Lightweight Deep Learning Model for Nondestructive Evaluation of Crack Defects

by

Yixiang Jia
B.Sc., Beihang University, 2022

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Lightweight Deep Learning Model for Nondestructive Evaluation of Crack Defects

by

Yixiang Jia
B.Sc., Beihang University, 2022

Supervisory Committee

Dr. Daler N. Rakhmatov, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Stephen W. Neville, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Ultrasonic nondestructive evaluation (NDE) is an essential tool in various industries including aerospace, energy, and civil engineering, for assessing the structural integrity of manufactured products without damaging them. This thesis is focused on the automated analysis of ultrasonic NDE data by means of low-cost machine learning (ML) techniques, particularly in the context of inline pipeline inspection. We propose two lightweight neural network architectures for efficient multi-attribute classification to characterize surface-breaking crack defects in terms of their location, size, and tilt. Our networks have under 2M parameters and incorporate novel design elements inspired by the latest MobileNet models. Their computational footprint is also small, not exceeding 100M floating-point operations (FLOPs) per data sample. The proposed models process raw channel data acquired by a transducer array, as opposed to multi-view beamformed image patches utilized in related works, thus eliminating the computational burden associated with image reconstruction. Our evaluation results, based on a public-domain NDE dataset, demonstrate that our networks offer a balanced combination of their competitively high classification performance and low cost. These findings highlight the potential of lightweight deep learning models in ultrasonic NDE data analysis, which contributes to the development of more advanced and intelligent inspection systems.

Contents

List of Tables

List of Figures

List of Acronyms

CH	Classification Head
CNN	Convolutional Neural Network
DW	Depthwise
EDM	Electrical Discharge Machining
EMAC	Efficient Multi-Attribute Classification
ExtraDW	Extra Depthwise
FFN	Feed-Forward Network
FLOPs	Floating Point Operations Per Second
GAP	Global Average Pooling
GT	Ground Truth
IB	Inverted Bottleneck
ML	Machine Learning
MQA	Multi-Query Attention
MFL	Magnetic Flux Leakage
NAS	Neural Architecture Search
NDE	Nondestructive Evaluation
PW	Plane Wave
PWC	Plane Wave Capture
PWI	Plane Wave Imaging
SE	Squeeze-and-Excitation
UIB	Universal Inverted Bottleneck
ViT	Vision Transformer

ACKNOWLEDGEMENTS

First and foremost, I would like to express my most sincere gratitude to my supervisor, **Dr. Daler N. Rakhmatov**, for his mentorship, enthusiasm, and encouragement throughout all stages of my study and research. Without his invaluable guidance and inspiration, this thesis would not have been possible.

I would also like to thank **Dr. Stephen W. Neville** (my supervisory committee member) for his support and insightful feedback, which have contributed to the improvement of this thesis.

Lastly, I would like to thank my family and friends for their love, understanding, and support along the way.

DEDICATION

To my mother, for her unwavering support throughout my studies abroad.

Chapter 1

Introduction

This thesis focuses on the application of advanced machine learning (ML) techniques to the analysis of ultrasonic nondestructive evaluation (NDE) data. Ultrasonic NDE is a critical technology used across various industries, including aerospace, automotive, civil engineering, and manufacturing, to assess the integrity and properties of materials and structures without damaging a sample being tested. The primary objective of this work is to leverage ML capabilities to improve the accuracy and efficiency of automated data analysis tasks arising in the context of ultrasonic NDE.

The material presented in this introductory chapter is based on [41], providing a concise overview of the industrial context in which this research is applied. It begins by discussing the significance of ultrasonic NDE in maintaining operational safety and quality of critical infrastructure components, such as pipelines. The chapter then delves into the fundamental principles of machine learning, elucidating how these techniques can be tailored to address specific challenges inherent in NDE data analysis, such as feature extraction and defect classification.

Additionally, this chapter reviews the current state-of-the-art in ML applications within the field of NDE, highlighting recent advancements and identifying existing gaps that this thesis aims to address. By combining domain knowledge of ultrasonic NDE with the latest developments in ML, this research contributes to the development of more intelligent and efficient automated inspection systems.

1.1 Ultrasonic Pipeline Inspection

The purpose of NDE techniques is to assess the health of a component under test without causing any damage to it. The basic principle of ultrasonic NDE considered in this thesis involves three basic steps: generating of an excitation stimulus (e.g., steered plane wave pulses) used to image the component, recording the returning echoes that constitute the transient response, and the subsequent analysis of that response to infer the component's integrity.

One of the major ongoing industrial applications of NDE is the inspection of oil and gas pipelines. This is typically achieved using mechatronic tools that travel inside the pipeline, assessing the integrity of the pipe's walls—a process commonly referred to as 'inline pipe inspection'. In this context, characterization of surface-breaking crack defects from ultrasonic inline pipe inspection data serves as a primary example of the industrial application for the research presented in this thesis.

The importance of NDE in the oil and gas industry cannot be overstated due to the scale of operations and the potential damage caused when defects are not detected in time. The global network of oil and gas pipelines extends over 2 million kilometers, with many individual pipelines spanning hundreds of kilometers, and some exceeding 3000 kilometers in length [14]. Environmental factors and in-service stresses can lead to corrosion and cracking, which, if not detected and repaired promptly, can result in significant spills. For instance, the Colonial pipeline oil spill in 2020, caused by a crack in the pipe wall, resulted in the release of at least 1.4 million gallons of oil, causing extensive environmental damage and incurring a clean-up cost exceeding $55 million [39].

Inline inspection plays a crucial role in preventing oil and gas leaks. One of the earliest inline inspection techniques, magnetic flux leakage (MFL), has been in use since the 1960s [24]. MFL involves applying a strong magnetic field to a ferromagnetic material and monitoring changes in the field to detect discontinuities. In recent years, ultrasound has become increasingly prevalent in inline pipe inspection, due to its ability to produce imaging data suitable for defect detection and sizing [77, 46].

Ultrasonic data acquisition can be carried out by a cylindrical device known as a 'pig', which is transported by the flow of the product while imaging the surrounding pipe structure. For example, the UltraScan™ DUO device (Baker Hughes Co.) typically inspects every 2 mm of the pipe, and to maintain a high oil flow rate, it travels at approximately 2 m/s [46].

To maximize the inspection coverage, a circumferential ring of transducer arrays is mounted on the pig. Each array is capable of transmitting a sequence of several plane wave pulses at different steering angles at every inspection location, with all transducer elements activated during reception. The high rate of such transmit-receive events taking place along the pipe quickly generates a vast amount of data. Each plane wave produces approximately 140 kB of relevant time-domain data (e.g., about 270 MB/s from each array [41]).

While it may be possible to perform on-device detection of crack defects, their sizing remains a challenge due to the on-device storage size and computational speed limitations. This has led to the practice of discarding most of the received signals, retaining only one peak amplitude per plane wave, which allows for basic amplitude-based sizing [46, 3, 26]. However, this approach can fail to accurately size tilted cracks, as their amplitude response can be significantly affected by small changes in the crack tilt [32].

Recent advancements in compression software, data storage, and computational hardware have made it increasingly feasible to store full time-traces, enabling offline plane wave imaging (PWI) [45, 28] and the application of image-based sizing algorithms. These algorithms can be based on physical principles (e.g., the 6 dB drop method [75]) or learned from data, offering enhanced defect detection and sizing capabilities. The main direction of our research efforts presented here is to explore lightweight data-driven ML techniques for classifying the size, tilt, and location of a detected crack defect. Our emphasis on lightweight models is motivated by the goal of enabling such defect characterization on the resource-constrained inspection device itself, as opposed to offline processing. It is important to mention that this work does not deal with the task of detecting the presence of a crack. In other words, our proposed models are applied at the next step (defect characterization) after a crack has already been detected by some other methods.

1.2 Machine Learning in Nondestructive Evaluation

NDE data analysis has traditionally relied on skilled operators. However, as the dimensionality of NDE data increases and the frequency of inspections rises, manual interpretation becomes slow, expensive, and susceptible to human error. The chal-

lenges associated with cost, complexity, speed, and inconsistency in human analysis highlight the need for automated methods.

Currently, many automated NDE data analysis methods are based on physics, empirical rules, or experimental thresholds. For instance, the 6 dB drop method is commonly used to size defects from reconstructed (beamformed) images. It estimates the defect's extent as the smallest box that encloses the pixels within the top 6-dB intensity range. While traditional approaches like these are effective when relationships between imaging data and the estimated quantity are relatively simple, they often fail to capture more complicated correspondence patterns. Analyzing high-dimensional, complex NDE data is fundamentally a pattern recognition task, making it well-suited for machine learning (ML). As the field advances towards 'NDE 4.0' [37] integrating cyber-physical systems and the Internet of Things, ML is becoming an increasingly popular method for processing the vast amounts of data generated [68, 5, 6].

ML encompasses a wide range of techniques aimed at learning functions from data. It has been demonstrated to achieve human-level interpretation performance in NDE [62, 1, 38, 30, 73, 66, 49, 61, 51] as well as in the traditional ML research fields of computer vision [67] and medical imaging [33, 53]. This thesis focuses exclusively on supervised learning, which involves learning from labeled data. In the context of our target application, it means that all training data is associated with three known ground-truth characteristics of a crack defect: size, tilt, and location.

ML techniques are further categorized into 'deep' and 'shallow' learning. Shallow learning in NDE dates back to the 1990's with the use of decision trees to detect defects based on ultrasonic wall reflection amplitude loss [40], and it remains an active research area [17]. For example, neural networks have been applied to traditional ultrasonic measurement features to estimate material properties [1, 38] and classify defects [62, 49], while support vector machines and random forests have been used to size cracks from eddy current field peaks [4] and detect defects in fluorescent penetrant inspection images [51], respectively.

While the transition to deep learning is well underway in medical imaging [53], its application in NDE is limited by the high cost of generating experimental training sets and the challenge of qualifying 'black box' algorithms. Data augmentation techniques can expand training set sizes, but they must produce realistic examples. In some specialized cases, a sufficient number of defect samples can be available to create adequate training sets [13, 8], or the ML task scope can be restricted to reduce training set requirements [36]. However, these approaches do not provide a general-

purpose solution for the shortage of large NDE training sets or address concerns about qualifying 'black box' algorithms.

1.3 Thesis Contribution and Organization

Despite the significant advancements in the field of nondestructive evaluation (NDE) using deep learning techniques, the application of these methods to ultrasonic imaging still faces several challenges. These challenges can be categorized into four main areas:

1. Scarcity of Ultrasonic Imaging Data: The high cost of acquiring ultrasonic imaging data results in relatively small datasets. Additionally, employing artificial (i.e., simulated or synthesized) data may not be a robust option for supplementing real-world data because of the latter's complexity. Training a model with strong generalization capabilities on such limited data is particularly challenging.

2. Black Box Nature of Deep Learning Models: Deep learning models often suffer from poor interpretability. This lack of transparency is further exacerbated by the small dataset sizes, making it crucial yet difficult to analyze and interpret the model's behavior.

3. Special Characteristics of Ultrasonic Imaging Data: The data processing methods for ultrasonic NDE typically involve generating multiple views of an imaged section from the same collected raw data, taking into account multiple wave propagation phenomena, such as reflections, refractions, diffractions, and mode conversions. Due to the complexity of both raw and beamformed data, defect analysis can be difficult.

4. Hardware Resource Constraints of Ultrasonic Equipment: Ultrasonic field equipment is often limited in terms of available computational resources. Designing deep learning models that consider computational complexity and memory efficiency is essential for the deployment of algorithms on such hardware. Lightweight deep learning models are therefore of paramount importance for practical ultrasonic NDE applications.

To address the last two challenges, this thesis proposes two lightweight networks, EMACnet and EMACnet++, inspired by the recent advances in MobileNet architecture designs. The abbreviation EMAC stands for Efficient Multi-Attribute Classification. Our models are tasked with classifying three attributes of an imaged surface-breaking crack defect, namely its location, size, and tilt (see Chapter 2). The ground truth for these attributes is organized into 27 location classes, 5 size classes, and 11 tilt classes (see Chapter 2). The relevant dataset has been created by the Ul-

trasonics and Non-destructive Testing (UNDT) Research Group at the University of Bristol (https://data.bris.ac.uk/data/dataset/2o82rzo6d5ly32h7msblzq4y8v) and used in several of their articles [42, 43, 41]. However, in our case, we use 561×40 raw channel data frames, as opposed to multi-view 32×32 beamformed image patches used in [42, 43, 41]. It should be noted that these patches (used to predict crack sizes and tilts) correspond to predetermined crack locations, whereas our models predict not only crack sizes and tilts, but also their locations. It is also important to note that in this work the term "prediction" refers to the result of producing a model output during classification of unseen data samples from the test portion of the utilized dataset.

The rest of this thesis is organized as follows. Chapter 2 provides a brief background on the utilized ultrasound dataset and existing lightweight network examples. Chapter 3 presents the main contributions of this work, covering proposed network architectures and their general design principles. Chapter 4 discusses the training setup and compares our EMACnet and EMACnet++ models to several competitors, including evaluation results for both prediction accuracy and computational cost. Finally, Chapter 5 concludes the thesis and offers perspectives for future work.

