Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Analysis of Defect Localization in Images Employing Explainability Methods within Deep Learning Models

Master's Final Degree Project

Lukas Zabulis

Project author

assoc. prof. dr. Arūnas Lipnickas

Supervisor

Kaunas, 2024

Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Analysis of Defect Localization in Images Employing Explainability Methods within Deep Learning Models

Master's Final Degree Project

Control Technologies (6211EX014)

Lukas Zabulis

Project author

assoc. prof. dr. Arūnas Lipnickas

Supervisor

prof. dr. Vidas Raudonis

Reviewer

Kaunas, 2024

Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Lukas Zabulis

Analysis of Defect Localization in Images Employing Explainability Methods within Deep Learning Models

Declaration of Academic Integrity

I confirm that the final project of mine, Lukas Zabulis, on the topic „Analysis of defect localization in images employing explainability methods within deep learning models“ was written completely by myself; all the provided data and research results are correct and have been obtained honestly. None of the parts of this thesis have been plagiarized from any printed, Internet-based, or otherwise recorded sources. All direct and indirect quotations from external resources are indicated in the list of references. No monetary funds (unless required by Law) have been paid to anyone for any contribution to this project.

I fully and completely understand that any discovery of any manifestations/case/facts of dishonesty inevitably results in me incurring a penalty according to the procedure(s) effective at Kaunas University of Technology.

Lukas Zabulis

Confirmed electronically

Zabulis Lukas. Analysis of defect localization in images employing explainability methods within deep learning models. Master's Final Degree Project/supervisor assoc. prof. Arūnas Lipnickas; Faculty of Electrical and Electronics Engineering, Kaunas University of Technology.

Study field and area (study field group): Electronics Engineering (engineering sciences).

Keywords: deep learning, convolutional neural network, vision transformer, explainability, defect detection.

Kaunas, 2024. 54 p.

Summary

Integrating computer vision systems into modern manufacturing for quality control offers substantial benefits. These systems, particularly those powered by deep learning, surpass human capabilities in both speed and accuracy for visual inspections. Nevertheless, significant challenges remain, particularly the need for task-specific data and the labor-intensive process of precisely annotating datasets for methods like segmentation or object detection. Explainability maps of neural networks can help address this issue by visually highlighting the regions that the network deems essential for predictions. This approach allows classifier networks to output class labels and an associated explainability map, functioning as a preliminary form of object detection. This technique only requires class labels, which are easier to obtain. This study conducts a comparative analysis of explainability methods across various neural network architectures, focusing on Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs). For CNNs, a Class Activation Mapping (CAM)-based technique generates an attention map for defect localization, while ViTs employ an Attention Rollout calculation for the same purpose. Both architectures are adapted to efficiently identify defects in images during a single iteration, eliminating the need for detailed pixel-wise annotations and substantial model complexity increase. This research evaluates and compares these methods using segmentation metrics of generated explainability maps with two datasets: Printed Circuit Boards (PCB) and Gear Defect Inspection (GID). When considering different architectures, CNNs explainability output is more precise, enhancing the F1 score by up to 19%, the Jaccard index by up to 22%, recall by up to 48%, and the pointing game metric by up to 13%.

Zabulis Lukas. Defektų lokalizavimo nuotraukose tyrimas taikant giliojo mokymo modelių paaiškinamumo metodus. Magistro baigiamasis projektas / vadovas / doc. dr. Arūnas Lipnickas; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): elektronikos inžinerija (inžinerijos mokslai).

Reikšminiai žodžiai: gilusis mokymas, konvoliucinis neuroninis tinkas, vaizdų transformeris, paaiškinamumo metodai, defektų aptikimas.

Kaunas, 2024. 54 p.

Santrauka

Kompiuterinės regos sistemų integravimas į modernią gamybą turi daug privalumų. Sistemos paremtos giliuoju mokymu pranoksta žmogaus galimybes tiek greičiu, tiek tikslumu atliekant vizualinius patikrinimus. Nepaisant to, susiduriama su iššūkiais, kurie apima dideles specifinių duomenų apimtis ir daug darbo reikalaujantį procesą, skirtą tiksliai anotuoti duomenų rinkinius tokiems uždaviniams kaip segmentavimas ar objektų aptikimas. Neuroninių tinklų paaiškinamumo metodai gali padėti išspręsti šią problemą, vizualiai išskiriant regionus, kuriais remiantis atliekamas spėjimas. Šis metodas leidžia klasifikatorių tinklams išvesti klasę bei susijusį paaiškinamumo žemėlapį, naudojamą objekto lokalizavimui. Šis metodas reikalauja tik klasės anotacijų, kurias lengviau sudaryti, lyginant su segmentavimo ar objektų aptikimo anotacijomis. Šiame tyrime atliekama lyginamoji paaiškinamumo metodų analizė dvejose neuroninių tinklų architektūrose, dėmesį skiriant konvoliuciniams neuroniniams tinklams (CNN) ir vaizdų transformeriams (ViT). CNN atveju klasės aktyvacijos žemėlapio pagrįsta technika sukuria paaiškinamumo žemėlapį defektams nustatyti, o ViT tam pačiam tikslui naudoja dėmesio nukreipimo mechanizmą. Abi architektūros pritaikytos efektyviai identifikuoti defektus paveiksluose vienos iteracijos metu, be detalių anotacijų ir modelio kompleksiškumo padidėjimo. Šiame tyrime paaiškinamumo metodai vertinami lyginant paaiškinamumo išėjimo segmentavimo metrikas su dvejais duomenų rinkiniais: spausdintų montažinių plokščių (PCB) ir krumpliaračių defektų (GID). Atsižvelgiant į skirtingas architektūras, CNN paaiškinamumo išvestis yra tikslesnė, padidinant F1 balą iki 19%, „Jaccard“ indeksą iki 22%, „recall“ iki 48%, o „pointing game“ metriką iki 13%.

Table of contents

List of figures

List of tables

List of abbreviations

Abbreviations:

CNN – Convolutional Neural Network

ViT – Vision Transformer

CAM – Class Activation Mapping

FLOPs – Floating Point Operations

GMM – Gaussian Mixture Model

XAI – Explainable Artificial Intelligence

ML – Machine Learning

PCB – Printed Circuit Board

GID – Gear Defect Inspection

IoU – Intersection over union

Introduction

The advent of computer vision in modern manufacturing has become a pivotal element in ensuring quality control. These systems surpass human capabilities in conducting defect inspections, both in terms of speed and accuracy. The increasing adoption of deep learning solutions for visual inspection tasks marks a significant advancement in this field. Nevertheless, the need for new, specific data for each problem, coupled with the time-intensive nature of precise dataset annotation, especially for accuracy-dependent methods like segmentation or object detection, presents notable challenges.

Explainability maps of neural networks provide a promising solution by visually highlighting regions deemed essential for predictions. This approach enables classifier networks to output both class labels and associated explainability maps, functioning as a preliminary form of object detection that requires only class labels. This makes it easier to obtain relevant data without demanding precise, pixel-wise annotations.

This work delves into the comparative analysis of explainability approaches across different neural network architectures, focusing on Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs). For CNNs, a Class Activation Mapping (CAM)-influenced technique generates an attention map to highlight defective areas in image classification algorithms. ViTs employ an Attention Rollout calculation for the same purpose. Both architectures incorporate lightweight operations, enabling them to localize defects efficiently in a single iteration, eliminating the need for subsequent calculations.

The effectiveness and efficiency of these methods are rigorously tested using explainability output segmentation metrics on two datasets: Printed Circuit Boards (PCB) and Gear Defect Inspection (GID). By comprehensively understanding how these neural network architectures interpret classifications, this research seeks to advance quality control by providing explainable, efficient defect localization without the need for intensive data annotation.

The Aim

Design and implement a comparative study analyzing the explainability of Convolutional Neural Networks (CNN) and Vision Transformers (ViT) in the context of defect localization in images, to quantify the effectiveness of each model's explainability features, facilitating a deeper understanding of how model design influences interpretability.

The Objectives

1. Conduct a comprehensive review of the existing literature focused on Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs), with an emphasis on understanding the methodologies for explainability within these models. Additionally, explore their applications in defect detection scenarios.
2. Develop a versatile experimental framework and train neural network models (CNN and ViT) on PCB and GID datasets with hyperparameter searching optimization strategies and select the best-performing ones for subsequent experimental evaluation.
3. Enhance the capability of neural network models by integrating a mechanism to generate explainability maps concurrently with class predictions during the forward pass.
4. Evaluate and compare the explainability aspects of the developed models by employing segmentation metrics.

1. Literature review

The following chapters investigate methods for detecting surface defects employing a machine-learning approach. The primary focus is on Convolutional Neural Network (CNN) and Vision Transformer (ViT) architectures as key image processing tools. The literature review also addresses explainability approaches for these networks. Finally, the application of these technologies is examined across various industries to enhance quality control.

1.1. Surface analysis and defect detection from images

There are several approaches to analyzing surface defects [1]:

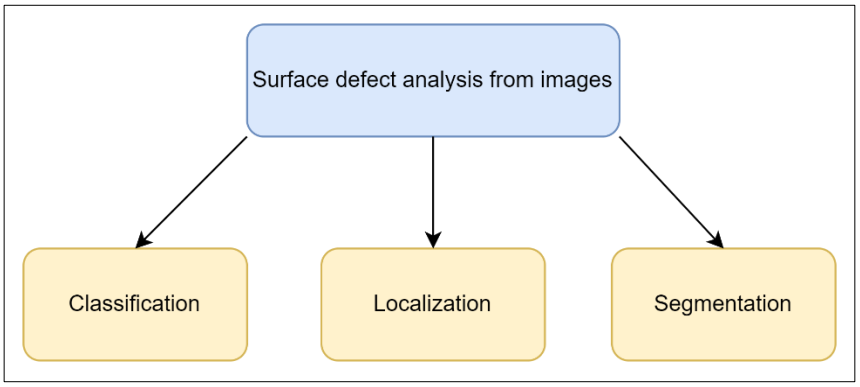


Fig. 1.1. Different approaches for surface defect analysis from images

The classification method is the simplest, aiming to distinguish between defective and non-defective samples or classify multiple defect categories if necessary. Localization identifies the precise location of the defect while simultaneously enabling classification. Segmentation isolates defects down to the pixel level. The selection of these methods depends on the specific requirements of the problem being solved.

The mentioned surface defect analysis methods are solved by distinguishing two directions of image analysis (Fig. 1.2). The first direction relies on expert knowledge-based features of visual data, while the second relies on features learned by artificial neural networks [1], [2].

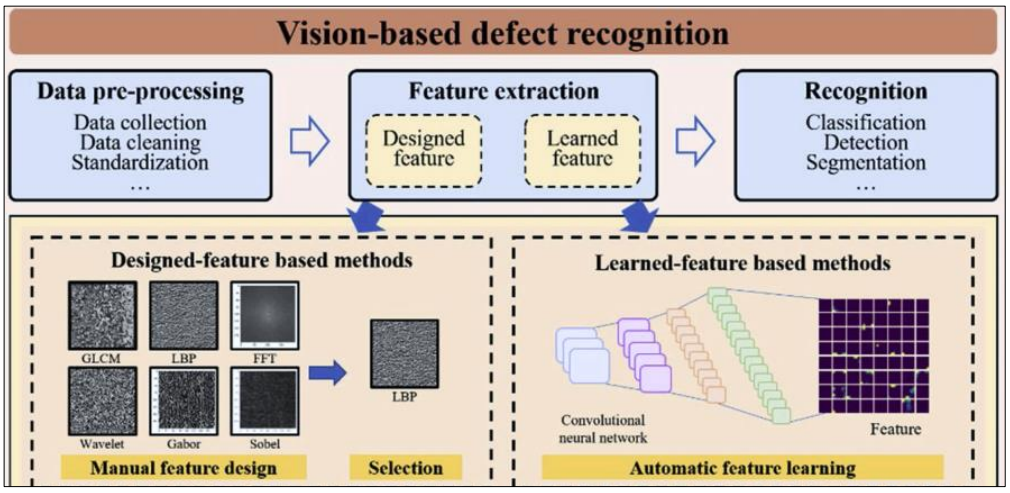


Fig. 1.2. The visual representation of defect recognition using vision-based technology [1]

The authors highlight the pros and cons of both approaches. While deep learning-based neural networks demand significant computational resources, this challenge is mitigated by rapid advancements in the accelerated computing field [3]. As a result, deep learning and neural networks are emerging as primary tools and research subjects in image analysis.

1.2. Convolutional neural networks

This section reviews the architectures of convolutional neural networks and their modifications, which aim to enhance accuracy, improve training efficiency, and reduce model complexity. Since the area of research is defect detection, which usually needs to be done in real-time [4], the focus is on convolutional neural networks with fewer parameters or floating-point operations (FLOPs). Such networks are more efficient and can operate faster. Typically, all emerging architectural solutions are trained using the ImageNet database [5] and their achievable classification accuracy is also reported.

1.2.1. EfficientNet

Convolutional Neural Networks (ConvNets), traditionally developed with fixed resource constraints, are often scaled up to enhance performance as more computational resources become available. In their study [6], the authors systematically examine the scaling of ConvNets and highlight the importance of achieving an optimal balance between the network's depth, width, and resolution (Fig. 1.3) to improve performance. From this analysis, they propose an innovative scaling technique that leverages a compound coefficient to uniformly adjust the network dimensions of depth, width, and resolution. This method is noted for its simplicity and effectiveness.
Extending their research, the authors employed a neural architecture search to design an optimized baseline network. This baseline was then scaled, resulting in the creation of a new suite of models termed EfficientNets. These models distinguish themselves by setting new benchmarks for accuracy and efficiency beyond previous ConvNets. Remarkably, the EfficientNet-B7 model not only achieves state-of-the-art accuracy rates of 84.4% top-1 and 97.1% top-5 on the ImageNet dataset but also presents a significant improvement in efficiency, being 8.4 times smaller and 6.1 times faster at inference compared to leading existing ConvNets.

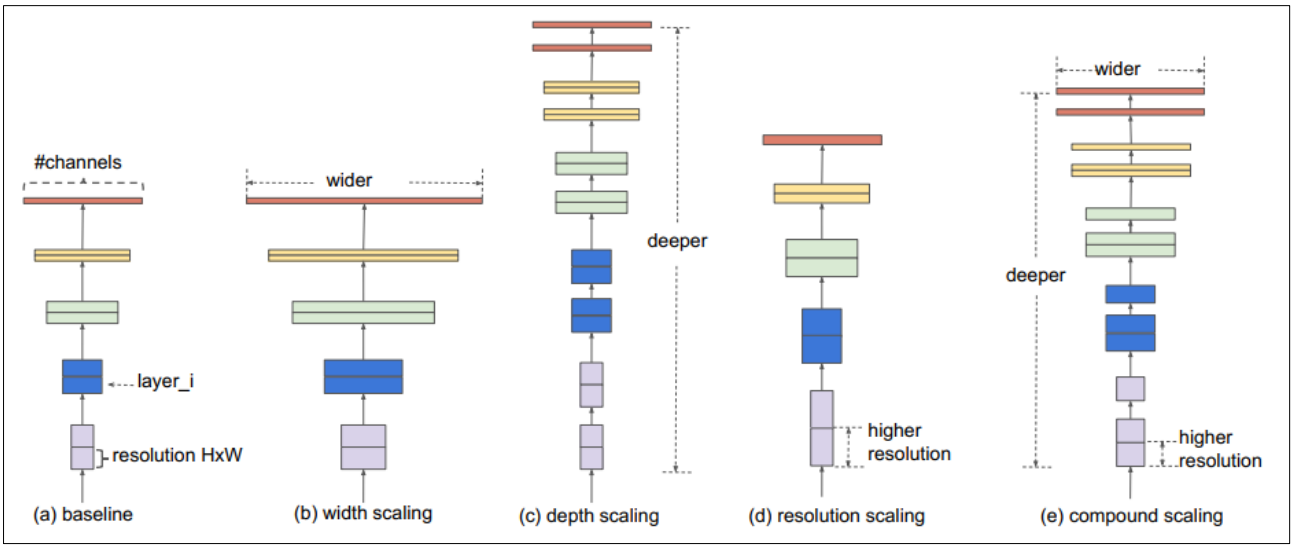


Fig. 1.3 Model scaling techniques [6]

In another paper, the same authors introduce EfficientNetV2 [7], further enhancing the architecture. The author's method for creating these models incorporates an innovative blend of training-aware neural architecture search [8] and scaling techniques, designed to simultaneously enhance training speed and parameter efficiency. This optimization process benefited from an enriched search space that included innovative operations like Fused-MBConv (Fig. 1.4).

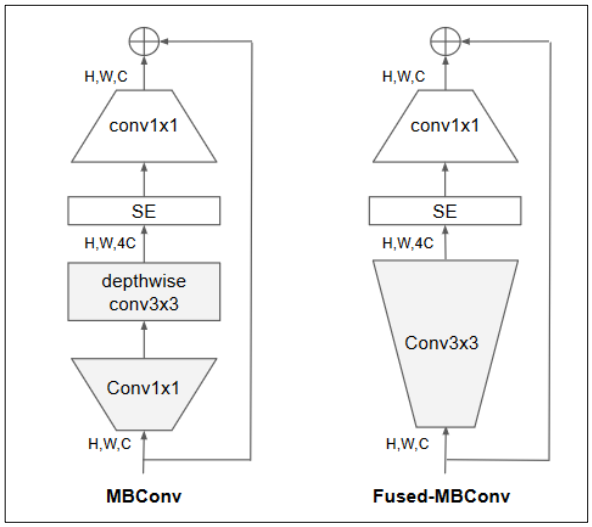


Fig. 1.4. MBConv block (EfficientNet) and Fused-MBConv block (EfficientNetV2) [7]

Through their comparative experiments, the authors demonstrate that EfficientNetV2 models not only train substantially faster than existing state-of-the-art models but also manage to be up to 6.8 times smaller in terms of parameter size. The introduction of progressively increasing image size during training, a method that typically risks a reduction in accuracy, is counterbalanced by an improved progressive learning technique. This technique adaptively adjusts regularization measures, such as data augmentation, alongside the image size increase, effectively preserving accuracy.

1.2.2. MobileNet

MobileNet [9] is a family of CNN architectures specially designed for devices with limited computing resources (Resource-constrained environments). The main goal of MobileNet is to find a balance between model accuracy and model size. The authors implement these features by using a combination of two different layers (depthwise separable convolutions, and pointwise convolutions). These layers help to extract information very similar to standard convolution layers, but the number of multiplication operations is reduced during the convolution operation. The first version of MobileNet was followed by MobileNetV2 [10] and MobileNetV3 [11], each of which introduced improvements and optimizations. The convolutional block of MobileNet is referenced below (Fig. 1.5).

