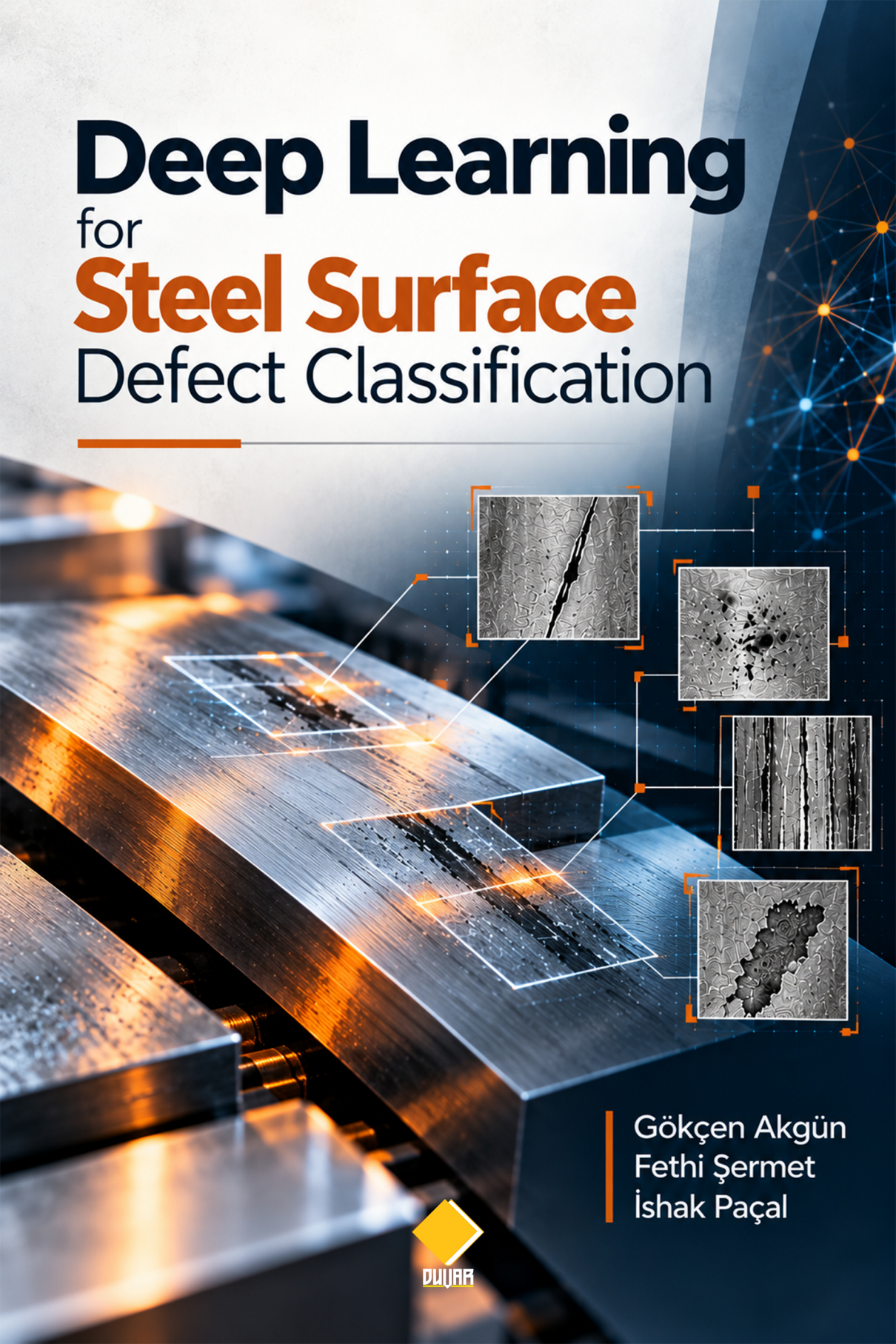
Deep Learning for Steel Surface Defect Classification



Gökçen Akgün
Fethi Şermet
İshak Paçal

DUVAR

Deep Learning for Steel Surface Defect Classification

Convolutional Networks, Vision Transformers, and Hybrid Models

Gökçen Akgün

Fethi Şermet

İshak Paçal



Deep Learning for Steel Surface Defect Classification
Convolutional Networks, Vision Transformers, and Hybrid Models
Gökçen Akgün
Fethi Şermet
İshak Paçal

General Director: Berkan Balpetek
Cover and Page Design: Duvar Design
Printing : November-2026
Publisher Certificate No: 49837
E-ISBN: 978-625-5471-37-6

© Duvar Yayınları
853 Sokak No:13 P.10 Kemeraltı-Konak/İzmir
Tel: 0 232 484 88 68
www.duvaryayinlari.com
duvarkitabevi@gmail.com

Abstract

Deep learning can distinguish steel surface patterns with high benchmark accuracy, yet an inspection system must also account for class-specific errors, training variability, predictive confidence, and changing acquisition conditions. This book develops the foundations of machine learning, convolutional neural networks, vision transformers, and hybrid architectures, then examines seven pretrained recognition pipelines on SteelDefectX10. EfficientNetV2-M, ConvNeXt-Base, InceptionNeXt-Base, FasterNet-M, ViT-Base/16, Swin-Base, and CAFormer-M36 are evaluated using seeds 0, 1, and 2 on the same balanced test set of 450 images across ten classes.

ConvNeXt-Base achieves the highest mean accuracy and macro-F1, both 99.48%, with an accuracy sample standard deviation of 0.13 percentage points. Swin-Base achieves 99.19% accuracy and CAFormer-M36 98.89%. The 0.30-percentage-point difference between the leading pair corresponds to approximately 1.33 images per run; its paired image-bootstrap interval includes zero. None of 63 seed-specific exact McNemar comparisons remains significant after global Holm adjustment. Across 9,450 predictions, 112 errors are concentrated on 25 distinct images, predominantly involving Oil spot and Inclusion. One Oil spot image is misclassified in all 21 runs. Swin-Base has the lowest mean negative log likelihood and Brier score, while FasterNet-M has the lowest ECE under the specified 15-bin scheme.

These findings establish a comparison of complete pretrained pipelines rather than an isolated architectural effect. Differences in initialization and preprocessing, incomplete development-set provenance, and the absence of independent industrial validation constrain generalization. The combined analysis connects recognition accuracy with uncertainty, residual failure patterns, and the measurements needed for engineering deployment.

Keywords: steel surface defects; image classification; transfer learning; convolutional neural networks; vision transformers; hybrid architectures; calibration; SteelDefectX.

Preface

Visual inspection links what a camera records to decisions about manufactured products. Learning a defect category from a prepared crop is one component of that process. Estimating defect extent, determining its physical consequence, and deciding whether a product meets its specification require additional measurements and engineering criteria.

This book introduces the learning principles behind steel surface recognition and studies their practical implications through a common ten-class benchmark. The progression begins with classical machine learning and representation learning, continues through convolution, attention, and hybrid networks, and develops mathematical descriptions of seven contemporary architectures. The later chapters connect these mechanisms to a comparative experiment with three training seeds, class-level measurements, paired statistical analysis, and predictive-confidence assessment.

The emphasis is on interpreting strong recognition results with sufficient resolution to see their remaining weaknesses. A mean accuracy close to 99% can conceal recurring confusion in one category, greater sensitivity to initialization, or probabilities that differ substantially from observed correctness. Retaining individual runs and class-level tables makes those differences visible.

Researchers and engineering students can use the foundations and model descriptions to understand the computation. Practitioners can use the experimental analysis to distinguish a recognition benchmark from an industrial acceptance system. The final chapters address acquisition changes, normal surfaces, human review, and performance measurements needed when moving from image classification to production inspection.

Contents

Abstract **3**

Preface **4**

1. Introduction **9**

The inspection problem 9

Why compare several model families 10

Research questions 10

Main findings and scope 11

2. Steel Surfaces and Visual Inspection Tasks **12**

Appearance and manufacturing context 12

The ten categories in the experiment 12

Classification and localization 13

Closed label sets and unknown defects 14

Sampling units and physical independence 15

3. Machine Learning for Surface Inspection **16**

Learning from features and labels 16

Representing steel texture 16

Classical classifiers 17

From classification to anomaly detection 18

4. Deep Learning and Transfer **19**

Learning the representation 19

Optimization and normalization 19

Regularization and augmentation 20

Pretraining and adaptation 21

5. Convolutional Neural Networks **23**

Local filtering and spatial structure 23

Depth and information flow 24

Channel weighting and model scaling 25

Convolution in surface recognition 26

6. Vision Transformers 28
Images as token sequences 28
Global and hierarchical processing 29
Data efficiency and pretraining objectives 30
Reading transformer results 31
7. Hybrid Visual Architectures 32
Combining local and contextual processing 32
The MetaFormer interpretation 32
Architecture hybrids and system hybrids 33
What hybridization can establish 33
8. Research on Steel Surface Defect Analysis 35
Organizing a diverse literature 35
Image classification with edges and multiscale features 35
Hybrid recognition and detection pipelines 37
Few-shot classification and prototype adaptation 38
Detection through scale and feature interaction 38
Annotation quality and irregular defect regions 39
Segmentation and class-guided spatial prediction 40
Scarce data and generated samples 41
Efficiency and production data 42
What the literature establishes for this study 44
9. SteelDefectX10 and Data Preparation 45
Source dataset and study subset 45
Class selection and balanced sampling 45
Exact repetitions and visual dependence 46
Partitioning and model selection 47
Label mapping and sample identity 48
Dataset documentation and the overview figure 48

10. The Seven Recognition Models 50
A common notation 50
EfficientNetV2 M 50
ConvNeXt Base 52
InceptionNeXt Base 53
FasterNet M 54
Vision Transformer Base with 16 pixel patches 56
Swin Base 57
CAFormer M36 58
Interpreting the comparison 59
11. Experimental Protocol and Reproducibility 60
Comparative design 60
Training and computing environment 60
Evaluation transformations 62
Model selection and test interpretation 63
Randomness and experimental replication 63
Reproducibility and provenance 63
12. Evaluation Measures and Statistical Interpretation 65
Hard decisions and the confusion matrix 65
Multiclass correlation 65
Ranking scores 66
Probability quality 67
Seed means and standard deviations 67
Paired image bootstrap 68
Exact paired tests and multiplicity 69
13 Comparative Results 70
Overall recognition performance 70
Variation across the three seeds 71
Individual runs and multiclass correlation 73
Class recall 74

Direction of confusion 79
Ranking and hard decisions 82
Conditional intervals and paired differences 82
Complete paired comparisons 84
14. Residual Errors and Predictive Confidence 87
A small set of difficult images 87
Shared errors and complementarity 88
Probability scores and calibration 89
From confidence to review 91
Explaining decisions 92
15. Engineering Implications 93
Choosing a candidate pipeline 93
Measuring speed meaningfully 94
The missing normal class 95
Acquisition and distribution shift 95
Failure costs and human decisions 96
Monitoring after deployment 97
16. Limitations and Future Research 98
Data provenance and independence 98
Architecture and pretraining are confounded 98
Limited repetitions and near-saturated metrics 99
Incomplete fitting records 99
Focused literature rather than a systematic review 99
A practical sequence of extensions 100
17. Conclusions 101
Glossary 103
References 106

1. Introduction

The inspection problem

Steel manufacturing produces surfaces with varied textures, reflections, and process traces. Some patterns are associated with cracks, inclusions, oxidation, pits, scratches, or contamination [1, 2]. Other visible features are tied to a manufacturing operation and are not automatically equivalent to a structural failure [2]. An image recognition system encounters the appearance of these patterns. It does not directly observe the complete process history, material composition, subsurface condition, or engineering consequence of a defect.

Recent research reflects several approaches to this problem. Zong and colleagues place visual monitoring within a wider discussion of rolling-process control, where prediction and parameter adjustment are linked to dimensional consistency and product quality [3]. Zhang and colleagues emphasize the relationship between welding processes, defect formation, and the imaging methods used to inspect them [4]. These perspectives help define the boundary of a surface-image classifier. It can contribute an observation to a quality-control process, but the observation must be interpreted alongside the relevant engineering requirements.

This book concentrates on image-level classification. Each input image receives one label from a fixed set of ten categories. The experiment uses a balanced test set, a common resolution, seven pretrained architectures, and three training seeds. The scope is deliberately specific. It permits direct comparisons of predictions on the same images while avoiding claims about object localization, pixel-level defect extent, material strength, or real-time line operation that the experiment does not measure. The dataset source, SteelDefectX, provides a broader setting than this classification task alone. Zhao and colleagues introduce 7,778 images spanning 25 categories with several forms of textual annotation, and evaluate vision-language tasks [2]. The present study uses ten selected categories as a conventional supervised image classification problem. It does not train a language model, assess text generation, or reproduce the original vision-language benchmark.

Why compare several model families

A steel-defect image may require attention to a small local texture or to the larger spatial arrangement of a surface pattern [5, 6]. Convolutional networks, patch-based transformers, and hybrid architectures process that information differently [7]. Their pretrained weights also encode experience acquired before the defect images are seen [8]. A useful comparison therefore considers both the architectural family and the complete transfer-learning pipeline.

The seven models cover modern convolutional designs, a conventional patch-based transformer, a hierarchical transformer, and a hybrid model. This range allows the study to ask whether different recognition approaches produce similar errors or whether they specialize in different classes. A common test set makes the answer observable at the image level. If two models have similar accuracy but fail on different images, their relationship is different from two models that share almost every mistake [9].

Three training seeds add another dimension. An individual run can benefit from a favorable initialization of the classification head, data ordering, or stochastic training transformation [10, 11]. Reporting one run hides this sensitivity. Reporting three runs exposes some of it, although three seeds remain a limited sample of training randomness. The purpose is to describe variability rather than to imply that every possible source of experimental uncertainty has been captured. The comparison also helps separate different meanings of performance. Accuracy measures hard decisions. Ranking metrics assess how class scores order positives and negatives. Probability-based measures evaluate the distribution assigned to the correct label. Model size describes parameter storage and part of the computational burden. None of these quantities can substitute for a missing measurement of latency, memory use, or performance on a new industrial source [12; 13].

Research questions

The experiment is organized around five questions. First, how do the seven pipelines compare in accuracy and macro-F1 on a fixed ten-class test set? Second, how much do their results change across seeds 0, 1, and 2? Third,

which classes and individual images account for the remaining errors? Fourth, does the model with the highest hard-label accuracy also provide the best probability estimates? Fifth, what conclusions remain valid when pretraining sources, preprocessing, and the size of the test set are considered explicitly?

The comparison contributes a common-test analysis of seven pretrained pipelines, with three complementary levels of evidence: variation across repeated fits, class-specific discrimination, and image-level agreement in residual errors. Probability scoring and paired inference extend the interpretation beyond the highest mean accuracy. The design evaluates existing architectures under a defined recognition task; it does not propose a new network.

Main findings and scope

ConvNeXt-Base has the highest mean accuracy and macro-F1. Its three runs make three, two, and two errors on 450 images. Swin-Base follows with three, four, and four errors. The average accuracy difference between the two is about 0.30 percentage points, equivalent to approximately 1.33 correctly classified images per run. The ranking is real as a description of this experiment, but the difference is too small to support an unrestricted claim of superiority.

Several other findings change the interpretation. CAFormer-M36 makes five errors in every run, yet its probability outputs are not identical. FasterNet-M and ViT-Base/16 show larger seed-related accuracy variation than the other models. Oil spot is the most difficult class when all runs are considered together. One image in this class is wrong in every run. Swin-Base provides better average negative log-likelihood and Brier score than ConvNeXt-Base, showing that different evaluation criteria can lead to different preferred pipelines. The conclusions apply to balanced recognition among the ten selected categories. Production prevalence, unknown defects, localization, and a new acquisition population define additional evaluation questions that require corresponding data and outcomes.

2. Steel Surfaces and Visual Inspection Tasks

Appearance and manufacturing context

The appearance of a steel surface combines material texture, illumination, imaging geometry, and process-related variation [2, 14]. A dark mark may arise from contamination, an inclusion, oxidation, or a change in reflectance [14, 15]. A line may be a scratch, a crack-like pattern, or a trace associated with joining or processing [2, 4, 14]. A classifier operates on the recorded appearance, so its labels must be understood as dataset categories rather than complete physical diagnoses [2, 16]

The rolling review by Zong and colleagues describes quality monitoring in conjunction with operational parameters and feedback control [3]. This context is useful because many image datasets contain only cropped surface views [2, 13, 14]. The cropped image omits information that a production system might use, such as coil identity, rolling temperature, tool condition, line speed, or upstream alarms [13, 17]. A model that performs well on crops can therefore still be incomplete as a monitoring system [5, 13].

The review of welding inspection by Zhang and colleagues provides a related caution [4]. Welding defects are influenced by process mechanisms and can require imaging modalities beyond a conventional surface photograph [4, 16]. A category called Welding line in a surface dataset is not equivalent to a diagnosis of an internal weld discontinuity [2, 16, 18]. Linking recognition to quality assessment requires the acquisition modality and annotation meaning to remain explicit [14, 19, 20].

The ten categories in the experiment

The selected classes are Crazing, Dark scratches, Inclusion, Oil spot, Patches, Pitted surface, Punching, Rolled in scale, Silk spot, and Welding line [2]. Their operational role is to define the supervised label set. The names suggest different visual cues, but the classifier receives only the pixels and the label during training [2, 14]. Any explanation of the physical origin of a particular test image would require additional process evidence [3, 17].

Crazing concerns a crack-like surface pattern with distributed local structure [21]. Dark scratches emphasize elongated marks with low intensity relative to the surrounding surface [2]. Inclusion and Oil spot can both involve dark irregular regions, which makes their separation an important test of whether the model learns fine appearance differences rather than only a generic dark-region cue [2, 21]. Patches represent a spatially extended pattern, while Pitted surface concerns distributed small-scale surface variation [6, 21].

Punching, Silk spot, and Welding line broaden the visual and process context of the label set [22, 23]. They should not be mapped automatically to a common severity scale. Punching may identify a hole-related category in the originating dataset. Welding line identifies a line-related category rather than a complete assessment of joint quality [4]. Rolled in scale introduces a category associated with irregular surface material [21]. These labels can be useful for recognition while remaining insufficient for acceptance or repair decisions [13].

A practical annotation manual should include reference images, positive and negative examples, borderline cases, and an explanation of whether labels describe appearance or confirmed process origin [19]. Without that manual, a researcher may assume a sharper physical distinction between categories than the dataset supports [14]. The shared errors between Oil spot and Inclusion in the present experiment make such semantic clarification particularly important.

Classification and localization

Image-level classification answers which category is assigned to an image [13, 14]. Object detection additionally estimates where an object or defect region lies, usually through bounding boxes [13, 14]. Segmentation assigns labels to pixels or regions [14]. These outputs serve different inspection needs [13]. A classifier may be sufficient when the input is already a carefully selected defect crop [13]. It is insufficient when a system must search a large surface containing multiple regions or distinguish defect extent from background [13].

The distinction also changes the evaluation metric. Classification accuracy counts correct image labels [107]. Detector mAP depends on both class

assignment and localization criteria, including the intersection-over-union threshold [13, 14]. Segmentation overlap depends on the agreement between predicted and reference masks [2, 14]. A classification accuracy of 99% cannot be interpreted as stronger than a detector mAP of 85% without considering that the latter solves a different problem [13].

Moon and colleagues explicitly separate localization and classification in a hybrid framework [24]. Shen and colleagues use class knowledge to guide a salient-object decoder [25]. These studies show that class information can support spatial prediction, but they do not make image-level classification interchangeable with localization. For the present study, confusion matrices and class-level metrics are appropriate [26, 27]. Bounding-box and mask claims remain outside the measured task.

Closed label sets and unknown defects

The ten-class experiment is a closed-set problem: every evaluated sample belongs to one of the ten labels, and the model chooses among those labels [28, 29]. In a production setting, an image may be normal, out of focus, partially occluded, or affected by a defect absent from the training data [30, 31]. A closed-set classifier can still assign a high score to one of its known classes [28, 32]. This behavior cannot be assessed from a test set containing only known defect categories [29]. The distinction is particularly relevant because the selected subset has no separate normal class. The experiment evaluates recognition among defect categories, not defect-versus-normal screening [13, 33]. A normal surface cannot be treated as correctly classified merely because the model returns one of the defect labels with low confidence [32, 34]. A screening system would require a suitable normal-data population and an explicit decision rule [13, 33].

Confidence-based unknown-category baselines provide one starting point for investigation [32]. Unknown-category handling could be implemented through rejection, novelty detection, or a broader model trained to distinguish normal and abnormal regions [33, 35, 36].

Those additions would change the task and require validation [29]. They should not be implied by the use of softmax probabilities alone [29, 32]. The probability assigned to a known class reflects the model's allocation

within the available label set, not a guarantee that the input belongs to that set [29, 34].

Sampling units and physical independence

A file is a convenient unit for model input, but it may not be the appropriate unit for assessing independence [13]. Two crops can come from the same defect, adjacent locations on the same strip, or repeated views of the same surface. Such images may have different filenames while sharing strong visual dependence [13]. If related images appear in both training and test partitions, the model can exploit familiarity that would not transfer to a new production unit [13, 37].

Removing exact duplicate pixels is useful because it eliminates one clear form of repetition [13, 38]. It does not remove every relationship between crops or acquisition sessions. Production identifiers, image coordinates, timestamps, and source-dataset provenance help establish stronger partitions [13, 19]. The appropriate split may be by coil, production run, or acquisition site rather than by individual image [13].

This study confirms that all models are evaluated on the same 450 unique sample identifiers. The observation supports a paired model comparison [39, 40]. It does not establish that those images are independent production units or that related content is absent from development data [13]. The results should therefore be interpreted as an image-level benchmark. A future production study should define its physical sampling unit, such as the coil, production run, or acquisition site, before partitioning the data and before selecting a final deployment model.

3. Machine Learning for Surface Inspection

Learning from features and labels

Machine learning links a representation of an observation to a prediction rule [41]. For surface inspection, an observation may be an image, a vector of texture statistics, or a record of manufacturing measurements [3, 14]. Supervised learning estimates a mapping from inputs to reference labels, while unsupervised learning studies structure without the same label information [3, 14]. The distinction concerns the information available during fitting rather than whether the final algorithm uses a neural network [3, 14]. Statistical learning texts provide the probabilistic and optimization framework behind these tasks [42, 43].

Let a training set contain n pairs of an input x and a class y. The empirical objective averages a loss over those pairs and can include a penalty on model complexity [42-44]. A low training loss establishes agreement with the development examples; useful inspection requires agreement on observations drawn from the intended application [13, 15, 45]. The loss, class distribution, and sampling unit therefore belong to the definition of the learning problem [13, 15]. If related crops of the same surface appear in both development and test sets, even a simple classifier can benefit from information that would be unavailable on a new coil [13, 37]. Generalization depends on the relationship between sampled observations, not only on parameter count [37].

Representing steel texture

Classical image recognition separates feature extraction from classification [14]. Gray-level co-occurrence features describe relationships between neighboring intensities [46]. Local binary patterns encode local intensity comparisons and summarize recurring texture configurations [47]. Histograms of oriented gradients describe the distribution of edge directions [48], while scale-invariant keypoints and descriptors support matching local image structure [49]. An edge detector can expose contours before higher-level processing [50]. These representations emphasize different aspects of a surface: distributed texture, narrow lines, corners, or irregular boundaries [14]. For a steel

image, the feature choice determines which distinctions remain available to the classifier [1]. A texture histogram may describe Pitted surface effectively while suppressing the position of one small Inclusion [21]. An edge histogram may emphasize Dark scratches but respond strongly to illumination boundaries [14]. Selecting a descriptor is therefore a task-dependent modeling decision [14]. It should be assessed on validation data with the same transformation and feature pipeline used during testing [51, 45, 52].

Feature scaling is part of that pipeline [51]. Means, standard deviations, principal components, and feature-selection thresholds must be estimated within the training data [43, 53]. Computing them from the entire dataset makes test observations influence development [38, 52, 53]. The same rule applies to apparently harmless operations such as choosing an intensity threshold after inspecting all available images [45]. A complete implementation preserves the fitted transformation together with the classifier [51].

Classical classifiers

Nearest-neighbor classification assigns a label through similarity to stored examples and provides a direct reference for the role of the representation [54]. A support-vector classifier seeks a separating decision boundary with a margin, using a kernel when a nonlinear feature geometry is needed [44]. Random forests average predictions from randomized trees [55], and gradient boosting constructs an additive model through successive improvements to a loss [56]. Their different assumptions make them useful comparators rather than interchangeable names for one baseline.

XGBoost introduces an efficient implementation of boosted trees with regularization and system-level design choices [57]. CatBoost addresses categorical features and ordering-related issues in boosting [58]. These methods are particularly relevant to manufacturing records or combinations of image descriptors and process measurements [3]. They are not image backbones by themselves: a suitable vector representation is needed before a tree model receives an image-derived input [14]. Software libraries such as scikit-learn support reproducible implementations of the classical methods and their preprocessing stages [51]. A classical baseline

becomes valuable when it addresses a concrete question [38]. For instance, comparing local texture descriptors against a pretrained backbone can reveal the extent to which recognition performance improves when feature extraction is learned rather than handcrafted [6, 14]. Comparing image features with process variables can examine whether contextual measurements resolve visual ambiguity [3]. Neither comparison is performed in the present seven-model experiment. Classical methods provide the conceptual starting point and candidates for subsequent studies; the empirical tables report the seven deep recognition pipelines only.

From classification to anomaly detection

Whereas a supervised classifier learns to discriminate between known categories, an anomaly detector typically models a reference population of normal samples and seeks to identify deviations from it [14, 29, 33]. MVTec AD formalizes an industrial anomaly-detection setting containing normal training images and anomalous test images [33]. PatchCore uses representative features of normal patches for anomaly assessment [35]. These tasks explain why a normal-versus-abnormal system requires evidence beyond high accuracy on preselected defect crops [13].

SteelDefectX10 contains ten known surface categories and has no separate normal class. It supports supervised category recognition. Adding anomaly screening would require representative normal steel, an explicit abnormality score, a validation threshold, and an independent evaluation of false alarms and missed defects [13, 33, 35]. Such a component can precede category classification, but its errors and coverage must be measured separately [13, 24]. A two-stage system can fail in screening even when the downstream classifier is accurate on the defect crops it receives [13, 24].

4. Deep Learning and Transfer

Learning the representation

Deep learning embeds both feature extraction and prediction within a composition of trainable transformations [41, 59]. Whereas early layers may represent local intensity variations, subsequent transformations combine these responses into increasingly selective features [41, 60]. However, the meaning of each channel cannot be specified in advance by a class name; rather, it may emerge through learning [60, 61]. General references on deep learning discuss representation learning, optimization, and the relationship between network depth and statistical estimation [41, 59].

In a sequence of L layers, each transformation operates on the output of the preceding layer [59]. Each layer typically applies a learned linear mapping, defined by weights and bias terms, followed by a nonlinear activation function; together, these weights and biases constitute the model parameters [59]. The nonlinearities enable the composition to express interactions beyond those captured by a single linear mapping [59]. ReLU and related activation functions are common examples; rectifier behavior has been investigated in studies of both learning and weight initialization [62, 63]. During training, backpropagation applies the chain rule to compute the derivatives of the loss with respect to the parameters [64]. The central practical challenge is to learn a useful representation without fitting the incidental characteristics of a small training population [3, 14, 65].

Optimization and normalization

Gradient-based optimization updates parameters using the direction indicated by a loss derivative. Stochastic methods estimate that direction from a minibatch [66], so data ordering and batch composition can change the training trajectory [67]. General optimization analyses distinguish the objective, gradient estimator, learning-rate rule, and computational budget [67]. Initialization also matters: variance-aware initialization helps prevent activations and derivatives from becoming poorly scaled before learning begins [36].

$$\theta_{t+1} = \theta_t - \eta_t \widehat{\nabla}_\theta \mathcal{L}_t \quad (1)\ [68]$$

Here eta is the learning rate and the hatted derivative is a minibatch estimate [59, 67]. Momentum and Nesterov acceleration [69, 70], as well as adaptive methods [71], modify this basic update. Adam maintains exponentially decaying averages of past gradients and squared gradients, that is, estimates of the first and second gradient moments [71], while decoupled weight decay distinguishes a regularization update from the adaptive gradient operation [72]. Their presence in the literature does not imply their use in this experiment: the present experiment uses SGD with momentum 0.9 and Nesterov acceleration. Batch normalization normalizes activations with minibatch statistics during training and with stored running estimates during inference [73, 74]. Layer normalization instead computes its statistics within each observation over specified feature dimensions [74].