Chapter 2

Background

2.1 Experimental Setup

This chapter describes the experimental setup used in [41] for ultrasonic NDE data acquisition and the resulting dataset used in this thesis. We also briefly review of several representative deep learning models that have inspired the development of our own lightweight model presented in Chapter 3.

2.1.1 Data Acquisition

Due to the complex physical and chemical operational conditions encountered during pipeline lifetime, various defects such as corrosion and cracks can develop on the pipe's inner and outer walls. This study aims to utilize raw ultrasound data to determine critical information such as the size, location, and tilt of surface-breaking cracks on the outer wall, thus facilitating timely pipeline maintenance and repair. Given the difficulty in obtaining representative experimental data from actual pipelines, our work follows [43, 42] in adopting a pipeline inspection model that emulates real-world NDE data collection. The setup of this model is illustrated in Figure 2.1 from [41].

Due to the short distance between the device and the pipe's inner wall (in relation to the radius of the latter), the ultrasonic NDE data can be approximated by the experimental data acquired from an immersed stainless steel plate. The inspection setup employs a 5 MHz phased array with 40 elements, each spaced 0.3 mm apart. Three plane waves are emitted at steering angles 0° and $\pm 19^\circ$ in water, producing longitudinal waves at 0° and shear waves at $\pm 45^\circ$ in steel, respectively. Each element records its own signal time series during reception, forming full plane wave

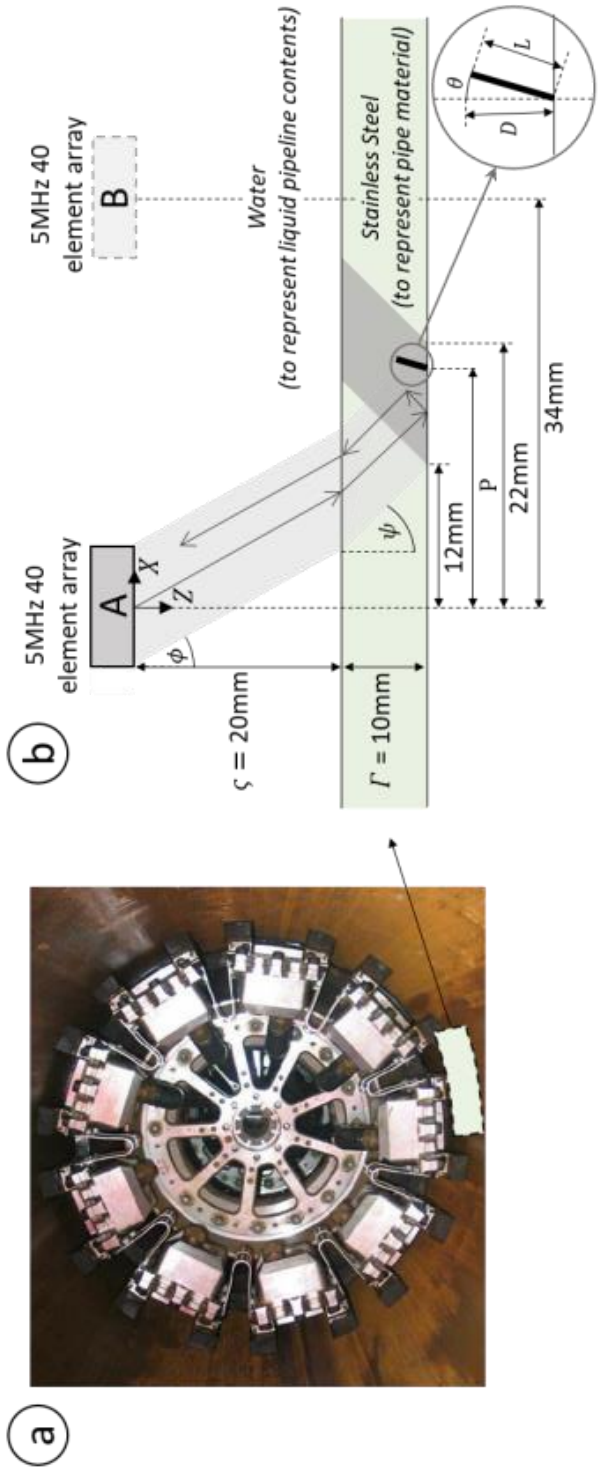


Figure 2.1: Ultrasonic Inspection Model from [41]: a) Front View of an Inspection Device, b) Illustration of Data Acquisition.

capture (PWC) data. The two nonzero-angle plane waves are used to characterize crack defects, while the zero-angle plane wave is used only to determine the standoff distance (nominally 20 mm) and the specimen's thickness (nominally 10 mm). The sound speeds were calculated using a calibration sample of known thickness at known standoff, which produced a longitudinal speed of 5759 m/s in steel, a shear speed of 3165 m/s in steel, and a speed of 1480 m/s in water. The PWC data is collected from either side of the defect to simulate data acquisition from the circumferential ring of arrays on the inspection device, as shown in Figure 2.1.

All experimental crack defects have been manufactured using electrical discharge machining (EDM). This method produces cracks having the width of 0.3 mm and controlled size (length) L, tilt θ, and location (lateral position) P, as illustrated in Figure 2.1.

2.1.2 Utilized Dataset

While both simulated and experimental data are used in the study by Pyle et al. [43], this thesis focuses solely on the real-world experimental data, which is more challenging due to the realistic noise presence and smaller data volume. The total number of experimental data samples is 1485, covering five sizes L (1, 2, 3, 4, 5 mm), eleven tilts θ (0°, ±2°, ±5°, ±8°, ±15°, ±20°), and 27 locations P ranging from 13.2 mm to 21.0 mm and spaced at the regular intervals of 0.3 mm.

Typically, crack characterization begins with acquiring raw data from the transducer array, followed by beamforming to produce images representing various reconstruction modes (views), and subsequent image analysis. Although this is a well-established method, the beamforming process itself can be time consuming and prone to errors due to transducer misalignment, speed-of-sound mismatch, noise, and other factors. Hence, our approach is to analyze the raw channel data directly, which means that the beamforming step is no longer necessary. This approach reduces the data processing burden, which is crucial for devices with limited computational and storage resources.

2.2 Deep Learning

2.2.1 Traditional Convolutional Networks

Since AlexNet[27] won the ImageNet competition in 2012, convolutional neural networks (CNNs) have become a focal point of research in deep learning and computer vision due to their superior performance across various tasks. For instance, VGGNet[52] highlighted the advantages of deeper networks with simpler structures. GoogLeNet[54] introduced the Inception module, which enables multi-scale feature extraction while reducing computational complexity. ResNet[18] tackled the gradient degradation problem in deep networks with residual connections, allowing for the training of deeper architectures. DenseNet[22] enhanced feature propagation and reuse by densely connecting layers. SENet[21] introduced the Squeeze-and-Excitation (SE) structure to recalibrate channel-wise feature responses, enhancing representational power. EfficientNet[57] proposed a compound scaling method to uniformly scale all dimensions of depth, width, and resolution, improving performance with fewer parameters. More recently, ConvNeXt [31] modernized the standard CNN architecture by integrating design principles from Vision Transformers (ViTs) [12], achieving competitive performance on various benchmarks. These advancements have collectively shaped the evolution of CNNs, enhancing their efficiency, scalability, and performance across numerous tasks.

Although deep CNNs have achieved numerous successes, their high computational and memory requirements pose challenges for training and deployment on resource-constrained devices. Consequently, extensive research has focused on designing new lightweight network architectures to reduce computational and memory requirements while maintaining high performance.

2.2.2 Mobile Convolutional Networks

Major mobile network architectures include the MobileNet series. MobileNetV1[20] employed depthwise separable convolutions to maintain good performance with fewer parameters and computations. MobileNetV2[50] introduced the linear bottleneck structure, decoupling the network's capacity and expressiveness, further improving performance and interpretability. MobileNetV3[19] incorporated the SE module for attention mechanisms and optimized the network structure through neural architecture search, enhancing performance. MobileNetV4[44] unified Inverted Bottleneck

(IB), ConvNeXt, FFN, and Extra Depthwise (ExtraDW) IB blocks through the Universal Inverted Bottleneck (UIB) module and introduced the Multi-Query Attention (MQA) mechanism for better adaptation to lightweight devices. Meanwhile, MobileOne[64] reduced network parameters and improved performance through structural re-parameterization. MobileDet[72] demonstrated the effectiveness of regular convolutions (as opposed to separable convolutions) across various mobile accelerators.

2.2.3 Hybrid Convolutional Networks

Since the advent of ViT[12] and DETR[7], transformer attention mechanisms and encoder-decoder structures have been gradually introduced into computer vision. However, for lightweight network models, transformers' computational and memory overheads are substantial. Researchers have begun to integrate transformer attention mechanisms and encoder-decoder structures with mobile CNNs. For example, MobileViT[35] applied global attention blocks within the ViT framework. MobileFormer[9] bridged transformer and MobileNet structures through a novel bridge design. FastViT[63] combined large convolutional kernels and attention mechanisms to enhance network performance. EfficientFormerV2[29] explored the latency-performance trade-off with a more efficient dimensional consistent structure.

2.2.4 Other Lightweight Networks

In addition to the aforementioned MobileNet series and some lightweight ViT variants, other network designs have also received widespread attention. For example, ShuffleNet [76] achieves efficient and cost-effective channel information exchange through group convolution and channel shuffling. ShuffleNetV2 [34] introduces additional design enhancements for ShuffleNet and proposes four principles for designing lightweight networks. EfficientNet [57] improves performance through a compound scaling method that uniformly scales the network's depth, width, and resolution. MnasNet [58] proposes a lightweight network design method through automated neural architecture search. GhostNet [16] achieves efficient feature reuse by introducing ghost modules.

2.2.5 Summary

In summary, deep learning has made significant progress in the field of computer vision, with CNNs serving as the backbone for various tasks. Traditional CNNs have evolved from AlexNet to ConvNeXt, with advancements in depth, width, and resolution scaling, feature reuse, and recalibration mechanisms. Mobile convolutional networks have been developed to address the computational and memory limitations of resource-constrained devices, with the MobileNet series being the most representative. Hybrid convolutional networks have emerged to integrate transformer attention mechanisms and encoder-decoder structures lightweight CNNs. Automatic neural architecture search has also been employed to reduce computational and memory requirements while maintaining high performance. These advancements have collectively shaped the evolution of deep learning, enhancing the efficiency, scalability, and performance of CNNs across various tasks.

Chapter 3

Proposed Models

3.1 Hardware Considerations

The performance of CNNs in terms of inference speed is critically dependent on the specific hardware platforms employed, such as NVIDIA GPUs, ARM embedded processors for mobile devices, general-purpose CPUs, and edge NPUs. Key factors influencing CNN inference speed include the computational complexity of operations (measured in Floating Point Operations, FLOPs), the number of parameters (Params), the memory access costs associated with convolutional blocks (memory bandwidth), and the inherent parallelism within the network. It is important to recognize that on a given hardware platform with a fixed network architecture, the percentage of accelerated FLOPs directly affects the inference time improvements, while having fewer FLOPs does not necessarily guarantee faster inference speed, particularly on hardware platforms equipped with acceleration capabilities like GPUs.

In [72], it was found that models with the number of channels being a multiple of 16 are relatively hardware-friendly in mobile computing environments. In terms of convolutional design, 3x3 convolutions exhibit a computational density four times greater than that of 1x1 and 5x5 convolutions[11]. Depthwise separable convolutions, despite having fewer FLOPs, are less efficient on hardware with robust parallel computing capabilities due to their intensive data read-write operations. Thus, depthwise separable convolutions are generally more suitable for non-GPU hardware (e.g., ARM processors), where the available memory bandwidth is a closer match for the amount of available FLOP parallelism. Their benefits in terms of lower parameter count and lower computational complexity make such convolutions desirable in

hardware-constrained designs[70, 72]. Overall, VGG-like convolutional architectures and highly parallel structures are better-suited for leveraging GPU computing power, while serialized architectures impose reduced memory bandwidth demands. Notably, in the initial layers of lightweight networks with low channel counts, standard convolutions outperform depthwise separable convolutions on hardware with acceleration capabilities[72].

In conclusion, the design of lightweight networks must account for factors such as the total number of FLOPs, the percentage of parallelizable FLOPs, memory bandwidth, and hardware acceleration capabilities to optimize inference speed performance. This work focuses primarily on the application of these principles using GPU-like hardware platforms.

3.2 Proposed Network Design

We propose two networks in this thesis: EMACnet and EMACnet++. The former is a lightweight model inspired by MobileNetV3 and MobileNetV4, while the latter (based on the same design principles) incorporates additional architectural enhancements. Tables 3.1 and 3.2 with Figures 3.1 and 3.2 summarize the overall architecture of EMACnet and EMACnet++, respectively. Both networks have five sequential stages: Stem, Backbone, Feature Expansion, Pooling, and Classification. The last stage is a collection of classification heads operating in parallel. In the context of our target application, we designate one classification head (CH) for each of the three attributes characterizing a crack defect: location, size, and tilt. This section presents the network architecture design of EMACnet, including the overall design concept, the details of each stage, and the parameters of key building blocks.

