADAS dataset v1.3 is utilized for both training and evaluation purposes. It comprises both thermal and RGB images, yet we exclusively employed thermal images in our experiments. The thermal images were captured by Teledyne FLIR Tau 2, featuring a resolution of 640x512. The dataset encompasses 10,228 images (8,862 training images, 1,366 validation images), including instances of persons, cars, and bicycles categories. Due to an insufficient number of dog instances in both the training and validation sets, we have removed the dog annotations from the dataset. Some examples can be seen in Figure 6.

Figure 6: Example images from FLIR ADAS dataset.

4.1.2 Model

All experiments were conducted on four models: RetinaNet and YOLOv8 (small, medium, large). The PyTorch [34] implementation of RetinaNet [29] (v2) with a ResNet-50-FPN backbone and the Ultralytics implementation of YOLOv8 with a custom CSPDarknet53 backbone were utilized. MS-COCO pretrained models were obtained by utilizing pretrained weights shared by PyTorch and Ultralytics. In the experiments involving the FLIR ADAS dataset, the classification heads of the models were modified for a three-class output: people, cars, and bicycles. Since thermal images consist of 8-bit one-channel data, we fed them into the model by duplicating the data across three input channels.

4.2 Implementation Details

4.2.1 Training

We fine-tuned all the models using the FLIR ADAS dataset. All layers of all models were trainable throughout the fine-tuning process.

For RetinaNet, the parameters for fine-tuning are selected to be consistent with the original RetinaNet paper. We use stochastic gradient descent (SGD) with an initial learning rate of 0.01, a weight decay of 0.0001, and a momentum of 0.9. The learning rate is divided by five after every ten epochs. A batch size of 8 images is utilized, and the training lasts 40 epochs. For the YOLOv8 models, the same training hyperparameters are utilized. We use SGD with an initial learning rate of 0.01, a weight decay of 0.0005, and a momentum of 0.937. A batch size of 64 is used. Training is conducted for 80 epochs for the YOLOv8 small model, 100 epochs for the YOLOv8 medium model, and 120 epochs for the YOLOv8 large model.

4.2.2 Quantization

All weights and activations of the models are quantized to 8 bits. We utilized TensorRT's [69] PyTorch-Quantization toolkit for the quantization process, which was conducted using distilled data. The calibration process was also conducted using the

same toolkit. There are publicly available custom implementations for quantization, such as the implementation of ZeroQ [21] or HAST [56]. However, these implementations restrain the measurement of true quantized inference. Additionally, there are limitations to observing the effects of quantization in these implementations since they do not completely quantize the model. Therefore, the PyTorch-Quantization toolkit was utilized for the experiments to observe the quantization effect.

4.2.3 Zero-shot Quantization

The hyperparameters for data distillation, initially set for the RGB experiments, were also applied to the infrared experiments, consistent with the ZeroQ [21] implementation. We used an initial learning rate of 0.1 with the Adam optimizer [70]. The number of distillation iterations varied depending on the model size. The best data was distilled from RetinaNet after 82 iterations. The YOLOv8 small model required 80 iterations, the medium model required 95 iterations, and the large model required 100 iterations. Most of the fine-tuned models achieved their best performance with the same number of iterations, except for the fine-tuned YOLOv8 large model, which required 110 iterations. Additionally, we experimented with different loss functions: one that included both mean and variance statistics, and another that used only the mean statistic. We found that using standard deviation statistics had a negligible effect, as its value became insignificant compared to the mean statistics. Therefore, we utilized variance statistics instead.

4.2.3.1 Data Distillation for YOLOv8

YOLOv8 models apply a preprocessing function before passing the image to the first layer of the neural network. This preprocessing helps the model converge faster during training, makes it less sensitive to the scale of input data, and improves generalization across different datasets [33]. The preprocessing of YOLOv8 models divides the input image by 255 if its maximum value is above 1.0. However, this preprocessing decreases the performance of zero-shot quantization, as defined in Section 3.3 and based on [21], because the data distillation process does not include a normalization step. Since removing this preprocessing function would reduce performance and prevent the use of pretrained model weights, disabling the preprocessing is not a viable solution. To address this issue, various solutions were tested, including normalizing the distillation data and adding a normalization step to the data distillation process. The solutions and results are discussed in Chapter 5.

4.2.4 Post Training Quantization

We investigated the performance of the models using both the FLIR ADAS and MS-COCO datasets for post-training quantization. For fine-tuned models, we divided the training set of the FLIR ADAS dataset into batches of size 128 and used these batches

for calibration during post-training quantization. The accuracy result of each batch was collected to observe the diversity of quantization performance across the training dataset. The same procedure was repeated for pretrained models with the training set of the MS-COCO dataset. Since the MS-COCO dataset is much larger than the FLIR ADAS dataset, the training set was divided into batches of size 1024. We then compared the results with the corresponding results of zero-shot quantization.

CHAPTER 5

RESULTS AND DISCUSSIONS

Zero-shot quantization (ZSQ) is a technique that emerges when training data is unavailable. This thesis utilizes a batch normalization-based ZSQ method. Our approach is based on ZeroQ [21]. We thoroughly investigated the method to adapt it to the infrared domain and applied it to YOLOv8 models. First, we experimented with and discussed the use of a loss function based on only the mean statistic, as well as one using both mean and standard deviation. Then, we presented methods to apply ZSQ to YOLOv8 models.

Quantization may become necessary for real-time embedded applications. Model size is one of the key parameters for these applications. We measured and compared model sizes to demonstrate the effectiveness of the quantization we applied. Next, the accuracy drop, which is the bottleneck of quantization, was measured by comparing the accuracy performance of full-precision models with that of these models after ZSQ. However, to see the true potential of ZSQ, comparing it with post-training quantization (PTQ) is essential. Therefore, we analyzed the PTQ performance of the models in detail. Finally, we compared the performance of ZSQ with both PTQ and full-precision models.

