

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF M'HAMED BOUGARRA-BOUMERDES



Faculty of Hydrocarbons and Chemistry

Doctoral Thesis

Presented by

KATEB Yousra

Field: Petrochemical industry

Option: Automation and Control

INTELLIGENT MONITORING OF AN INDUSTRIAL SYSTEM USING IMAGE CLASSIFICATION

(Surveillance Intelligente d'un Système Industriel par la
Classification d'Images)

Before the jury members:

BENDJEGHABA	Omar	Prof	UMBB	President
BOUMEDIENE	Mohamed Said	MC/A	UMBB	Examiner
GROUNI	Said	Prof	U/Tamanghasset	Examiner
BENDIB	Riad	MC/A	U/Skikda	Examiner
MEGLOULI	Hocine	Prof	UMBB	Supervisor
KHEBLI	Abdelmalek	MC/A	UMBB	Co-Supervisor

College year: 2024/2025

Dedication and Sincerity

Without *my mother*, the love of my life, none of this would have been possible. Please accept my deepest gratitude for your unwavering encouragement and friendship throughout my life, and notably throughout my time as a PhD candidate.

To the first man I ever loved, I am always grateful for being both *my father* and the greatest man who ever existed. I have greatly abused your charm, patience, and affection.

Mustapha, my husband and partner, was so valuable for the completion of this endeavor. Your encouragement to keep going and the sacrifices you've made over the years are deeply appreciated. On this historic day, I pray you feel proud of me.

Ghaith and Rizk, my adorable kids, were on my mind constantly while I worked in the library, far away from them, so I dedicate this modest effort to them. Your love for me knows no bounds. You are making a wonderful mother, and I am grateful.

Whoever it was that left us years ago, *my grandmother* will always be with me. I pray that my sisters *Sawsan and Aya* find eternal peace in the holy paradise. For my brother *Riad*, the stronghold of our family, and his lovely great wife "*Soumia*," my sister-in-law and my friend. Their lovely boys, *Anas and Anouar*, too. My sweet old sister, you are my charming and motivational heroine, *Nafissa*. All the little moments that you've shared with me are appreciated.

Thank you for your good advice and love-sharing, *Houda* — my soulmate. I really admire your character. I hope that you and your spouse "*Redwan*" have a wonderful life together with your sweet lovely daughters *Ritadj and Assala*.

Zakaria, my second brother, was great in more ways than one. Thank you so big much.

For being my rock, *Hasnaa*, I am forever honored that you are my lovely little sister. Having a sister like you makes me feel like a really fortunate girl. Also, your husband "*Abdelbasset*" and your gorgeous little prince "*Amir*" — may Allah protect you all.

Djaber, my younger brother, I am overjoyed to bring your name up. Lot of love. *Sahla and Sahl*, my beloved nephews, whom I will cherish eternally, I appreciate your words of support, many thanks.

This thesis is deeply dedicated to my sister-in-law *Imene*, her husband *Kamel*, and their kids, "*Sylia and Samy*". Your help was indispensable for completing this task.

All the best to you! *my husband's mother*, all my family and friends.

Aknowledgment

I would like to express my profound gratitude to my committed supervisor, Professor MEGLOULI Hocine, for their constant support, guidance, and motivation during this academic endeavor.

I am incredibly grateful for their expertise, patience, and unwavering commitment throughout my research journey. Their guidance has been absolutely invaluable in helping me overcome the various challenges and complexities I encountered along the way.

I am also appreciative of Professor KHEBLI Abdelmalek, my co-supervisor, for their insightful advice, encouragement, and leadership. I am deeply grateful for their expertise and invaluable feedback, which played a crucial role in shaping and enhancing my research work.

I want to extend my heartfelt appreciation to the esteemed members of the jury, Professor BENDJEGHABA Omar, Mr. BOUMEDINE Mohamed Said, Professor GROUNI Said and Mr. BENDIB Riad, for generously dedicating their time, providing valuable insights, and offering constructive criticism during the evaluation of my Ph.D. thesis.

I am deeply grateful for your expertise and guidance, which have played a crucial role in shaping the quality and depth of my research work.

I ought to express my sincere gratitude Mr. Cherif who has provided unwavering support and endless inspiration throughout my journey in completing this Ph.D. thesis.

I am deeply grateful to each and every one of you for your invaluable contributions to this important chapter in my academic and professional journey.

Sincerely,

Abstract

The importance of automation in saving time and money while reducing human intervention is a major reason why automated industries have recently experienced massive growth across all sectors worldwide. However, there are a lot of obstacles for these sectors to overcome when it comes to checking for flaws in the finished product. In this case, intelligent monitoring systems seemed to aid in the automatic and less-human-intervention recognition of defects in the quality of these products. But there are far too many critical components missing from these systems, which causes production to be either late or inefficient. Inadequate execution time, high costs, and insufficiently powerful processors utilized for monitoring are the primary challenges encountered in this domain. This thesis employs a newly-developed convolutional neural network, NasNet-Mobile, in an effort to circumvent these restrictions. In this study, we utilized a deep neural network that has been trained to recognize six different defects on steel surfaces. The results were quite promising, and the suggested method allowed for the classification of six different kinds of defective images in industrial steel. Despite using 27 times less data than other research datasets, the model nonetheless managed an accuracy of 99.51%, surpassing previous state-of-the-art flaw assessment methodologies. The study's importance lies in the fact that it will aid intelligent industry monitoring in situations where the faulty image dataset is limited, the software is not very promising, or time is of the essence. Once these problems are avoided, the steel industry will be able to decrease costs and time spent on the whole process while simultaneously improving worker safety and product quality.

Keywords: *Intelligent Monitoring Systems; NasNet-Mobile; CNN; computer vision, Image classification; Steel Surface Defect inspection.*

ملخص

أهمية الأتمتة في توفير الوقت والتكاليف مع تقليل التدخل البشري تُعدّ من الأسباب الرئيسية للنمو الهائل الذي شهدته الصناعات المؤتمتة في جميع القطاعات حول العالم مؤخرًا. ومع ذلك، تواجه هذه القطاعات العديد من العقبات عند فحص العيوب في المنتجات النهائية. في هذا السياق، ظهرت أنظمة المراقبة الذكية كوسيلة للمساعدة في التعرف التلقائي على العيوب بجودة المنتجات مع تدخل بشري أقل. ولكن لا تزال هناك العديد من المكونات الأساسية المفقودة في هذه الأنظمة، مما يؤدي إلى تأخير الإنتاج أو تقليل كفاءته. تُعدّ محدودية وقت التنفيذ، وارتفاع التكاليف، وضعف قدرة المعالجات المستخدمة في المراقبة من التحديات الرئيسية في هذا المجال.

تعتمد هذه الأطروحة على شبكة عصبية التلافيفية مطوّرة حديثًا، وهي NasNet-Mobile، بهدف التغلب على هذه القيود. في هذه الدراسة، تم استخدام شبكة عميقة تم تدريبها على التعرف على ستة أنواع مختلفة من العيوب على أسطح الفولاذ. وقد أظهرت النتائج أداءً واعدًا، حيث مكّن المنهج المقترح من تصنيف ستة أنواع مختلفة من الصور المعيبة في الصناعات الفولاذية. وعلى الرغم من استخدام بيانات أقل بمقدار 27 مرة مقارنةً بمجموعات البيانات الأخرى في الأبحاث السابقة، فقد حقق النموذج دقة بلغت 99.51%، متفوقًا على أحدث منهجيات تقييم العيوب في هذا المجال.

تكمن أهمية هذه الدراسة في أنها تساهم في تحسين أنظمة المراقبة الذكية في الحالات التي تكون فيها مجموعة بيانات الصور المعيبة محدودة، أو البرمجيات غير متطورة بالشكل الكافي، أو يكون الوقت عنصرًا حاسمًا. وبمجرد التغلب على هذه المشكلات، سيتمكن قطاع صناعة الفولاذ من تقليل التكاليف والوقت المستغرق في العملية ككل، مع تحسين سلامة العمال وجودة المنتج في الوقت نفسه.

كلمات مفتاحية:

أنظمة المراقبة الذكية; *NasNet-Mobile*; تصنيف الصور; فحص عيوب أسطح الفولاذ; الشبكات العصبية التلافيفية

Résumé

L'importance de l'automatisation pour améliorer l'efficacité et la rentabilité, tout en réduisant au minimum la participation humaine, a entraîné une augmentation considérable des industries automatisées dans divers secteurs à l'échelle mondiale. Cependant, ces industries doivent surmonter un certain nombre d'obstacles pour trouver des défauts dans le produit fini. Dans ce scénario particulier, la mise en oeuvre de systèmes de surveillance intelligents semble contribuer grandement à l'inspection automatisée des défauts de la qualité des produits tout en réduisant la nécessité d'une intervention humaine excessive. Cependant, ces systèmes manquent de plusieurs composants essentiels, ce qui entraîne des retards et des inefficacités de la production. Le domaine fait face à des défis importants, notamment un temps d'exécution limité, des coûts élevés et l'utilisation de processeurs qui manquent de puissance suffisante à des fins de surveillance.

Cette thèse utilise un réseau neuronal convolutionnel récemment développé, NasNet-Mobile, dans une tentative de surmonter ces limitations. Notre étude a impliqué l'utilisation d'un réseau neuronal profond personnalisé qui a fait l'objet d'une formation approfondie pour classer avec précision six défauts distincts trouvés sur les surfaces d'acier. L'algorithme a été amélioré en utilisant la fonction d'activation de l'ELU au lieu de ReLU pour résoudre le problème des neurones en train de mourir. Les optimisateurs ont été changés entre ADAM et ADAMAX pour trouver le meilleur. De nombreuses autres couches ont été ajoutées pour améliorer l'apprentissage du modèle, comme Dropout et le Global Average Pooling. Les résultats ont été très prometteurs, car l'approche proposée a réussi à classer six types différents d'images défectueuses dans le domaine de l'acier industriel. Même avec beaucoup moins de données que d'autres ensembles de données de recherche, le modèle a obtenu une précision impressionnante de 99,51 % avant le réglage fin et de 100 % après le réglage fin, dépassant les méthodes précédentes d'évaluation des défauts de pointe. L'importance de cette recherche réside dans son potentiel d'améliorer la surveillance intelligente de l'industrie dans des scénarios difficiles, tels que des échantillons d'images défectueux limités, des logiciels moins puissants ou des situations sensibles au temps. En répondant à ces défis, l'industrie de l'acier peut réduire les coûts et le temps consacré à l'ensemble du processus tout en améliorant la sécurité des travailleurs et la qualité des produits.

Mots-clés : *Systèmes de surveillance intelligents ; NasNet-Mobile ; Réseaux de neurones convolutifs (CNN) ; Classification d'images ; Inspection des défauts de surface de l'acier.*

Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
ASIS	Automated Surface Inspection Systems
AVI	Automatic Visual Inspection
BP	Back-Propagation
CCD	Charge-Coupled Device
CMOS	Complementary Metal Oxide Semi-Conductor
CNN	Convolution Neural Network
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
ELU	Exponential Linear Unit
FCDNN	Fully Connected Deep neural network
GAN	Generative Adversarial Network
GbpS	GigaBytes per Second
GPU	Graphic Processing Unit
HOG	Histogram of Oriented Gradients
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
KNN	K-Nearest Neighbors
LED	Light Emitting Diode
LSTM	Long Short-Term Memory
MSE	Mean Square Error

ML	Machine Learning
MLP	Multi Layer Perceptron
NAS	Neural Architecture Search
NDT	Non-Destructive Testing
NN-BP	Neural Network-Back-Propagation
RBF	Radial Basis Function
ReLU	<i>Rectified Linear Unit</i>
RNN	Reccurent Neural Network
SLP	Single Layer Perceptron
SOM	Self-Organizing Map
SVM	Support Vector Machine
TL	Transfer Learning

CONTENT

Abstract	III
List of Content	VIII
List of Figures	XII
List of Tables	XIV
GENERAL INTRODUCTION.....	1
1 CHAPTER I : LITERATURE REVIEW	8
1.1 Introduction	8
1.2 Manual Inspection.....	8
1.3 Machine Learning-Based Surface Inspection Algorithms.....	9
1.3.1 Subfields of Machine Learning	9
1.3.1.1 Supervised Machine Learning Algorithms.....	9
1.3.1.1.1 Regression	10
1.3.1.1.2 Classification	10
1.3.2 Different Feature Extraction algorithms used for Images Classification.....	12
1.3.2.1 Texture feature-based classifiers	12
1.3.2.2 Grey Level Covariance Matrix GLCM and Discrete Shearlet Transform.....	12
1.3.2.3 Support Vector Machines SVM	135
1.3.3 Distance Function Classifiers.....	16
1.3.3.1 K-Nearest-Neighbor KNN	16
1.3.4 Classification based on Neural Networks NN.....	18
1.4 Deep-Learning-based surface inspection algorithms.....	19
1.4.1 Subfields of Deep Learning.....	19
1.4.1.1 Supervised learning in DL	19
1.4.1.1.1 SqueezeNet	19
1.4.1.1.2 ResNet	20
1.4.1.1.3 Visual Geometry Group VGG.....	23
1.4.1.1.4 Xception	25
1.4.1.1.5 U-Net CNN.....	26
1.4.1.1.6 MobileNets, a CNN Architecture for Mobile Devices.....	28
1.4.1.2 Unsupervised learning methods	30
1.4.1.2.1 Types of Unsupervised Learning	31
1.4.1.2.2 Different algorithms of unsupervised learning	32
1.4.1.2.3 Advantages and disadvantages of unsupervised learning	34
1.4.1.3 Reinforcement Machine Learning.....	35

1.4.1.3.1	Types of Reinforcement Machine learning.....	35
1.4.1.3.2	Semi-supervised (or Weakly supervised) learning methods.....	36
1.5	Conclusion	39
2	CHAPTER II: ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, DEEP LEARNING OVERVIEW	40
2.1	Introduction	40
2.2	Artificial Intelligence Definition.....	40
2.3	Machine Learning Overview.....	41
2.4	The history of Artificial Neural Networks ANNs	43
2.4.1	Biological neural network.....	44
2.4.2	Definition of artificial neural network ANN	45
2.4.2.1	Modeling a formal neuron	47
2.4.2.2	Activation function types.....	49
2.4.2.2.1	Sigmoid activation function	50
2.4.2.2.2	Tanh activation function	50
2.4.2.2.3	Softmax activation function	51
2.4.2.2.4	Rectified Linear Unit ReLU	51
2.4.2.2.5	Exponential Linear Unit ELU.....	52
2.4.2.3	ANN Architecture Design	55
2.4.2.3.1	Recurrent neural networks	55
2.4.2.3.2	Feed-forward neural networks	55
2.4.2.3.3	Perceptron definition	55
2.4.2.3.4	The single-layer perceptron	56
2.4.2.3.5	Multi-layer perceptron MLP.....	56
2.4.2.4	ANN model selection.....	57
2.4.2.5	The importance of Generalization in Machine Learning.....	58
2.4.2.6	Concept of Forward propagation and Back-propagation in neural networks	59
2.4.2.7	Optimizers	63
2.4.2.8	Hyperparameters Tuning	65
2.4.2.8.1	Different types of hyperparameters	65
2.4.2.8.2	Learning rate selection α	65
2.4.2.8.3	Adding the Momentum term μ	66
2.4.2.8.4	Initial Weight Values	67
2.5	Deep Learning Overview	68
2.5.1	Applications of Deep Learning	69
2.5.1.1	Computer vision	69
2.5.1.2	Natural Language Processing	69
2.5.1.3	Reinforcement Learning RL	70

2.5.2	Deep Learning Model Architectures	70
2.5.2.1	Fully connected Deep neural network (FCDNN)	70
2.5.2.2	Recurrent Neural Networks RNNs	71
2.5.3	Advantages and Disadvantages of Recurrent Neural Networks	73
2.5.4	Challenges faced in Deep Learning	73
2.5.5	Advantages and disadvantages of Deep Learning.....	73
2.6	Difference Between Artificial Intelligence, Machine Learning, and Deep Learning	74
2.7	Conclusion	75
3	Chapter III : CONVOLUTION NEURAL NETWORK	Erreur ! Signet non défini.
3.1	Introduction	77
3.2	Convolutional Neural Network CNN definition.....	77
3.3	The layers of CNNs with their operating mode.....	78
3.3.1	Kernel or the filter importance	79
3.3.2	Different convolution layers.....	80
3.3.2.1	The convolution layer.....	81
3.3.2.2	Depth.....	83
3.3.2.3	Stride	83
3.3.2.4	Zero-Padding	84
3.3.2.5	Correction layer.....	88
3.3.2.6	Pooling layer.....	89
3.3.2.7	Flattening layer.....	91
3.3.2.8	Fully-connected layer	91
3.3.2.9	Batch Normalization.....	92
3.3.2.10	Loss layer (LOSS).....	93
3.3.3	Training a Convolutional Neural Network.....	93
3.3.3.1	Dataset preparation	93
3.3.3.2	Overfitting	94
3.3.3.3	Underfitting.....	97
3.3.3.4	Performance evaluation metrics.....	98
3.3.3.5	Confusion Matrix.....	98
3.3.3.5.1	Area Under Curve (AUC)	101
3.3.3.5.2	Receiver Operating Characteristics (ROC) Curve	102
3.3.3.5.3	AUC-ROC curve.....	102
3.3.3.6	Different Types of CNN Architectures.....	103
3.4	Conclusion	109
4	Chapter IV : STEEL MAKING PROCESS AND DEFECT INSPECTION IN STEEL SHEET.....	112
4.1	Introduction	112

4.2	Definition of Steel.....	112
4.3	Importance Of Steel Surface Production	113
4.4	Economic And Industrial Impact Of Steel Production.....	113
4.5	Steel making process.....	115
4.5.1	Steel Processing And Finishing	116
4.5.2	Different steel surface categories	117
4.5.3	Quality inspection in steel surfaces.....	119
4.5.3.1	Key Components Of The Automatic Surface Inspection System's Hardware Setup	120
4.5.3.2	Different Types Of Defects In Industrial Steel Surfaces	123
4.5.3.3	Defect inspection algorithms of steel surface.....	125
4.5.3.4	Challenges and Solutions	126
4.6	Conclusion	128
5	Chapter V : EXPERIMENTATIONS, RESULTS AND DISCUSSION	12930
5.1	Introduction	129
5.2	Approach Description.....	130
5.2.1	Dataset Praparation	130
5.2.2	Image Preprocessing	1312
5.2.3	Image Data Generator And Data Augmentation.....	131
5.2.4	Classification model—Improved NASNet-Mobile	133
5.2.4.1	MobileNet	133
5.2.4.2	NASNet Framework.....	136
5.2.4.3	Developed NASNet-Mobile for Steel Sheet Inspection.....	136
5.2.4.4	Model Optimization For Steel Surface Defect Classification.....	138
5.2.4.5	Model Implementation	139
5.3	Results And Discussions	140
5.3.1	Model Evaluation	140
5.3.1.1	Test results visualization	144
5.3.1.2	Comparative Study.....	145
5.4	GENERAL CONCLUSION AND PERSPECTIVES.....	147

List of Figures

Figure 1-1 Schematic diagram of the method DST–GLCM used in [8]	13
Figure 1-2 SVM max separation hyperplane with margin.	15
Figure 1-3 : K-nearest neighbor classifier KNN	17
Figure 1-4 Organization of the convolution filters in a Fire module	20
Figure 1-5 Residual block, the basic building block of ResNet.....	21
Figure 1-6 VGG vs. 34-layer Plain CNN vs. 34-layer ResNet.....	22
Figure 1-7 VGG16 network architecture diagram [37].....	24
Figure 1-8 Model outline Diagram [38].....	24
Figure 1-9 Xception overall architecture	26
Figure 1-10 U-NET architecture	27
Figure 1-11 MobileNet architecture.	28
Figure 1-12 Depthwise and pointwise operations.....	29
Figure 1-13 Operation of the designed AutoML model used in [47].....	30
Figure 1-14 GAN training example [17].....	32
Figure 1-15 Autoencoder framework	33
Figure 1-16 Diagram of the outline of different DL techniques. [31].....	38
Figure 2-1 Biological neuron morphology [65].	45
Figure 2-2 An artificial neuron and a biological neuron.....	46
Figure 2-3 Mathematical modeling of a neuron xi	47
Figure 2-4 Sigmoid function	50
Figure 2-5 Tanh activation function graph.....	51
Figure 2-6 The output of a rectified linear unit is 0 across half of its domain [67].....	52
Figure 2-7 ELU activation function (left) and its derivative (right)	52
Figure 2-8 Single-layer perceptron.....	56
Figure 2-9 Architecture of a multi-layer perceptron (MLP).....	57
Figure 2-10 Generalization and training errors as capacity-related functions. There is an ideal capacity between the underfitting and overfitting zones.....	59
Figure 2-11 A simple neural network with forward propagation.....	60
Figure 2-12 Errors vs training steps.....	64
Figure 2-13 Optimization using Gradient Descent.....	64
Figure 2-14 Difference between using big learning rate and small learning rate	66
Figure 2-15 Fully connected Deep neural network	71
Figure 2-16 Recurrent Neural Network work-flow.....	72
Figure 3-1 A convolutional neural network model with different layers.	78
Figure 3-2 Representation of an image with RGB colors.....	79
Figure 3-3 Example of values of a matrix used as a filter.....	79
Figure 3-4 The pathway of the kernel moving across different parts of the image.....	80
Figure 3-5 Convolution operation example.....	81
Figure 3-6 Operation of Convolution using four filter results in four feature maps.....	82
Figure 3-7 An image with different stride values : 0, 1, 2 respectively from the left to the right.....	84
Figure 3-8 Padding the borders with zeros	84
Figure 3-9 Calculating the output values of a convolution operation.....	87

Figure 3-10 Feature map correction using ReLU	88
Figure 3-11 The application of (2×2) max pooling in (4×4) image	89
Figure 3-12 Average Pooling example	90
Figure 3-13 A Sum pooling with Image size (4×4) , Filter size (3×3) and Strides= 1 [73]	90
Figure 3-14 Flatten layer operation.....	91
Figure 3-15 A network of fully-connected layers. The dense block's final output is a membership distribution over classes.....	92
Figure 3-16 Overfitting, appropriate fitting and underfitting graphs.	94
Figure 3-17 Different data augmentation techniques.....	95
Figure 3-18 Early stopping before the model overfits.	96
Figure 3-19 Dropping out random nodes.	97
Figure 3-20 Confusion Matrix.....	99
Figure 3-21 Area Under the Curve AUC	101
Figure 3-22 AUC ROC curve	102
Figure 3-23 The architecture of Neocognitron network.....	103
Figure 3-24 The architecture of LeNet-5.....	104
Figure 3-25 AlexNet architecture[79].....	105
Figure 3-26 ML-related training steps.	106
Figure 3-27 Main NAS setups.....	106
Figure 3-28 Global search space (left) and cell-based search space.....	107
Figure 3-29 Reinforce method to train controller network.....	108
Figure 3-30 Difference between Traditional ML and Transfer Learning.	109
Figure 4-1 Steel making process.....	115
Figure 4-2 Rolled Metal Products : Steel Profiles and Tubes	117
Figure 4-3 Different types of steel products [31].....	118
Figure 4-4 Some final steel products made from rolling.	119
Figure 4-5 ASIS basic hardware structure [1].....	120
Figure 4-6 Surface inspection system for steel strips based on machine vision.	123
Figure 4-7 Sample photos of six types of typical surface defects.	124
Figure 5-1 Some images from the NEU-CLS dataset [90].....	130
Figure 5-2 Data augmentation of different defect types	132
Figure 5-3 Data preprocessing using flipping, rotating and Gaussian Noise.....	132
Figure 5-4 Depthwise Separable Convolution	133
Figure 5-5 Depthwise Convolution.....	134
Figure 5-6 Pointwise convolution transforms a three-channel image into a one-channel image.....	134
Figure 5-7 Diagrammatic explanation of Depthwise Separable Convolutions.....	135
Figure 5-8 General structure of the proposed approach.....	136
Figure 5-9 Graph demonstrating the distinction between ELU (green) and ReLU (red) activation functions.....	137
Figure 5-10 Accuracy and loss curves before fine-tuning the NASNet-Mobile	141
Figure 5-11 Precision and accuracy curves before fine-tuning the NASNet-Mobile.....	142
Figure 5-12 Training and validation Loss	143
Figure 5-13 accuracy and loss after fine-tuning the Model	143
Figure 5-14 Precision and recall loss in the fine-tuned Model.....	143
Figure 5-15 The model predicts the defects correctly for unseen data.	144
Figure 5-16 The model predicts some faults incorrectly (red) and correctly (green).	145

List of Tables

Tableau 1-1 Summary of the advantages and disadvantages of different DL techniques.	38
Tableau 2-1 Comparison of the formal neuron with the biological neuron.....	46
Tableau 2-2 Summary of different activation functions and their derivatives[67].....	54
Tableau 2-3 key differences between Artificial Intelligence, Machine Learning and Deep Learning.....	74
Tableau 4-1 Different defect categories according to steel surface types.	125
Tableau 5-1 Hyper-parameters used to train NASNet-Mobile Convolutional Neural Network.	139
Tableau 5-2 Performance metrics of NASNet-Mobile in the training and validation dataset before fine-tuning.....	142
Tableau 5-3 Performance metrics of NASNet-Mobile in the training and validation dataset after fine-tuning.....	142
Tableau 5-4 The classification accuracy (%) for multiple state-of-the-art steel surface defect classifiers, taking into consideration the triplet model lightness, execution time, and data size.	146
Tableau 5-5 Evaluation of Classification errors depending on the activation function, the model employs the name of the function enclosed in parenthesis.	146

General Introduction

The industry is a collection of methods, processes, and/or actions carried out on raw materials with the goal of producing completed goods. The industry, on the other hand, needs not just raw materials but also a big quantity of energy, specialized machinery, a huge number of human resources, a significant amount of capital investment, and excellent marketplaces with high purchasing power in order to achieve the pointed out aim. According to economic discipline, industries are classified as heavy or light based on the degree of complexity of the manufacturing processes. The former is in charge of converting raw materials into semi-finished products, and they are distinguished by a higher concentration of capital and ongoing technological renewal. Steel, metallurgy, and petrochemicals are commonly considered. Capital goods industries, on the other hand, use semi-finished products from basic industries to make machinery and tools for other sectors, such as construction and mechanical. The light or processing industries are in charge of generating commodities that can be consumed directly by the populace. Food, textile, chemical, electronic, automotive, and marine industries are all included.

The steel industry is one of the most crucial industries globally. The steel industry is a global enterprise that produces both raw steel and finished products. Steel is an iron alloy and is commonly linked to the iron industry. The steel sector's current prosperity is closely tied to the economic achievements of developed as well as developing countries, owing to its significant role in the Industrial Revolution.

Steel has played a significant role in both the developed and developing worlds throughout its history. Because it is so widely used in building and other applications, the state of the sector may be utilized to evaluate overall world economic success. If the steel sector is in good condition and there is a strong demand for its goods, this might be seen as a positive indicator for the world economy.

However, the steel industry faces significant risks due to high-temperature thermal or chemical transformations, which pose significant risks to worker safety. Inspection and maintenance of machinery and structures are also challenging due to corrosion, wear, and tear. Hence, the work environment becomes unsafe and stressful

for employees, with strict time limitations and a limited ability to perform critical tasks. To address these issues, the steel industry should prioritize the development and implementation of innovative solutions, such as advanced inspection techniques and remote steel diagnosis. This will empower the workforce with tools and technologies that reduce stress and improve productivity, ultimately contributing to the industry's long-term sustainability and competitiveness.