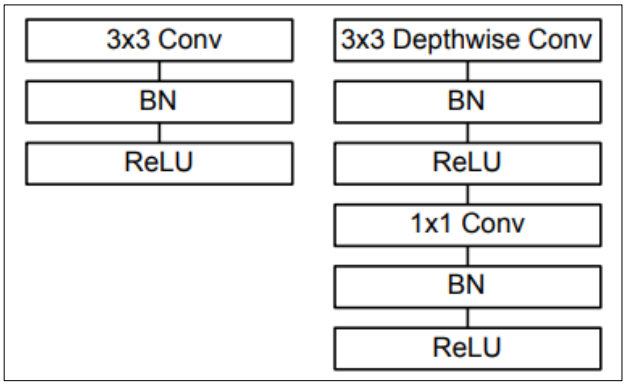


Fig. 1.5. On the left – standard convolutional block, on the right – MobileNet convolutional block [9]

1.3. Vision transformers

Traditional convolutional neural networks (CNNs) have been foundational in computer vision, excelling in tasks by extracting hierarchical features from images through localized filters. In contrast, the more recent Vision Transformers (ViT) leverage the transformer architecture [12], originally developed for natural language processing [13], to handle vision tasks. ViTs differentiate themselves by processing images in sequences of patches, like how transformers process words in sentences. This approach allows ViTs to consider global interactions between patches, offering a broader contextual understanding than the inherently local perspective of CNNs [14]. This shift from local to global processing marks a significant advancement in the field, enabling ViTs to achieve remarkable performance on various vision tasks, especially when trained on large-scale datasets.

1.3.1. ViT

Unlike traditional convolutional neural networks (CNNs) that process images through convolutional layers, a ViT [15] first divides an image into a sequence of fixed-size patches. These patches are then flattened and linearly embedded (transformed into a series of numbers that can be processed by the model). Since the transformer architecture does not inherently consider the order of the input, positional encodings are added to the patch embeddings to retain information about the position of each patch in the original image. This is crucial for understanding the spatial relationships between different parts of the image.

The embedded patches, along with their positional encodings, are then fed into a series of transformer encoder layers. Each layer consists of multi-head self-attention mechanisms and feed-forward neural networks. The self-attention mechanism allows the model to weigh the importance of different patches relative to each other, capturing the global context of the image. For tasks like image classification, the output of the transformer encoder is typically passed through additional layers (often a simple neural network) to produce the final classification. ViTs are trained using large datasets and powerful computational resources [16]. They have demonstrated impressive performance on various image recognition tasks, sometimes even surpassing traditional CNNs. A key advantage of Vision Transformers is their ability to capture global dependencies within an image, as opposed to CNNs which primarily focus on local features. Vision transformer architecture is referenced below (Fig. 1.6).

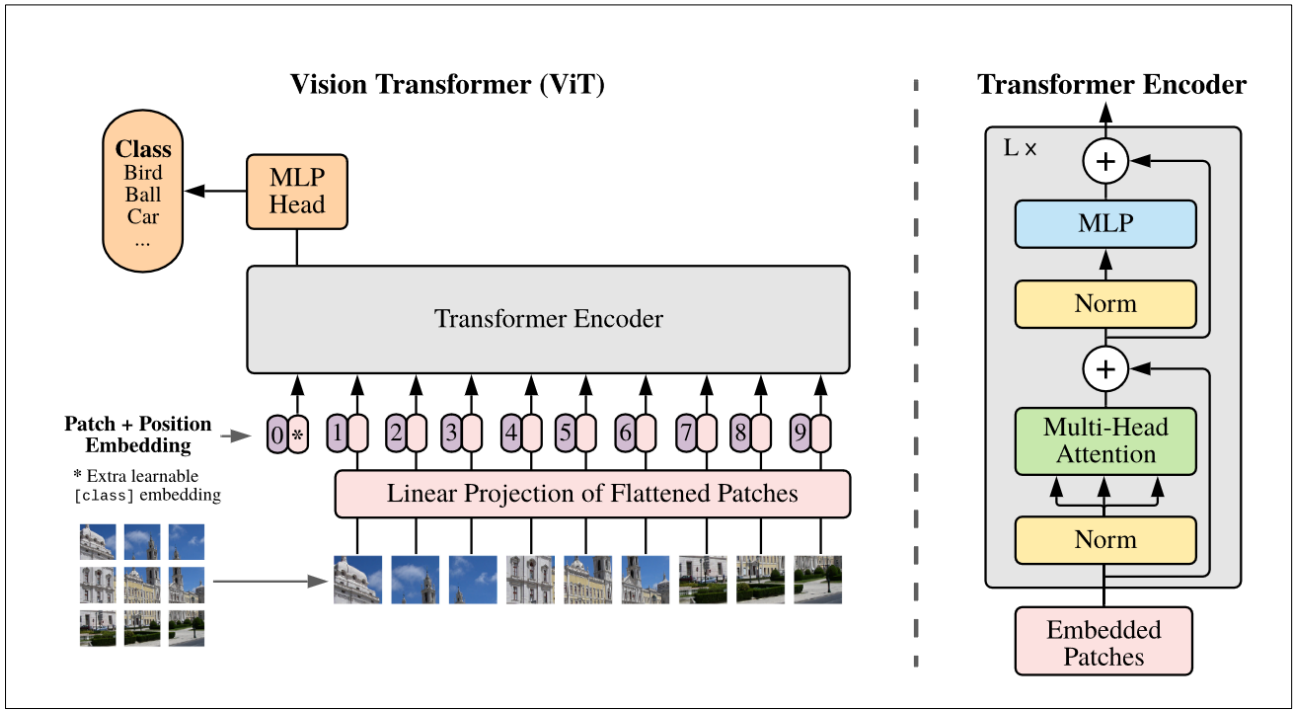


Fig. 1.6. ViT architecture diagram [15]

Authors in their paper "How do vision transformers work?" [17] compare CNN and ViT architectures. The main difference highlighted between architectures is that multi-head self-attention improves generalization by flattening the loss landscapes.

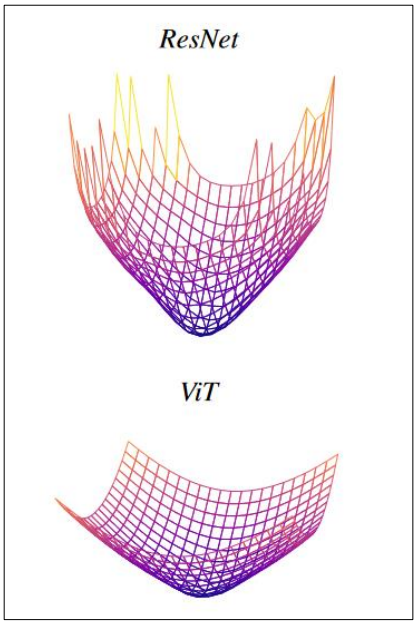


Fig. 1.7. Loss landscape visualizations of ResNet and ViT [17]

1.3.2. DeiT

Researchers in their paper [18] introduced another kind of transformer called DeiT (data-efficient transformer). The core concept behind the architecture is very similar to ViT. DeiT operates by treating images as sequences of patches. DeiT introduced a novel training methodology involving a distillation token, which is an additional learnable vector that aids in distilling knowledge [19] from

a teacher model (typically a pre-trained CNN or Transformer) to the Transformer model. This approach helps improve the training efficiency and effectiveness of the DeiT model, especially when labeled data is limited. The key advantages of the model are highlighted:

1. Efficiency: DeiT is designed to be more data efficient than traditional CNNs, achieving comparable results on benchmarks like ImageNet with less data.
2. Scalability: the transformer architecture scales well with the amount of data and the size of the model, allowing DeiT to effectively leverage larger datasets and model capacities.

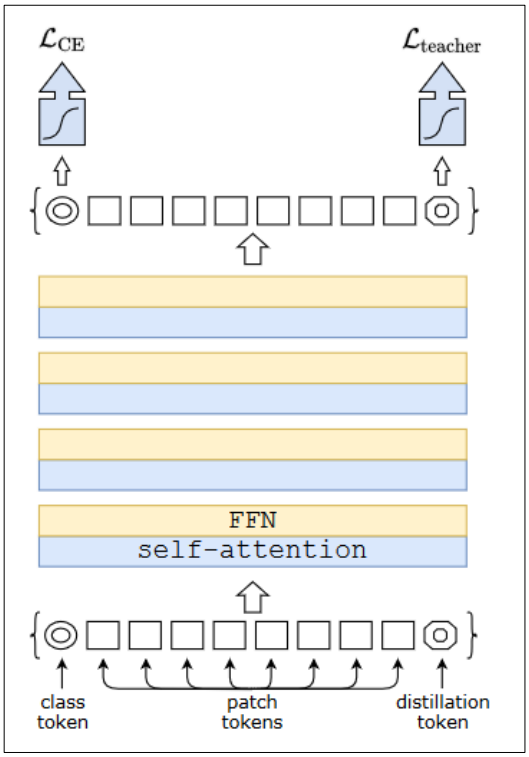


Fig. 1.8. DeiT model architecture diagram with a distillation process [18]

1.4. Overview of neural network models

In this literature review, the advancements, and characteristics of both CNNs and ViTs were explored. Through comparative analysis, it is evident that while CNNs such as EfficientNet and MobileNet are renowned for their efficiency and adaptability in handling diverse image recognition tasks with constrained computational resources, vision transformers like ViT and DeiT demonstrate comparable performance on tasks requiring understanding of complex patterns and large-scale training datasets. The Table 1.1 highlights the distinctions in accuracy, and floating-point operations among different model architectures.

Table 1.1. Comparison of models.

Model	Release year	Accuracy (ImageNet1K Top-1), %	Floating point operations (GFLOPs)
EfficientNet_B4	2019	83.38	4.39
EfficientNetV2_S	2021	84.23	8.37
MobileNetV2	2018	71.88	0.30
MobileNetV3_large	2019	74.04	0.22
ViT_tiny	2021	79.10	1.30
DeiT_tiny	2021	74.50	1.08

EfficientNetV2_S has the highest accuracy at 84.23% but also has the highest GFLOPs at 8.37, indicating higher computational complexity. MobileNetV3_large has the lowest GFLOPs at 0.22 but a relatively lower accuracy of 74.04%. Models like ViT_tiny and DeiT_tiny show a balance between accuracy and computational cost with 79.1% and 74.5% accuracy, and 1.3 and 1.08 GFLOPs respectively.

1.5. Neural networks explainability

As the application of advanced machine learning models like CNNs and ViTs expands across critical sectors such as healthcare [20], finance [21], and autonomous driving [22], the imperative for explainability in these models grows [23], [24]. CNNs, known for their deep hierarchical structure for feature extraction, and ViTs, which utilize self-attention mechanisms to process images, both achieve high levels of performance yet often operate as "black boxes" with decision-making processes that are opaque and complex [25]. This chapter explores the methods of making such models interpretable and transparent, discussing the internal mechanisms.

1.5.1. Class activation mapping (CAM) for CNN’s

Class Activation Mapping (CAM) is a technique that enables the visualization of the regions within an image that are significant for predicting a class label in CNNs. This method is particularly useful for understanding which features in an image contribute to the classification decision made by the network. Introduced by Zhou et al. (2016), CAM [26] works by using the global average pooling layer in CNNs to directly relate the spatial activations to the output class probabilities. This not only helps in interpreting the decision-making process of CNNs but also aids in the diagnosis of whether the network is focusing on the relevant objects in the image. Further, CAM has been effectively used in various applications, including image classification, object detection, and segmentation, providing a tool for improving model transparency and reliability by highlighting potential biases and errors in the training process. This method's ability to produce visual explanations without requiring architectural modifications or additional parameter training makes it an asset in the toolbox of techniques for enhancing the interpretability of CNNs. Fig. 1.9 demonstrates that the global average pooling layer produces the spatial average of each unit's feature map in the final convolutional layer. The final output is generated by applying a weighted sum to these values. In a similar manner, class activation maps are derived by computing a weighted sum of the feature maps from the last convolutional layer.

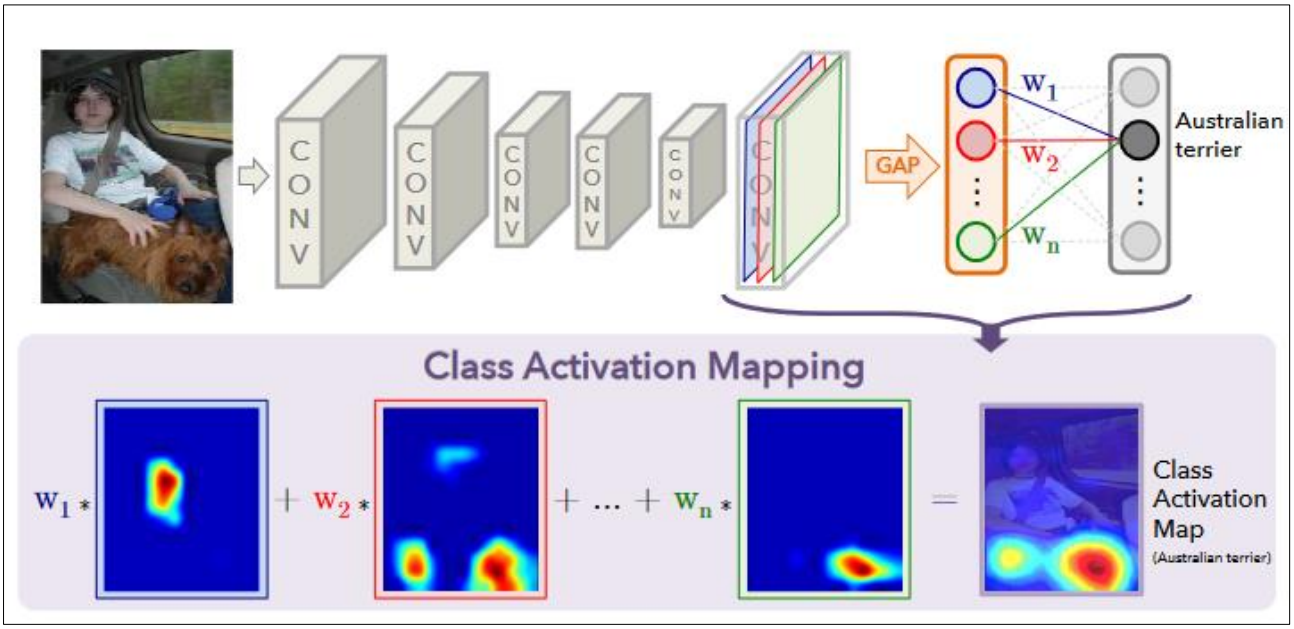


Fig. 1.9. Visual representation of class activation map calculation. The CAM highlights the class-specific discriminative regions [26]

Fig. 1.10 highlights the differences in the CAMs for a single image when using different classes to generate the maps. It is observed that the discriminative regions for different categories are different for a given image. For the highest probability class (palace), the attention map area (red) is the biggest and the most precise.

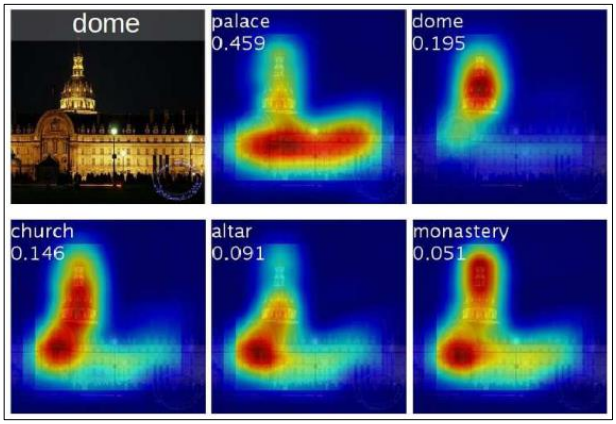


Fig. 1.10. Examples of CAMs generated from the top 5 predicted categories for the given image with ground-truth as dome [26]

1.5.2. Attention Rollout for transformer models

Authors in their paper [27] present explainability methods for transformer models. They are called “attention rollout” and “attention flow” and their main goal is to compute attention scores to input tokens at each layer, by taking the raw attentions of the layer as well as those from the precedent layers. In the transformer model, each layer's self-attention mechanism integrates information from the attended embeddings of the preceding layer to generate new embeddings for each token. Consequently, as the information from various tokens becomes progressively intermingled across the transformer's layers, the attention weights in the *i*-th self-attention layer cannot be directly interpreted as attention to the input tokens or embeddings from the input layer. This intermingling renders

attention weights unreliable for explaining which parts of the input are most crucial in generating the output. To answer this question the authors, propose the computation of the information flow graph of a transformer model:
Given transformer model attention module with residual connections, the values in layer *l+1* are computed as

$$V_{l+1} = V_l * W_{att}V_l, \tag{1.1}$$

where, W_{att} – attention matrix, V_l – input values of *l* layer.

Simplified:

$$V_{l+1} = (W_{att} + I)V_l, \tag{1.2}$$

where, I – identity matrix.

Keeping in mind residual connections this results in the following:

$$A = 0.5W_{att} + 0.5I, \tag{1.3}$$

where, A – raw attention updated by residual connections.

The last equation approximates how information propagates through the self-attention layers. Using this graph, attention weights across all layers can be analyzed, and each layer's attention weights can be mapped back to the input tokens. In Fig. 1.11, example (a), the model predicts the word 'his'. Thus, the model should focus on the word 'author' rather than 'Sara'. Both Attention Rollout and Attention Flow align with this expectation, while the final layer of Raw Attention does not align with the model's prediction and shows significant variation across different layers.