To observe the difference between RGB-trained and infrared-trained models, we fine-tuned pretrained models with infrared imagery. We then compared the pretrained and fine-tuned models in the context of zero-shot quantization. All experiments are conducted for both of the pretrained and fine-tuned models for a fair comparison. Moreover, using the small, medium, and large models of YOLOv8 enabled us to observe and interpret the effect of model size on zero-shot quantization.

Comparison of our ZSQ framework with others is quite difficult. First, to the best of our knowledge, there is no other work that applies a ZSQ method to the infrared domain or utilizes state-of-the-art YOLOv8 models for ZSQ. Adapting YOLO models to the infrared domain makes our work unique. Therefore, we have conducted extensive experiments to demonstrate the potential of our framework.

5.1 Data Distillation

Data distillation, also known as synthetic data generation, is the core part of our ZSQ framework. In the absence of training data, distilled data can be utilized to determine quantization parameters such as the clipping range of activation functions. The data distillation process is based on the internal statistics (mean and standard deviation) of the model. The initial data is updated to fit these statistics as described in 3.3. We experimented with data distillation using different distillation settings and identified the best ones, as described in 4.2. One of the main parameters for data distillation is the number of distillation iterations. The optimal number of iterations depends on the model architecture, size, and other distillation parameters. We conducted experiments with both the pretrained and fine-tuned models of RetinaNet and YOLOv8 (small, medium, large). Our results can be seen in Figures 8, 9, 10, and 11.

ZeroQ [21] utilizes a loss function that includes both mean and standard deviation statistics. However, during our experiments, we realized that the loss from the standard deviation statistic is much smaller than the loss from the mean statistic, and it has no noticeable impact on performance. Therefore, we utilized variance statistics, which is the square of the standard deviation, as used in [56]. In further experiments, we demonstrated that adding variance statistics has a negligible or even negative impact on performance. To compare the use of loss with mean statistics versus loss with both mean and variance statistics, we measured accuracy performance using both types of loss functions.

Instead of providing only the best results obtained, different numbers of distillation iterations are presented in Figures 8, 9, 10, and 11. In all cases, we achieved better accuracy using the loss with only mean statistics. The initial loss was much higher in the RetinaNet models compared to the YOLOv8 models. Consequently, the accuracy drop in the early phase of distillation was larger in RetinaNet. For both pretrained and fine-tuned models, the initial data is generated from a random Gaussian distribution with zero mean and unit variance. The generated image is naturally an RGB image since we utilized three-channel inputs in all experiments, as described in 3.2. Therefore, the initial loss was much higher in fine-tuned models than in pretrained models. Since the channels of the batch normalization layers are similar but not identical, we did not start with duplicated values for the initial data.

To illustrate the changes in the distilled data after fine-tuning with thermal images, we visualized the distilled data from both the pretrained RetinaNet and the fine-tuned RetinaNet in Figure 7. We also observed similar distilled data from the YOLOv8 models. The distilled data from the fine-tuned RetinaNet is close to grayscale, as it reflects the statistics of the training dataset. For a deeper understanding of the distilled data, we presented it at various iteration numbers.

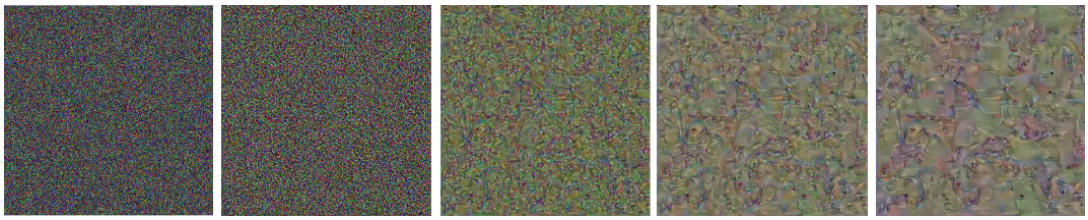
We also tested how well our models performed after quantization using data that was randomly generated from a normal distribution (mean of zero and variance of one) without using any distillation process. The results are presented in Table 2. Since the initial data is three-channel data, we expected it to perform better in pretrained

models. This expectation was met in the YOLOv8 models, but the initial Gaussian data performed better in the fine-tuned version of RetinaNet. Insufficient training of RetinaNet with infrared imagery may have caused this result. In that case, the channels of the batch normalization layers may not be similar enough, leading the initial Gaussian data to perform worse in the pretrained model than in the fine-tuned one.

The gap between the accuracy of the initial data and the distilled data after a few iterations is considerably large. To ensure the graphs have finer detail, we did not include these results, which do not belong to the distillation process, in Figures 8, 9, 10, and 11. All YOLOv8 results in these figures and in Table 2 were obtained by adding the post and periodic normalization steps. The details are explained in the next section.

Table 2: Performance Results (mAP) after Quantization Using Random Gaussian Data (Zero Mean, Unit Variance) without Distillation.

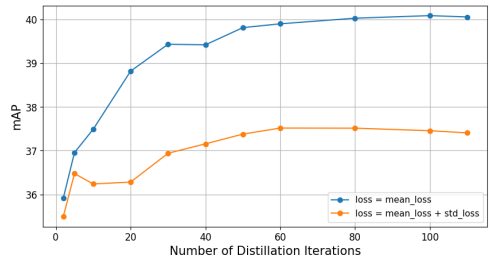
Model	**Pretrained mAP**	**Fine-tuned mAP**
RetinaNet	16.5	19.18
YOLOv8s	39.6	38.5
YOLOv8m	45.3	39.1
YOLOv8l	46.8	41.6



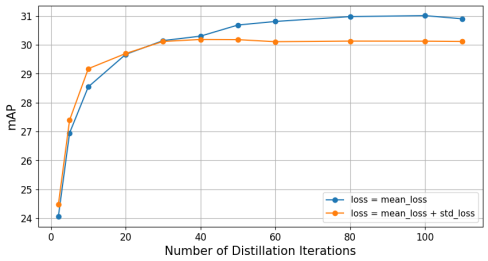
(a) Distilled Data from MS-COCO pretrained RetinaNet

(b) Distilled Data from FLIR ADAS fine-tuned RetinaNet

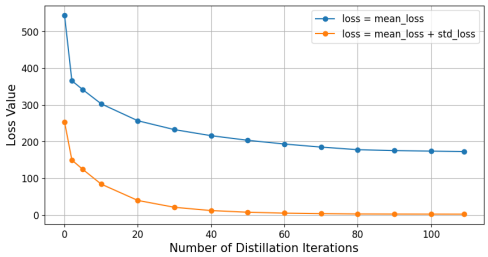
Figure 7: Distilled data comparison. Each column corresponds to an output of ith iteration of data distillation. The 1st through 5th columns correspond to the 1st, 10th, 50th, 100th, and 200th iterations, respectively.