Because batch statistics become unreliable when batches are small, and because training and inference use different statistics, the two layer types behave differently under small batches and in evaluation mode [74, 75]. Among the studied architectures, ConvNeXt uses layer normalization, whereas InceptionNeXt and FasterNet use batch normalization (Chapter 10); the running statistics of the batch-normalization layers are therefore part of each checkpoint. Evaluation must set the model to inference mode; otherwise a batch-normalization layer can change the result when batch composition changes [74]

Regularization and augmentation

Regularization comprises modifications to the learning procedure that aim to reduce generalization error rather than training error, limiting the extent to which a model fits idiosyncrasies of the training data [59, 76]. Dropout randomly suppresses unit activations during training [65]. Data augmentation generates transformed versions of the training images, such as cropped, scaled, flipped, or colour-shifted copies [14]. Each transformation must preserve the class label; whether it does so depends on the task and on the magnitude of the transformation [77]. Aggressive geometric or photometric changes that are acceptable for natural-object photographs can remove or distort narrow surface marks, whereas defect inspection requires the texture continuity and local morphology of the

defect to be preserved [5, 77]. The label-preserving range of each transformation should therefore be established for the defect classes before an augmentation policy is adopted.

Mixup forms new training examples by linearly interpolating pairs of images and their labels [15, 78, 79]. CutMix pastes a region of one image into another and mixes the labels in proportion to the pasted area [79, 80]. AutoAugment searches for augmentation policies with reinforcement learning [81], whereas RandAugment replaces this search with a reduced space of uniformly sampled operations controlled by a small number of parameters [79, 82]. These methods are candidates for future defect-recognition studies, but their effects should be measured under a declared validation protocol, and the complete augmentation configuration of every run should be reported so that observed differences are not attributed to augmentation retrospectively (Chapter 11).

Learning-rate schedules and numerical precision are independent design choices. Warm-restart schedules periodically reset the learning rate to a higher value and decay it again [83]. Mixed-precision training performs part of the computation in reduced precision to lower memory use and computation time, and uses safeguards such as a full-precision master copy of the weights and loss scaling to prevent loss of numerical range [84]. All runs evaluated here use float32, with automatic mixed precision disabled in both training and evaluation (Chapter 11). A future precision comparison should assess both execution cost and the resulting predictions, rather than assume identical behaviour.

Pretraining and adaptation

Transfer learning uses knowledge acquired on a source domain and task to improve learning on a different target domain or task [85, 86]. In image classification, the pretrained backbone can be kept frozen while only a new classifier is trained, or all parameters can be fine-tuned on the target data [21, 37]. Transfer learning does not require the source and target data to be identically distributed [14, 85]. The low-level features of a network pretrained on natural images, such as edge and texture detectors, can be shared with steel recognition [37]. However, its deeper representations are oriented toward semantic objects and scene-level appearance, which can

introduce a representation bias when they are transferred to grayscale, texture-dominant industrial images [5, 14, 15].

ImageNet established a large-scale recognition benchmark [87], and ImageNet-pretrained weights have become the usual initialization for backbones in steel defect recognition [5]. The seven pipelines use model-specific pretrained weights, and the pretraining data also differ between them. CAFormer-M36, InceptionNeXt-Base, and FasterNet-M were pretrained on ImageNet-1k. ConvNeXt-Base, Swin-Base, ViT-Base, and EfficientNetV2-M were first pretrained on ImageNet-21k/22k and then fine-tuned on ImageNet-1k (Table 3). The comparison therefore reflects both the architectural design and the source representation. Using the same 224 × 224 input resolution for all models does not make their initialization histories equivalent. For the ten-class target task, the original classifier is replaced by a new head with ten outputs [21].

The output order of this head must match the class-index mapping of the dataset (Chapter 9). Input preprocessing must use the same channel order and normalization statistics as the pretrained model [88] (Chapter 11). When single-channel grayscale images are supplied to a three-channel backbone, the conversion rule should be reported, because studies on steel datasets also use different channel conventions [6]. Replicating the intensity across three channels is a representation choice; it does not add independent measurements of the surface.

5. Convolutional Neural Networks

Local filtering and spatial structure

A convolutional layer applies learned filters repeatedly across an image or feature map. Because the same filter weights are shared across all spatial positions, a filter can respond to a pattern wherever it appears, while local connectivity preserves the spatial relationships between neighbouring pixels [89, 90]. Early image-recognition networks showed that convolution, nonlinearities, and subsampling can be trained together end to end [89]. In steel inspection, this local connectivity is used to capture local defect information such as edges, small pits, elongated marks, and fine-grained texture variations [5, 91].

Each output value of a convolutional layer is computed as a weighted sum over a k × k spatial neighbourhood across all C input channels, plus a bias term; each output channel has its own set of filter weights [59, 90]. Padding and stride determine how the filter is positioned at the borders and how far it moves between successive outputs. Because the kernel is not flipped, the operation implemented in deep-learning libraries is formally a cross-correlation, although it is conventionally called convolution [59]. A larger kernel covers a wider neighbourhood, whereas a stack of smaller kernels can achieve the same receptive field with fewer parameters and additional nonlinearities [92]. The theoretical receptive field identifies which input locations can affect an output, but it does not imply that all of these locations contribute equally; in practice, the effective contribution is concentrated near the centre of the field [93].

Depthwise convolution filters each channel separately, whereas pointwise (1 × 1) convolution mixes channels at each spatial location. Combining the two reduces computation and parameters compared with a standard convolution [94, 95]. Xception develops depthwise-separable processing as a design principle [94], and MobileNet uses it for compact recognition [95]. MobileNetV2 reorganizes the expansion and projection sequence through inverted residuals and linear bottlenecks [96]. A reduction in floating-point operations does not necessarily translate into lower latency,

because the speed of these operations also depends on memory access, the device, and the implementation [97].

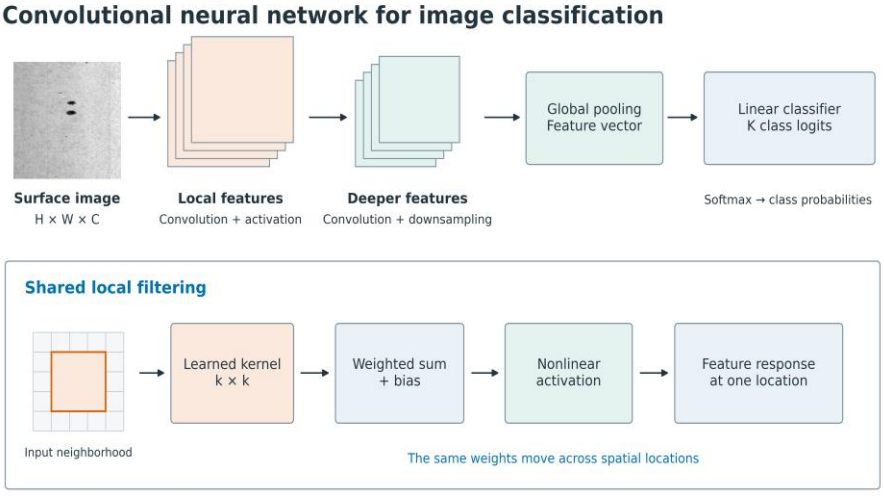


Figure 1. Shared local filtering and a generic CNN classification pipeline. The example input is a SteelDefectX10 test image; stacked feature maps are schematic and do not represent computed activations. A sliding kernel applies the same learned weights across locations before nonlinear transformation and subsequent aggregation.

Depth and information flow

VGG showed that recognition accuracy can be improved by stacking many small (3 × 3) convolutional filters in deeper networks [92]. Inception uses parallel convolutional paths with different kernel sizes to capture information at several scales within the same block [98]. These two designs correspond to two ways of representing defect appearance: local features composed progressively through depth, and spatial scales processed explicitly in parallel. Inception-based backbones have also been used for steel strip defect recognition under limited data and large appearance diversity [21]. A family name such as VGG or Inception describes a design strategy; the evaluated model is defined by its selected depth, width, and input transformation.

Residual learning adds the output of a learned transformation to a shortcut connection that carries the block input unchanged, so that each block learns

a correction to its input rather than a complete new representation [18, 99]. In the original ResNet, a ReLU is applied after this addition [99], whereas later designs keep the addition free of any subsequent nonlinearity so that the shortcut passes the signal directly between blocks [100]. The residual blocks of the seven studied models follow this latter form (Chapter 10). Dense connectivity concatenates the outputs of all preceding layers within a block [101]. Aggregated residual transformations split the residual branch into groups of parallel transformations with identical topology [102]. These designs change how features and gradients propagate through a deep network, and they should not be treated as equivalent merely because each contains a skip or concatenation operation. An identity shortcut requires the block input and the output of the residual branch to have the same shape; when the dimensions change, a projection or downsampling operation is applied to the shortcut [99]. In inspection, a shortcut can preserve useful low-level information while the residual branch learns a correction, but it does not ensure that the model disregards background cues. The shortcut is an architectural mechanism and does not by itself ensure that defects are localized correctly.

Channel weighting and model scaling

Squeeze-and-excitation (SE) compresses each feature map into a channel descriptor by global average pooling and uses a small gating network to learn a weight for each channel [103]. SE-based channel attention has also been used in steel surface defect classification [6]. Because the same weight is applied to every spatial position of a channel, the mechanism changes the relative contribution of channels; it does not produce a pixel-level map indicating where a defect is located, which is the role of spatial attention [104]. EfficientNet scales network depth, width, and input resolution jointly with a single compound coefficient [105]; the model evaluated here is from its successor, EfficientNetV2 (Chapter 10). RegNet derives families of network structures by progressively constraining a design space rather than designing individual networks [106]. Normalizer-free networks show that high recognition accuracy can be reached without batch normalization when training is stabilized by other means, such as adaptive gradient clipping [107]. Together, these studies show that the accuracy of an architecture depends not only on its capacity but also on

how it is scaled and trained. Structural reparameterization trains a model with a multi-branch structure and converts it after training into an equivalent single-path convolutional form for inference, as in RepVGG [108]. ConvNeXt V2 combines a revised convolutional block with masked-autoencoder pretraining [109]. Neither development implies that an inference model with a familiar CNN appearance has the same fitting history as another model. Architecture diagrams should therefore identify the tested variant rather than combine features from several generations of a family.

Convolution in surface recognition

The four convolutional models in the experiment differ in how they organize spatial and channel processing. EfficientNetV2 uses fused inverted-bottleneck blocks, in which a standard convolution replaces the depthwise and expansion layers, in its early stages and depthwise inverted bottlenecks in its later stages [110]. ConvNeXt uses large (7 × 7) depthwise kernels within a residual block redesigned along the lines of vision transformers [111]. InceptionNeXt splits the channels of a depthwise convolution into parallel branches: an identity path, a small square kernel, and orthogonal horizontal and vertical band kernels [112]. FasterNet applies a standard spatial convolution to only one quarter of the channels and leaves the remaining channels unchanged, which reduces both computation and memory access [97]. Classifying all four as CNNs is useful for orientation, but it does not remove these computational differences (Chapter 10).

Steel surface defects range from small or elongated marks to defects that are spread across the surface [22]. Inspection networks must therefore preserve small and elongated structures while providing a sufficiently large receptive field and contextual information [5, 13]. On this basis, a band kernel could help represent an elongated mark such as a dark scratch, and a large receptive field could help capture crazing, which appears in a scattered manner across the surface rather than as a single localized mark [21]. Partial convolution could offer a useful compromise between accuracy and execution speed [97]. These links remain hypotheses until they are tested. The present experiment compares complete pipelines, not

isolated mechanisms. Attributing a gain to one operation requires a matched ablation in which capacity, initialization, input transformation, training budget, and model selection are held constant, as in steel-defect studies that evaluate one module at a time [6]. Without such control, training procedure alone can produce gains comparable to those attributed to architecture: in the development of ConvNeXt, a modernized training recipe by itself raised the accuracy of an unchanged ResNet-50 [111].

6. Vision Transformers

Images as token sequences

A vision transformer divides an image into non-overlapping patches and represents it as a sequence of patch embeddings [7, 113]. Each flattened patch is mapped by a learned linear projection to a common feature dimension, and learned position embeddings are added to supply the spatial order that the attention operation itself does not encode [113]. The original ViT showed that, with sufficiently large pretraining data, a transformer applied to this representation with minimal modification can achieve competitive large-scale image recognition [7, 113]. In strip steel defect detection, this global self-attention has been adopted to strengthen the modelling of relationships across the whole image [91]. With a 224 × 224 input and 16 × 16 patches, the ViT evaluated here receives 14 × 14 = 196 patch tokens, to which a class token is prepended, giving 197 tokens (Chapter 10).

Attention computes relationships among token representations. Each token is projected by three learned matrices into a query, a key, and a value [7, 114]. The similarity between the queries and keys, normalized by a softmax, determines the attention weights, and the output is the corresponding weighted sum of the values [114]. This scaled dot-product attention, introduced in the original Transformer, is the basis of the attention operations considered in this book [114].

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^{\mathsf{T}}}{\sqrt{d}} + B\right)V \quad (6)\ [115]$$

Here Q, K, and V are the query, key, and value matrices, d is the query/key dimension, and B is a learned relative position bias added to the attention logits [115]. Without B, the expression reduces to the scaled dot-product attention of the original Transformer [114]. Multihead attention computes several such attention functions in parallel, each with its own learned projections, and concatenates and projects their outputs, allowing the model to attend to information from different representation subspaces [7, 114]. The learned associations of the individual heads are not assigned a predefined physical meaning. Attention weights are intermediate

quantities of the computation and should not be interpreted directly as explanations of a defect's causal importance: attention maps do not necessarily identify the inputs that determine a prediction [117, 118], and in steel inspection, attention can favour cues that are specific to the training distribution and degrade under a domain shift [13].

Figure 2 shows patch embedding, the class token and position embeddings, and the two pre-normalized residual sublayers (multihead self-attention and a channel MLP) of a conventional vision transformer; its structure is adapted from [113]. The 4 × 4 grid in the figure illustrates partitioning only; the evaluated ViT uses a 14 × 14 grid of 16-pixel patches (Chapter 10).

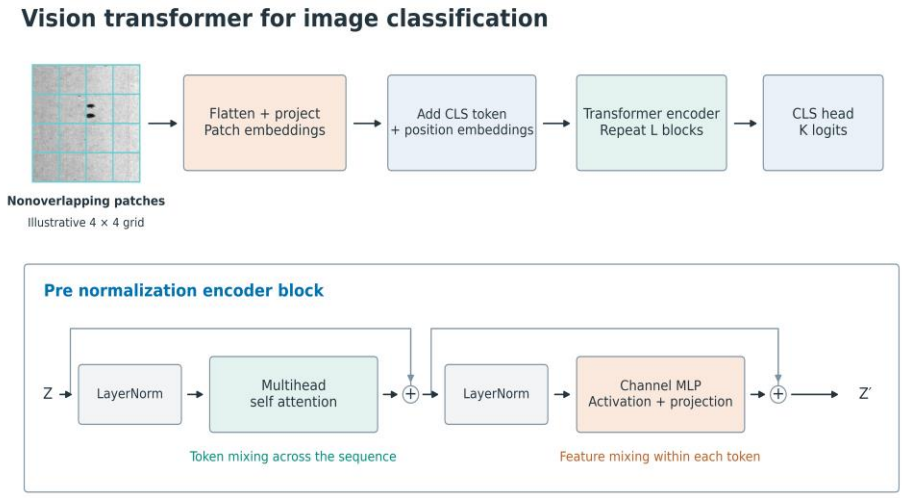


Figure 2. Patch embedding and a conventional pre-normalization vision transformer. The class token and position embeddings enter a sequence processed by attention and channel-MLP residual sublayers. The 4 × 4 display grid illustrates partitioning; the evaluated ViT uses 196 image patches plus one class token. Stochastic depth is omitted.

Global and hierarchical processing

Global self-attention relates every token to every other token within a block, so the size of its attention matrix, and thus its cost, grows quadratically with the number of tokens [7, 115]. Patch size and input resolution are therefore computationally important: smaller patches

preserve finer structure but increase the number of tokens. Resizing also matters for steel images, because tiny defects are easily degraded during downsampling [5], and resizing a steel crop can change the apparent width of a scratch or the number of pixels that represent a pit. Resolution is therefore both a computational choice and a limit on the visual detail available to the model.

Hierarchical transformers progressively reduce the spatial resolution of their feature maps. Pyramid Vision Transformer builds a multiscale feature pyramid within a transformer backbone [116], and Twins revisits the design of spatial attention by combining locally grouped and global sub-sampled attention [119]. Swin computes self-attention within local non-overlapping windows and shifts the window partition between successive blocks, which introduces connections between neighbouring windows [115], window-based Swin blocks have also been used in strip steel defect detection [91]. Swin V2 extends this design to larger model capacity and higher input resolution [120]. These hierarchical designs differ from the plain ViT, which applies global attention repeatedly at a single, fixed patch-grid resolution.

The computational cost of the two attention schemes illustrates this difference. In global attention, each of the N tokens is compared with all N tokens, so the number of pairwise associations grows with N^2. In window attention, each token is compared only with the m tokens of its own window, so the number of associations grows with $N \times m$; when the window size is fixed, the cost therefore increases linearly rather than quadratically with the number of tokens [115]. Both schemes also include the cost of the linear projections that produce the queries, keys, and values, which is the same for the two [115].

This comparison concerns the attention modules only; channel MLPs and other layers add further computation, so it does not describe the FLOPs of the complete network.

Data efficiency and pretraining objectives

DeiT investigates data-efficient image transformers and distillation [7, 121]. Compact convolutional tokenization supplies another route to transformer recognition with smaller models and different data

requirements [122]. These developments show that the original patch projection is one possible input mechanism, rather than a necessary choice for every architecture containing attention.

Self-supervised pretraining also changes the initial representation. BEiT uses a masked-image modeling formulation [123], while masked autoencoders learn from reconstructing missing image content [124]. DINO studies self-supervised transformer features [125], and DINOv2 scales visual feature learning without ordinary supervised category labels [126]. The resulting weights should not be assumed to be interchangeable with the supervised variants used in this benchmark (Table 3). CLIP learns transferable visual representations through paired image and language supervision [127]. Its relevance is particularly clear for a source dataset with textual descriptions such as SteelDefectX [2]. However, the experiment uses conventional image-only classifiers and ten categorical labels (Chapter 11). Evaluating language-conditioned recognition or zero-shot transfer would require different models, prompts, and scoring rules [2]. The availability of text in the source does not turn the present image-classification experiment into a vision-language evaluation.

Reading transformer results

A transformer can learn long-range appearance relations, but that capacity does not ensure an advantage on every small image dataset [22, 113]. Pretraining, input resolution, optimization, and the sampled class boundaries can dominate a finite comparison [13, 111]. The present ViT and Swin have different spatial organizations and different pretrained variants (Chapter 10, Table 3). Their comparison therefore evaluates two specified pipelines rather than isolating global versus window attention as the only changing factor.

Transformer performance should be examined jointly through accuracy variation, class-level errors, and probability quality [34, 128]. Shared failures with convolutional models suggest a problem that may persist across token-mixing mechanisms, whereas systematic differences can motivate a controlled representation study [129]. The model-specific equations in Chapter 10 identify the computations evaluated in the experiment.

7. Hybrid Visual Architectures

Combining local and contextual processing

A hybrid visual model combines more than one processing mechanism within the learned representation [7]. Convolution can establish local features or perform downsampling before attention operates on a smaller token population [130; 131]. Attention can then communicate across locations that a local filter would connect only after several layers [91]. This arrangement changes the distribution of computation across spatial scales and should be described through its actual stage sequence.

CvT incorporates convolutional elements into a transformer hierarchy [130]. ConViT introduces a convolutional inductive bias into attention [132], and CoAtNet combines convolution and attention across network stages [131]. Their common hybrid label conceals different mechanisms: a convolutional projection, an attention constraint, and a stage-level transition are distinct design choices. Comparing them requires variant-specific definitions rather than one generic convolution-plus-transformer formula. MobileViT studies a compact combination of local convolution and transformer processing [133]. MaxViT uses complementary attention organizations with convolutional processing [134]. These designs extend the set of possible compromises among spatial detail, contextual association, memory, and execution [13]. Published family motivations do not establish their relative cost on the RTX 5080 or on the present fine-tuning task; those costs require measurements under a common device protocol [97].

The MetaFormer interpretation

MetaFormer separates the surrounding normalized residual architecture from the particular token mixer [135, 136]. A token mixer exchanges information across spatial positions, while a channel MLP transforms features at each position [135]. The distinction makes it possible to compare convolution, attention, pooling, or other mixing operations within a related block structure [135, 136]. It also prevents the entire residual block from being described as attention when only one of its sublayers performs attention.

In this template, each block applies a normalized token-mixing sublayer and a normalized channel-MLP sublayer, each with its own residual connection; implementations can also include residual scaling and stochastic depth [136]. CAFormer instantiates this framework with separable convolution in its early stages and attention in its later stages [136]. In the M36 variant, this transition occurs after two convolutional stages [136] (Chapter 10). Its computation is therefore a single trained hierarchy with changing token mixers, not an ensemble of independently trained CNN and transformer classifiers.

Architecture hybrids and system hybrids

The term hybrid can also describe an inspection system that combines several models or tasks. In steel inspection, such systems can combine image classification with a separate detector that localizes defects, as in a framework that uses CNN and transformer classifiers alongside a real-time detector [137]. A process-data model can also provide context to an image model [3]. These systems differ from CAFormer's internal stage design. They have additional interfaces where detection failures, crop selection, class mapping, or missing process measurements can affect the final decision [13].

An ensemble is another distinct construction. Combining the outputs of several independently fitted classifiers may exploit complementary errors, but it requires a specified combination rule and validation [129, 138]. In steel defect prediction, for example, different combination rules applied to the same base learners led to different best metrics [139]. A CNN-attention architecture does not become an ensemble merely because it contains two types of operation. Conversely, an ensemble of two CNNs is a multi-model system without being a CNN-transformer architecture [21]. Precise terminology helps readers understand which components were trained and what the experiment actually tested.

What hybridization can establish

The relevant scientific claim is narrower than an expectation that local and global features must always improve performance [13]. A hybrid adds design choices about where mechanisms change, how features are

normalized, and how resolution and width evolve [131, 136]. Those choices may improve a particular task or may create unnecessary complexity [13]. A matched experiment is needed to determine whether the hybrid's benefit comes from its mechanism rather than from capacity, pretraining, or preprocessing [13, 111].

CAFormer-M36 is included here as a practical hybrid recognition pipeline [136]. Its mean accuracy, ranking scores, shared errors, and parameter count can be compared directly with the other six evaluated pipelines (Chapter 13). The benchmark provides those measurements (Chapter 13). It does not include a controlled replacement of each CAFormer mixer or a stage-by-stage ablation (Chapter 11). Such an ablation would be a suitable extension when the representation question has been defined and an independent evaluation population is available [22; 13; 45].

8. Research on Steel Surface Defect Analysis

Organizing a diverse literature

Contemporary steel inspection research uses similar terminology for substantially different tasks [13, 14]. Some papers describe image-level classification as defect detection [140]. Others use the term for bounding-box localization [141], salient-region segmentation [25], or prediction from process variables [139]. A useful review begins by inspecting the output and the experimental protocol, rather than relying on the title alone [13].

The studies considered here are organized by their role in understanding the present benchmark. Image classifiers provide the closest methodological context [142, 143]. Detectors and segmenters explain problems of scale, localization, and annotation [25, 141, 144]. Few-shot methods address adaptation to scarce or new categories [145]. Generative methods examine the production of additional training samples [146]. Process-data studies show how image recognition may be combined with other measurements in an industrial system [3]. The literature synthesis compares the learning task, supervision, acquisition population, and evaluation criterion of each study (Table X). It identifies recurring design choices and their relevance to the present recognition problem; reported values retain the scope of their original protocols [13].

Image classification with edges and multiscale features

Wang and colleagues propose EAF-DenseNet121, which combines a learnable edge branch with entropy-guided feature fusion [142]. The design addresses a common recognition difficulty: an image may contain weak contours or visually similar regions whose distinction is lost in a deep representation [21, 22]. The reported classification accuracy is 99.17% on the NEU dataset and 82.89% on their GC10-DET-derived experiment [142]. The paper describes different dataset populations and separate training, validation, and test allocations [142]. These values therefore describe performance within those protocols, rather than a direct comparator for the ten-class SteelDefectX test.

The gap between the two reported accuracies is informative even without establishing external transfer. It shows that benchmark difficulty is linked to the image population and class definitions [13]. A six-class cropped benchmark and a ten-class dataset with more variable backgrounds do not impose the same recognition problem [13, 14]. The architecture may be unchanged while the measured outcome shifts substantially. The present study consequently preserves the label set and sample count when discussing its own results (Chapter 9). Chen and Liu develop MSANet using a SqueezeNet backbone, features from multiple layers, and an attention-based fusion strategy [143]. Their clean-data classification result reaches 99.72% [143]. Their experiments use an 80/20 split and repeated evaluations, with additional experiments involving noisy images and limited training data [143].

This work provides useful context for multiscale representation and small-data behavior. Its protocol also illustrates why the number of repetitions and the partitioning rule must accompany a headline accuracy [13]. Noise augmentation introduces a further question: whether repeated noisy versions remain tied to the same original image when partitions are created. That question matters in any augmentation study because a transformed copy retains information about its source [38]. The relevant reporting unit is the original image, together with the transformations derived from it [13]. A larger count of augmented files should not be interpreted automatically as a larger count of independent observations [13, 37].

Tuti and colleagues introduce an attention-guided multiscale residual framework and report 97.9% accuracy and macro-F1 on a six-class NEU experiment [140]. The paper discusses a dual-path encoder, entropy-related attention, early-exit inference, and energy-related optimization [140]. Its classification setting is relevant to the present task, while its efficiency claims require their own measurement conditions [13]. No energy reduction is attributed to the seven models in this book because the current experiment does not include energy measurements (Chapter 11). Taken together, these studies show that edge cues, multiscale features, and attention are frequently used to distinguish fine defect patterns [13, 15]. They also show that high classification accuracy is already reported on

some prepared benchmarks [13]. The research need therefore extends beyond increasing a headline value. More informative questions concern which images remain difficult, whether errors persist across models, and whether favorable results survive a change in source or acquisition conditions [13; 15].

Hybrid recognition and detection pipelines

Sabri and Gaceb compare image classifiers and a detector within a steel-monitoring framework [137]. Their Xception and ViT classifiers both achieve 100% accuracy on a test set with 72 images, while YOLOv8n is evaluated through detection metrics [137]. The distinction between these results is important. Perfect classification of a small cropped test population does not establish perfect localization over a larger surface, and detector mAP cannot be read as classification accuracy [13, 14].

The size of the test set is also part of the interpretation. A 72-image test set contains fewer opportunities to observe rare failures than a 450-image test set (Chapter 9). It may nevertheless be useful if it represents a clearly defined acquisition population [13]. The lesson is not that one score is invalid because its test set is smaller. The lesson is that claims should match the support and scope of the evaluation. Small-sample perfect accuracy should retain its denominator [147, 148].

Moon and colleagues construct a two-stage framework with class-agnostic localization followed by a hybrid classifier combining CNN and transformer information [24]. Their classification stage reports an F1-score of 99.7% on NEU-DET and 94.8% on GC10-DET [24]. Their methods describe training and evaluating models on both datasets [24]. This is valuable evidence of performance across two benchmark settings, but it should not be redescribed as a frozen model tested on a completely unseen external dataset unless the corresponding protocol establishes that condition [13].