3.2.1 Universal Inverted Bottleneck and Fused Inverted Bottleneck

The bottleneck structure was first proposed as part of ResNet [18], introducing 1x1 convolution layers to reduce the number of parameters and computational complexity. MobileNetV2 [50] introduced the Inverted Bottleneck (IB) structure, which improves the network's representational capacity by first expanding and then compressing the number of channels. MobileDets [72] reconsidered the design of the IB block, pointing out that the Fused IB layers and Tucker Convolution layers could improve network

Table 3.1: EMACnet Architecture

Stage	Module	Block	In Channels	Out Channels	Kernel	DW1 Kernel	DW2 Kernel	Exp. Ratio	Stride
Stem	Conv0	ConvBn	2	32	3x3	-	-	-	2x2
	Layer1	ConvBn	32	64	3x3	-	-	-	2x2
		ConvBn	64	32	1x1	-	-	-	1x1
	Layer2	ConvBn	32	96	3x3	-	-	-	2x2
		ConvBn	96	64	1x1	-	-	-	1x1
Backbone	Layer3	ExtraDW	64	64	-	3x3	3x3	3	2x2
		IB	64	64	-	-	3x3	2	1x1
		IB	64	64	-	-	3x3	2	1x1
	Layer4	ExtraDW	64	64	-	3x3	3x3	6	2x2
		IB	64	64	-	-	3x3	4	1x1
		ConvNeXt	64	128	-	3x3	-	4	1x1
Feature Expansion	Layer5	ConvBn	128	960	1x1	-	-	-	1x1
		ConvBn	960	1280	1x1	-	-	-	1x1
Pooling	GAP	Global Average Pooling	1280	1280	-	-	-	-	-
Classification	Location CH	Linear	1280	27	-	-	-	-	-
	Size CH	Linear	1280	5	-	-	-	-	-
	Tilt CH	Linear	1280	11	-	-	-	-	-

Table 3.2: EMACnet++ Architecture

Stage	Module	Block	In Channels	Out Channels	Kernel	DW1 Kernel	DW2 Kernel	Exp. Ratio	Stride
Stem	conv0	ConvBn	2	32	3x3	-	-	-	2x2
	Layer1	ConvBn	32	32	3x1	-	-	-	2x1
		ConvBn	32	32	3x1	-	-	-	2x1
		ConvBn	32	64	3x3	-	-	-	2x1
	Layer2	ConvBn	64	64	3x3	-	-	-	2x2
		ConvBn	64	128	3x3	-	-	-	2x2
Backbone	Layer3	ExtraDW++	128	128	-	3x3+1x1	3x3	4	1x1
		ExtraDW++	128	128	-	3x3+1x1	3x3	2	1x1
		ExtraDW++	128	128	-	3x3+1x1	3x3	2	1x1
		ConvNext++	128	128	-	3x3+1x1	-	2	1x1
		FFN++	128	128	-	-	-	4	1x1
		ConvNext++	128	128	-	3x3+1x1	-	2	1x1
	Layer4	ExtraDW++	128	128	-	3x3+1x1	3x3	4	1x1
		ExtraDW++	128	128	-	3x3+1x1	3x3	2	1x1
		ExtraDW++	128	128	-	3x3+1x1	3x3	2	1x1
		ExtraDW++	128	128	-	3x3+1x1	3x3	2	1x1
		ConvNext++	128	128	-	3x3+1x1	-	2	1x1
		FFN++	128	128	-	-	-	4	1x1
		FFN++	128	128	-	-	-	4	1x1
		ConvNext++	128	128	-	3x3+1x1	-	2	1x1
Feature Expansion	Layer5	ConvBn	128	960	1x1	-	-	-	1x1
		ConvBn	960	1280	1x1	-	-	-	1x1
Pooling	GAP	Global Average Pooling	1280	1280	-	-	-	-	-
Classification	Location CH	Linear	1280	27	-	-	-	-	-
	Size CH	Linear	1280	5	-	-	-	-	-
	Tilt CH	Linear	1280	11	-	-	-	-	-

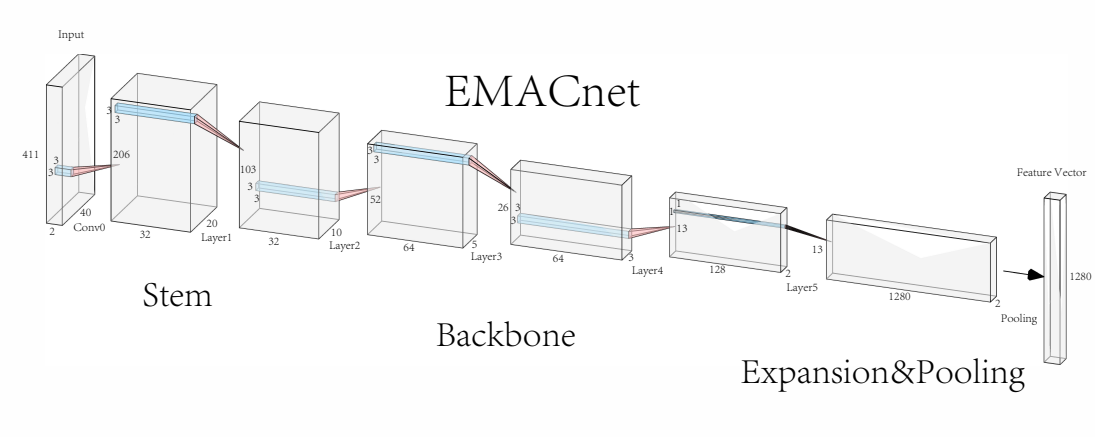


Figure 3.1: EMACnet Architecture: Stem, Backbone, Feature Expansion and Pooling stages. The Classification stage is not shown.

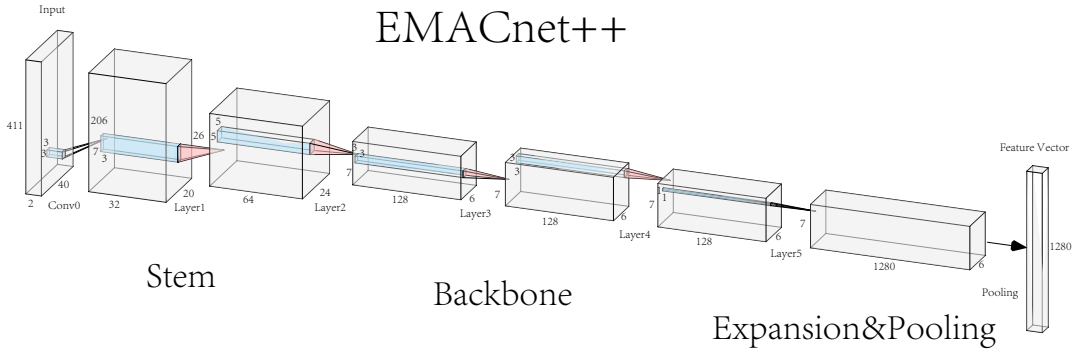


Figure 3.2: EMACnet++ Architecture: Stem, Backbone, Feature Expansion and Pooling stages. The Classification stage is not shown.

performance. ConvNeXt [31] achieved better performance by placing extra depthwise separable convolution (ExtraDW) layers before the feed-forward network (FFN) layers. In MobileNetV4 [44], the Universal Inverted Bottleneck (UIB) block unified the ExtraDW, IB, ConvNeXt, and FFN structures, as shown in Figure 3.3. Additionally, MobileNetV4 uses a Fused IB block as the network's stem, similar to the MobileDets design.

The EMACnet model adopts the UIB block as a basic building cell of the network. In MobileNetV4 [44], the optimal network structure was obtained through Neural Architecture Search (NAS) [79]. However, this method requires extensive computational resources. Here, we analyze the UIB design concepts and combine them with the design principles from other models to construct a network structure suitable for our target application.

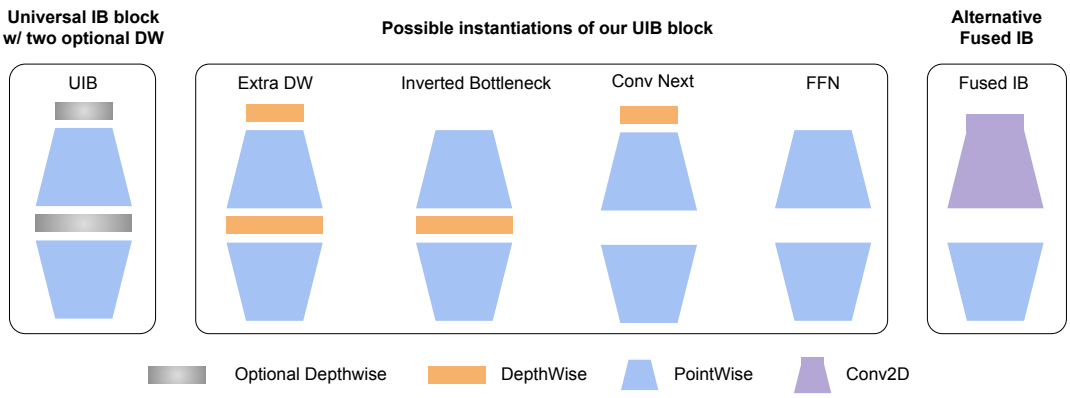


Figure 3.3: Universal Inverted Bottleneck (UIB) Block and Its Various Configurations from [44]

3.2.2 Stem

The stem part of our network is the entry point, responsible for initial feature extraction and processing of the input data. In MobileNetV4 [44], NAS found that the stem part of small networks is more suitable for implementation with a Fused IB block (see Figure 3.3) as opposed to UIB. MobileDets [72] also found that in few-channel situations (e.g., we start with only two input channels in our case), using standard convolution on hardware devices with acceleration capabilities is more efficient than depthwise separable convolution.

Therefore, the Stem stage of the EMACnet model uses the Fused IB structure, as shown in Table 3.1 and Figure 3.4(a). It has three modules: Conv0, Layer1, and Layer2. The first one is a convolution layer with 2 input channels, 32 output channels, a 3x3 convolution kernel, and a stride of 2x2, responsible for performing initial feature extraction and processing. Layer1 consists of a convolution layer with 32 input channels, 64 output channels, a 3x3 convolution kernel, and a stride of 2x2, followed by a convolution layer with 64 input channels, 32 output channels, a 1x1 convolution kernel, and a stride of 1x1. Layer2 is similar to Layer1, consisting of a convolution layer with 32 input channels, 96 output channels, a 3x3 convolution kernel, and a stride of 2x2, followed by a convolution layer with 96 input channels, 64 output channels, a 1x1 convolution kernel, and a stride of 1x1. The two-layer convolutions in Layer1 and Layer2 follow the Fused IB design in Figure 3.3. In Table 3.1, the convolution layers forming Conv0, Layer1, and Layer2 are listed as ConvBn blocks, where Conv refers to Convolution, and Bn refers to Batch Normalization.

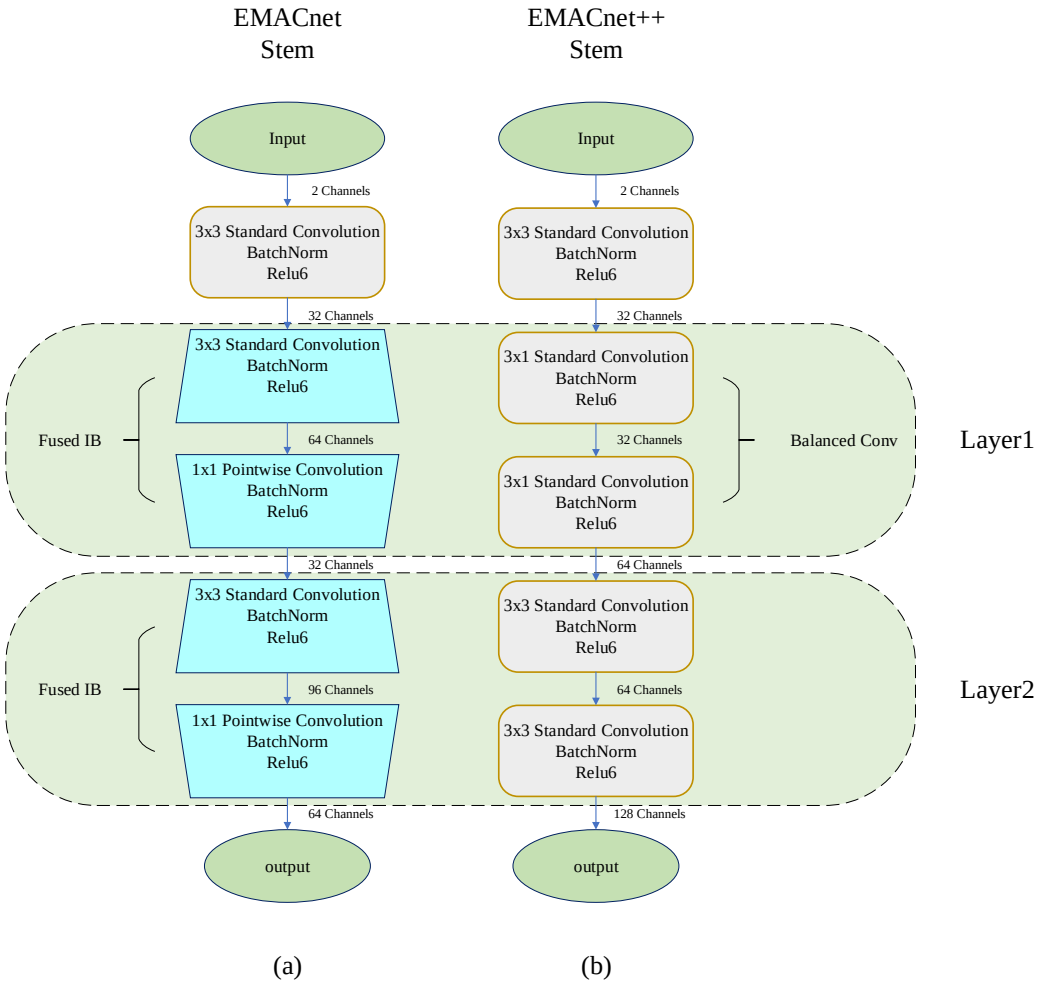


Figure 3.4: Stem Stage: (a) EMACnet, (b) EMACnet++

3.2.3 Backbone

The Backbone stage of the EMACnet model uses the UIB structure, and it consists of the Layer3 and Layer4 modules as specified in Figure 3.5 and Table 3.1, where DW1 and DW2 refer to two depthwise separable convolutions before and after expansion, respectively. In Layer3, the first block is ExtraDW responsible for extracting spatial information at the beginning of the stage and performing downsampling. This block uses two depthwise separable convolution layers with 64 input channels, 64 output channels, a 3x3 kernel size, a 2x2 stride, and a bottleneck expansion ratio of 3. The second block in Layer3 is an IB, responsible for processing of downsampled data, expanding the ExtraDW output, extracting features in a richer high-dimensional space,