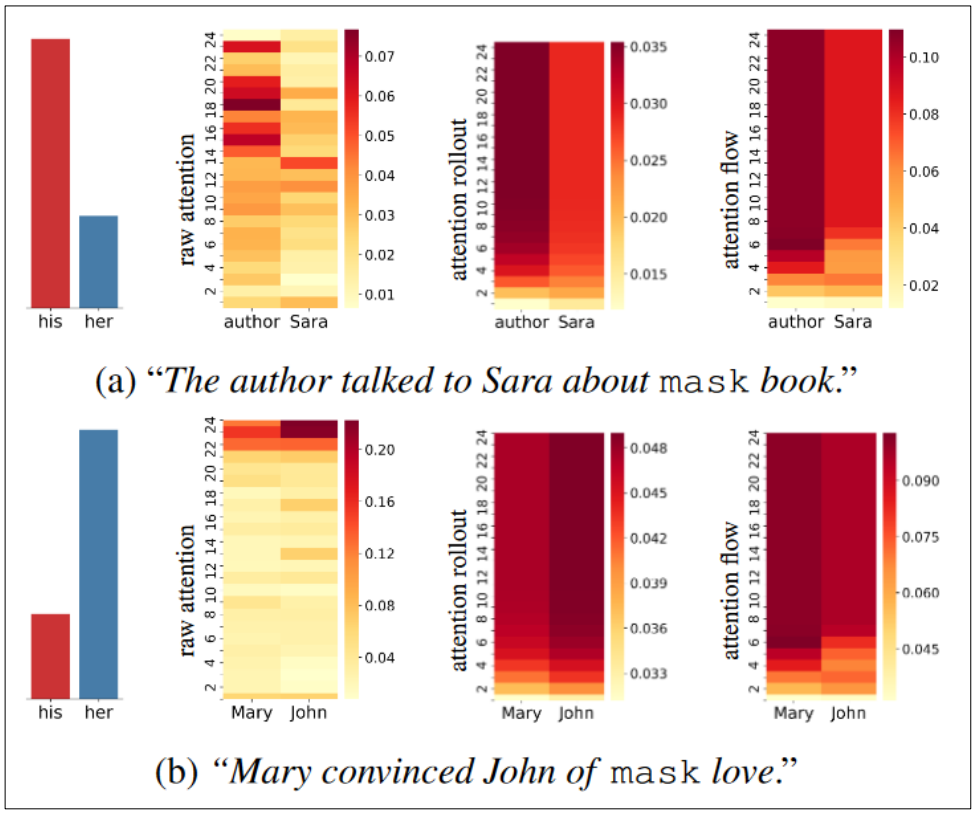
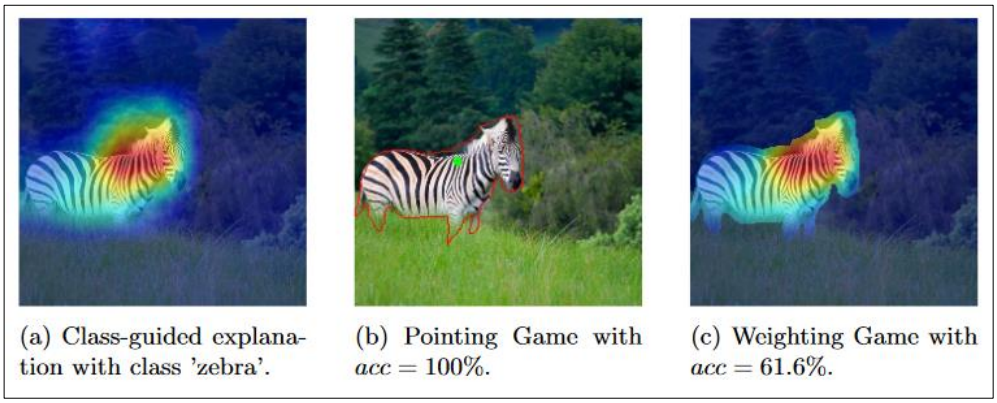


Fig. 1.11. Bert attention maps. Looking at attention weights from the mask embedding to the two potential references for it, e.g. "author" and "Sara" in (a) and "Mary" and "John" in (b). The bars, at the left, show the relative predicted probability for the two possible pronouns, "his" and "her" [27].

1.5.3. Evaluating the quality of explainable methods

When working with neural network explainability, assessing the quality of explanation heatmaps can be challenging. Several methods have been developed to address this issue. The Pointing Game metric [28] evaluates the accuracy of an explainability map by determining whether its highest-magnitude pixel is located within the correct class's segmentation map, recording individual instances as either hits or misses. Similarly, the authors in [29] introduced the Weighting Game metric, which measures the proportion of an explainability map's mass or magnitude that lies within the correct class's segmentation map. This approach provides greater flexibility in evaluating individual explanations, as it is not binary like the Pointing Game. Fig. 1.12. illustrates the differences between the Weighting Game and the Pointing Game.



(a) Class-guided explanation with class 'zebra'. (b) Pointing Game with $acc = 100\%$. (c) Weighting Game with $acc = 61.6\%$.

Fig. 1.12. Differences of the weighting game and pointing game [29]

1.6. Applications of defect detection using explainability of neural networks

The integration of deep neural networks in defect detection has revolutionized quality assurance across industries by enhancing detection accuracy and reducing operational costs [30]. However, the opacity of these systems often limits their trustworthiness and scalability. Network explainability addresses this challenge by making the decision-making processes of neural networks transparent and interpretable. This chapter focuses on the applications of network explainability in defect detection, emphasizing how making model decisions understandable can improve both the performance and reliability of automated systems. The subsequent chapters explore the methodologies, case studies, and impacts of explainable AI in industrial defect detection.

1.6.1. Mini/micro-LED-chip defect recognition

Paper [31] presents a small-scale ViT network using Convmixer [32] and interpreting local and global information. First, the image is split into individual windows, then each window is encoded into the multichannel space, thus creating a display map. This is fed to a convolutional neural network that works similarly to ViT, so the information is interpreted both globally and locally. Fig. 1.13 presented Grad-CAM [33] visualizations of the last layers of the various tested networks. From the visual material in the last column, the network of the presented architecture focuses more precisely on the defective locations. Also, the network achieves over 3% higher accuracy than MobileNetV3 or MobileViT [34] considering that it is the smallest network (0.138 GFLOPs) compared to the latter (0.4 and 0.7 GFLOPs). In this situation, explainability facilitates a more effective comparison between models.

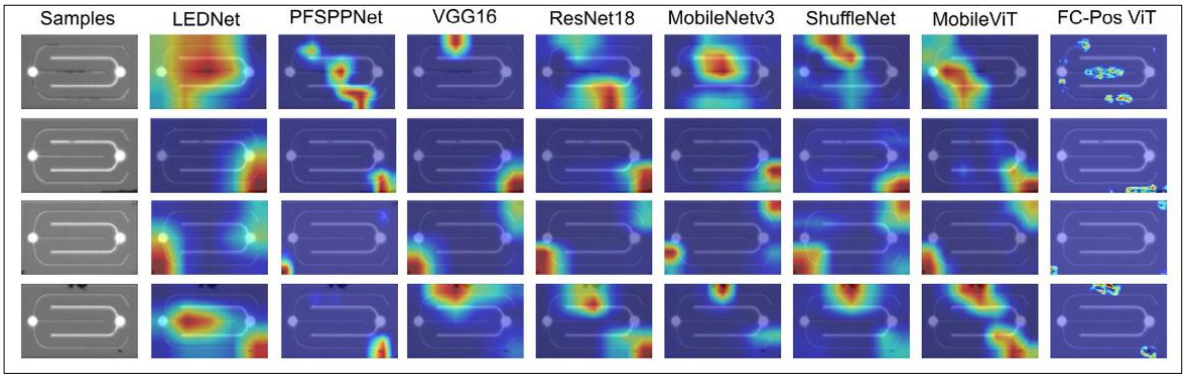


Fig. 1.13. The visualization of the Grad-CAM of different models [31]

1.6.2. An explainable laser welding defect recognition

Paper [35] explores advancements in vision-based online defect recognition for laser welding, emphasizing the enhancement of quality control systems through more sophisticated imaging techniques. While visual signals offer richer quality information compared to one-dimensional signals, the abstract nature of the information they contain poses challenges in interpretation and application. To address these challenges, the paper introduces a novel approach to improve the explainability of convolutional neural networks (CNNs) used in laser welding defect recognition (LWDR).

The proposed method, termed Class Activation Mapping with Multi-Scale Fusion Features (CAM-MSFF), leverages a multi-scale features adaptive fusion technique that encompasses three key processes: feature squeeze, feature mapping, and feature recalibrating (Fig. 1.14). This approach is designed to enhance the model's ability to interpret multi-scale features.

Explainability tests reveal that CAM-MSFF provides more precise and human-comprehensible explanations of the model’s decision-making processes.

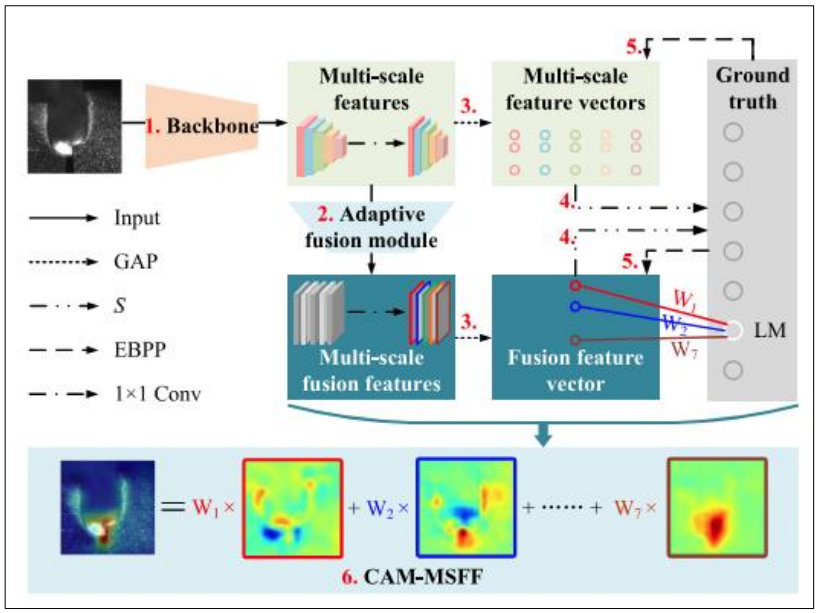


Fig. 1.14. The overall framework of CAM-MSFF [35]

1.6.3. Automated surface defect detection based on a bilinear model

The paper [36] introduces a new approach using a bilinear model, specifically adapted from the VGG16 [37] architecture, named Double-Visual Geometry Group16 (D-VGG16). This model effectively extracts both global and local features critical for detecting complex surface textures.
The paper also introduces the use of Gradient-weighted Class Activation Mapping (Grad-CAM) [33] to localize defects. Grad-CAM generates heat maps from D-VGG16's output, which are then processed using a threshold segmentation method for defect identification. This method improves interpretability and reduces the likelihood of false positives.
Tested on two open-source and two industrial datasets, the proposed method demonstrated great performance, achieving an average precision above 99%. Fig. 1.15 presents the overall network structure. There are two feature extraction networks. Features from both networks are concatenated and used for class prediction. Localization of the defective location is made through the Grad-CAM algorithm backpropagating through one of the feature extractors.

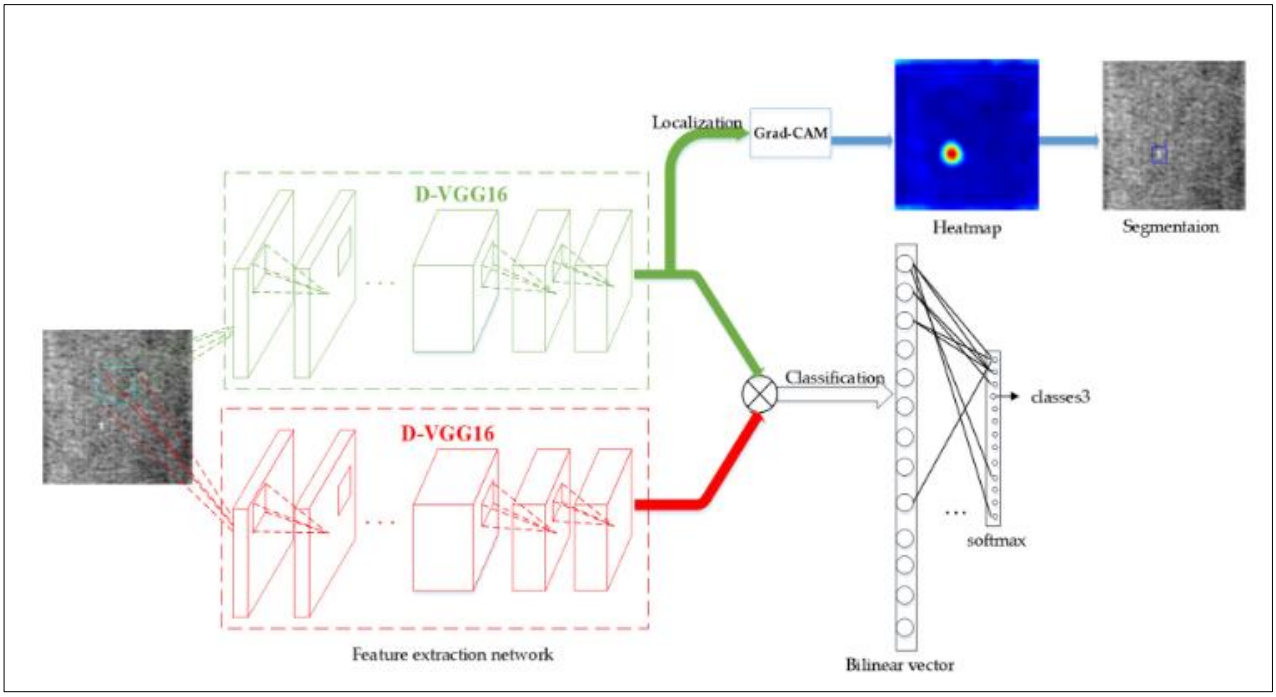


Fig. 1.15. Neural Network Structure [36]

1.6.4. A post hoc analysis of deep learning for defect classification of TFT-LCD panels

Authors in their paper [38] use XAI techniques to analyze the predicted results of defective TFT-LCD panels. VGG16 architecture model was trained and the defects were visualized using a layer-wise relevance propagation-based method [39]. Each neuron has a specific level of contribution (certain relevance), which is redistributed from the output of each neuron to the input in a top-down manner, preserving the overall contribution. As shown in Fig. 1.16 (a , the heatmap highlights the defect area, whereas Fig. 1.16 (b) demonstrates that the heatmap is dispersed in the background pattern. The visualization technique faces limitations in analysis due to the unstable display of prediction results. Consequently, researchers have supplemented this approach by defining classification rules for the learned model using a decision tree.

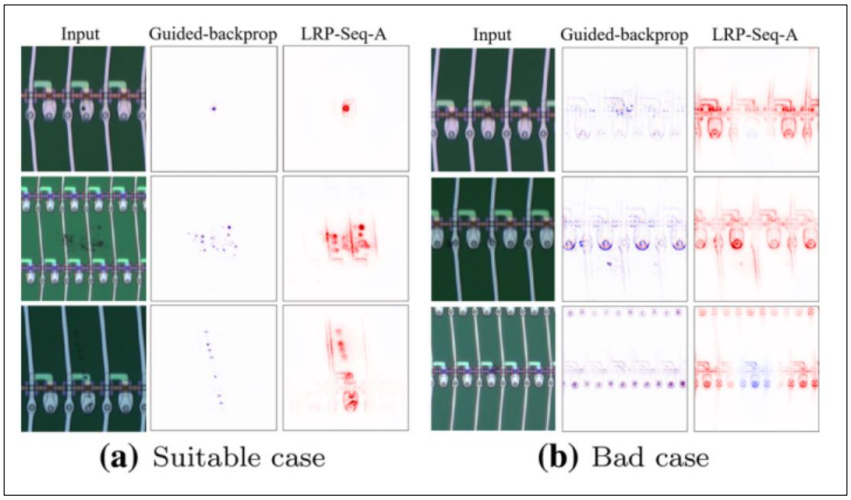


Fig. 1.16. Examples of visualization results [38]

1.7. Conclusions

For further research, lightweight models that perform well on the ImageNet1K database were selected for analysis. EfficientNetV2_s [7] and MobileNetV3_large [11] were chosen as convolutional neural networks (CNNs), while ViT_tiny [15] and DeiT_tiny [18] were selected as vision transformers (ViTs). The Class Activation Mapping (CAM) [26] method will be utilized for CNN explainability, and the attention rollout technique [27] will be used for ViT explainability. The quality of the explainability outputs will be evaluated using the pointing game metric [28] alongside traditional segmentation metrics. From an applications perspective [31], [35], [36], [38], examples demonstrate the interest of explainability methods for defect localization and model decision interpretation, validating the relevance of this research.

2. Methods and concepts used in the research

2.1. Development environment setup

In this section, the specifics of the development environment used for this research project are described. The setup is designed to streamline experimentation, foster modularity, and simplify tracking and reproducibility. The core components used for the research are described below.

The main ideas behind the creation of a development environment:

- Minimal boilerplate code (easily adding new models, datasets, tasks, experiments, and different accelerator configurations). This is done through dynamic hydra configuration.
- Logging experiments to one place for easier comparison of performance metrics.
- Hyperparameter searching integration.

Development language:

Python is the primary programming language for this project. It is renowned for its simplicity and readability, which makes it particularly appealing for scientific computing. It supports various programming paradigms and features an extensive ecosystem of libraries, making it an ideal choice for machine learning projects.

Libraries and frameworks:

- PyTorch: python package offering two high-level capabilities: tensor computation (similar to NumPy) with GPU acceleration, and deep neural networks utilizing a tape-based autograd system.
- PyTorch Lightning: a lightweight PyTorch wrapper for high-performance AI research and faster prototyping.
- TorchMetrics: a set of PyTorch metric implementations accompanied by a user-friendly API for creating custom metrics.
- Hydra: a framework for configuring complex applications, featuring the ability to dynamically create a hierarchical configuration through composition and override it via config files and the command line.
- Optuna: framework agnostic hyperparameters searching library.
- OpenCV: an open-source computer vision and machine learning software library.
- WandB: a machine learning development platform that allows tracking and visualize various aspects of the model training process in real-time.
- Hugging Face: a library for storing models and datasets.

The diagram (Fig. 2.1) outlines a setup for a developed project that uses configuration files and Python scripts for training and testing models. Here is a breakdown of each component and how they interconnect:

Configuration Section

This part of the diagram illustrates how configuration files (*train.yaml*, *test.yaml*, *model.yaml*, etc.) are used to manage different aspects of the project, such as data preprocessing, model parameters, and training settings. These files are in YAML format, which is common for various configurations.

Hydra Loader

The diagram shows how Hydra loads all configuration files and combines them into a single configuration object (DictConfig). This unified configuration object simplifies the management of settings across different modules and aspects of the project, such as data handling, model specifics, callbacks, logging, and the training process.

Train/Test Script

- This section represents the operational part of the project. Scripts *train.py* and *test.py* are required for training and testing the model.
- DictConfig: The combined configuration object passed to these scripts, guiding the instantiation of the subsequent components.