As mentioned above, advanced inspection techniques can help overcome the problem of human inspection in such hard situations. The main advanced method used to diagnose steel during the process is called non-destructive testing (NDT).

Non-Destructive Testing (NDT) is used by steel engineers to assure high-quality products. NDT is a non-destructive testing process that does not harm the examined object permanently. It identifies subsurface faults and abnormalities without compromising the integrity of the component, making it vital for quality control on critical items such as steel components. There are various non-destructive testing procedures that are frequently utilized to ensure the quality of steel components. Automatic Visual Inspection (AVI) with computer vision, or visual inspection, is a primitive non-destructive testing method that has evolved from manual to automated processes. It's particularly effective for detecting defects in steel components and controlling dimensions. It's often combined with machine learning algorithms for efficient inline inspection, making it a more efficient method than other NDT methods.

There are other inspection methods, like, for example, : Ultrasonic Testing (UT), Eddy Current Testing (ECT), Radiography Testing (RT), Magnetic Particle Inspection (MPI), ...etc. Every one of these techniques is convenient to a particular task, depending on the specific situation or the case in use. We are interested in our thesis about the visual inspection techniques, their advantages and disadvantages, and how to overcome the challenges that steel inspection faces nowadays, like high cost, long inspection time, and impowerful processing units.

Motivation

It is critical to have an effective method of continually inspecting the goods flowing through the manufacturing line in large-scale metal sheet processes. Because it does not meet the stated specifications, a metal sheet with significant flaws must be removed, disposed of, or recycled. Currently, the sheets are inspected manually, which is time-consuming, costly, and, in certain circumstances, unclear because the errors are difficult to identify in the hard production environment. However, recent improvements in computer vision provide exciting opportunities to overcome this problem using deep learning and real-time data. Deep learning enables scholars as well as scientific investigators to train Deep Neural Networks (DNNs) to perform complex tasks, including anomaly detection and visual inspection. Therefore, they are well-suited for automated recognition and classification systems that consistently analyze images of metal sheets on the production line.

The main reason that motivated us to work on this thesis is to help small and powerless steel mills make the final steel product valid and adequate, just like big industries do.

Problem Statement

Steel metal is a prominent product within the iron and other metal sectors, making up over 65% of all products. It serves as a crucial material for various industries like architecture, aircraft, machinery, and automobiles. If flat steel were to have any quality issues, it could lead to significant economic and reputational consequences for steel manufacturers. The primary threat to product quality in thin and wide flat steel is surface flaws. Even with just a few internal imperfections, there is a high probability of morphological abnormalities on the surface. Automated Visual Inspection (AVI) equipment, which emphasizes surface quality, is increasingly being adopted by flat steel mills to improve product quality and boost production efficiency. Identifying surface defects on-site in real-world steel factories can be quite challenging for several logical reasons:

1. Inadequate imaging environments. High temperatures, plenty of cooling water droplets and dense mist [1], uneven lighting, stochastic sounds [2, 3], and aperiodic vibration [4] are all problems on continuous casting and rolling manufacturing lines. The unsatisfactory image quality necessitates superior detection algorithms that can tolerate substantial intra-class fluctuation and modest inter-class distance [2, 5].
2. Long image streams. For normal flat steel mills to do online dual-surface inspections, the surface AVI instrument needs to handle about 2.56 GbpS of image flows all the time [1]. In order to recognize faulty images, inspection algorithms need to strike a fine balance between precision, computational complexity, and trustworthiness.

Research Questions

To solve the problems we've already talked about, we need to ask some important questions. Here's how we can cite them:

- Can we develop a surface defect inspection technique using computer vision and deep learning algorithms ?
- How can we tackle the algorithm long execution time issue, which in fact delays the inspection time in the production line ?
- What are the alternative settings when the number of defected steel images is not sufficient, which is so familiar in small industries that do not own very sophisticated cameras ?
- Can we find a CNN architecture for defect inspection that gives satisfying accuracy even though the processing units are powerless ?

Research Objectives

Over time, both industry and academia have worked together to overcome the aforementioned issues, which range from hardware upgrades to algorithmic optimization. Due to the challenges of Moore's law (Moore's Law says that the number of transistors on a chip is going to double about every two years, but the price will merely go up just a bit), it's hard to see big improvements in hardware in such a brief

period of time. In order to complete this assignment, we aim to achieve the following objectives in our thesis :

The initial objective is to create a flaw identification technique for surface inspection. The approach must be broad enough to inspect a variety of defects and be usable in many different types of quality inspections.

The second objective is to select the dataset and preprocess it to fit the developed model. We use Image-Data-Generator from Tensorflow and Keras to apply different changes like rotation, horizontal flipping, shearing, and zooming. This method allows the model to learn how to extract deeper features from different images.

The third objective is to design a CNN architecture that fits the needs of steel inspection operations. We talk here about severe conditions like illumination and the lack of a dataset.

This study aims to correctly classify defects in steel sheet images using advanced deep learning algorithms and image preprocessing techniques. In order to achieve this, we will train and assess our developed convolutional neural networks on a publicly accessible dataset that includes different steel defect types. To enhance the accuracy of these models, we will implement optimization, transfer learning, regularization, and preprocessing techniques.

Significance of the Research

This thesis provides valuable insights into surface defect classification in scenarios with a limited sample size, less powerful software, or time constraints. Addressing these issues will enhance safety and product quality in the steel industry, leading to time and cost savings. Ensuring quality in the steel industry is crucial for preventing damaged products from reaching the client and for identifying defect types and reasons. This will lead to a deeper understanding of production techniques and, consequently, enhance steel manufacturing processes. Improving manufacturing and production procedures is essential for reducing the cost of low-quality products.

Contributions

The main contributions of this thesis can be summarized below:

- 1- We propose a novel algorithm based on Network Architecture Search NAS and MobileNet convolution neural networks. This algorithm will help the MobileNet to find the best architecture design and the best hyperparameters to increase accuracy and reduce time in addition of using less data samples.
- 2- We evaluate the influence this architecture search algorithm on the training and testing dataset, it shows that the use of this algorithm improves the classification accuracy and reduces execution time;
- 3- We apply our approach to the NEU steel dataset and report a significant improvement over the benchmark results on the same database.

Thesis Outline

This research will focus on the creation of a deep learning-based system for classifying surface flaws in steel sheets. In this paragraph, we will outline broadly the key chapters that comprise our thesis; we will provide a summary of each chapter and how it contributes in this work.

There are four main chapters, organized as follows:

In Chapter 1, the most commonly used NDT techniques for steel surface inspection are thoroughly reviewed. The focus is on evaluating their appropriateness for the task at hand and providing a comprehensive overview of the pros and cons of each technique discussed.

Chapter 2 describes the background of artificial intelligence, its history and applications, and gives a general idea of deep learning, its applications, its subfields, its advantages and disadvantages in real-world applications.

Chapter 3 explains the appearance of convolutional neural networks known as CNNs and how they work, their parameters and architecture in detail, as well as their importance in computer vision tasks and especially image classification.

Chapter 4 explains the discovery of steel and how it is made in industry, as well as its importance in our society. The main defects that can occur during steel production and their reasons will be mentioned in this chapter.

Chapter 5 reviews the experimentation part, more specifically materials such as data acquisition, image collection techniques, and preprocessing. Hence, we give a view of the implemented algorithm gradually with details, from the selection of the model to the implementation details. As we discuss and analyze the results obtained during this research.

Finally, we conclude this thesis, we answer the question previously asked, and we present some perspectives for future works.

1 CHAPTER I

LITERATURE REVIEW

1.1 Introduction

In industrial manufacturing, product quality inspection is vital, and manual inspection was once the most prevalent approach. It is, however, subjective and not very accurate. In recent years, visual perception technology has mostly replaced manual inspection and has become an essential component in inspecting steel surface quality and classifying its defects. This non-contact automated inspection system has a high level of precision and can function in complicated manufacturing environments, lowering labor costs and increasing production efficiency. In this chapter, we will describe the different methods available to inspect surface defects in steel products.

1.2 Manual Inspection

In the past, experts would manually assess the surface quality of products in coil form by cutting around 30 meters of coils at random in a batch for inspection. When conducting manual inspection, the surface being examined usually accounts for approximately 0.05% of the entire steel surface sheet. Operators stationed in cold rolling mill complexes test the final product online for any defects. However, the inspection process is not ideal due to the fast line speed, fatigue, and other drawbacks. Due to the limitations of human inspection, there arose a need for automated surface inspection to ensure defect-free steel sheets with a reliable level of accuracy [6].

1.3 Machine Learning-Based Surface Inspection Algorithms

Machine learning is a discipline of computer science that employs statistical techniques to estimate complicated functions from data. There are two main machine learning approaches: supervised and unsupervised learning. Supervised learning requires providing inputs as well as the desired outputs, whereas unsupervised learning does not. Machine learning works well for modeling difficult-to-form situations and using learning algorithms to replace human programming. Machine learning is becoming highly practical and virtually ubiquitous as processing capability and data availability have increased. It has evolved into a valuable tool for modeling complicated issues [6].

With this in consideration, Tom Mitchell provides a good definition of machine learning in the book he wrote entitled "*Machine Learning*":

"A computer program is said to learn from experience E with respect to some class of tasks T, and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." [7]. Here the algorithm's learning target is task T and the performance measure P is the metric used to assess the algorithm's ability. The experience E includes the supplied dataset as well as any additional information from which the machine learning system can learn.

Traditional machine learning algorithms fall into three categories: texture, color, and shape feature-based approaches. In our work, we don't have an interest in color-based approaches, as the images we took into consideration are gray-level.

1.3.1 Subfields of Machine Learning

Machine learning algorithms learn with specific goals. These goals can be broken down into three main groups: supervised learning, unsupervised learning, and reinforcement learning. Another kind of learning is semi-supervised learning, which is a mix of supervised and unsupervised learning.

1.3.1.1 Supervised Machine Learning Algorithms

Explaining supervised learning is straightforward: you collect a set of examples for a specific task, each consisting of an input and the desired output. The goal is to

develop a system that can accurately predict the output for any new input it faces. Classifying objects in a pool using supervised learning is achievable by leveraging the available features, attributes, or annotations. For instance, after students finish an exam and receive feedback from the teacher on the questions they got wrong, it is a common type of supervised learning. It is essential for students to understand how to answer such questions accurately after receiving the correct answers.

There are typically two categories of issues that supervised learning is used for. These tasks vary primarily based on the types of pairs (input, output) they are associated with: regression and classification.

1.3.1.1.1 Regression

It is a statistical method used to examine how one or more than one independent variables (variables used for making predictions) and a dependent variable (target variables) are related. Finding the best function that best describes the relationship between these variables is the goal. It looks for the model that fits the data the best so that it may be used to forecast or draw inferences. Because it is supervised learning, model training necessitates labeled data. Next, the dependent variable's value is predicted by the model with brand-new, unobserved data. When the output variable, such "salary" or "weight," is a real or continuous value, regression analysis is required. In our thesis, we work on a classification task, so we will not dive deeper into regression.

1.3.1.1.2 Classification

Classification is a sort of supervised machine learning method that trains a model to predict a categorical class label for newly acquired data points. Put differently, fresh data points are classified into one of a predetermined number of groups using classification algorithms.

Dealing with category output variables like "red or blue" or "defected or no-defected" can lead to classification problems. A classification model looks at observable data and makes an attempt to infer a conclusion. A classification model will attempt to predict the value of one or more outputs given one or more inputs. For instance, deciding whether to classify emails as "spam" or "not spam," or as

"fraudulent" or "authorized" while examining transaction data. To put it briefly, classification uses the training set and the values (class labels) in the classifying features to either build a model or forecast categorical class labels. The model is then applied to the new data to be classified.

Classification can take several distinct forms: Binary classification, multiclass classification, multi-label classification and unbalanced classification.

- **Binary classification:** it aims to forecast if a new data piece belongs to one of two distinct classes. An algorithm for binary classification, for instance, might be used to identify if an email is spam or not, or to identify whether a picture of handwritten numbers belongs to the digit 1 or the digit 0.
- **Multiclass classification:** it is the process of predicting which of many classes a given data piece belongs to. A multiclass classification method, for instance, might be used to categorize text materials into many categories, such as news, entertainment, or sports, or to identify photographs of handwritten numbers as belonging to one of the digits 0–9.
- **Multi-label classification:** it aims to determine which of many labels a new data piece belongs to. This differs from multiclass classification, in which each data point may only be assigned to one class. For example, a multi-label classification method might be used to categorize animal photos as cat, dog, bird, or fish.
- **Unbalanced Classification:** Unbalanced classification attempts to predict whether a newly added data point will be assigned to the minority class, notwithstanding the significantly larger number of instances in the majority class. For example, a medical diagnosis algorithm might be used to forecast if a patient has a rare condition, despite the fact that there are many more patients with common diseases.

➤ Application of Classification

Classification applications include spam filtering, fraud detection, medical diagnosis, image recognition, and recommendation systems, which help identify spam

emails, detect fraudulent transactions, diagnose diseases, recognize objects in images, and recommend products or services.

Classification offers advantages such as ease of understanding, handling diverse data types, and suitable for binary and multiclass classification tasks, but may be sensitive to outliers, require large labeled data, and not suitable for complex relationships between features and classes.

1.3.2 Different Feature Extraction algorithms used for Images Classification.

In recent years, experts have studied a variety of conventional machine learning techniques. Every single technique has its own advantages and disadvantages. We will define Machine Learning and its importance, as well as its subfields, and then we will review its application in steel inspection as mentioned in the literature.

1.3.2.1 Texture feature-based classifiers

Texture features regularly comprise three components: locally repeated sequences, nonrandom permutations, and texture regions that are roughly uniform. The identification of repetitive local patterns and the establishment of regulations controlling their arrangement within the image are the roles of the texture feature. The following is a concise synopsis of several frequently employed methodologies.

1.3.2.2 Grey Level Covariance Matrix GLCM and Discrete Shearlet Transform

GLCM is a statistical method used to analyze texture by considering the spatial relationships of pixels. It is used in image processing and computer vision for analyzing textures, allowing researchers to study how often different combinations of pixel brightness values appear in an image. An image matrix, also referred to as a gray-level matrix or GLCM, is generated by analyzing the distribution of pixel values that occur together at a specific position within the image. It is commonly used as a method to analyze textures in various applications, such as X-Ray images.

Ashour et al. Proposed in [8] two frameworks to enhance the steel surface inspection performance using GLCM. The first framework serves to produce the most

discriminating surface texture feature representation through the DST-GLCM. Here, both GLCM and DST were used to detect strip steel surface flaws (**Figure I.1**). The Shearlet transform uses affine systems to extract geometric characteristics from multidimensional inputs [8]. Images are used in the GLCM computation to get multi-directional shear properties. These are then put into a support vector machine (SVM) for major component analysis using high-dimensional feature vectors. The pictures are then put into six groups based on the features of their textures. But this method has a lot of matrix dimensions and needs a lot of math, which could make it a potential piece of software.

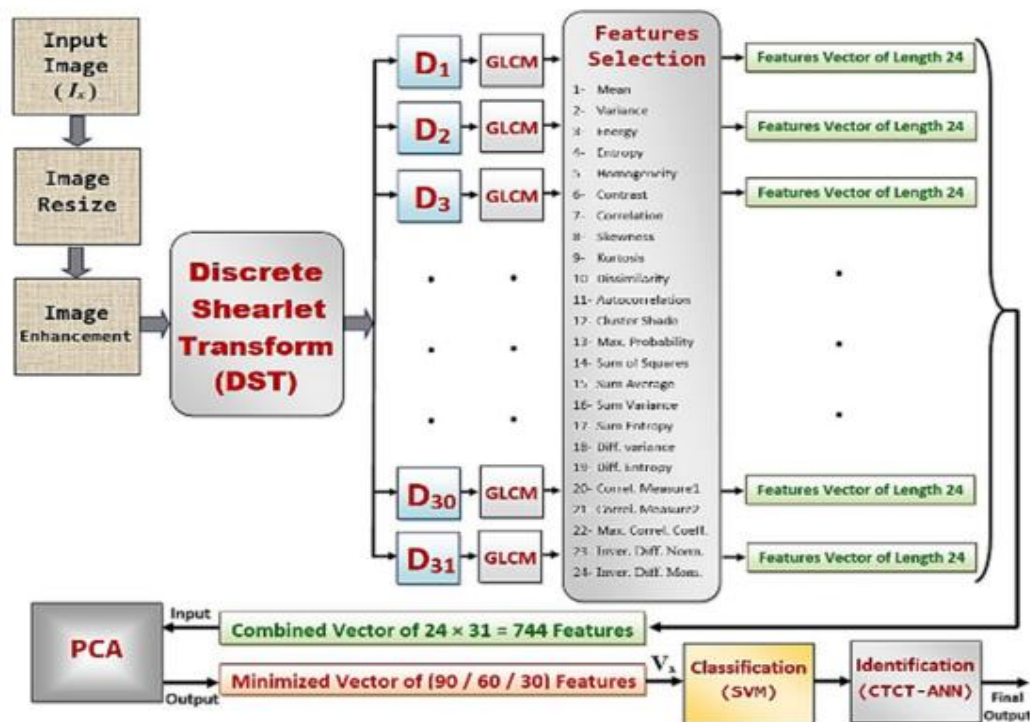


Figure 1-1 Schematic diagram of the method DST-GLCM used in [8]

1.3.2.3 Support Vector Machines SVM

The theory behind SVMs is that the most generalizability may be achieved by determining the highest margin and, thus, the most ideal hyperplane. The best classification effectiveness was found for both training and testing data. The main aim of SVM is to maximize the function given in the equation (1.1) with respect to w and b in order to determine the maximum marginal hyperplanes:

$$L_p = \frac{1}{2} \|\vec{w}\|^2 - \sum_{i=1}^t \alpha_i \gamma_i (\vec{w} \cdot \vec{x} + b) - \sum_{i=1}^t \alpha_i \quad (1.1)$$

Where t is the number of learning points. L_p represents the Lagrangien and α_i is the Lagrangien multipliers. The hyperplane is defined by the vector \vec{w} and the constant b . The data points on the edge of the ideal separation hyperplane are designated as the support vector points. All other data points are eliminated and these support vector points are combined linearly to get the solution. This indicates that the complexity of an SVM is not impacted by the amount of features contained in the training data (**Figure I.2**), making SVMs ideal for learning issues that have a significant number of features in relation to the number of features in comparison of the training examples [9].

When data contains instances that are incorrectly classified, the SVM implementation faces an issue where a separation hyperplane can not be identified. On the other hand, identifying data in a modified feature space can help solve challenges with nonlinearly separable data that frequently arise in real-world scenarios. As a result, normal training sets might be divided. The use of SVM implementation offers benefits like fewer dimensions, hundreds of training examples, and an established theory. Nevertheless, throughout the training and classification stages, it uses a lot of memory and time due to its intricate learning and classification algorithms [10].

In computer learning theory, support vector machines (SVMs) are a way to classify things that are different from each other. They are based on the idea of structural risk reduction. The goal of SVM is to find the best classification function for telling the difference between class units in the training data. Making a hyperplane that maximizes the distance between the datasets can help find the best classification function for a dataset that is linearly separable. This will give you the largest possible distance between the datasets [11].

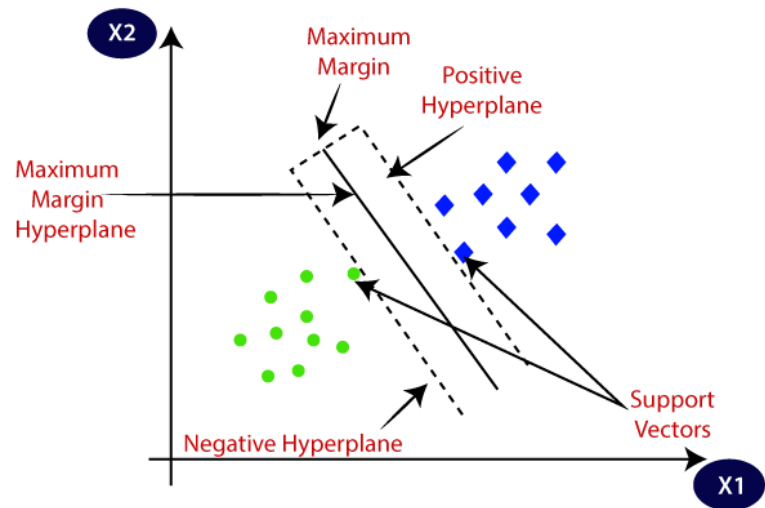


Figure 1-2 SVM max separation hyperplane with margin.

SVM is a complex linear classifier commonly utilized to identify flaws in flat steel surfaces[12]. In a recent study, Zhang and his colleagues found seven different kinds of flaws on the surface of steel by making the classic SVM method's kernel function selection and parameter settings better. Drawing inspiration from reference [13].

Agarwal et al. [14] showed a way to group things using a processing-knowledge-based multiclass SVM called PK-MSVM. This method combined the tasks of automatically finding defects and learning about the process. The PK-MSVM demonstrated superior performance compared to the traditional multiclass SVM.

The researchers in [15] introduced a distinctive multi-hyper-sphere SVM along with an extra information (MHSVM+) method that employs an additive learning model to uncover concealed insights in flawed data sets. When working with flawed datasets, especially those that are corrupted, there is an increase in classification accuracy. Nevertheless, the SVM method struggles with large datasets with noise and overlapping target classes, especially when feature samples outnumber training data samples.

The creators of [16] developed a one-class classification method using generative adversarial networks (GAN) [17] and SVM. It utilizes an SVM classifier with features generated by GAN. It also enhances the loss function, leading to

improved model stability. GANs are highly regarded for their ability to generate top-notch data. Nevertheless, training them can be challenging due to the ongoing competition between generators and discriminator networks, the requirement for extensive training data, and the risk of network collapse resulting in limited output variety.

1.3.3 Distance Function Classifiers

Distance is a valuable metric for explaining the connections among pixels. Understanding distance is a fundamental method for classifying faults. The nearest neighbor classifier (NNC) is a commonly employed distance classification algorithm for identifying surface defects, with the chi-square distance serving as a fundamental yet efficient measure of similarity between training and test images.

1.3.3.1 K-Nearest-Neighbor KNN

KNN, also known as the k-nearest neighbor, is a machine learning technique that identifies a set of k objects in a learning instance that are closest to the test object and labels them based on the prevalence of a class in that proximity. For this approach, three essential elements are required: a set of identified objects; a proximity measure ; and the number k of closest neighbors. One common proximity metric for kNN classification is the "Euclidean distance" which can be understood using the equation that follows:

$$D(x, y) = (\sum_{i=1}^m |x_i - y_i|^2)^{1/2} \quad (1.2)$$

The distance between dataset instances may be defined using several metrics. Minkowsky, Canberra, and Chebychev measures are a few examples, while weighting schemes that alter the vote's influence are frequently employed to get more accurate results [18].

Following the acquisition of the list of k nearest neighbors, the test objects are categorized using the majority class, as shown by equation (1.3):

$$MajorityVoting = Y' = \underset{(x_i, y_i) \in D_Z}{argmax} I(v = y_i) \quad (1.3)$$

Where $I(\cdot)$ is the flag function that returns zero for an invalid argument and one for a valid argument, and v and y_i are the class labels of the i -th nearest neighbor and the class labels, respectively [19]. Understanding and using the k -nearest neighbor algorithms is simple (**figure I.3**). Even with its simplicity, the method performs admirably in many situations, particularly with multimodal classes.

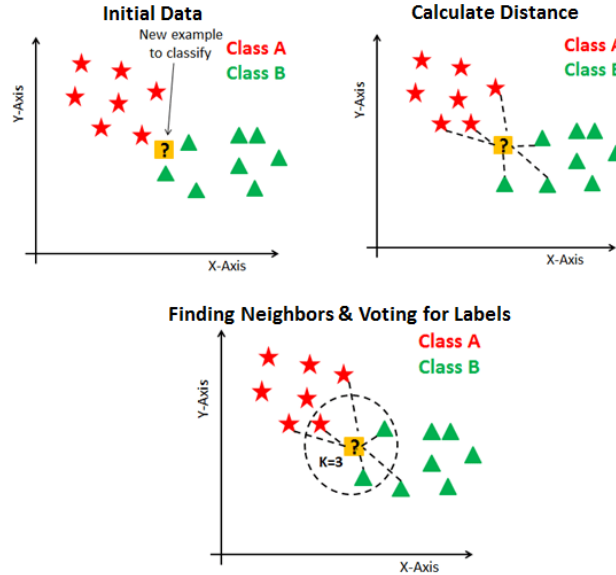


Figure 1-3 : K-nearest neighbor classifier KNN

Nonetheless, fundamental issues affect how the algorithm behaves. Three primary causes are described by Kotsiantis et al. [9]:

1. Extensive storage needs;
2. Highly sensitive similarity function selection;
3. Absence of a crucial k -selection technique.

Dupont et al. [20], proposed utilizing the K Nearest Neighbors (KNN) classifier for fault classification in cold-rolled steel by employing a co-occurrence matrix. In their study, Luo et al. [21] utilized the NNC as a dissimilarity metric between two multi-region histograms to identify LBP-like histograms. In addition, Zhao and colleagues [22] examined the variation and sample distances of the NNC to evaluate the similarity of local models. The precision of the obtained features largely influences the accuracy of this traditional classifier. Improving classification accuracy requires exploring high-dimensional features, which can impact the speed of classification.

1.3.4 Classification based on Neural Networks NN

Traditional machine learning-based algorithms, as previously indicated, are frequently impacted by defect size and noise. Furthermore, this method's accuracy is insufficient to fulfill the practical criteria of automated defect identification. Some elements must be developed by hand, and the scope of the application is highly limited. As well as the afore-mentioned standard machine-learning techniques, they often require sophisticated feature engineering, which considerably raises costs [15]. An artificial neural network uses huge nodes to automatically identify flaws in test photos by extracting characteristics. This approach accurately identifies surface defects in flat steel [23].

Neural networks with back-propagation NN-BP has been widely utilized to classify such faults. NN-BP is typically employed with a single hidden layer. The number of nodes employed in the input layer (feature space: from four to more than 50 layers), hidden layer (between 10 and 100), and output layer (defect class: 2 to 24) varied significantly. NN-BP is most commonly used for hot- and cold-rolled strip surfaces because to the presence of several fault classes.

In a study by Caleb and Steuer [24], the efficacy of the multilayer perceptron (MLP) along with the group method of data handling (GMDH) was examined in classifying defective and non-defective images in hot strips. In three trials, MLP consistently outperformed GMDH, achieving the best result of 97% accuracy for defect classification. However, the accuracy dropped to 89% and 78% for classifying between five defect classes, and decreased further to 86% and 80% for all six classes. The study proposes combining two networks to simplify solving problems at the class level.

With the use of complex features and classifiers, traditional methods have achieved some success, but these approaches suffer from three major flaws:

- 1) Features developed manually depend on in-depth expertise and intricate design techniques. Feature design and classifier design are carried out separately, potentially leading to suboptimal features for the classifier.
- 2) Designing features and classifiers can vary depending on the task at hand. The study employs deep learning algorithms, particularly convolutional neural networks (CNNs), to classify surface defects on steel sheets.