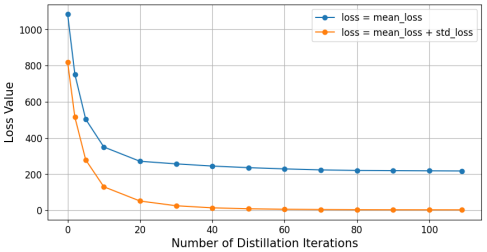
(a) Number of distillation iterations vs. accuracy (mAP) for zero-shot quantization of pretrained RetinaNet.



(b) Number of distillation iterations vs. accuracy (mAP) for zero-shot quantization of fine-tuned RetinaNet.

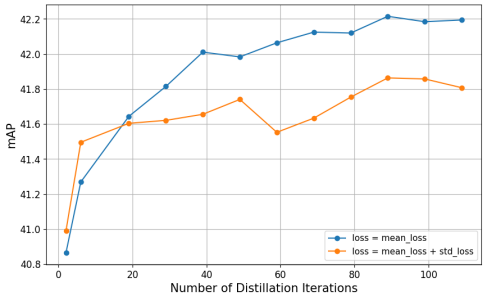


(c) Loss during distillation vs. the number of distillation iterations using pretrained RetinaNet.

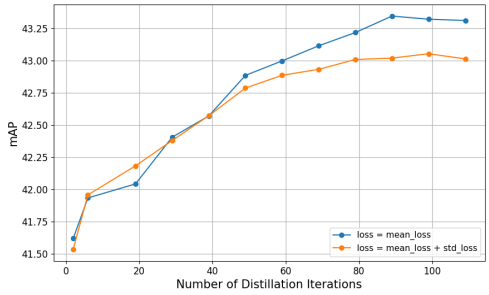


(d) Loss during distillation vs. the number of distillation iterations using fine-tuned RetinaNet.

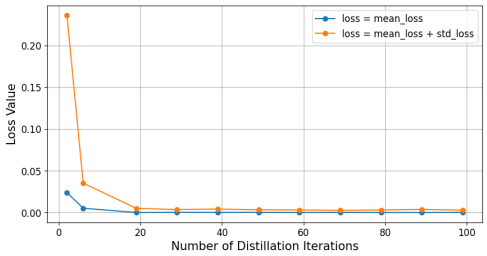
Figure 8: Comparison of accuracy and loss during distillation with varying numbers of iterations for both pretrained and fine-tuned RetinaNet models.



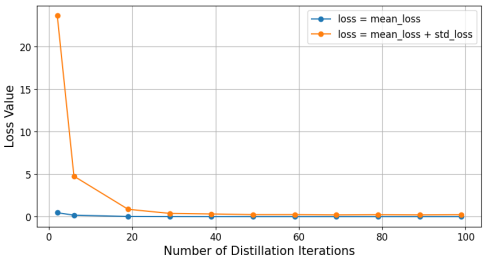
(a) Number of distillation iterations vs. accuracy (mAP) for zero-shot quantization of pretrained YOLOv8s.



(b) Number of distillation iterations vs. accuracy (mAP) for zero-shot quantization of fine-tuned YOLOv8s.

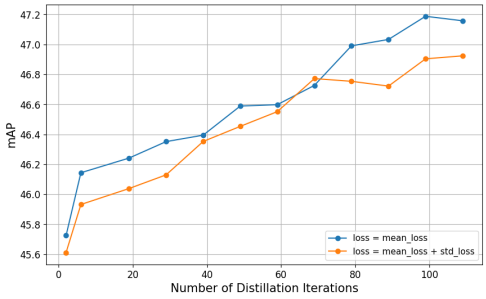


(c) Loss during distillation vs. the number of distillation iterations using pretrained YOLOv8s.

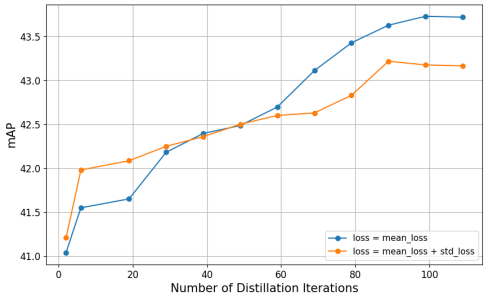


(d) Loss during distillation vs. the number of distillation iterations using fine-tuned YOLOv8s.

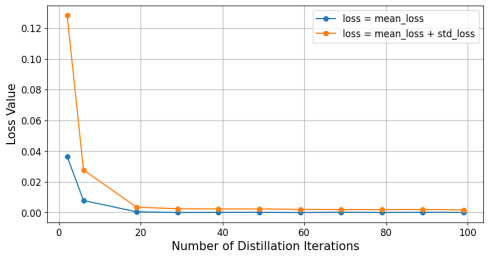
Figure 9: Comparison of accuracy and loss during distillation with varying numbers of iterations for both pretrained and fine-tuned YOLOv8-small models.



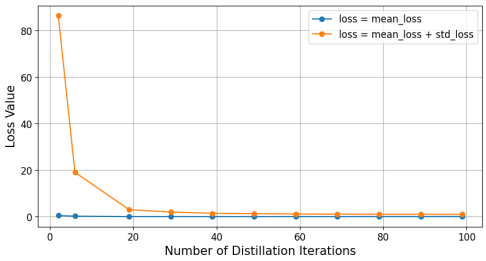
(a) Number of distillation iterations vs. accuracy (mAP) for zero-shot quantization of pretrained YOLOv8m.