Instantiating Objects

- LightningDataModule: manages data loading and processing specific to training, validation, and testing phases.
- LightningModule (model): defines the model, including the computation that transforms inputs into outputs, loss computation, and metrics.
- Callbacks: provide a way to insert custom logic into the training loop, such as model checkpointing, early stopping, etc.
- Logger: handles the logging of training, testing, and validation metrics for monitoring progress.
- Trainer: the central object in PyTorch Lightning that orchestrates the training process, leveraging all the other components.
- The trainer uses the model, data module, logger, and callbacks to execute the training process through the *trainer.fit()* method, integrating all the configuration settings specified through Hydra.

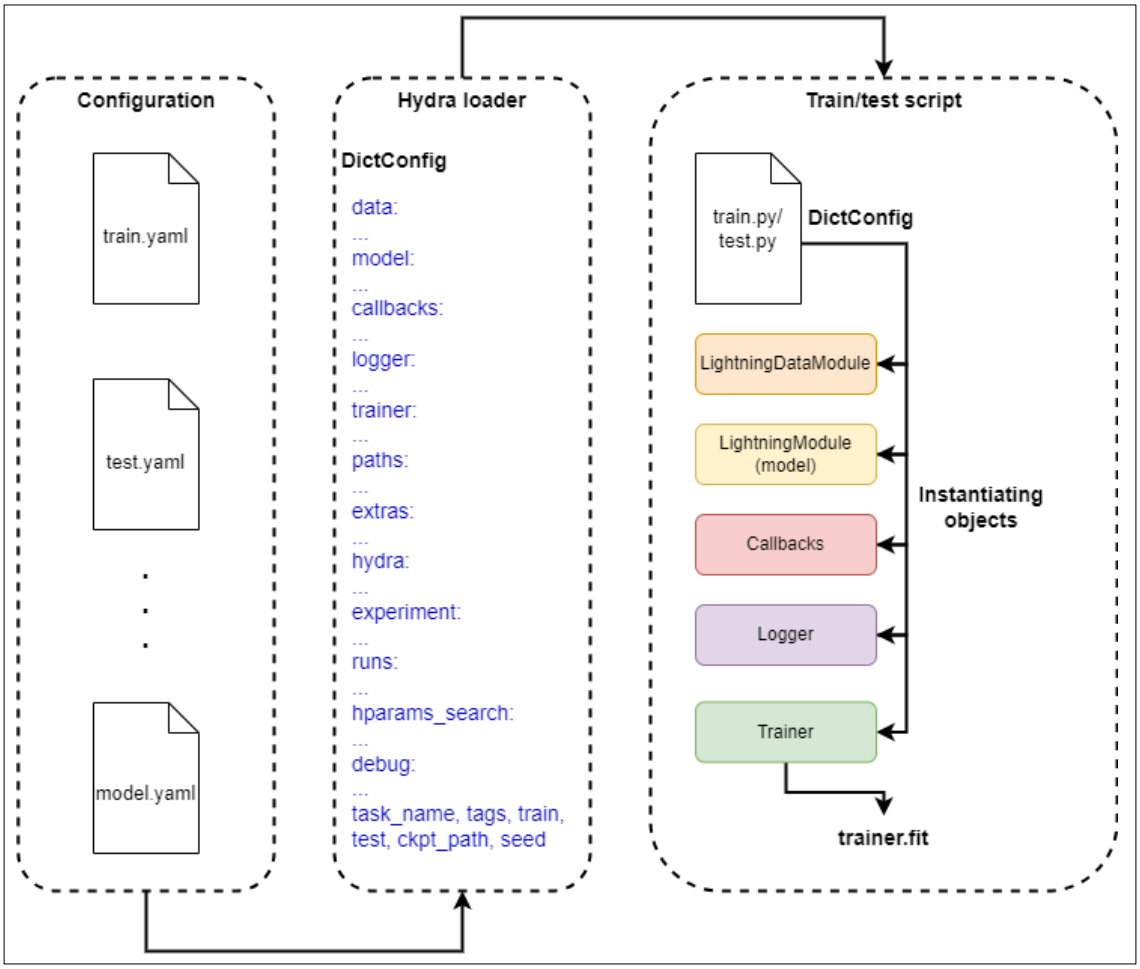


Fig. 2.1. The flow diagram of model training and testing

Once the development environment is properly set up, the workflow comes to 4 steps:

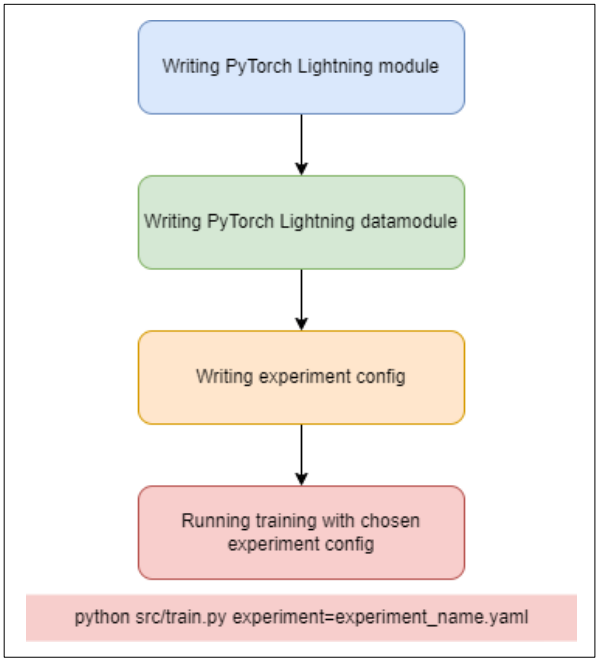


Fig. 2.2. The workflow diagram

2.2. Multi-head CNN with explainability output

An additional output is incorporated into the generic binary image classifier to enhance prediction explainability, serving as a rough segmentation or localization of defects. A Class Activation Map (CAM) [26] is employed for this purpose, as it offers one of the simplest methods for explaining the model with minimal modifications to the architecture and few additional calculations. Moreover, since the output consists of a single neuron (binary classification with sigmoid activation), this general explainability and rough defect estimation remain constant and specific to the given prediction and its weights. The formula for generating the CAM-based output is as follows:

$$X_{CAM} = X_{Latent} * W_{B,} \tag{2.1}$$

where, X_{CAM} – CAM output matrix which is equal to X_{Latent} – feature extractor latent space tensor before global average pooling and W_B – binary prediction output neuron weights dot product.

The diagram of the modified convolutional neural network structure is given in Fig. 2.3.

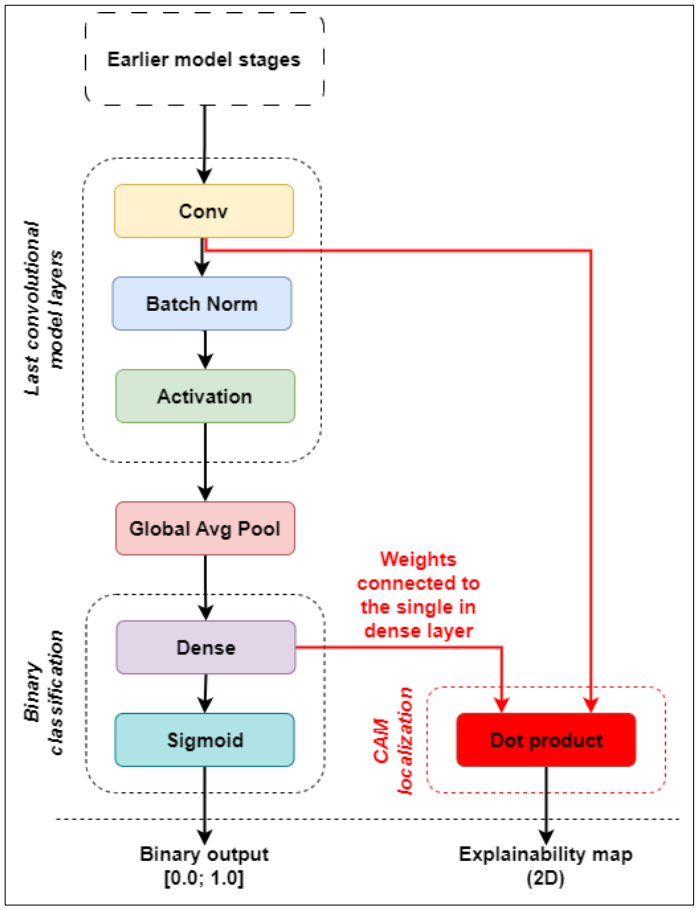


Fig. 2.3 Diagram of modified convolutional neural network structure

Additionally for experiments, the model architecture's (EfficientNetV2_s and MobileNetV3_large) final feature extraction stage was taken out (Fig. 2.4). This modification increases explainability output resolution 2 times resulting in 1/16 of original input resolution. The experiments were also conducted on full architecture which gives two times lower resolution (1/32 of the original input resolution). The intuition behind using higher-resolution feature maps is that they might allow for more detailed and granular visual explanations of model decisions.

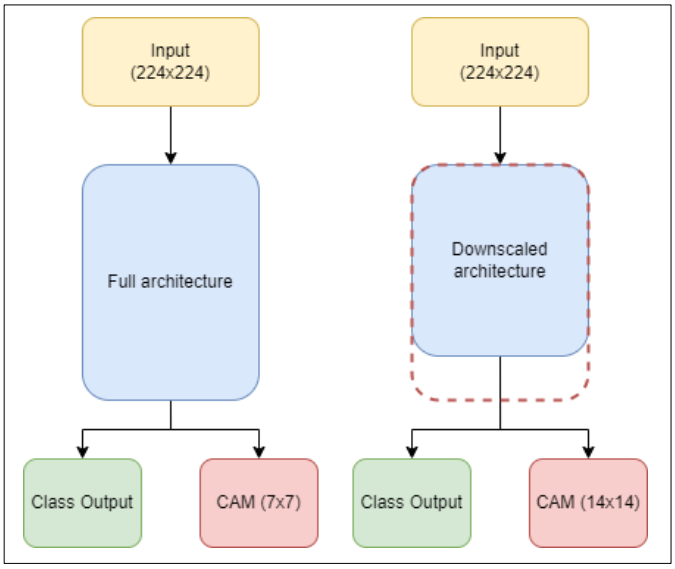


Fig. 2.4. The comparison of CNN full and downscaled architecture outputs

2.3. Multi-head ViT with explainability output

Attention Rollout [27] is a technique used to provide explainability for Vision Transformer (ViT) models. Vision Transformers use attention mechanisms to determine the importance or relevance of different parts of the input data (image patches in this case). Each attention head in the transformer layers computes a set of attention weights, indicating how much focus should be given to each patch when considering a specific patch.

Attention Rollout involves combining these attention weights across all the layers of the transformer in a way that represents the overall flow of attention through the network. This is done by multiplying the attention matrices of successive layers. The idea is to track how the attention to each patch evolves from the input layer to the output layer.

In every Transformer block, an attention matrix A_{ij} is created, which determines the flow of attention from the token j in the previous layer to token i in the next layer. Multiplying the matrices between each pair of layers calculates the overall attention flow between them. Additionally, there are residual connections. These can be incorporated by adding the identity matrix I to the attention matrices of each layer, resulting in $A_{ij} + I$. With multiple attention heads, the "Attention Rollout" study [27] suggests averaging the heads. Other methods, such as using the minimum or maximum fusion of heads, are also viable. Additionally, the quality of attention maps can be enhanced by discarding a specific percentage of the lowest attention scores. The attention rollout matrix at the layer L is recursively computed as follows (it is also necessary to normalize the rows to ensure the total attention flow remains 1):

$$AttentionRollout_L = (A_L + I) \cdot AttentionRollout_{L-1} \tag{2.2}$$

The result of the attention rollout can be visualized as a heatmap (upscaled to the original input dimensions) overlaid on the original image. This heatmap shows which parts of the image were most influential in the model's decision-making process (similarly to CAM in the CNNs). Regions with higher attention weights are highlighted, indicating they played a more significant role in the model's output (e.g., in classifying the image). The diagram of the modified ViT architecture with explainability output is given in Fig. 2.5.

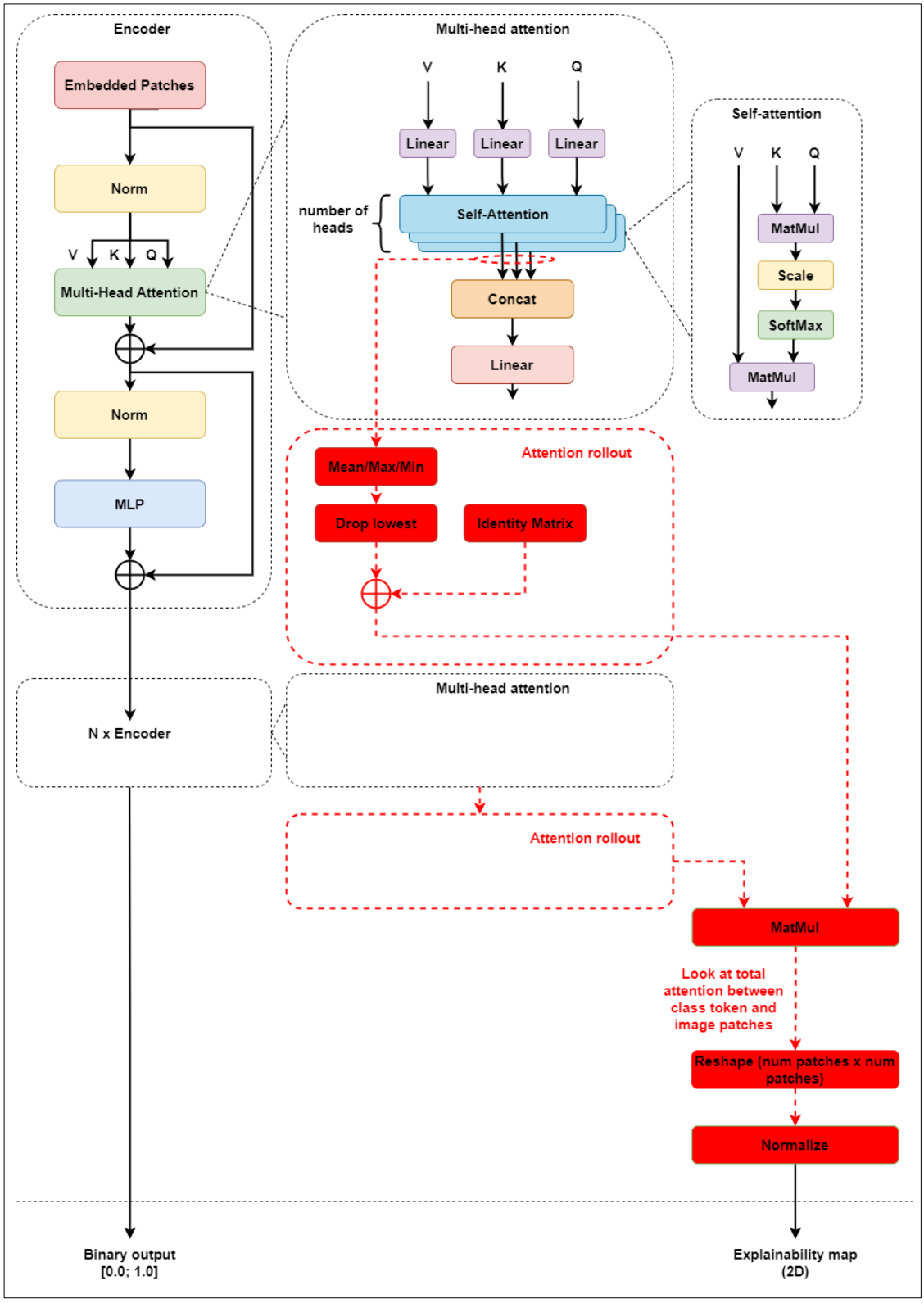


Fig. 2.5. Diagram of modified transformer model structure

2.4. Datasets

2.4.1. Overview

In this study, two different open-source image datasets are utilized, each containing various defects in visual data. The first dataset, the printed circuit board (PCB) defects dataset , includes 1386 high-

resolution images of printed circuit boards of varying sizes. The defects featured in this dataset are missing holes, mouse bites, open circuits, shorts, spurs, and spurious copper. This dataset is semi-artificial, as the defects were generated using image editing software. It includes both region and class-wise annotations for each defect. The second dataset is the Gear Inspection Dataset (GID), created for Baidu's 'National Artificial Intelligence Innovation Application Competition.' It contains two thousand grayscale images with a total of 28,575 annotations for three types of defects. Each image is accompanied by a separate JSON file detailing the image name, label categories, bounding boxes, and segmentation polygons. However, the label categories are represented by numbers rather than specific defect types, making it challenging to identify their similarities. The summary of the datasets is given the Table 2.1. Cropped samples of images are given in Fig. 2.6.

Table 2.1. Datasets summary

Parameter	**PCB defects**	**GID**
Samples	1386	2000
Width [px]	1586-2530	1000-1400
Height [px]	2240-3056	1500-2000
Type	RGB	Grey

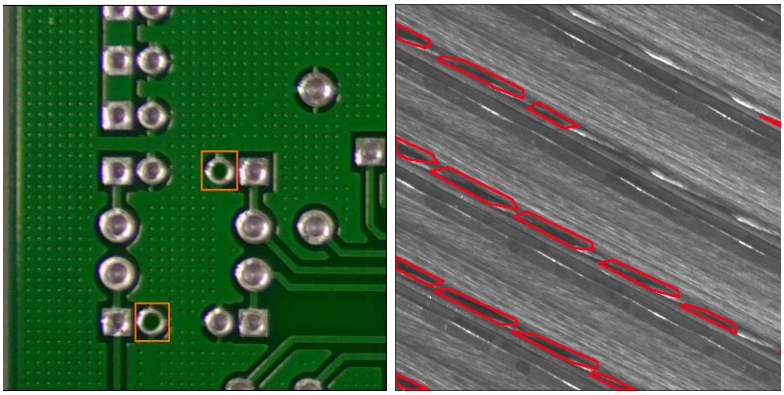


Fig. 2.6. Cropped image samples from datasets with defects highlighted. Left – PCB dataset, right – GID dataset

2.4.2. Preprocessing

This research employed a binary image classification method. To achieve this, each dataset was organized into two classes/directories: defective and good. Pixel-wise and region labels were not used during training. Given the high resolution and varying sizes of the original samples, images were divided into smaller, overlapping regions using a sliding window approach. The PCB and GID datasets were split into 224x224 pixel regions with a 2-pixel overlap. These regions were labeled as defective or good based on whether the defective area exceeded a 1% threshold. This threshold was selected considering the size and characteristics of partially visible defects. Additionally, all defect classes in both the PCB and GID datasets were combined into a single defect category. The preprocessing method is shown in Fig. 2.7.