- 3) By utilizing this method, the model can derive more impactful features from labeled photos of surface defects through supervised learning.

Deep Learning helped tackling these issues as we will see.

1.4 Deep-Learning-based surface inspection algorithms

Deep Learning (DL) appears to have taken computer vision to a new level, providing better solutions to a wide range of complicated image and object identification challenges, including steel surface flaw detection [25]. DL has acquired appeal for its high accuracy when trained with large volumes of data [26]. Furthermore, DL has reduced the requirement for feature engineering, allowing for a concentrate on learning the essential features. Wang et al. (2018) emphasized the possibilities of DL approaches in smart mills [27], demonstrating how DL influenced future industry trends. Lately, several noteworthy studies—including Psuj in [28], Yang et al. [29], Tabernik et al. [30] have been published that emphasize the importance of DL in the steel production sector.

1.4.1 Subfields of Deep Learning

Deep learning methods are commonly used for defect identification in industrial inspection settings because they can extract complex visual features using operations such as convolution and pooling. Convolutional neural networks utilize convolutional procedures to extract semantic information and enhance the generalization abilities of the model. CNNs utilize pooling layers and sparse connections to optimize computing resources and enhance network performance [31]. The four classes of deep-learning: supervised learning, unsupervised learning, reinforcement learning and weakly supervised learning.

1.4.1.1 Supervised learning in DL

1.4.1.1.1 SqueezeNet

Fu et al. [32] suggested a light-weight CNN model using SqueezeNet to perform accurate defect identification with a low amount of defect samples. This model focuses on learning low-level characteristics and includes an MRF module (**Figure I.4.**). It is worth noting that the classification accuracy of the paper approached

100 fps. SqueezeNet, on the other hand, has two drawbacks: low classification accuracy as well as significant processing complexity. Despite the fact that the design parameters are quite modest, it is not suitable to be used on devices with insufficient power.

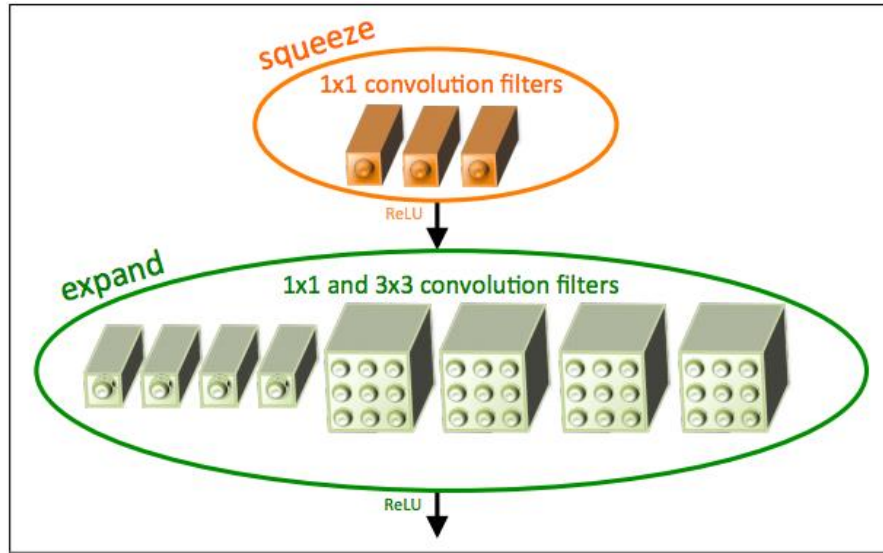


Figure 1-4 Organization of the convolution filters in a Fire module

1.4.1.1.2 ResNet

He et al. [33] presented ResNet, an extension of deep Nets, which transformed the CNN architectural race by integrating residual learning (**Figure I.5.**) and an efficient deep Net training mechanism. ResNet, which is 20 times deeper than AlexNet and 8 times deeper than VGG, demonstrated lower complexity of computations than previous Nets. Empirical results indicated that ResNet with 50/101/152 layers performed better on image classification challenges than 34 layers of plain Net. Furthermore, ResNet improved by 28.0% on the COCO classification of images benchmark dataset, highlighting the significance of depth in visual identification tasks.

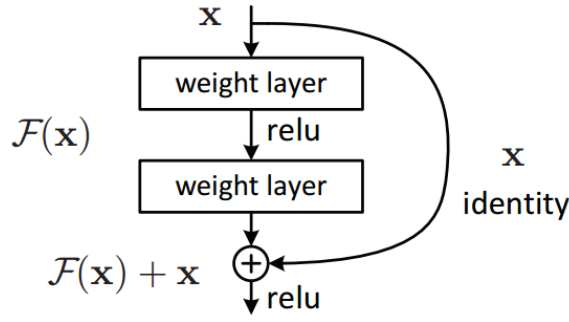


Figure 1-5 Residual block, the basic building block of ResNet.

To overcome a difficulty encountered during deep net training, He et al. [33] introduced ResNet, which makes use of bypass channels. The mathematical formulation of ResNet is represented in equations (1.4 and 1.5) :

$$g(x_i) = f(x_i) + x_i \quad (1.4)$$

$$f(x_i) = g(x_i) - x_i \quad (1.5)$$

Where $f(x_i)$ is a transformed signal, x_i is an original input and is added to $f(x_i)$ through bypass pathways. In basic terms $g(x_i) - x_i$ performs the residual learning

ResNet introduced shortcut connections within layers to enable interconnections across layers, although these connections are data-independent and do not require parameters, unlike Highway Networks. ResNet consistently transmits residual data and does not eliminate identity shortcuts. Residual links, also known as shortcut connections, expedite the convergence of deep networks, allowing ResNet to circumvent gradient vanishing problems.

ResNet's 34-layer simple network design is inspired by VGG-19, and the shortcut connection is added after that. These short-cut connections turn the design into the residual network, as seen in **Figure I.6**.

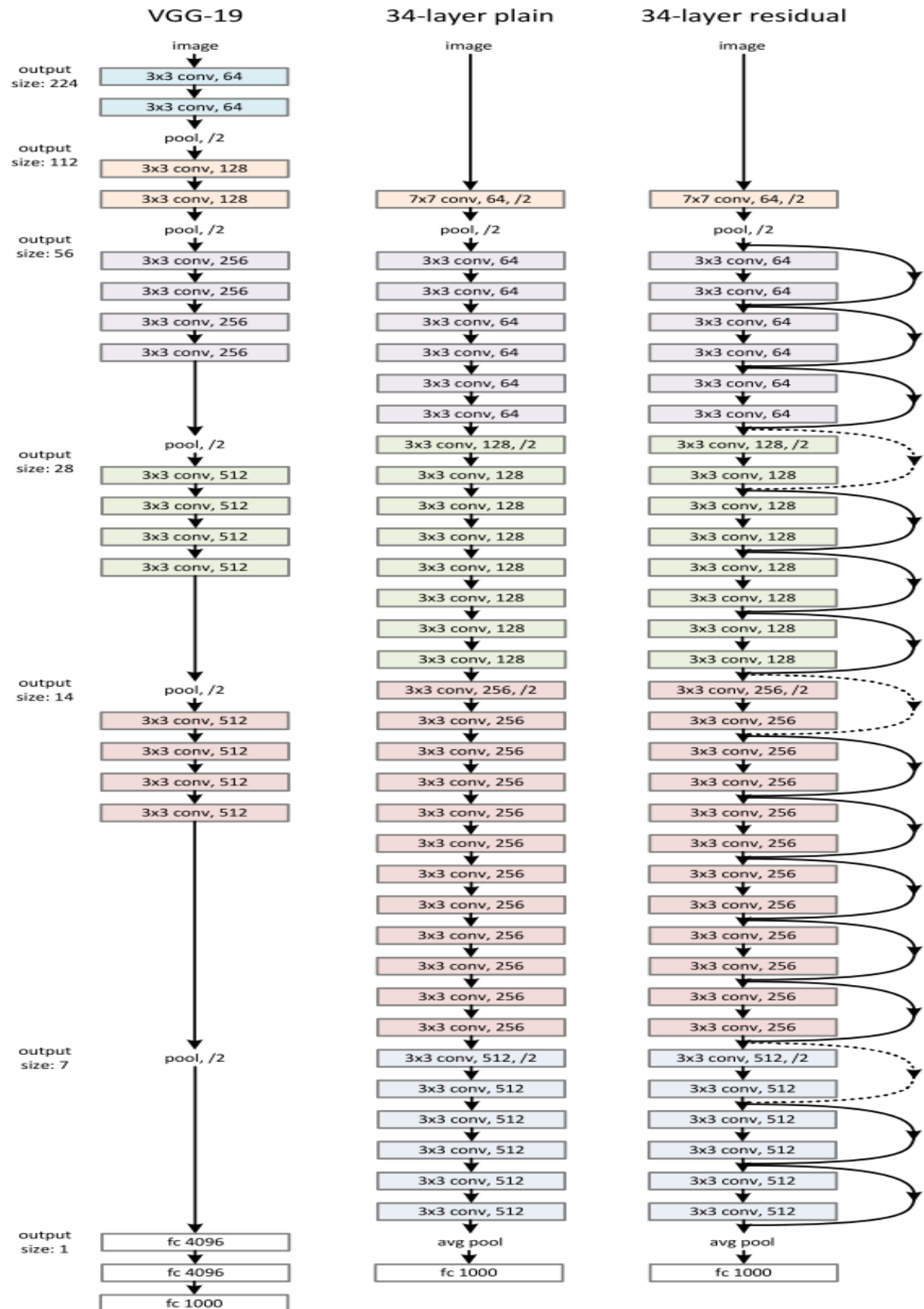


Figure 1-6 VGG vs. 34-layer Plain CNN vs. 34-layer ResNet.

Researchers used binary focus loss to address the issue of data sample imbalance. Through experiments, Ihor et al. [34] and Yousra et al. [35] showed that the pre-trained ResNet50 algorithm was the best choice as a classification network for identifying steel surface defects. ResNet-50, on the other hand, comes with certain limitations. One problem is that the intricate design requires a significant amount of computer resources and memory. In addition, the process of training ResNet-50 can be quite time-consuming and challenging to achieve.

Konovalenko et al. [36] carried out a study on deep residual neural networks ResNets using photographs of flat surfaces of rolled metal. The models demonstrated great accuracy in detecting surface flaws, particularly struggling with identifying class 2 and 3 faults. Identifying Class 1 damage, which is comparable to surface formations, proved to be the most challenging. The ideal depth for the model's structure was 50 layers, while simpler models exhibited worse generalization skills. The top-performing model, using ResNet50, achieved an average accuracy of 0.9691 across all damage categories. It was trained with the binary focus loss function and the SGD optimizer.

1.4.1.1.3 Visual Geometry Group VGG

At Oxford University, Karen and colleagues [37] developed VGG-16, a CNN architecture with 16 layers containing convolutional and fully connected layers (see Figure III.27). VGGNet was trained on more than a billion pictures and now includes over 95 million parameters. The system can process a large number of input images with 4096 convolutional features, leading to high training costs and data requirements. Architectures like GoogLeNet (AlexNet) from CNN show superior performance in classifying tasks when given input images sized between 100 x 100 pixels and 350 x 350 pixels. VGGNet is known for its high computational requirements and serves as a strong base for computer vision tasks such as object detection.

Below is a diagram illustrating the standard VGG16 network architecture (Figure I.7).

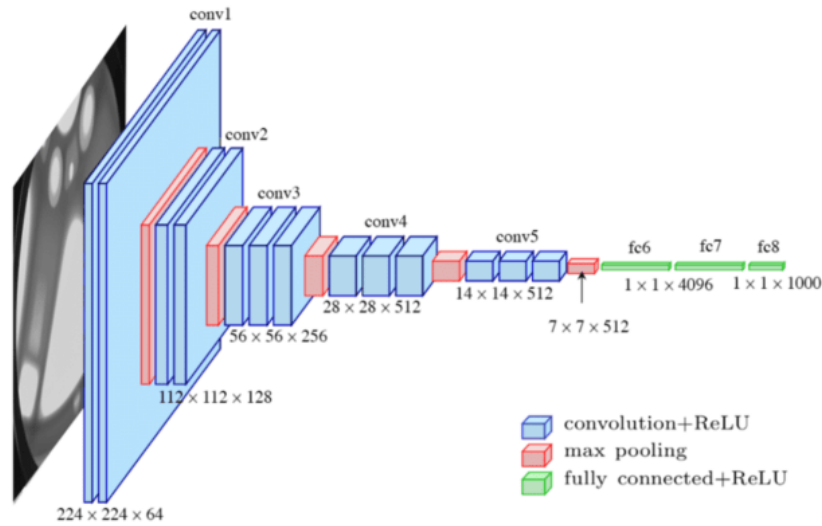


Figure 1-7 VGG16 network architecture diagram [37].

Wang et al. enhanced the VGG-19 network, dividing it into online detection and offline training [38]. The online part aims to pull out region of interest (ROI) regions from defective pictures (Figure I.8.). In contrast, the offline part adds ROI regions to a balanced mixed dataset, improving VGG19 performance and solving the problem of limited or unbalanced data samples. Skip connections and inception modules are used in modern designs to decrease the amount of trainable parameters, boosting both accuracy as well as training time. Nonetheless, VGG has two significant drawbacks: it takes a long time for training and has an enormous model size of 500MB.

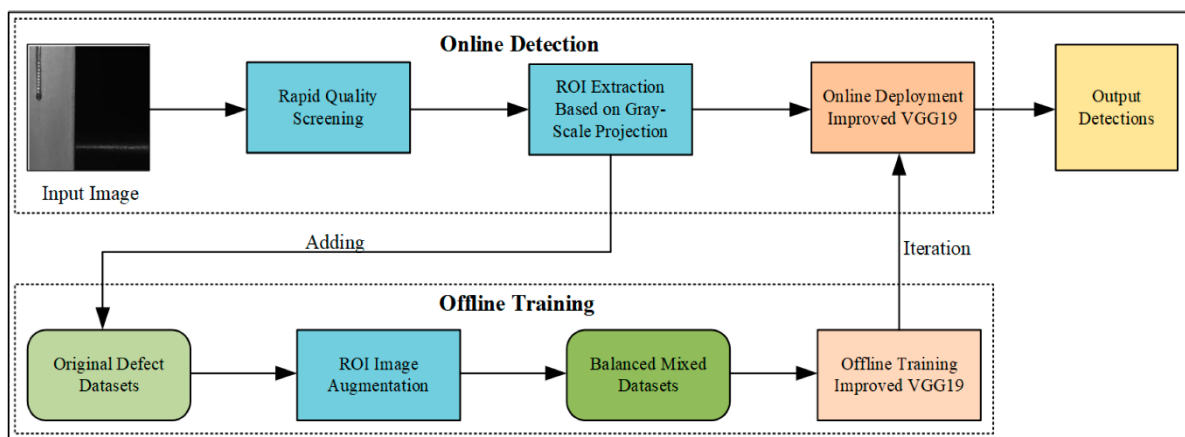


Figure 1-8 Model outline Diagram [38].

Guan et al. Presented in [39] an advanced deep learning network framework that includes feature representation and quality inspection to classify steel surface flaws. The suggested method for classifying steel surface defects showcased better prediction accuracy and speed when compared to algorithms relying on VGG19 and ResNet. To accurately assess image quality and visualize the underlying features of steel flaws, the network framework was fine-tuned to achieve this. Nevertheless, the proposed approach requires top-notch and excellent hardware, resulting in an unavoidable increase in the price of industrial implementation.

Feng et al. [40] devised a plan in 2021 to identify surface flaws in hot-rolled strips by merging the RepVGG framework with a spatial attention mechanism. For a hot-rolled steel product, a novel defect dataset called X-SDD was put out. However, because there were so many characteristics, the recognition efficiency was low.

Developed in[41], a 26-layer CNN was created in order to identify defects existed in roller bearing components, and its efficiency was evaluated against MobileNet, VGG-19[37], and ResNet-50. VGG-19 attained a mAP (mean average precision) of 83.86%. Nevertheless, this model experienced a long processing time of 83.3 ms.

1.4.1.1.4 Xception

Xception is a sophisticated convolutional neural network architecture that makes use of Depthwise Separable Convolutions [42]. Developed by researchers at Google. Google explained Inception modules in convolutional neural networks as a transitional step between standard convolution and the depthwise separable convolution function. A depthwise separable convolution can be viewed as an Inception module with numerous towers (**FigureI.9.**). A new neural network architecture inspired by Inception has been proposed, where Inception modules are replaced with depthwise separable convolutions.

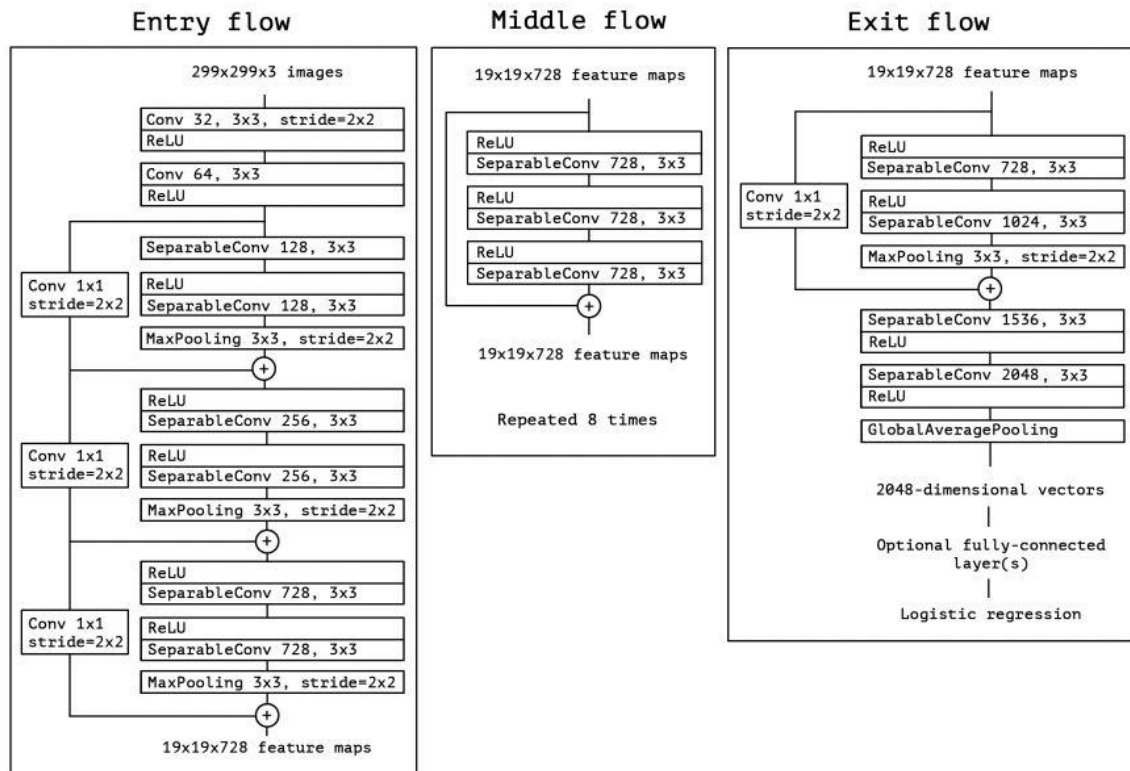


Figure 1-9 Xception overall architecture

1.4.1.1.5 U-Net CNN

The U-Net is a prevalent deep learning framework first presented in the research article entitled "U-Net: Convolutional Networks for Biomedical Image Segmentation." [43] This architecture was primarily made up to tackle the issue of insufficient labeled data in the medical sector. The network was created to efficiently use a limited quantity of data without compromising on speed and precision.

The design of U-Net comprises a contracting route and an expanding path. The contracting route processes contextual information and decreases input spatial resolution, while the expanding path decodes the encoded data and produces a segmentation map. The contracting route decreases input spatial resolution, whereas the expanding path increases feature maps and conducts convolutional processes. Skip connections maintain spatial information, allowing for more precise feature localization (**Figure I.10.**).

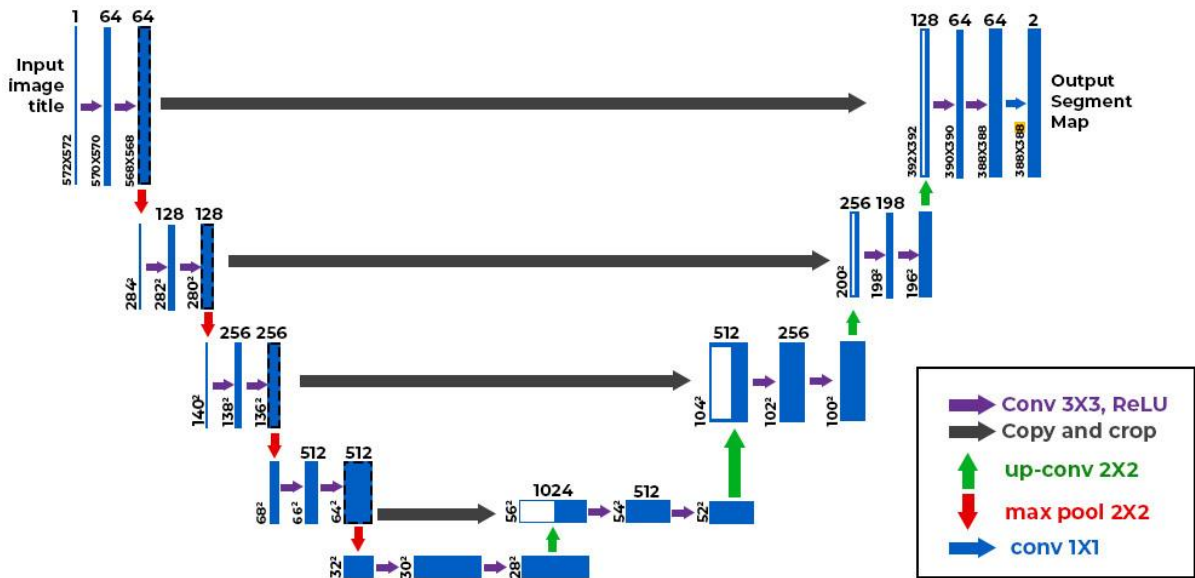


Figure 1-10 U-NET architecture

Researchers in [44] independently trained U-Net and Xception as classification algorithms to classify surface defects on rolled steel. They used synthetic datasets for training and the normal dataset for testing to assess the viability and utility of manually created datasets. There are, however, certain drawback. The intricacy of the standard U-Net architecture is one restriction that might impair training and accuracy. Another restriction is the time-consuming necessity for manual segmentation of steel surface pictures. Furthermore, the poor resolution of defective photos and insufficient data might make enhancing steel inspection accuracy difficult.

A technique for automatically segmenting and quantifying pictures of titanium-coated metal surfaces was suggested in [45] utilizing a customized deep learning framework for the purpose of defect detection. By implementing the necessary preprocessing and postprocessing procedures, the U-Net was trained to segment biological pictures in [16]. To remove impulsive noise, the network utilized a median filter on the input pictures. When tested against industry-standard benchmarks, the created model's detection and segmentation capabilities yielded an accuracy of 93.46 percent.

1.4.1.1.6 MobileNets, a CNN Architecture for Mobile Devices

Since the rise of Convolutional Neural Networks (CNNs), network depth has risen. Multiple layers result in a significant computational expense, which restricts the use of neural networks. In 2017, Google created MobileNet, a compact neural network designed for use on mobile devices. MobileNet's design is shown in Table 3. Depthwise separable convolutions are utilized instead of traditional convolution layers. Depthwise convolution separates the convolution kernel into single-channel forms, distinguishing it from regular convolution. Depthwise separable convolution involves splitting kernels into two separate operations: depthwise convolution and pointwise convolution. When the input feature depth is consistent, each channel's convolution results in output feature maps with the same number of channels as the input feature maps. Pointwise convolution is a 1×1 convolution method that can change the dimensionality of a feature map. Chapter III is an in-depth overview of the MobileNet architecture.

MobileNets are convolutional neural networks that may be deployed on a mobile device to categorize images or recognize things quickly. Andrew G. and colleagues [46] developed Mobile Networks as shown in **Figure I.11**. They are often compact CNN architectures, allowing for real-time execution on embedded devices such as smartphones and drones. The design is highly versatile and has been tested on CNNs with 100–300 layers, consistently surpassing other architectures such as VGGNet. MobileNet's CNN architecture is used in Android phones with Google's Mobile Vision API to automatically identify frequent object labels in pictures.

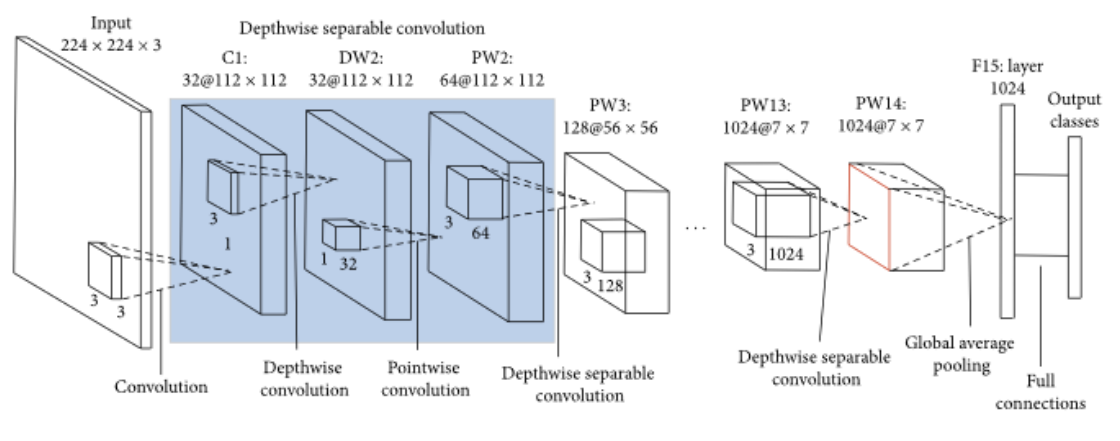


Figure 1-11 MobileNet architecture.

Convolutions used by MobileNet are depth-separable. In contrast to a network with conventional convolutions and the same depth in the network, it greatly reduces the number of factors. This makes deep neural networks that are light. There are two ways to make a depthwise separable convolution: pointwise convolution and depthwise convolution.

➤ Depthwise Separable Convolution

A depthwise separable convolution begins with a channel-Wise $D_K \times D_K$ spatial convolution, where five channels are utilized for 5 $D_K \times D_K$ spatial convolutions, followed by a pointwise convolution, which is a 1×1 convolution to shift the dimension (Figure I.12.).

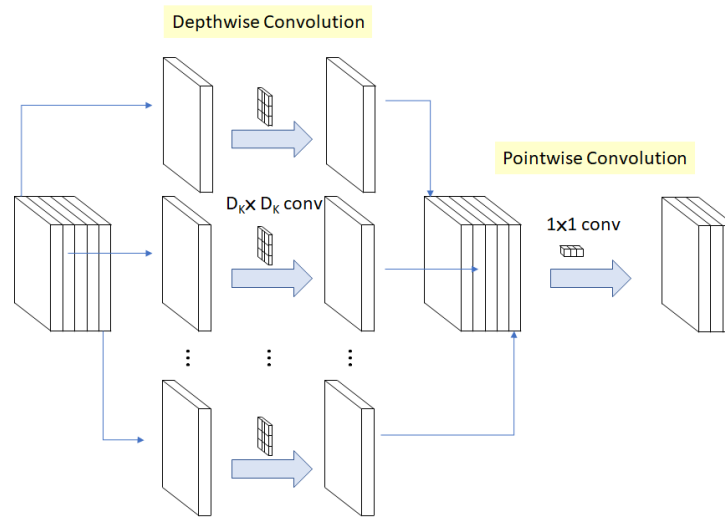


Figure 1-12 Depthwise and pointwise operations.