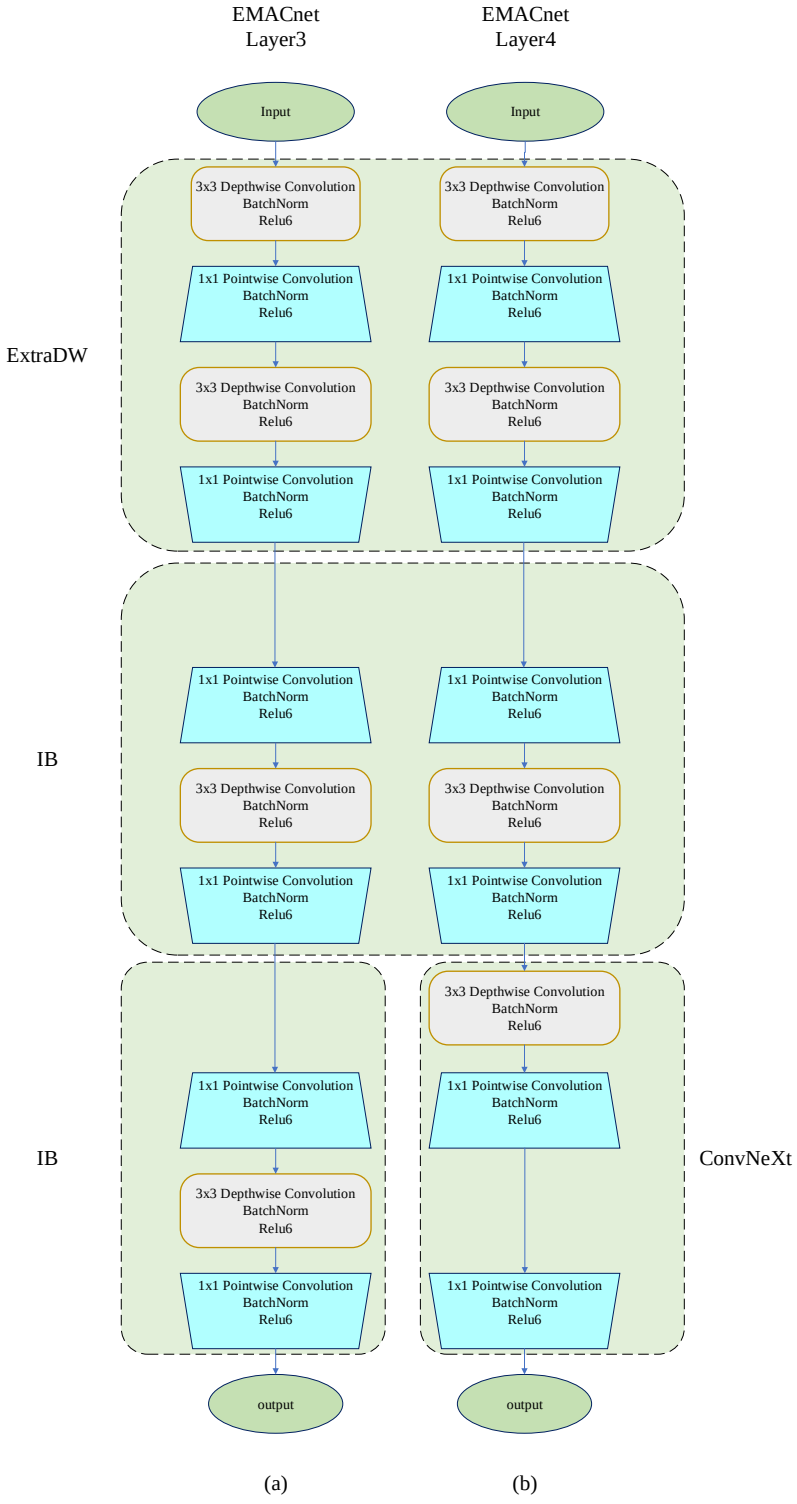


Figure 3.5: ECMACnet Backbone Stage: (a) Layer3, (b) Layer4

and then reducing the dimensions again. It has 64 input and 64 output channels, and it uses a 3x3 separable convolution kernel with a stride of 1x1 and a bottleneck expansion ratio of 2. The last Layer3 block is another IB that has the same configuration as the preceeding IB block.

In Layer4, the first and second block follow the same design as the first (ExtraDW) and second (IB) block of Layer3. However, due to the increased module depth, a larger expansion ratio is required to meet the feature extraction and processing needs. Consequently, the ExtraDW block of Layer4 has a bottleneck expansion ratio of 6, while the IB block of Layer4 has a bottleneck expansion ratio of 4. At the end of Layer4, the EMACnet model has a ConvNeXt block that implements the transition from spatial feature extraction to channel feature extraction.

3.2.4 Feature Expansion and Pooling

The Feature Expansion stage is responsible for further channel-level feature expansion, extraction, and processing of the data processed by the network backbone, facilitating subsequent classification of the individual attributes of interest. This stage is formed by Layer5 that consists of two 1x1 pointwise convolution layers: the first one has 128 input channels and 960 outputs channels, while the second one has 960 input channels and 1280 output channels.

To reduce the number of parameters and computational complexity, the EMACnet model relies on the Pooling stage that follows the Feature Expansion stage. A global average pooling (GAP) layer is used to compute the mean of the output feature map, obtaining a fixed-size feature vector.

3.2.5 Classification

In the context of our target application, there are three attributes associated with a crack defect: location, size, and tilt. Therefore, as shown in Table 3.1, we use three classification heads to predict which class (for a given attribute) an input data sample belongs to. The Location CH consists of a linear layer with 1280 input channels and 27 output channels (equal to the number of location classes). The Size CH consists of a linear layer with 1280 input channels and 5 output channels (equal to the number of size classes). The Tilt CH consists of a linear layer with 1280 input channels and 11 output channels (equal to the number of tilt classes).

3.3 Enhanced Network Design

In this section, we will introduce the network structure of EMACnet++, which is an improved version of EMACnet, designed to further enhance model performance. This section presents the design concepts, module composition, and working principles of EMACnet++, as well as the differences and advantages compared to EMACnet.

3.3.1 Stem Convolutions

Recall that in our case, the input samples are 2D raw channel data in the (t, x) domain, where t denotes time, and x refers to the sensor position. The nature of such data is closer to multivariate time series than to camera images. Each data frame in the utilized dataset is of size $N_t \times N_x = 561 \times 40$ (cropped to 411×40), i.e., its vertical and horizontal dimensions are highly imbalanced.

In response to these issues, the EMACnet++ model employs a different convolution configuration in Layer1 of the Stem stage, using a 3x1 kernel with a 2x1 stride (see Table 3.2 and Figure 3.4(b)) instead of a 3x3 kernel with a 2x2 stride in the original EMACnet case. Such convolutions appear to be better suited for our target application, which is supported by the experimental evidence presented in Chapter 4. Also note that Layer1 has three ConvBn blocks (instead of two) and yields 64 output channels (instead of 32).

Layer2 of the Stem stage in EMACnet++ is similar to that of EMACnet (except for minor adjustments in the convolution kernel size and stride); however, the number of input/output channels is doubled to enhance initial feature extraction prior to the Backbone stage processing.

3.3.2 General Design Concepts

In EMACnet, we adopted the Universal Inverted Bottleneck (UIB) block from MobileNetV4 [44] as a building cell for the model. As we demonstrate in Chapter 4, EMACnet achieved good performance on the targeted dataset when predicting crack size (5 classes) and tilt (11 classes); however, the accuracy of crack location predictions (27 classes) was low. Aside from having a relatively large location class size and small network size, one possible reason behind poor handling of this attribute could be global pooling that potentially compromises spatial information fidelity.

Generally, the spatial information extraction capability of a model includes both

absolute position information and relative position information. Since we require the network to be lightweight, it is difficult to replace the global pooling layer with another competitive alternative. Hence, given the imbalanced aspect ratio and the presence of global pooling, the model needs to learn the relative position information between different regions to compensate. In [74], it has been noted that the separation of the channel and token mixers is key to the excellent performance of transformer-based models like ViT, DeiT, and ResMLP [12, 60, 59]. We have also adopted such a separation mechanism, aiming to improve the spatial information extraction ability of the EMACnet++ model.

In [54, 23, 55, 56, 10], the design concept of the Inception structure that concatenates feature maps of different sizes of convolution kernels has been shown to be beneficial for feature learning in a network. We have adopted this concept as well, by concatenating feature maps of differently sized convolution kernels.

Furthermore, based on [21], EMACnet++ also incorporates the Squeeze-and-Excitation (SE) layer as an efficient attention mechanism. Although there are various attention mechanisms, such as Transformer and CBAM [65, 71, 15], practical experience suggests that with a small dataset size and a small number of model parameters, the SE structure is not only simple and easy to train but also more effective. Next, we describe a novel block that combines aforementioned ideas into a new building cell used by the Backbone stage of EMACnet++.

3.3.3 EMACblock++

EMACblock++ is the core cell of EMACnet++, and its structure is illustrated in Figure 3.6. It is inspired by a Universal IB, and it can take three forms, as shown in Figure 3.7. They are enhanced versions of the FFN, ConvNeXt, and ExtraDW blocks derived from our EMACblock++. We refer to these configurations as FFN++, ConvNeXt++, and ExtraDW++, respectively. By comparing our EMACblock++ with an UIB, one can see that the first layer of the depthwise separable convolution is replaced by a combination of the 1x1 and 3x3 depthwise separable convolutions, followed by residual concatenation. The 1x1 depthwise convolution is a weighted feature of the spatial dimension, which can be seen as a spatial attention mechanism, whereas the 3x3 depthwise convolution is a larger feature extractor. Their combination is essentially the token mixer. Its output is supplied to the SE layer that implements a channel attention mechanism, enabling channel feature weighting and

facilitating the subsequent learning of the channel mixer. The channel mixing part is essentially an IB with an optional depthwise convolution layer, as depicted in Figure 3.6.