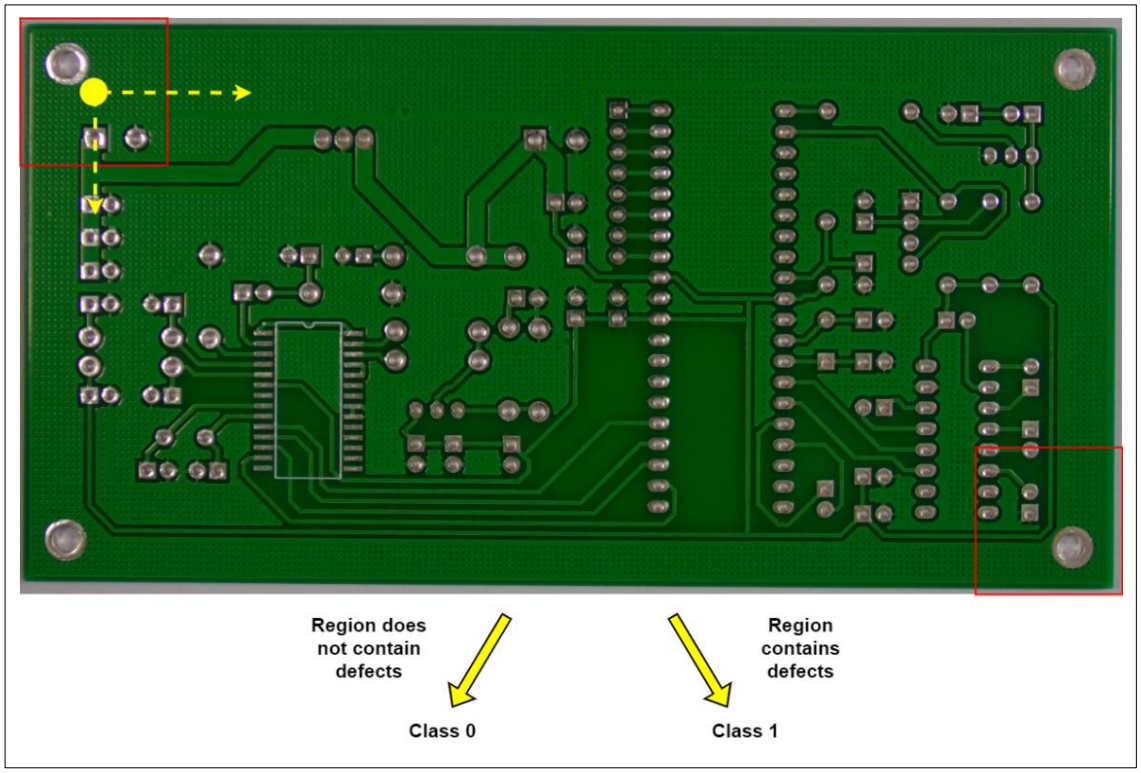


Fig. 2.7. Image preprocessing technique with tiling

The properties of processed data are given in the Table 2.2.

Table 2.2. The properties of processed data

Parameter	**PCB defects**	**GID**
Samples	68270	77978
Train/Validation/Test ratio	0.7/0.1/0.2	
Tile size (width x height)	224 x 224	
Tiling overlap	2 px	
Allowed defective area for good tile	< 1 % (501 px)	

For training, random horizontal and random vertical flip augmentations were used with occurrence probability of 0.5. The tile size of 224x224 was chosen because of the fixed input shape of pre-trained vision transformer models.

2.5. Explainability output segmentation evaluation

2.5.1. Evaluation methodology

Explainability evaluation using annotations from the original dataset was conducted by normalizing the activation map output to a range of 0 to 255 and applying a threshold with a lower intensity value of 127. This output label was considered only when the binary prediction exceeded 0.5, indicating the model predicted the image as defective. Otherwise, the output was black (defect-free), signifying a defect-free image. The diagram illustrating the evaluation of the explainability output is shown in Fig. 2.8.

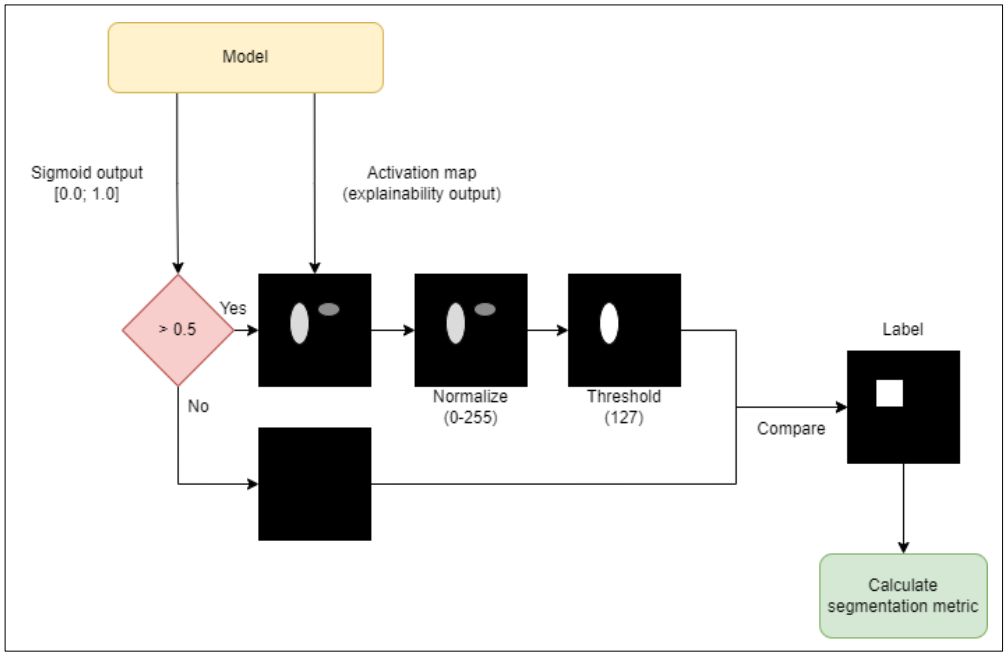


Fig. 2.8. Explainability output evaluation diagram

2.5.2. Segmentation metrics

Explainability maps were evaluated using traditional segmentation metrics to quantify how well the maps align with the ground truth. The description of each metric is provided below.

Precision:
Measures the proportion of correctly predicted positive pixels (true positives) out of all pixels predicted as positive. It answers the question: "Out of the pixels that were predicted to belong to a certain class, how many belong to that class?"

$$Precision = \frac{TP}{TP + FP}, \tag{2.3}$$

where, *TP* – true positives and *FP* – false positives.

Recall:
Measures the proportion of correctly predicted positive pixels out of all actual positive pixels in the ground truth. It answers the question: "Out of the pixels that belong to a certain class, how many were correctly predicted?"

$$Recall = \frac{TP}{TP + FN}, \tag{2.4}$$

where, *TP* – true positives and *FN* – false negatives.

F1 score:
Harmonic mean of precision and recall. It provides a balance between these two metrics, making it useful to account for both false positives and false negatives.

$$F1 = 2\frac{Precision * Recall}{Precision + Recall} \tag{2.5}$$

Jaccard Index (Intersection over union, IoU):

Quantifies the overlap between the predicted segmentation and the ground truth. It is defined as the size of the intersection divided by the size of the union of the predicted and ground truth sets.

$$IoU = \frac{|A \cap B|}{|A \cup B|}, \tag{2.6}$$

where *A* is the set of pixels in the predicted segmentation and *B* is the set of pixels in the ground truth label.

Accuracy:

Measures the proportion of correctly classified pixels (both true positives and true negatives) out of all pixels. It provides a general measure of how often the model is correct.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{2.7}$$

where, *TP* – true positives, *TN* – true negatives, *FP* – false positives, *FN* – false negatives.

Pointing game:

The Pointing game [28] is a qualitative metric often used to evaluate the interpretability of models. The Pointing game metric assesses whether the peak of the explainability heatmap falls within the ground truth annotated area. This is typically used in scenarios where the goal is to identify key regions rather than pixel-level accuracy.

$$Pointing\ Game\ Accuracy = \frac{X}{Y}, \tag{2.8}$$

where, *X* – the number of images where the peak of the explainability heatmap falls within the ground truth annotated area, and *Y* – the total number of images.

2.6. Conclusions

This section described the principles for creating the development environment. The design minimized boilerplate code and incorporated logging for improved experiment tracking. Modifications to CNN and ViT models were introduced, enabling them to output both class predictions and explainability map by embedding calculations into the model architecture. A dataset preprocessing methodology using a tiling approach was also introduced, preserving the original image resolution while ensuring a constant image size. Additionally, a strategy for evaluating the explainability output was presented.

3. Research results

The experiments were conducted as follows: hyperparameter tuning was performed for each model across all datasets. The experimental runs that achieved the highest test accuracy were then selected for further analysis. An explainability output segmentation test was conducted on these selected models, followed by a comparison of the results. The flow diagram of experiments is shown in Fig. 3.1.

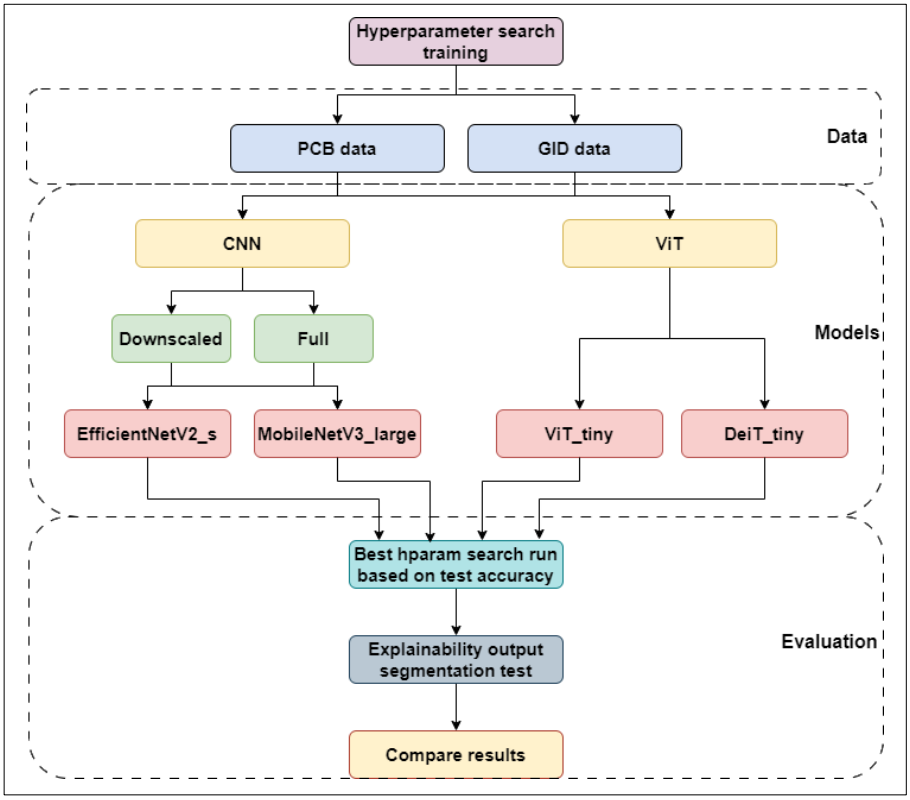


Fig. 3.1. Experiments flow diagram

3.1. CNN and ViT training results

For each dataset, hyperparameter search training was performed. Common hyperparameters for model training are given in Table 3.1. The Adam optimizer was utilized for Convolutional Neural Networks (CNNs) and AdamW for Vision Transformers (ViTs). The learning rate was managed by the ReduceLROnPlateau scheduler, which dynamically adjusts the rate based on the improvement of metrics during training. To enhance model accuracy and accelerate the training phase, pre-trained weights from ImageNet1K were utilized. The training was set to proceed for 20 epochs.

Table 3.1. Common hyperparameters for model training

Parameter	Model	
	CNN	**ViT**
Optimizer	Adam	AdamW
Learning rate (interval)	[0.0001; 0.01]	[0.00001; 0.0001]
Learning rate scheduler	ReduceLROnPlateau	
Pretrained weights	ImageNet1K	
Epochs	20	
Hyperparameter searching trials	10	
Batch size (choice)	[32; 64]	

Hyperparameter optimization was conducted using the TPE (Tree-structured Parzen Estimator) algorithm. During each trial, TPE fits a Gaussian Mixture Model (GMM) $l(x)$ to the set of parameter values associated with the best objective scores and another GMM $g(x)$ to the remaining parameter values. It then selects the parameter value x that maximizes the ratio $l(x)/g(x)$. Batch size and learning rate were selected as optimized hyperparameters. During 10 trial runs, optimal batch size and learning rate were selected based on the optimization objective: validation accuracy.

3.1.1. Training results of the PCB dataset

Table 3.2 shows the comparison of the models trained on the PCB dataset picking the best hyperparameter search trial run. From the table, it can be noticed that vision transformers require a much lower learning rate to converge. However, observing the batch size, it is hard to make any insights. The EfficientNetV2_s_d (downscaled) version achieved the best test accuracy even though the model has fewer parameters compared to EfficientNetV2_s.

Table 3.2. Comparison of the models trained on PCB dataset based on loss and accuracies.

Model	Learning rate	Batch size	Loss	Val accuracy, %	Test accuracy, %
EfficientNetV2_s	0.0028	64	0.079	98.41	98.49
EfficientNetV2_s_d	0.0020	32	**0.073**	**98.46**	**98.54**
MobileNetV3_large	0.0080	64	0.082	98.33	98.15
MobileNetV3_large_d	0.0028	64	0.087	98.10	98.12
DeiT_tiny_patch16_224	0.00008	64	0.088	98.13	98.26
ViT_tiny_patch16_224	0.00007	32	0.084	98.30	98.36

The effectiveness of hyperparameter searching can be seen from the parallel coordinates graph (Fig. 3.2). Particularly for EfficientNetV2_s_d (downscaled version), the test accuracy ranges from 97% to 98.5% depending on different hyperparameters.

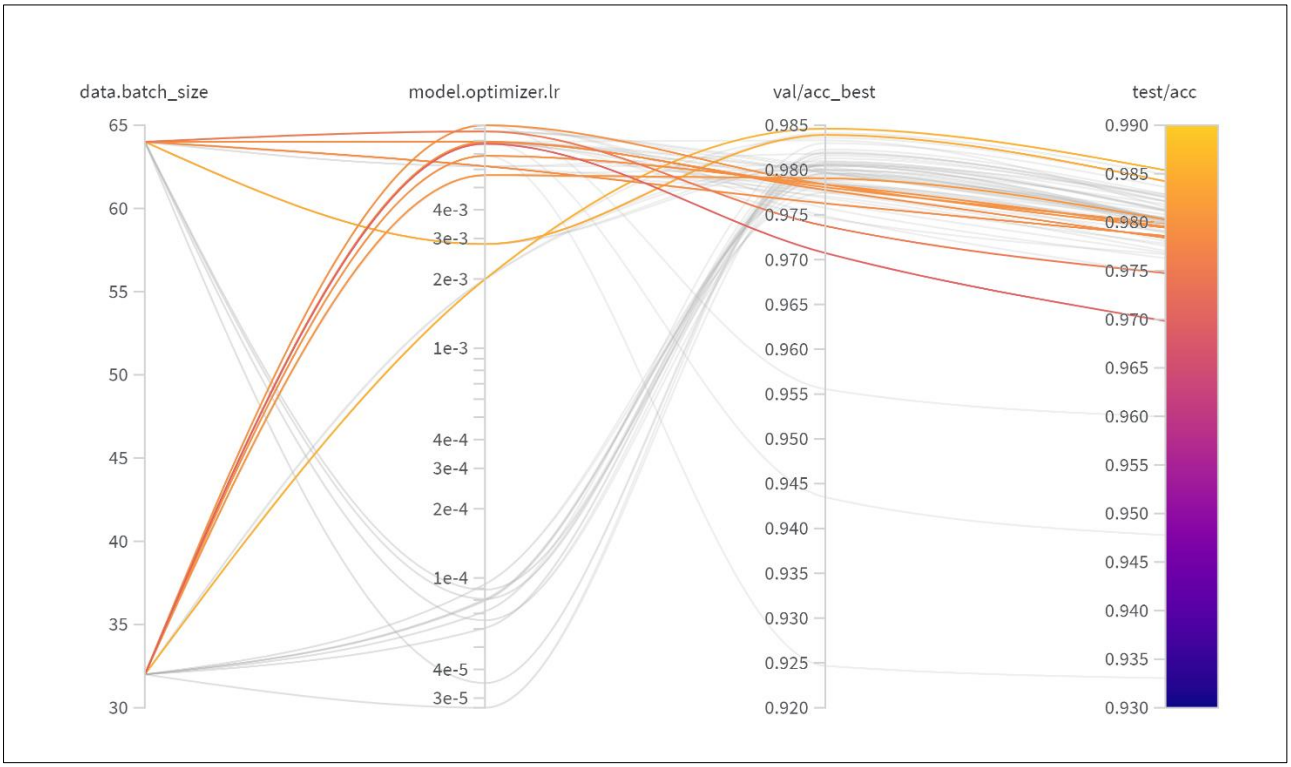


Fig. 3.2. Parallel coordinates graph showing how batch size and learning rate influences validation and test accuracy. 10 trial runs of EfficientNetV2_s_d (downscaled) version.

3.1.2. Training results of the GID dataset

Table 3.3 shows the comparison of the models trained on the GID dataset picking the best hyperparameter search trial run. Compared to the PCB dataset, lower batch sizes show better results. On this specific dataset, ViT_tiny_patch16_224 achieved the best test accuracy. The parallel coordinates graph for 10 trial runs of ViT_tiny_patch16_224 is given in Fig. 3.3. In this case, the test accuracy varies from 92.96% to 93.74%.

Table 3.3. Comparison of the models trained on GID dataset based on loss and accuracies.