The VGG-16, ResNet-50, and MobileNet v1 models all have modest architectures that avoid gradient vanishing. Deploying the model using the combination of these three CNNs was applied in [47] by Huang et al, in accordance with the principles of TL technique which could yield initial predictive accuracy after a few attempts. The AutoML model was developed using a core layer module that combines modules from VGG-16, ResNet-50, and MobileNet v1 (Figure I.13.). This architecture enhances defect identification and decreases training costs. The approach offers significant benefits in implementing proof-of-concept for defect identification

by choosing an improved candidate for the CNN. This study's findings can serve as a guide for the advancement of innovative diagnostic technologies in smart manufacturing.

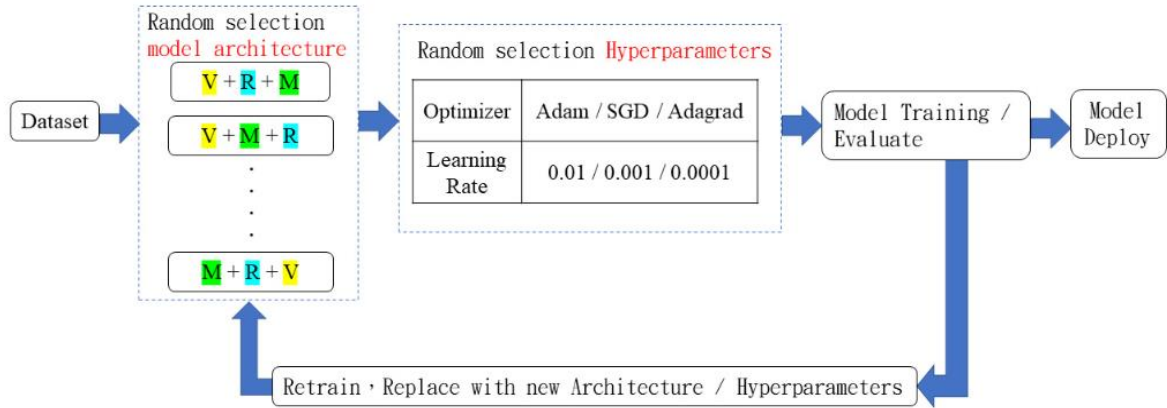


Figure 1-13 Operation of the designed AutoML model used in [47]

Supervised Deep learning approaches rely on accurately labeled sets of training and testing data to verify their performance, often yielding more precise and dependable results compared to other methods. Still, the data quality for training significantly influences the algorithm's performance, and creating a large, well-labeled dataset for real-world industrial defect identification is quite difficult.

1.4.1.2 Unsupervised learning methods

Researchers employ unsupervised approaches to cluster pixels or pictures into related kinds without prior knowledge, enabling classification algorithms to infer correlations between inputs without label pretraining [25]. This is especially beneficial in practical applications flat steel industrial production lines, wherein manual labeling of surface faults takes considerable time and is labor-intensive owing to hostile manufacturing settings. The study of unsupervised classification algorithms improves surface fault identification for flat steel.

Unsupervised learning is a sort of machine learning that utilizes unlabeled data. This signifies that the data has no pre-existing labels or classifications. The purpose of unsupervised learning is to identify patterns and correlations in data without explicit instruction.

Unsupervised learning involves training a computer with unclassified information, allowing the algorithm to work on similar data without specific guidance. The machine's task involves organizing unsorted data by identifying commonalities, trends, and distinctions without the need for previous training data. Unlike supervised learning, there is no teacher to provide guidance, so the machine does not receive any instructions. Consequently, the computer's capacity to identify concealed patterns in untagged information is restricted.

1.4.1.2.1 Types of Unsupervised Learning

Unsupervised learning is divided into two types of algorithms:

- **Clustering**

Clustering in unsupervised machine learning is a strategy that categorizes unlabeled data based on similarities, detecting patterns and correlations without prior understanding. It organizes raw, unclassified data into groups, such as customers based on input parameters. This approach is used to classify customers without specifying output parameter values, as seen in the example image.

Some typical clustering methods :

- 1) K-means Clustering: dividing data into K clusters.
- 2) Hierarchical Clustering creates a hierarchical structure of clusters, while density-based clustering (DBSCAN) identifies clusters based on density.
- 3) Mean-Shift Clustering: Finding Clusters Using Mode Seeking
- 4) Spectral Clustering: Using Spectral Graph Theory for Clustering

- **Association**

When dealing with association rule learning problems, the goal is to identify patterns that reveal significant insights in the data, like determining if customers who buy product X also purchase product Y.

- **Dimensionality Reduction**

Reducing the number of features in a dataset while preserving as much information as possible is known as dimensionality reduction. This approach can enhance the efficiency of machine learning algorithms and data visualization. Principal Component Analysis (PCA) [48] is one of dimensionality reduction techniques. It is a linear transformation that reduces dimensions.

1.4.1.2.2 Different algorithms of unsupervised learning

In this section we will see the most used unsupervised algorithms as cited in the literature.

A. Generative Adversarial Networks GAN

A Generative Adversarial Network (GAN) represents an unsupervised deep learning framework [17]. The system trains two different neural networks to compete in generating more realistic new data from a provided training data set. As an illustration, you can create new pictures from a current image collection or unique music from a song collection. A GAN is referred to as adversarial due to the involvement of two distinct networks that are positioned against each other.

A network creates fresh data by selecting a portion of the input data and making extensive modifications to it. The other network attempts to determine if the generated data is part of the original data set (refer to the **Figure I.14**). Put simply, the prediction network decides if the generated data is fake or genuine. The system creates updated false data values until the prediction network cannot differentiate between the false and original data.

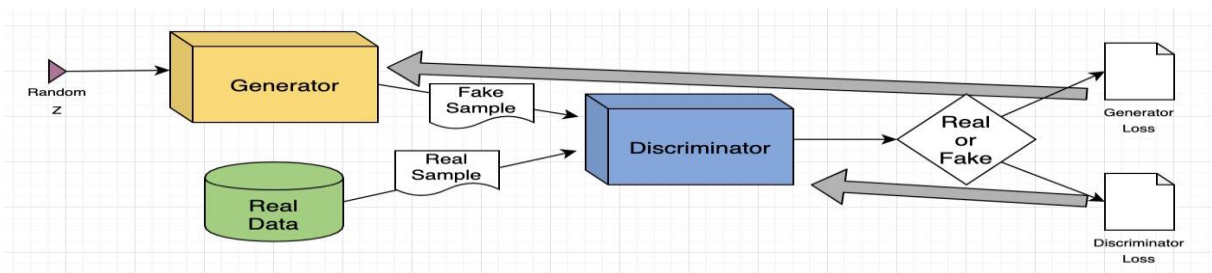


Figure 1-14 GAN training example [17].

B. Convolutional Autoencoders CAE

Autoencoders are unique neural networks with an equal number of neurons on both the input and output layers. The objective of an auto-encoder is to ensure that the exit is as close as possible to the entrance. Learning is considered "self-supervised" as it involves minimizing the cost of the reconstruction between exit and entry. Since the data does not require labeling, it functions as its own label, classifying this model as unsupervised.

➤ Autoencoder architecture

An autoencoder has a very specific architecture because the hidden layers are smaller than the input layers. This type of architecture is called a bottleneck architecture. One can break down an auto encoder into two parts on the left and right of this bottleneck. The left-hand part is called the encoder. The encoder turns the input into a representation in a smaller-dimensional space called a latent space. The encoder thus compresses the input into a less expensive representation. The right part is called the decoder because it must reconstruct, using the latent representation of the input, the most faithful output to the input (**Figure I.15.**)

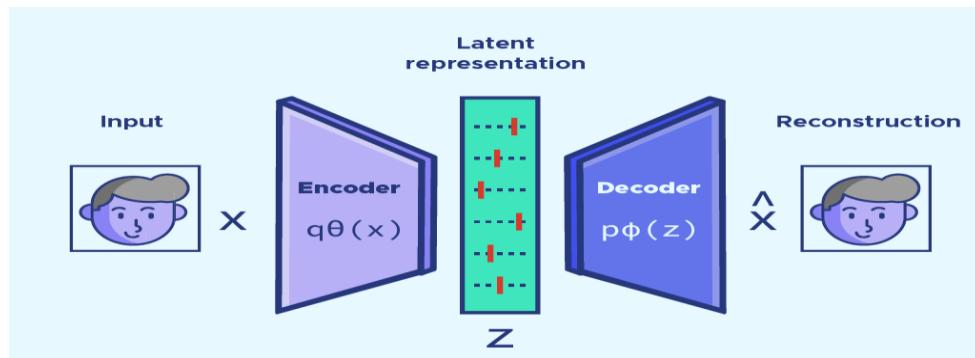


Figure 1-15 Autoencoder framework

In the literature, LIU et al. Proposed a single classification technique for steel strip surface defect identification that was based on GAN [16]. However, this technique was able to recognize the presence or absence of defects but was unable to differentiate between the different categories. Using convolutional autoencoder (CAE)

and sharpening processing, the authors of [49] were able to extract the defect features from the missing input image. After that, they employed Gaussian blurring and thresholding as post-processing techniques to clarify the defects in order to achieve defect segmentation. Niu et al. [50] suggested a global low-rank non-negative reconstruction method with background limitations. This method is used to combine the detection results of 2D important maps with 3D contour information in order to find problems with rail surfaces.

C. Self-Organizing Map (SOM)

A self-organizing map (SOM) is meant to automatically turn the training samples into a condensed representation of the input space when an unsupervised learning method is used. After being received, the SOM categorizes the pertinent areas based on the external stimulus. Various response traits are generated automatically. Set apart from Similar to the CNN, the SOM establishes connections between neurons within the same layer to promote neuronal responses and suppress inactive neurons through scale adjustments. SOMs are excelling in this field. Well-known for evaluating clusters, identifying composite damaged materials, and classifying surface defects on steel.

In a study, the self-organizing map (SOM) was improved by the neural network (NN) using error back-propagation (NN-BP) [51] and applied to classify defects on cold-rolled strips. In their study, Martins et al. [52] utilized a blend of PCA and SOM to classify three types of surface defects found in rolled steel with intricate textures. The application scenarios show the potential of SOM and its modifications for classifying surface defects in the steel production sector. However, the input picture's quality and the original classification model parameters play a crucial role in determining the unsupervised learning model's classification performance.

1.4.1.2.3 Advantages and disadvantages of unsupervised learning

Unsupervised learning offers many advantages, including reducing dimensionality, detecting previously unknown patterns, and getting insights from unlabeled data. However, because there are no specified responses during training,

measuring accuracy or efficacy might be difficult, leading in lower accuracy. Users must interpret and label classes, and unsupervised learning is subject to data quality, such as missing values, outliers, and noise. Furthermore, assessing the effectiveness of unsupervised learning approaches without labeled data can be difficult.

1.4.1.3 Reinforcement Machine Learning

This learning approach involves interacting with the environment through actions and fault detection. Key components of reinforcement learning include experimentation, faults, and time lapses. Using this approach, the model leverages reward feedback to understand behaviors or patterns and enhance its performance over time. These algorithms are designed for specific challenges, like Google's self-driving vehicle or AlphaGo, where a bot competes against humans and itself to enhance its performance in the Go game. Every time information is inputted, it is absorbed and integrated into their knowledge base, a process referred to as training data. The more knowledge it acquires, the more skilled and seasoned it becomes.

1.4.1.3.1 Types of Reinforcement Machine learning

There are two primary forms of reinforcement learning:

- **Positive Reinforcement:** Reward the agent for doing the required activity. And encourages the agent to continue the behavior.
- **Negative Reinforcement:** Removes an unpleasant stimulus in order to promote a desired response and discourages the agent from continuing the activity.

➤ Reinforcement Machine Learning Advantages and Disadvantages

Advantages of Reinforcement ML include autonomous decision-making for certain tasks, preferable for long-term, tough outcomes and used to solve complicated issues that are beyond the scope of conventional procedures.

Whereas, main Disadvantages are: Training is computationally expensive and takes long time for execution, it is not suitable for tackling basic issues and it requires a massive amount of data and many calculation, the reason that makes it impractical and costly.

1.4.1.3.2 Semi-supervised (or Weakly supervised) learning methods

Semi-supervised learning (sometimes refers to as weakly supervised) is the way of exploring a type of machine learning that falls between supervised and unsupervised learning. When training a model, a small amount of labeled data is mixed with a significant quantity of unlabeled data. Similar to supervised learning, semi-supervised learning focuses on training a function to predict the output variable using the input variables. Contrary to supervised learning, the system undergoes training using a dataset that contains both labeled and unlabeled data.

Utilizing semi-supervised learning can be highly beneficial in scenarios where a large amount of data remains unlabeled due to cost or complexity.

➤ Application of Semi-supervised learning algorithms

Semi-Supervised Learning is utilized in many studied areas. It is ideal for labeling audio files, ranking webpage relevance in Google search algorithms, and analyzing large DNA strands, making it a natural approach for many tasks related to speech analysis, categorizing internet content, and classifying protein sequences.

➤ Drawbacks of Semi-supervised learning algorithms

The most basic drawback of any Supervised Learning technique is that the dataset must be manually labeled by either a Machine Learning Engineer or a Data Scientist. This is an extremely costly operation, especially when working with massive amounts of data. The most fundamental drawback of Unsupervised Learning is that its practical scope is constrained. To address these drawbacks, the notion of semi-supervised learning was created. This method of learning entails training an algorithm using a mixture of labeled as well as unlabeled data, often with a small number of labeled data and a significant number of unlabeled data. The approach employs an unsupervised learning algorithm to cluster comparable data before labeling the unlabeled data with the already labeled data. This approach is often used for low-cost unlabelled data gathering.

In general, semi-supervised machine learning offers more generalization than supervised learning by accepting both labeled and unlabeled inputs and can handle a

wide range of data. However, it can be more challenging to implement, requires labeled data, which may not always be readily available, and can affect the model's performance if unlabeled data is not included.

In 2019, a technique was proposed by He et al. [5] that combined a convolutional autoencoder with a semi-supervised GAN. This approach also incorporated a passthrough layer inside the CAE to extract detailed features, leading to high recognition accuracy.

In 2019, the researchers at the Google Institute created a new algorithm called MixMatch [53] by merging existing semi-supervised algorithms. They carried out thorough detection tests on this algorithm, showing significant performance improvements compared to earlier weakly supervised methods.

He et al. [54] presented as well a multiple learning technique based on ResNet-18 and GAN that was able to produce new data samples with their labels, allowing the dataset to be expanded and thereby improving fault identification with few samples. In brief, weakly supervised learning is a sort of machine learning that does not need a large amount of labeled data. The system can learn from a tiny amount of data while still making accurate estimates. It does, however, have certain negative aspects. It might not be equally accurate as supervised learning, and training the model may take longer. Another key disadvantage of this kind of machine learning technique is that its algorithms are incapable of generalization. The technology can only make precise predictions for particular cases, not for novel ones.

Finally, to aid comprehension of the three deep learning techniques' attributes, **figure I.16** provides an overview of a pipeline diagram showcasing the various deep learning techniques, while **Table I.1** outlines the different aspects of these techniques.



Figure 1-16 Diagram of the outline of different DL techniques. [31].

Table 1-1 Summary of the advantages and disadvantages of different DL techniques.

Deep learning method	Advantages	Disadvantages
Supervised methods	High precision, versatility, and a diverse range of applications.	Annotating the dataset is time-consuming and challenging.
Unsupervised methods	It may be trained directly with label-free data and basic methods.	Noise and beginning settings have a significant impact on the training results, which are rather low accuracy and unstable.
Semi-supervised methods	It combines the benefits of the two supervised and unsupervised approaches.	The training procedure is tedious, and the execution is complex.

1.5 Conclusion

Manual identification methods to recognize steel surface flaws are restricted due to low efficiency, high cost of labor, and unsatisfying accuracy. Because the human eye's observation area and speed are constrained, manual approaches are unsuited for industrial production situations. Traditional machine learning techniques, which depend on expert knowledge, are weak in terms of resilience and generalization. They are sensitive to flaw size and noise and make assumptions. Whereas, deep learning techniques have been prominent in surface defect identification due to advances in processing power, autonomously extracting visual information using neural network models, and overcoming the disadvantages of older methods.

In this chapter, we examined a variety of fascinating research projects to examine cutting-edge vision-based surface defect assessment methods in steel sheets. We additionally addressed the general topics of vision-based surface defect inspection for steel, including manual inspection and AVI methods, where we described traditional machine learning-based and deep learning-based methods and summed up their advantages and drawbacks at the end of the chapter.

2 CHAPTER II

ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, AND DEEP LEARNING OVERVIEW

2.1 Introduction

In this chapter, we aim to give a good understanding of artificial intelligence including its subfield. More precisely, machine learning and deep learning-based algorithms that are currently employed in many tasks related to image recognition and computer vision-based classification.

2.2 Artificial Intelligence Definition

When we try to give a clear definition of what artificial intelligence is, we need to explain it according to many terms. According to Haugeland's notes in his famous book: "Artificial Intelligence, the very idea", artificial intelligence is founded on a really excellent idea, which may or may not be correct. The idea asserts that human thinking and machine computing reasoning are "radically identical" [55]. This concept serves as the core premise of Haugeland's insightful and intriguing book about this exciting new field, which is based on a really excellent idea and serves as the core focus of his enlightening and fascinating book about this exciting new discipline.

Another definition claims that artificial intelligence is the skill of constructing machines that execute activities that, when performed by humans, require intelligence. It is the "study of the computations that make it possible to perceive, reason and act." as stated by Winston [56].

➤ What is Alen Turing Test?

One of the most widely used intelligence tests within artificial intelligence is the Turing test. Alan Turing developed the Turing test in 1950. It is an examination to discover whether or not a machine is capable of thinking like a human. According to this test, a machine is only intelligent as long as it can imitate human behaviors in specific situations. A Turing test requires two people and one computer. One person will act as an interrogator (i.e., the questioner), questioning one human and one computer. Three of the participants will be in their own rooms. The interrogator only recognizes them as A and B, so it must determine which is human and which is machine. The machine's purpose is to convince the interrogator that what it shows is the person's answer. If the computer is successful in tricking the interrogator, it will behave like a human. It is then artificially intelligent. It is extremely difficult to program a computer so it can pass the Turing test.

2.3 Machine Learning Overview

Machine learning is a discipline of computer science that employs statistical methods to estimate complicated functions from data. (See chapter I)

➤ Training a machine learning algorithm

Selecting the network type and activation functions is the first step in designing a neural network. The next step is to figure out the model's other adjustable parameters, such as the weights that link inputs to hidden neurons and hidden neurons to output neurons. The process of altering these settings such that the network is able. What we call "learning" is actually an approach to the functional connection between inputs and targets.

To train a neural network, one must first determine how to adjust its parameters so that, given a series of training instances, the network produces outputs that are very close to the target output. This might be the code of the class in which the shape is to be categorized, the intended result of the procedure to be commanded, the value of the function to be approached, or the output of the process to be modeled. It could also be anything else. In order to minimize the discrepancy between the expected and

actual network responses, neural network learning algorithms make successive parameter adjustments (called "iterations"). The neuron network's output becomes more data-aware as learning progresses.

In order to train a machine learning algorithm, two functions, $\hat{y}(x)$ and $y(x)$, can be approximated. The method looks for the closest distance, in a given metric, between $\hat{y}(x)$ and $y(x)$. One may use a linear regression to demonstrate the fundamentals of training as in equation (2.1):

$$\hat{y} = w^T x \quad (2.1)$$

Thus, w^T is a vector of parameters—weights in the background of machine learning that the algorithm is able to optimize. They ascertain the correlation between the attributes x_i and the output \hat{y} ; predicting y from x is the process of determining the closest "distance" between y and \hat{y} . The optimization of parameters by an algorithm can take many different forms. One potential learning technique in the example given is to reduce the mean squared error (MSE) on the training set x in equation (2.2):

$$MSE = \frac{1}{n} \sum_i (\hat{y} - y)_i^2 \quad (2.2)$$

Here, n represents the total number of events x with the features i . And \hat{y} is the model's prediction of the training set. The gradient is solved for 0 with regard to weights w in order to minimize the MSE (equation 2.3):

$$\nabla_w MSE = 0 \quad (2.3)$$

The MSE is also computed for an independent validation set x_{val} with n_{val} events, here, the algorithm does not use it for training, in order to verify the training process (equation 2.4):

$$MSE_{val} = \frac{1}{n_{val} \sum_i (\widehat{y_{val}} - y_{val})_i^2} \quad (2.4)$$

The validation set output values is denoted by y_{val} , and the algorithm's prediction for y_{val} based on x_{val} is represented by $\widehat{y_{val}}$. The method keeps iterating the

training and validation steps until the error meets a task-specific threshold of sufficiently minimal error.

Generally speaking, the learner has to develop a learning strategy and a model that, like the previous linear regression, represents the output y in terms of input x . The next sections include descriptions of models used in deep learning.

2.4 The history of Artificial Neural Networks ANNs

In 1890, American psychologist W. James presented the idea of associative memory and put forward a foundational principle for learning on neural networks [57], which was later recognized as the Hebb law. In 1943, J. McCulloch and W. Pitts [58] presented an illustration of the biological neuron (the neurons with binary behavior). They were leaders in demonstrating that networks of basic formal neurons have the capability to carry out intricate logical, arithmetic, and symbolic operations, at least in theory. During that time, American physiologist D. Hebb discussed how animals' conditioning could be attributed to the properties of neurons. As an illustration, when a dog is fed at a consistent time each day, it will start salivating at that time even if there is no food present. The theory about altering the features of relationships between neurons he suggests helps to clarify this particular experimental outcome.

In 1958, F. Rosenblatt [59] created the model of the Perceptron. It's a network of neurons inspired by the visual system. It has two layers of neurons: a perception layer and a decision-making layer. It is the first artificial system capable of experiential learning. Later, B. Widrow [60] introduced in 1960 the linear perceptron called ADALINE (adaptive linear element), which can give any output value other than zero and a . This model will then be the basic model of the gradient retropropagation algorithm widely used today with multi-layer perceptrons. A few years later, Mr. Minsky and S. Papert [61] published a book (Perceptrons), which highlights the theoretical limitations of Perceptrons, because they are unable to solve nonlinear problems such as the case of the XOR (the Exclusive OR), this logical function that is not linearly separable, which caused an abandonment of research until 1982, when T. Kohonen [62] presented his work on associative memories and proposed applications

to the recognition of forms. It was in the same year that J. Hopfield [63] presented the study of a completely rebound network, in which he analyzed its dynamics.

The Boltzmann machine [64] is the first known model to satisfy the limitations identified in the case of the perceptron. But practical use is difficult, with the convergence of the algorithm being extremely long (the calculation times are considerable). In the same period, Gradient retropropagation appeared in 1985. It is a learning algorithm adapted to multi-layer neuron networks (also called multilayer perceptrons, or MLP). His discovery was made by three groups of independent researchers. From this discovery, we have the possibility to realize a nonlinear input/output function on a network by breaking this function down into a sequence of linearly separable steps. Nowadays, multi-layer networks and gradient retropropagation remain the most studied and productive models at the application level.

➤ **Application areas of Artificial Neural Networks**

Artificial neural networks are used in a multitude of fields nowadays, including image processing, signal processing, control, defense, optimization, simulation...etc.

2.4.1 Biological neural network

A neuron is a kind of nerve cell that transmits an electrical signal under particular conditions. It serves as a relay between one layer of neurons and the one below it. A neuron's body is connected to a collection of dendrites (neuron entrances) on the one hand and an axon, the stretched component of the cell that would symbolize its exit, on the other; an illustration is illustrated in **figure II.1**. The neuron under study is linked to the neurons around it ; it receives electric signals from "upward" neurons, which are propagated by the latter's axons, at the level of its dendrites [65]. The electrical charges accumulate in the neuron until they exceed a certain threshold; at this point, the transmission of the electrical signal is triggered via its axon to other "downward" neurons.

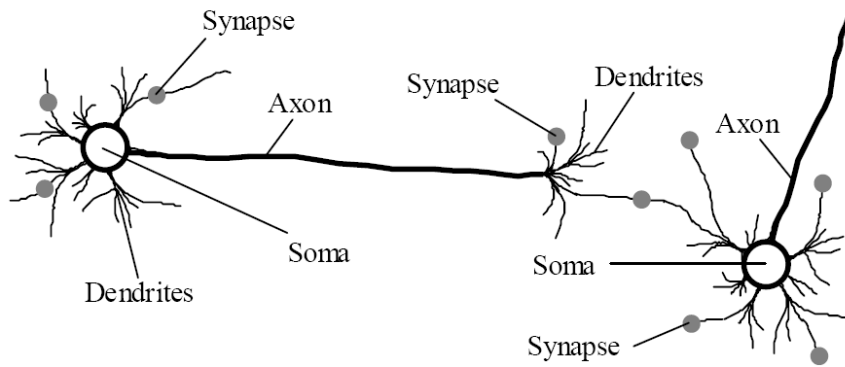


Figure 2-1 Biological neuron morphology [65].

It is observed that not all synaptic connections – also known as axone/dendritic bonds – between two neurons function similarly. An input associated with a specific dendritis may have a greater effect on the output value than another. A sort of coefficient that is added to the input signal and may be used to indicate the bond's quality is called a weight. The weight will rise in direct proportion to the bond's quality. An entrance will be accentuated by a positive weight and inhibited by a negative weight.

As a result, researchers and specialists tried to reproduce the biological neurons and create the formal neurons. Thus, it is a mathematical simulation that takes on the fundamental principles of the functioning of the biological neuron.

2.4.2 Definition of artificial neural network ANN

In 1943, McCulloch and Pitts [58] developed an artificial neural network system that mimics biological neurons with a sophisticated framework (**Table II.1 and Figure II.2**). The ANN system is seen as a network of elements with similar structures, known as neurons, interconnected in a way that resembles the cells of the human nervous system. The structure consists of a series of interconnected layers where each neuron retains input from the previous neuron's output. Each neuron operates autonomously from the others to create a cohesive system. Data is stored and shared throughout the network as synaptic coefficients. The formal neuron consistently computes a value that it then sends to the next neuron; each computation is linked to a weight that determines the intensity of the connection (**Figure II.2**).