The separation of localization and classification has a practical advantage: the classifier can concentrate on a region selected by the detector [24]. It also introduces error propagation. A missed or poorly cropped region can prevent the classifier from receiving the relevant information [13]. Evaluation of the classifier alone therefore describes a component, while

end-to-end evaluation describes the inspection system. The present experiment concerns the component task of image classification (Chapter 11).

Few-shot classification and prototype adaptation

Zhang and colleagues propose a forward-reverse rectification network for few-shot strip-steel classification [145]. The forward process classifies queries using support-set prototypes [145, 149]. A reverse training process uses correctly classified query samples to improve consistency between the support and query representations [145]. The method aims to reduce prototype shifts when support examples vary strongly within a category [145].Their FSC-20 protocol separates categories into training, validation, and testing groups and evaluates episodes under few-shot conditions [145].

A reported 5-way, 1-shot accuracy of 85.58% describes recognition among five classes with a single support example per class [145, 150]. This is a different estimand from training a ten-class classifier with a substantially larger fixed training subset. Comparing the two accuracies as though they measure the same problem would obscure the purpose of the few-shot experiment. Few-shot research is nevertheless useful for planning extensions to the present study. A production system may encounter a new defect category for which only a few reference images are available [86, 146]. That situation requires adaptation or a different training protocol [150]. The existing seven-model results do not establish performance on novel categories [28], but their representations could serve as starting points for such an experiment. A suitable study would define base and novel classes before training, and preserve the independence of support and query examples [150].

Detection through scale and feature interaction

Several detector studies address the mismatch between small defects and features computed at reduced spatial resolution [5, 13]. Ren and colleagues introduce CFL-YOLO with context fusion, a large-kernel attention-related module, and a modified loss [141]. Their experiments span NEU-DET, GC10-DET, and SteelDEF-7795, with reported mAP50 values of 81.0%, 79.3%, and 91.9% [141]. Those results support the study of multiscale

localization. They do not provide an image-classification baseline for the current experiment.

Liao and colleagues combine attention-based intra-scale processing, poly-kernel feature extraction, and generalized pyramid fusion [151]. Their reported detector results differ between NEU-DET and GC10-DET, again illustrating the role of the image population and annotation [13, 151]. Wei and colleagues develop AMFNet around branch interaction, dilated context, and spatial correlation [152]. Both papers motivate the explicit interaction of features computed at different depths or scales. Wang and colleagues introduce a cross-scale connection network with a gather-and-distribute structure, preprocessing, and coordinated feature fusion [153]. The reported mAP at an IoU threshold of 0.5 is 84.7% on NEU-DET [153].

This threshold is part of the meaning of the result. A detector measured at IoU 0.5 allows a different level of localization agreement from a metric averaged over stricter thresholds [14, 154, 155]. The threshold should therefore remain visible when detector studies are summarized.

Liu, Miao, and Zhou propose YOLO-CESL, which modifies downsampling and combines multiscale feature extraction with large separable kernel attention [156]. Xie and colleagues propose SDMS-YOLOv10 with changes to the loss, backbone, feature processing, and detection head [157]. Xu and colleagues introduce YOLO-DBL with dilation-wise residual processing, bidirectional feature fusion, and attention [158]. These approaches share an interest in preserving useful defect information while controlling model complexity [13, 141, 158]. The relevance to image classification is conceptual rather than a direct numerical comparison. A crop-level classifier can also lose a small structure during resizing or successive downsampling [5]. However, a localization head and its matching rules solve additional problems [154]. The seven-model benchmark should therefore be interpreted as a comparison of recognition pipelines, not as a replacement for a detector benchmark.

Annotation quality and irregular defect regions

Liu and colleagues propose ReForm-Net to reconstruct coarse rectangular annotations into patch-level supervision [144]. The method identifies low-

recall categories and refines the label structure for irregular defects, then combines this supervision with attention-guided feature fusion [144]. The reported mAP50 results are 85.2% on NEU-DET and 74.3% on GC10-DET [144]. When the reconstructed labels were used to train other detectors, their mAP50 also increased [144]. The study places annotation structure at the center of the detection problem rather than treating the labels as fixed and unquestionable. The same principle is useful for classification, although the annotation is different. An image-level label can be ambiguous if the crop contains more than one pattern, if the category definition is broad, or if the image does not display the distinguishing feature [2, 13, 14]. A shared mistake across several strong models should therefore prompt examination of the input and the annotation [159]. It should not prompt automatic relabeling based on model agreement. Models may share a bias or respond to the same misleading background cue [13, 160]. This distinction informs the error analysis later in the book (Chapter 14). The repeatedly misclassified Oil spot examples are identified by filename and frequency of error (Figure 15). Their existence is a measured result. Their physical origin and label correctness remain questions for image review and source annotation inspection. Preserving this separation prevents an error analysis from silently altering the benchmark after test results have been observed [38; 53].

Segmentation and class-guided spatial prediction

Zhou and colleagues examine few-shot segmentation using a pretrained variational autoencoder representation and latent Gaussian process regression [161]. The approach builds relationships between support and query information in latent space and incorporates contextual processing [161]. Its goal is to predict defect regions under limited supervision. Such a method addresses a spatial task that cannot be assessed by the hard-label accuracy used in this experiment [13, 14]. Shen and colleagues introduce CKLNet, in which class knowledge guides lightweight salient-object prediction [25].

The method includes stages for general region prediction, extraction of class information, and class-specific decoding [25]. This connection

between recognition and spatial prediction is relevant when different defect categories require different localization behavior. It also suggests why a classification backbone may be useful without being a complete inspection model [137]. Segmentation can provide an estimate of surface area or shape, but even an accurate mask is not automatically a calibrated physical measurement [88]. The conversion from pixels to dimensions requires imaging scale, geometry, and an appropriate ground truth [88, 162]. The classification benchmark does not estimate defect size, depth, or mechanical consequence (Chapter 11). These distinctions should remain explicit when the work is discussed with engineers who may need measurement rather than only recognition.

Scarce data and generated samples

Nam and Lee propose context-aware conditional diffusion for generating defect samples while maintaining consistency with surrounding surface information [163]. Their experiments involve several inspection datasets, including metal and fabric populations [163]. The relevant idea is that a generated defect should fit its background rather than appear as an isolated pasted pattern [15, 146]. This question is important when training examples are scarce or imbalanced [15, 146]. Song and colleagues develop FS-MSDANet for few-shot detection in wear-resistant steel, combining an improved CycleGAN-related augmentation strategy, multiscale attention, and feature-space constraints [164]. This setting links synthetic data generation with a specialized small-sample detector. It provides context for methods that seek more useful supervision from limited data, but it is distinct from the balanced supervised classification experiment described here (Chapter 9). The current benchmark does not claim an experiment with synthetic defects. A future augmentation study should train generators using only development data and keep every derivative linked to its original image [13, 38]. Evaluation should use genuine held-out images. Otherwise, a model may appear to generalize because the train and test sets contain transformations of the same underlying surface [13, 37]. Generated sample counts and independent image counts should be reported separately.

Efficiency and production data

Tan and colleagues propose LDR-YOLO to reduce detector complexity while retaining dynamic upsampling and **regional** attention [165]. Fu and colleagues develop a lightweight NanoDet-Plus-related model and report measurements on a Raspberry Pi 4B [166]. These papers show why a deployment claim needs a specified device and inference protocol [13]. A detector measured on a compact processor and a classifier measured on a workstation cannot be compared through model size alone [13, 97]. Singh and colleagues study defect prediction using industrial process data, imputation, feature encoding, and an ensemble framework [139]. Their work concerns tabular features and class imbalance, rather than the cropped image population of the present study [139]. Its relevance lies in the possibility of combining image observations with production variables [3]. Process parameters may provide information that resolves an image-level ambiguity, but that benefit would have to be measured in a multimodal experiment [3]. Efficiency and process integration concern different outcomes from crop-level recognition accuracy. Parameter counts characterize model size, while latency, energy, peak memory, and production throughput require measurement under a specified device and acquisition protocol [13, 97].

Table 1. Selected steel-inspection studies by learning task, data context, and methodological contribution.

Study	Task	Data context	Relevant contribution
[3] Zong et al.	Review	Steel rolling	Process control and the wider role of visual monitoring
[4] Zhang et al.	Review	Welding inspection	Connects imaging, defect formation, and physical assessment
[142] Wang et al.	Classification	NEU and GC10	Dual-path features and entropy-based attention fusion
[143] Chen and Liu	Classification	NEU	Multiscale attention with repeated classification evaluation
[137] Sabri and Gaceb	Hybrid inspection	NEU	Classifier and detector address different output requirements
[24] Moon et al.	Hybrid inspection	NEU and GC10	Classification, detection, and explanation within one framework

Study	Task	Data context	Relevant contribution
[140] Tuti et al.	Classification	NEU	Residual attention and conditional computation
[145] Zhang et al.	Few-shot classification	FSC20 and additional sets	Prototype rectification in episodic recognition
[144] Liu et al.	Detection	NEU and GC10	Label reconstruction and region-aware feature processing
[141] Ren et al.	Detection	NEU, GC10, SteelDEF	Context fusion, attention, and modified training loss
[163] Nam and Lee	Generation	Five inspection datasets	Conditional diffusion with defect and background consistency
[139] Singh et al.	Process-data prediction	Tata Steel process records	Feature selection and ensembles on tabular measurements
[25] Shen et al.	Salient-object detection	Strip-steel imagery	Class knowledge guides a compact pixel-level model
[164] Song et al.	Few-shot detection	Wear-resistant steel	Shape-aware generation and dynamic attention
[161] Zhou et al.	Few-shot segmentation	Strip-steel imagery	Latent Gaussian process with a pretrained variational encoder
[157] Xie et al.	Detection	NEU and additional sets	YOLOv10 adaptation for multiscale surface features
[151] Liao et al.	Detection	NEU and GC10	Attention and multiscale feature fusion
[152] Wei et al.	Detection	NEU and GC10	Interactions between multilevel feature branches
[153] Wang et al.	Detection	NEU	Cross-scale gathering and redistribution with gray equalization
[165] Tan et al.	Detection	NEU and GC10	Compact network with dynamic upsampling and regional attention
[156] Liu et al.	Detection	NEU, GC10, SSDD	YOLOv8 modifications evaluated across several defect datasets
[166] Fu et al.	Detection	NEU and GC10	Attention fusion and lightweight deployment study
[158] Xu et al.	Detection	NEU and GC10	YOLOv8n extensions for feature processing and fusion

What the literature establishes for this study

The reviewed studies support several concrete design concerns: fine contours, multiscale context, irregular regions, scarce labels, and limited computational resources [141, 144, 146, 158]. They also reveal differences in class sets, partition sizes, supervision, and metric definitions [13]. A useful benchmark must therefore preserve its complete experimental identity. Its model names, initialization sources, label mapping, preprocessing, checkpoint selection, and test sample list should be recorded together [13, 38, 167]. The benchmark evaluates seven image-recognition pipelines through three seeds, a common test population, class-specific discrimination, paired uncertainty, and probability quality (Chapters 11 and 12). This combination makes agreement between models and concentration of failures directly observable, providing a stronger basis for follow-up experiments than an aggregate accuracy ranking [40, 129].

9. SteelDefectX10 and Data Preparation

Source dataset and study subset

SteelDefectX is a multi-form vision-language resource whose published description includes 25 defect categories and several forms of textual supervision [2]. The subset used here is called SteelDefectX10 to identify the ten selected labels (Table 2). This name denotes the study subset rather than a separate official dataset release. The experiment uses image-level labels and ordinary image classifiers. Text descriptions and segmentation masks are not additional inputs to the seven models [2] (Chapter 11). The source repository is Zhaosxian/SteelDefectX on Hugging Face [168]. The snapshot fixes the source files and labels from which the study subset was prepared. The source combines material from established surface-defect collections, including NEU, GC10-DET, X-SDD, and S3D [2, 169-171]. Consequently, evaluating a trained model on one of those originating collections requires an overlap check before the collection is treated as external [13, 38]. The source-dataset relationship is an issue of provenance. A new directory or a different download URL does not make an image population independent of the development data [13].

Class selection and balanced sampling

The nominal sampling design selects 300 images per class and divides them into 210 training, 45 validation, and 45 test images (Table 2). This gives a target allocation of 2,100, 450, and 450 images. The target describes the sampling design. The evaluation records directly establish the realized test population of 450 images and 45 examples in each category. Final development-set counts should be retained in the exported data manifest when exact-duplicate resolution changes the available pool. Equal class counts simplify interpretation of the recognition task. Each category contributes the same number of test decisions, so overall accuracy is also the average class recall [172]. A classifier cannot achieve a high score simply by favoring the most common test category [173]. However, balance changes the population under study. Real production data may contain a large normal majority and unequal defect prevalence [13, 139]. The measured balanced accuracy is therefore not automatically an estimate

of production-line accuracy [13, 26]. Sampling also trades dataset size for comparability. Classes with more available images are reduced to the common allocation (Table 2). This decision prevents class size from dominating the experiment, but it leaves some source images unused. A separate study could examine training-set size or unequal sampling. The current comparison keeps the selected subset fixed across model families so that the data population is not another changing factor.

Table 2. Class mapping and partition design. Candidate counts are reported before duplicate handling. Training and validation are nominal targets; test supports are verified from prediction records.

Index	Class	Candidates	Train target	Val target	Verified test
0	Crazing	300	210	45	45
1	Dark scratches	334	210	45	45
2	Inclusion	795	210	45	45
3	Oil spot	498	210	45	45
4	Patches	300	210	45	45
5	Pitted surface	300	210	45	45
6	Punching	399	210	45	45
7	Rolled in scale	300	210	45	45
8	Silk spot	578	210	45	45
9	Welding line	500	210	45	45

Exact repetitions and visual dependence

The preparation workflow checks decoded image pixels as well as file-level hashes. This distinction matters because two files can store the same image under different compression or metadata. Identical decoded pixels represent repeated content even when the file bytes differ. A pixel-level hash provides a direct test for that form of repetition [174]. The intended resolution strategy retains only one representative of a repeated pixel group across the study partitions. If identical pixels have conflicting labels,

the group is excluded rather than arbitrarily assigned to one category. Additional candidates can be sampled from the same class and permitted source pool. If a class cannot supply the desired number of unique examples, the balanced allocation must be adjusted explicitly rather than replenished with duplicate copies or counted augmentations. Because the final development manifests are not available, the application of this strategy across partitions has been verified only for the test partition (see below and Table 6). Near-duplicate checks require more care. Perceptual hashes can identify candidates with similar low-frequency appearance, but their distances are not proof that two crops come from the same physical defect [175]. A visually similar pair may consist of genuinely distinct surfaces. Conversely, a small crop shift or localized alteration may evade a simple hash rule [174]. Candidate pairs therefore require review and provenance information where available.

The test partition contains 450 unique identifiers and 450 distinct decoded RGB pixel arrays, with no exact repeated image within that partition. A within-test pixel check does not establish independence from development images or from related crops of the same specimen [13, 37]. Source and acquisition groups remain necessary for a complete cross-partition assessment [38].

Partitioning and model selection

Training, validation, and testing serve different purposes [43, 59]. Training estimates model parameters. Validation selects checkpoints or development settings. Testing evaluates the chosen procedure on images excluded from those decisions [45, 52]. The same fixed test set is used for all models and seeds in this experiment, enabling a paired analysis of their predictions [40]. The preparation design preserves the distinction between the source repository's development and validation pools while creating a study validation and held-out test split [168]. The important experimental property is the final assignment of each selected image. The assignment should remain fixed when the training seed changes. Otherwise, seed variation would mix optimization randomness with differences in test composition [176]. Each evaluated pipeline uses a saved model checkpoint. Twenty training logs cover epochs 0 through 49; the

FasterNet-M seed 1 training log is unavailable, although its predictions are included (Table 6). Checkpoint-selection criteria and selected epochs are not fully specified, limiting exact training reproduction. Because the selection criterion cannot be confirmed for every run, the separation of checkpoint selection from the test set is a design intention that the available records do not fully verify [45].

Label mapping and sample identity

The exported mapping uses indices 0 through 9 in alphabetical directory order: Crazing, Dark scratches, Inclusion, Oil spot, Patches, Pitted surface, Punching, Rolled in scale, Silk spot, and Welding line (Table 2). The same mapping appears in all prediction metadata. This consistency is necessary for interpreting probability columns and confusion matrices. A change in class ordering can create a serious evaluation error without changing any model weights [177]. For example, a probability column associated with Oil spot during training may be interpreted as Patches after a different directory ordering. Preserving a class-to-index file and checking it during evaluation prevents this problem [167]. The hard-label and probability columns in these predictions agree with the mapping (Table 6). Sample identifiers contain a class directory and filename. They are useful keys for alignment across models. Before any paired comparison, records are sorted by these identifiers and checked for uniqueness. The analysis also verifies that each identifier has the same true label in every run (Chapter 11). Alignment by row position without an identifier check could compare predictions from different images and invalidate a paired result [40].

Dataset documentation and the overview figure

Figure 3 presents one image from each of the ten test folders, selected by the authors for visual clarity, together with a magnified display region and a short description of its visible cue. The images come from the test partition of SteelDefectX [2]. The descriptions follow the visual characteristics commonly reported for these defect categories [14, 21]. Cyan rectangles and arrows identify manually chosen display regions, not model predictions or ground-truth defect boundaries.

The crops help readers compare morphology; they do not add localization supervision to the classification task. Dataset documentation should also record acquisition, annotation, intended use, and known limitations, as advocated by datasheets for datasets [19].

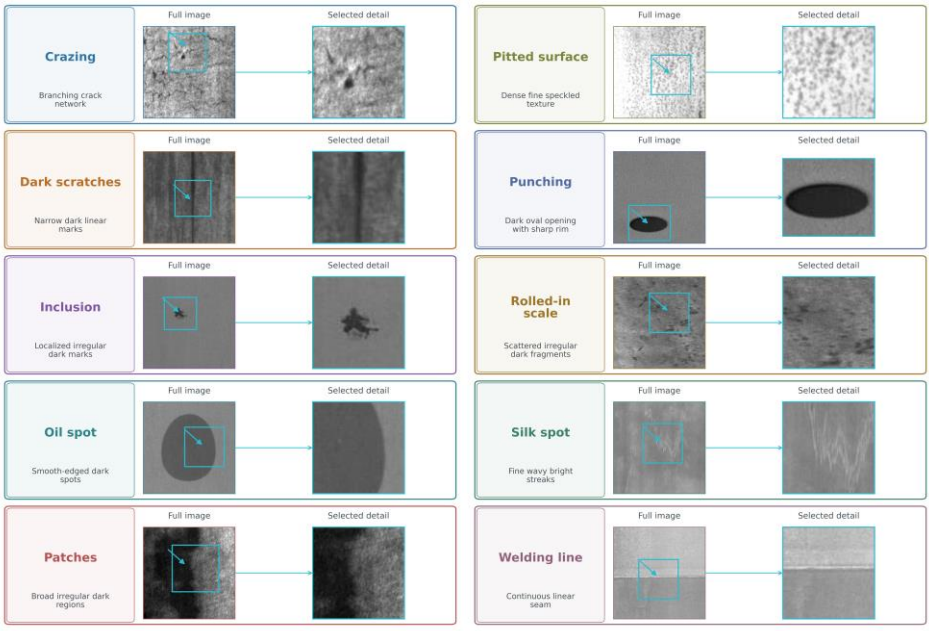


Figure 3. Representative test images from all ten SteelDefectX10 classes, with short appearance descriptions and magnified display regions. The cyan rectangles and arrows mark manually selected display regions; they are not ground-truth annotations or model predictions. Representative images are from the held-out SteelDefectX10 test partition.

10. The Seven Recognition Models

A common notation

Each studied model receives a 224 by 224 three-channel tensor and produces ten class logits (Chapter 11). Spatial feature maps are denoted by X, with C channels, height H, and width W. Conv denotes standard convolution, DW a depthwise convolution, P a pointwise projection [94], GAP global average pooling [178], and LN or BN the indicated normalization [73, 74]. Learned biases are explicit where they clarify a dense classifier; convolutional operators can include the biases permitted by the implementation. Softmax converts logits to the probability vector p after the classifier [42].

The architecture diagrams follow the implemented model definitions [179]. The equations describe the principal inference operations and compatible residual paths, with stochastic depth [180] omitted and scaling specified where relevant. Table 3 gives the pretrained variants and parameter counts of the ten-class models. Initialization and evaluation transformations are part of each compared pipeline (Chapter 11).

Table 3. Evaluated model identifiers, parameter counts with ten-class heads, and pretrained weight variants. Counts are millions of parameters.

Model identifier	Parameters M	Weight variant
convnext_base	87.5767	fb_in22k_ft_in1k
swin_base_patch4_window7_224	86.7535	ms_in22k_ft_in1k
caformer_m36	53.9229	sail_in1k
inception_next_base	83.6299	sail_in1k
fasternet_m	52.2533	in1k
vit_base_patch16_224	85.8063	augreg2_in21k_ft_in1k
tf_efficientnetv2_m	52.8712	in21k_ft_in1k

EfficientNetV2 M

EfficientNetV2 combines fused spatial processing in early stages with mobile inverted bottlenecks in later stages [110]. At the 224-pixel

resolution used here, its seven stage widths are 24, 48, 80, 160, 176, 304, and 512, with depths 3, 5, 5, 7, 14, 18, and 5. The feature resolutions are 112, 56, 28, 14, 14, 7, and 7. Early fused blocks avoid a separate depthwise operation; later blocks use depthwise filtering and squeeze-and-excitation [103, 110].

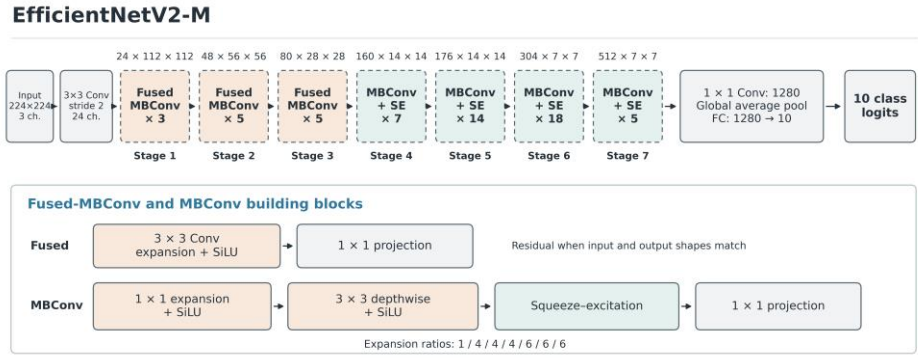


Figure 4. EfficientNetV2-M for ten-class recognition. Seven stages of fused and mobile inverted bottlenecks. The inset distinguishes fused convolution from the depthwise and squeeze-and-excitation path. The first stage has no separate projection because its expansion ratio is one. Tensor labels in the hierarchy use C × H × W at a 224 × 224 input. The diagrams omit training-time stochastic depth for clarity.

For an expanded fused block, a spatial convolution produces the expanded representation, followed by a pointwise projection [110]. The expansion ratio determines the intermediate channel width. SiLU is the activation used in these blocks [181]. Stage 1 has expansion one, so its fused operation reduces to a 3 by 3 convolution, BN, and SiLU without the separate pointwise projection [110]. For MBConv, expansion and spatial filtering are separate operations [96, 105].

The squeeze-and-excitation gate uses a pooled representation of the depthwise output. It applies a reduction projection followed by an expansion projection, and the resulting channel gate is broadcast across spatial locations [103]. Projection returns the block to its output width. A shortcut is added only when stride and channel dimensions permit it: the block input is added to the projected branch when the stride is one and the input and output channel counts agree; otherwise, the block output is the

projected branch alone [96]. The final 1 by 1 convolution produces 1280 channels before pooling. This order distinguishes EfficientNetV2's head from the pooled projection used by FasterNet [97]. The selected tf_efficientnetv2_m uses in21k_ft_in1k weights and approximately 52.87 million parameters after adaptation (Table 3). Its family-level progressive-learning strategy [110] is not asserted to be the fine-tuning schedule used in this experiment.

ConvNeXt Base

ConvNeXt modernizes a convolutional recognition hierarchy while retaining convolutional token mixing [111]. The Base variant has four stages with 128, 256, 512, and 1024 channels and depths 3, 3, 27, and 3 [111]. A stride-4 stem produces the first 56 by 56 representation. LayerNorm and stride-2 convolution connect later stages, ending at 7 by 7 spatial resolution.

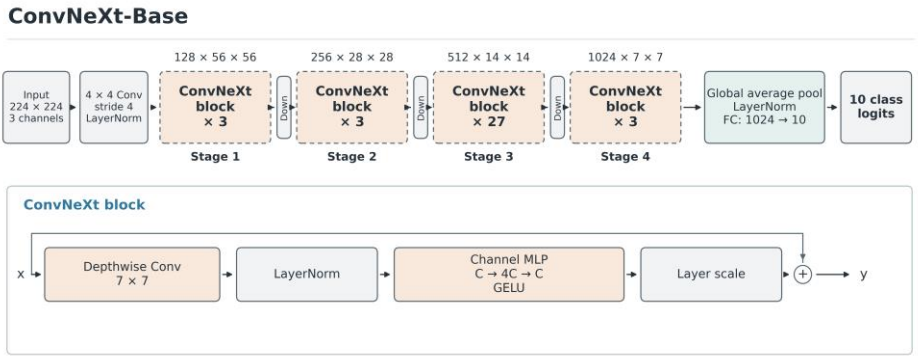


Figure 5. ConvNeXt-Base for ten-class recognition. Four stages of large-kernel depthwise residual blocks. The inset identifies normalization, channel expansion and projection, and layer scale. Tensor labels in the hierarchy use C × H × W at a 224 × 224 input. The diagrams omit training-time stochastic depth for clarity.

A depthwise 7 by 7 filter mixes local spatial information within each channel [111]. Because its weights are channel-specific, it differs from a dense spatial convolution that combines channels during filtering [94]. LayerNorm supplies the normalized representation for channel transformation [74, 111]. The implementation handles the relevant tensor layout so normalization operates over the intended channel dimension.

The channel MLP expands the channel dimension fourfold, applies GELU, and projects back to the original width [111, 182]. Because it acts independently at each spatial location, this dense transformation is equivalent to a pair of 1 by 1 convolutions [178]. Layer scale multiplies the output of the channel MLP by a learned per-channel factor before the residual addition [111, 183]. Training-time stochastic depth is omitted from this inference description [180]. In the selected Base definition, global average pooling precedes the head normalization, and a final linear layer produces ten logits from the 1024-dimensional pooled representation, which softmax converts to class probabilities. The tested model uses ImageNet-22k pretraining with ImageNet-1k fine-tuning and has approximately 87.58 million parameters (Table 3). Its leading observed accuracy belongs to this complete pipeline; it is not an isolated proof that large-kernel convolution dominates attention.

InceptionNeXt Base

InceptionNeXt replaces one large depthwise spatial operation with parallel channel branches [112]. The Base variant retains the widths 128, 256, 512, and 1024 and depths 3, 3, 27, and 3 [112]. Its stem and downsampling use BatchNorm with convolution. The parallel token mixer contains an identity branch, a small square filter, and two orthogonal band filters [112].

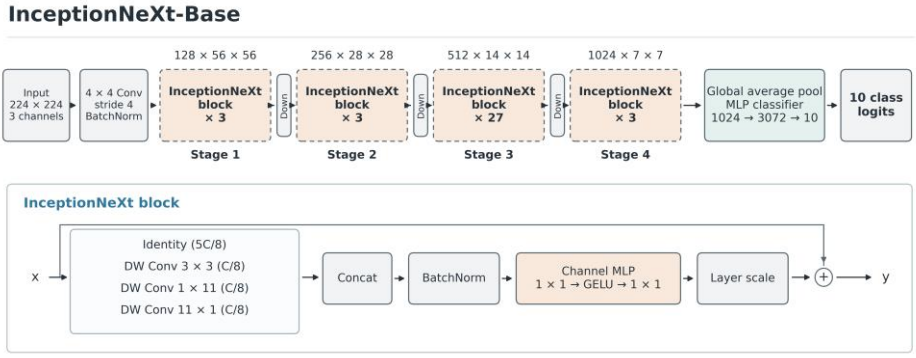


Figure 6. InceptionNeXt-Base for ten-class recognition. Four stages with parallel identity and depthwise spatial branches. The inset identifies channel splitting, orthogonal band kernels, channel mixing, and residual addition. Tensor labels in the hierarchy use C × H × W at a 224 × 224 input. The diagrams omit training-time stochastic depth for clarity.