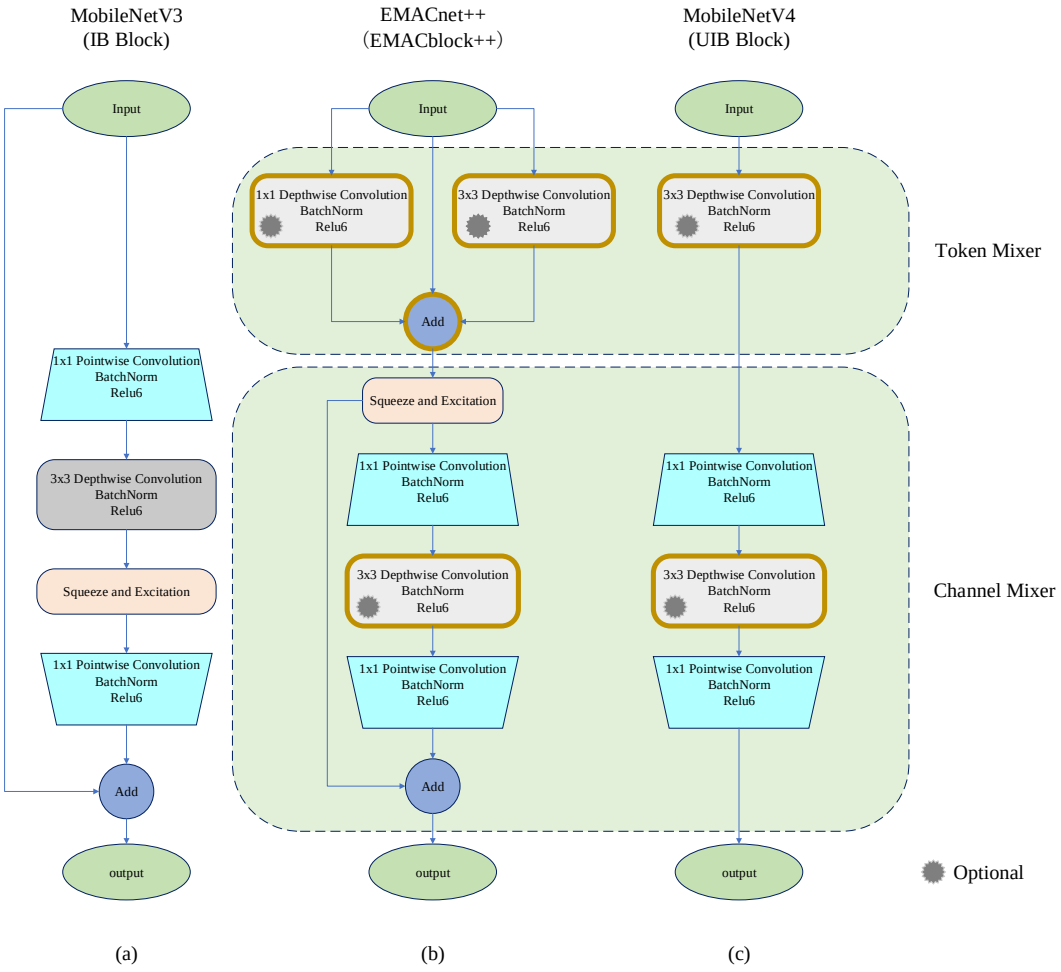


Figure 3.6: Comparison of MobileNetV3, EMACnet++, and MobileNetV4 Building Cells. (a) Inverted Bottleneck, (b) EMACblock++, (c) Universal Inverted Bottleneck.

The overall network structure of EMACnet++ is summarized in Table 3.2. Its organization is similar to that of EMACnet, but with one key architectural enhancement in the Backbone stage. Layer3 and Layer4 are now made of more complex EMACblock++ configurations (i.e., FFN++, ConvNeXt++, and ExtraDW++) that have been introduced to improve the feature extraction and fitting ability of the model. The goal of this network design is to facilitate enhanced learning of relative position information to mitigate the potential loss of absolute position information

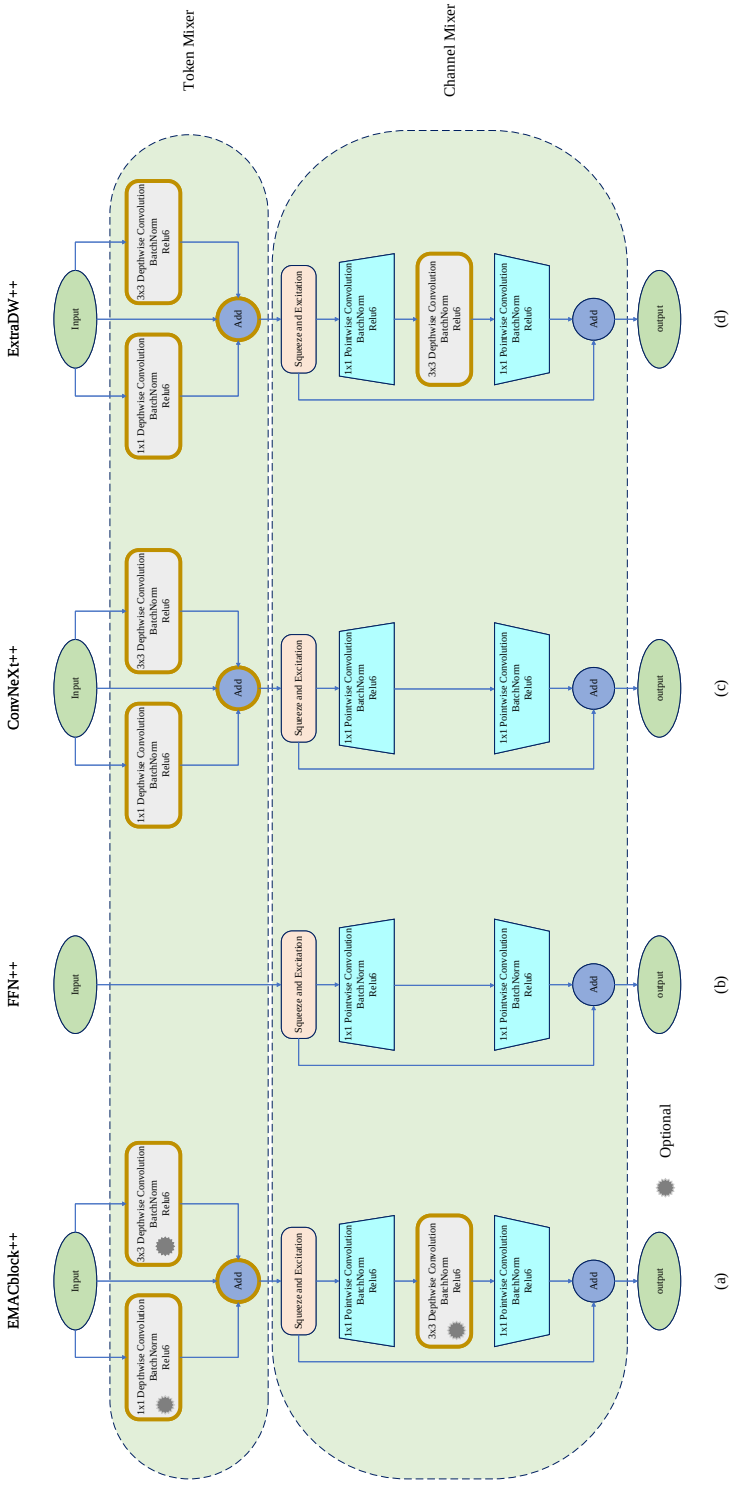


Figure 3.7: (a) General EMACblock++ Structure, (b) FFN++ Configuration, (c) ConvNeXt++ Configuration, (d) ExtraDW++ Configuration.

due to global pooling prior to multi-attribute classification. The next chapter describes our training settings and demonstrates that EMACnet++ does perform better than EMACnet in terms of crack location classification.

Chapter 4

Model Training and Evaluation

4.1 Training Setup

4.1.1 Data

This work relies on a publicly available dataset[43] providing raw channel data acquired during PW imaging of steel plate specimens (10-mm thickness), immersed in water, 20 mm away from the probe plane. A 40-element probe (0.3-mm pitch) was used to transmit two 5-MHz PWs (angled at $\pm 19°$) and to receive returning signals. Each acquisition (one per crack sample) produced two raw channel data frames of size $N_t \times N_x = 561 \times 40$, where x refers to a lateral coordinate of the sensor position, and t denotes a time instance under the sampling frequency of 16.7 MHz. The total number of acquisitions was 1,485, representing all combinations of 5 sizes L (1.0, 2.0, 3.0, 4.0, 5.0 mm), 11 tilts θ (0°, $\pm 2°$, $\pm 5°$, $\pm 8°$, $\pm 15°$, $\pm 20°$), and 27 locations P ($13.2 \leq P \leq 21.0$, equally spaced 0.3-mm apart), over all imaged crack variants. The acquisition setup is illustrated in Figure 2.1.

In terms of data preprocessing, we reduced the size of input data frames to 411×40 by removing the upper 150×40 section (i.e., early-arrival signals), which does not contain useful defect-related information. We also computed the envelope of each sensor's signal, and the resulting 2D data frame was normalized to restrict the numerical values to the $[0, 1]$ range. To summarize, our original preprocessed dataset consisted of 1,485 two-channel samples, where the first and second channels of each sample contained the corresponding 411×40 normalized envelopes of negative- and positive-angle raw channel data frames, respectively.

4.1.2 Multi-Attribute Classification

Given an input sample from the dataset described above, the model's task is to output the three target attributes of a crack defect: location (27 classes), size (5 classes), and tilt (11 classes). Such multi-attribute outputs are essentially the model's solutions to the corresponding *multioutput classification* problem instances.

The model's Classification stage includes three branches, each containing a classification head (CH) responsible for one of the three classification outputs. During model training (optimized using backpropagation), the loss function F was the weighted sum of the three branches' cross-entropy losses given by Equations (4.1) and (4.2) below.

$$\begin{aligned} F_i &= \alpha_{loc} \cdot F_{loc} + \alpha_{size} \cdot F_{size} + \alpha_{tilt} \cdot F_{tilt} \\ &= \alpha_{loc} \cdot \sum_{j=1}^{C_{loc}} y_{loc}^{(j)} \cdot \log(\hat{y}_{loc}^{(j)}) + \alpha_{size} \cdot \sum_{j=1}^{C_{size}} y_{size}^{(j)} \cdot \log(\hat{y}_{size}^{(j)}) + \alpha_{tilt} \cdot \sum_{j=1}^{C_{tilt}} y_{tilt}^{(j)} \cdot \log(\hat{y}_{tilt}^{(j)}) \end{aligned} \tag{4.1}$$

$$F = \sum_{i=1}^{N} F_i \tag{4.2}$$

where

- F_i is the loss for the i-th sample.
- F_{loc}, F_{size}, and F_{tilt} are the cross-entropy losses for the location, size, and tilt branches, respectively.
- $y_{loc}^{(j)}$, $y_{size}^{(j)}$, and $y_{tilt}^{(j)}$ are the ground truth labels for the j-th class of the location, size, and tilt branches, respectively.
- $\hat{y}_{loc}^{(j)}$, $\hat{y}_{size}^{(j)}$, and $\hat{y}_{tilt}^{(j)}$ are the probability-like softmax scores (ranging from 0 to 1) for the j-th class of the location, size, and tilt branches, respectively.
- C_{loc}, C_{size}, and C_{tilt} are the number of classes for the location, size, and tilt attributes, respectively.
- α_{loc}, α_{size}, and α_{tilt} are the weights for the location, size, and tilt attributes, respectively.
- N is the total number of samples.

As a special case, our model can also be trained to predict fewer than three attributes. For example, when training solely for the location classification task, the loss function weights for the outputs of the size and tilt classification heads can simply be set to 0. This allows for evaluating the model performance limits, which we explore in Section 4.2.

4.1.3 Training and Evaluation

Given a relatively small amount of data, we used the 75:25 split ratio to randomly partition our input dataset into the training and test sets. We repeated our randomized computational experiments ten times for each evaluated model, so that we could report the model's average observed performance and related standard deviation figures.

To ensure fair comparisons among the evaluated models, a unified training setup was adopted. The Adam optimizer was used [25], with a learning rate of 0.01, the loss function was given by Equations (4.1)-(4.2), the training batch size was 256, and the number of training epochs was 300.

4.1.4 Evaluation Criteria

Evaluating the performance of deep learning models involves several key criteria, including accuracy, network size, and computational complexity. For lightweight network models, which are often deployed on embedded devices in real-time environments, it is essential to consider not only general metrics, such as the number of parameters and FLOPS, but also the actual inference speed values. For instance, the MobileNetv4[44] employs the Roofline Model to analyze inference speed from two perspectives: MACtime and Memtime. In this thesis, our model assessment is simplified to include the following four principal evaluation criteria: classification accuracy, the number of parameters, the number of FLOPs, and the inference latency.

Since our target task is to produce multi-attribute classification results, we consider the following accuracy measurements.

- Single-attribute accuracy: The models are trained to predict only one attribute of the crack defect, i.e., either its location, or its size, or its tilt.

- Multi-attribute accuracy: The models are trained to predict more than one attribute of the crack defect, i.e., either its size and tilt, or all three attributes.

The multi-attribute accuracy is calculated as follows: if all the target attributes are predicted correctly, the overall outcome is treated as a success; otherwise, it is considered to be a failure.

We have used the PyTorch [2] and Thop [78] library to determine the number of parameters and FLOPs for each evaluated model. The inference latency (average execution time per batch of 256 test samples) has been measured on a Windows 10 workstation with an Intel Core i5 CPU (2.5 GHz clock, 16 GB RAM) and NVIDIA GeForce RTX3060Ti GPU (16 GB VRAM).

4.2 Evaluation Results and Analysis

In this section, we present the experimental results and comparative analysis of various neural network models, highlighting the competitive performance of EMACnet and EMACnet++. Table 4.1 summarizes the prediction accuracy of our proposed models in comparison to the baseline ResNet18 [18] and three state-of-the-art networks: MobileNetV3-L [19], FastViT [63], and RepViT [69]. We evaluate the following three prediction scenarios.

- Each model is trained to predict only one attribute of the crack defect, i.e., either its location, or its size, or its tilt.
- Each model is trained to predict two attributes simultaneously, namely, the crack size and tilt.
- Each model is trained to predict all three attributes simultaneously.