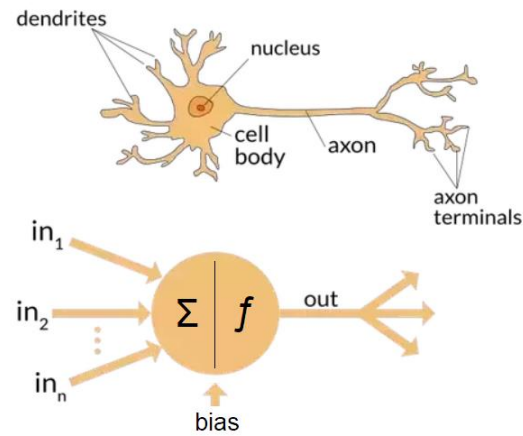


Figure 2-2 An artificial neuron and a biological neuron

Table 2-1 Comparison of the formal neuron with the biological neuron

Biological neuron	Formal neuron
Synapses	connection weight
Axons	Output signals
Dendrites	Input signals
Soma	Activation function

An "artificial neuron," also referred to as a formal neuron, is a nonlinear, bordered mathematical function whose value is determined by parameters known as coefficients or weights. The neuron's inputs are referred to as its variables, while the result of the function is known as its output. Therefore, a neuron serves as a mathematical instrument that can be computed with just a few lines of software. As illustrated in **Figure II.3**, the concept of visually depicting a neuron was adopted.

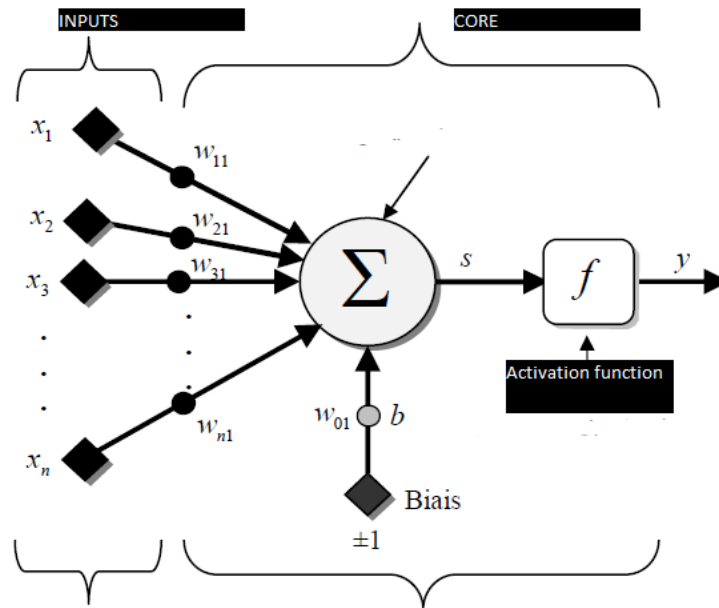


Figure 2-3 Mathematical modeling of a neuron xi

Based on observations of biological neurons, W. M. Culloch and W. Pitts introduced a formal neuron model in 1943: The xi symbolizes the input vectors; they originate from the outputs of additional neurons or from sensory input stimuli like a visual sensor or sound.

-The $w_{i,j}$ are the *synaptic weights* of the j^{th} neuron, they correspond to synaptic effectiveness in biological neurons ($w_{i,j} > 0$: excitatrice synapse ; $w_{i,j} < 0$: inhibiting synapsis). These weights weighed the entries and can be modified by learning.

-*Biais*: Values of -1 or +1 are commonly used for input, providing flexibility to the network by enabling the adjustment of results in thresholds for neurons through weight and bias modifications during learning.

-*Core*: The system combines various inputs and biases to determine the neuron's output based on a nonlinear activation function, which enables more flexible learning.

2.4.2.1 Modeling a formal neuron

Figure II.3 depicts the mathematical model of an artificial neuron. A neuron consists of an integrator that calculates the weighted sum of its inputs. After calculating the sum, it is passed through a transfer function f to generate the neuron's output y . Based on the notations provided earlier, the n inputs of the neuron relate to the vector x , while w symbolizes the weight vector of the neuron. The vectors x and w as well as

the output of the interrogator s are given by the following equations respectively (eq 2.5 to 2.8):

$$x = [(x_1, x_2, \dots x_n)]^T \quad (2.5)$$

$$w = [(w_{11}, w_{21} \dots w_{n1})]^T \quad (2.6)$$

$$s = \sum_{i=1}^n w_{i1} * x_i \pm b \quad (2.7)$$

$$s = w_{11} x_1 + w_{21} x_2 + w_{31} x_3 + \dots + w_{n1} x_n \pm b \quad (2.8)$$

Which can also be written in matrix form (eq 2.9):

$$s = w^T * x \pm b \quad (2.9)$$

This output is the result of a weighted sum of inputs and weights, together with the bias term of the neuron. The outcome of the weighted sum is referred to as the neuron's degree of activity. The bias "b" is also known as the activation threshold of the neuron. If the activation level hits or goes beyond threshold b, the argument of f becomes positive or null. Otherwise, it is negative.

The mathematical model of a biological neuron consists of three primary components: dendrites, the cell body, and the axon. Dendrites are nerve receptors that transmit electrical impulses to the neuron's cell body, functioning as integrators. Excitement in the neuron leads to the generation of an electrical potential that propagates via its axon to stimulate neighboring neurons. The spatial arrangement and caliber of synaptic connections determine the performance of a biological neural network.

The weight of an artificial neuron reveals the strength of a synaptic link. A negative weight stops a post, and a positive weight makes it stand out. It's important to keep in mind that this is only a rough sketch of a real synapse. A very intricate chemical process that depends on a variety of external factors that we still don't fully understand creates the real thing.

It must be understood that our artificial neuron is a pragmatic model that, as we will see later, will enable us to accomplish interesting tasks. The biological likelihood of this model doesn't matter much to us. What matters is the result that this

model will enable us to achieve. Another limiting factor in the model we have given ourselves is its discreet nature. In fact, in order to be able to simulate a network of neurons, we're going to make time discrete in our equations. In other words, we will assume that all neurons are synchronous; that is, at each time t , they will simultaneously calculate their weighted sum and produce an output (eq 2.10):

$$y(t) = f(s(t)) \quad (2.10)$$

In reality, every neuron in organic neural networks functions asynchronously. In order to obtain the neuron's output, let's return to our artificial model that was created using equation (2.5) and add the activation function f (eq 2.11):

$$y = f(s) = f(w^T * x \pm b) \quad (2.11)$$

By replacing w^T with a single-line W matrix, the following general form is obtained (eq 2.12):

$$y = f(W * x \pm b) \quad (2.12)$$

2.4.2.2 Activation function types

In machine learning, a mathematical function known as the activation function (or transfer function) processes data sent to an artificial neuron. This is similar to how brain cells process the electrical data they get. It changes the output value from the weighted sum of a neuron's entries to a different value. This is done by figuring out the neuron's state and adding nonlinearity to the way the nerve works [66]. "Flattering" effects are common with activation functions. The bias b acts as a threshold. If the weighted sum result is greater than this threshold, the transfer function value is positive or null. If it is less than this threshold, it is negative. Finally, if the weighted sum comes out as:

Less than the threshold, the neuron is thought to be inactive;

Around the threshold, the neuron is thought to be in a transitional phase;

Above the threshold, the neuron is thought to be active.

There are various types of transfer functions that may be employed in ANNs. The most commonly utilized activation functions are given below :

2.4.2.2.1 Sigmoid activation function

Over a binary variable, a probability distribution is represented by the sigmoid function (**Figure II.4**). That is described as in equation 2.13:

$$\sigma(z) = \frac{1}{1+\exp(-z)} \quad (2.13)$$

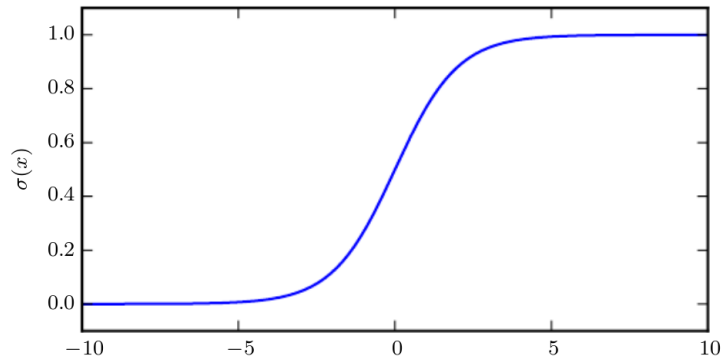


Figure 2-4 Sigmoid function

When z gets extremely negative, the sigmoid activation function saturates to 0, and when z becomes extremely positive, it saturates to 1. Even though these areas are densely populated with training data, the gradient may become excessively small for learning for significant absolute values of z .

2.4.2.2.2 Tanh activation function

Figure II.5 depicts the nonlinearity of \tanh . It shrinks a real-valued number onto the interval $[-1, 1]$. Its activations saturate similarly to the sigmoid neuron, but its output is zero-centered. In fact, \tanh non-linearity is often preferable over sigmoid non-linearity. It's also worth noting that the $\tanh(x)$ neuron is essentially a scaled sigmoid neuron (**Figure II.5**) ; specifically, the following applies (eq 2.14) :

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1 \quad (2.14)$$

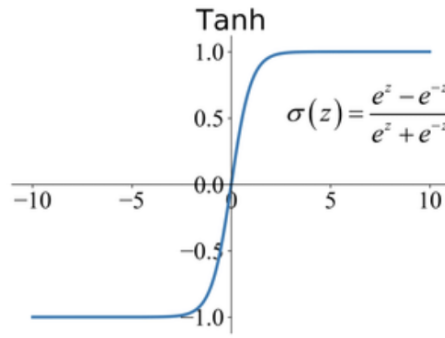


Figure 2-5 Tanh activation function graph

2.4.2.2.3 Softmax activation function

The sigmoid function is expanded to reflect a probability distribution throughout a discrete variable with n possible values, and this is known as the softmax function of z . Softmax functions are commonly utilized as a classifier's output units. The softmax activation function is expressed formally in equation (2.15):

$$\text{softmax}(z) = \frac{\exp(z_i)}{\sum_j \exp(z_i)} \quad (2.15)$$

2.4.2.2.4 Rectified Linear Unit ReLU

In current neural networks, the max function is the activation function that is used by default [67].

$$g(z) = \max(0, z) \quad (2.16)$$

A rectified linear unit (ReLU) is a unit that makes use of this function [67]. Since the derivative is either positive constant value or 0 across the domain, ReLUs may be tuned easily. Due to this, learning may benefit far more from the gradient direction than it might from activation functions with non-vanishing or higher order derivatives. ReLUs have a limitation in that they are unable to learn from cases when the activation is zero using gradient-based techniques (**Figure II.6**).

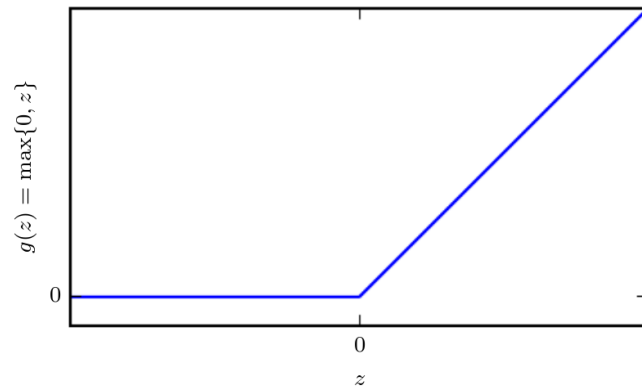


Figure 2-6 The output of a rectified linear unit is 0 across half of its domain [67].

2.4.2.2.5 Exponential Linear Unit ELU

Advanced neural network structures utilize the Exponential Linear Unit (ELU) as an activation function (**Figure II.7**). ELUs have the ability to include negative values [68], unlike ReLUs, which helps in decreasing computational complexity while driving average unit activations closer to zero, similar to batch normalization. The learning process speeds up when the mean moves towards zero due to a reduced bias shift effect, bringing the normal gradient nearer to the unit normal gradient. While LReLUs and PReLUs have negative values, they do not ensure a noise-resistant deactivation state. ELU saturates to a negative value with fewer inputs, diminishing the information and variance sent.

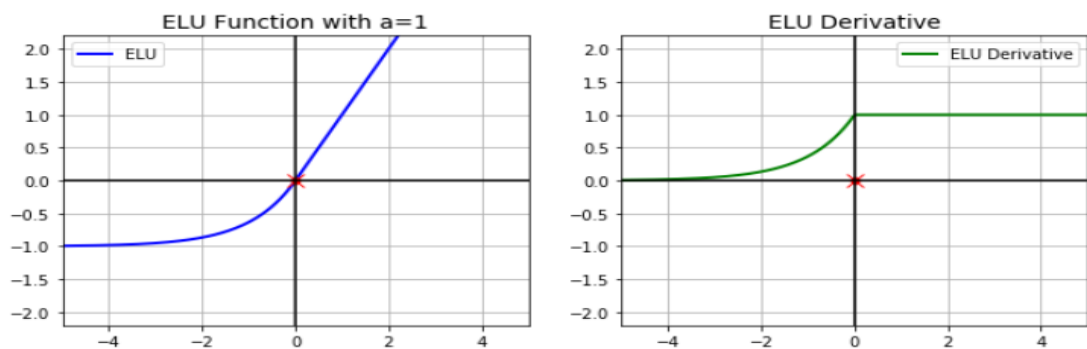


Figure 2-7 ELU activation function (left) and its derivative (right)

The exponential linear unit (ELU) with $0 < \alpha$ is defined in equation (2.17) :

$$f(x) = \begin{cases} x, & \text{for } x \geq 0 \\ \alpha(e^x - 1), & \text{for } x < 0 \end{cases} \quad (2.17)$$


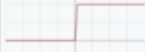


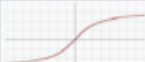



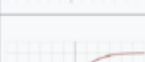
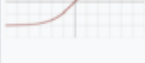



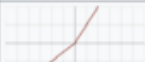






ELU derivative can be presented mathematically by equation (2.18):

$$f'(x) = \begin{cases} 1, & \text{for } x \geq 0 \\ f(x) + \alpha, & \text{for } x < 0 \end{cases} \quad (2.18)$$

ELU is more viable than ReLU because it generates smoother output, prevents the dead ReLU problem, and helps push weights and biases in the proper direction. However, the exponential operation increases computing time, it fails to learn the 'a' value, and it has the potential to worsen the gradient issue.

The table below (**Table II.2**), summarizes different activation functions developed by researchers, their names and abbreviations, their derivatives, and their graphical illustrations.

Table 2-2 Summary of different activation functions and their derivatives [67].

Name	Plot	Equation	Derivative (with respect to x)
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$ ^[1]	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
ElliotSig ^{[9][10][11]} Softsign ^{[12][13]}		$f(x) = \frac{x}{1 + x }$	$f'(x) = \frac{1}{(1 + x)^2}$
Inverse square root unit (ISRU) ^[14]		$f(x) = \frac{x}{\sqrt{1 + \alpha x^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1 + \alpha x^2}} \right)^3$
Inverse square root linear unit (ISRLU) ^[14]		$f(x) = \begin{cases} \frac{x}{\sqrt{1 + \alpha x^2}} & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \left(\frac{1}{\sqrt{1 + \alpha x^2}} \right)^3 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Square Nonlinearity (SQNL) ^[11]		$f(x) = \begin{cases} 1 & : x > 2.0 \\ x - \frac{x^2}{4} & : 0 \leq x \leq 2.0 \\ x + \frac{x^2}{4} & : -2.0 \leq x < 0 \\ -1 & : x < -2.0 \end{cases}$	$f'(x) = 1 \mp \frac{x}{2}$
Rectified linear unit (ReLU) ^[15]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Leaky rectified linear unit (Leaky ReLU) ^[16]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric rectified linear unit (PReLU) ^[17]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Randomized leaky rectified linear unit (RRReLU) ^[18]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ ^[3]	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus ^[23]		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$
Bent identity		$f(x) = \frac{\sqrt{x^2 + 1} - 1}{2} + x$	$f'(x) = \frac{x}{2\sqrt{x^2 + 1}} + 1$
SoftExponential ^[26]		$f(\alpha, x) = \begin{cases} -\frac{\ln(1 - \alpha(x + \alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x} - 1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \frac{1}{1 - \alpha(x + \alpha)} & \text{for } \alpha < 0 \\ e^{\alpha x} & \text{for } \alpha \geq 0 \end{cases}$
Soft Clipping ^[27]		$f(\alpha, x) = \frac{1}{\alpha} \log \frac{1 + e^{\alpha x}}{1 + e^{\alpha(x-1)}}$	$f'(\alpha, x) = \frac{1}{2} \sinh\left(\frac{p}{2}\right) \operatorname{sech}\left(\frac{px}{2}\right) \operatorname{sech}\left(\frac{p}{2}(1-x)\right)$
Sinusoid ^[28]		$f(x) = \sin(x)$	$f'(x) = \cos(x)$
Sinc		$f(x) = \begin{cases} 1 & \text{for } x = 0 \\ \frac{\sin(x)}{x} & \text{for } x \neq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x = 0 \\ \frac{\cos(x)}{x} - \frac{\sin(x)}{x^2} & \text{for } x \neq 0 \end{cases}$
Gaussian		$f(x) = e^{-x^2}$	$f'(x) = -2xe^{-x^2}$

2.4.2.3 ANN Architecture Design

The way that neurons are arranged among one another inside a single network is known as its architecture. Stated differently, it concerns their arrangement and relationship. The neurons used in most neural networks are of the same kind. A few further uncommon layouts rely on certain neurons. A neural network's design is determined by the task it has to learn. From inputs to outputs, a neuron network typically consists of many layers of neurons. Generally, every architectural style has a unique structure tailored to certain uses.

2.4.2.3.1 Recurrent neural networks

Recurrent Neural Networks (RNNs) are networks of neurons in which information can spread in both directions, including from the deep layers to the first layers. The most commonly used architectures are: Kohonen, Hopfield, Adaptive Resonance Theory ART...etc.

2.4.2.3.2 Feed-forward neural networks

The most basic kind of artificial neural network design is called a feedforward neural network, in which data passes from input to output only in a single direction. Every neuron in a layer is linked to every other neuron in the layer above it, indicating that the layers are completely connected.

2.4.2.3.3 Perceptron definition

One of the most basic designs for artificial neural networks is the perceptron. Frank Rosenblatt first presented it in the 1957s [69]. With just one layer of input nodes fully connected to one layer of output nodes, it is the most basic kind of feedforward neural network. The linearly separable patterns are teachable to it. It makes use of artificial neurons that are somewhat different, called threshold logic units (TLU). McCulloch and Walter Pitts presented it for the first time in the 1940s [58].

2.4.2.3.4 The single-layer perceptron

Frank Rosenblatt [59] developed a single-layer feedforward neural network in the late 1950s. It was the nascent stage of artificial neural networks as well as deep learning. Statistical machine learning or traditional code programming are employed for prediction at that period. One of the earliest and most basic types of artificial neural networks is the perceptron (**Figure II.8**). The perceptron is a simple model that has been shown to be effective in addressing certain classification problems. The patterns that can be learned by this kind of perceptron are only linearly separable. It is useful for tasks where a straight line may be utilized for separating data into distinct categories.

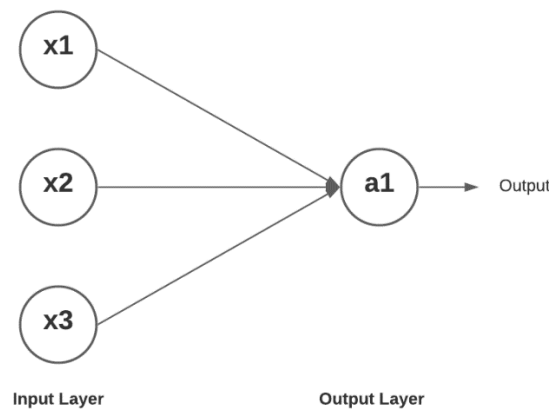


Figure 2-8 Single-layer perceptron

2.4.2.3.5 Multi-layer perceptron MLP

An MLP is a variant of the single-layer perceptron that includes one or more hidden layers between the input and output. Each neuron in a layer is linked to all neurons in the previous and next levels (with the exception of the input and output layers), yet there are no connections between cells in the exact same layer (**Figure II.9**). The number of hidden layers is generally determined by the complexity of the issue being solved. In principle, a single hidden layer may be adequate to solve any given problem, although it is possible that having numerous hidden layers makes resolving difficult problems simpler.

MLPs use a learning method to adjust their weights, with the most prevalent being gradient backpropagation, a generalized form of the Widrow-Hoff rule. The

focus is on minimizing the square error by adjusting weights from the output layer to the input layer. This technique consists of two parts.

- Inputs are sent through each layer until they reach the output layer.
- If the output of the MLP does not match the expected output, the error is passed back from the output layer back to the input layer by adjusting the weights throughout this process.

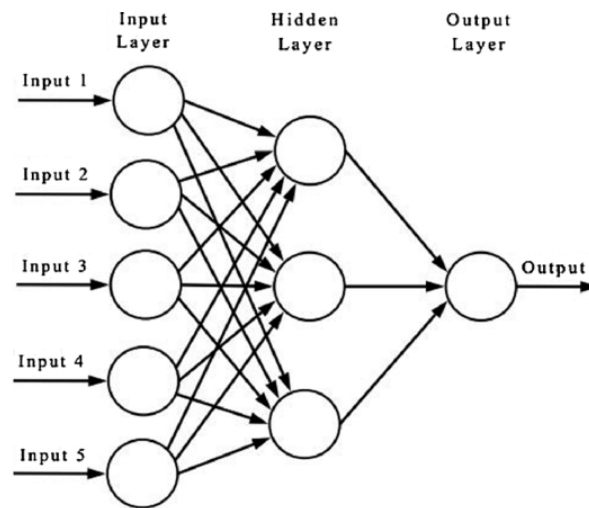


Figure 2-9 Architecture of a multi-layer perceptron (MLP)

2.4.2.4 ANN model selection

Artificial neural network models come in a variety of forms. The architecture, treatment, and learning rule of each model define it. Selecting the model that works best for a certain application requires consideration of a number of factors, some of which are listed below:

- The desired function (classification, prediction, diagnosis or recognition);
- The nature of the data to be processed. This data may be dynamic, static or random in nature and may take different forms;
- Available hardware and/or software resources for the implementation of the network;
- Time constraints generally related to real-time applications;
- Efforts to prepare the learning database as well as the test database and validation as needed;

- Learning time corresponding to the time required before considering the network as an expert and starting the decision.

2.4.2.5 The importance of Generalization in Machine Learning

Machine learning is the process of generalizing using examples to categorize previously unknown data samples. A model is deemed generalizable if it can anticipate data samples from other sets. To generalize, the learning algorithm must be exposed to diverse data subsets. However, there are two major limitations: the data samples on which the model is trained and the learning method utilized in the background. In supervised learning, the model seeks to predict a target function (Y) from a variable of input (X), and if the algorithm generalizes knowledge, the predicted variable (Y') is naturally near to the ground truth.

An approach for machine learning to be effective must function well on unknown input samples of the same kind as the training and validation data sets. Consequently, the algorithm's success ought to be assessed based on its capacity to reduce both the training error and the variance between the training and validation errors. The generalization problem is the name given to this discrepancy. The algorithm's representational capacity, or its ability to adjust to a variety of outputs, has a significant impact on both the training as well as generalization errors.

For instance, when the training model was a linear regression. The linear model cannot capture the intended behavior if the user seeks to represent data whose underlying distribution is a higher order polynomial. There will never be a sufficiently small training error. We refer to this issue as underfitting the model.

In theory concept, the algorithm will finally fully fit every single finite event set to its outputs if its representational capacity is permitted to be arbitrarily high. However, because the algorithm is unnecessarily specialized to the training set under this environment, it is less effective on new data. Overfitting is the term used to describe a situation in which an algorithm may achieve a modest training error but cannot achieve a small generalization error. Looking for a suitably complicated model that maintains its ability to generalize is the key to prevent overfitting.

The conventional relationship between error and capacity is seen in **figure II.10**.

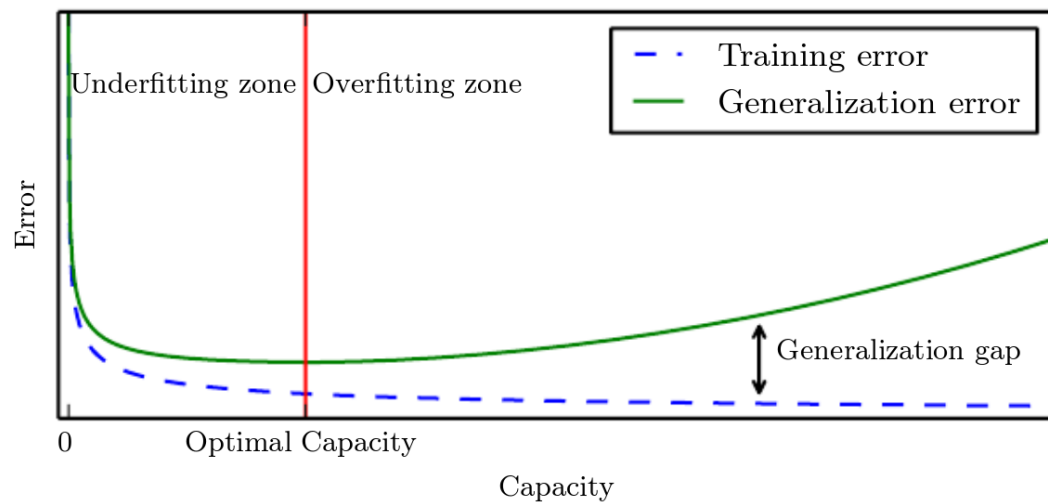


Figure 2-10 Generalization and training errors as capacity-related functions. There is an ideal capacity between the underfitting and overfitting zones.

2.4.2.6 Concept of Forward propagation and Back-propagation in neural networks

The simplest kind of neural network is the perceptron, which is based on biological neurons in artificial neural networks, as we have already discussed. Since the approach was the initial stage intended for a computer implementation for picture recognition, Geoffrey Hinton et al. [70] developed backpropagation in 1980. Neural networks are now less effective as artificial neural networks could not learn appropriately. Neurons in the input layer of the network are linked and have undergone preprocessing. The network consists of an input, hidden, and output layer. Based on the issue description, the output is utilized to assess whether or not a system is defective. The industry standard for image recognition algorithms is now using this idea. There are two principle types of propagation; forward propagation and back-propagation.

A. Forward Propagation

Assume that we have four inputs (x_0, x_1, x_2 and x_3) in the input layer of our binary classification issue. After that, it enters the neuron, but not before the neuron

receives certain weights assigned to each of the inputs denoted by $(w_0, w_1, w_2 \text{ and } w_3)$. It then goes through an activation function. A hidden neuron performs two operations: first, we sum the product of each single input value with its appropriate weight and bias.

Assuming that " y " represents the hidden neuron's output, then (2.19) :

$$y = x_0 w_0 + x_1 w_1 + x_2 w_2 + x_3 w_3 + b_i \quad (2.19)$$

We employ bias where $b_i = \text{bias}$ (it prevents the output from being zero or neutral) because the total of the product of all input values times their associated weight might occasionally equal zero. Then, we won't receive any output if we run our y value through any activation function. Bias terms will therefore aid y in become non-zero.

Next, y will pass to an activation function, with the assumption that z is the function's output (2.20).

$$z = \text{activation}(y) \quad (2.20)$$

Assuming the sigmoid activation function is used in this case, the result will be a number between 0 and 1. Accordingly, neurons will not become active if Z is less than 0.5; on the other hand, if Z is larger than 0.5, neurons will become active. Although we have only used one neuron in this instance for comprehension, there will really be many neurons, and all of the neurons in the hidden layer should have the activation function applied to them. **Forward propagation** is the name given to this kind of propagation or sometimes called **Feedforwad (Figure II.11)**.