The token mixer divides the channels into four groups [112]. The identity group retains five eighths of the channels, and each filtered group receives one eighth; these ratios refer to the selected implementation. The first filtered group is processed by a 3 by 3 depthwise convolution, and the other two by horizontal 1 by 11 and vertical 11 by 1 depthwise band kernels; the identity group is passed through unchanged [112]. The outputs of the four groups are concatenated in the channel dimension. Horizontal and vertical band kernels provide different spatial filtering paths without applying every kernel to every channel, following the multi-branch principle of Inception [98].

The concatenated features are batch-normalized [73] and then enter a pointwise channel MLP consisting of an expansion projection, GELU [182], and a return projection. The expansion is four in stages 1 to 3 and three in stage 4, rather than four in every stage. A learned layer-scale vector [183] and residual addition complete the block. These operations preserve the shape within an ordinary stage. The classifier is an MLP head rather than one linear layer directly after pooling. After global average pooling, a hidden layer of dimension 3072 for the 1024-channel final representation is followed by GELU and layer normalization, and a final linear layer produces the ten logits [112]. The selected model has approximately 83.63 million parameters (Table 3). Band filtering offers a plausible representation for elongated patterns [5], but the current comparison does not isolate that mechanism or measure its speed on the training device [97].

FasterNet M

FasterNet uses partial convolution to reduce the amount of spatial filtering while retaining ordinary channel mixing [97]. The M variant has widths 144, 288, 576, and 1152 and depths 3, 4, 18, and 3 [97]. A convolutional embedding and later patch-merging convolutions reduce spatial resolution. BatchNorm and ReLU are used in its pointwise block transformations.

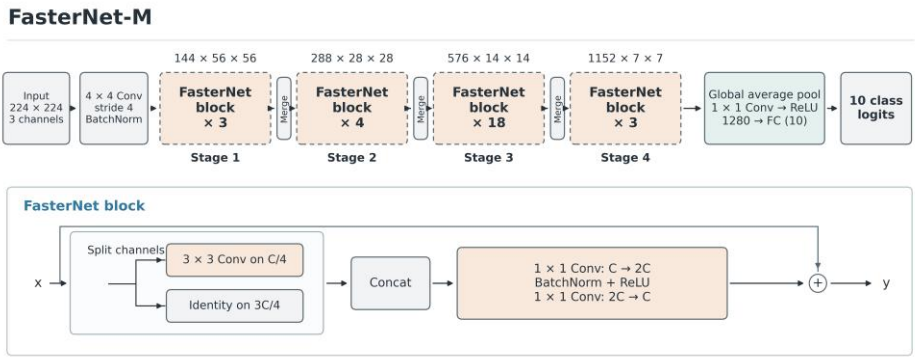


Figure 7. FasterNet-M for ten-class recognition. Four stages of partial-convolution blocks. One quarter of the channels receive standard spatial convolution, followed by pointwise channel mixing and a residual connection. Tensor labels in the hierarchy use C × H × W at a 224 × 224 input. The diagrams omit training-time stochastic depth for clarity.

The spatial operation begins by separating a filtered channel subset from a bypass subset; the selected division factor is four, so one quarter of the channels is filtered and the remaining three quarters bypass the spatial operation [97]. A standard 3 by 3 convolution acts on the filtered subset, while the bypass subset passes unchanged, and the two subsets are then concatenated. This is partial convolution, not a depthwise convolution over all channels [97].

Pointwise mixing doubles the channel dimension, applies BN and ReLU, and projects back to the original width. The bypassed channels can consequently interact with the filtered channels in the MLP [97]. The stage block adds the learned transformation to its input. Layer-scale parameters are disabled in the selected default M configuration and stochastic depth is omitted from this inference description. Global average pooling occurs before the 1 by 1 projection to 1280 features. ReLU and the ten-output linear classifier follow that projection, and softmax converts the classifier outputs to class probabilities. The selected model, pretrained on ImageNet-1k, has approximately 52.25 million parameters (Table 3). Its partial-convolution design motivates execution measurements, but FLOPs, parameter count, and observed accuracy cannot substitute for measured

latency [13, 97]. All three seed evaluations remain in the results, including the run with the missing training log (Table 6).

Vision Transformer Base with 16 pixel patches

ViT-Base/16 uses a 16 by 16 patch projection, a 768-dimensional representation, twelve encoder blocks, and twelve attention heads [113]. At the evaluated resolution, there are 196 patch tokens and one learned class token [113]. The sequence resolution remains fixed throughout the encoder instead of being reduced between four spatial stages [113, 115].

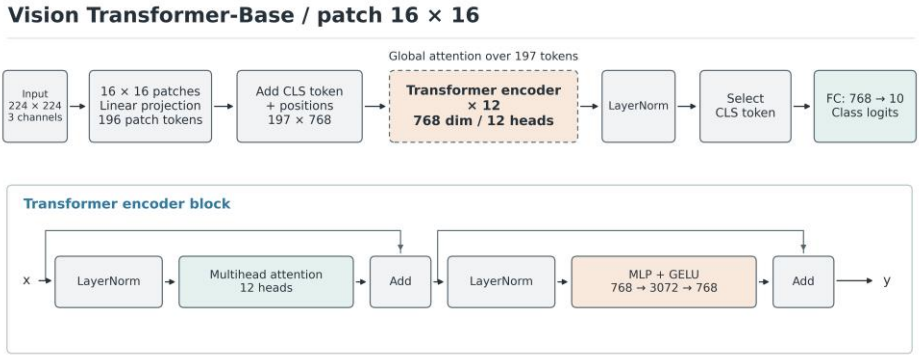


Figure 8. ViT-Base/16 for ten-class recognition. Patch projection and a twelve-block global-attention encoder. A class token aggregates the image sequence before the ten-class head; the inset shows both normalized residual sublayers. Tensor labels in the hierarchy use C × H × W at a 224 × 224 input. The diagrams omit training-time stochastic depth for clarity.

Each flattened patch is projected linearly to the 768-dimensional representation. The class token is prepended, and learned position embeddings are added to the complete sequence of 197 tokens [113]. For twelve heads, the key dimension is 64. Attention acts globally over the sequence, and the head outputs are concatenated and projected [113, 114]. The first normalized residual sublayer supplies the attention update; its layer normalization precedes the learned attention projections [113]. The second sublayer uses a GELU channel MLP [182] with hidden dimension 3072. It transforms token features while the residual path retains the previous representation. After twelve blocks, the normalized class-token representation enters the ten-class head, and softmax converts the head outputs to class probabilities [113]. Its interpretation is a learned aggregate of the image

sequence rather than a physically annotated defect region. The selected model, pretrained on ImageNet-21k and fine-tuned on ImageNet-1k, has approximately 85.81 million parameters (Table 3). Patch size and input transformation define the spatial detail supplied to the network. Their effects are relevant for future ablations, while the present results concern this one patch size and evaluation pipeline.

Swin Base

Swin-Base builds a hierarchy through local-window attention and patch merging [115]. The evaluated variant has a stride-4 patch embedding, widths 128, 256, 512, and 1024, depths 2, 2, 18, and 2, and head counts 4, 8, 16, and 32 [115]. Windows contain 7 by 7 spatial tokens. Alternating blocks change the window arrangement to exchange information across earlier boundaries [115].

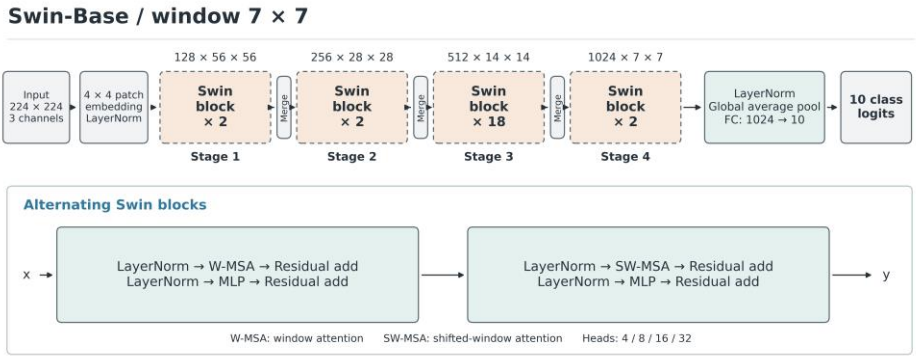


Figure 9. Swin-Base for ten-class recognition. Hierarchical window attention with patch merging. The inset shows the alternating ordinary and shifted-window blocks. Shifts are disabled at the final feature resolution. Tensor labels in the hierarchy use C × H × W at a 224 × 224 input. The diagrams omit training-time stochastic depth for clarity.

The first normalized patch embedding produces a 56 by 56 grid of 128-dimensional tokens. Unlike ViT's class-token sequence, this hierarchy is aggregated spatially at its output [115]. Within a window, attention includes a learned relative-position bias. A mask prevents associations across incompatible shifted-window regions; for an ordinary window block it is absent [115]. Each block contains attention and channel-MLP residual sublayers, each preceded by layer normalization. The attention

sublayer uses regular or shifted windows according to the block's window arrangement [115]. Patch merging concatenates a two by two neighborhood. Normalizing the concatenated channels and applying a learned projection doubles the channel count at half the height and width [115]. The final feature map is normalized before global average pooling. The linear ten-class head operates on its 1024-dimensional pooled representation. At the final 7 by 7 spatial resolution, a window spans the complete map and shifts are disabled. The selected model, pretrained on ImageNet-22k and fine-tuned on ImageNet-1k, has approximately 86.75 million parameters (Table 3). Its probability-based results are considered separately from the small hard-label accuracy difference relative to ConvNeXt.

CAFormer M36

CAFormer places separable convolution in its first two stages and global attention in its last two stages within a MetaFormer hierarchy [136]. M36 has depths 3, 12, 18, and 3 and widths 96, 192, 384, and 576 [136]. A stride-4 stem is followed by normalized convolutional downsampling between stages [136]. Attention consequently acts at 14 by 14 and 7 by 7 feature resolutions.

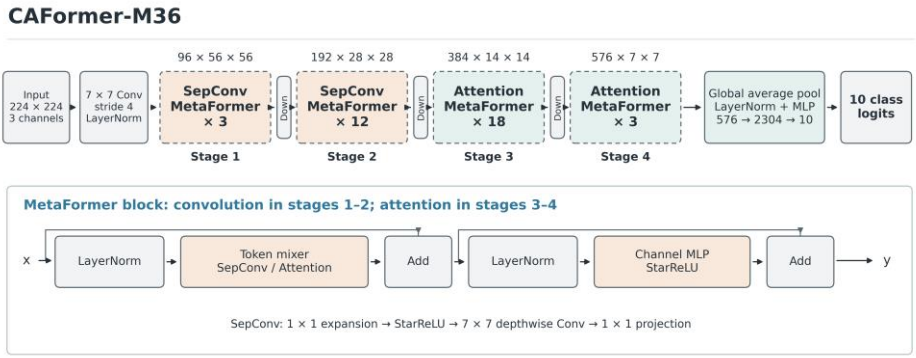


Figure 10. CAFormer-M36 for ten-class recognition. A MetaFormer hierarchy with separable convolution in stages 1 and 2 and global attention in stages 3 and 4. The inset identifies token mixing and the separate channel MLP. Tensor labels in the hierarchy use C × H × W at a 224 × 224 input. The diagrams omit training-time stochastic depth for clarity.

The early-stage token mixer expands channels, applies StarReLU, filters spatially with a 7 by 7 depthwise convolution, and projects back. Its expansion ratio is two in the selected separable-convolution mixer [96, 136]. StarReLU uses a squared rectifier with learned scale and bias [136]. Those activation parameters differ from the layer-scale vector used in ConvNeXt and InceptionNeXt [183]. Later-stage token mixing uses ordinary multihead self-attention. Each head applies scaled dot-product attention before the outputs are combined and projected [114]. The mixer and a StarReLU channel MLP occupy separate normalized residual sublayers. In the attention stages, the residual path is additionally scaled by learned parameters [136]. The pooled representation is normalized before the classifier MLP. Its hidden dimension is 2304, and its activation is squared ReLU, followed by another head normalization [136, 184]. This head activation differs from StarReLU in the backbone. The selected model, trained on ImageNet-1k, has approximately 53.92 million parameters (Table 3). Its convolution-attention transition describes a hybrid mechanism within one network. The experiment measures the complete model rather than an ablation of that transition.

Interpreting the comparison

The models differ in spatial mixing, channel mixing, normalization, stage depth, classifier design, and initialization. The diagrams connect each model's spatial processing, channel transformation, and classifier with its complete hierarchy. This representation alone does not determine the observed accuracy [13, 111]. A controlled mechanism study would need matched training and initialization, whereas the present experiment compares the specified pretrained pipelines under a shared ten-class task [8, 13]. The comparison measures recognition, variation across repeated fits, shared residual errors, and predictive confidence. These outcomes distinguish practical pipeline behavior without assigning an observed difference to one untested architectural component.

11. Experimental Protocol and Reproducibility

Comparative design

Seven architectures are evaluated with three training seeds, giving 21 fitted pipelines. All runs use the same 450 test images, ten classes, and class-index mapping. Predictions are aligned by image identifier before paired analysis. Each run supplies a hard label and ten class probabilities; the maximum-probability class agrees with the hard decision. Accuracy, macro-precision, macro-recall, macro-F1, macro-AUROC, macro-AP, and multiclass MCC are independently recomputed, with agreement to within 0.000001 of the reported scores [26, 172, 185]. The comparison uses pretrained weights rather than training the architectures from random initialization. The variants listed in Table 3 have different pretraining histories. Model-specific cropping and normalization also differ. The experimental unit is consequently the complete initialization, transformation, and network pipeline, not the architecture alone.

Training and computing environment

The computing platform comprises an NVIDIA GeForce RTX 5080 GPU with 16 GB of device memory, an Intel Core i9-14900K processor, and 128 GB of system RAM, running Ubuntu with a conda-managed Python environment. The evaluation metadata identify timm version 1.0.29. Hardware specifications describe the platform; the experiment does not measure latency, throughput, peak memory, or energy consumption. Seeds are 0, 1, and 2. Twenty available training logs show epochs indexed from 0 through 49, establishing 50 executed epochs for those runs. The FasterNet-M seed 1 training log is unavailable; its held-out predictions remain included in every three-seed summary. The evaluated checkpoint may precede the final epoch. The batch size is 16 and the classifier has ten outputs. Optimization uses SGD with momentum 0.9, Nesterov acceleration, and weight decay 0.00002 [70]. The initial effective learning rate is 0.00625, corresponding to a base-rate argument of 0.1 scaled to batch size 16 [186]. Training and evaluation use float32 with automatic mixed precision disabled.

The training criterion is Label Smoothing Cross Entropy [187]. Its coefficient, complete augmentation policy, learning-rate schedule, and checkpoint-selection options are not fully specified in the available logs. These omissions limit exact reproduction and causal interpretation of the confidence results [188]; defaults from another software version are not assumed.

Table 4. Training and evaluation settings. Fifty executed epochs are documented for 20 runs; the FasterNet-M seed 1 training log is unavailable.

Item	Setting
GPU	NVIDIA GeForce RTX 5080; 16 GB
Processor	Intel Core i9-14900K
System memory	128 GB RAM
Operating system	Ubuntu
Architectures and seeds	Seven architectures; seeds 0, 1, and 2; 21 evaluations
Training epochs	50 in all 20 available training logs
Batch size	16
Classes	10
Initialization	Pretrained, model-specific variants
Optimizer	SGD; momentum 0.9; Nesterov enabled
Weight decay	0.00002
Effective learning rate	0.00625 printed in optimizer; scaled base rate 0.1
Loss criterion	LabelSmoothingCrossEntropy; coefficient unspecified
Precision	Float32; AMP disabled in training and evaluation
Evaluation inputs	224 × 224; bicubic interpolation
Library version	timm 1.0.29 in prediction metadata
Unspecified training details	Full augmentation and scheduler; smoothing coefficient; exact selection options

Evaluation transformations

All inputs are resized to 224 × 224 using bicubic interpolation [189]. Crop fraction, crop mode, and normalization follow the selected pretrained pipeline's evaluation configuration. EfficientNetV2-M uses a crop fraction of 1.0 with squash resizing and channel mean and standard deviation of 0.5. ConvNeXt-Base uses a fraction of 1.0 with center cropping and ImageNet normalization. InceptionNeXt-Base uses 0.95 with center cropping and ImageNet normalization. FasterNet-M and Swin-Base use 0.9 with center cropping and ImageNet normalization. ViT-Base/16 uses 0.9 with center cropping and 0.5 normalization. CAFormer-M36 uses 1.0 with center cropping and ImageNet normalization. ImageNet normalization uses channel means (0.485, 0.456, 0.406) and standard deviations (0.229, 0.224, 0.225). Squash resizing and center cropping can preserve different portions and proportions of a surface, so a common final resolution does not imply identical visual input [190]. Table 5 specifies the transformations used in the comparison.

Table 5. Model-specific evaluation preprocessing. ImageNet normalization denotes mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225).

Pipeline	Crop fraction	Mode	Normalization
EfficientNetV2-M	1.00	Squash	Mean/std 0.5
ConvNeXt-Base	1.00	Center	ImageNet
InceptionNeXt-Base	0.95	Center	ImageNet
FasterNet-M	0.90	Center	ImageNet
ViT-Base/16	0.90	Center	Mean/std 0.5
Swin-Base	0.90	Center	ImageNet
CAFormer-M36	1.00	Center	ImageNet

Model selection and test interpretation

Training and validation support fitting, hyperparameter choice, and checkpoint selection. The held-out test assesses the resulting procedure [43]. Mixing selection and assessment introduces optimistic bias [45, 52]. Once a test result guides a revised configuration, an independent test is needed to assess that revised pipeline [38, 191]. All three seeds receive equal weight in the reported model means. No lower-accuracy run is excluded. The precise selected epoch, checkpoint criterion, and use of exponential moving averages are not established for every run [192]. The comparison therefore supports the reported predictions and their paired interpretation, while leaving those upstream selection details unresolved.

Randomness and experimental replication

A training seed influences classification-head initialization, image ordering, augmentation, and stochastic network operations [10, 11]. It need not produce identical random draws in different architectures. Equal numerical seed labels organize repeated fits; pairing arises from the same test images, not from an assumption of identical training trajectories [40]. The sample standard deviation describes variation across three fits on a fixed partition. It does not represent three independent datasets or production sites [40]. Zero observed accuracy variation can arise when different fitted networks make the same number of hard decisions correctly; it does not imply identical probability vectors or zero uncertainty on new images. Exact reproduction requires the random-number generator settings, data-loader worker initialization, deterministic-operation policy, software versions, and complete training command [11, 167]. These settings complement source manifests and checkpoint identifiers rather than replacing them.

Reproducibility and provenance

Data provenance connects each prepared image to its source, class, partition, pixel hash, and acquisition group where known [19].

Training provenance identifies the weight variant, configuration, code revision, and selected checkpoint. Evaluation provenance identifies transformations, ordered test images, probabilities, and metric definitions.

Structured reporting helps connect these components to intended use and evaluation scope [20]. The evaluation has 21 aligned prediction sets with consistent labels and verified metric calculations. The test images also pass a within-partition exact-pixel check. Final development manifests and acquisition groups are needed for cross-partition independence, while complete fitting configurations are needed for exact training reproduction. Table 6 summarizes the available experimental information and the remaining limits.

Table 6. Dataset provenance and reproducibility of the experiment.

Component	Experimental information
Source	Zhaosxian/SteelDefectX on Hugging Face
Source revision	82574f846982fd37a9e5f9dea96021c8d9704c39
Test sample count	450 distinct identifiers; 45 per class
Prediction sets	21; same sample identifiers, true labels, and mapping
Metric verification	Seven metrics reproduce to within 0.000001
Training logs	20 complete epoch logs; FasterNet seed 1 absent
Source and partition independence	Final manifest and source/acquisition groups required
Nominal allocation	210 train / 45 validation / 45 test per class
Probability normalization	Rows normalized before added probability analyses; argmax unchanged
Bootstrap	10,000 class-stratified paired image replicates; seed 20261009
Exact paired testing	63 two-sided McNemar tests; one global Holm family
Calibration	15 equal-width confidence bins; no fitted calibration
Industrial evaluation	Not included; no normal class in current label set

12. Evaluation Measures and Statistical Interpretation

Hard decisions and the confusion matrix

For each image, the predicted label is the class with the largest probability. Accuracy is the fraction of predictions equal to the reference label. A confusion matrix organizes the same decisions by true class and predicted class: a diagonal entry counts correct predictions, and an off-diagonal entry identifies a particular confusion [26, 172]. The direction matters. Oil spot predicted as Inclusion is different from Inclusion predicted as Oil spot, especially if the engineering consequences differ. Per-class recall is the fraction of true examples of that class that receive the correct label. Per-class precision is the fraction of predictions assigned to that class that are correct. Their harmonic mean gives the class F1 score [172]. Macro averages give each of the ten classes equal weight. Macro-F1 is computed by averaging the ten class F1 scores, rather than by inserting macro-precision and macro-recall into one harmonic-mean calculation [51, 193].

The test set contains 450 images in ten classes. The balanced test design creates a useful identity. Each class contributes 45 images, so averaging the ten class recalls produces the same value as overall accuracy. Balanced accuracy, which is also the mean class recall in this setting, is therefore numerically equal to accuracy [194]. Reporting all three as independent improvements would exaggerate the amount of evidence. Their equality is a property of the class supports, not a universal property of classifiers. The smallest change in a single run’s overall accuracy is one divided by 450, or approximately 0.2222 percentage points. The smallest recall change within a class is one divided by 45, or approximately 2.2222 percentage points. These scales help interpret the apparent precision of a table. Several decimal places can preserve calculations, but they cannot turn one extra correct image into a large body of independent evidence.

Multiclass correlation

The multiclass generalization of correlation measures provides the basis for the reported coefficient [185]. The Matthews correlation coefficient

summarizes association between the true and predicted labels using all entries of the multiclass confusion matrix [195, 196]. It complements accuracy by considering the distribution of predictions and references [197]. In this experiment it is close to one for all models, consistent with the very small number of errors. It does not identify the physical severity of those errors or eliminate the need for class-specific analysis. Every test class is represented among the predictions of each model, and all macro measures are well-defined. Precision and F1 use the same class conventions in every run, permitting direct comparison across the balanced label set.

Ranking scores

ROC analysis characterizes score-based discrimination across decision thresholds [26]. Macro-AUROC is calculated in a one-versus-rest formulation. For each class, its probability is used to rank positive examples against all other examples, and the ten class AUROCs are averaged [26, 51]. This assesses score ordering over possible thresholds rather than the single maximum-probability decision. A model can give a high rank to the correct class while still assigning a slightly larger probability to another class on a few images.

Precision-recall analysis is sensitive to positive-class prevalence and complements ROC analysis [27]. Macro average precision averages ten one-versus-rest average-precision scores, each calculated by weighting precision by successive recall increments. This definition differs from trapezoidal integration of a precision-recall curve; the reported metric is therefore denoted macro-AP throughout [51, 198, 199]. Each one-versus-rest task has 45 positives and 405 negatives. The balanced ten-class design therefore yields a positive prevalence of 0.1 for each class-specific ranking problem. A different class prevalence can change precision-related behavior even when the score distributions are similar [26, 27]. High AUROC and AP on this benchmark should not be interpreted as established screening performance at the prevalence of a production line [13].

Probability quality

Negative log-likelihood evaluates the probability assigned to the true class, using the negative natural logarithm and averaging over images. It penalizes confident mistakes strongly and rewards assigning high probability to correct labels [34, 200]. The Brier score originates in the evaluation of probabilistic forecasts [128]. The multiclass Brier score averages the sum of squared differences between the ten predicted probabilities and the one-hot reference vector. The sum is taken over classes without dividing by ten. Under this convention the score has a range from zero to two, and lower is better [128]. For calibration analysis, confidence is the largest predicted probability. The 15 equal-width confidence bins divide the interval from zero to one. In each occupied bin, mean confidence is compared with empirical accuracy, and the absolute gap is weighted by the bin's fraction of observations. The weighted sum is the expected calibration error under this particular binning scheme [34, 201]. Guo and colleagues provide an influential discussion of calibration and post-processing approaches [34].

The calculation uses the exported probabilities after row normalization to remove small serialization deviations from a sum of one. Normalization does not change the maximum-probability class. The analysis applies no fitted calibrator, temperature adjustment, or test-driven threshold selection. Reliability diagrams describe the probabilities produced by the existing pipelines; they do not represent a newly trained calibration model [202, 203]. Calibration error depends on binning and sample size. With only 450 observations, some confidence bins contain few or no images. A small ECE difference may therefore be fragile, and empty bins supply no evidence [204, 205]. NLL and Brier score avoid that particular binning choice but describe different aspects of probability quality [200]. Using all three is more informative than treating one calibration statistic as a complete reliability certificate.

Seed means and standard deviations

The model-level mean is the arithmetic average of the three seed values. The sample standard deviation uses two degrees of freedom. It describes the spread of the observed runs; it is not the standard error of independent

test images and does not directly provide a confidence interval for deployment performance [40, 206]. Accuracy and macro-F1 are reported as percentages, with their standard deviations expressed in percentage points. A conventional Student t interval around a three-seed mean can extend above 100% when accuracy is near the upper bound.

Such an endpoint reflects the use of an unbounded approximation with few observations, not a feasible accuracy value [207]. The main comparison therefore emphasizes mean and standard deviation. The image-resampling analysis provides a bounded, explicitly conditional interval with a different interpretation [208]. Three seeds are also too few to support strong claims about the full distribution of training outcomes [11, 176]. FasterNet-M and ViT-Base/16 each include one lower-accuracy run, which makes their sample spreads relatively large. Additional prespecified seeds would clarify this pattern. Selecting extra seeds only until a desirable mean is obtained would undermine the purpose of repeated training [209].

Paired image bootstrap

Bootstrap resampling estimates a sampling distribution under the stated resampling design [208]. The bootstrap analysis retains each fitted model and each of its three seeds. Within a model, correctness is averaged across seeds for each image. The procedure then resamples 45 images with replacement from each of the ten true classes [210]. Identical sampled indices are used for every model, preserving their image-level pairing. Ten thousand replicates are generated with a fixed analysis seed of 20261009, and the 2.5th and 97.5th percentiles define a 95% percentile interval [211].

This design estimates uncertainty associated with resampling images under a fixed balanced class composition. It is conditional on the three fitted runs and the observed test population. It does not estimate new training-seed variability, source-domain change, or acquisition-group dependence. If related images belong to the same physical specimen or production batch, independent image resampling can understate uncertainty; a group bootstrap would then be preferable [210, 212]. The paired procedure also generates intervals for differences in mean accuracy. For ConvNeXt minus Swin, the point difference is about 0.296 percentage points and the interval extends from approximately minus 0.074 to plus

0.741 percentage points. It contains zero. Intervals for several other differences exclude zero under this conditional procedure, but they are exploratory and unadjusted [213]. The leading model was identified from the same observed results, so the intervals are not a prespecified confirmatory test of a universal winner [214].

Exact paired tests and multiplicity

Paired correctness comparisons originate in the analysis of correlated proportions [39]. For each model pair and matching seed, the exact McNemar test uses images on which one model is correct and the other is wrong [40. If both are correct or both are wrong, that image does not contribute to the discordant count. Under the null of symmetric discordance, the two discordant directions follow an equal-probability binomial calculation. A two-sided exact probability is used because the error counts are small [215].