(b) Number of distillation iterations vs. accuracy (mAP) for zero-shot quantization of fine-tuned YOLOv8m.

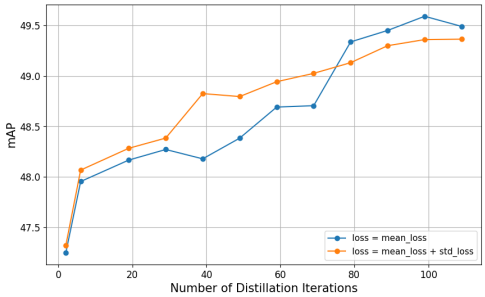


(c) Loss during distillation vs. the number of distillation iterations using pretrained YOLOv8m.

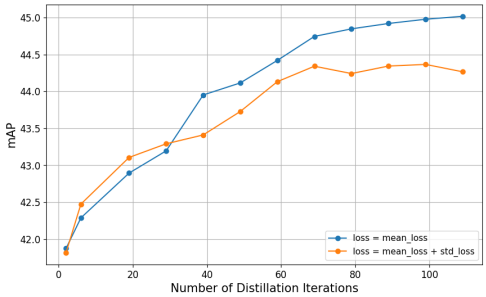


(d) Loss during distillation vs. the number of distillation iterations using fine-tuned YOLOv8m.

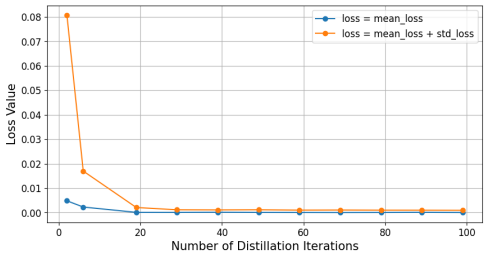
Figure 10: Comparison of accuracy and loss during distillation with varying numbers of iterations for both pretrained and fine-tuned YOLOv8-medium models.



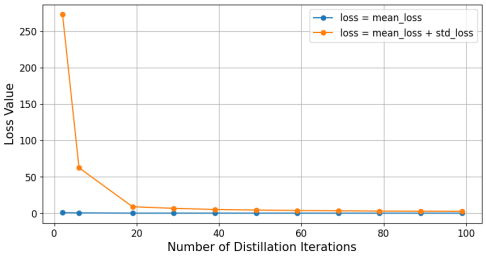
(a) Number of distillation iterations vs. accuracy (mAP) for zero-shot quantization of pretrained YOLOv8l.



(b) Number of distillation iterations vs. accuracy (mAP) for zero-shot quantization of fine-tuned YOLOv8l.



(c) Loss during distillation vs. the number of distillation iterations using pretrained YOLOv8l.



(d) Loss during distillation vs. the number of distillation iterations using fine-tuned YOLOv8l.

Figure 11: Comparison of accuracy and loss during distillation with varying numbers of iterations for both pretrained and fine-tuned YOLOv8-large models.

5.1.1 Adaptation of Data Distillation for YOLOv8

The YOLOv8 models include a normalization step as a preprocessing, as described in 4.2. The reason for not disabling this step is also discussed in that section. We provided a solution to address this issue. YOLO preprocessing simply divides all values of the input image by 255 if the maximum value is above 1.0. This process generally causes the values of the distilled data to approach zero, resulting in near-zero accuracy after quantization. To counter this, we added a post-normalization step that normalizes the distilled data between zero and one using the minimum and maximum values of the data before passing it to the model for quantization calibration. The post-normalization step made the accuracy results feasible. However, since the distilled data is not intended to be normalized during its generation, the performance in this setting is not optimal. To incorporate the normalization effect into data distillation, which is a learning process, we added a periodic normalization step. This step periodically normalizes the data during distillation, ensuring that the data fits between zero and one while learning to match the batch normalization statistics. In the periodic normalization process, we experimented with different normalization methods to observe the performance results. The best result was obtained using the sigmoid function. The non-linearity of the sigmoid function offers some advantages, as it tends to mitigate the effect of extreme values by compressing them.

Our results can be seen in Table 3. The use of min-max normalization as post-normalization and the sigmoid function for periodic normalization consistently performed better. In that experiment, the loss with only the mean statistic was used. The loss with both mean and variance statistics performed worse, as discussed in 5.1.

5.2 Comparison of Zero-Shot Quantization with Full-Precision Models

A drop in accuracy due to quantization is expected, especially in the absence of training data. To evaluate the performance of zero-shot quantization (ZSQ), we compared full-precision and zero-shot quantized models. The results are presented in Table 4. We measured the difference in mAP and the percentage of mAP loss after ZSQ. In the comparison between fine-tuned and pretrained models, the fine-tuned versions performed better after ZSQ across all tested models. We observed a smaller accuracy drop in pretrained RetinaNet and a larger accuracy drop in fine-tuned RetinaNet compared to the YOLOv8 models. However, the mAP loss ratios between pretrained and fine-tuned models across the YOLOv8 models were quite similar. Therefore, we concluded that batch normalization statistics-based zero-shot quantization depends on both the model architecture and training strategies. The higher accuracy drop in fine-tuned RetinaNet, as discussed in Section 5.4, may indicate that RetinaNet was not fine-tuned as efficiently as the YOLOv8 models.

To observe the effect of model size on ZSQ, it is necessary to use the same model architecture for an accurate comparison. Therefore, our experiments utilized the small, medium, and large YOLOv8 models. As the model size increases, the number of

batch normalization layers also increases. As shown in Table 4, increasing the model size from small to large YOLOv8 results in a greater accuracy drop after ZSQ in both pretrained and fine-tuned models. Therefore, we concluded that increasing model size decreases the performance of ZSQ.