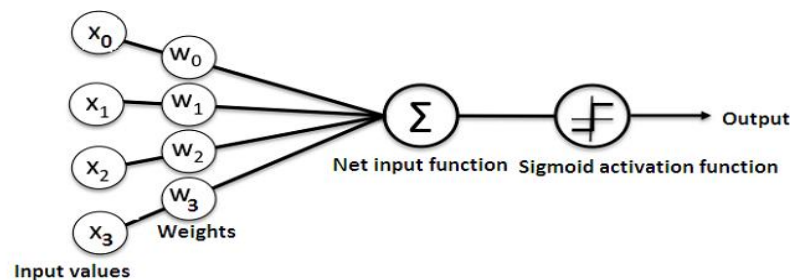


Figure 2-11 A simple neural network with forward propagation

B. Back-Propagation

In order to train a neural network to approach the target outputs of the given inputs, all of the neurons' weights are chosen. Analytically solving the neural weights of a multilayer network is challenging. The iterative resolution of weights may be accomplished simply and effectively with the help of the backpropagation algorithm [71, 72]. The optimization technique used in the traditional version is gradient descent. Although gradient descent can be time-consuming and may not always result in the overall minimum error, when used with the right settings (referred to as machine learning hyperparameters), it can actually function rather effectively. An input vector propagates through the neural network in the first stage of the algorithm. Prior to then, the network's neurons' weights were started with specific values, such as very small random numbers. Using a loss function, the output from the network is compared to the intended output, which ought to be known from the training samples. The gradient of the loss function is determined and is termed error value.

When we utilize the root mean square error RMSE to be a function of loss, the value of the error of the output layer simply represents the difference between the actual output and the desired output. To determine the error value of the neurons in the hidden layers, the error values are then transmitted across the network. The derivative chain rule may be applied to solve the gradients of the loss function of hidden neurons.

The weights of the neuron are then updated by computing the gradient of the weights and deducting a certain amount from it. The learning rate is the name given to this connection. Both fixed and dynamic learning rates are possible. After updating the weights, the algorithm repeats each step with a new entry until the weights converge. The e-learning algorithm computes weight adjustments following each new entry [71]. Online learning may result in "zig-zag" behavior, in which the gradient's estimate of a single-data point shifts direction and doesn't approach the minimum straight. Full batch learning is a further method of updating, in which weight updates are computed for the entire data set. This has additional disadvantages and is highly computationally intensive. Mini-batch learning is a compromise version where we use only a portion of the training data every update.

➤ Back-propagation formula

We've seen what a neural network looks like. The parameters—weights and biases combined—must be adapted in order to activate the neuron and enable the neural network to generate an accurate prediction. As an illustration, consider a basic neural network with three input features, $(x_1, x_2, x_3 \text{ and } x_4)$, each of which has a weight given to it $(w_1, w_2, w_3 \text{ and } w_4)$. Following the passage via a hidden layer (neuron), w_5 is assigned as the weight.

We assume it is binary classification problem hence sigmoid is used as activation function which convert the output between 0 to 1. Say that y is the actual value and \hat{y} is predicted value. The next step is to find out if actual y and predicted y are nearly same. In the event that real y differs from anticipated y , the loss function is as follows (2.21) :

$$Loss = \frac{1}{2} \sum_{i=1}^m (\hat{y} - y)^2 \quad (2.21)$$

Here, m represents the number of neurons, \hat{y} is the predicted output and y is the actual output.

We will try to minimize this Loss function.

The loss function has $value \geq 0$ when squared. We will attempt to minimize this loss function by modifying the weights of $(w_1, w_2, w_3 \text{ and } w_4)$. Assuming that predicted y is 1 and actual y is 0, our neural network prediction is incorrect and loss will be 1. It is important to modify the weight in a way that ensures the difference between the predicted and actual values of y . With an optimizer, we can reduce our loss function. Backpropagation then enters the scene, and we will modify the weight in such a manner that each iteration results in a smaller loss function. To ensure that our loss function is zero, we are going to update our old weights with the new weights. Stated otherwise, predicted and actual y ought to be equal (2.22).

$$\hat{y} \approx y \quad (2.22)$$

The suffix new will be used to signify the new weight (2.23).

$$w_{new} = w_{old} - \alpha * \frac{\delta L}{\delta w_{old}} \quad (2.23)$$

We apply this rule with our parameters to get the new weights (2.24 to 2.27):

$$w_{4,new} = w_4 - \alpha * \frac{\delta L}{\delta w_4} \quad (2.24)$$

$$w_{3,new} = w_3 - \alpha * \frac{\delta L}{\delta w_3} \quad (2.25)$$

$$w_{2,new} = w_2 - \alpha * \frac{\delta L}{\delta w_2} \quad (2.26)$$

$$w_{1,new} = w_1 - \alpha * \frac{\delta L}{\delta w_1} \quad (2.27)$$

Here, α represents the learning rate (positive and between 0 and 1) and L is the Loss function.

2.4.2.7 Optimizers

As we can see, the loss function has a number of local minima that might lead our model astray. If we can appropriately monitor and adjust the learning rate, we may prevent this from occurring. It is now not feasible to accomplish this manually; optimizers take care of it for us. It accelerates the optimization process and automatically adjusts the learning rate to keep the model from reaching a local minimum.

➤ Gradient descent

One of the strongest optimizers that exists for backpropagation is gradient descent. Every iteration, it adjusts the weight in a way that minimizes our loss function (**Figure II.12** and equation 2.28).

$$w_{1,n} = w_1 - \alpha * \frac{\delta L}{\delta w_1} \quad (2.28)$$

Where $w_{1,n}$ is updated weight, w_1 is the old weight, α is the learning rate and L is the Loss function.

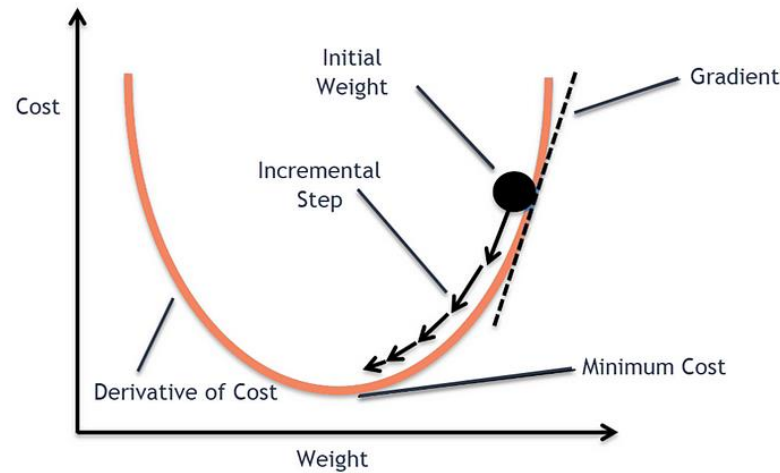


Figure 2-12 Errors vs training steps.

In the case that our loss reaches the dark black point in the figure, we need to minimize the loss. The weight will continue to increase unless we attain the global minimum. At that moment, the gradient (slope) is positive. Therefore, we will deduct the product of the learning rate with the weight at each iteration to derive the loss function, which is in fact positive. Consequently, weight will decrease with each iteration.

Conversely, if the loss occurs at this time, the slope will be negative, and our new weight will continue to rise in accordance with the new weight formula until we never reach global minima (**Figure II.13**).

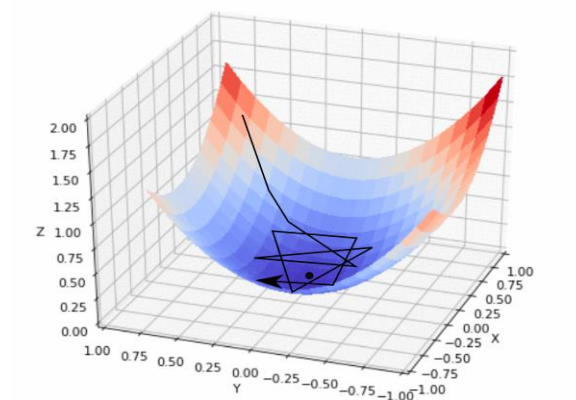


Figure 2-13 Optimization using Gradient Descent.

2.4.2.8 Hyperparameters Tuning

In the realm of machine learning, hyperparameters are configuration parameters that are established before a model's training begins. They regulate the learning process rather than being taught by data. Hyperparameters are commonly used to optimize model performance, and they may have a considerable influence on accuracy, generalization, as well as other metrics.

2.4.2.8.1 Different types of hyperparameters

Hyperparameters are configuration parameters that influence the process of learning of a machine learning algorithm. They are separate from the parameters of the model, which are the weights and biases that have been learnt from the data. There are many different kinds of hyperparameters, the learning rate, epochs, layers, architecture, and activation function are all important aspects in optimizing a neural network. The learning rate sets the step size chosen by the optimizer during training, which influences convergence speed and stability. The number of epochs reflects how many times the full dataset is run through the model. The number of layers defines the model's depth, which influences its complexity and learning capacity. The number of nodes in each layer affects the model's ability to describe complicated data connections. The architecture specifies the general organization of the neural network, such as the number of layers, neurons per layer, and connections between layers. The activation function introduces nonlinearity, which enables the model to learn complicated decision relationships.

2.4.2.8.2 Learning rate selection α

One of the most crucial hyperparameters to adjust for a neural network to perform better is learning rate. Learning Rate moves toward a loss function optimum by determining the step size after each training iteration. Typically, the symbol α represents the learning rate. The range from 0 to 1 defines the value of α . For instance, if the learning rate is $\alpha = 0.15$, then only 15% of the gradient term is updated in the weight parameters during each training iteration.

Finding the optimal learning rate is a complex task, as it controls the model's learning rate. A high rate can overshoot the minimum point, while a small rate allows

for slower learning and less convergence (**Figure II.14**). Too small can lead to undesirable local minimums. The learning rate value depends on the neural network architecture and training dataset. Therefore, it's crucial to avoid using too large or too small rates.

In order to find a suitable learning rate value — one that is neither too high nor too low — is essential to improving the performance of your neural network. There are several methods for determining the optimal learning rate:

- Learning Rate Decay
- Learning Rate Schedule
- Adaptive Learning Rate

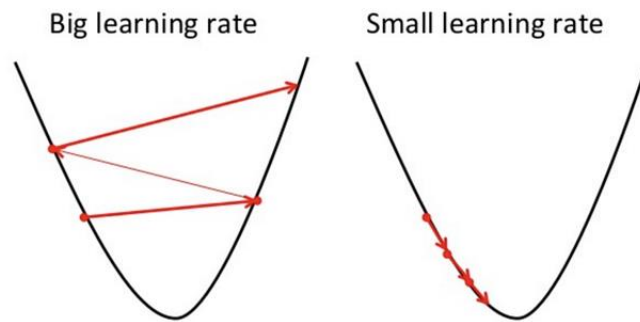


Figure 2-14 Difference between using big learning rate and small learning rate

We implemented a few hyperparameter optimization strategies to choose a suitable learning rate. The learning rate is often assumed to be 0.001. Therefore, till our cost function is not optimized, backpropagation and forward propagation will continue to execute simultaneously. We refer to this as one epoch.

2.4.2.8.3 Adding the Momentum term μ

One technique to expedite the convergence of learning is to add a word called "moment" in the learning rule, determined by the equation (2.29).

$$\mu \cdot \Delta w_{ij}(n-1) \quad (2.29)$$

So the learning equation identified by $\Delta w_{ij} = -\alpha \frac{\partial L}{\partial w_{ij}}$ becomes (2.30) :

$$\Delta w_{ij}(n) = -\alpha \frac{\partial L}{\partial w_{ij}} + \mu \cdot \Delta w_{ij}(n-1) \quad (2.30)$$

where n is the index of the current learning iteration (and therefore $n - 1$ the index of the iteration previous) and with (2.31):

$$0 < \mu < 1 \quad (2.31)$$

This momentum factor allows for an increase in the learning rate without producing weight changes, hence avoiding local minima. The weights' variation is now determined not only by the gradient but also by their past variation. This momentum provides a certain "softness" to learning via a recurrence: if the variation in weights is significant in one example, it is likely to remain so in the next.

To avoid divergence, μ must be less than one in absolute value. It thus enables the removal of minor discontinuities in the error surface, preventing it from becoming stuck in local minima. Usually, time is minimised while learning.

2.4.2.8.4 Initial Weight Values

If the initial values of the weights are very low, learning is slow as they interfere with the modification rule. If they are large, learning is also slow because the weighted sum of input in a neuron is large and the derivative of the transfer function (activation function) is low. However, this interferes with the rule of weight modification.

➤ The selection of the number of hidden layers and neurons in the hidden layer

A single hidden layer can handle most problems. Using 2 or 3 or more can define more complex regions in the input space.

If the number of neurons in the hidden layers is too large, the network will tend to carry out core learning on the learning data and therefore generalize poorly to new data. If it is too small, it will not have enough internal variables to solve the problem to be dealt with.

Finally, choosing the right number of neurons is therefore a compromise between these two aspects.

➤ **Connectivity between network layers**

Generally, a complete connectivity is used between the inputs and the different layers, that is, the neurons are connected to all the neural cells in the upper layer, and to all entries in the case of the first hidden layer.

➤ **Pros and cons of hyperparameters**

Hyperparameter tuning entails efficiently exploring and optimizing high-dimensional hyperparameter spaces, managing expensive function evaluations, incorporating domain knowledge, developing adaptive methods, and applying them to model selection, regularization, feature preprocessing, and algorithmic parameters. The advantages of hyperparameters include better model performance, less overfitting and underfitting, increased model generalizability, optimum resource use, and improved interpretability. However, it has several disadvantages, which include high computing costs, a time-consuming procedure, the possibility of overfitting, no guarantee of optimal performance, and the need for experience.

2.5 Deep Learning Overview

Deep learning is a branch of machine learning that is based on artificial neural networks. The system has the ability to acquire complex patterns and relationships from datasets. Within the realm of deep learning, there is no need for explicit programming of every aspect. The popularity of this technology has increased in recent years due to advancements in computing power and the availability of large datasets. Due to its use of artificial neural networks (ANNs), this technology is commonly known as deep neural networks (DNNs). The neural networks are designed to mimic the structure and functionality of biological neurons in the human brain, with the goal of learning from extensive datasets.

Deep learning has demonstrated significant progress across various domains, such as image recognition, natural language processing, speech recognition, and recommendation systems. Fully Connected Deep Neural Networks (FCDNNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) are among the prevalent deep learning architectures.

Deep neural networks are commonly trained using a substantial amount of data and computational resources. With the rise of cloud computing and advancements in specialised hardware such as Graphics Processing Units (GPUs), the training of deep neural networks has become more accessible.

2.5.1 Applications of Deep Learning

Deep learning has three primary applications: computer vision, reinforcement learning and natural language processing NLP.

2.5.1.1 Computer vision

Deep learning models are essential in computer vision for understanding and recognizing visual input. They are used to identify and recognize objects, as well as to classify and segment images. Object detection and recognition allow machines to execute functions such as self-driving automobiles, surveillance, and robotics. Image categorization divides pictures into groups such as plants, animals, and buildings, which is important for medical imaging, control of quality, and image retrieval. Image segmentation enables the discovery of certain characteristics inside pictures.

2.5.1.2 Natural Language Processing

Deep learning models in Natural Language Processing (NLP) enable machines to interpret and generate human language. Automated systems have the capability to generate content like summaries and essays, in addition to facilitating text translation across different languages and enabling communication among individuals with varying linguistic proficiencies. It is possible for these systems to identify positive, negative, or neutral sentiment in text, a capability that holds significance for applications such as customer service, social media tracking, and political evaluation. Deep learning models have the capability to identify and transcribe spoken words, facilitating tasks like speech-to-text conversion, speech recognition, and vocal-controlled devices.

2.5.1.3 Reinforcement Learning RL

Deep learning is a training approach for reinforcement learning that maximizes rewards in a variety of situations. It has been used effectively in games such as Go, Chess, and Atari, as well as in robotics to train robots for complicated tasks such as object recognition and navigation. Deep reinforcement learning models may also be used to manage complex systems such as power grids, traffic flow, and supply chain optimization.

2.5.2 Deep Learning Model Architectures

Deep Learning models can automatically learn characteristics from data, making them ideal for applications like image identification, audio recognition, and natural language processing. Deep learning's most prevalent designs include feedforward neural networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs).

2.5.2.1 Fully connected Deep neural network (FCDNN)

A fully connected deep neural network (FCDNN) consists of an input layer and one or several hidden layers that are connected one after another. Each neuron gets information from either the preceding layer's neurons or the input layer. The output of a single neuron becomes an input to the next neurons in the network's next layer, and so on until the network's output is produced by the final layer. The neural network's layers apply a sequence of nonlinear changes to the input data, allowing the network to learn complicated representations of the data.

Figure II.15 represents a fully connected deep neural network.

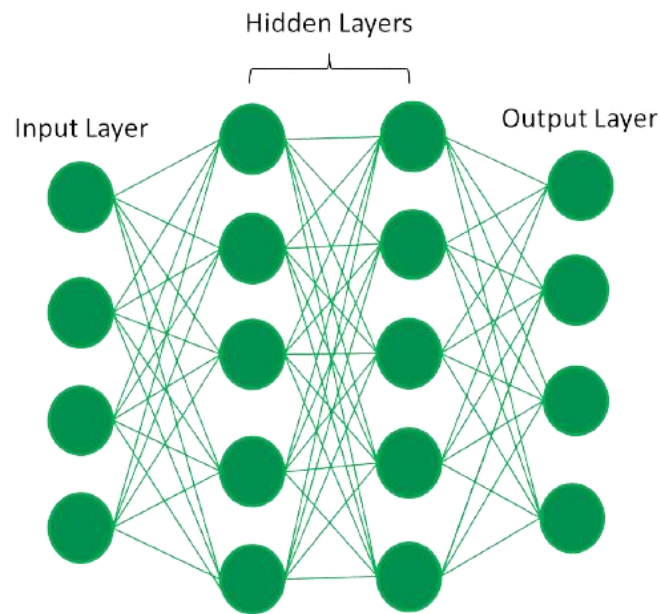


Figure 2-15 Fully connected Deep neural network

2.5.2.2 Recurrent Neural Networks RNNs

Recurrent Neural Networks (RNNs) are a type of artificial neural network designed to process sequential input data. Unlike traditional feedforward neural networks, Recurrent Neural Networks (RNNs) have the ability to take into account previous states of a sequence when analysing the current state, enabling them to capture temporal dependencies in data. RNNs are characterised by the existence of recurrent connections among the hidden units (**Figure II.16**), allowing information to flow from one-time step to the next. This implies that both the current input and the previous hidden state have an impact on the hidden state at each time step.

RNNs are machine learning models that employ a vector to analyze the current state of the sequence at each time step, along with a vector to denote the projected value or classification. The hidden state is a vector that is updated based on the current input and the previous hidden state. The vanishing gradient issue, however, hinders the fundamental RNN structure and makes training on lengthy sequences difficult.

Variations such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks were developed to address this issue. Recurrent Neural Networks

(RNNs) are commonly applied in various fields such as speech recognition, language simulation, automatic language translation, sentiment analysis, and stock forecasting. Overall, recurrent neural networks (RNNs) have demonstrated effectiveness in analyzing sequential data and forecasting temporal connections.

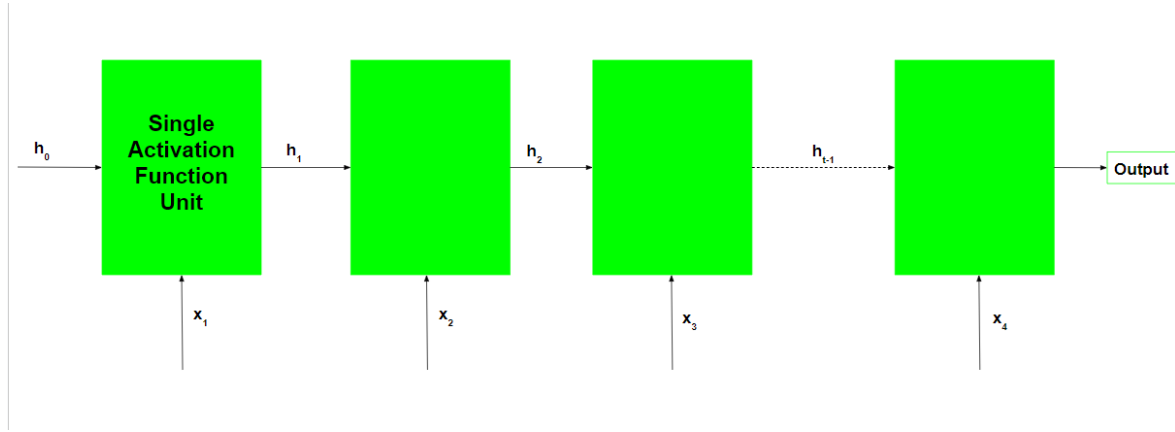


Figure 2-16 Reccurent Neural Network work-flow

Machine learning techniques are used to manage a wide range of data, including sequential data, which is challenging to anticipate due to its order-independence. To deal with this sort of data, Recurrent Neural Networks were created, which differ from conventional Artificial Neural Networks in structure. Instead of a linear feed-forward or back-propagation procedure, recurrent networks employ a recurrence relation and back-propagation through time to learn.

The Recurrent Neural Network is made up of numerous fixed activation function units, one for each time step. Each unit has a secret state. This hidden state represents the prior information that the network now possesses at a particular time step. This concealed state is updated at each time step to reflect any changes in the network's knowledge of the past. The concealed state is updated using the following recurrence relation in equation (2.32):

$$h_t = f_w(x_t, h_{t-1}) \quad (2.32)$$

Where h_t represents the new hidden state, h_{t-1} is the old hidden state, x_t is the current input and f_w the fixed function with trainable weights

2.5.3 Advantages and Disadvantages of Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are effective machine learning tools because of their capacity to process sequential data, remember prior inputs, and be adaptable. They may be applied to applications such as natural language processing, audio recognition, and time series analysis. RNNs may be integrated with various neural network designs to form hybrid models for specific tasks. However, they have limitations, such as the vanishing gradient problem, which makes understanding long-term dependencies difficult. Furthermore, RNNs can be computationally costly, particularly when analyzing lengthy sequences or utilizing complicated topologies. Despite these disadvantages, RNNs are nevertheless a useful tool for many machine learning applications because of their capacity to analyze sequential input and maintain memory.

2.5.4 Challenges faced in Deep Learning

Deep learning has made great progress in a variety of domains, but it still confronts obstacles such as data availability, computing resources, time-consuming work with sequential data, interpretability, and overfitting. Data availability is critical for deep learning, which demands enormous volumes of data for training. Computational resources are expensive because of specialized gear like GPUs and TPUs, and processing sequential data might take days or months. Interpretability is challenging owing to the complexity of deep learning models, and overfitting happens when the model becomes overly specialized for the training data, resulting in poor performance on unknown data.

2.5.5 Advantages and disadvantages of Deep Learning

Deep learning algorithms are very accurate, scalable, and flexible in applications such as image identification and natural language processing. They can automatically identify and learn useful characteristics from the data, which eliminates the need for manual feature development. These models can handle enormous datasets and learn from vast volumes of data. They may be used with a variety of data types and constantly enhance their efficiency as more data is collected. However, they have significant processing demands, need a large quantity of labeled data, and can be

difficult to understand. Furthermore, they may overfit training data, resulting in poor performance on fresh data. Finally, they are frequently considered black boxes, making it difficult to understand their operation and forecasts. As a result, these constraints must be carefully examined while selecting deep learning for a specific task.

2.6 Difference Between Artificial Intelligence, Machine Learning, and Deep Learning

The table below compares the key differences between Artificial Intelligence, Machine Learning, and Deep Learning notions.

Table 2-3 key differences between Artificial Intelligence, Machine Learning and Deep Learning

Artificial Intelligence	Machine learning	Deep Learning
AI refers to Artificial Intelligence, and it is essentially the study/process that allows machines to imitate human behaviour using certain algorithms.	ML signifies Machine Learning, which is the study of statistical approaches that allow machines to develop with experience.	DL means Deep Learning, and it is the study of using neural networks (similar to neurons found in the human brain) to mimic human brain capability.
AI is a wider family that includes ML and DL as its areas.	Machine learning (ML) is an AI method that enables systems to learn from data.	Deep learning (DL) is a machine learning technique that analyses input and produces appropriate output using deep neural networks (with several layers).
AI's efficiency is essentially the same as that of ML and DL.	It is less effective than DL since it cannot operate with longer dimensions or larger amounts of data.	More efficient and powerful than ML since it can handle massive amounts of data.

AI applications include Google's AI-powered predictions, ridesharing apps such as Uber and Lyft, commercial flights that use AI Autopilot, and so on.	Examples of machine learning applications include virtual personal assistants such as Siri, Alexa, and Google, as well as email malicious content and spam filtering.	Examples of DL applications involve sentiment-based news aggregation, image analysis, natural language processing, speech recognition and caption generation, among others.
---	---	---

2.7 Conclusion

In this chapter, we have seen that machine learning algorithms with low error rates and short computation times, such as SVM and KNN, perform well. With continuous multidimensional functionality, SVM performs worse. Both overfitting and underestimation may result from the hidden layer size selection. SVMs are more precise than neural network; however, KNNs can be inefficient, need a lot of storage, and are susceptible to noise and unimportant features. When multidimensional or continuous functionality is present, as well as when multicollinearity and input-output interactions are present, SVMs and neural networks function well. It is advised to benchmark and choose the most promising algorithm for a particular application.

A neural network's hyperparameters include the learning rate, epochs, number of layers, nodes per layer, architecture, and activation function. The learning rate determines the size of the steps taken during training, whereas epochs indicate how many times the full dataset is run through the model. The number of layers, nodes per layer, architecture, and activation function are all factors that influence the model's performance and learning capacity.

Deep learning is a neural network-based approach for extracting higher-level characteristics from input data. It has greatly enhanced performance in several artificial intelligence subfields, including machine vision, speech recognition, and image classification.

Recurrent neural networks (RNNs) are instances of deep learning since they transmit signals through layers repeatedly. RNNs can be trained via gradient descent; however, the problem of vanishing gradients may trigger long-term gradients to

disappear or expand. In most circumstances, the long-short-term memory (LSTM) approach can help to avoid this problem. Convolutional neural networks are commonly employed in deep learning, with each neuron receiving input from a specific area of the preceding layer, resulting in a hierarchical structure analogous to that of an animal's visual cortex.

In the 1990s, convolutional neural networks, in particular, were frequently employed in artificial neural networks. By demonstrating a notable improvement in picture classification accuracy as part of the ImageNet Large-Scale Visual Recognition Challenge (**ILSVRC**), several writers sparked renewed interest in CNNs. The creation of a sizable CNN with millions of labeled photos was the cause of this discovery. CNN is chosen as the best entrant algorithm due to the volume of visual data now accessible as well as the actual proof for its high accuracy used for many data types. We will touch on the CNN algorithms in the next chapter.