Table 4.1: Model Accuracy Comparison

Model Name	**Loc. Acc**	**Size Acc**	**Tilt Acc**	**Size/Tilt Acc**	**Loc./Size/Tilt Acc**
ResNet18	0.82 ± 0.03	1.00 ± 0.00	0.99 ± 0.01	0.98 ± 0.01	0.65 ± 0.02
MobileNetV3-L	0.57 ± 0.04	1.00 ± 0.00	0.91 ± 0.02	0.90 ± 0.03	0.23 ± 0.02
FastViT	0.41 ± 0.02	1.00 ± 0.00	0.92 ± 0.01	0.91 ± 0.01	0.20 ± 0.02
RepViT	0.86 ± 0.02	1.00 ± 0.00	0.95 ± 0.01	0.97 ± 0.01	0.59 ± 0.03
EMACnet	0.66 ± 0.04	0.96 ± 0.02	0.94 ± 0.03	0.92 ± 0.03	0.47 ± 0.04
EMACnet++	0.79 ± 0.07	1.00 ± 0.00	0.99 ± 0.01	0.96 ± 0.01	0.57 ± 0.03

Table 4.2: Cost Comparison for Evaluated Models

Model Name	Latency(s)	Params(M)	FLOPs(M)
ResNet18	0.003	11.2	699.2
MobileNetV3-L	0.004	4.23	87.5
FastViT	0.009	3.27	207.1
RepViT	0.110	4.73	329.8
EMACnet	0.002	1.65	68.0
EMACnet++	0.003	1.68	97.0

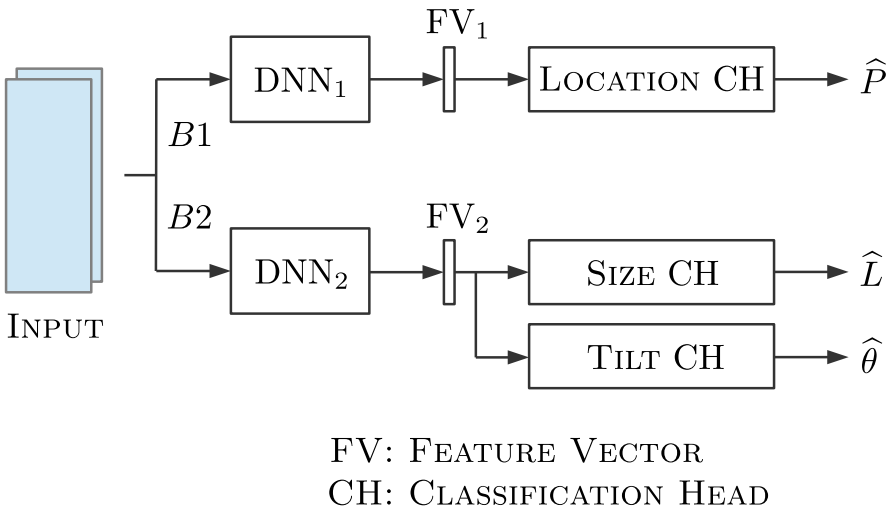


Figure 4.1: Organization of Modified Models. Note: DNN_1 and DNN_2 Are Identical Networks Trained for Two Different Classification Tasks.

It is evident that ResNet18, RepViT, and EMACnet++ are the strongest performers in terms of location-only classification, achieving the prediction accuracy of 0.82, 0.86, and 0.79, respectively. When classifying the tilt-only or combined size/tilt attributes, the same three models yield the prediction accuracy of at least 0.95. In the case of three-attribute predictions, the combined accuracy drops dramatically: it is equal to 0.65 for ResNet18, 0.59 for RepViT, and 0.57 for EMACnet++.

Table 4.2 compares the evaluated models in terms of their inference latency, number of parameters, and number of FLOPs. Among the three best performing models, namely ResNet18, RepViT, and EMACnet++, ours is the least expensive. It has the same latency as ResNet18 (3 ms per batch), but 6.7x fewer parameters (1.68M versus 11.2M) and 7.2x fewer FLOPs (97.0M versus 699.2M). It is also 36.7 times

faster than RepViT, with 2.8x fewer parameters and 3.4x fewer FLOPs. Compared to EMACnet++, the original EMACnet model has approximately the same number of parameters, but the latter is 33.3% faster and has 29.9% fewer FLOPs. On the other hand, EMACnet++ can predict crack locations with substantially higher accuracy (0.79 versus 0.66 in Table 4.1).

The last two columns of Table 4.1 suggest that adding the location attribute to the combined size/tilt classification compromises the network's ability to learn all three types of attribute features effectively using the same pre-classification layers before branching to three different classification heads. Since EMACnet and EMACnet++ have the smallest number of parameters (see Table 4.2), we can afford a modified network organization shown in Figure 4.1. First, the pre-classification layers are simply replicated to form two processing branches $B1$ and $B2$ for the same input sample. Then, the $B1$ output is supplied to a single CH for the location-only classification, while the $B2$ output is supplied to a pair of CH branches for the combined size/tilt classification.

We have observed that the resulting accuracy for the combined location/size/tilt predictions increased to 0.62 $\pm$ 0.04 for modified EMACnet and 0.78 $\pm$ 0.07 for modified EMACnet++. Although the number of parameters and FLOPs effectively doubled in our modified networks, they are still cheaper than ResNet18 and RepViT, while offering a comparable (EMACnet) or better (EMACnet++) three-attribute prediction accuracy than 0.65 of ResNet18 and 0.59 of RepViT.

In conclusion, while other networks, such as ResNet18 and RepViT exhibit strong performance in certain metrics, EMACnet and EMACnet++ stand out due to a balanced combination of their competitively high classification performance (see Table 4.1) and their low operating cost (see Table 4.2). These characteristics make EMACnet and EMACnet++ desirable choices for applications requiring high performance in real-time and resource-constrained environments. When comparing EMACnet to EMACnet++, the latter shows improvements in location-limited multi-attribute classification accuracy (0.78 versus 0.62 after modification), which is due to the architectural enhancements incorporated into EMACnet++.

4.2.1 Further Discussion

Table 4.2 shows that the inference latency and the number of FLOPs do not necessarily follow a linear relationship. Models with fewer FLOPs may have longer latencies

compared to those with more FLOPs. This is because the compatibility of the model design with the underlying hardware can also affect the inference speed. For example, the number of FLOPs in ResNet18 is much larger compared to MobileNetV3-L, primarily because the former utilizes standard convolutions, while the latter relies on separable ones. However, having fewer FLOPs did not translate into a smaller inference latency (due to mismatched hardware support), while it still paid a price in terms of diminished location-only classification accuracy (MobileNet's 0.57 versus ResNet's 0.82). For our EMACnet++, its selective use of both standard and depth-wise convolutions has resulted in a balanced tradeoff between the accuracy and speed.

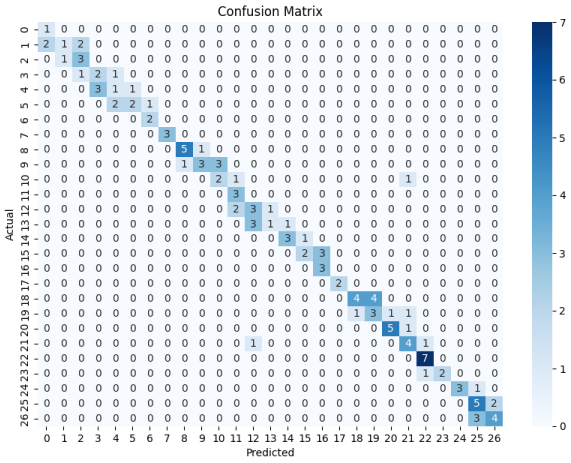
Meanwhile, Table 4.1 shows that all models performed unsatisfactorily in terms of the location predictions. This could potentially be due to the fact that the location class size of 27 is larger than those of the crack size and tilt classes (5 and 11, respectively), i.e., the location classification is expected to have higher sensitivity to the input data. Another possible reason could be that the spatial information included in location-related features differs from that of the size- and tilt-related features. The former is likely to be affected by the global spatial information, while the latter is likely to be affected by localized data properties. This issue is discussed further in the Appendix.

4.2.2 Soft Location Prediction

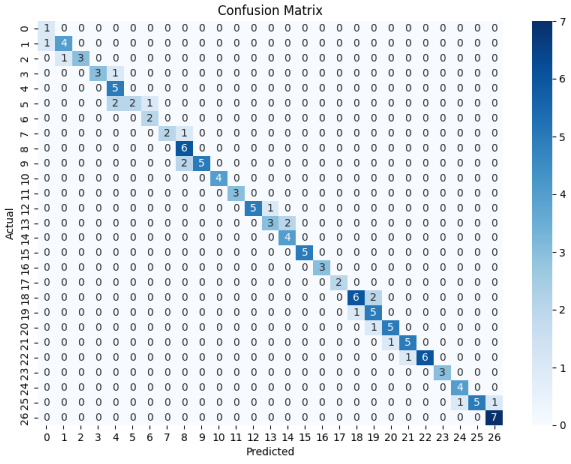
As mentioned earlier, the location prediction accuracy is not satisfactory. However, by examining the confusion matrices of the best-case location-only results produced by EMACnet and EMACnet++, we can see that the errors are mainly concentrated near the matrix diagonal, as shown in Figure 4.2. This indicates that most mispredictions are only one or two class bins away from the ground truth. Recall that any pair of adjacent location classes are separated from each other by the physical distance of 0.3 mm. If the user can tolerate the crack location error of ± 0.3 mm, a mispredicted class that is immediately next to the ground truth can be treated as an acceptable classification outcome. In other words, we can enable soft predictions by relaxing adjacent class boundaries.

Using this relaxation method, we can significantly improve the location prediction accuracy of EMACnet and EMACnet++ without compromising its physical significance. The results are shown in Table 4.3, where the term "Soft" designates prediction scenarios with relaxed location classification: if the predicted location is within ± 0.3

mm (i.e., within a single-class distance) from the ground truth, it is considered to be successful. The location-only prediction accuracy of EMACnet and EMACnet++ increases to 0.93 and 0.97, respectively. More importantly, their accuracy of the combined location/size/tilt predictions become equal to 0.83 and 0.87, respectively. As for the modified versions of EMACnet and EMACnet++ (see Figure 4.1), their accuracy of the corresponding combined location/size/tilt predictions increases to 0.89 $\pm$ 0.02 and 0.95 $\pm$ 0.03. These results provide another confirmation that the EMACnet++ performance is superior to that of EMACnet.



(a) Confusion Matrix of EMACnet



(b) Confusion Matrix of EMACnet++

Figure 4.2: Confusion Matrices for Location-Only Classification Results

Table 4.3: Soft Accuracy Comparison

Model Name	Loc. Acc	Loc./Size/Tilt Acc	Soft Loc. Acc	Soft Loc./Size/Tilt Acc
EMACnet	0.66 ± 0.04	0.47 ± 0.04	0.93 ± 0.02	0.83 ± 0.03
EMACnet++	0.79 ± 0.07	0.57 ± 0.03	0.97 ± 0.02	0.87 ± 0.03

4.2.3 Heatmap Visualization

For CNNs, heatmap analysis can be valuable for understanding the inner behavior of the network. In this section, we examine the functionality of EMACnet and EMACnet++ through two visualization techniques: gradient heatmaps and activation heatmaps. The gradient heatmap reveals the network's sensitivity to the input by computing the gradient information, while the activation heatmap reveals the network's activation patterns by computing the activation information. Through these methods, we aim to provide insights into the network's inner workings.

Gradient Heatmap

In Section 4.2.2, we demonstrated that EMACnet and EMACnet++ mispredicted crack locations primarily due to confusion between adjacent ground truth (GT) classes. We compare several outliers picked from the confusion matrices (see Figure 4.2) with correctly classified samples using gradient heatmaps to analyze the network's sensitivity to these outliers.

The examined two-channel input samples (marked with letters A to H) are shown in Figure 4.3, where each sample consists of two raw channel data frames (for positive- and negative-angle PW emissions), with Location GT Class indicating the true label for the location of an imaged crack defect. The corresponding heatmap images are shown in Figure 4.4. They are displayed in pairs, with the left and right images depicting the gradient heatmap for EMACnet and EMACnet++, respectively. Figure 4.4 also includes the class predictions by EMACnet and EMACnet++, as well as the confidence scores of these predictions. As one can see, EMACnet misclassified input samples C and F, while EMACnet++ correctly classified all samples under consideration.

It is important to note that even within the same location GT class, the input samples can be very different depending on the size and/or tilt of an imaged crack defect at a fixed lateral position. For instance, clearly distinct samples C, D, and E (see Figure 4.3) belong to the same class 21. EMACnet was successful when classifying D

and E, but incorrectly put C into class 12 (with a fairly low confidence score). For these three cases, their gradient heatmaps in Figure 4.4 indicate that both EMACnet and EMACnet++ rely on similar input data patterns, but the former utilizes additional "background" data points. Undue influence of this "background" noise could be one possible explanation for EMACnet's incorrect decision to put sample C into the same class as samples A and B.