Model	Learning rate	Batch size	Loss	Val accuracy, %	Test accuracy, %
EfficientNetV2_s	0.0020	32	0.170	94.06	93.34
EfficientNetV2_s_d	0.0020	32	0.169	93.97	93.56
MobileNetV3_large	0.0020	32	0.217	93.59	93.33
MobileNetV3_large_d	0.0028	64	0.172	93.59	93.18
DeiT_tiny_patch16_224	0.00009	32	0.177	94.04	93.49
ViT_tiny_patch16_224	0.00008	32	**0.165**	**94.20**	**93.74**

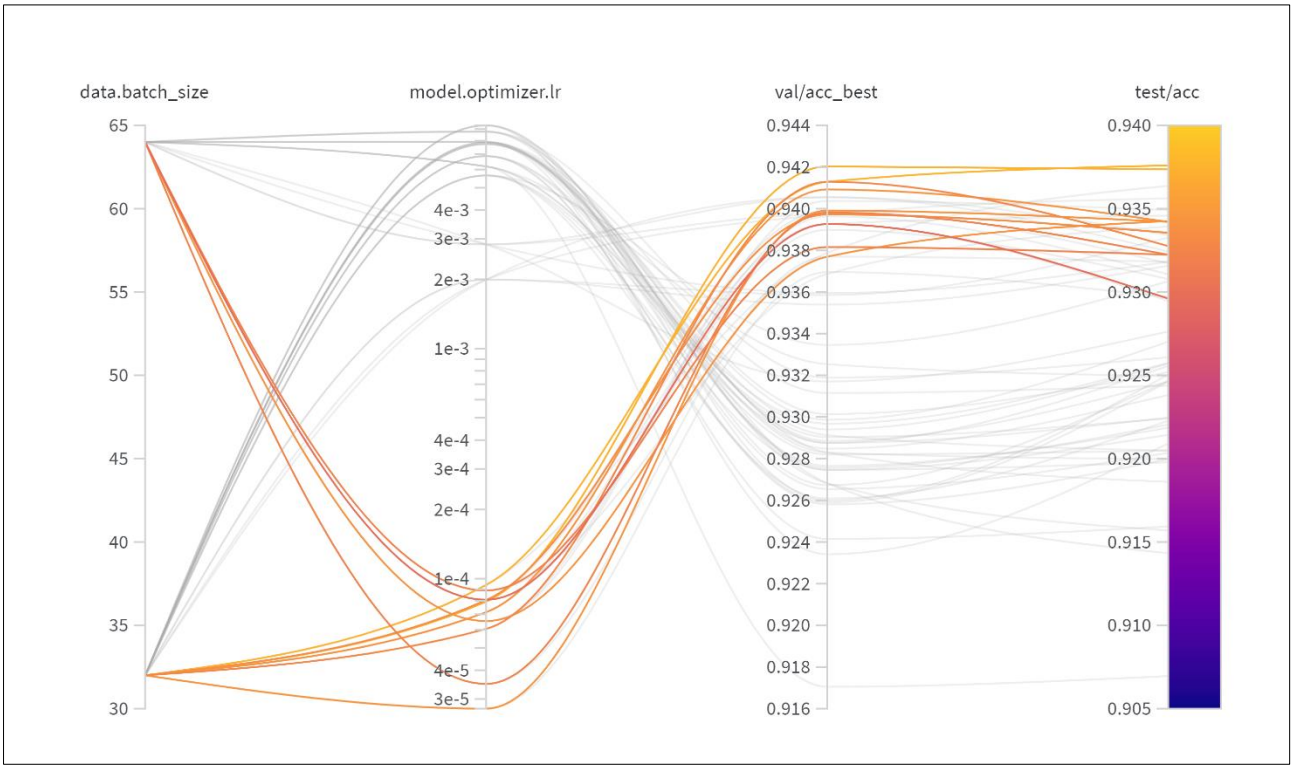


Fig. 3.3. Parallel coordinates graph showing how batch size and learning rate influences validation and test accuracy. 10 trial runs of ViT_tiny_patch16_224.

3.2. Explainability output segmentation evaluation

This chapter presents a comparative analysis of two explainability methods, CNN Class Activation Mapping (CAM) and Vision Transformer (ViT) Attention Rollout, in the context of image defect segmentation. Evaluating these techniques on two distinct datasets, PCB and GID, aims to determine their efficiency in generating meaningful explainability maps. The comparison is based on how well the generated maps align with the ground truth segmentation labels. The best-performing models from the previous section were picked for analysis. The models were ranked based on their performance across various segmentation metrics. The model with the lowest average rank was deemed the best for localizing defects using explainability output.

3.2.1. PCB dataset

The segmentation results obtained are compared across various models considering metrics described in section 2.5.2. The purpose of this comparison is to determine the effectiveness of full-scale versus downscaled CNN architectures and ViTs explainability outputs with different fusion modes and to identify the best-performing model with the most accurate interpretations for the decision-making process. The results are given in Table 3.4.

Full-scale vs. downscaled CNN architectures:

The downscaled EfficientNetV2_s_d model achieves higher Precision (0.8918) and Pointing Game score (0.823) compared to the full-scale EfficientNetV2_s model. This suggests that the downscaled architecture, which provides higher resolution explainability output, is more effective for defect segmentation. Similarly, MobileNetV3_large_d performs better in F1 score (0.6397 vs. 0.5214), Recall (0.6941 vs. 0.5518), and Jaccard Index (0.4703 vs. 0.3527), confirming the advantage of the downscaled model.

Vision Transformers with different attention fusion modes:

The comparison of DeiT (Data-efficient image Transformer) and ViT (Vision Transformer) models using different fusion modes (min, mean, and max) for attention heads reveals that the max fusion mode generally provides better performance across both model types. Among the evaluated models, the ViT_Tiny_patch16_224_fmax model stands out with the highest F1 score (0.5142), Jaccard Index (0.346), and Recall (0.3901).

Table 3.4. Explainability output segmentation metrics on PCB test set.

Model	F1	Precision	Jaccard Index	Accuracy	Recall	Pointing game	Classification accuracy
EfficientNetV2_s_d	0.5482	0.8918	0.3776	**0.9962**	0.3958	**0.823**	0.9853
EfficientNetV2_s	0.4652	0.7886	0.3031	0.9956	0.321	0.7867	0.9849
MobileNetV3_large_d	**0.6397**	0.5933	**0.4703**	0.9954	**0.6941**	0.7598	0.9812
MobileNetV3_large	0.5214	0.4943	0.3527	0.9941	0.5518	0.5701	0.9815
DeiT_tiny_patch16_224_fmin	0.3262	**0.9620**	0.1949	0.9953	0.1964	0.7877	0.9825
DeiT_tiny_patch16_224_fmean	0.43	0.8791	0.2739	0.9956	0.2846	0.7838	
DeiT_tiny_patch16_224_fmax	0.5113	0.5993	0.3434	0.995	0.4458	0.7239	
ViT_tiny_patch16_224_fmin	0.2726	0.886	0.1578	0.9949	0.1611	0.7912	0.9836
ViT_tiny_patch16_224_fmean	0.4041	0.8896	0.2532	0.9955	0.2613	0.7924	
ViT_tiny_patch16_224_fmax	0.5142	0.7543	0.346	0.9957	0.3901	0.7724	

Best performing model:

Based on Table 3.5 model ranking, the best performing CNN model is EfficientNetV2_s_d with an average rank of 1.67. ViT_tiny_patch16_224_fmax stands out as the best performing transformer model with an average rank of 2.5. Based on segmentation metrics, EfficientNetV2_s_d demonstrates improvements over ViT_tiny_patch16_224_fmax across all metrics. It enhances the F1 Score by 6.61%, Precision by 18.23%, Jaccard Index by 9.14%, Accuracy by 0.05%, Recall by 1.46%, and Pointing game by 6.55%. These results highlight that **EfficientNetV2_s_d** is the best performing model for defect localization on PCB dataset.

Table 3.5. CNN and ViT models ranking based on explainability output segmentation metrics (PCB dataset).

Model	F1	Precision	Jaccard Index	Accuracy	Recall	Pointing game	Average rank
EfficientNetV2_s_d	2	1	2	1	3	1	**1.67**
EfficientNetV2_s	4	2	4	2	4	2	3
MobileNetV3_large_d	1	3	1	3	1	3	2
MobileNetV3_large	3	4	3	4	2	4	3.3
DeiT_tiny_patch16_224_fmin	5	1	5	3	5	2	3.5
DeiT_tiny_patch16_224_fmean	3	2	3	2	3	4	2.83
DeiT_tiny_patch16_224_fmax	2	6	2	5	1	6	3.67
ViT_tiny_patch16_224_fmin	6	3	6	6	6	3	5
ViT_tiny_patch16_224_fmean	4	4	4	4	4	1	3.5
ViT_tiny_patch16_224_fmax	1	5	1	1	2	5	**2.5**

The heatmaps in Fig. 3.4 reveal that DeiT and ViT focus their attention more precisely on the defective areas, even when the labels encompass significant non-defective regions. In contrast, CNNs display more dispersed attention, occasionally missing finer details. Additionally, the figure suggests that the lower performance metrics for ViTs could be attributed to rough labeling, where the labels include a substantial amount of unnecessary background.

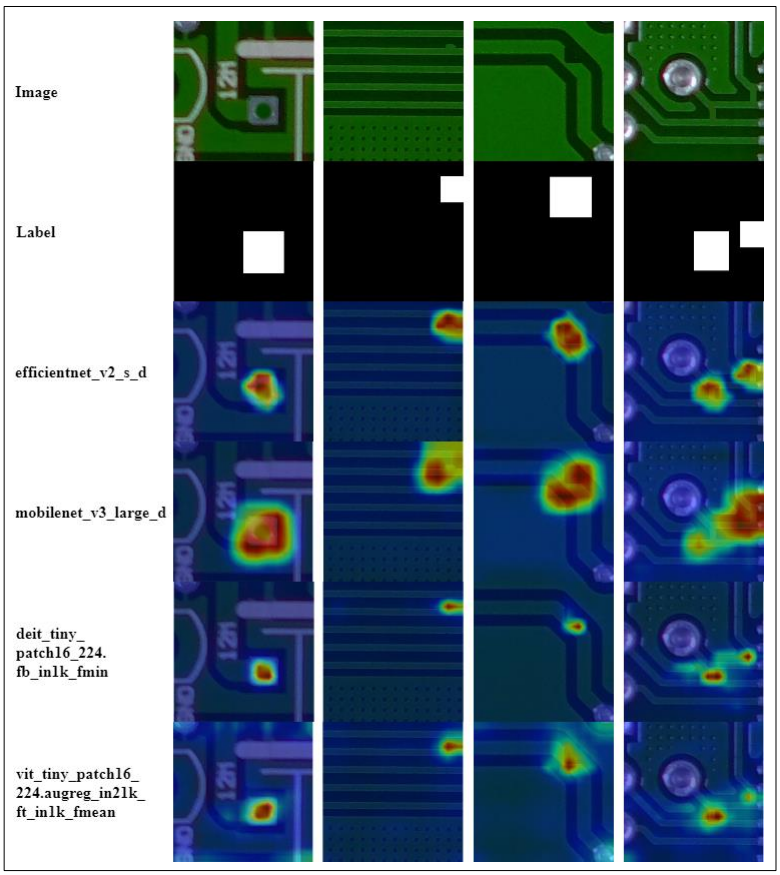


Fig. 3.4. Comparison of explainability outputs (PCB dataset)

3.2.2. GID dataset

Similarly, segmentation metrics for different models trained with the GID dataset are compared. The results are given in Table 3.6.

Full-Scale vs. Downscaled CNN Architectures:

The downscaled EfficientNetV2_s_d model demonstrates the highest Precision (0.5144) and Pointing Game score (0.5827) when compared to its full-scale counterpart. MobileNetV3_large achieves better results in F1 score (0.254 vs. 0.2003), Jaccard Index (0.1455 vs. 0.1113), and Recall (0.2183 vs. 0.1338) compared to the downscaled version.

Vision Transformers with different attention fusion modes:

The comparison of DeiT and ViT models employing different fusion modes for attention heads yields varied results. The DeiT_tiny_fmin model excels in Precision (0.3811) and Pointing Game score (0.4603). Meanwhile, among the ViT models, the ViT_tiny_patch16_224_fmean model stands out with the highest F1 score (0.2223).

Table 3.6. Explainability output segmentation metrics on GID test set.

Model	F1	Precision	Jaccard Index	Accuracy	Recall	Pointing game	Classification accuracy
EfficientNetV2_s_d	0.1364	**0.5144**	0.0732	**0.9812**	0.0787	**0.5827**	0.9356
EfficientNetV2_s	0.2171	0.3103	0.1218	0.9773	0.1669	0.3411	0.9334
MobileNetV3_large_d	0.2003	0.398	0.1113	0.9798	0.1338	0.5419	0.9318
MobileNetV3_large	**0.2540**	0.3038	**0.1455**	0.9758	0.2183	0.5228	0.9332
DeiT_tiny_patch16_224_fmin	0.2123	0.3811	0.1188	0.9794	0.1472	0.4603	0.9349
DeiT_tiny_patch16_224_fmean	0.1749	0.1889	0.0958	0.971	0.1628	0.2516	
DeiT_tiny_patch16_224_fmax	0.1333	0.1354	0.0715	0.9678	0.1314	0.1888	
ViT_tiny_patch16_224_fmin	0.1264	0.3539	0.0675	0.9799	0.0769	0.4524	0.9373
ViT_tiny_patch16_224_fmean	0.2223	0.2784	0.1251	0.9756	0.1851	0.4043	
ViT_tiny_patch16_224_fmax	0.2035	0.1865	0.1133	0.9669	**0.2239**	0.2995	

Best performing model:

Based on Table 3.7 model ranking, the best performing CNN model is MobileNetV3_large with an average rank of 2.33. DeiT_tiny_patch16_224_fmin stand out as the best performing transformer model with an average rank of 2. Based on segmentation metrics, MobileNetV3_large shows notable improvements over DeiT_tiny_patch16_fmin in several metrics. It enhances the F1 Score by 19.63%, Jaccard Index by 22.48%, Recall by 48.33%, and the Pointing game by 13.57%. These results highlight that **MobileNetV3_large** is the best performing model for defect localization on GID dataset.

Table 3.7. CNN and ViT models ranking based on explainability output segmentation metrics (GID dataset).

Model	F1	Precision	Jaccard Index	Accuracy	Recall	Pointing game	Average rank
EfficientNetV2_s_d	4	1	4	1	4	1	2.5
EfficientNetV2_s	2	3	2	3	2	4	2.67
MobileNetV3_large_d	3	2	3	2	3	2	2.5
MobileNetV3_large	1	4	1	4	1	3	**2.33**
DeiT_tiny_patch16_224_fmin	2	1	2	2	4	1	**2**
DeiT_tiny_patch16_224_fmean	4	4	4	4	3	5	4
DeiT_tiny_patch16_224_fmax	5	6	5	6	5	6	5.5
ViT_tiny_patch16_224_fmin	6	2	6	1	6	2	3.83
ViT_tiny_patch16_224_fmean	1	3	1	3	2	3	2.17
ViT_tiny_patch16_224_fmax	3	5	3	5	1	4	3.5

The heatmaps in Fig. 3.5 shows the differences in explainability output among different models.

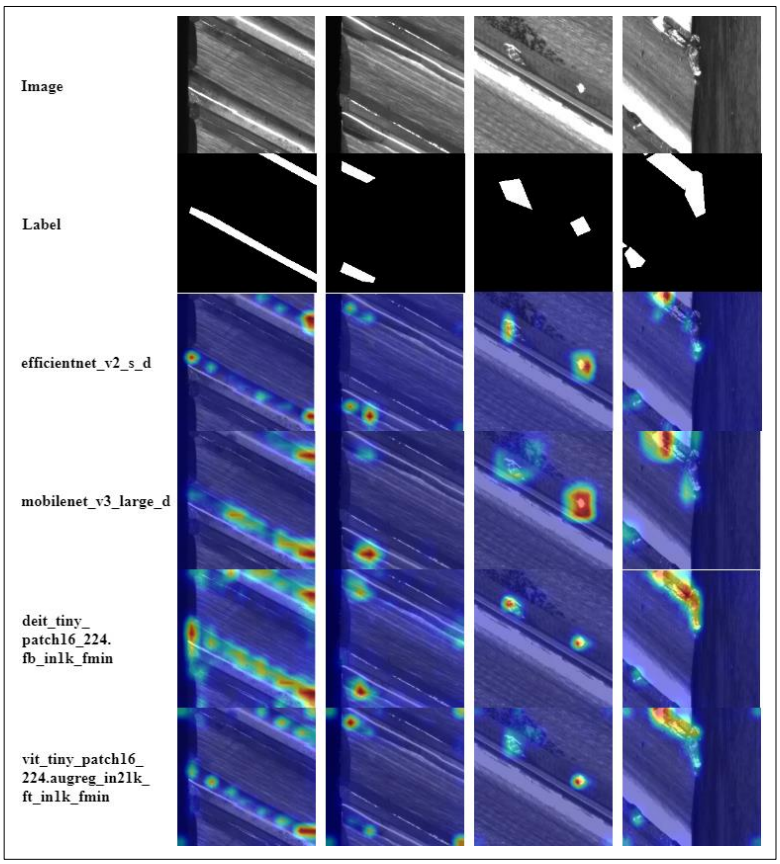


Fig. 3.5. Comparison of explainability outputs (GID dataset)

3.3. The effect of CNN downscaling on CAM output

Taking one feature extraction stage from CNNs (MobileNetV3 and EfficientNetV2) increases the explainability output resolution by two times, from 7x7 pixels to 14x14 pixels. This output is then upscaled to the original input resolution. Figures Fig. 3.6 and Fig. 3.8 showcase the comparison of segmentation metrics between the full and downscaled architectures. A substantial increase in all

metrics with the downscaled version can be observed for the PCB dataset (Fig. 3.6). For the GID dataset (Fig. 3.8), the downscaled versions perform better in terms of precision and the pointing game metric. Fig. 3.7 and Fig. 3.9 illustrate the effect of downscaling on the CAM output.