There are 21 distinct architecture pairs and three seed-specific tests per pair, giving 63 tests. A Holm adjustment is applied across this entire family [213]. The smallest unadjusted probability is 0.001953125; the smallest adjusted probability is 0.123046875. No comparison has an adjusted probability below 0.05. The dependence of the tests does not invalidate Holm's family-wise control, although the underlying image-independence assumption remains important [213]. This result does not establish that all models are equivalent. A failure to reject a difference is not an equivalence test, particularly with few discordant images [216, 217].

It indicates that these data do not support a family-wide significant difference under the specified exact testing procedure. Demšar's work concerns comparisons across multiple datasets [218]. Classifier-comparison research also emphasizes that a test must match its experimental replication structure [40]. The three seeds here are not treated as three datasets, and no multi-dataset ranking test is used to create an artificial replication level.

The bootstrap and McNemar analyses address different summaries and use different multiplicity treatment. Their outcomes need not match. Reporting their conditions is more useful than choosing only the method that favors a preferred model. Both reinforce the practical conclusion that a descriptive ranking should be accompanied by uncertainty and a careful account of what constitutes an independent observation.

13 Comparative Results

Overall recognition performance

All seven models achieve high test accuracy. ConvNeXt-Base leads with 99.48% mean accuracy and 99.48% macro-F1. Swin-Base follows at 99.19%, and CAFormer-M36 reaches 98.89%. InceptionNeXt-Base, FasterNet-M, ViT-Base/16, and EfficientNetV2-M follow with mean accuracies of 98.67%, 98.59%, 98.52%, and 98.37%, respectively. These values describe the balanced ten-class test and all three seeds rather than a selected best run.

Table 7. Main held-out results across three seeds. Accuracy and macro-F1 are mean ± sample SD in percent and percentage points, respectively. AUROC and macro-AP are mean values; AP is average precision.

Model	Accuracy %	Macro F1 %	AUROC	Macro AP
ConvNeXt-Base	99.48 ± 0.13	99.48 ± 0.13	0.99911	0.99630
Swin-Base	99.19 ± 0.13	99.19 ± 0.13	0.99985	0.99878
CAFormer-M36	98.89 ± 0.00	98.89 ± 0.00	0.99990	0.99917
InceptionNeXt-Base	98.67 ± 0.22	98.66 ± 0.22	0.99965	0.99780
FasterNet-M	98.59 ± 1.10	98.58 ± 1.11	0.99967	0.99783
ViT-Base/16	98.52 ± 1.03	98.52 ± 1.03	0.99974	0.99792
EfficientNetV2-M	98.37 ± 0.26	98.36 ± 0.26	0.99964	0.99627

The numerical range from the first to the last model is about 1.11 percentage points. On a test set of 450 images this corresponds to five additional correct classifications per run on average. The leading pair differs by only about 1.33 images per run. This scale is central to interpretation: the ranking can be stated clearly, while the small number of decisions behind it remains visible.

The macro-precision and macro-F1 values are close to accuracy, indicating that the small residual error distribution does not create a major aggregate precision imbalance. The closeness does not imply that every class

performs equally. Because several classes have no observed errors, performance loss is concentrated in the remaining categories. A high macro average can coexist with materially lower recall for one class [13]. These results describe a prepared, balanced population of known categories. They should be distinguished from inspection under naturally occurring prevalence [27]. A higher numerical score than another publication does not establish an improvement unless the data, partitions, task, and selection procedure are comparable [13, 38]. The most defensible use of this ranking is to identify candidate pipelines for independent testing and examine their errors on the shared images.

Variation across the three seeds

ConvNeXt-Base and Swin-Base each have an accuracy standard deviation of approximately 0.13 percentage points. Their seed-specific error counts remain small: three, two, and two for ConvNeXt, and three, four, and four for Swin.

InceptionNeXt has a standard deviation of approximately 0.22 percentage points, while EfficientNetV2 has approximately 0.26. These models display a relatively narrow spread within the three observed runs. FasterNet-M and ViT-Base/16 have larger spreads, approximately 1.10 and 1.03 percentage points. FasterNet's seed 1 makes twelve errors, compared with three and four for its other seeds. ViT's seed 0 makes twelve errors, compared with four in both other runs. Figure 11 shows the individual values alongside the mean and sample standard deviation, making the influence of these runs visible.

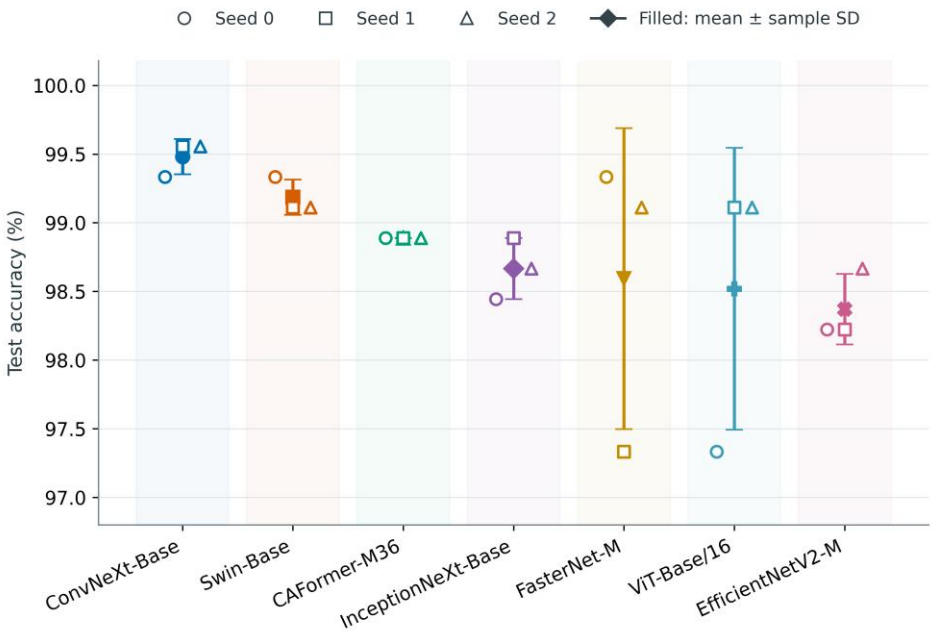


Figure 11. Test accuracy across three training seeds. Open circles, squares, and triangles identify seeds 0, 1, and 2. Filled model markers and error bars show the mean and sample standard deviation; colors distinguish pipelines. All runs use the same 450 test images.

CAFormer-M36 has five errors in each run and consequently zero observed accuracy standard deviation. Its probability vectors, macro-precision, and MCC values differ. ConvNeXt seeds 1 and 2 have identical hard decisions but different probabilities. These checks distinguish repeated decision outcomes from copied files. They also illustrate the coarseness of accuracy: different fitted functions can have the same count of correct labels on a finite test set. The observed spreads should influence follow-up design rather than retrospective exclusion. A model whose mean depends strongly on one run may need more prespecified repetitions before an engineering choice is made [219]. The additional runs should use the same protocol and should be included regardless of whether they raise or lower the mean [209]. If a documented implementation error is discovered, the corrected experiment should be reported with its reason and complete new results [167].

Individual runs and multiclass correlation

Table 8 reports all 21 runs, including the lowest-accuracy seed of every model. Accuracy, precision, recall, and F1 are percentages; AUROC and macro-AP are unitless. Errors count incorrect decisions among 450 images. Table 9 summarizes multiclass MCC over the same three fits.

Table 8. Individual seed metrics on the same 450 images. Macro-P and Macro-R denote macro-precision and macro-recall. Displayed values are rounded; means and variability are computed before rounding.

Model	Seed	Errors	Acc %	Macro P %	Macro R %	Macro F1 %	AUROC	Macro AP
ConvNeXt	0	3	99.33	99.38	99.33	99.33	0.99866	0.99522
ConvNeXt	1	2	99.56	99.57	99.56	99.56	0.99892	0.99612
ConvNeXt	2	2	99.56	99.57	99.56	99.56	0.99975	0.99755
Swin	0	3	99.33	99.36	99.33	99.33	0.99991	0.99923
Swin	1	4	99.11	99.13	99.11	99.11	0.99973	0.99789
Swin	2	4	99.11	99.11	99.11	99.11	0.99991	0.99921
CAFormer	0	5	98.89	98.91	98.89	98.89	0.99987	0.99893
CAFormer	1	5	98.89	98.97	98.89	98.89	0.99995	0.99954
CAFormer	2	5	98.89	98.91	98.89	98.89	0.99989	0.99904
InceptionNeXt	0	7	98.44	98.48	98.44	98.44	0.99955	0.99760
InceptionNeXt	1	5	98.89	99.00	98.89	98.89	0.99974	0.99826
InceptionNeXt	2	6	98.67	98.69	98.67	98.66	0.99967	0.99755
FasterNet	0	3	99.33	99.35	99.33	99.33	0.99984	0.99870
FasterNet	1	12	97.33	97.49	97.33	97.30	0.99931	0.99581
FasterNet	2	4	99.11	99.14	99.11	99.11	0.99987	0.99899
ViT	0	12	97.33	97.44	97.33	97.33	0.99960	0.99710
ViT	1	4	99.11	99.16	99.11	99.11	0.99986	0.99883
ViT	2	4	99.11	99.13	99.11	99.11	0.99975	0.99783
EfficientNetV2	0	8	98.22	98.23	98.22	98.21	0.99963	0.99701
EfficientNetV2	1	8	98.22	98.24	98.22	98.20	0.99948	0.99346
EfficientNetV2	2	6	98.67	98.71	98.67	98.66	0.99980	0.99835

Table 9. Multiclass Matthews correlation coefficient across seeds. The mean and sample SD summarize the same three fitted runs.

Model	Seed 0 MCC	Seed 1 MCC	Seed 2 MCC	Mean ± SD
ConvNeXt-Base	0.99264	0.99508	0.99508	0.99427 ± 0.00141
Swin-Base	0.99262	0.99015	0.99012	0.99096 ± 0.00144
CAFormer-M36	0.98768	0.98775	0.98768	0.98770 ± 0.00004
InceptionNeXt-Base	0.98276	0.98779	0.98522	0.98526 ± 0.00251
FasterNet-M	0.99261	0.97060	0.99016	0.98446 ± 0.01206
ViT-Base/16	0.97050	0.99018	0.99015	0.98361 ± 0.01135
EfficientNetV2-M	0.98028	0.98031	0.98524	0.98195 ± 0.00286

Class recall

Figure 12 summarizes mean recall for each class and model. Across all 21 runs, Crazing, Pitted surface, Punching, and Rolled in scale have perfect observed recall. Dark scratches, Patches, Silk spot, and Welding line have smaller losses. Inclusion has aggregate mean recall of approximately 97.14%, and Oil spot is lowest at approximately 92.70%.

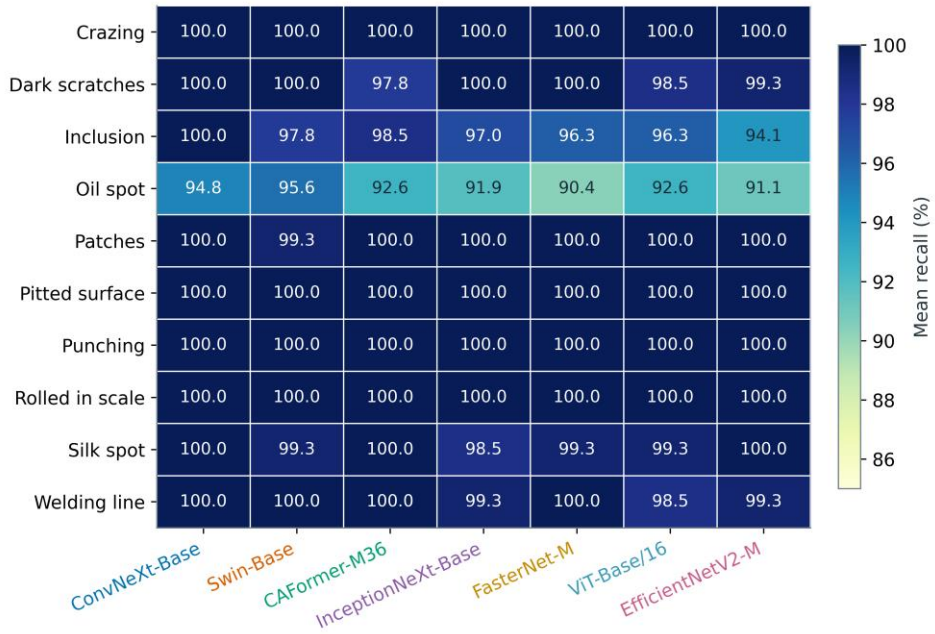


Figure 12. Class recall averaged across three seeds. Color and printed percentages show the recognition pattern over 45 unique images per class. The common scale spans 85% to 100%.

Perfect observed recall on 45 examples per seed is an encouraging result, but the three runs repeatedly evaluate the same 45 images. They do not create 135 independent class examples [220].

The absence of an error in those observations leaves uncertainty about new textures, lighting conditions, and borderline instances [2, 147]. Class difficulty should therefore be described for the sampled images rather than ranked as a universal physical property of the defect category. Oil spot deserves attention because its lower recall is shared across model families. This finding shifts the research priority from simply adding a larger backbone toward examining the class boundary and the data [159]. Images may be visually ambiguous, labels may reflect source-specific conventions, or relevant information may be weakened by cropping and resizing [2, 5]. These are hypotheses to investigate through source images and annotation review, not diagnoses established by the confusion matrix alone. Tables 10-16 report precision, recall, and F1 for each model-class combination. Precision loss identifies false positives from other categories,

while recall loss identifies missed members of a true category. Their distinction matters when predictions route surfaces to different inspection or repair actions.

Table 10. ConvNeXt-Base: class means across three seeds on 45 test images per class.

Class	Precision %	Recall %	F1 %
Crazing	100.00	100.00	100.00
Dark scratches	100.00	100.00	100.00
Inclusion	95.08	100.00	97.48
Oil spot	100.00	94.81	97.34
Patches	100.00	100.00	100.00
Pitted surface	100.00	100.00	100.00
Punching	100.00	100.00	100.00
Rolled in scale	100.00	100.00	100.00
Silk spot	100.00	100.00	100.00
Welding line	100.00	100.00	100.00

Table 11. Swin-Base: class means across three seeds on 45 test images per class.

Class	Precision %	Recall %	F1 %
Crazing	99.28	100.00	99.63
Dark scratches	100.00	100.00	100.00
Inclusion	95.65	97.78	96.69
Oil spot	98.52	95.56	97.00
Patches	100.00	99.26	99.63
Pitted surface	100.00	100.00	100.00
Punching	100.00	100.00	100.00
Rolled in scale	98.55	100.00	99.27
Silk spot	100.00	99.26	99.63
Welding line	100.00	100.00	100.00

Table 12. CAFormer-M36: class means across three seeds on 45 test images per class.

Class	Precision %	Recall %	F1 %
Crazing	100.00	100.00	100.00
Dark scratches	100.00	97.78	98.88
Inclusion	93.02	98.52	95.68
Oil spot	98.45	92.59	95.42
Patches	100.00	100.00	100.00
Pitted surface	100.00	100.00	100.00
Punching	100.00	100.00	100.00
Rolled in scale	100.00	100.00	100.00
Silk spot	100.00	100.00	100.00
Welding line	97.83	100.00	98.90

Table 13. InceptionNeXt-Base: class means across three seeds on 45 test images per class.

Class	Precision %	Recall %	F1 %
Crazing	99.28	100.00	99.63
Dark scratches	97.86	100.00	98.91
Inclusion	92.32	97.04	94.57
Oil spot	100.00	91.85	95.74
Patches	100.00	100.00	100.00
Pitted surface	100.00	100.00	100.00
Punching	99.28	100.00	99.63
Rolled in scale	99.28	100.00	99.63
Silk spot	99.24	98.52	98.88
Welding line	100.00	99.26	99.63

Table 14. FasterNet-M: class means across three seeds on 45 test images per class.

Class	Precision %	Recall %	F1 %
Crazing	99.28	100.00	99.63
Dark scratches	100.00	100.00	100.00
Inclusion	91.66	96.30	93.91
Oil spot	100.00	90.37	94.84
Patches	100.00	100.00	100.00
Pitted surface	99.28	100.00	99.63
Punching	100.00	100.00	100.00
Rolled in scale	97.13	100.00	98.54
Silk spot	99.26	99.26	99.26
Welding line	100.00	100.00	100.00

Table 15. ViT-Base/16: class means across three seeds on 45 test images per class.

Class	Precision %	Recall %	F1 %
Crazing	99.28	100.00	99.63
Dark scratches	100.00	98.52	99.25
Inclusion	92.84	96.30	94.53
Oil spot	100.00	92.59	96.13
Patches	100.00	100.00	100.00
Pitted surface	100.00	100.00	100.00
Punching	97.28	100.00	98.58
Rolled in scale	100.00	100.00	100.00
Silk spot	97.83	99.26	98.53
Welding line	98.55	98.52	98.51

Table 16. EfficientNetV2-M: class means across three seeds on 45 test images per class.

Class	Precision %	Recall %	F1 %
Crazing	100.00	100.00	100.00
Dark scratches	98.55	99.26	98.89
Inclusion	92.72	94.07	93.37
Oil spot	98.41	91.11	94.61
Patches	100.00	100.00	100.00
Pitted surface	100.00	100.00	100.00
Punching	99.28	100.00	99.63
Rolled in scale	97.83	100.00	98.90
Silk spot	98.58	100.00	99.28
Welding line	98.58	99.26	98.90

Direction of confusion

Figures 13a and 13b pool the confusion counts across the three runs of each model. Each row contains 135 predictions, obtained by evaluating 45 unique class images three times. The counts and row fractions summarize repeated decisions and should not be mistaken for an enlarged independent test set. The pooled view is useful for showing persistent confusion directions, while the individual-seed records remain the primary units for the reported seed variation.

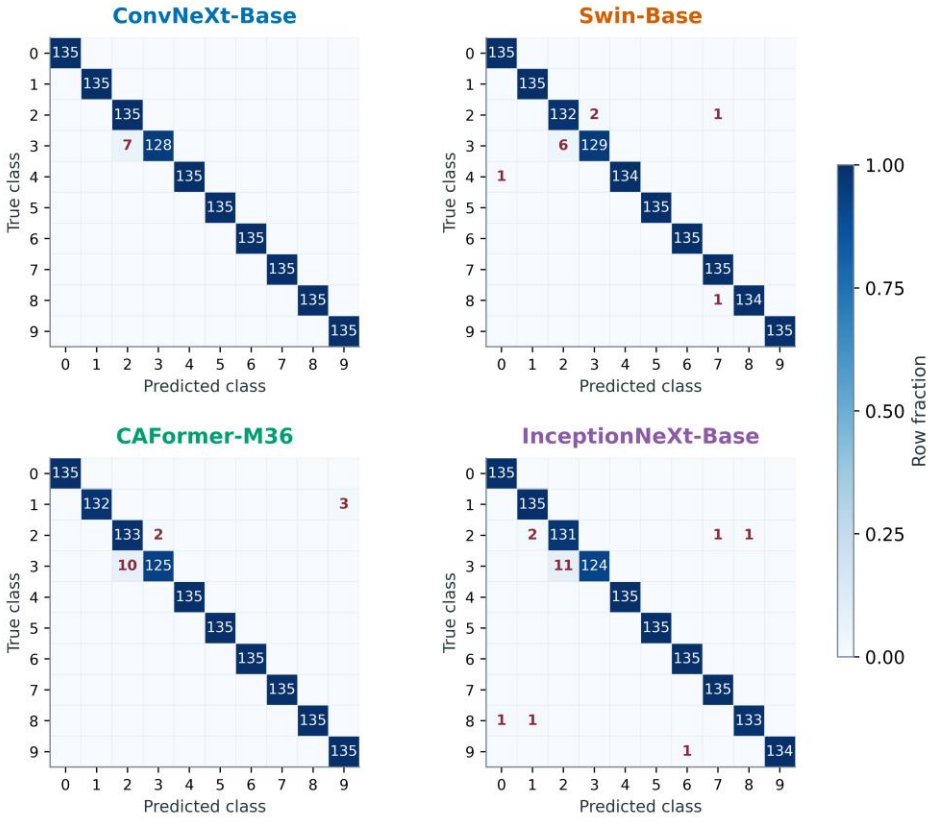


Figure 13a. Pooled confusion matrices for ConvNeXt, Swin, CAFormer, and InceptionNeXt. Rows are true classes and columns predicted classes. Blue intensity denotes the row fraction; printed values are counts. Each row comprises 135 predictions from repeated evaluation of 45 unique images.

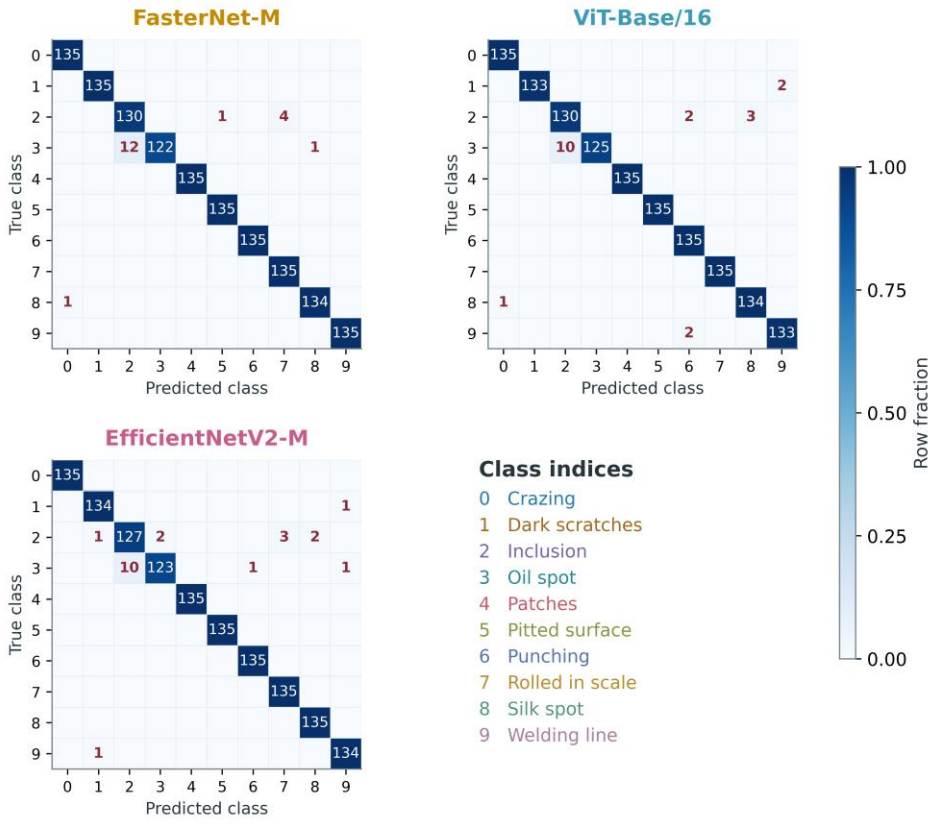


Figure 13b. Pooled confusion matrices for FasterNet, ViT, and EfficientNetV2, with the common class-index legend. Nonzero errors are printed in red. Counts pool three seeds on the same 45 unique images per class, rather than additional independent test images.

Oil spot predicted as Inclusion is the most important shared direction. Its pooled count is seven for ConvNeXt, six for Swin, ten for CAFormer, eleven for InceptionNeXt, twelve for FasterNet, ten for ViT, and ten for EfficientNetV2. The broad recurrence across convolutional, transformer, and hybrid pipelines suggests that the remaining issue is not exclusive to one token-mixing design. The pattern does not establish the correctness of either label interpretation. An image that every model calls Inclusion could still be a valid Oil spot under the annotation manual. Conversely, it could reveal a mislabeled or inherently ambiguous example [159]. Model agreement is evidence about learned appearance, not a replacement for an

independent labeling decision [221]. Review should use the original image, source metadata, and explicit category criteria [2].

Ranking and hard decisions

CAFormer-M36 has the highest mean macro-AUROC, approximately 0.99990, and macro-AP, approximately 0.99917. ConvNeXt-Base leads in accuracy and macro-F1 but has a lower mean macro-AUROC, approximately 0.99911, and macro-AP, approximately 0.99630. These differences illustrate that ranking the true class against other images and choosing the largest score within one image are related but distinct operations [26].

A hard error occurs when another class receives a larger score than the correct class on one image. One-versus-rest ranking asks how the score for a particular class orders that class's positives against its negatives across images [26]. A few hard errors can coexist with excellent one-versus-rest ordering, especially when scores are close. Conversely, a model can make almost every hard decision correctly while its probability ranking is less favorable on a small number of examples. Near-saturated AUROC and AP should be read with particular care. Small changes in a few observations can alter the fourth decimal place [222]. The metrics preserve useful information but do not independently validate defect severity assessment, unknown-class handling, or production prevalence [27, 28]. Their main role here is to describe a further aspect of the same held-out predictions.

Conditional intervals and paired differences

The stratified image bootstrap gives ConvNeXt a 95% conditional accuracy interval from approximately 98.81% to 100.00%. Swin's interval extends from approximately 98.44% to 99.78%. The remaining intervals overlap much of this high-accuracy region. These intervals refer to resampling the test images while retaining the fitted seeds; they should not be read as the full uncertainty of a newly trained model deployed on a new site.

Table 17. Stratified image-bootstrap accuracy intervals. Ten thousand paired replicates resample images within each class while keeping all fitted seeds fixed; intervals are conditional and exploratory.

Model	Mean accuracy %	95% interval %
ConvNeXt-Base	99.48	98.81 to 100.00
Swin-Base	99.19	98.44 to 99.78
CAFormer-M36	98.89	98.00 to 99.63
InceptionNeXt-Base	98.67	97.78 to 99.41
FasterNet-M	98.59	97.63 to 99.41
ViT-Base/16	98.52	97.63 to 99.33
EfficientNetV2-M	98.37	97.41 to 99.19

Figure 14 displays paired differences between ConvNeXt and the other six models. The interval for the leading pair includes zero. The other five intervals remain positive, but all are exploratory, unadjusted image-bootstrap intervals. Their apparent precision is conditional on an independent-image assumption and on the chosen balanced composition. They cannot substitute for an independent external experiment.

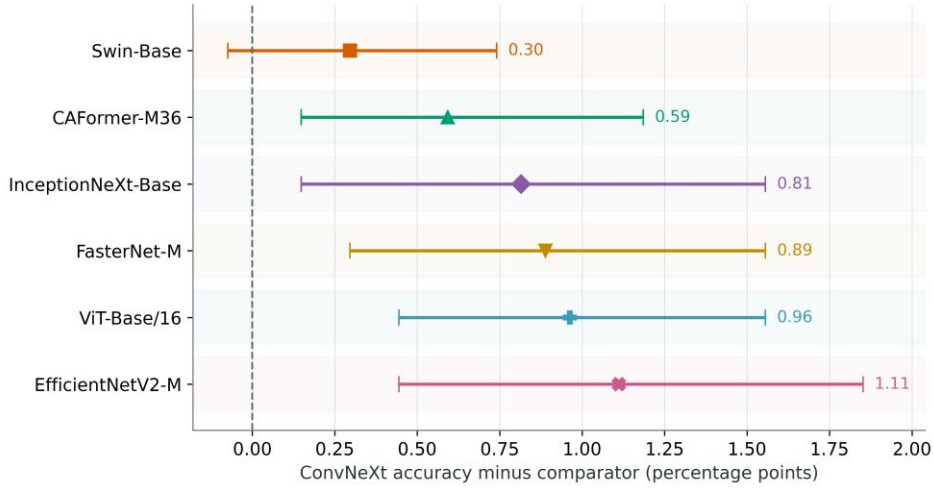


Figure 14. ConvNeXt minus each comparator in mean accuracy, with 95% paired class-stratified image-bootstrap percentile intervals. Intervals are conditional on the fitted seeds, exploratory, and unadjusted; the dashed line denotes zero. Printed values are mean differences in percentage points.