On the other hand, when misclassifying sample F that belongs to the same class 10 as samples G and H, EMACnet utilized fewer data points than EMACnet++ (see Figure 4.4), but more importantly, the two models relied on quite different patterns in the upper third section of the input data. This example suggests that the difference between feature extraction capabilities of EMACnet and EMACnet++ goes beyond "background" noise filtering.

Activation Heatmap

Figures 4.5 and 4.6 show the activation heatmaps for the same input sample supplied to both EMACnet and EMACnet++ during location-only inference. From these heatmaps, it can be observed that both EMACnet and EMACnet++ extract locally concentrated (sparse) and visually distinguishable features in Conv0, Layer1, and Layer2. However, in Layer3 and Layer4 of EMACnet, the features become more spread out and less distinguishable. We hypothesize that in this case, visually less structured features are indicative of potential overfitting that might explain poor EMACnet performance in the subsequent Classification stage after GAP. In contrast, EMACnet++ continues to extract sparse and distinguishable features in Layer3 and Layer4, which is correlated with its superior classification performance in comparison to EMACnet.

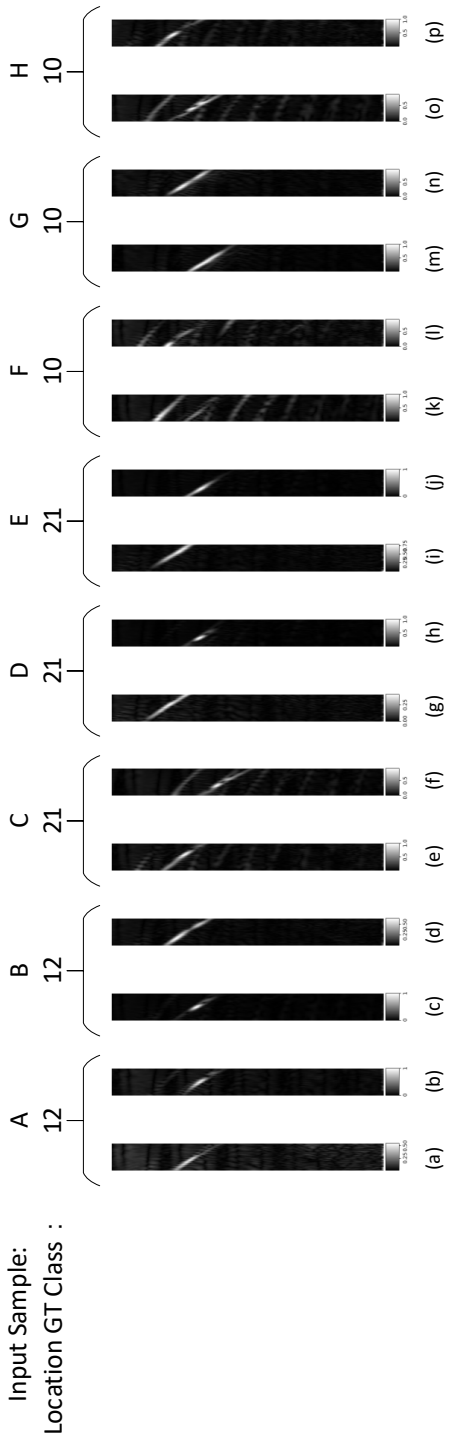


Figure 4.3: Two-Channel Input Samples from Location GT Classes 12, 21, and 10.

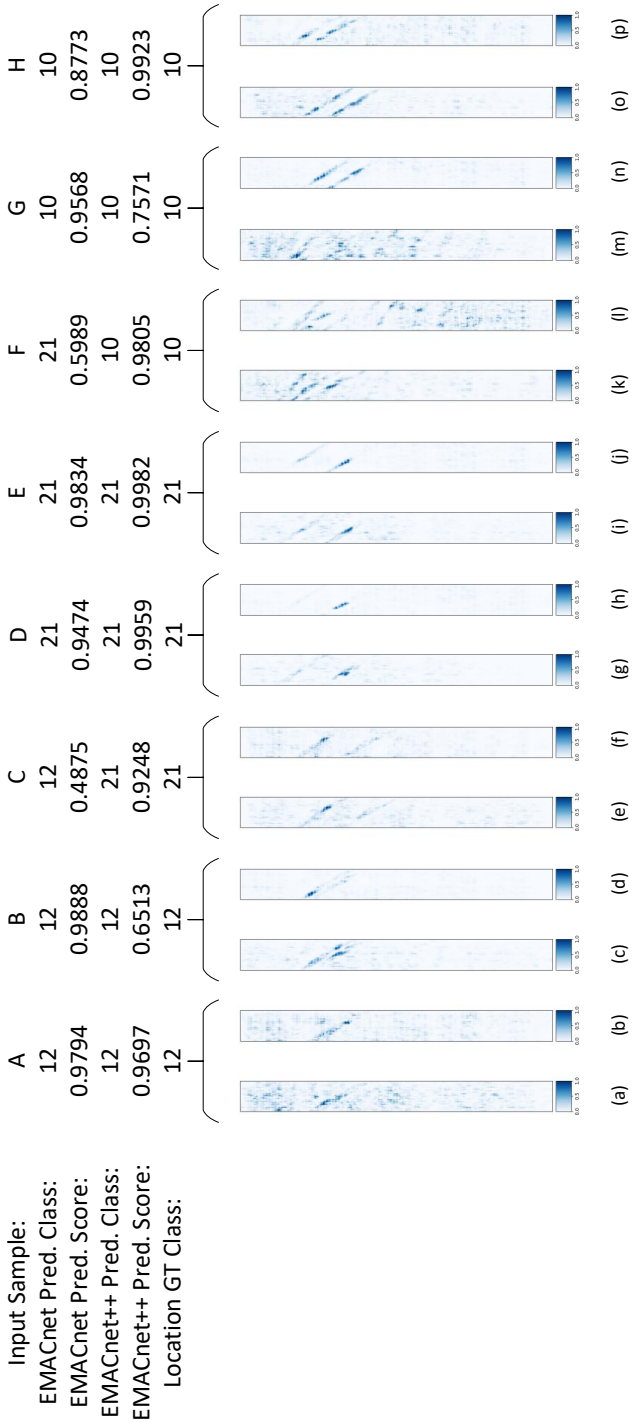


Figure 4.4: Gradient Heatmaps During Location-Only Inference Given Inputs from Figure 4.3.

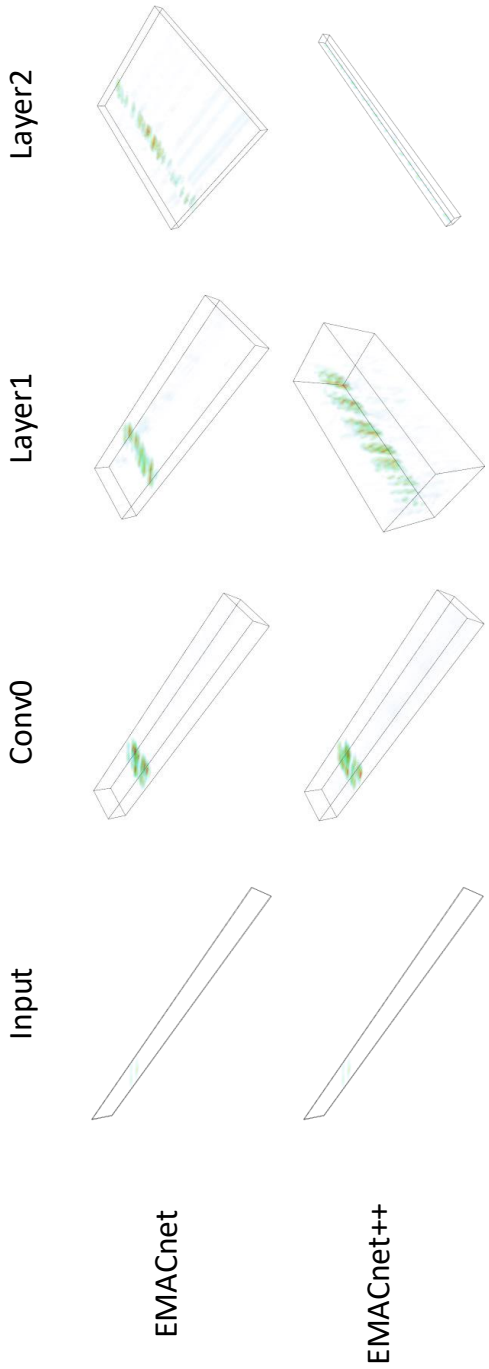


Figure 4.5: Activation Heatmaps for Conv0, Layer1, and Layer2 During Location-Only Inference.

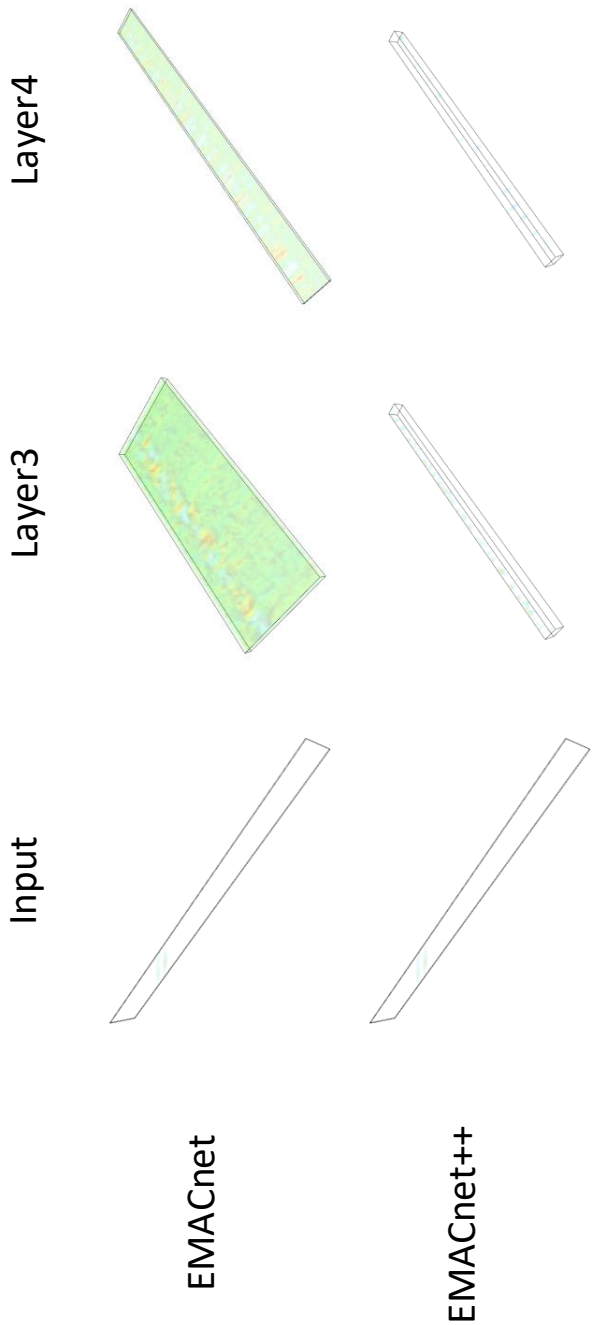


Figure 4.6: Activation Heatmaps for Layer3 and Layer4 During Location-Only Inference.

Chapter 5

Conclusion and Future Work

5.1 Concluding Remarks

In this thesis, we have explored ultrasonic nondestructive evaluation (NDE) data analysis by means of advanced machine learning (ML) techniques. Our primary objective was to leverage ML algorithms to improve the efficiency of inline pipeline inspection.

We introduced two lightweight network structures, EMACnet and EMACnet++, which are tailored for handling raw ultrasound imaging data. These models incorporate modern structures, such as the Universal Inverted Bottleneck (UIB) and the newly proposed EMACblock++, to enhance the models' ability to extract and process spatio-temporal features. The experimental results demonstrated that both EMACnet and EMACnet++ outperform several competitors either in terms of classification accuracy associated with predicting the location, size, and tilt of surface-breaking crack defects (e.g., MobileNetV3-L and FastViT), or in terms of the computational cost measured by the inference latency, the number of parameters, and FLOPs (e.g., ResNet18 and RepViT).

5.2 Future Work

Our findings highlight the potential of lightweight deep learning models in enhancing ultrasonic NDE data analysis, indicating that EMACnet and EMACnet++ are viable solutions for real-time, resource-constrained applications. These models can contribute to more intelligent automated inspection systems, enhancing the safety

and efficiency of critical infrastructure maintenance.

Future work will focus on further refining these models to improve their interpretability and adaptability to various NDE scenarios. Additionally, following the discussion on the loss of spatial information presented in the appendix, further research into network designs that better extract both local and global spatial information is a promising direction. We will also explore appropriate data augmentation techniques to enhance the generalization capability of the models. Moreover, we plan to apply EMACnet and EMACnet++ to other fields, such as medical imaging and computer vision, to validate their applicability across different domains.