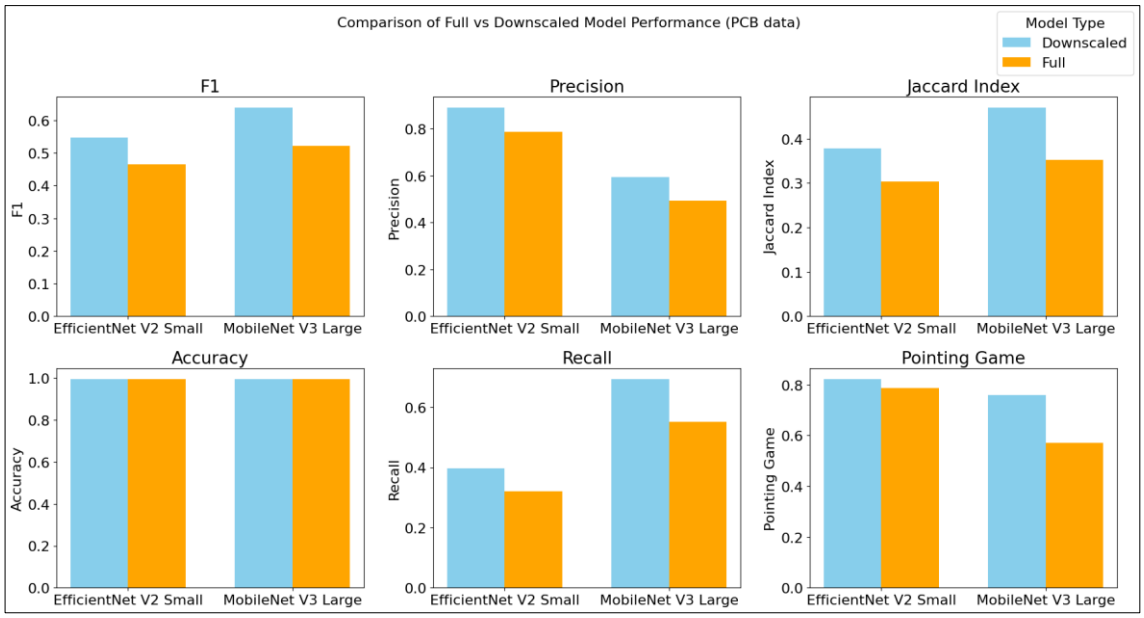


Fig. 3.6. Comparison of full vs downscaled model performance (PCB dataset)

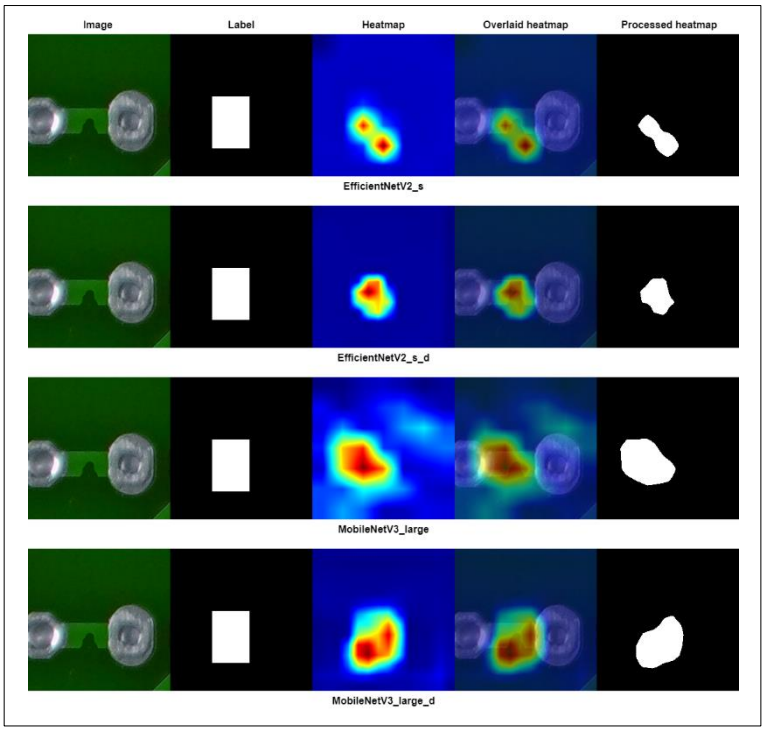


Fig. 3.7. CNN CAM comparison considering full and downscale architectures (PCB dataset)

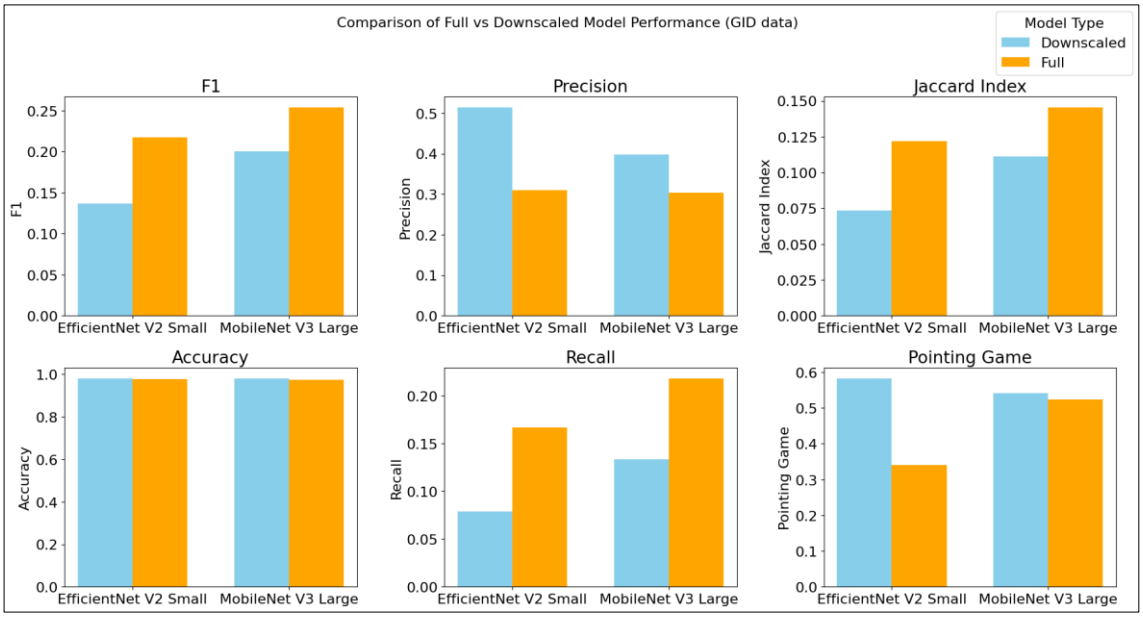


Fig. 3.8. Comparison of full vs downscaled model performance (GID dataset)

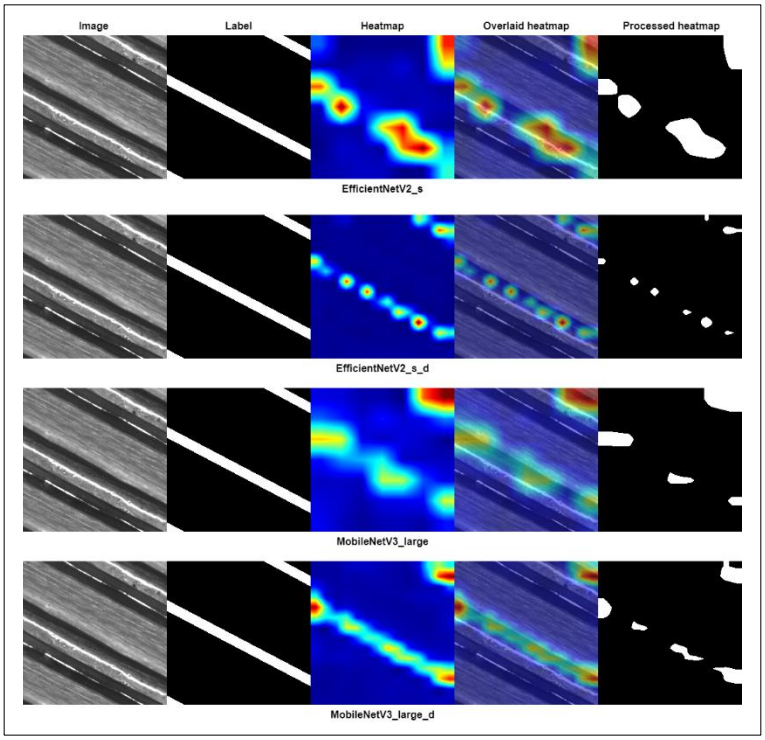


Fig. 3.9. CNN CAM comparison considering full and downscale architectures (GID dataset)

3.4. The effect of ViT head fusion and discard ratio

Analyzing the results from the heatmaps (Fig. 3.10) generated for the different performance metrics across fusion modes and discard ratios, some insights could be observed.

Impact of fusion modes:

Min fusion mode: this mode selects the minimum attention values across different heads. Lower performance metrics in this mode suggest that the least attended areas are not always crucial for

accurate explainability. This could indicate that the areas where attention is minimally focused might not be essential for understanding the model's decisions or might be missing key features.

Mean fusion mode: averages attention across heads, potentially smoothing out the specific peaks of attention that highlight critical features. Moderate performance in this mode suggests that while averaging provides a more comprehensive view than the minimum, it might dilute the impact of highly attended areas that are more relevant for pinpointing the model's focus and reasoning.

Max fusion mode: selecting the maximum attention values across heads for each region emphasizes the areas where the model pays the most attention. The high performance observed here underscores the importance of these highly focused areas in creating effective explainability outputs. The 'max' mode likely captures the most influential features or decision points in the input, aligning well with the objectives of explainability to reveal what the model considers most important.

Impact of discard ratio:

Generally, improves performance in the 'max' fusion mode, suggesting that eliminating the least attended regions helps to clarify and enhance the visibility of crucial decision-making areas. This supports a focus on quality (high attention) over quantity (all attention values). In the min and mean modes, the impact is mixed, indicating that in these modes, some of the discarded low-attention areas might still hold value for a fuller understanding or might contribute necessary context that supports the overall interpretability.

Model comparison (DeiT vs VIT):

ViT often performs better, especially in the max mode. DeiT shows less sensitivity to changes in discard ratios, possibly indicating a more uniform distribution of attention across heads.

Metric-specific observations:

Precision and Jaccard Index: these metrics improve notably in the max mode with higher discard ratios, reinforcing the value of focusing on areas with the highest attention as these areas are most relevant to the model’s decision-making process.

Recall and Pointing Game: the variability in these metrics suggests a trade-off between achieving comprehensive coverage (recall) and focusing on the most critical areas (precision). High discard ratios in 'max' mode might miss some relevant but less attended features, impacting recall and pointing accuracy.

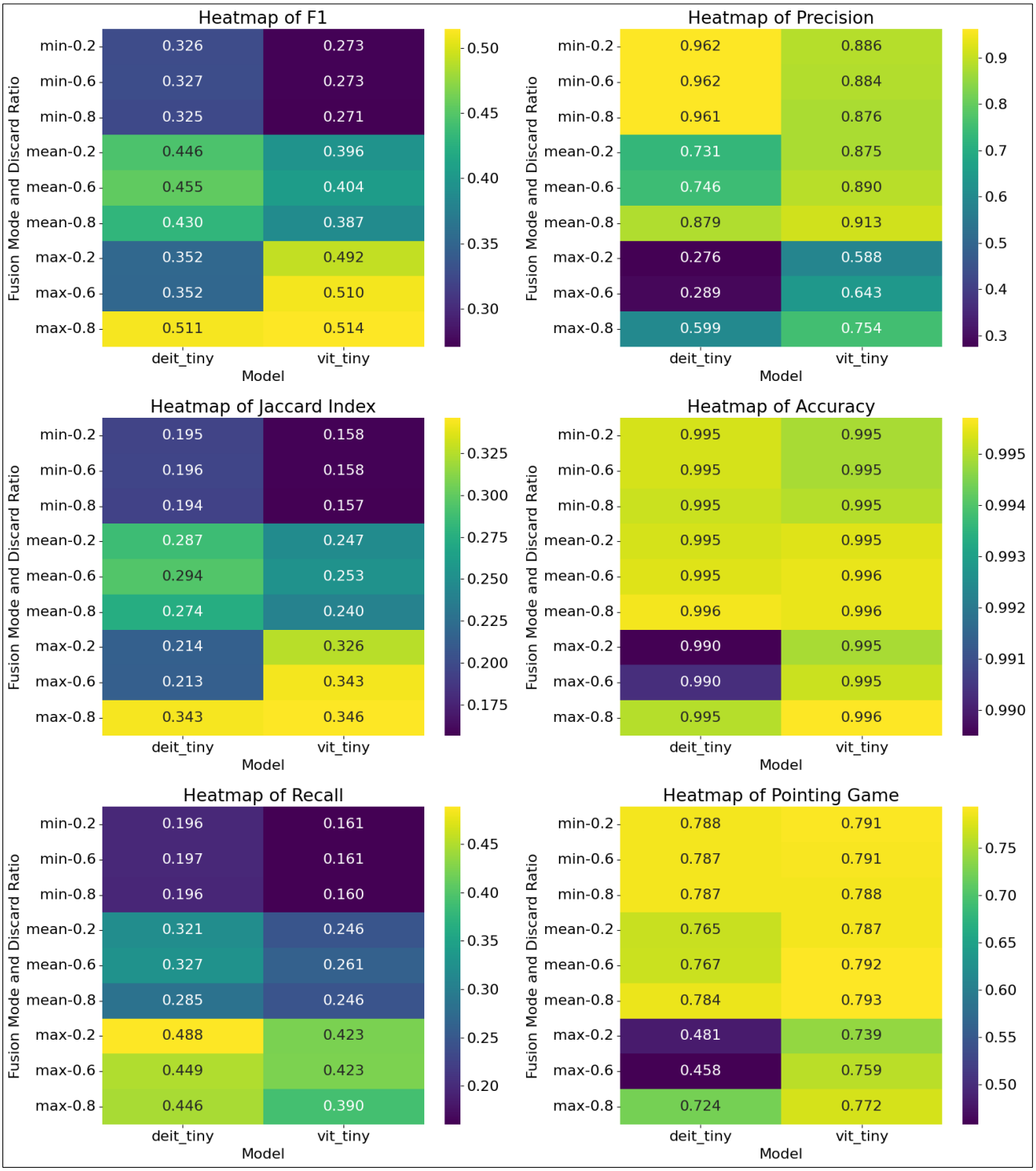


Fig. 3.10. Performance metrics across fusion modes and discard ratios for vision transformer models (PCB dataset)

The visual outputs from the DeiT model (Fig. 3.11) demonstrate the impact of different fusion modes (min, mean, max) with a 0.8 discard ratio on the model's attention focus and segmentation results. The min mode displays a focused and narrow heatmap, indicating a conservative attention strategy that precisely targets key areas. The mean mode provides a similar attention spread. Conversely, the max mode shows the most extensive heatmap coverage, highlighting multiple potential areas. These differences illustrate how each fusion mode affects the model's interpretative focus and can guide the selection of fusion strategies for specific applications requiring varying levels of detail and coverage.

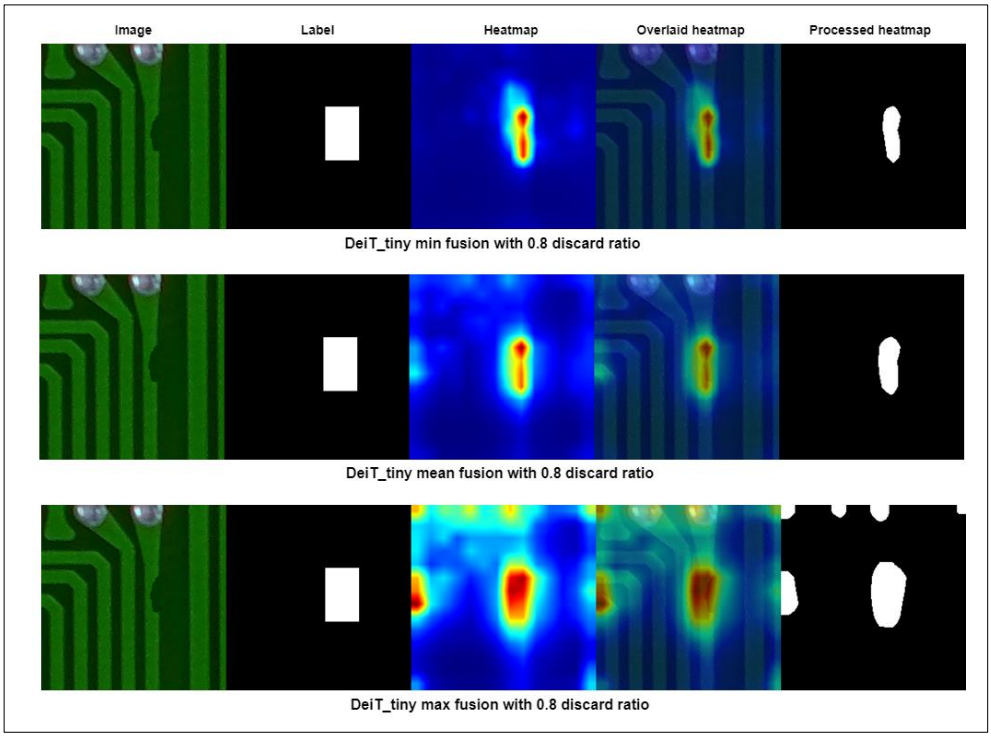


Fig. 3.11. Comparison of DeiT_tiny model explainability output with different fusion modes and 0.8 discard ratio

Fig. 3.12 displays segmentation results from a DeiT model using a mean fusion mode with varying discard ratios (0.2, 0.6, 0.8). As the discard ratio increases, the focus of the model's explainability map narrows, highlighting areas that receive consistently higher attention across the heads. Initially, with a low discard ratio of 0.2, the heatmap is more dispersed, showing broad areas of attention.