5.3 Comparison of Zero-Shot Quantization with Post Training Quantization

Comparing ZSQ solely with full-precision accuracy may lead to underestimation. ZSQ replaces the training data with distilled (synthetic) data, so one of the critical success parameters of ZSQ is the accuracy difference between PTQ and ZSQ results. This comparison is conducted as described in Section 4.2.4.

Calibration data directly affects the performance of PTQ. To achieve the best performance, a small portion of the training data is typically used [71]. However, not all images in the training dataset perform equally well, making the selection of calibration data critical. To demonstrate the potential of PTQ, we partitioned the datasets into smaller subsets and measured the accuracy performance using these subsets in PTQ. The MS-COCO dataset contains 118,287 training images, which we split into sets of 1,024 images each. Since the FLIR ADAS dataset is much smaller, with 8,862 training images, we split it into sets of 128 images. The size of the calibration dataset is consistent with [72], where the MS-COCO training dataset was split into sets of 1,000 images.

The accuracy performance of PTQ with each calibration dataset was measured, and the distribution is presented in Figure 12 for pretrained models and Figure 13 for fine-tuned models. The corresponding zero-shot quantization results are also included in these figures. There are two main conclusions from these results. Firstly, ZSQ results are much closer to PTQ results in fine-tuned models. Secondly, increasing model size decreases the performance of ZSQ. These findings are discussed in Section 5.4.

5.4 Discussion

The experiments are based on comparing ZSQ with full-precision models and PTQ. Both sets of experiments showed that fine-tuned models trained with the infrared dataset performed better in ZSQ in terms of accuracy. Additionally, in both experiments, increasing model size decreased the ZSQ performance.

Batch normalization statistics-based ZSQ can be seen as a regression problem, where the image is updated to fit the batch normalization statistics of the model. Increasing the model size increases the number of batch normalization layers, making the regression problem more complex. Our experiments showed that increasing the number of batch normalization layers decreased ZSQ performance, as it is harder to fit more statistics.

On the other hand, we observed a similar effect in the infrared domain adaptation. The infrared dataset, which is FLIR ADAS in our experiments, has single-channel images that are fed to the network by duplicating them to three channels during training. This process makes all channels of a layer similar, as they all observe the same or similar input and backpropagation. This results in batch normalization layers with similar channel values, simplifying the regression problem. Consequently, zero-shot quantization performed better in fine-tuned models.

There is one exception in our experiments. Zero-shot quantization in the fine-tuned RetinaNet performed worse than in the pretrained counterpart. We suspect insufficient training of RetinaNet with the FLIR ADAS dataset. If the model is not well-trained, the channels may not be as close to each other as in YOLO models. Therefore, this may have reduced the performance improvement because the layer channels being close to each other.

Table 3: Comparison of Data Distillation Setups for YOLOv8 models

(a) YOLOv8s

Setup	Model Type	Post-Normalization	Periodic-Normalization	mAP
Setup 0	Pretrained	None	None	0.8
Setup 1	Pretrained	Min-max	None	41.4
Setup 2	Pretrained	Min-max	Min-max	41.5
Setup 3	Pretrained	Min-max	Sigmoid	**42.2**
Setup 4	Fine-tuned	None	None	0.1
Setup 5	Fine-tuned	Min-max	None	42.5
Setup 6	Fine-tuned	Min-max	Min-max	42.6
Setup 7	Fine-tuned	Min-max	Sigmoid	**43.3**

(b) YOLOv8m

Setup	Model Type	Post-Normalization	Periodic-Normalization	mAP
Setup 0	Pretrained	None	None	1.2
Setup 1	Pretrained	Min-max	None	46.9
Setup 2	Pretrained	Min-max	Min-max	47.0
Setup 3	Pretrained	Min-max	Sigmoid	**47.2**
Setup 4	Fine-tuned	None	None	0.1
Setup 5	Fine-tuned	Min-max	None	42.4
Setup 6	Fine-tuned	Min-max	Min-max	42.6
Setup 7	Fine-tuned	Min-max	Sigmoid	**43.7**

(c) YOLOv8l

Setup	Model Type	Post-Normalization	Periodic-Normalization	mAP
Setup 0	Pretrained	None	None	1.7
Setup 1	Pretrained	Min-max	None	49.3
Setup 2	Pretrained	Min-max	Min-max	49.3
Setup 3	Pretrained	Min-max	Sigmoid	**49.6**
Setup 4	Fine-tuned	None	None	0.1
Setup 5	Fine-tuned	Min-max	None	44.7
Setup 6	Fine-tuned	Min-max	Min-max	44.8
Setup 7	Fine-tuned	Min-max	Sigmoid	**45.0**

Table 4: Accuracy comparison of full precision (32-bit weights, 32-bit activations) and zero-shot quantized (8-bit weights, 8-bit activations) pretrained and fine-tuned models (RetinaNet, YOLOv8s, YOLOv8m, YOLOv8l).

(a) RetinaNet

Model Type	**Evaluation Dataset**	**Precision-bit**	**Model Size (MB)**	**mAP**	**mAP Difference**	**mAP Loss (%)**
Pretrained	MS-COCO	W32-A32	145.7	41.6	-1.9	4.57%
Pretrained	MS-COCO	W8-A8	36.4	39.7		
Fine-tuned	FLIR ADAS	W32-A32	138.8	32.3	-1.2	3.72%
Fine-tuned	FLIR ADAS	W8-A8	34.7	31.1		

(b) YOLOv8s

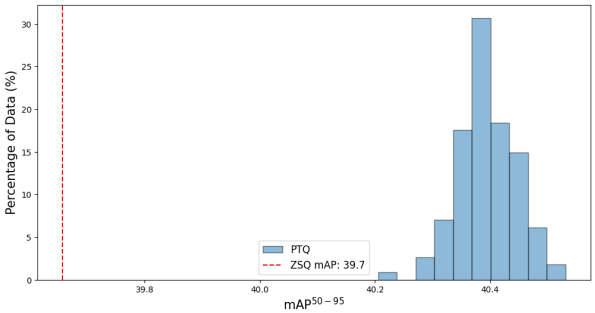
Model Type	**Evaluation Dataset**	**Precision-bit**	**Model Size (MB)**	**mAP**	**mAP Difference**	**mAP Loss (%)**
Pretrained	MS-COCO	W32-A32	44.9	44.7	-2.5	5.59%
Pretrained	MS-COCO	W8-A8	11.4	42.2		
Fine-tuned	FLIR ADAS	W32-A32	44.7	44.2	-0.9	2.04%
Fine-tuned	FLIR ADAS	W8-A8	11.4	43.3		