None of the exact McNemar comparisons remains significant after global Holm correction. ConvNeXt has the highest observed mean accuracy, but this finite-test lead does not establish a persistent advantage across acquisition sources. That question requires independent data and a comparison specified before evaluating them.

Complete paired comparisons

The table reports all 63 seed-specific exact McNemar tests. The two discordant counts identify images on which the first model alone or the second model alone is correct. Raw probabilities are two-sided. Holm-adjusted probabilities refer to the entire family. Zero discordant counts in both directions yield a probability of one. These comparisons concern the sampled classification task; they are not equivalence tests.

Table 18. Exact McNemar comparisons and global Holm adjustment. A only and B only count discordant correctness. Model abbreviations identify the same variants used throughout the book.

Model A / Model B	Seed	A only	B only	Raw p	Holm p
ConvNeXt / Swin	0	1	1	1.000000	1.000000
ConvNeXt / Swin	1	2	0	0.500000	1.000000
ConvNeXt / Swin	2	3	1	0.625000	1.000000
ConvNeXt / CAFormer	0	2	0	0.500000	1.000000
ConvNeXt / CAFormer	1	3	0	0.250000	1.000000
ConvNeXt / CAFormer	2	3	0	0.250000	1.000000
ConvNeXt / InceptionNeXt	0	5	1	0.218750	1.000000
ConvNeXt / InceptionNeXt	1	3	0	0.250000	1.000000
ConvNeXt / InceptionNeXt	2	5	1	0.218750	1.000000
ConvNeXt / FasterNet	0	1	1	1.000000	1.000000
ConvNeXt / FasterNet	1	10	0	0.001953	0.123047
ConvNeXt / FasterNet	2	2	0	0.500000	1.000000
ConvNeXt / ViT	0	9	0	0.003906	0.242188
ConvNeXt / ViT	1	2	0	0.500000	1.000000
ConvNeXt / ViT	2	2	0	0.500000	1.000000

Model A / Model B	Seed	A only	B only	Raw p	Holm p
ConvNeXt / EfficientNetV2	0	5	0	0.062500	1.000000
ConvNeXt / EfficientNetV2	1	6	0	0.031250	1.000000
ConvNeXt / EfficientNetV2	2	4	0	0.125000	1.000000
Swin / CAFormer	0	3	1	0.625000	1.000000
Swin / CAFormer	1	3	2	1.000000	1.000000
Swin / CAFormer	2	3	2	1.000000	1.000000
Swin / InceptionNeXt	0	5	1	0.218750	1.000000
Swin / InceptionNeXt	1	3	2	1.000000	1.000000
Swin / InceptionNeXt	2	2	0	0.500000	1.000000
Swin / FasterNet	0	1	1	1.000000	1.000000
Swin / FasterNet	1	9	1	0.021484	1.000000
Swin / FasterNet	2	2	2	1.000000	1.000000
Swin / ViT	0	9	0	0.003906	0.242188
Swin / ViT	1	2	2	1.000000	1.000000
Swin / ViT	2	3	3	1.000000	1.000000
Swin / EfficientNetV2	0	6	1	0.125000	1.000000
Swin / EfficientNetV2	1	6	2	0.289062	1.000000
Swin / EfficientNetV2	2	3	1	0.625000	1.000000
CAFormer / InceptionNeXt	0	5	3	0.726562	1.000000
CAFormer / InceptionNeXt	1	2	2	1.000000	1.000000
CAFormer / InceptionNeXt	2	3	2	1.000000	1.000000
CAFormer / FasterNet	0	1	3	0.625000	1.000000
CAFormer / FasterNet	1	8	1	0.039062	1.000000
CAFormer / FasterNet	2	2	3	1.000000	1.000000
CAFormer / ViT	0	8	1	0.039062	1.000000
CAFormer / ViT	1	1	2	1.000000	1.000000
CAFormer / ViT	2	2	3	1.000000	1.000000
CAFormer / EfficientNetV2	0	5	2	0.453125	1.000000
CAFormer / EfficientNetV2	1	6	3	0.507812	1.000000
CAFormer / EfficientNetV2	2	4	3	1.000000	1.000000

Model A / Model B	Seed	A only	B only	Raw p	Holm p
InceptionNeXt / FasterNet	0	1	5	0.218750	1.000000
InceptionNeXt / FasterNet	1	8	1	0.039062	1.000000
InceptionNeXt / FasterNet	2	2	4	0.687500	1.000000
InceptionNeXt / ViT	0	7	2	0.179688	1.000000
InceptionNeXt / ViT	1	1	2	1.000000	1.000000
InceptionNeXt / ViT	2	3	5	0.726562	1.000000
InceptionNeXt / EfficientNetV2	0	5	4	1.000000	1.000000
InceptionNeXt / EfficientNetV2	1	5	2	0.453125	1.000000
InceptionNeXt / EfficientNetV2	2	3	3	1.000000	1.000000
FasterNet / ViT	0	9	0	0.003906	0.242188
FasterNet / ViT	1	1	9	0.021484	1.000000
FasterNet / ViT	2	2	2	1.000000	1.000000
FasterNet / EfficientNetV2	0	5	0	0.062500	1.000000
FasterNet / EfficientNetV2	1	4	8	0.387695	1.000000
FasterNet / EfficientNetV2	2	2	0	0.500000	1.000000
ViT / EfficientNetV2	0	2	6	0.289062	1.000000
ViT / EfficientNetV2	1	5	1	0.218750	1.000000
ViT / EfficientNetV2	2	4	2	0.687500	1.000000

14. Residual Errors and Predictive Confidence

A small set of difficult images

The 21 runs produce 9,450 predictions, but these correspond to repeated evaluation of only 450 distinct images. Across those predictions, 112 are wrong. The errors occur on 25 unique images, meaning that 425 images are correctly classified by every run. This concentration gives the residual errors a structure that is hidden by a single mean accuracy value. The Oil spot image os_460.jpg is misclassified in all 21 runs. The images os_130.jpg and os_155.jpg are misclassified in 18 and 12 runs, respectively. Inclusion images in_124.jpg and in_439.jpg are misclassified in ten and seven runs. Figure 15 identifies the most frequently misclassified images, allowing the numerical pattern to be connected to individual cases.

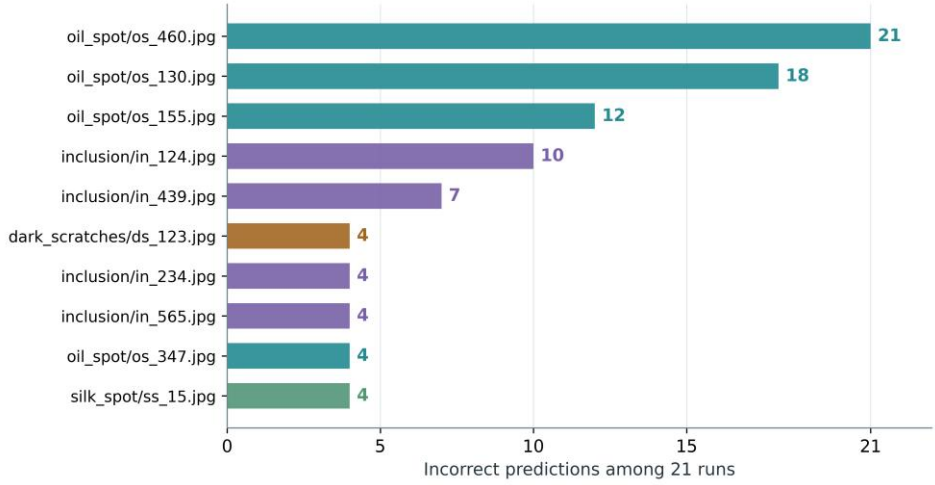


Figure 15. Test images most frequently misclassified across all 21 runs. Bar colors identify their reference class and labels give the number of incorrect decisions. Counts describe repeated predictions on individual images.

A sample-level review should begin with the most consistently wrong images [221]. For each one, reviewers should examine whether the stored reference label is supported by the annotation criteria, whether another

class is visually plausible, and whether image processing has obscured relevant detail [2]. The review should also compare the original-resolution image with the transformed input actually presented to the model [5]. This separates a labeling question from an acquisition or preprocessing problem. The outcome of review must be recorded before any label correction is incorporated into a revised experiment. Relabeling a test image because several models disagree with it would make evaluation circular [214]. If independent experts identify an annotation error, a corrected dataset version can be released with a documented change, and the original and revised evaluations can be distinguished [19]. If the image remains genuinely ambiguous, a single-label task may not express the uncertainty in the reference annotation [223].

Shared errors and complementarity

Different architectures are sometimes expected to make complementary errors, but the expectation must be checked [21, 129; 224]. The universally wrong Oil spot image shows an error that an ordinary majority vote cannot remove, because every run makes an incorrect decision.

Other images are wrong only for a subset of models or seeds, leaving some potential for complementary information. That potential is not itself an ensemble result. No ensemble is trained, tuned, or evaluated in this book. An ensemble experiment would need a defined combination rule selected using validation data, an appropriate checkpoint set, and a new or properly retained test protocol [225]. Choosing the most favorable vote after inspecting these test errors would overstate the generalization evidence [45]. The practical value of shared-error analysis is to identify whether further capacity is likely to address the main failure pattern. If all families fail on the same image, improving labels or acquisition may be more productive than adding another similar pretrained model [159]. If failures differ systematically by class or imaging condition, a specialist model or review policy may be worth evaluating. Both decisions require evidence beyond the aggregate count.

Probability scores and calibration

Table 19 reports mean negative log-likelihood, multiclass Brier score, ECE with 15 bins, and mean confidence. Swin-Base has the lowest mean NLL, approximately 0.10995, and the lowest mean Brier score, approximately 0.02078. FasterNet-M has the lowest mean ECE under the stated binning scheme, approximately 0.07925. EfficientNetV2-M has the highest mean NLL, Brier score, and ECE in this comparison.

Table 19. Mean probability measures across three seeds. NLL uses natural logarithms; Brier sums across ten classes; ECE uses 15 equal-width confidence bins. Lower is better for the first three measures.

Model	NLL	Brier	ECE 15	Confidence %
ConvNeXt-Base	0.13276	0.02605	0.10129	89.35
Swin-Base	0.10995	0.02078	0.08564	90.92
CAFormer-M36	0.12352	0.02521	0.09607	89.74
InceptionNeXt-Base	0.14276	0.03343	0.10249	88.78
FasterNet-M	0.12116	0.03148	0.07925	90.75
ViT-Base/16	0.13948	0.03263	0.09793	89.45
EfficientNetV2-M	0.17966	0.04886	0.12813	85.90

The mean confidence values are below mean accuracy for all seven models. For example, ConvNeXt has mean confidence of approximately 89.35% against mean accuracy of 99.48%. Swin has approximately 90.92% confidence against 99.19% accuracy. This indicates average underconfidence on the present test set. It does not establish that every image or every confidence bin is underconfident. The recorded use of a label-smoothing criterion provides one plausible contributor to less extreme probabilities, but the experiment does not isolate its causal effect [187, 188]. A comparison with a fully specified unsmoothed objective would be required. Other factors, including transfer, optimization, and the selected checkpoints, can affect confidence [34, 226]. The interpretation therefore identifies an observed pattern without attributing it to one untested mechanism.

Figure 16 provides reliability diagrams for ConvNeXt, Swin, and CAFormer, with separate seed traces and corresponding confidence-bin populations. Bins above the diagonal indicate empirical accuracy greater than confidence [202]. Sparse bins require caution because one or two observations can create a large visual deviation [227]. The figure supplements the weighted calibration summary rather than claiming a precisely estimated calibration curve at every possible probability.

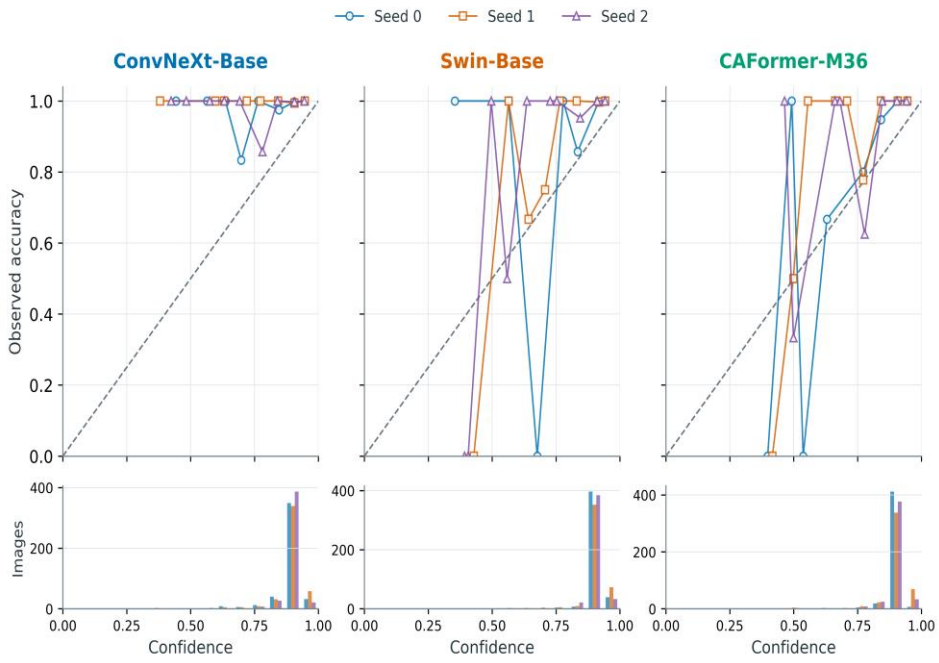


Figure 16. Reliability curves and confidence-bin populations for ConvNeXt, Swin, and CAFormer. Colors and open marker shapes identify seeds 0, 1, and 2. Curves use occupied bins among 15 equal-width confidence intervals; lower panels show image counts in each bin. The dashed line represents ideal calibration.

Figure 17 compares all seven models through NLL, Brier score, and ECE15, retaining individual seed scores beside their means and sample standard deviations. The different orderings show that ranking discrimination, hard decisions, and confidence agreement are distinct outcomes. In particular, FasterNet has the lowest mean ECE despite

greater accuracy variation; its Brier spread is also larger than that of the leading hard-label models.

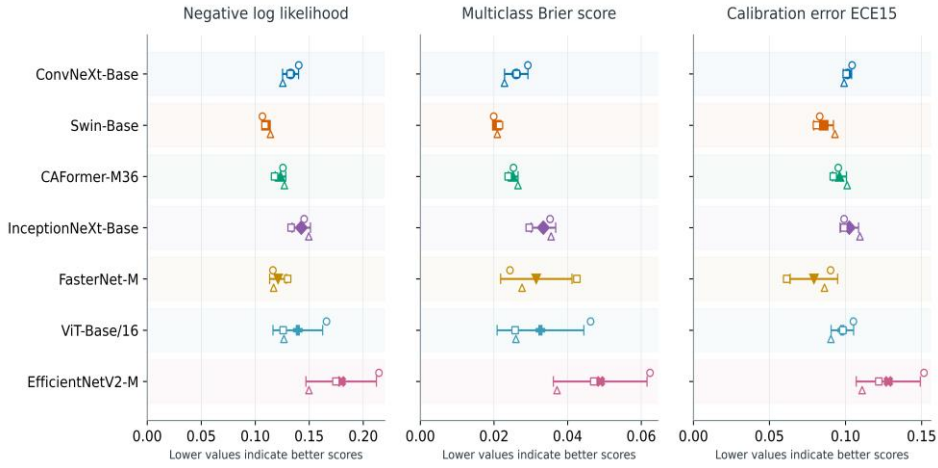


Figure 17. Probability quality for all seven pipelines. Filled model markers and horizontal bars show the mean and sample standard deviation over three seeds. Open circles, squares, and triangles show individual seeds 0, 1, and 2. Lower NLL, Brier, and ECE15 values are better; all quantities are dimensionless.

From confidence to review

A confidence value is not automatically a probability of engineering acceptability. Here it concerns the classifier's distribution over ten dataset labels. A low-confidence image may involve a difficult known category, an unknown pattern, an unusual background, or acquisition degradation [30, 32]. A high-confidence image can still be wrong, including when the model encounters conditions absent from training [28, 228]. A practical review policy would specify which outputs require human examination and what decision the reviewer can make [229]. Such a policy can use validation data to fit a calibrator or choose a confidence threshold [34]. Its evaluation should report coverage, retained-set error, class-specific review rates, and the errors among automatically processed images [230]. The current test has not been used to fit such a policy, so no deployment threshold is proposed as an established result. Underconfidence can increase review load if a threshold is chosen as though probabilities were perfectly calibrated. Overconfidence can hide difficult errors. Both

patterns matter, and the observed direction in this experiment does not guarantee the same direction under a new camera or surface population [31]. Calibration should be evaluated again after the intended input distribution and operational decision have been defined.

Explaining decisions

Gradient-based activation maps can help inspect the image regions associated with a class decision [231]. Attention maps and perturbation-based explanations offer related investigative tools [118, 232]. Such tools are relevant to future analysis here, especially for Oil spot and Inclusion. Explanation maps are outside the recorded evaluation; interpretation here instead uses sample identifiers, confusion patterns, and class probabilities. An explanation should be assessed against both the original image and the model's transformed input. A visually appealing heatmap can highlight a defect region without showing that the class decision is causally justified [233].

Sanity checks of explanation methods help test whether a map responds to the learned model rather than merely reproducing image structure [234]. Comparison with background controls, image perturbations, and independent localization annotations can provide stronger evidence [235]. Explanations are most useful when they answer a specific failure question rather than serve as decoration beside a high accuracy score. The recurring difficult images provide an appropriate starting set for such work. They can reveal whether a pipeline relies on the marked region, a border, an illumination artifact, or a source-specific cue [13, 160]. Their role remains investigative. A heatmap cannot establish the physical cause of a mark or correct an uncertain reference label without further annotation evidence.

15. Engineering Implications

Choosing a candidate pipeline

ConvNeXt-Base is the strongest descriptive candidate when mean hard-label accuracy and macro-F1 are the primary criteria in this experiment. Swin-Base is a strong candidate when probability-based quality is also important. CAFormer-M36 combines a smaller parameter count than these two models with high recognition performance and favorable ranking scores. The choice among them should be connected to the intended operational output rather than reduced to one universal order. Selecting an inspection pipeline requires more than a recognition ranking. Processing time, memory, error cost, review capacity, and acquisition stability define operational constraints [13]. The present comparison quantifies recognition and probability quality; those additional constraints remain subjects for deployment testing. Figure 18 plots parameter count against mean accuracy, with sample standard deviations across the three seeds. The relation is not monotonic. CAFormer uses approximately 53.92 million parameters, while ConvNeXt and Swin use approximately 87.58 and 86.75 million. FasterNet and EfficientNetV2 have similar parameter counts to CAFormer but different observed accuracies. Parameter count alone therefore does not predict the result [97].

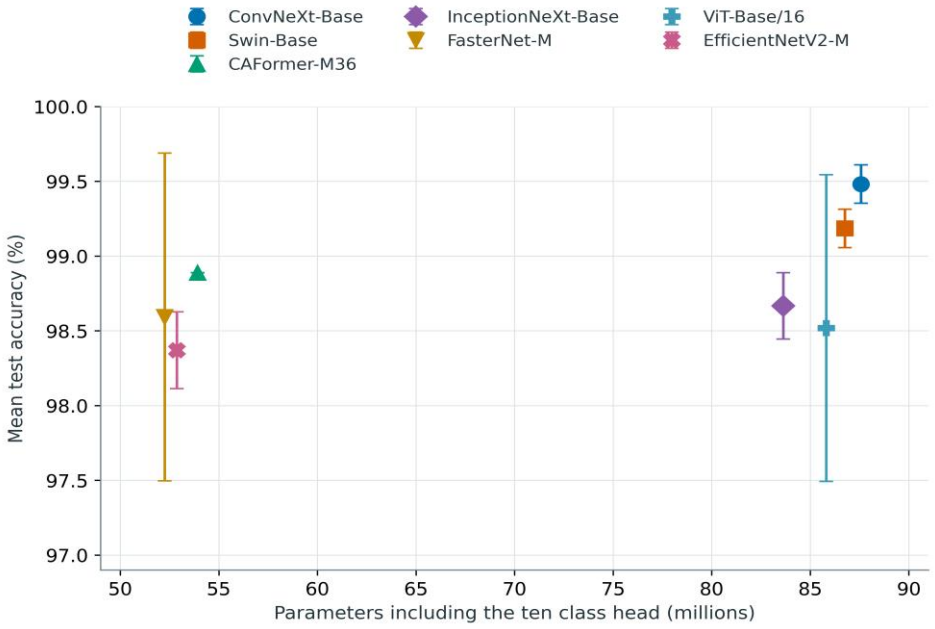


Figure 18. Model size and recognition accuracy. Model colors and marker shapes are consistent across comparison plots; vertical bars show the sample standard deviation across three seeds. Parameter counts include the ten-class classifier. Model size does not measure device latency or energy.

Parameter count is also an incomplete proxy for deployment cost. Activation sizes, operator implementations, memory access, precision, batch size, and hardware can change latency [12, 97]. The efficiency motivations of FasterNet and InceptionNeXt reinforce the need for measured execution rather than assumptions based only on model names or parameter storage [97, 112]. No inference-time or energy advantage is claimed for the present runs.

Measuring speed meaningfully

A future speed study should distinguish throughput from single-image latency. Throughput describes the number of images processed over time, often under batching. Latency describes the delay for one decision, which may be more relevant when each image must trigger a downstream action [236]. Both need a clearly stated measurement boundary: preprocessing, transfer to the accelerator, network computation, and post-processing can each contribute [13]. Measurements should use warm-up iterations, device

synchronization where required, and a sufficient number of timed observations. Report the distribution of latency, including a high percentile, rather than only a favorable mean [236]. Record input resolution, batch size, precision, framework, hardware, and power or thermal conditions. A quoted FPS value from another paper cannot be assigned to the current model without these conditions [13]. The reviewed lightweight detectors provide examples of deployment-oriented priorities. LDR-YOLO reports a compact detector and throughput under its own setup, while the lightweight attention study includes a Raspberry Pi setting [165, 166]. These studies motivate appropriate engineering measurements. Their detection architectures, devices, and workloads differ from the seven classification pipelines, so their speeds are not entries in a shared ranking.

The missing normal class

The ten-class subset contains named defect or surface-pattern categories and no normal category. The classifier therefore answers which known category best matches an input under the closed label set. It does not establish the sensitivity and specificity of a normal-versus-defective screening system. Feeding a normal image to the classifier would still force a prediction among the ten labels unless another mechanism were added [28, 237]. A production screening study requires representative normal surfaces as well as defects [35]. The class prevalence should reflect the intended use or be explicitly adjusted in the evaluation. False positives among normal surfaces can dominate review workload when defects are rare [13, 26]. Conversely, false negatives can have high consequences depending on the material requirement. Those trade-offs cannot be inferred from a balanced test containing only the ten current categories. Unknown defect types create a related problem. A closed-set softmax classifier may assign them a familiar label with high confidence [32, 228]. An open-set or anomaly-detection component could be studied, but it would require a defined unknown set and its own protocol [28, 35]. The current accuracy does not measure rejection of unknown patterns.

Acquisition and distribution shift

Industrial images can differ from public benchmark crops in illumination, camera geometry, motion blur, scale, surface treatment, and background

[2]. They can also change over time as equipment or processes change [238]. Common-corruption benchmarks formalize some acquisition-related perturbations [30], while uncertainty studies show that probability quality must be assessed under distribution shift [31]. A robust deployment study should sample these conditions deliberately instead of assuming that a random split covers them [13]. Source-aware evaluation is especially important for a merged benchmark. If particular labels originate mainly from particular source datasets, a model may learn acquisition cues correlated with the class [2, 239]. High within-mixture accuracy can then coexist with weaker transfer to a new source that lacks those cues [13, 240]. Establishing this possibility requires a source manifest and controlled source-held-out analysis; the current predictions alone do not prove or exclude it. A suitable external dataset should contain the same defined labels or an explicitly documented mapping, and should not reuse original images already included in the merged source [38, 174]. Merely naming one of SteelDefectX's originating datasets as external is insufficient. Overlap and acquisition relationships must be checked before independent validation is claimed. Temporal evaluation can also be useful. A model fitted on earlier production data can be assessed on later data without mixing the two periods [13, 38]. This creates a closer approximation to the direction of deployment, although it still requires representative labels and acquisition documentation. Any update after the later test becomes a new development step and should be assessed on a further held-out period or site.

Failure costs and human decisions

The benchmark weights the ten classes equally in its macro metrics. A real application may attach different costs to confusions [241]. Predicting Oil spot as Inclusion could lead to a different action from predicting a scratch-like pattern as a harmless process trace, depending on the plant's definitions. The correct cost structure requires engineering input; it cannot be invented from class names alone. A decision study should therefore specify the action associated with each prediction and the consequence of each error. It can then evaluate cost-weighted performance in addition to recognition metrics [42, 241]. The model's probability distribution may support that decision, but only after calibration and distributional

suitability are assessed [34, 242]. The operational threshold belongs to the action policy rather than to the benchmark accuracy table. Human review is most effective when reviewers receive the information needed to resolve the uncertainty. This may include the original image, neighboring frames, process history, and the model's plausible alternative classes, such as Inclusion, the most frequent confusion for Oil spot in this experiment. Reviewing only a resized crop and one predicted label can reproduce the same limitations faced by the classifier. The human process should itself be evaluated for consistency and correction effectiveness [243].

Monitoring after deployment

Deployment changes the role of evaluation from one-time comparison to ongoing evidence collection. Input drift, changes in review rates, class prevalence, and recurring errors should be monitored [244, 245]. A small audited sample of accepted and rejected decisions can help detect performance changes that would be missed by logging confidence alone [31]. Monitoring does not justify silently rewriting a historical benchmark. Updated labels, models, or thresholds should be versioned. Their performance should be evaluated under the updated policy, with the earlier result retained as an account of the previous system [19, 20]. This version discipline is the operational counterpart of the experimental protocol described in Chapter 11. The next priorities are independent testing, review of Oil spot and Inclusion errors, and assessment of probabilities under the intended operational conditions.

16. Limitations and Future Research

Data provenance and independence

The main data limitation is source and acquisition independence. The 450 test images have unique identifiers and distinct decoded pixels, but physical specimen identifiers, crop relationships, and acquisition events are not fully documented [13]. Distinct image content alone does not establish independent production observations [38]. The sampling plan targets 210 training, 45 validation, and 45 test images per class. Test support is confirmed at 45 per class. Final training and validation manifests are needed to establish the realized development counts after duplicate handling.

Future work should release a manifest connecting each prepared image to its source, source split, duplicate group, and acquisition group where available [19]. A perceptual or near-duplicate review can supplement exact-pixel hashing [174]. Group-aware splits can then prevent related images from entering different partitions [246]. Their effect on accuracy should be measured rather than assumed to be negligible or necessarily large [37].

Architecture and pretraining are confounded

The seven pipelines use different pretrained weight variants and different evaluation transformations. The experiment consequently does not identify an isolated architecture effect. A model's advantage could reflect its architecture, its initial representation, its input pipeline, its interaction with the fine-tuning settings, or several factors together [8, 13]. A matched study could compare architectures pretrained on the same source with comparable training procedures, then apply a common fine-tuning search protocol [111]. A complementary study could retain the best supported model-specific pipelines but allocate an equal validation-search budget to each [247, 248]. These designs answer different questions: architectural mechanism in the first case, practical achievable pipeline performance in the second. Neither design should tune against the current test set. Once these results guide a new configuration, a fresh independent evaluation is needed for a final comparison [45, 191]. The current experiment can

remain a development reference, but it cannot repeatedly certify every new choice as though no prior test information had been used.