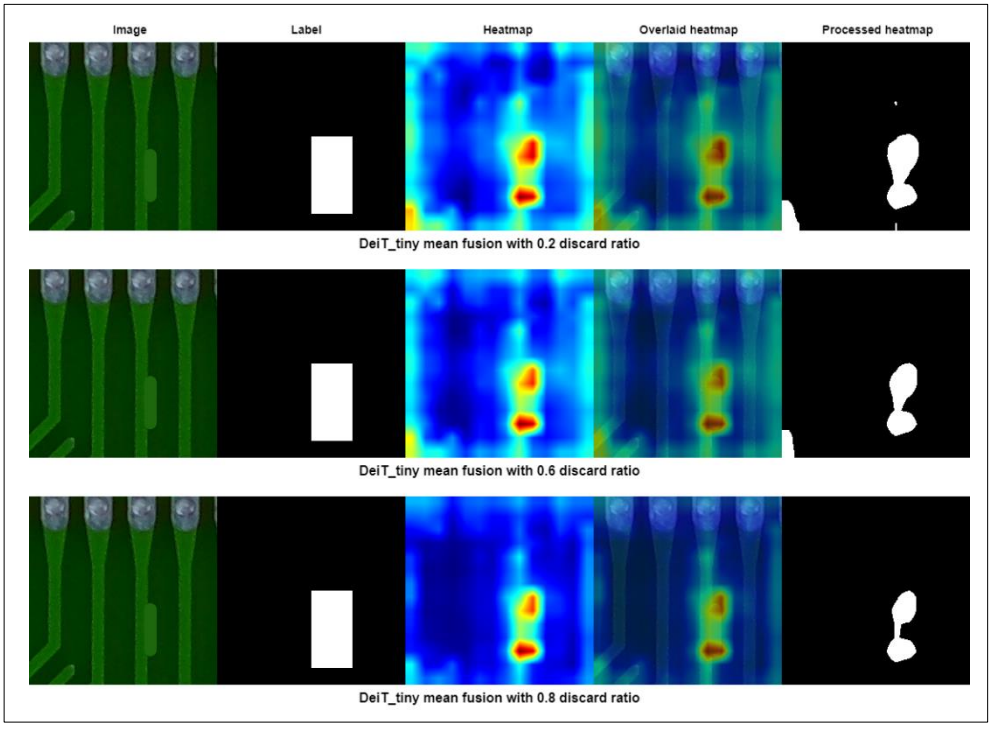


Fig. 3.12. Comparison of DeiT_tiny model explainability output with different discard ratios and mean fusion mode

3.5. Conclusions

The main observations from the study:

- The EfficientNetV2_s_d (downscaled) model achieved the highest test accuracy of 98.54% on the PCB dataset. Hyperparameter tuning improved the test accuracy from 97% to 98.5%.
- The ViT_tiny_patch16_224 model achieved the highest test accuracy of 93.74% on the GID dataset. Hyperparameter tuning improved the test accuracy from 92.96% to 93.74%.
- Vision transformer models require a significantly lower learning rate to converge. The optimal learning rate for the EfficientNetV2_s_d model is 0.002, while for the ViT_tiny_patch16_224 model, it is 0.00008.
- Regarding explainability and output segmentation metrics, in 3 out of 4 cases, downscaled CNN models with higher output resolution performed better. An average increase of 32% in precision and 27% in the pointing game metric was observed.
- Introducing fusion modes and discard ratios in the explainability maps of vision transformers enhances their versatility and adaptability for specific tasks. Max fusion achieves better F1 score, Jaccard index, and recall, while min and mean fusions perform better in terms of precision, accuracy, and pointing game metrics. As the discard ratio increases, the model's explainability map narrows its focus, highlighting areas that consistently receive higher attention.
- Considering different architectures, CNNs explainability output is more precise. On PCB dataset EfficientNetV2_s_d outperformed ViT_tiny_patch16_224_fmax, improving the F1 Score by 6.61%, Precision by 18.23%, Jaccard Index by 9.14%, Recall by 1.46%, and Pointing game metric by 6.55%. On GID dataset MobileNetV3_large outperformed DeiT_tiny_patch16_fmin, improving the F1 Score by 19.63%, Jaccard Index by 22.48%, Recall by 48.33%, and the Pointing Game metric by 13.57%.

Conclusions

1. Literature review of efficient and lightweight CNN and ViT models was conducted. For further analysis, EfficientNetV2_s and MobileNetV3_large were chosen as convolutional neural networks (CNNs), while ViT_tiny and DeiT_tiny were selected as vision transformers (ViTs). Different architectures explainability methods were reviewed and their applications in the industry demonstrated the interest of explainability methods utilization for defect localization, validating the relevance of this research.
2. The EfficientNetV2_s_d (downscaled) model achieved the highest test accuracy of 98.54% on the PCB dataset, with hyperparameter tuning increasing the accuracy from 97% to 98.54%. Similarly, the ViT_tiny_patch16_224 model achieved the highest test accuracy of 93.74% on the GID dataset, with hyperparameter tuning enhancing the accuracy from 92.96% to 93.74%.
3. Modifications to CNN and ViT models were introduced. CAM and attention rollout techniques were optimized to generate outputs (class and explainability map) in a single forward pass without significantly increasing model complexity, requiring only a few additional matrix multiplications.
4. Considering different architectures, CNNs stand out as better explainable models. On PCB dataset EfficientNetV2_s_d outperformed ViT_tiny_patch16_224_fmax, improving the F1 Score by 6.61%, Precision by 18.23%, Jaccard Index by 9.14%, Recall by 1.46%, and Pointing Game metric by 6.55%. On GID dataset MobileNetV3_large outperformed DeiT_tiny_patch16_fmin, improving the F1 Score by 19.63%, Jaccard Index by 22.48%, Recall by 48.33%, and the Pointing Game metric by 13.57%.

List of references

1. GAO, Yiping, LI, Xinyu, WANG, Xi Vincent, WANG, Lihui and GAO, Liang. A Review on Recent Advances in Vision-based Defect Recognition towards Industrial Intelligence. *Journal of Manufacturing Systems*. January 2022. Vol. 62, p. 753–766. DOI 10.1016/j.jmsy.2021.05.008.

2. YANG, Jing, LI, Shaobo, WANG, Zheng, DONG, Hao, WANG, Jun and TANG, Shihao. Using Deep Learning to Detect Defects in Manufacturing: A Comprehensive Survey and Current Challenges. *Materials*. 16 December 2020. Vol. 13, no. 24, p. 5755. DOI 10.3390/ma13245755.

3. WANG, Yu Emma, WEI, Gu-Yeon and BROOKS, David. *Benchmarking TPU, GPU, and CPU Platforms for Deep Learning*. Online. 22 October 2019. arXiv. arXiv:1907.10701. [Accessed 11 May 2024]. Available from: http://arxiv.org/abs/1907.10701arXiv:1907.10701 [cs, stat]

4. CHEN, Yajun, DING, Yuanyuan, ZHAO, Fan, ZHANG, Erhu, WU, Zhangnan and SHAO, Linhao. Surface Defect Detection Methods for Industrial Products: A Review. *Applied Sciences*. 20 August 2021. Vol. 11, no. 16, p. 7657. DOI 10.3390/app11167657.

5. DENG, Jia, DONG, Wei, SOCHER, Richard, LI, Li-Jia, LI, Kai and FEI-FEI, Li. ImageNet: A Large-Scale Hierarchical Image Database. .

6. TAN, Mingxing and LE, Quoc. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In : *Proceedings of the 36th International Conference on Machine Learning*. Online. PMLR, 24 May 2019. p. 6105–6114. [Accessed 28 March 2024]. Available from: https://proceedings.mlr.press/v97/tan19a.html

7. TAN, Mingxing and LE, Quoc. EfficientNetV2: Smaller Models and Faster Training. In : *Proceedings of the 38th International Conference on Machine Learning*. Online. PMLR, 1 July 2021. p. 10096–10106. [Accessed 28 March 2024]. Available from: https://proceedings.mlr.press/v139/tan21a.html

8. REN, Pengzhen, XIAO, Yun, CHANG, Xiaojun, HUANG, Po-yao, LI, Zhihui, CHEN, Xiaojiang and WANG, Xin. A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions. *ACM Computing Surveys*. 31 May 2022. Vol. 54, no. 4, p. 1–34. DOI 10.1145/3447582.

9. HOWARD, Andrew G., ZHU, Menglong, CHEN, Bo, KALENICHENKO, Dmitry, WANG, Weijun, WEYAND, Tobias, ANDREETTO, Marco and ADAM, Hartwig. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. Online. 16 April 2017. arXiv. arXiv:1704.04861. [Accessed 3 April 2024]. Available from: http://arxiv.org/abs/1704.04861arXiv:1704.04861 [cs]

10. SANDLER, Mark, HOWARD, Andrew, ZHU, Menglong, ZHMOGINOV, Andrey and CHEN, Liang-Chieh. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. Online. 21 March 2019. arXiv. arXiv:1801.04381. [Accessed 3 April 2024]. Available from: http://arxiv.org/abs/1801.04381arXiv:1801.04381 [cs]

11. HOWARD, Andrew, SANDLER, Mark, CHU, Grace, CHEN, Liang-Chieh, CHEN, Bo, TAN, Mingxing, WANG, Weijun, ZHU, Yukun, PANG, Ruoming, VASUDEVAN, Vijay, LE, Quoc V. and ADAM, Hartwig. *Searching for MobileNetV3*. Online. 20 November 2019. arXiv. arXiv:1905.02244. [Accessed 3 April 2024]. Available from: http://arxiv.org/abs/1905.02244arXiv:1905.02244 [cs]

12. VASWANI, Ashish, SHAZEER, Noam, PARMAR, Niki, USZKOREIT, Jakob, JONES, Llion, GOMEZ, Aidan N, KAISER, Ł ukasz and POLOSUKHIN, Illia. Attention is All you Need. In : *Advances in Neural Information Processing Systems*. Online. Curran Associates, Inc., 2017. [Accessed 16 April 2024]. Available from: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

13. KHURANA, Diksha, KOLI, Aditya, KHATTER, Kiran and SINGH, Sukhdev. Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*. January 2023. Vol. 82, no. 3, p. 3713–3744. DOI 10.1007/s11042-022-13428-4.

14. RAGHU, Maithra, UNTERTHINER, Thomas, KORNBLITH, Simon, ZHANG, Chiyuan and DOSOVITSKIY, Alexey. Do Vision Transformers See Like Convolutional Neural Networks? In : *Advances in Neural Information Processing Systems*. Online. Curran Associates, Inc., 2021. p. 12116–12128. [Accessed 11 May 2024]. Available from: https://proceedings.neurips.cc/paper_files/paper/2021/hash/652cf38361a209088302ba2b8b7f51e0-Abstract.html

15. DOSOVITSKIY, Alexey, BEYER, Lucas, KOLESNIKOV, Alexander, WEISSENBORN, Dirk, ZHAI, Xiaohua, UNTERTHINER, Thomas, DEHGHANI, Mostafa, MINDERER, Matthias, HEIGOLD, Georg, GELLY, Sylvain, USZKOREIT, Jakob and HOULSBY, Neil. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. Online. 3 June 2021. arXiv. arXiv:2010.11929. [Accessed 6 April 2024]. Available from: http://arxiv.org/abs/2010.11929arXiv:2010.11929 [cs]

16. STEINER, Andreas, KOLESNIKOV, Alexander, ZHAI, Xiaohua, WIGHTMAN, Ross, USZKOREIT, Jakob and BEYER, Lucas. *How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers*. Online. 23 June 2022. arXiv. arXiv:2106.10270. [Accessed 11 May 2024]. Available from: http://arxiv.org/abs/2106.10270arXiv:2106.10270 [cs]

17. PARK, Namuk and KIM, Songkuk. *How Do Vision Transformers Work?*. Online. 8 June 2022. arXiv. arXiv:2202.06709. [Accessed 11 April 2024]. Available from: http://arxiv.org/abs/2202.06709arXiv:2202.06709 [cs]

18. TOUVRON, Hugo, CORD, Matthieu, DOUZE, Matthijs, MASSA, Francisco, SABLAYROLLES, Alexandre and JÉGOU, Hervé. *Training data-efficient image transformers & distillation through attention*. Online. 15 January 2021. arXiv. arXiv:2012.12877. [Accessed 18 April 2024]. Available from: http://arxiv.org/abs/2012.12877arXiv:2012.12877 [cs]

19. GOU, Jianping, YU, Baosheng, MAYBANK, Stephen J. and TAO, Dacheng. Knowledge Distillation: A Survey. *International Journal of Computer Vision*. June 2021. Vol. 129, no. 6, p. 1789–1819. DOI 10.1007/s11263-021-01453-z.

20. SHAMSHIRBAND, Shahab, FATHI, Mahdis, DEHZANGI, Abdollah, CHRONOPOULOS, Anthony Theodore and ALINEJAD-ROKNY, Hamid. A review on deep learning approaches in healthcare systems: Taxonomies, challenges, and open issues. *Journal of Biomedical Informatics*. January 2021. Vol. 113, p. 103627. DOI 10.1016/j.jbi.2020.103627.

21. HUANG, Jian, CHAI, Junyi and CHO, Stella. Deep learning in finance and banking: A literature review and classification. *Frontiers of Business Research in China*. December 2020. Vol. 14, no. 1, p. 13. DOI 10.1186/s11782-020-00082-6.

22. MOZAFFARI, Sajjad, AL-JARRAH, Omar Y., DIANATI, Mehrdad, JENNINGS, Paul and MOUZAKITIS, Alexandros. Deep Learning-Based Vehicle Behavior Prediction for Autonomous

Driving Applications: A Review. *IEEE Transactions on Intelligent Transportation Systems*. January 2022. Vol. 23, no. 1, p. 33–47. DOI 10.1109/TITS.2020.3012034.

23. SHAH, Varun and KONDA, Sreedhar Reddy. Neural Networks and Explainable AI: Bridging the Gap between Models and Interpretability. *INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY*. 30 June 2021. Vol. 5, no. 2, p. 163–176. DOI https://n2t.net/ark:/70114/ijcst.v5i2.387.

24. LINARDATOS, Pantelis, PAPASTEFANOPOULOS, Vasilis and KOTSIANTIS, Sotiris. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy*. 25 December 2020. Vol. 23, no. 1, p. 18. DOI 10.3390/e23010018.

25. ALZUBAIDI, Laith, ZHANG, Jinglan, HUMAIDI, Amjad J., AL-DUJAILI, Ayad, DUAN, Ye, AL-SHAMMA, Omran, SANTAMARÍA, J., FADHEL, Mohammed A., AL-AMIDIE, Muthana and FARHAN, Laith. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*. 31 March 2021. Vol. 8, no. 1, p. 53. DOI 10.1186/s40537-021-00444-8.

26. ZHOU, Bolei, KHOSLA, Aditya, LAPEDRIZA, Agata, OLIVA, Aude and TORRALBA, Antonio. Learning Deep Features for Discriminative Localization. In : *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Online. Las Vegas, NV, USA : IEEE, June 2016. p. 2921–2929. [Accessed 2 April 2024]. ISBN 978-1-4673-8851-1. DOI 10.1109/CVPR.2016.319.

27. ABNAR, Samira and ZUIDEMA, Willem. *Quantifying Attention Flow in Transformers*. Online. 31 May 2020. arXiv. arXiv:2005.00928. [Accessed 19 April 2024]. Available from: http://arxiv.org/abs/2005.00928arXiv:2005.00928 [cs]

28. ZHANG, Jianming, BARGAL, Sarah Adel, LIN, Zhe, BRANDT, Jonathan, SHEN, Xiaohui and SCLAROFF, Stan. Top-Down Neural Attention by Excitation Backprop. *International Journal of Computer Vision*. October 2018. Vol. 126, no. 10, p. 1084–1102. DOI 10.1007/s11263-017-1059-x.

29. RAATIKAINEN, Lassi and RAHTU, Esa. *The Weighting Game: Evaluating Quality of Explainability Methods*. Online. 12 August 2022. arXiv. arXiv:2208.06175. [Accessed 9 April 2024]. Available from: http://arxiv.org/abs/2208.06175arXiv:2208.06175 [cs]

30. CZIMMERMANN, Tamás, CIUTI, Gastone, MILAZZO, Mario, CHIURAZZI, Marcello, ROCCELLA, Stefano, ODDO, Calogero Maria and DARIO, Paolo. Visual-Based Defect Detection and Classification Approaches for Industrial Applications—A SURVEY. *Sensors*. 6 March 2020. Vol. 20, no. 5, p. 1459. DOI 10.3390/s20051459.

31. WEI, Linyu, CAI, Jueping, WEN, Kailin and ZHANG, Chengkai. Local–global lightweight ViT model for mini/micro-LED-chip defect recognition. *Engineering Applications of Artificial Intelligence*. August 2023. Vol. 123, p. 106247. DOI 10.1016/j.engappai.2023.106247.

32. TROCKMAN, Asher and KOLTER, J. Zico. *Patches Are All You Need?*. Online. 24 January 2022. arXiv. arXiv:2201.09792. [Accessed 22 April 2024]. Available from: http://arxiv.org/abs/2201.09792arXiv:2201.09792 [cs]

33. SELVARAJU, Ramprasaath R., COGSWELL, Michael, DAS, Abhishek, VEDANTAM, Ramakrishna, PARIKH, Devi and BATRA, Dhruv. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *International Journal of Computer Vision*. February 2020. Vol. 128, no. 2, p. 336–359. DOI 10.1007/s11263-019-01228-7. arXiv:1610.02391 [cs]

34. MEHTA, Sachin and RASTEGARI, Mohammad. *MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer*. Online. 4 March 2022. arXiv. arXiv:2110.02178. [Accessed 22 April 2024]. Available from: http://arxiv.org/abs/2110.02178arXiv:2110.02178 [cs]

35. LIU, Tianyuan, ZHENG, Hangbin, BAO, Jinsong, ZHENG, Pai, WANG, Junliang, YANG, Changqi and GU, Jun. An Explainable Laser Welding Defect Recognition Method Based on Multi-Scale Class Activation Mapping. *IEEE Transactions on Instrumentation and Measurement*. 2022. Vol. 71, p. 1–12. DOI 10.1109/TIM.2022.3148739.

36. ZHOU, Fei, LIU, Guihua, XU, Feng and DENG, Hao. A Generic Automated Surface Defect Detection Based on a Bilinear Model. *Applied Sciences*. 3 August 2019. Vol. 9, no. 15, p. 3159. DOI 10.3390/app9153159.

37. SIMONYAN, Karen and ZISSERMAN, Andrew. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Online. 10 April 2015. arXiv. arXiv:1409.1556. [Accessed 23 April 2024]. Available from: http://arxiv.org/abs/1409.1556arXiv:1409.1556 [cs]

38. LEE, Minyoung, JEON, Joohyoung and LEE, Hongchul. Explainable AI for domain experts: a post Hoc analysis of deep learning for defect classification of TFT–LCD panels. *Journal of Intelligent Manufacturing*. August 2022. Vol. 33, no. 6, p. 1747–1759. DOI 10.1007/s10845-021-01758-3.

39. BINDER, Alexander, MONTAVON, Grégoire, BACH, Sebastian, MÜLLER, Klaus-Robert and SAMEK, Wojciech. *Layer-wise Relevance Propagation for Neural Networks with Local Renormalization Layers*. Online. 4 April 2016. arXiv. arXiv:1604.00825. [Accessed 11 May 2024]. Available from: http://arxiv.org/abs/1604.00825arXiv:1604.00825 [cs]

40. HUANG, Weibo and WEI, Peng. *A PCB Dataset for Defects Detection and Classification*. Online. 23 January 2019. arXiv. arXiv:1901.08204. [Accessed 6 April 2024]. Available from: http://arxiv.org/abs/1901.08204arXiv:1901.08204 [cs]