(c) YOLOv8m

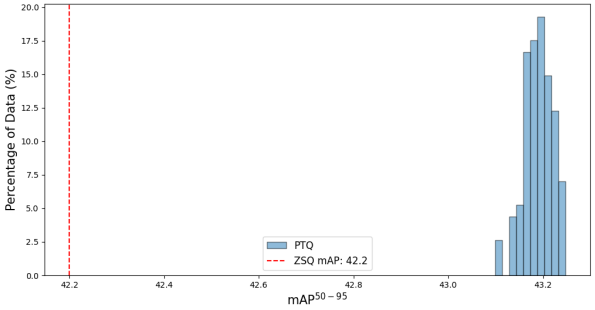
Model Type	**Evaluation Dataset**	**Precision-bit**	**Model Size (MB)**	**mAP**	**mAP Difference**	**mAP Loss (%)**
Pretrained	MS-COCO	W32-A32	103.7	50.1	-2.9	5.79%
Pretrained	MS-COCO	W8-A8	26.2	47.2		
Fine-tuned	FLIR ADAS	W32-A32	103.9	44.7	-1.0	2.24%
Fine-tuned	FLIR ADAS	W8-A8	26.1	43.7		

(d) YOLOv8l

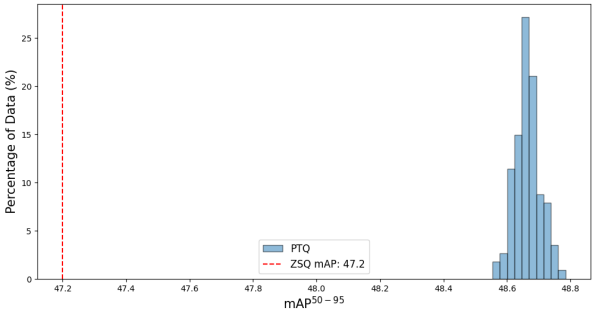
Model Type	**Evaluation Dataset**	**Precision-bit**	**Model Size (MB)**	**mAP**	**mAP Difference**	**mAP Loss (%)**
Pretrained	MS-COCO	W32-A32	175.9	52.9	-3.3	6.24%
Pretrained	MS-COCO	W8-A8	44.1	49.6		
Fine-tuned	FLIR ADAS	W32-A32	175.1	46.1	-1.1	2.39%
Fine-tuned	FLIR ADAS	W8-A8	44.1	45.0		



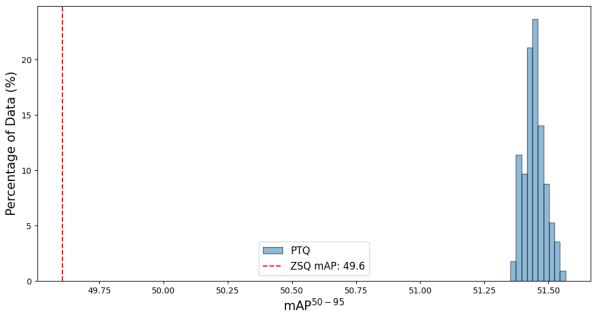
(a) Pretrained RetinaNet



(b) Pretrained YOLOv8s

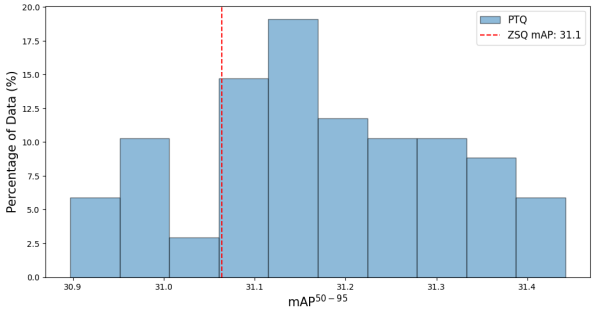


(c) Pretrained YOLOv8m

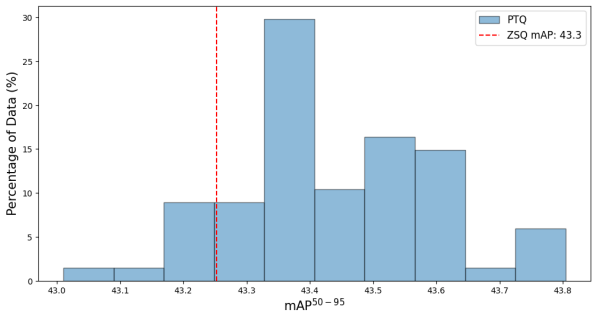


(d) Pretrained YOLOv8l

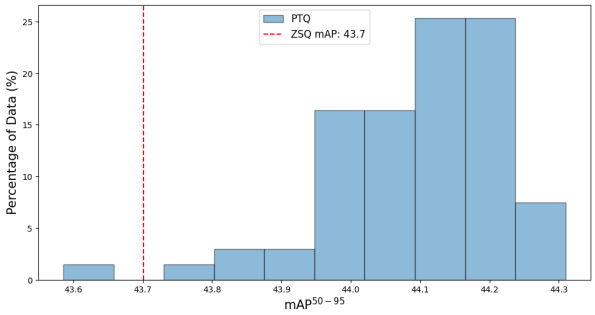
Figure 12: Comparison of Post Training Quantization, Zero-Shot Quantization and Full-precision of pretrained models in terms of accuracy.