3 Chapter III

CONVOLUTION NEURAL NETWORKS

3.1 Introduction

Convolution Neural networks are utilized for image-related activities, which usually include pixel data. Even with 256x256 RGB pixels, the input has approximately 200,000 elements. The fully-connected layer links all neurons in the subsequent layer, resulting in billions of weights. The localization of information in pictures provides for a more straightforward approach, since crucial information may be extracted from local neighborhood relationships. For example, sharp contrasts show edges, aligned edges produce lines, joined lines provide circles and contours, circles outline a wheel, and several neighboring wheels indicate the presence of an automobile. Convolutional neural networks take advantage of information locality by replacing fully connected layers with layers of convolution.

CNN is a deep learning kind of artificial neural network that is used for image processing. CNN designs have developed significantly since their inception, accomplishing outcomes that were previously only attainable with human interaction. Today, a variety of convolutional neural network architectures exist, but they are too complex to perceive and are frequently viewed as black boxes. Now, let's take a look at the history of CNN architectures and their earliest topologies.

3.2 Convolutional Neural Network CNN definition

Convolutional neural networks are now one of the most significant deep learning techniques based on image data. Unlike classical machine learning, which requires human extraction of relevant functionality, deep learning learns specific

capabilities using raw pictures as input. CNNs are made up of three types of layers: input and output, as well as multiple hidden layers in between. Intermediate layers include convolutional layers, max pooling layers, and activation layer and many more. CNN designs differ in the number and kind of layers used for a given application. For continuous response, the network must contain a regression layer at the end, but for categorical responses, the system must have a classification function and layer.

In the convolutional layer, neurons link image areas to generate a three-dimensional output. The CNN model has a large number of hidden layers. The activation in each layer is adjusted using differentiable functions. Typically, different types of layers are used to generate CNN configurations, which we shall discuss and explain in this section.

3.3 The layers of CNNs with their operating mode

A convolutional neural network comprises various layers, such as fully connected, pooling, convolutional, input, and output layers. The Convolutional layer convolves filters with the input image to extract features; the pooling layer performs downsampling to reduce computational complexity; and the fully connected layer is responsible for making the final prediction.

The optimal filters are learned by the network through the processes of backpropagation and gradient descent (**Figure III.1**).

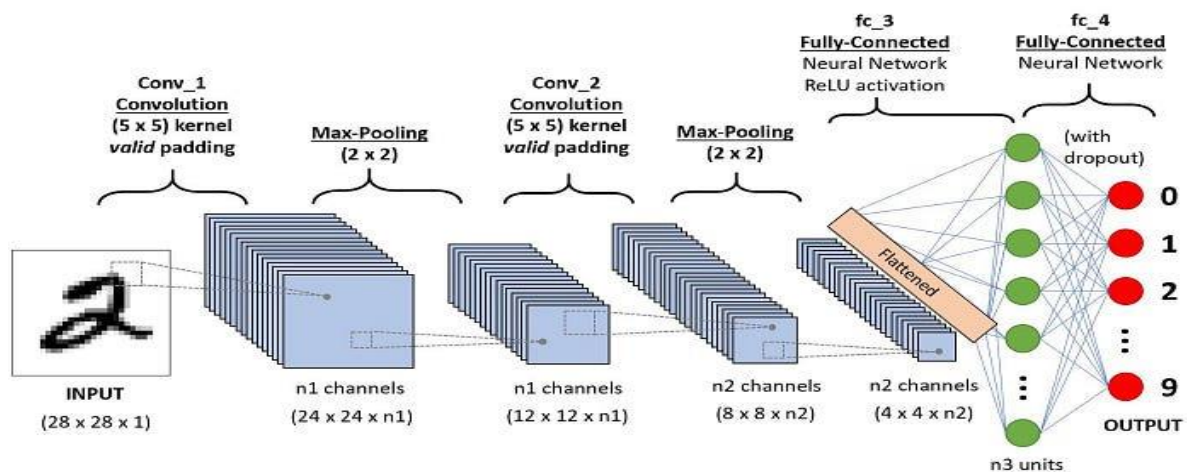


Figure 3-1 A convolutional neural network model with different layers.

Convolution Neural Networks, or ConvNets, are neural networks with shared parameters. If we got an image. It may be represented by a cuboid with three dimensions: length, width, and height (**Figure III.2**).

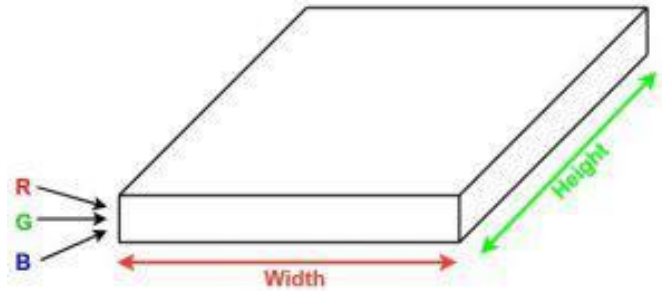


Figure 3-2 Representation of an image with RGB colors.

By selecting a small patch of the image and applying a neural network filter or kernel with K outputs, it is possible to represent the outputs vertically. When the neural network is slid across the entire image, a new image is generated with different dimensions in terms of width, height, and depth. Instead of the traditional red, green, and blue (R, G, and B) channels, additional channels with decreased width and height are now available. This process is commonly referred to as convolution. When the patch size matches that of the picture, the neural network will exhibit regular behaviour. Due to the restricted patch, there are a reduced number of weights available.

3.3.1 Kernel or the filter importance

The filter, or sometimes refers to as kernel, plays a vital role in convolutional neuron networks. The filter, like a picture, is a matrix of values, however it is generally small and square. The most typical filter size is 3 by 3. The image below depicts an example of matrix values used as a filter (**Figure III.3**).

1	0	0
1	0	0
0	1	1

Figure 3-3 Example of values of a matrix used as a filter.

A filter is used to highlight particular features of a picture (color, shape, brightness, sharpness, etc.). This kernel will be applied to the image gradually. **Figure III.4** depicts the path of the filter window over the picture.

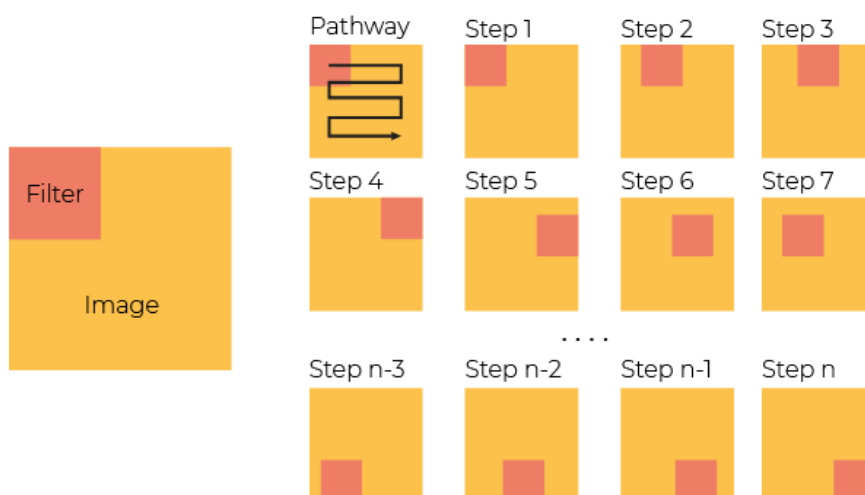


Figure 3-4 The pathway of the kernel moving across different parts of the image.

For each filter point, the two overlapping matrices (filter and image to be processed) are multiplied. Each inferred value is projected onto a new matrix. This matrix represents a new picture that highlights the characteristics that were searched with the filter.

3.3.2 Different convolution layers

Convolution is a mathematical operation that allows for the combination of two sets of data. Convolutional Neural Networks (CNNs) utilise convolutional operations to process the input data and generate a feature map. The filter, also referred to as a kernel or feature detector, may have dimensions of 3x3. When performing convolution, the kernel iterates over the input image, conducting matrix multiplication on each element. The feature map documents the outcomes for each receptive field, which is the area where convolution takes place. The filter is adjusted continuously until the feature map is fully generated (**Figure III.5**).

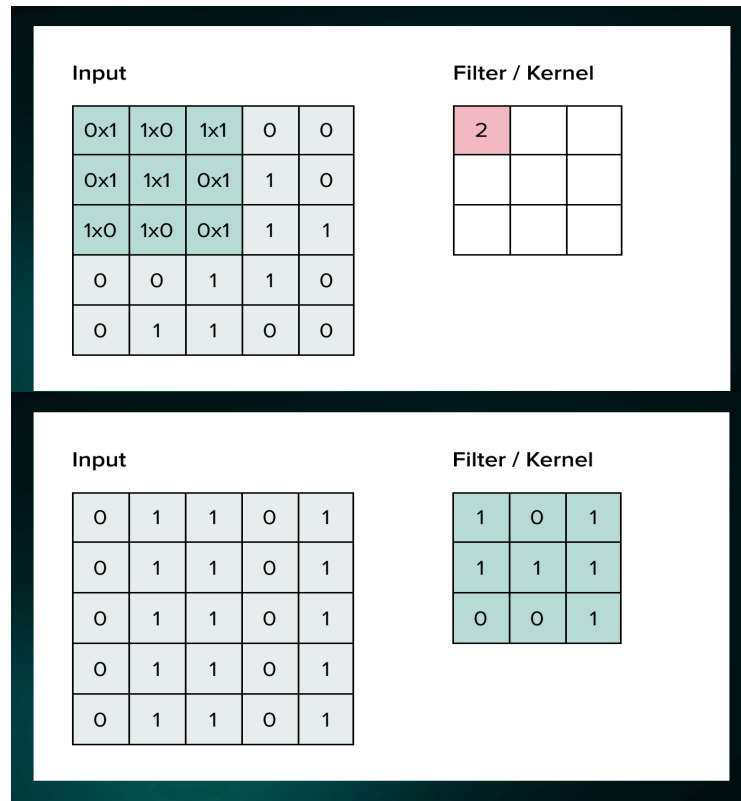


Figure 3-5 Convolution operation example

3.3.2.1 The convolution layer

The initial layer in convolutional neural network will always be a convolutional layer that serves as the feature extractor. This layer and succeeding layers execute convolutional operations on the input by sliding $m \times m$ filters (kernels) to find texture information. The convolutional layers provide a series of feature maps, which are the cross-correlation product (despite the name) of each filter and the input. These feature maps give information on the semantic meaning of the image, which is learnt hierarchically throughout the design. For an image with two dimensions I and a two-dimensional filter or the kernel K , the feature map G can be obtained as follows (3.1):

$$G(i, j) = (I \circledast K)(i, j) = \sum_u \sum_v I(i + u, j + v) \cdot K(u, v) \quad (3.1)$$

The convolution layer uses a $5 \times 5 (3 \times 3)$ filter to scan the whole image. In the output, we obtain an "image" size of $28 \times 28 \times 1$. It's called an activation map. In general, a convolution layer applies a collection of k filters rather than only one. We then acquire a stack of k "activation maps" which build up our new "image". **Figure**

III.6 illustrates how convolving (04) filters over an input image produces four unique feature maps.

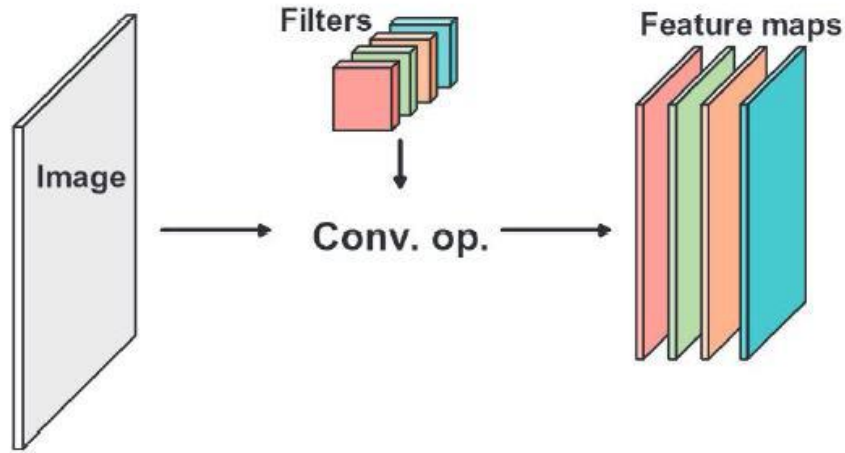


Figure 3-6 Operation of Convolution using four filter results in four feature maps.

Convolution calculation is rather easy: for a 3x3 convolution, we take the first 3x3 pixels from the input picture and produce a new one in the output image. This new value is equivalent to the following: up to the ninth pixel, "pixel 1 of the selection in the input image" * "pixel 1 of filter" + "pixel 2 of selection in input image" * "pixel 2 of filter" +... Next, we will utilize the convolution's "stride" parameter to determine the output image's second value. It indicates the number of pixels in the input picture that will be moved in order to apply the convolution again.

The following figure represents an illustration of the convolution procedure. Equation (3.2) is used to calculate it, and H is the convolution's resultant product.

$$H = \sum_{k=0}^M \sum_{l=0}^N \sum_{i=0}^R \sum_{j=0}^C I_{(i+k),(j+l)} F_{(R+i-1),(C-j+1)} \quad (3.2)$$

With;

I : The pixel matrix representing the image;

F : The convolution filter matrix to be applied to the image;

H : resulting from the convolution product, called feature map;

R : Number of rows in the filter matrix;

C : number of columns in the filter matrix;

$M = \frac{M'-R}{S} + 1$ with M' is the number of lines in the image matrix, and S is the number of steps in the filter following the line and column.

$N = \frac{N'-R}{S} + 1$ with N' the number of columns in the image matrix.

The size of an output volume is controlled by three parameters: depth, stride, and zero-padding size, which we will get into in detail below.

3.3.2.2 Depth

The depth of the output volume determines the number of neurons (or filters) in the convolutional layer that are connected to a specific region of the input volume. Each filter produces an activation map that is triggered by the presence of oriented edges, blobs, or colours.

For a particular Convolution layer, the depth of an activation map will be K , which is just the number of filters that are learning in this layer. The depth column refers to the collection of filters that are "looking at" the same input point (x, y) .

3.3.2.3 Stride

A common requirement when dealing with a convolutional layer is to have an output that is smaller than the input. This may be accomplished by using a pooling layer (we will discuss later) or striding, which skips regions the kernel slides across, reducing the spatial resolution and increasing the computing efficiency of the network. The strides are the number of pixels that are moved to the input matrix. By limiting the filter's convolution against the input, steps prevent features from being lost during picture flattening. They indicate how many steps there are in every convolution. Filters are moved one pixel at a time when the total number of strides is one; they are moved to two pixels when the number is two, and so on (**Figure III.7**).

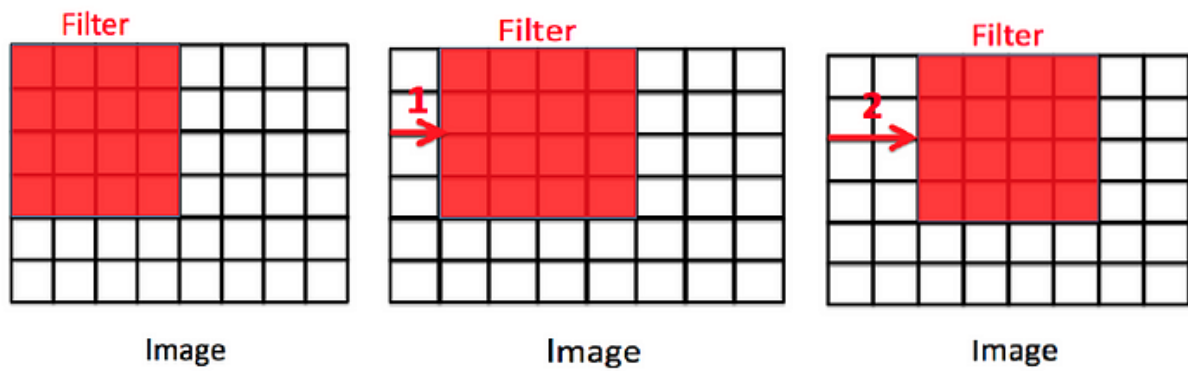


Figure 3-7 An image with different stride values : 0, 1, 2 respectively from the left to the right.

3.3.2.4 Zero-Padding

Padding is essential to the creation of CNN. Following the convolution process, the image's initial size is reduced. Also, we don't want our original image to be downsized after each step in the image classification assignment due to the several convolution layers. Second, there is an overflow because the kernel crosses the central layer more frequently than the outer layers when moving over the original image.

Padding is a novel idea that was proposed to solve this issue. It is an extra layer that may increase an image's borders without changing the size of the original (**Figure III.8**). The researcher used the term zero-padding when using zeros in all the borders of the image. The figure below illustrates the zero-padding concept :

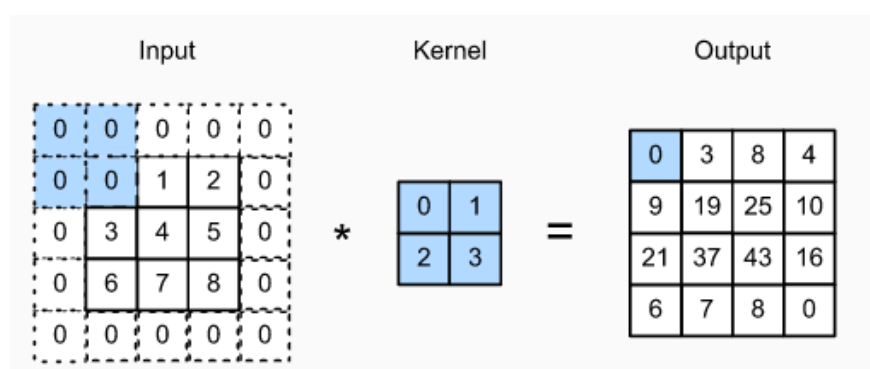


Figure 3-8 Padding the borders with zeros

Without the zero-padding, the input volume's spatial dimensions would shrink too rapidly, making it unfeasible to train deep networks. Thus, the size of the resulting

image will be as follows if an $n \times n$ matrix is convolved with a $f \times f$ matrix and a padding of p :

$$Size = (n + 2p - f + 1) \times (n + 2p - f + 1) \quad (3.3)$$

By expanding fake pixels to the matrix's borders, adding expands the input matrix. Convolution causes the matrix's size to decrease, which is why this is done. For instance, when a filter is applied to a 5×5 matrix, it becomes a 3×3 matrix.

To provide the convolutional layer's output size calculating algorithm. The following input information is provided:

The input image dimensions : $W_{in} \times H_{in}$

The filter is of dimension $K \times K$

The stride is S and the padding is P .

The dimensions of the output activation map are going to be as follows:

$$W_{out} = \frac{W_{in} - K + 2P}{S} + 1 \quad (3.4)$$

$$H_{out} = \frac{H_{in} - K + 2P}{S} + 1 \quad (3.5)$$

For example, we'll demonstrate how to calculate a convolutional layer's output size. Assume for the moment that we have a 125×49 input image, a 5×5 filter, padding $P = 2$, and stride $S = 2$. The following are the output dimensions in such case (eq 3.6 and 3.7):

$$W_{out} = \frac{125 - 5 + 2 \times 2}{2} + 1 = \frac{124}{2} + 1 = 67 \quad (3.6)$$

$$H_{out} = \frac{49 - 5 + 2 \times 2}{2} + 1 = \frac{48}{2} + 1 = 25 \quad (3.7)$$

Consequently, the activation map output will have these dimensions : $[67, 25]$

Figure III.9 gives an example of discrete convolution. The green grid is the matrix of pixels representing the input image with the following pixel value (5×5 image matrix whose pixel values are 0, 1) (eq 3.8):

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad (3.8)$$

The yellow grid represents our feature detector or kernel and the feature map resulting is presented with pink. To keep the drawing simple, a single input feature map (pixel matrix representing the image) is represented, but it is not uncommon for several feature maps to be stacked on top of each other. A core (filter) of value (filter matrix as 3x3) (eq 3.9):

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad (3.9)$$

This kernel is dragged onto the input feature map. At each location, the output between each element of the kernel and the input element that it overlaps is calculated and the results are added to obtain output at the current location. The procedure can be repeated using different cores to form as many output feature maps as you want.

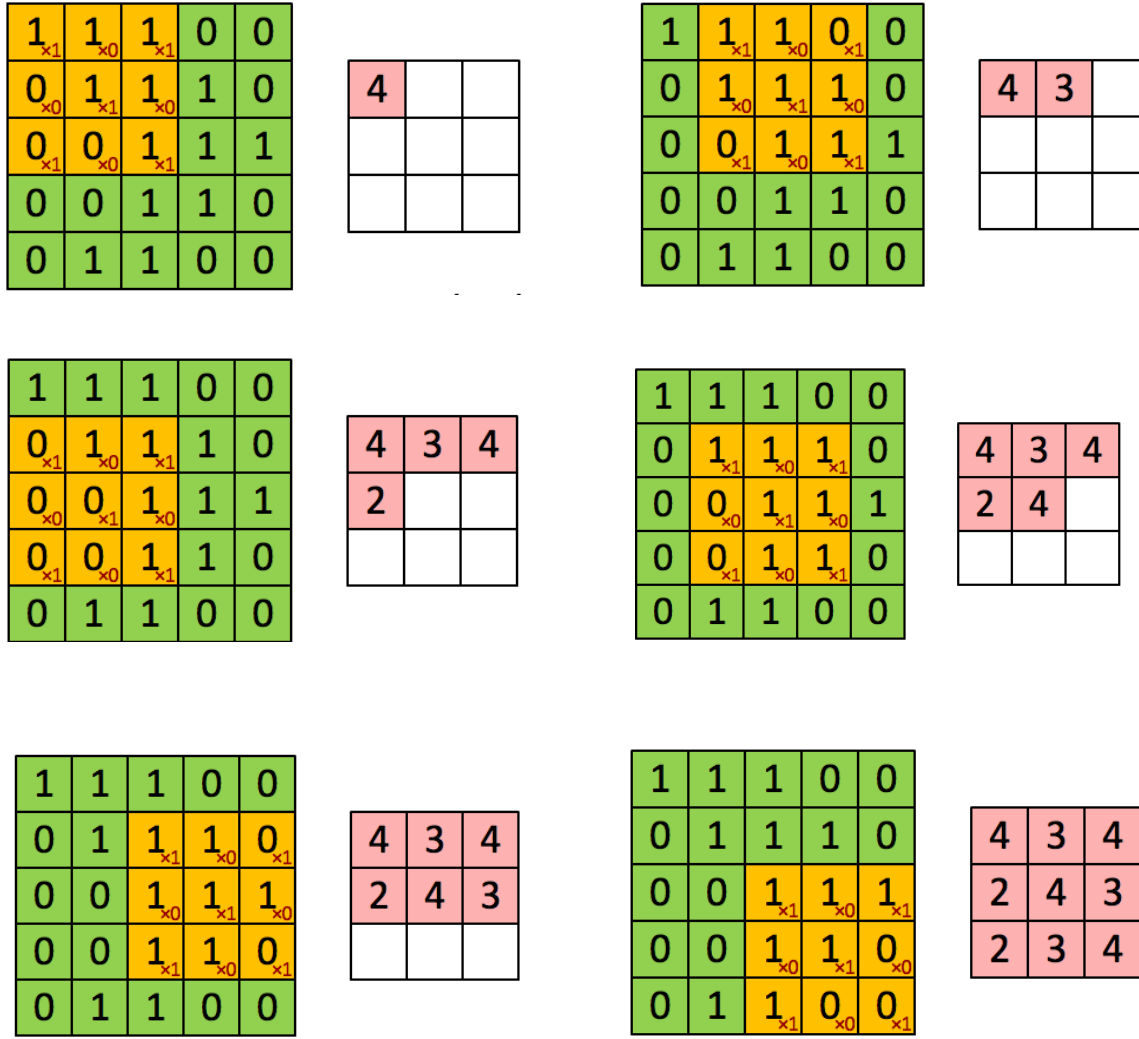


Figure 3-9 Calculating the output values of a convolution operation.

After multiplying a 5x5 image by a 3x3 filter, the output matrix of the final convolution layers will be (eq 3.10):

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 3 & 4 \\ 2 & 4 & 3 \\ 2 & 3 & 4 \end{bmatrix} \quad (3.10)$$

We previously explained how convolution layers operate above. They are important to CNNs, allowing them to detect characteristics in pictures on their own. However, the convolution method produces a significant quantity of input, making it difficult to train the convolutional neural network. To compress the data, we need to process using pooling.

3.3.2.5 Correction layer

The correction layer plays the role of the activation function. It is applied after the convolution and leaves the output volume size unchanged. In this context, we find, for example, that the activation function ReLU (Rectified Linear Units) refers to the non-linear real function defined by $ReLU(x) = \max(0, x)$. The ReLU correction layer thus replaces all negative values received as entries with zeros (**Figure III.10**).

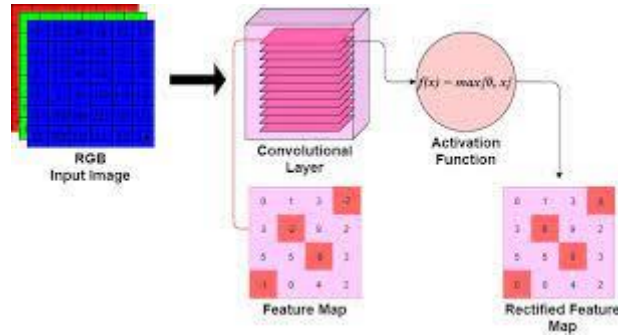


Figure 3-10 Feature map correction using ReLU

Often, **ReLU** correction is preferable, but there are other forms such :

Hyperbolic tangent correction (eq 3.11)

$$f(x) = \tanh(x) \quad (3.11)$$

Sigmoid function correction (eq 3.12)

$$f(x) = (1 + e^{-x})^{-1} \quad (3.12)$$

Correction using the ELU function (eq 3.13)

$$f(x) = \begin{cases} x, & \text{for } x \geq 0 \\ \alpha(e^x - 1), & \text{for } x < 0 \end{cases} \quad (3.13)$$

An activation function is used to process nodes in the feature map. It should be noted that when negative numbers reach a certain threshold, they are outputted as 0. (For more details about activation function refer to **chapter II**)

3.3.2.6 Pooling layer

Pooling is a layer that is frequently used after a convolutional layer. Pooling layers facilitate the compression of observed shape information into a smaller number of pixels. Every pixel in the input picture will be recorded by the convolutional layer's filters, but the forms they identify typically span many nearby pixels; for example, a line may occasionally be a few pixels long and a few pixels broad. The convolutional layer creates a picture, which is then scanned by pooling layers. They perform more straightforward procedures, such as averaging the data (average pooling) or taking the maximum value they see (max pooling) or sum pooling. There are three types of spatial pooling in fact: max, average and sum pooling.

➤ Max Pooling

Max pooling is a rule that helps to proceed with the most significant features from the picture by taking the maximum of a region. The method converts continuous functions into discrete equivalents using samples (**Figure III.11**). Its main goal is to downsize an input by decreasing its dimensionality and adopting features of the rejected sub-region that are present in it.