Limited repetitions and near-saturated metrics

Three seeds provide more information than one run, but they remain a small sample of training variation [11, 176]. The larger spreads for FasterNet and ViT motivate additional repetitions. CAFormer's equal accuracy across three runs does not remove this need, because the observed equality is based on a finite and relatively easy test population. Near-saturated accuracy also makes the ranking sensitive to a handful of images. The leading pair differs by approximately 1.33 correct decisions per run on average. Larger and more diverse independent tests would provide stronger evidence about persistent differences. Merely evaluating the same images with more seeds cannot create that new image-level diversity [220]. An equivalence or noninferiority study could be appropriate when two pipelines perform similarly and one has a measured operational advantage. It would require a prespecified acceptable margin and adequate sample planning [217, 249].

The current failure to obtain Holm-adjusted significance does not establish such equivalence and should not be used as a shortcut to it [216].

Incomplete fitting records

The missing FasterNet seed 1 training log limits investigation of its lower accuracy. The unspecified smoothing coefficient, full scheduler, complete augmentations, and checkpoint-selection details limit exact reproduction across all models. Evaluation integrity and fitting reproducibility are related but distinct: the former is strongly supported by the prediction audit, while the latter needs the remaining records. Complete training configurations, checkpoint metadata, source manifests, and environment versions would permit exact reproduction and stronger investigation of the lower-accuracy runs [167, 250]. Any recovered discrepancy should be evaluated through a documented correction rather than retrospective exclusion of an unfavorable seed [209].

Focused literature rather than a systematic review

The literature synthesis covers 23 selected steel-inspection studies together with foundational methods. Its scope supports a focused methodological

discussion. A systematic review or meta-analysis would require a separate search strategy, eligibility criteria, screening process, and comparable outcomes [251]. Many reviewed papers report excellent results on NEU, GC10, or task-specific datasets, but their class counts, splits, augmentation, labels, and metrics differ [13]. They provide design insights and context rather than a controlled numerical leaderboard against SteelDefectX10. A future systematic review could extract protocol details more broadly and identify which comparisons are genuinely comparable. It should preserve the task distinctions developed here.

A practical sequence of extensions

The most immediate extension is to complete the data and fitting manifests. This addresses reproducibility before adding complexity. The next step is independent review of the recurring Oil spot and Inclusion errors, using explicit category criteria and original images. The resulting annotation evidence can clarify whether the failure is primarily semantic, visual, or related to preprocessing. Independent source or production data should then test the strongest candidate pipelines under conditions absent from development. This stage should include normal surfaces if screening is intended, and should retain a clear unknown-class policy. Calibration or selective review can be fitted using validation data and assessed separately from the unmodified classifier. Operational testing should then measure latency, memory, and review burden on the intended hardware. Resolution and regularization ablations belong within a separate validation protocol.

17. Conclusions

The seven-model experiment establishes high accuracy for ten-class steel surface recognition while exposing differences in repeatability, class discrimination, and probability quality. Three training seeds and a common balanced test of 450 images permit paired analysis and case-level investigation of the remaining failures.

ConvNeXt-Base obtains the highest observed mean accuracy and macro-F1. Swin-Base is close in hard-label accuracy and has the best mean negative log-likelihood and Brier score. CAFormer-M36 has the best mean macro-AUROC and macro average precision. FasterNet-M has the lowest ECE under the chosen binning, while FasterNet and ViT show the largest seed-related accuracy spreads. These findings demonstrate that recognition performance has several dimensions and that a preferred pipeline depends on the criterion being used.

The residual errors are highly concentrated. Only 25 distinct images account for all 112 wrong predictions across the 21 runs. Oil spot and Inclusion dominate the recurring confusion, and one Oil spot image is wrong in every run. This pattern directs attention toward annotation semantics, original-image review, and acquisition or preprocessing evidence. It does not support an assumption that adding capacity alone will remove the remaining mistakes.

Statistical analysis qualifies the descriptive ranking. The paired image-bootstrap interval for ConvNeXt minus Swin includes zero. None of the 63 seed-specific exact McNemar comparisons remains significant after global Holm adjustment. These results do not prove equivalence, but they prevent a small finite-test lead from being presented as a universal architectural advantage.

The benchmark also has explicit limits. Model-specific pretraining and preprocessing differ; the final training manifests and several fitting details are incomplete; the same 450 images are repeatedly evaluated; and independent production-line validation is absent. The ten-class label set contains no normal class and therefore does not establish a complete defect-screening system.

The supported conclusion is a strong comparative baseline for the specified SteelDefectX10 recognition task, accompanied by a transparent account of what remains uncertain. Completing provenance, reviewing shared errors, collecting independent data, and measuring operational performance are the next steps toward a dependable inspection application. The present results provide the first link in that chain for the SteelDefectX10 task

Glossary

Accuracy. The fraction of images assigned the correct reference label. Its relevance depends on the class distribution and task. Here the equal support of every class makes accuracy equal to macro-recall and balanced accuracy.

Annotation. The reference information attached to an image, such as a class label, bounding box, or pixel mask. Annotation uncertainty and inconsistency can limit evaluation even when the numerical calculation is correct.

Average precision. A summary of a precision-recall ranking curve that weights precision by increments in recall. The macro-AP values in this book average ten one-versus-rest average precisions; they are not trapezoidal curve areas.

AUROC. Area under the receiver operating characteristic curve. In the one-versus-rest setting it summarizes how scores order positive examples against negatives. It evaluates ranking across thresholds rather than only the final predicted class.

Balanced accuracy. The arithmetic mean of class recalls. Each class contributes equally regardless of prevalence. Because the present test has 45 images in every class, its balanced accuracy and overall accuracy coincide.

Batch size. The number of training examples processed together for an optimizer update, or evaluation examples processed together for inference. Batch size can affect optimization, memory requirements, and throughput; the recorded setting here is 16.

Bootstrap. A resampling method used to study a statistic under a specified sampling model. The image bootstrap here retains pairing across pipelines and class support; it conditions on the fitted models and their three seeds.

Brier score. The mean squared discrepancy between probabilities and one-hot labels. This book sums the discrepancies over all ten classes before averaging over images, giving a multiclass score between zero and two.

Calibration. Agreement between stated predictive confidence and observed correctness at that confidence. Calibration is distribution-dependent and distinct from high accuracy. The current analysis measures existing probability outputs without fitting a calibration transformation.

Checkpoint. A saved state of a trained model. The checkpoint evaluated at test time should be identified precisely, together with its selection criterion. A filename containing best does not by itself establish a valid selection protocol.

Closed set. A task that assigns every input to one of a fixed collection of known labels. This experiment is closed-set classification and does not measure rejection of normal surfaces or previously unseen defect categories.

Confusion matrix. A table of true and predicted labels. Diagonal entries represent correct decisions; off-diagonal entries identify the direction of errors. The direction can matter operationally even when two confusions have identical counts.

Data leakage. Information from evaluation examples entering fitting or model selection in a way that compromises the intended held-out assessment. Related crops or augmented versions can create leakage despite having different filenames or distinct file hashes.

Distribution shift. A difference between development and intended-use populations, including acquisition, texture, prevalence, or annotation changes. Strong results within one public-data mixture do not establish robustness to every such difference.

ECE. Expected calibration error under a specified binning rule. The measure weights absolute confidence-accuracy gaps by bin support. Results depend on the number of bins, their boundaries, and the observations available within them.

F1 score. The harmonic mean of precision and recall for a class. Macro-F1 averages the class-specific F1 scores. It generally differs from the harmonic mean of macro-precision and macro-recall, so the computational definition should remain explicit.

Fine tuning. Updating a pretrained recognition model using the target training data. The resulting performance depends on initial weights, transformations, optimization, and selection as well as architecture. It is not an architecture-only experiment.

Holm adjustment. A procedure that adjusts a family of hypothesis-test probabilities to control family-wise error. The present analysis uses one family containing all 63 seed-specific exact McNemar comparisons, rather than correcting each selected pair separately.

Label smoothing. A training approach that replaces a fully concentrated class target with a less extreme target distribution. The logs identify a smoothing criterion, but its coefficient and its causal effect on confidence are not established here.

Macro average. An unweighted average over classes. A rare and a common category contribute equally to a macro measure. This weighting choice describes a particular evaluation objective and may differ from the cost or prevalence of production decisions.

McNemar test. A paired test based on examples where two classifiers disagree in correctness. The exact version used here evaluates the symmetry of the discordant directions. Nonsignificance does not establish model equivalence.

Negative log likelihood. The average negative natural logarithm of the probability assigned to the reference label. Lower values are better. A confident wrong prediction can receive a large penalty even when most hard labels are correct.

Pretraining. Learning a representation before adaptation to the target task. The seven pipelines use different weight variants, including different indicated pretraining populations. This prior training is part of the pipeline being compared.

Precision and recall. Precision asks how many predictions of a class are correct; recall asks how many true examples of that class are recovered. Their denominators differ, so one can remain high while the other declines.

Sampling unit. The entity regarded as an independent observation, such as an image, specimen, coil, or acquisition batch. Resampling images assumes image-level independence; physical or source grouping can require a different uncertainty analysis.

Seed. An initial setting for random-number generation. Repeated seeds expose some fitting variation on a fixed partition, but do not create new test images or independent production sites. Identical accuracy need not mean identical learned parameters.

Standard deviation. A measure of spread among observed values. The book reports sample standard deviation across three seeds. It describes those runs and should not be treated as the complete uncertainty of performance on future populations.

Validation and test. Validation supports development decisions, while testing assesses the chosen procedure. Once test results guide a new configuration, those results become development evidence and a fresh independent assessment is needed for the revised pipeline.

References

1. Luo, Q., Fang, X., Liu, L., Yang, C., & Sun, Y. (2020). Automated visual defect detection for flat steel surface: A survey. *IEEE Transactions on Instrumentation and Measurement*, *69*(3), 626-644. https://doi.org/10.1109/TIM.2019.2963555.
2. Zhao, S., Gui, J., Yu, B., & Tao, D. (2026). SteelDefectX: A Multi-Form Vision-Language Dataset and Benchmark for Steel Surface Defect Analysis. *arXiv preprint arXiv:2603.21824*. https://arxiv.org/abs/2603.21824v2.
3. Zong, N., Jing, T., & Gebelin, J. C. (2026). Steel rolling in the age of artificial intelligence: a review. *Frontiers in Materials*, *13*, 1921706. https://doi.org/10.3389/fmats.2026.1921706
4. Zhang, M., Feng, M., Chen, C., Yu, X., & Lian, G. (2026). Weld defect detection: deep learning-based image processing and the mechanisms of defect formation. *Archives of Computational Methods in Engineering*, *33*(3), 3525-3563. https://doi.org/10.1007/s11831-025-10407-4.
5. Gao, K., Wang, S., & Li, B. (2026). A domain-adaptive contrastive and context-enhanced framework for steel surface defect detection. *Measurement Science and Technology*, *37*(33), 335404. https://doi.org/10.1088/1361-6501/ae9570.
6. Zheng, X., Liu, W., & Huang, Y. (2025). Legendre multiwavelet-based feature attention guidance lightweight network for accurate steel surface defect classification. *Engineering Applications of Artificial Intelligence*, *161*, 112179. https://doi.org/10.1016/j.engappai.2025.112179.
7. Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). Transformers in vision: A survey. *ACM computing surveys (CSUR)*, *54*(10s), 1-41. https://doi.org/10.1145/3505244.
8. Kornblith, S., Shlens, J., & Le, Q. V. (2019). Do better imagenet models transfer better?. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2661-2671). https://doi.org/10.1109/CVPR.2019.00277.
9. Geirhos, R., Meding, K., & Wichmann, F. A. (2020). Beyond accuracy: quantifying trial-by-trial behaviour of CNNs and humans by measuring error consistency. *Advances in neural information processing systems*, *33*, 13890-13902.
10. Summers, C., & Dinneen, M. J. (2021, July). Nondeterminism and instability in neural network optimization. In *International Conference on Machine Learning* (pp. 9913-9922). PMLR.
11. Pham, H. V., Qian, S., Wang, J., Lutellier, T., Rosenthal, J., Tan, L., ... & Nagappan, N. (2020, December). Problems and opportunities in training deep learning software systems: An analysis of variance. In *Proceedings of the 35th IEEE/ACM international conference on automated software engineering* (pp. 771-783). https://doi.org/10.1145/3324884.3416545.
12. Ma, N., Zhang, X., Zheng, H. T., & Sun, J. (2018, September). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *European conference*

on computer vision (pp. 122-138). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-01264-9_8

13. Wang, D., Xie, S., Li, Y., Zhu, L., Lin, Y., & Yuan, C. (2026). A Review of Steel Surface Defect Detection for Industrial Applications: Evaluation Frameworks, Data Challenges, and Model Mechanisms. *Technologies*, *14*(10), 644. https://doi.org/10.3390/technologies14100644.
14. Wen, X., Shan, J., He, Y., & Song, K. (2023). Steel Surface Defect Recognition: A Survey. *Coatings*, *13*(1), 17. https://doi.org/10.3390/coatings13010017.
15. Zhao, L., Yuan, Y., & Wu, Y. (2026). A Survey of Surface Defect Detection in Machine Vision: Addressing Core Challenges, Methodologies, and Dataset Analysis. *Computers, Materials & Continua*, *88*(2), 5. https://doi.org/10.32604/cmc.2026.080232.
16. Xu, K., Vallabh, C. K. P., & Manoochehri, S. (2025). Integrating machine learning with multimodal monitoring system utilizing acoustic and vision sensing to evaluate geometric variations in laser directed energy deposition. *The International Journal of Advanced Manufacturing Technology*, *141*(7), 4335-4359. https://doi.org/10.1007/s00170-025-16946-1.
17. Frydrych, K., Tomczak, M., Jasiński, J., & Papanikolaou, S. (2025). Steel surface defects analysis with machine vision and deep learning. *The International Journal of Advanced Manufacturing Technology*, *140*(7), 3691-3710. https://doi.org/10.1007/s00170-025-16539-y.
18. Chen, Y., Peng, X., Kong, L., Dong, G., Remani, A., & Leach, R. (2021). Defect inspection technologies for additive manufacturing. *International Journal of Extreme Manufacturing*, *3*(2), 022002. https://doi.org/10.1088/2631-7990/abe0d0.
19. Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Daumé III, H., & Crawford, K. (2021). Datasheets for datasets. *Communications of the ACM*, 64(12), 86–92. https://doi.org/10.1145/3458723.
20. Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., ... & Gebru, T. (2019, January). Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency* (pp. 220-229). https://doi.org/10.1145/3287560.3287596.
21. Tang, W. P., Liong, S. T., Chen, C. C., Tsai, M. H., Hsieh, P. C., Tsai, Y. T., ... & Wang, K. C. (2021). Design of multi-receptive field fusion-based network for surface defect inspection on hot-rolled steel strip using lightweight dataset. *Applied Sciences*, *11*(20), 9473. https://doi.org/10.3390/app11209473.
22. Wang, X., Jowers, C., Mazur, M., Buddery, A., Kent, D., Bab-Hadiashar, A., & Easton, M. (2022). Automated quality control inspection of geometric tip defects in medical needle manufacturing. *The International Journal of Advanced Manufacturing Technology*, 123, 2371–2384. https://doi.org/10.1007/s00170-022-10249-5.

23. Neogi, N., Mohanta, D. K., & Dutta, P. K. (2014). Review of vision-based steel surface inspection systems. *EURASIP Journal on Image and Video Processing*, *2014*(1), 50. https://doi.org/10.1186/1687-5281-2014-50.
24. Moon, C., Al-antari, M. A., and Gu, Y. H. (2026). Explainable hybrid AI CAD framework for advanced prediction of steel surface defects. Scientific Reports, 16, 10796. https://doi.org/10.1038/s41598-025-34320-9.
25. Shen, K., Zhou, X., and Liu, Z. (2026). Class Knowledge-Guided Lightweight Network for Salient Object Detection of Strip Steel Surface Defect. IEEE Transactions on Circuits and Systems for Video Technology, 36(6), 8941-8951. https://doi.org/10.1109/TCSVT.2026.3664887
26. Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, *27*(8), 861-874. https://doi.org/10.1016/j.patrec.2005.10.010.
27. Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, *10*(3), e0118432. https://doi.org/10.1371/journal.pone.0118432.
28. Scheirer, W. J., de Rezende Rocha, A., Sapkota, A., & Boult, T. E. (2012). Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, *35*(7), 1757-1772. https://doi.org/10.1109/TPAMI.2012.256.
29. Geng, C., Huang, S. J., & Chen, S. (2021). Recent advances in open set recognition: A survey. *IEEE transactions on pattern analysis and machine intelligence*, *43*(10), 3614-3631. https://doi.org/10.1109/TPAMI.2020.2981604.
30. Hendrycks, D., & Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*. https://arxiv.org/abs/1903.12261
31. Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., ... & Snoek, J. (2019). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, *32*. https://arxiv.org/abs/1906.02530
32. Hendrycks, D., & Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*. https://arxiv.org/abs/1610.02136.
33. Bergmann, P., Batzner, K., Fauser, M., Sattlegger, D., & Steger, C. (2021). The MVTec anomaly detection dataset: a comprehensive real-world dataset for unsupervised anomaly detection. *International Journal of Computer Vision*, *129*(4), 1038-1059. https://doi.org/10.1007/s11263-020-01400-4.
34. Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017, July). On calibration of modern neural networks. In *International conference on machine learning* (pp. 1321-1330). PMLR.
35. Roth, K., Pemula, L., Zepeda, J., Schölkopf, B., Brox, T., & Gehler, P. (2022, June). Towards total recall in industrial anomaly detection. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 14298-14308). IEEE. doi: 10.1109/CVPR52688.2022.01392.

36. Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249-256). JMLR Workshop and Conference Proceedings.
37. Yagis, E., Atnafu, S. W., García Seco de Herrera, A., Marzi, C., Scheda, R., Giannelli, M., ... & Diciotti, S. (2021). Effect of data leakage in brain MRI classification using 2D convolutional neural networks. *Scientific reports*, *11*(1), 22544. https://doi.org/10.1038/s41598-021-01681-w.
38. Kapoor, S., & Narayanan, A. (2023). Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*, *4*(9). https://doi.org/10.1016/j.patter.2023.100804.
39. McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, *12*(2), 153-157. https://doi.org/10.1007/BF02295996.
40. Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, *10*(7), 1895-1923. https://doi.org/10.1162/089976698300017197.
41. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444. https://doi.org/10.1038/nature14539.
42. Bishop, C.M. (2006). Pattern Recognition and Machine Learning. Springer. ISBN: 978-0-387-31073-2
43. Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2, pp. 1-758). New York: springer. https://doi.org/10.1007/978-0-387-21606-5
44. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297. https://doi.org/10.1007/BF00994018.
45. Cawley, G. C., & Talbot, N. L. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. *The Journal of Machine Learning Research*, *11*, 2079-2107. https://www.jmlr.org/papers/v11/cawley10a.html.
46. Haralick, R. M., Shanmugam, K., & Dinstein, I. H. (1973). Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6), 610-621. https://doi.org/10.1109/TSMC.1973.4309314
47. Ojala, T., Pietikäinen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, *29*(1), 51-59. https://doi.org/10.1016/0031-3203(95)00067-4.
48. Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* (Vol. 1, pp. 886-893). Ieee. https://doi.org/10.1109/CVPR.2005.177
49. Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110. https://doi.org/10.1023/B:VISI.0000029664.99615.94.

50. Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6), 679-698. https://doi.org/10.1109/TPAMI.1986.4767851.
51. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830. https://www.jmlr.org/papers/v12/pedregosa11a.html
52. Varma, S., & Simon, R. (2006). Bias in error estimation when using cross-validation for model selection. *BMC bioinformatics*, *7*(1), 91. https://doi.org/10.1186/1471-2105-7-91.
53. Ambroise, C., & McLachlan, G. J. (2002). Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the national academy of sciences*, *99*(10), 6562-6566. https://doi.org/10.1073/pnas.102102699.
54. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, *13*(1), 21-27. https://doi.org/10.1109/TIT.1967.1053964.
55. Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32. https://doi.org/10.1023/A:1010933404324.
56. Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 29(5), 1189-1232. https://doi.org/10.1214/aos/1013203451.
57. Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794). https://doi.org/10.1145/2939672.2939785
58. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. *Advances in neural information processing systems*, *31*. https://arxiv.org/abs/1706.09516.
59. Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. MIT Press. https://www.deeplearningbook.org/
60. Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798-1828. https://doi.org/10.1109/TPAMI.2013.50
61. Bau, D., Zhu, J. Y., Strobelt, H., Lapedriza, A., Zhou, B., & Torralba, A. (2020). Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, *117*(48), 30071-30078. https://doi.org/10.1073/pnas.1907375117
62. Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807-814). https://www.cs.toronto.edu/~hinton/absps/reluICML.pdf

63. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026-1034).
64. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533-536. https://doi.org/10.1038/323533a0.
65. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958. https://www.jmlr.org/papers/v15/srivastava14a.html
66. Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, 22(3), 400-407. https://doi.org/10.1214/aoms/1177729586.
67. Bottou, L., Curtis, F. E., & Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM review*, *60*(2), 223-311. https://doi.org/10.1137/16M1080173.
68. Bottou, L. (2010, September). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers* (pp. 177-186). Heidelberg: Physica-Verlag HD. https://doi.org/10.1007/978-3-7908-2604-3_16
69. Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, *4*(5), 1-17. https://doi.org/10.1016/0041-5553(64)90137-5.
70. Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, May). On the importance of initialization and momentum in deep learning. In *International conference on machine learning* (pp. 1139-1147). pmlr. https://proceedings.mlr.press/v28/sutskever13.html.
71. Sun, S., Cao, Z., Zhu, H., Zhao, J. (2020). A survey of optimization methods from a machine learning perspective. *IEEE Transactions on Cybernetics*, 50(8), 3668–3681. https://doi.org/10.1109/TCYB.2019.2950779
72. Loshchilov, I. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*. https://arxiv.org/abs/1711.05101
73. Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). pmlr. https://proceedings.mlr.press/v37/ioffe15.html
74. Wu, Y., He, K. Group Normalization. *Int J Comput Vis* **128**, 742-755 (2020). https://doi.org/10.1007/s11263-019-01198-w
75. Huang, L., Qin, J., Zhou, Y., Zhu, F., Liu, L., & Shao, L. (2023). Normalization techniques in training dnns: Methodology, analysis and application. *IEEE transactions on pattern analysis and machine intelligence*, *45*(8), 10173-10196. https://doi.org/10.1109/TPAMI.2023.3250241.

76. Moradi, R., Berangi, R. & Minaei, B. (2020). A survey of regularization strategies for deep models. *Artif Intell Rev* **53**, 3947-3986. https://doi.org/10.1007/s10462-019-09784-7
77. Shorten, C., Khoshgoftaar, T.M. (2019). A survey on Image Data Augmentation for Deep Learning. *J Big Data* **6**, 60. https://doi.org/10.1186/s40537-019-0197-0
78. Zhang, H., Cisse, M., Dauphin, Y., & Lopez-Paz, D. (2018, April). mixup: Beyond empirical risk management. In *6th Int. Conf. learning representations (ICLR)* (pp. 1-13). https://doi.org/10.48550/arXiv.1710.09412.
79. Xu, M., Yoon, S., Fuentes, A., & Park, D. S. (2023). A comprehensive survey of image augmentation techniques for deep learning. *Pattern Recognition*, *137*, 109347. https://doi.org/10.1016/j.patcog.2023.109347
80. Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6023-6032). https://doi.org/10.1109/ICCV.2019.00612.
81. Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. V. (2019, June). Autoaugment: Learning augmentation strategies from data. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 113-123). IEEE. https://doi.org/10.1109/CVPR.2019.00020.
82. Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. (2020). Randaugment: Practical automated data augmentation with a reduced search space. *Advances in neural information processing systems*, *33*, 18613-18624. https://doi.org/10.1109/CVPRW50498.2020.00359.
83. Loshchilov, I., & Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*. https://arxiv.org/abs/1608.03983.
84. Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., ... & Wu, H. (2017). Mixed precision training. *arXiv preprint arXiv:1710.03740*. https://doi.org/10.48550/arXiv.1710.03740.
85. Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345-1359.https://doi.org/10.1109/TKDE.2009.191
86. Saberironaghi, A., Ren, J., & El-Gindy, M. (2023). Defect detection methods for industrial products using deep learning techniques: A review. *Algorithms*, *16*(2), 95. https://doi.org/10.3390/a16020095
87. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211-252. https://doi.org/10.1007/s11263-015-0816-y.
88. Ngo, A., Hernandez, K., Olatunde, A.E. *et al.* Image-Based Fracture Surface Defect Characterization Methods for Additively Manufactured Ti-6Al-4V Tested in Fatigue. *JOM* **76**, 4813–4824 (2024). https://doi.org/10.1007/s11837-024-06655-7.

89. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324. https://doi.org/10.1109/5.726791
90. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern recognition*, *77*, 354-377. https://doi.org/10.1016/j.patcog.2017.10.013
91. Du, Y., Chen, H., Fu, Y., Zhu, J., & Zeng, H. (2024). AFF-Net: a strip steel surface defect detection network via adaptive focusing features. *IEEE Transactions on Instrumentation and Measurement*, *73*, 1-14. https://doi.org/10.1109/TIM.2024.3398131
92. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*., https://arxiv.org/abs/1409.1556
93. Luo, W., Li, Y., Urtasun, R., & Zemel, R. (2016). Understanding the effective receptive field in deep convolutional neural networks. *Advances in neural information processing systems*, *29*. https://proceedings.neurips.cc/paper/2016/hash/c8067ad1937f728f51288b3eb986afaa-Abstract.html
94. Chollet, F. (2017, July). Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1800-1807). Ieee. https://doi.org/10.1109/CVPR.2017.195
95. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*. https://arxiv.org/abs/1704.04861
96. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018, June). Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF conference on computer vision and pattern recognition* (pp. 4510-4520). Ieee.https://doi.org/10.1109/CVPR.2018.00474
97. Chen, J., Kao, S. H., He, H., Zhuo, W., Wen, S., Lee, C. H., & Chan, S. H. G. (2023, June). Run, don't walk: chasing higher FLOPS for faster neural networks. In *2023 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 12021-12031). IEEE.https://doi.org/10.1109/CVPR52729.2023.01157
98. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).https://doi.org/10.1109/CVPR.2015.7298594
99. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).https://doi.org/10.1109/CVPR.2016.90

100. He, K., Zhang, X., Ren, S., Sun, J. (2016). Identity Mappings in Deep Residual Networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9908. Springer, Cham. https://doi.org/10.1007/978-3-319-46493-0_38
101. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017, July). Densely connected convolutional networks. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2261-2269). Ieee.https://doi.org/10.1109/CVPR.2017.243
102. Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017, July). Aggregated residual transformations for deep neural networks. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 5987-5995). IEEE. https://doi.org/10.1109/CVPR.2017.634
103. Hu, J., Shen, L., & Sun, G. (2020). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7132-7141). https://doi.org/10.1109/TPAMI.2019.2913372
104. Guo, M. H., Xu, T. X., Liu, J. J., Liu, Z. N., Jiang, P. T., Mu, T. J., ... & Hu, S. M. (2022). Attention mechanisms in computer vision: A survey. *Computational visual media*, *8*(3), 331-368. https://doi.org/10.1007/s41095-022-0271-y
105. Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PmLR. https://proceedings.mlr.press/v97/tan19a.html
106. Radosavovic, R. P. Kosaraju, R. Girshick, K. He and P. Dollár, "Designing Network Design Spaces," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 2020, pp. 10425-10433 https://doi.org/10.1109/CVPR42600.2020.01044.
107. Brock, A., De, S., Smith, S. L., & Simonyan, K. (2021, July). High-performance large-scale image recognition without normalization. In *International conference on machine learning* (pp. 1059-1071). PMLR. https://proceedings.mlr.press/v139/brock21a.html
108. Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., & Sun, J. (2021, June). Repvgg: Making vgg-style convnets great again. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 13728-13737). IEEE.https://doi.org/10.1109/CVPR46437.2021.01352
109. Woo, S., Debnath, S., Hu, R., Chen, X., Liu, Z., Kweon, I. S., & Xie, S. (2023, June). Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 16133-16142). Ieee. https://doi.org/10.1109/CVPR52729.2023.01548
110. Tan, M., & Le, Q. (2021, July). Efficientnetv2: Smaller models and faster training. In *International conference on machine learning* (pp. 10096-10106). PMLR. https://proceedings.mlr.press/v139/tan21a.html
111. Liu, Z., Mao, H., Wu, C. Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022, June). A convnet for the 2020s. In *2022 IEEE/CVF conference on computer vision and*

pattern recognition (CVPR) (pp. 11966-11976). IEEE.https://doi.org/10.1109/CVPR52688.2022.01167

112. Yu, W., Zhou, P., Yan, S., & Wang, X. (2024, June). Inceptionnext: When inception meets convnext. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 5672-5683). IEEE.https://doi.org/10.1109/CVPR52733.2024.00542
113. Dosovitskiy, A. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. https://openreview.net/forum?id=YicbFdNTTy
114. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*. https://papers.nips.cc/paper/7181-attention-is-all-you-need
115. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021, October). Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF international conference on computer vision (ICCV)* (pp. 9992-10002). Ieee. https://doi.org/10.1109/ICCV48922.2021.00986
116. Wang, W., Xie, E., Li, X., Fan, D. P., Song, K., Liang, D., ... & Shao, L. (2021, October). Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *2021 IEEE/CVF international conference on computer vision (ICCV)* (pp. 548-558). IEEE. https://doi.org/10.1109/ICCV48922.2021.00061
117. Jain, S., & Wallace, B. C. (2019, June). Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 3543-3556).https://doi.org/10.18653/v1/N19-1357
118. Chefer, H., Gur, S., & Wolf, L. (2021, June). Transformer interpretability beyond attention visualization. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 782-791). IEEE. https://doi.org/10.1109/CVPR46437.2021.00084.
119. Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., ... & Shen, C. (2021). Twins: Revisiting the design of spatial attention in vision transformers. *Advances in neural information processing systems*, *34*, 9355-9366. https://proceedings.neurips.cc/paper/2021/hash/4e0928de075538c593fbdabb0c5ef2c3 Abstract.html
120. Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., ... & Guo, B. (2022, June). Swin transformer v2: Scaling up capacity and resolution. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 11999-12009). IEEE. https://doi.org/10.1109/CVPR52688.2022.01170
121. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2021, July). Training data-efficient image transformers & distillation through attention.