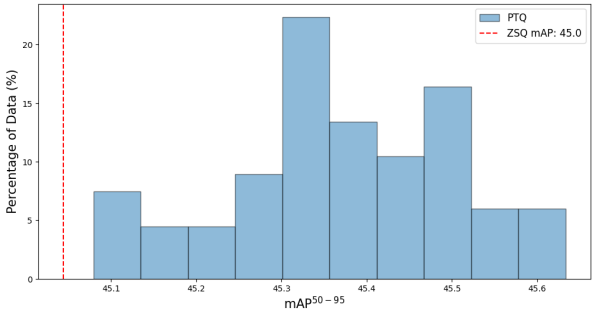
(a) Fine-tuned RetinaNet



(b) Fine-tuned YOLOv8s



(c) Fine-tuned YOLOv8m



(d) Fine-tuned YOLOv8l

Figure 13: Comparison of Post Training Quantization, Zero-Shot Quantization and Full-precision of fine-tuned models in terms of accuracy.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Zero-shot quantization is an emerging technique used when training data is not available. Despite the special nature of infrared datasets in security and medical applications, the performance of zero-shot quantization in infrared domain applications has not been investigated before. This thesis addresses this gap in the literature and investigates the infrared domain adaptation of zero-shot quantization for object detection models. We have shown that zero-shot quantization using batch normalization statistics can be effectively applied to fine-tuned models with thermal images. We evaluated RetinaNet and YOLOv8 (small, medium, large) models trained on RGB and thermal datasets, examining various aspects of zero-shot quantization.

Data distillation is the core component of zero-shot quantization, directly affecting performance. We investigated both hyperparameters and loss functions to achieve the best results. Our experiments showed that using only the mean statistic in the loss function leads to better accuracy. Additionally, we demonstrated that data distillation should be adapted for neural network models that utilize image normalization, such as YOLOv8. We provided a solution by incorporating post-normalization and periodic-normalization steps to include the effect of image normalization in the learning process of data distillation.

We interpreted updating the input image to fit the batch normalization layer statistics as a regression problem. Because of the single-channel nature of the infrared dataset, the channels of each layer in the fine-tuned model become similar to each other. This made the regression problem simpler, so models fine-tuned with the infrared dataset are more effective for zero-shot quantization. Additionally, we demonstrated that increasing model size makes this regression problem more complex due to the increasing number of batch normalization layers, resulting in worse performance for zero-shot quantization in larger models. We also analyzed the post-training quantization of both pretrained and fine-tuned models in detail to demonstrate the effectiveness of zero-shot quantization.

Since our work is the first of its kind in applying zero-shot quantization in the infrared domain, we could not compare our results with others. Additionally, we could not find another zero-shot quantization approach based on batch normalization statistics that utilizes state-of-the-art models like YOLOv8. Therefore, we conducted consistent and well-defined experiments to establish baseline results for the literature.

The research conducted in this thesis can be extended to other deep learning tasks, such as classification and segmentation. Additionally, there are various types of zero-shot quantization techniques, such as those based on GANs. These techniques can be utilized for infrared adaptation in future work.

REFERENCES

[1] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," in *Neural Information Processing Systems*, 2018.

[2] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *International Conference on Machine Learning*, 2015.

[3] Z. Liu, Y. Wang, K. Han, S. Ma, and W. Gao, "Post-training quantization for vision transformer," in *Neural Information Processing Systems*, 2021.

[4] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4820–4828, 2015.

[5] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2017.

[6] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *ArXiv*, vol. abs/1608.08710, 2016.

[7] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," in *British Machine Vision Conference*, 2015.

[8] I. Lazarevich, A. Kozlov, and N. Malinin, "Post-training deep neural network pruning via layer-wise calibration," *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 798–805, 2021.

[9] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *ArXiv*, vol. abs/1503.02531, 2015.

[10] M. Haroush, I. Hubara, E. Hoffer, and D. Soudry, "The knowledge within: Methods for data-free model compression," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8491–8499, 2019.

[11] H. Chen, Y. Wang, C. Xu, Z. Yang, C. Liu, B. Shi, C. Xu, C. Xu, and Q. Tian, "Data-free learning of student networks," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3513–3521, 2019.

[12] G. K. Nayak, K. R. Mopuri, V. Shaj, R. V. Babu, and A. Chakraborty, "Zero-shot knowledge distillation in deep networks," *ArXiv*, vol. abs/1905.08114, 2019.

[13] A. Chawla, H. Yin, P. Molchanov, and J. M. Álvarez, "Data-free knowledge distillation for object detection," *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 3288–3297, 2021.

[14] Z. Ding, Y. Chen, N. Li, D. Sun, and C. L. P. Chen, "Bnas: Efficient neural architecture search using broad scalable architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, pp. 5004–5018, 2021.

[15] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *ArXiv*, vol. abs/1802.03268, 2018.

[16] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[17] N. Aghli and E. Ribeiro, "Combining weight pruning and knowledge distillation for cnn compression," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3185–3192, 2021.

[18] J.-H. Kim, S. Chang, and N. Kwak, "Pqk: Model compression via pruning, quantization, and knowledge distillation," in *Interspeech*, 2021.

[19] V. Kryzhanovskiy, G. Balitskiy, N. Kozyrskiy, and A. Zuruev, "Qpp: Real-time quantization parameter prediction for deep neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10692, June 2021.

[20] Z. Liu, Y. Wang, K. Han, S. Ma, and W. Gao, "Instance-aware dynamic neural network quantization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12434–12443, June 2022.

[21] Y. Cai, Z. Yao, Z. Dong, A. Gholami, M. W. Mahoney, and K. Keutzer, "Zeroq: A novel zero shot quantization framework," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13166–13175, 2020.

[22] Y. Liu, W. Zhang, and J. Wang, "Zero-shot adversarial quantization," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1512–1521, 2021.

[23] S. Xu, H. Li, B. Zhuang, J. Liu, J. Cao, C. Liang, and M. Tan, "Generative low-bitwidth data free quantization," *ArXiv*, vol. abs/2003.03603, 2020.