Appendix

Spatial Information in CNN

In deep learning, particularly with convolutional neural networks (CNNs), pooling operations are commonly used to reduce the size of feature maps, thereby decreasing computational load and the number of parameters. However, pooling may cause a significant loss of spatial information at the feature level. For example, the sky is usually at the top of an image, the sun is in the sky, roads are at the bottom, and cars run on these roads. These features have a statistical distribution in images. This distribution allows models, even after pooling, to infer the position of features through multiple layers by simply recognizing their presence.

The spatial information post-pooling depends entirely on the probability distribution of specific features' locations in the dataset. CNNs, being numerical methods, essentially fit to the dataset and its underlying distribution. For instance, while the sun can theoretically appear anywhere in an image, it is typically in the sky in the dataset, enabling the model to infer its approximate location even after pooling. However, this reliance on dataset distribution can lead to poor performance when encountering differently distributed data.

The size of the receptive field significantly impacts the pooling operation. If the convolution's receptive field covers the entire image, the model can infer the global spatial distribution of features. However, if the receptive field is small, such as a 7x7 feature map before the final pooling, the original image is effectively divided into 7x7 patches. After pooling, the model struggles to infer global spatial information spanning these patches. Within each patch, spatial distribution of features can be partially retained, but the spatial relationships between different patches are lost.

The discussion further differentiates between two types of spatial information: one is the positional distribution of features within an image, such as the sun being usually at the top; the other is the spatial relationship between features, like the sun being in the sky. For image classification tasks, only the latter is needed, and this information can be effectively captured through convolution and non-linear layers, making pooling perform well. However, for tasks requiring precise location detection, such as crack localization, pooling can lead to severe information loss.

In our experiments, the crack size and tilt classification performed well, but the location classification was poor. This is likely due to the fact that location information

heavily relies on the first type of spatial information—specific positions within the image—which pooling diminishes. In contrast, size and tilt classification depends more on the spatial relationship between features, which convolution networks can capture well.

To mitigate the limitations of pooling, alternative methods can be used. Techniques like skip connections and pyramid features (e.g., in Unet [48]) can expand the receptive field. Models like YOLO [47] address absolute spatial information by dividing images into grids. These methods have shown better performance in various tasks. However, the spatial information loss caused by pooling remains a challenge in certain tasks, requiring more complex model structures and larger training datasets to compensate.

Overall, the limitations of pooling in specific tasks remind us to carefully design and adjust model structures according to task requirements, balancing computational efficiency and information retention.

Bibliography

[1] N. Amiri, G. H. Farrahi, K. R. Kashyzadeh, and M. Chizari. Applications of ultrasonic testing and machine learning methods to predict the static & fatigue behavior of spot-welded joints. *J Manuf Process*, 52:26–34, Apr. 2020. doi: 10.1016/j.jmapro.2020.01.047.

[2] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024. doi: 10.1145/3620665.3640366. URL `https://pytorch.org/assets/pytorch2-2.pdf`.

[3] A. Atto, M. Grigat, and J. Voss. Continuous depth sizing of ili ultrasonic crack detection. In *Proceedings of the International Pipeline Conference*, volume 50251, page V001T03A064, 2016.

[4] A. Bernieri, L. Ferrigno, M. Laracca, and M. Molinara. Crack shape reconstruction in eddy current testing using machine learning systems for regression. *IEEE Trans Instrum Meas*, 57(9):1958–1968, 2008. doi: 10.1109/TIM.2008.919011.

[5] M. Bertovic and I. Virkkunen. NDE 4.0: new paradigm for the NDE inspection personnel. *Handbook of Nondestructive Evaluation 4.0*, pages 1–31, 2021.

[6] N. Brierley et al. Advances in the UK toward NDE 4.0. *Research in Nondestructive Evaluation*, 31(5-6):306–324, 2020.

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–229. Springer, 2020.

[8] Y.-J. Cha, W. Choi, and O. Büyüköztürk. Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, 32(5):361–378, May 2017. doi: 10.1111/mice.12263.

[9] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5270–5279, 2022.

[10] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1251–1258, 2017.

[11] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13733–13742, 2021.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[13] J. Feng, F. Li, S. Lu, J. Liu, and D. Ma. Injurious or noninjurious defect identification from mfl images in pipeline inspection using convolutional neural network. *IEEE Trans Instrum Meas*, 66(7):1883–1892, 2017.

[14] GlobalData Energy. North America has the highest oil and gas pipeline length globally, 2019. URL `https://www.offshore-technology.com/comment/north-america-has-the-highest-oil-and-gas-pipeline-length-globally/`. Accessed: Aug. 19, 2022.

[15] Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu, Zheng-Ning Liu, Peng-Tao Jiang, Tai-Jiang Mu, Song-Hai Zhang, Ralph R Martin, Ming-Ming Cheng, and Shi-Min Hu. Attention mechanisms in computer vision: A survey. *Computational Visual Media*, 8(3):331–368, 2022.

[16] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1580–1589, 2020.

[17] J. B. Harley and D. Sparkman. Machine learning and NDE: Past, present, and future. In *AIP Conference Proceedings*, volume 2102, page 090001, May 2019. doi: 10.1063/1.5099819.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019.

[20] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[21] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018.

[22] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017.

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[24] R. C. Ireland and C. R. Torres. Finite element modelling of a circumferential magnetiser. *Sens Actuators A Phys*, 129(1-2):197–202, 2006.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] K. Korol, Y. Hubert, G. Fredine, P. Senf, and S.-A. Koon Koon. A study of crack detection ultrasonic attributes to manage leak threats associated with short crack-like flaws in erw pipelines. In *Proceedings of the International Pipeline Conference*, volume 50251, page V001T03A073, 2016.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6): 84–90, 2017.

[28] L. le Jeune, S. Robert, E. L. Villaverde, and C. Prada. Plane wave imaging for ultrasonic nondestructive testing: Generalization to multimodal imaging. *Ultrasonics*, 64:128–138, 2016.

[29] Yanyu Li, Ju Hu, Yang Wen, Georgios Evangelidis, Kamyar Salahi, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Rethinking vision transformers for mobilenet size and speed. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 16889–16900, 2023.

[30] Z. Lin, H. Pan, G. Gui, and C. Yan. Data-driven structural diagnosis and conditional assessment: from shallow to deep learning. In *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2018*, volume 10598, page 38, Mar. 2018. doi: 10.1117/12.2296964.

[31] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE*

conference on Computer Vision and Pattern Recognition (CVPR), pages 11976–11986, 2022.

[32] MG Lozev, RL Spencer, and D Hodgkinson. Optimized inspection of thin-walled pipe welds using advanced ultrasonic techniques. *Journal of Pressure Vessel Technology*, 127(3):237–243, 04 2005.

[33] A. S. Lundervold and A. Lundervold. An overview of deep learning in medical imaging focusing on MRI. *Z Med Phys*, 29(2):102–127, 2019.

[34] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[35] Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021.

[36] M. Meng, Y. J. Chua, E. Wouterson, and C. P. K. Ong. Ultrasonic signal classification and imaging system for composite materials via deep convolutional neural networks. *Neurocomputing*, 257:128–135, 2017.

[37] N. G. Meyendorf et al. NDE 4.0–NDE for the 21st century–the internet of things and cyber physical systems will revolutionize NDE. In *Proceedings of the Asia Pacific Conference for Nondestructive Testing (APCNDT)*, pages 1–8, Singapore, 2017.

[38] M. Mishra, A. S. Bhatia, and D. Maity. Predicting the compressive strength of unreinforced brick masonry using machine learning techniques validated on a case study of a museum through nondestructive testing. *J Civ Struct Health Monit*, pages 1–15, Mar. 2020. doi: 10.1007/s13349-020-00391-7.

[39] North Carolina Dept. of Environmental Quality. Colonial pipeline spill information - huntersville, n.c, Jul. 2022. URL `https://deq.nc.gov/about/divisions/waste-management/underground-storage-tanks-section/colonial-pipeline-spill-information-huntersville-nc`. Accessed: Aug. 23, 2022.

[40] P. O'Rorke, S. Morris, M. Amirfathi, W. Bond, and D. St. Clair. Machine learning for nondestructive evaluation. In *Machine Learning Proceedings 1991*, pages 620–624. Elsevier, 1991. doi: 10.1016/b978-1-55860-200-7.50126-4.

[41] Richard J. Pyle. *Application of machine learning to ultrasonic nondestructive evaluation*. PhD thesis, University of Bristol, 2023.

[42] Richard J. Pyle, Rhodri L. T. Bevan, Robert R. Hughes, Rosen K. Rachev, Amine Ait Si Ali, and Paul D. Wilcox. Deep learning for ultrasonic crack characterization in NDE. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 68(5):1854–1865, 2021. doi: 10.1109/TUFFC.2020.3045847.

[43] Richard J. Pyle, Robert R. Hughes, and Paul D. Wilcox. Interpretable and explainable machine learning for ultrasonic defect sizing. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 70(4):277–290, 2023. doi: 10.1109/TUFFC.2023.3248968.

[44] Danfeng Qin, Chas Leichner, Manolis Delakis, Marco Fornoni, Shixin Luo, Fan Yang, Weijun Wang, Colby Banbury, Chengxi Ye, Berkin Akin, et al. Mobilenetv4-universal models for the mobile ecosystem. *arXiv preprint arXiv:2404.10518*, 2024.

[45] R. Rachev. *Advanced ultrasonic array processing for pipeline inline inspection*. PhD thesis, University of Bristol, 2021.

[46] K. Reber, M. Beller, H. Willems, and O. A. Barbian. A new generation of ultrasonic in-line inspection tools for detecting, sizing and locating metal loss and cracks in transmission pipelines. In *Proceedings of the IEEE Ultrasonics Symposium*, volume 1, pages 665–671, 2002.

[47] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[48] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 234–241. Springer, 2015.

[49] S. Sambath, P. Nagaraj, and N. Selvakumar. Automatic defect classification in ultrasonic ndt using artificial intelligence. *J Nondestr Eval*, 30(1):20–28, Mar. 2011. doi: 10.1007/s10921-010-0086-0.

[50] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018.

[51] N. J. Shipway, T. J. Barden, P. Huthwaite, and M. J. S. Lowe. Automated defect detection for fluorescent penetrant inspection using random forest. *NDT & E International*, 101:113–123, 2019.

[52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[53] K. Suzuki. Overview of deep learning in medical imaging. *Radiological Physics and Technology*, 10(3):257–273, Sep. 2017. doi: 10.1007/s12194-017-0406-5.

[54] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[55] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[56] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[57] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 6105–6114. PMLR, 2019.

[58] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2820–2828, 2019.

[59] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, and Herv'e J'egou. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021.

[60] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 139, pages 10347–10357, July 2021.

[61] X. L. Travassos, S. L. Avila, and N. Ida. Artificial neural networks and machine learning techniques applied to ground penetrating radar: A review. *Applied Computing and Informatics*, 2020.

[62] L. Udpa and S. S. Udpa. Neural networks for the classification of nondestructive evaluation signals. *IEE Proceedings, Part F: Radar and Signal Processing*, 138 (1):41–45, 1991. doi: 10.1049/ip-f-2.1991.0007.

[63] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Fastvit: A fast hybrid vision transformer using structural reparameterization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5785–5795, 2023.

[64] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Mobileone: An improved one millisecond mobile backbone. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7907–7917, 2023.

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[66] I. Virkkunen, T. Koskinen, O. Jessen-Juhler, and J. Rinta-Aho. Augmented ultrasonic data for machine learning. *J Nondestr Eval*, 40(1):1–11, 2021.

[67] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis. Deep learning for computer vision: A brief review. *Comput Intell Neurosci*, 2018, 2018.

[68] J. Vrana and R. Singh. NDE 4.0—a design thinking perspective. *J Nondestr Eval*, 40(1):1–24, 2021.

[69] Ao Wang, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Repvit: Revisiting mobile cnn from vit perspective. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15909–15920, 2024.

[70] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 390–391, 2020.

[71] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[72] Yunyang Xiong, Hanxiao Liu, Suyog Gupta, Berkin Akin, Gabriel Bender, Yongzhe Wang, Pieter-Jan Kindermans, Mingxing Tan, Vikas Singh, and Bo Chen. Mobiledets: Searching for object detection architectures for mobile accelerators. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3825–3834, 2021.

[73] J. Ye, S. Ito, and N. Toyama. Computerized ultrasonic imaging inspection: From shallow to deep learning. *Sensors (Switzerland)*, 18(11), Nov. 2018. doi: 10.3390/s18113820.

[74] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10819–10829, June 2022.

[75] J. Zhang, B. W. Drinkwater, and P. D. Wilcox. The use of ultrasonic arrays to characterize crack-like defects. *J Nondestr Eval*, 29(4):222–232, 2010.

[76] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings*

of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), pages 6848–6856, 2018.

[77] Xiaoliang Zhao, Venugopal K. Varma, Gang Mei, Bulent Ayhan, and Chiman Kwan. In-Line Nondestructive Inspection of Mechanical Dents on Pipelines With Guided Shear Horizontal Wave Electromagnetic Acoustic Transducers, 04 2005.

[78] Ligeng Zhu. pytorch-opcounter, 2022. URL `https://github.com/Lyken17/pytorch-OpCounter`. Accessed: 2024-07-09.

[79] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.