In *International conference on machine learning* (pp. 10347-10357). PMLR. https://proceedings.mlr.press/v139/touvron21a.html

122. Hassani, A., Walton, S., Shah, N., Abuduweili, A., Li, J., & Shi, H. (2021). Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*.https://arxiv.org/abs/2104.05704
123. Bao, H., Dong, L., Piao, S., & Wei, F. (2021). Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*. https://openreview.net/forum?id=p-BhZSz59o4
124. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2022, June). Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 15979-15988). IEEE. https://doi.org/10.1109/CVPR52688.2022.01553
125. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., & Joulin, A. (2021, October). Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)* (pp. 9630-9640). IEEE. https://doi.org/10.1109/ICCV48922.2021.00951.
126. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., ... & Bojanowski, P. (2024). DINOv2: learning robust visual features without supervision. arXiv. *arXiv preprint arXiv:2304.07193*. https://openreview.net/forum?id=a68SUt6zFt
127. Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... & Sutskever, I. (2021, July). Learning transferable visual models from natural language supervision. In *International conference on machine learning* (pp. 8748-8763). PmLR. https://proceedings.mlr.press/v139/radford21a.html
128. Glenn, W. B. (1950). Verification of forecasts expressed in terms of probability. *Monthly weather review*, *78*(1), 1-3. https://doi.org/10.1175/1520-0493(1950)078%3C0001:VOFEIT%3E2.0.CO;2
129. Kuncheva, L.I., Whitaker, C.J. Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. *Machine Learning* **51**, 181–207 (2003). https://doi.org/10.1023/A:1022859003006
130. Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., & Zhang, L. (2021, October). Cvt: Introducing convolutions to vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 22-31). IEEE. https://doi.org/10.1109/ICCV48922.2021.00009
131. Dai, Z., Liu, H., Le, Q. V., & Tan, M. (2021). Coatnet: Marrying convolution and attention for all data sizes. *Advances in neural information processing systems*, *34*, 3965-3977.https://proceedings.neurips.cc/paper/2021/hash/20568692db622456cc42a2e853ca21f8-Abstract.html
132. d'Ascoli, S., Touvron, H., Leavitt, M. L., Morcos, A. S., Biroli, G., & Sagun, L. (2021, July). Convit: Improving vision transformers with soft convolutional inductive biases. In *International conference on machine learning* (pp. 2286-2296). PMLR. https://doi.org/10.1088/1742-5468/ac9830.

133. Mehta, S., & Rastegari, M. (2021). Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*. https://doi.org/10.48550/arXiv.2110.02178.
134. Tu, Z. *et al.* (2022). MaxViT: Multi-axis Vision Transformer. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds) Computer Vision – ECCV 2022. ECCV 2022. Lecture Notes in Computer Science, vol 13684. Springer, Cham. https://doi.org/10.1007/978-3-031-20053-3_27
135. Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., ... & Yan, S. (2022, June). Metaformer is actually what you need for vision. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 10809-10819). IEEE. https://doi.org/10.1109/CVPR52688.2022.01055
136. Yu, W., Si, C., Zhou, P., Luo, M., Zhou, Y., Feng, J., ... & Wang, X. (2024). Metaformer baselines for vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(2), 896-912. https://doi.org/10.1109/TPAMI.2023.3329173
137. Sabri, K., Gaceb, M. Real-time hybrid AI for strip steel surface monitoring in industry 4.0. *Int J Adv Manuf Technol* **142**, 1467–1493 (2026). https://doi.org/10.1007/s00170-025-17113-2
138. Kittler, J., Hatef, M., Duin, R. P., & Matas, J. (1998). On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, *20*(3), 226-239. https://doi.org/10.1109/34.667881
139. Singh, A. K., Singh, K. K., Singh, M. K., Ghosh, B., Pais, R. S., & Sharma, V. (2026). Steel defect detection based on feature selection and ensemble technique. *Engineering Applications of Artificial Intelligence*, *168*, 114047. https://doi.org/10.1016/j.engappai.2026.114047
140. Tuti, S., Chander, A. S., Varsitha, V., Appala, R. G., Jahnavi, V., Ramesh, G., & Reddy, G. S. (2026, March). Residual Attention-Enhanced Deep Learning Model for Robust Detection of Steel Surface Defects. In *Journal of Physics: Conference Series* (Vol. 3196, No. 1, p. 012098). IOP Publishing. https://doi.org/10.1088/1742-6596/3196/1/012098
141. Ren, J., Li, Z., Zhou, Z., Yang, M., Yu, L., He, H., & Liu, T. (2026). CFL-YOLO: A modified YOLOv8 for detection of steel surface defects. *Measurement*, 120509. https://doi.org/10.1016/j.measurement.2026.120509
142. Wang, Y., Lu, Y., Zhu, F., Du, G., & Li, Z. (2026). High precision classification of hot rolled strip steel surface defects using dual path features and entropy attention fusion. *Scientific Reports*, *16*(1), 5351.https://doi.org/10.1038/s41598-025-31683-x
143. Chen, C.J., Liu, RS. A multi-scale attention network for steel surface defect recognition. *Soft Comput* **30**, 2843–2859 (2026). https://doi.org/10.1007/s00500-025-11053-5
144. Liu, D., Liu, H., & Liang, J. (2026). ReForm-Net: Label reconstruction for high-precision detection of irregular steel surface defects. *Information Sciences*, 123444. https://doi.org/10.1016/j.ins.2026.123444

145. Zhang, H., Liu, H., Ma, W., Zheng, C., Zhang, T., Guo, R., ... & Liu, Q. (2026). Forward-reverse rectification network for real-time few-shot strip steel surface defect classification. *Measurement*, *265*, 120375. https://doi.org/10.1016/j.measurement.2026.120375
146. Yi, C., Chen, Q., Xu, B., & Huang, T. (2023). Steel Strip Defect Sample Generation Method Based on Fusible Feature GAN Model under Few Samples. *Sensors*, *23*(6), 3216. https://doi.org/10.3390/s23063216
147. Hanley, J. A., & Lippman-Hand, A. (1983). If nothing goes wrong, is everything all right?: interpreting zero numerators. *Jama*, *249*(13), 1743-1745.https://doi.org/10.1001/jama.1983.03330370053031
148. Clopper, C. J., & Pearson, E. S. (1934). The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, *26*(4), 404-413.https://doi.org/10.1093/biomet/26.4.404
149. Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. *Advances in neural information processing systems*, *30*.https://proceedings.neurips.cc/paper/2017/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html
150. Wang, Y., Yao, Q., Kwok, J. T., & Ni, L. M. (2020). Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, *53*(3), 1-34. https://doi.org/10.1145/3386252
151. Liao, J., Wang, D., Liang, J. *et al.* Enhanced multi-scale feature fusion for accurate steel surface defect detection. *Vis Comput* **42**, 401 (2026). https://doi.org/10.1007/s00371-026-04575-y
152. Wei, C., Bao, Y., Zheng, C. *et al.* AMFNet: aggregated multi-level feature interaction fusion network for defect detection on steel surfaces. *J Intell Manuf* **37**, 1615–1632 (2026). https://doi.org/10.1007/s10845-025-02613-5
153. Wang, J., Zheng, C., Qi, Y. *et al.* Development of a Cross-scale Connection Network With Gather-and-distribute Structure for Steel Surface Defect Detection. *J Nondestruct Eval* **45**, 24 (2026). https://doi.org/10.1007/s10921-025-01314-6
154. Everingham, M., Van Gool, L., Williams, C.K.I. *et al.* The Pascal Visual Object Classes (VOC) Challenge. *Int J Comput Vis* **88**, 303-338 (2010). https://doi.org/10.1007/s11263-009-0275-4
155. Lin, TY. *et al.* (2014). Microsoft COCO: Common Objects in Context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1_48
156. Liu, S., Miao, X. & Zhou, Y. Efficient Detection Algorithm of Steel Surface Defects Based on YOLOv8. *JOM* **78**, 4831–4847 (2026). https://doi.org/10.1007/s11837-026-08174-z
157. Xie, H., Zhou, H., Chen, R., & Wang, B. (2026). SDMS-YOLOv10: improved Yolov10-based algorithm for identifying steel surface flaws. *Nondestructive Testing and Evaluation*, *41*(2), 782-802.https://doi.org/10.1080/10589759.2025.2474103

158. Xu, K., Zhu, D., Shi, C. *et al.* YOLO-DBL: a multi-dimensional optimized model for detecting surface defects in steel. *J Membr Comput* **8**, 1–11 (2026). https://doi.org/10.1007/s41965-024-00178-5
159. Frénay, B., & Verleysen, M. (2014). Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, *25*(5), 845-869. https://doi.org/10.1109/TNNLS.2013.2292894
160. Geirhos, R., Jacobsen, JH., Michaelis, C. *et al.* Shortcut learning in deep neural networks. *Nat Mach Intell* **2**, 665–673 (2020). https://doi.org/10.1038/s42256-020-00257-z
161. Zhou, X., Wang, X., Li, G., Liu, D., She, Q., Xu, X., & Cong, R. (2026). Few-shot Strip Steel Surface Defect Segmentation via Pre-trained Variational Auto-Encoder based Latent Gaussian Process Regression. *IEEE Transactions on Image Processing*.https://doi.org/10.1109/TIP.2026.3671617
162. Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, *22*(11), 1330-1334. https://doi.org/10.1109/34.888718
163. Nam, C., & Lee, S. J. (2026). Context-aware defect sample generation using conditional diffusion model for surface defect inspection. *Measurement*, 121432.https://doi.org/10.1016/j.measurement.2026.121432
164. Song, J., Cui, G., Yang, C., Wu, H., Teng, Q., Zhou, F., & Wang, Y. (2026). FS-MSDANet: A Multi-Scale Dynamic Attention Network for Surface Defect Detection in Wear-Resistant Steel. *IEEE Access*. https://doi.org/10.1109/ACCESS.2026.3716250
165. Tan, S., Tan, J., Man, J. *et al.* LDR-YOLO: lightweight dynamic upsampling and regional attention enhancement network for steel surface defect detection. *J Real-Time Image Proc* **23**, 41 (2026). https://doi.org/10.1007/s11554-025-01838-0
166. Fu, Y., Wang, Y., Cheng, Z. *et al.* Steel surface defect detection based on lightweight network with attention feature fusion and multi-scale detection head. *J Ambient Intell Human Comput* **17**, 401-412 (2026). https://doi.org/10.1007/s12652-025-05034-7
167. Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché-Buc, F., ... & Larochelle, H. (2021). Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *Journal of machine learning research*, *22*(164), 1-20. https://www.jmlr.org/papers/v22/20-303.html
168. Zhao et al. 2026, SteelDefectX dataset, Hugging Face, https://huggingface.co/datasets/Zhaosxian/SteelDefectX
169. Song, K., & Yan, Y. (2013). A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. *Applied Surface Science*, *285*, 858-864. https://doi.org/10.1016/j.apsusc.2013.09.002.
170. Lv, X., Duan, F., Jiang, J.-j., Fu, X., & Gan, L. (2020). Deep Metallic Surface Defect Detection: The New Benchmark and Detection Network. *Sensors*, *20*(6), 1562. https://doi.org/10.3390/s20061562

171. Cui, W., Song, K., Jia, X., Chen, H., Zhang, Y., Yan, Y., & Jiang, W. (2024). An efficient targeted design for real-time defect detection of surface defects. *Optics and Lasers in Engineering*, *178*, 108174. https://doi.org/10.1016/j.optlaseng.2024.108174
172. Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, *45*(4), 427-437. https://doi.org/10.1016/j.ipm.2009.03.002
173. He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, *21*(9), 1263-1284. https://doi.org/10.1109/TKDE.2008.239
174. Barz, B., & Denzler, J. (2020). Do We Train on Test Data? Purging CIFAR of Near-Duplicates. *Journal of Imaging*, *6*(6), 41. https://doi.org/10.3390/jimaging6060041
175. Monga, V., & Evans, B. L. (2006). Perceptual image hashing via feature points: performance evaluation and tradeoffs. *IEEE transactions on Image Processing*, *15*(11), 3452-3465.https://doi.org/10.1109/TIP.2006.881948
176. Bouthillier, X., Delaunay, P., Bronzi, M., Trofimov, A., Nichyporuk, B., Szeto, J., ... & Vincent, P. (2021). Accounting for variance in machine learning benchmarks. *Proceedings of machine learning and systems*, *3*, 747-769. https://proceedings.mlsys.org/paper_files/paper/2021/hash/0184b0cd3cfb185989f858a1d9f5c1eb-Abstract.html
177. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Dennison, D. (2015). Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, *28*.https://proceedings.neurips.cc/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html
178. Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*. https://arxiv.org/abs/1312.4400
179. Wightman, R., and contributors (2026). PyTorch Image Models timm. Software repository; accessed 9 October 2026. https://github.com/huggingface/pytorch-image-models.
180. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q. (2016). Deep Networks with Stochastic Depth. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9908. Springer, Cham. https://doi.org/10.1007/978-3-319-46493-0_39
181. Elfwing, S., Uchibe, E., & Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, *107*, 3-11. https://doi.org/10.1016/j.neunet.2017.12.012.
182. Hendrycks, D., & Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*. https://doi.org/10.48550/arXiv.1606.08415.
183. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., & Jégou, H. (2021, October). Going deeper with image transformers. In *2021 IEEE/CVF international*

conference on computer vision (ICCV) (pp. 32-42). IEEE. https://doi.org/10.1109/ICCV48922.2021.00010

184. So, D., Mańke, W., Liu, H., Dai, Z., Shazeer, N., & Le, Q. V. (2021). Searching for efficient transformers for language modeling. *Advances in neural information processing systems*, *34*, 6010-6022. https://doi.org/10.48550/arXiv.2109.08668
185. Gorodkin, J. (2004). Comparing two K-category assignments by a K-category correlation coefficient. *Computational biology and chemistry*, *28*(5-6), 367-374., https://doi.org/10.1016/j.compbiolchem.2004.09.006
186. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., ... & He, K. (2017). Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*. https://doi.org/10.48550/arXiv.1706.02677
187. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).https://doi.org/10.1109/CVPR.2016.308
188. Müller, R., Kornblith, S., & Hinton, G. E. (2019). When does label smoothing help?. *Advances in neural information processing systems*, *32*. https://proceedings.neurips.cc/paper/2019/hash/f1748d6b0fd9d439f71450117eba2725-Abstract.html
189. Keys, R. (1981). Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, *29*(6), 1153-1160. https://doi.org/10.1109/TASSP.1981.1163711
190. Touvron, H., Vedaldi, A., Douze, M., & Jégou, H. (2019). Fixing the train-test resolution discrepancy. *Advances in neural information processing systems*, *32*. https://proceedings.neurips.cc/paper/2019/hash/d03a857a23b5285736c4d55e0bb067c8-Abstract.html
191. Dwork, C., Feldman, V., Hardt, M., Pitassi, T., Reingold, O., & Roth, A. (2015). The reusable holdout: Preserving validity in adaptive data analysis. *Science*, *349*(6248), 636-638.https://doi.org/10.1126/science.aaa9375
192. Polyak, B. T., & Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, *30*(4), 838-855. https://doi.org/10.1137/0330046
193. Opitz, J., & Burst, S. (2019). Macro f1 and macro f1. *arXiv preprint arXiv:1911.03347*. https://doi.org/10.48550/arXiv.1911.03347.
194. Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010, August). The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition* (pp. 3121-3124). IEEE. https://doi.org/10.1109/ICPR.2010.764
195. Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, *405*(2), 442-451. https://doi.org/10.1016/0005-2795(75)90109-9

196. Jurman, G., Riccadonna, S., & Furlanello, C. (2012). A comparison of MCC and CEN error measures in multi-class prediction. https://doi.org/10.1371/journal.pone.0041882.
197. Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics*, *21*(1), 6.https://doi.org/10.1186/s12864-019-6413-7.
198. Davis, J., & Goadrich, M. (2006, June). The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning* (pp. 233-240). https://doi.org/10.1145/1143844.1143874
199. Boyd, K., Eng, K. H., & Page, C. D. (2013, September). Area under the precision-recall curve: point estimates and confidence intervals. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 451-466). Berlin, Heidelberg: Springer Berlin Heidelberg.https://doi.org/10.1007/978-3-642-40994-3_29
200. Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, *102*(477), 359-378.https://doi.org/10.1198/016214506000000437
201. Naeini, M. P., Cooper, G., & Hauskrecht, M. (2015, February). Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 29, No. 1).https://doi.org/10.1609/aaai.v29i1.9602
202. DeGroot, M. H., & Fienberg, S. E. (1983). The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, *32*(1-2), 12-22. https://doi.org/10.2307/2987588
203. Niculescu-Mizil, A., & Caruana, R. (2005, August). Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning* (pp. 625-632). https://doi.org/10.1145/1102351.1102430
204. Kumar, A., Liang, P. S., & Ma, T. (2019). Verified uncertainty calibration. *Advances in neural information processing systems*, *32*. https://proceedings.neurips.cc/paper/2019/hash/f8c0c968632845cd133308b1a494967f-Abstract.html
205. Nixon, J., Dusenberry, M., Jerfel, G., Nguyen, T., Liu, J., Zhang, L., & Tran, D. (2019). Measuring calibration in deep learning. *arXiv preprint arXiv:1904.01685*. https://doi.org/10.48550/arXiv.1904.01685
206. Altman, D. G., & Bland, J. M. (2005). Standard deviations and standard errors. *BMJ: British Medical Journal*, *331*(7521), 903. https://doi.org/10.1136/bmj.331.7521.903.
207. Agresti, A., & Coull, B. A. (1998). Approximate is better than "exact" for interval estimation of binomial proportions. *The American Statistician*, *52*(2), 119-126.https://doi.org/10.1080/00031305.1998.10480550.
208. Efron, B. (1992). Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics: Methodology and distribution* (pp. 569-593). New York, NY: Springer New York. https://doi.org/10.1214/aos/1176344552

209. Simmons, J. P., Nelson, L. D., & Simonsohn, U. (2011). False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological science*, *22*(11), 1359-1366.https://doi.org/10.1177/0956797611417632
210. Davison, A. C., & Hinkley, D. V. (1998). Bootstrap methods and their applications. *INTERFACES-PROVIDENCE-INSTITUTE OF MANAGEMENT SCIENCES-*, *28*, 71-72. https://doi.org/10.1017/CBO9780511802843
211. Tibshirani, R. J., & Efron, B. (1993). An introduction to the bootstrap. *Monographs on statistics and applied probability*, *57*(1), 1-436. https://doi.org/10.1201/9780429246593
212. Field, C. A., & Welsh, A. H. (2007). Bootstrapping clustered data. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *69*(3), 369-390. https://doi.org/10.1111/j.1467-9868.2007.00593.x
213. Holm, S. (1979). A Simple Sequentially Rejective Multiple Test Procedure. Scandinavian Journal of Statistics, 6(2), 65-70. https://www.jstor.org/stable/4615733
214. Kriegeskorte, N., Simmons, W. K., Bellgowan, P. S., & Baker, C. I. (2009). Circular analysis in systems neuroscience: the dangers of double dipping. *Nature neuroscience*, *12*(5), 535-540. https://doi.org/10.1038/nn.2303.
215. Fagerland, M.W., Lydersen, S. & Laake, P. The McNemar test for binary matched-pairs data: mid-*p* and asymptotic are better than exact conditional. *BMC Med Res Methodol* **13**, 91 (2013). https://doi.org/10.1186/1471-2288-13-91.
216. Altman, D. G., & Bland, J. M. (1995). Absence of evidence is not evidence of absence. *British Medical Journal*, *311*(7003), 485-486. https://doi.org/10.1136/bmj.311.7003.485.
217. Schuirmann, D.J. A comparison of the Two One-Sided Tests Procedure and the Power Approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics* **15**, 657–680 (1987). https://doi.org/10.1007/BF01068419.
218. Demšar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research, 7, 1-30. https://www.jmlr.org/papers/v7/demsar06a.html
219. Nosek, B. A., Ebersole, C. R., DeHaven, A. C., & Mellor, D. T. (2018). The preregistration revolution. *Proceedings of the National Academy of Sciences*, *115*(11), 2600-2606.https://doi.org/10.1073/pnas.1708274114.
220. Hurlbert, S.H. (1984), Pseudoreplication and the Design of Ecological Field Experiments. Ecological Monographs, 54: 187-211. https://doi.org/10.2307/1942661
221. Northcutt, C., Jiang, L., & Chuang, I. (2021). Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, *70*, 1373-1411. https://doi.org/10.1613/jair.1.12125

222. Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, *143*(1), 29-36. https://doi.org/10.1148/radiology.143.1.7063747
223. Peterson, J. C., Battleday, R. M., Griffiths, T. L., & Russakovsky, O. (2019). Human uncertainty makes classification more robust. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9617-9626).https://doi.org/10.1109/ICCV.2019.00971.
224. Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, *12*(10), 993-1001. https://doi.org/10.1109/34.58871
225. Caruana, R., Niculescu-Mizil, A., Crew, G., & Ksikes, A. (2004, July). Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning* (p. 18).
226. Minderer, M., Djolonga, J., Romijnders, R., Hubis, F., Zhai, X., Houlsby, N., ... & Lucic, M. (2021). Revisiting the calibration of modern neural networks. *Advances in neural information processing systems*, *34*, 15682-15694. https://proceedings.neurips.cc/paper/2021/hash/8420d359404024567b5aefda1231af24-Abstract.html
227. Bröcker, J., & Smith, L. A. (2007). Increasing the reliability of reliability diagrams. *Weather and forecasting*, *22*(3), 651-661. https://doi.org/10.1175/WAF993.1
228. Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 427-436). https://doi.org/10.1109/CVPR.2015.7298640
229. Chow, C. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, *16*(1), 41-46. https://doi.org/10.1109/TIT.1970.1054406
230. Geifman, Y., & El-Yaniv, R. (2017). Selective classification for deep neural networks. *Advances in neural information processing systems*, *30*. https://proceedings.neurips.cc/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html
231. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. 2017 IEEE International Conference on Computer Vision (ICCV), 618-626. https://doi.org/10.1109/iccv.2017.74.
232. Zeiler, M.D., Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8689. Springer, Cham. https://doi.org/10.1007/978-3-319-10590-1_53.
233. Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, *1*(5), 206-215. https://doi.org/10.1038/s42256-019-0048-x.

234. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., & Kim, B. (2018). Sanity checks for saliency maps. *Advances in neural information processing systems*, *31*. https://doi.org/10.48550/arXiv.1810.03292.
235. Samek, W., Binder, A., Montavon, G., Lapuschkin, S., & Müller, K. R. (2016). Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, *28*(11), 2660-2673. https://doi.org/10.1109/TNNLS.2016.2599820
236. Reddi, V. J., Cheng, C., Kanter, D., Mattson, P., Schmuelling, G., Wu, C. J., ... & Zhou, Y. (2020, May). Mlperf inference benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)* (pp. 446-459). IEEE. https://doi.org/10.1109/ISCA45697.2020.00045.
237. Bendale, A., & Boult, T. E. (2016). Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1563-1572). https://doi.org/10.1109/CVPR.2016.173
238. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, *46*(4), 1-37.https://doi.org/10.1145/2523813
239. Torralba, A., & Efros, A. A. (2011, June). Unbiased look at dataset bias. In *CVPR 2011* (pp. 1521-1528). IEEE. https://doi.org/10.1109/CVPR.2011.5995347
240. Zech, J. R., Badgeley, M. A., Liu, M., Costa, A. B., Titano, J. J., & Oermann, E. K. (2018). Confounding variables can degrade generalization performance of radiological deep learning models. *arXiv preprint arXiv:1807.00431*.https://doi.org/10.1371/journal.pmed.1002683
241. Elkan, C. (2001, August). The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence* (Vol. 17, No. 1, pp. 973-978). Lawrence Erlbaum Associates Ltd. https://www.ijcai.org/Proceedings/01/IJCAI-2001-e.pdf
242. Zadrozny, B., & Elkan, C. (2002, July). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 694-699).https://doi.org/10.1145/775047.775151
243. See, J. E. (2015). Visual inspection reliability for precision manufactured parts. *Human factors*, *57*(8), 1427-1442.https://doi.org/10.1177/0018720815602389.
244. J. Lu, A. Liu, F. Dong, F. Gu, J. Gama and G. Zhang, "Learning under Concept Drift: A Review," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346-2363, 1 Dec. 2019, doi: 10.1109/TKDE.2018.2876857.
245. Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2017, December). The ML test score: A rubric for ML production readiness and technical debt reduction. In *2017 IEEE international conference on big data (big data)* (pp. 1123-1132). IEEE. https://doi.org/10.1109/BigData.2017.8258038.
246. Roberts, D. R., Bahn, V., Ciuti, S., Boyce, M. S., Elith, J., Guillera-Arroita, G., ... & Dormann, C. F. (2017). Cross-validation strategies for data with temporal,

spatial, hierarchical, or phylogenetic structure. *Ecography*, *40*(8), 913-929.https://doi.org/10.1111/ecog.02881.

247. Melis, G., Dyer, C., & Blunsom, P. (2017). On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*. https://doi.org/10.48550/arXiv.1707.05589.
248. Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, *13*(2). https://www.jmlr.org/papers/v13/bergstra12a.html
249. Piaggio, G., Elbourne, D. R., Pocock, S. J., Evans, S. J., Altman, D. G., & CONSORT Group, F. T. (2012). Reporting of noninferiority and equivalence randomized trials: extension of the CONSORT 2010 statement. *Jama*, *308*(24), 2594-2604.https://doi.org/10.1001/jama.2012.87802
250. Gundersen, O. E., & Kjensmo, S. (2018, April). State of the art: Reproducibility in artificial intelligence. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 32, No. 1). https://doi.org/10.1609/aaai.v32i1.11503.
251. Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., ... & Moher, D. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *bmj*, *372*. https://doi.org/10.1136/bmj.n71