[24] X. Zhang, H. Qin, Y. Ding, R. Gong, Q. Yan, R. Tao, Y. Li, F. Yu, and X. Liu, "Diversifying sample generation for accurate data-free quantization," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15653–15662, 2021.

[25] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," 2020.

[26] R. Girshick, "Fast r-cnn," 2015.

[27] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, *SSD: Single Shot MultiBox Detector*, p. 21–37. Springer International Publishing, 2016.

[29] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018.

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[31] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[32] G. Jocher, A. Stoken, J. Borovec, NanoCode012, ChristopherSTAN, L. Changyu, Laughing, tkianai, A. Hogan, lorenzomammana, yxNONG, AlexWang1900, L. Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, F. Ingham, Frederik, Guilhen, Hatovix, J. Poznanski, J. Fang, L. Yu, changyu98, M. Wang, N. Gupta, O. Akhtar, PetrDvoracek, and P. Rai, "ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements," Oct. 2020.

[33] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023.

[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[35] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," 2014.

[36] Z. Liu, Z. Shen, M. Savvides, and K.-T. Cheng, "Reactnet: Towards precise binary neural network with generalized activation functions," in *European Conference on Computer Vision*, 2020.

[37] P. Chen, B. Zhuang, and C. Shen, "Fatnn: Fast and accurate ternary neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5219–5228, October 2021.

[38] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," 2018.

[39] Y. Zhou, S.-M. Moosavi-Dezfooli, N.-M. Cheung, and P. Frossard, "Adaptive quantization for deep neural network," 2017.

[40] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," 2019.

[41] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *ArXiv*, vol. abs/2103.13630, 2021.

[42] M. van Baalen, B. Kahne, E. Mahurin, A. Kuzmin, A. Skliar, M. Nagel, and T. Blankevoort, "Simulated quantization, real power savings," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2756–2760, 2022.

[43] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin, "Understanding straight-through estimator in training activation quantized neural nets," 2019.

[44] A. Fan, P. Stock, B. Graham, E. Grave, R. Gribonval, H. Jegou, and A. Joulin, "Training with quantization noise for extreme model compression," 2021.

[45] M. Nagel, M. van Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," 2019.

[46] J. Kim, Y. Bhalgat, J. Lee, C. Patel, and N. Kwak, "Qkd: Quantization-aware knowledge distillation," 2019.

[47] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2020.

[48] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," 2023.

[49] C. Guo, Y. Qiu, J. Leng, X. Gao, C. Zhang, Y. Liu, F. Yang, Y. Zhu, and M. Guo, "Squant: On-the-fly data-free quantization via diagonal hessian approximation," 2022.

[50] Y. Zhong, M. Lin, G. Nan, J. Liu, B. Zhang, Y. Tian, and R. Ji, "Intraq: Learning synthetic images with intra-class heterogeneity for zero-shot network quantization," 2022.

[51] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015.

[52] X. Zhang, H. Qin, Y. Ding, R. Gong, Q. Yan, R. Tao, Y. Li, F. Yu, and X. Liu, "Diversifying sample generation for accurate data-free quantization," 2021.

[53] S. Xu, H. Li, B. Zhuang, J. Liu, J. Cao, C. Liang, and M. Tan, "Generative low-bitwidth data free quantization," 2020.

[54] X. He, Q. Hu, P. Wang, and J. Cheng, "Generative zero-shot network quantization," 2021.

[55] B. Zhu, P. Hofstee, J. Peltenburg, J. Lee, and Z. Alars, "Autorecon: Neural architecture search-based reconstruction for data-free compression," 2021.

[56] H. Li, X. Wu, F. Lv, D. Liao, T. H. Li, Y. Zhang, B. Han, and M. Tan, "Hard sample matters a lot in zero-shot quantization," 2023.

[57] Y. Jeon, C. Lee, and H. young Kim, "Genie: Show me the data for quantization," 2023.

[58] A. Rogalski, "Next decade in infrared detectors," in *Electro-Optical and Infrared Systems: Technology and Applications XIV* (D. A. Huckridge, R. Ebert, and H. Bürsing, eds.), vol. 10433, p. 104330L, International Society for Optics and Photonics, SPIE, 2017.

[59] K. I. Danaci and E. Akagunduz, "A survey on infrared image and video sets," 2023.

[60] N. Bustos, M. Mashhadi, S. K. Lai-Yuen, S. Sarkar, and T. K. Das, "A systematic literature review on object detection using near infrared and thermal images," *Neurocomputing*, vol. 560, p. 126804, 2023.

[61] H. Zhang, E. Fromont, S. Lefevre, and B. Avignon, "Multispectral fusion for object detection with cyclic fuse-and-refine blocks," 2020.

[62] C. Devaguptapu, N. Akolekar, M. M. Sharma, and V. N. Balasubramanian, "Borrow from anywhere: Pseudo multi-modal object detection in thermal imagery," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, June 2019.

[63] Y. Binbin, "An improved infrared image processing method based on adaptive threshold denoising," *EURASIP Journal on Image and Video Processing*, vol. 2019, p. 5, Jan 2019.

[64] J. Rauch, C. Doer, and G. F. Trommer, "Object detection on thermal images for unmanned aerial vehicles using domain adaption through fine-tuning," in *2021 28th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*, pp. 1–4, 2021.

[65] Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, "Low-bit quantization of neural networks for efficient inference," 2019.

[66] J. L. McKinstry, S. K. Esser, R. Appuswamy, D. Bablani, J. V. Arthur, I. B. Yildiz, and D. S. Modha, "Discovering low-precision networks close to full-precision networks for efficient embedded inference," 2019.

[67] F. A. Group, "Flir thermal dataset for algorithm training." `https://www.flir.com/oem/adas/adas-dataset-form/`, 2018.

[68] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[69] Nvidia, "Nvidia/tensorrt."

[70] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[71] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Accurate post training quantization with small calibration sets," in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 4466–4475, PMLR, 18–24 Jul 2021.

[72] Z. Liu, Y. Wang, K. Han, S. Ma, and W. Gao, "Post-training quantization for vision transformer," 2021.