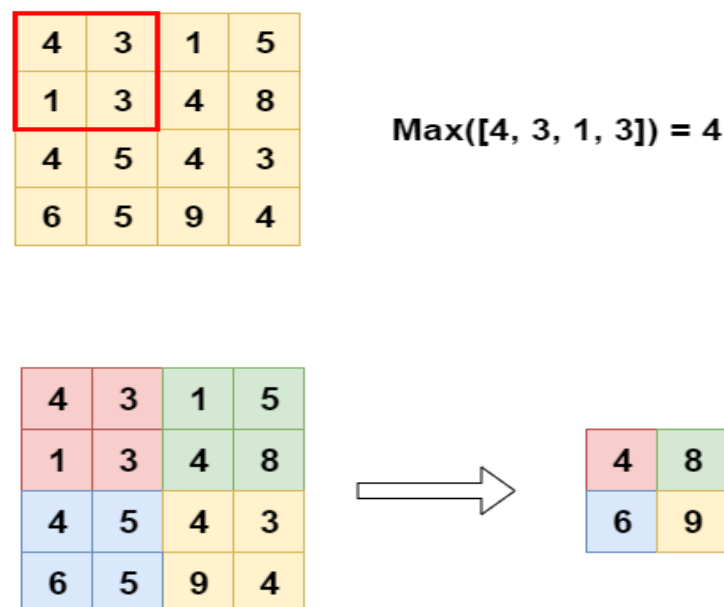


Figure 3-11 The application of (2×2) max pooling in (4×4) image

➤ Average Pooling

It differs from Max Pooling in that it preserves data regarding the less significant features. It basically downscales by splitting the input matrix onto rectangular areas and computing the average values in each one (**Figure III.12**).

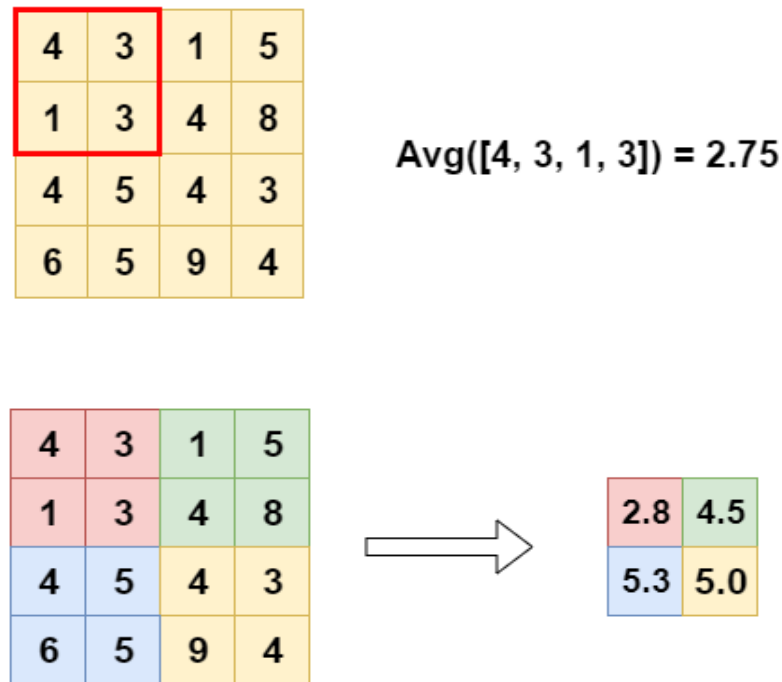


Figure 3-12 Average Pooling example

➤ Sum Pooling

The subregions in Sum pooling are identical as for Max pooling, except instead of using the max function, we use the sum (**Figure III.13**).

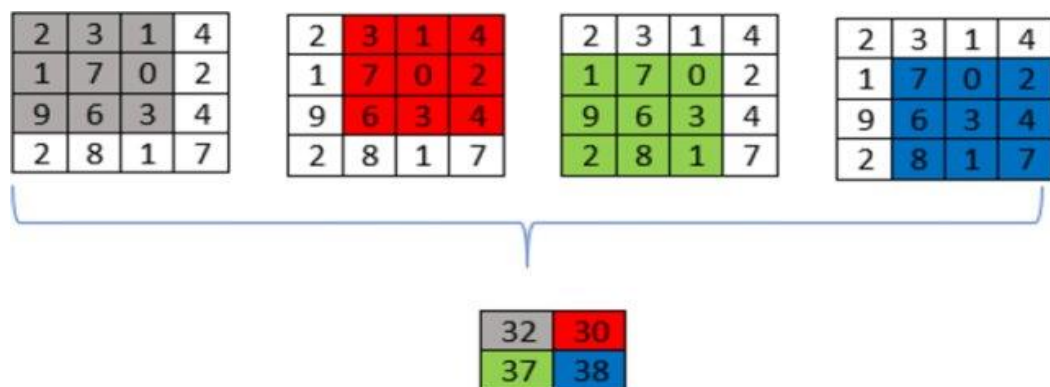


Figure 3-13 A Sum pooling with Image size (4×4), Filter size (3×3) and Strides= 1 [73]

We now need to flatten the input (convert it to a column vector) before passing it to a standard neural network for classification task.

3.3.2.7 Flattening layer

The flattening layer's intuition is to convert the data into a one-dimensional array that will be sent to the following layer. We flattened the convolutional layer's output into a single long feature vector. This connects to the final categorization model, known as the fully linked layer. Assume we have [5, 5, 5] pooled feature maps are flattened to 1x125 single vectors (**Figure III.14**). So, flattening layers reduces a multidimensional array to a single-dimensional vector.

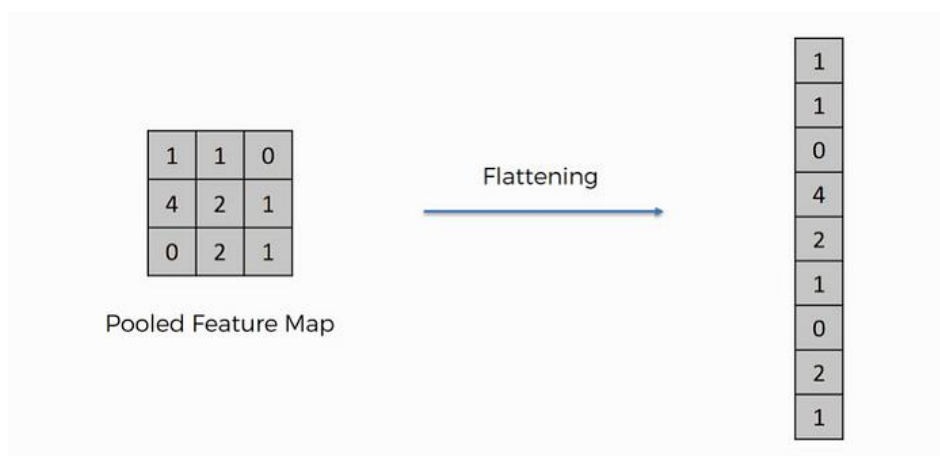


Figure 3-14 Flatten layer operation

After the flattening stage, we will have a lengthy vector of data input that we can then run through a fully-connected layer to be processed further.

3.3.2.8 Fully-connected layer

Fully-connected layers are commonly used in neural networks (NNs) and play a vital role in classification models. **Figure III.15** illustrates the connections of each fully connected (dense) layer with the layers before and after it. This suggests that each neuron within this layer receives input from all neurons in the preceding layer and sends output to all neurons in the subsequent layer. The process involves matrix-vector multiplication, where the vector denotes the input and the matrix values signify the weights of the layer, which requires training to establish the intricate relationship

between the input and the output. The final outcome of the classification process is a k -dimensional vector representing the distribution of membership in the input data (see **figure III.15**).

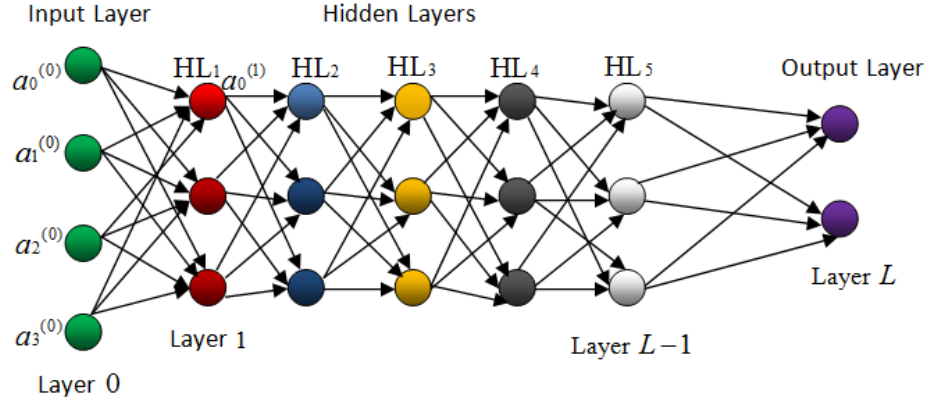


Figure 3-15 A network of fully-connected layers. The dense block's final output is a membership distribution over classes.

3.3.2.9 Batch Normalization

Batch normalization serves to overcome concerns with internal covariance shift in feature maps. The internal covariance shift is an adjustment in the distribution of hidden unit values that slows convergence (by driving the learning rate to a low value) and necessitates careful parameter setting [74]. Equation (3.14) shows batch normalization of a changed feature map T_l^k .

$$N_l^k = \frac{F_l^k - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (3.14)$$

N_l^k Is normalized feature map. F_l^k Represents the input feature map. μ_B and σ_B^2 are the mean and the variance of the feature map for a mini batch respectively. Batch normalization normalizes the distribution of feature map data, reducing them to zero average and unit variance. Furthermore, it softens the gradient flow and functions as a regulatory component, so enhancing the network's generalization.

3.3.2.10 Loss layer (LOSS)

The final layer in the network is referred to as the loss layer. The error between the neural network's estimation and the real value is computed. During a classification test, the random variable is considered discrete as it can only assume one of two values: 1 if it belongs to a class and 0 if it does not. This is why the cross-entropy function is the most often used and appropriate loss function. This function is derived from information theory and evaluates the overall difference among two probability distributions (the model's forecast and the actual) for a random variable or sequence of events. Officially, it is written in equation 3.15:

$$LOSS(x, class) = - \sum_{class=1}^C y_{i, class} \text{Log}(p_{x, class}) \quad (3.15)$$

Where y represents the estimated probability that the value x belongs to class i , and p is the true probability that x belongs to class i , provided that there are C classes.

3.3.3 Training a Convolutional Neural Network

The process of training a convolutional neural network (CNN) involves updating the values of its weights. The principle involves the use of convolutional neural networks (CNN) to process images from the training dataset. The CNN then generates an output that predicts the class to which the image belongs. Given our prior knowledge of the class of each training image, we can verify the accuracy of the obtained result. Based on the credibility of this finding, the CNN weights are adjusted using a technique known as error gradient backpropagation. Throughout the model training phase, the CNN weights are updated iteratively by processing all the images from the training database multiple times. The objective is for the model to accurately classify the data to the best of its ability. Upon completion of weight updates, the model undergoes evaluation by being tested on the validation dataset. The system categorises all these images, which are images that the model has not been previously exposed to, and computes its classification accuracy, known as the model's accuracy.

3.3.3.1 Dataset preparation

When developing a convolutional neural network model, it is essential to train the model using data to enable it to learn and recognise specific features or image classes. For this purpose, a database containing all images is used. A portion of this

database will be utilised for training the CNN model, commonly referred to as the "training or learning dataset." The portion of the database that remains is used for testing the model, commonly referred to as the "validation dataset." It is important to note that all images used must be pre-labelled, indicating that each image must be assigned to a specific class. For instance, when aiming to identify three defect types in steel sheet, this will result in three distinct classes. Each image within the dataset is required to be categorised into one of the three specified classes. This phenomenon is referred to as supervised learning.

3.3.3.2 Overfitting

Overfitting is a typical issue in algorithms using deep learning (**Figure III.16**), in which a model attempts to fit all of the training data and ends up retaining the data patterns as well as the noise/random fluctuations. Such models do not succeed to generalize or perform effectively in unforeseen data circumstances, which defeats the model's objective.

Overfitting happens when training data is not cleansed and contains "garbage" values, and the model collects noise, fails to generalize learning, has a large variance, not enough data for training size, and several neural network layers in its design. Deep neural networks, which take a long time to train, frequently result in overfitting the training set. Inadequate hyperparameter setting throughout the training phase might lead to excessive observation of the training set and feature memorization.

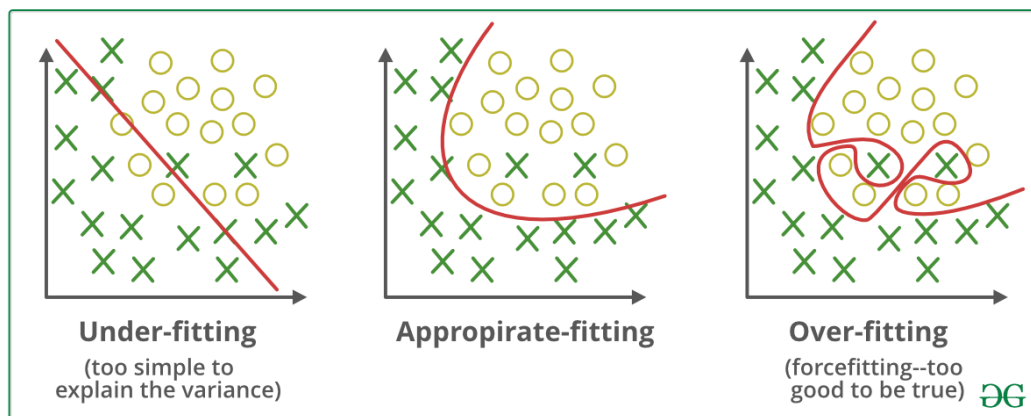


Figure 3-16 Overfitting, appropriate fitting and underfitting graphs.

➤ Reducing overfitting

In order to prevent overfitting, many techniques can be applied:

- Train with more data

Using additional data to train algorithms can increase their capacity to recognize signals. To prevent overfitting, the data must be clean and meaningful. However, this strategy may not always be effective, as noisier data may not increase the model's performance. In the case of modeling height vs. Age in children, sample additional schools to enhance the model. As a result, it is critical to verify that the data is accurate and useful.

- Data augmentation

Data augmentation is a cheaper and safer alternative to training using extra data. It entails generating slightly different sample data each time the algorithm processes it. A bigger dataset decreases overfitting, and if data is very limited, it can be artificially expanded. For example, in image classification training, numerous image transformations, including flipping, rescaling, rotating, and shifting, can be used to enhance the dataset size (**Figure III.17**).

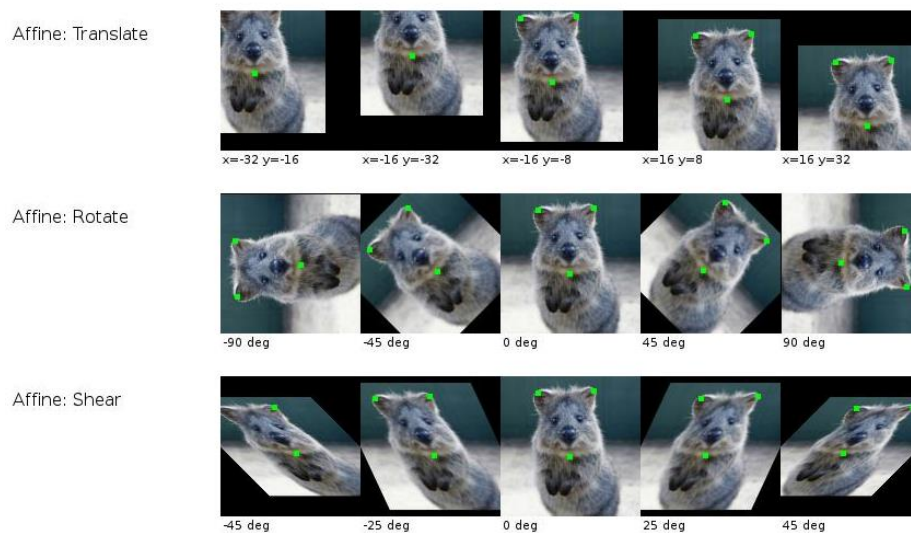


Figure 3-17 Different data augmentation techniques

- Early stopping

Early stopping is a strategy for pausing a model's training before it memorizes noise as well as random fluctuations in the input. It is critical to prevent overfitting by discontinuing training too early. Iterative learning algorithms assess the efficiency of each iteration, and subsequent iterations enhance the algorithm until it overfits the training dataset (**Figure III.18**). Early stopping is commonly utilized in deep learning, although other approaches like as regularization are used in conventional machine learning.

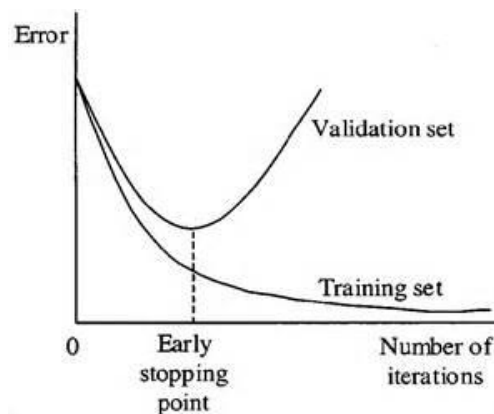


Figure 3-18 Early stopping before the model overfits.

- Adding dropout layers

Larger weights in the neural network indicate a more complicated network. To avoid overfitting, connections or nodes in the network might be dropped probabilistically. To minimize model complexity, certain layer outputs are arbitrarily ignored or "dropped out" during regularization. By adding dropout, a type of regularization, to one of our layers, we may disregard a subset of our network's units with a fixed probability.

Using dropout (as in **Figure III.19**), we may limit interdependent learning across units, which may have contributed to overfitting. However, with dropout, our model would require additional epochs to converge.

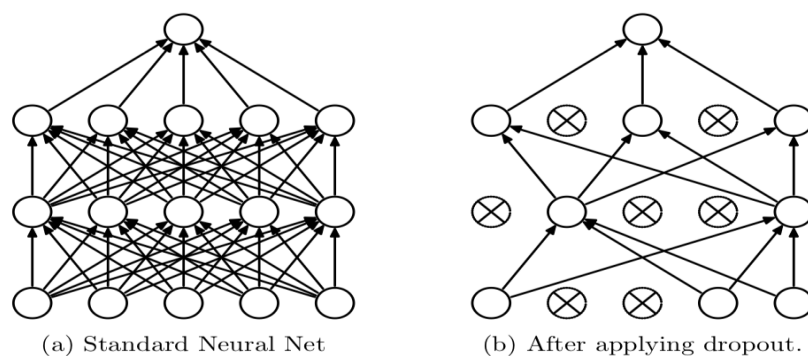


Figure 3-19 Dropping out random nodes.

3.3.3.3 Underfitting

Underfitting is a prevalent problem in the field of machine learning characterised by a model's inability to establish a proper relationship between input and target variables, resulting in increased errors on both training and unseen data instances. The phenomenon described is distinct from overfitting, which occurs when a model demonstrates high performance on the training data but struggles to apply its learning to new, unseen testing data. Underfitting occurs when the model is overly simplistic and fails to capture the underlying relationship between input and output variables. The phenomenon occurs when the training error rate is elevated, indicating that the model is struggling to learn from the provided training data. Some common signs of underfitting are training data that isn't clean and has noise or outliers, high bias because the correlation between input examples and target values wasn't captured, the model's assumption of being too simple, and wrong hyperparameter tuning because features weren't observed enough.

➤ Avoiding underfitting

Reducing underfitting in Machine Learning can be handled using:

- **Reduce regularization:** Regularization inhibits learning more complicated models, which lowers the risk of overfitting. Noise and outliers can be reduced using methods like L1 regularization, Lasso regularization, and dropout.
- **Increase the period of training:** Early termination of training might result in underfitting. Overtraining and underfitting are balanced in an optimal halting strategy.

- **Feature selection:** Adding additional characteristics improves the model's predictability. For example, in deep neural networks, adding additional hidden layers or dependent variables might make the model more complicated.
- **Remove noise from data:** Cleanup and preprocessing approaches can eliminate noise from the training set, hence reducing underfitting.

3.3.3.4 Performance evaluation metrics

Evaluation metrics are critical in machine learning because they allow us to assess a model's quality and predictability. Machine learning algorithms are evaluated using many measures, including classification and regression metrics. In our thesis we focus on classification metrics. Evaluation metrics are used to analyze model performance, monitor production, and regulate the model to meet customer demands. The objective is to develop and pick a model that is highly accurate on out-of-sample data. It is critical to employ several assessment measures to verify that a model operates correctly and effectively, as one metric may not be appropriate for another. To assess the performance of such an approach, the following measures are used.

3.3.3.5 Confusion Matrix

A confusion matrix, also referred to as an error matrix, is a tabular representation that displays the count of correct and incorrect predictions produced by the model compared to the true classifications in the test dataset, along with the different types of errors encountered. The matrix illustrates the performance of a classification model on test data with known true values. The matrix is of size $n \times n$, with n denoting the total number of classes.

The matrix was constructed subsequent to generating predictions regarding the test data (**Figure III.20**).

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Figure 3-20 Confusion Matrix.

Columns show the number of actual classification in the test data, while rows indicate the model's predicted classifications.

When making classification predictions, there are four possible outcomes:

- **True Positives (TP):** The number of results that are actually positive and expected to be positive.
- **True Negatives (TN):** are the number of results that are both negative and shown to be negative.
- **False Positives (FP):** are the number of outcomes that are truly negative but forecasted as positive. These faults are also known as Type 1 Errors.
- **False Negatives (FN):** are the number of positive outcomes that were projected to be negative. These faults are also known as Type 2 Errors.

Positive and Negative relate to the prediction by itself. Both true and false refer to the accuracy of the forecast. The Confusion Matrix provides four classification metrics :

A. Accuracy

The most basic criteria for evaluating models is accuracy. It is defined as the ratio of correct predictions to total predictions given a dataset. Accuracy is beneficial when the target category is well balanced, but it is not suitable for unbalanced classes. For instance, When the number of samples in each class is equal, the accuracy of the model is at its highest. For example, if there are 90% A samples and 10% B samples, the model forecasts all A samples with 90% accuracy. However, if the model is evaluated with 60% samples from class A plus 40% from class B, its accuracy drops to

60%. Because minor class samples might be misclassified, this causes a False Positive sense of high accuracy (eq 3.16).

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total numbers of predictions made}} \quad (3.16)$$

Accuracy offers us an overall view of how much we can trust our model's predictions. This measure does not distinguish between mistaken classifications or kinds. That is why it is not good enough for imbalanced datasets. It can also be computed as either positive or negative for binary classification (eq 3.17):

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.17)$$

It does not provide much information on the distribution between false positives and false negatives.

B. Precision or Positive Predictive Value (PPV)

It is the proportion of True Positives to all positive predictions made by the model. It is practical with skewed as well as imbalanced datasets. The greater the number of false positives the model predicts, the worse its accuracy (eq 3.18).

$$Precision = \frac{TP}{TP+FP} \quad (3.18)$$

C. Recall or Sensitivity or True Positive Rate (TPR)

It represents the ratio of true positives to all positives in your dataset. It evaluates the model's ability to identify positive samples. The recall decreases as the number of false negatives predicted by the model increases (eq 3.19).

$$Recall = TPR = \frac{TP}{TP+FN} \quad (3.19)$$

D. F1-score or F-measure

It is one single measure that includes both Precision and Recall. Our model's performance improves as the F1 score increases. The range of values for F1-score is (0,1).

The F1 score is a weighted average of accuracy and recall. The machine learning model is going to receive a high F-score if both accuracy and recall are good. This metric mainly favors classifiers with comparable accuracy and recall (eq 3.20).

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3.20)$$

3.3.3.5.1 Area Under Curve (AUC)

AUC, which stands for Area Under the Curve, represents the area under the ROC curve. The general effectiveness of the binary classification algorithm is assessed. Both true positive rate (TPR) and false positive rate (FPR) have values between 0 and 1, resulting in the area under the curve (AUC) also falling within this range. A higher AUC value signifies superior model performance, as shown in **Figure III.21**. The primary objective is to expand this area in order to achieve the highest True Positive Rate (TPR) and the lowest False Positive Rate (FPR) at the designated threshold. The Area Under the Curve (AUC) metric is used to measure the model's ability to rank a randomly selected positive case higher than a randomly selected negative case based on their projected probabilities. This statement describes the ability of our model to differentiate between the two classes present in our target variable.

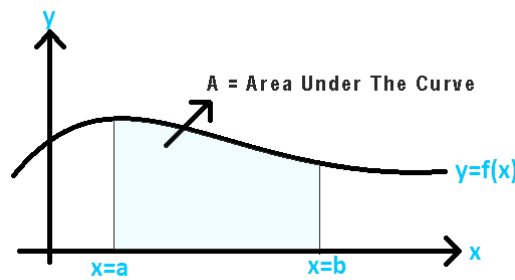


Figure 3-21 Area Under the Curve AUC

The area under the curve can be computed in three easy steps. To determine the area of a curve, one has to know its equation, limits, and axis. The second step is to

calculate the curve's integration (antiderivative). Finally, to calculate the area under the curve, we must apply the upper and lower limits to the integral result and subtract the difference (eq 3.21).

$$Area = \int_a^b y \cdot dx = \int_a^b f(x) \cdot dx = [g(x)]_a^b = g(b) - g(a) \quad (3.21)$$

3.3.3.5.2 Receiver Operating Characteristics (ROC) Curve

ROC refers to Receiver Operating Characteristics, where the ROC curve graphically represents the binary classification model's performance. It displays the true positive rate (TPR) against the false positive rate (FPR) at various classification levels.

3.3.3.5.3 AUC-ROC curve

The AUC-ROC curve, meaning Area Under the Receiver Operating Characteristic curve (**Figure III.22**), is a graphic representation of a binary classification model's performance over different categorization criteria. It is often used in machine learning to evaluate a model's capacity to differentiate between two classes, generally the positive class (the existence of a fault) and the negative class (the absence of a defect).

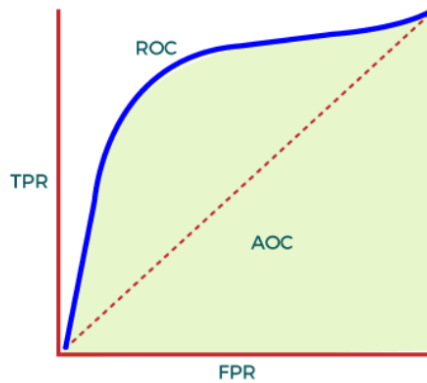


Figure 3-22 AUC ROC curve

At the end, ROC is simply the plot of TPR vs FPR across all potential thresholds, whereas AUC is the total area under this ROC curve.

3.3.3.6 Different Types of CNN Architectures

The CNN architecture is the most widely used deep learning framework. CNNs have demonstrated tremendous performance in image recognition, providing unprecedented precision and scalability. However, not all CNNs are made equal, and knowing the many types of CNN architectures is critical to revealing their full potential. In this section, we will see the first two CNNs appeared in the literature, further architectures are given in Chapter I.

a. Neocognitron

Neocognitron was the first design of its sort [75], pioneering feature extraction, pooling layers, and convolution into neural networks. The network's structure was inspired by vertebrates' visual nervous system and comprised of S-cells (simple cells or lower order hypercomplex cells) and C-cells (complex cells or higher order hypercomplex cells), with the procedure of feature extraction and positional shift tolerance repeated. Local features were merged with global ones. Neocognitron was used to recognize Japanese handwritten characters and patterns, paving the path towards convolutional neural networks (**Figure III.23**). It was used to recognize handwritten characters and do other pattern recognition tasks.

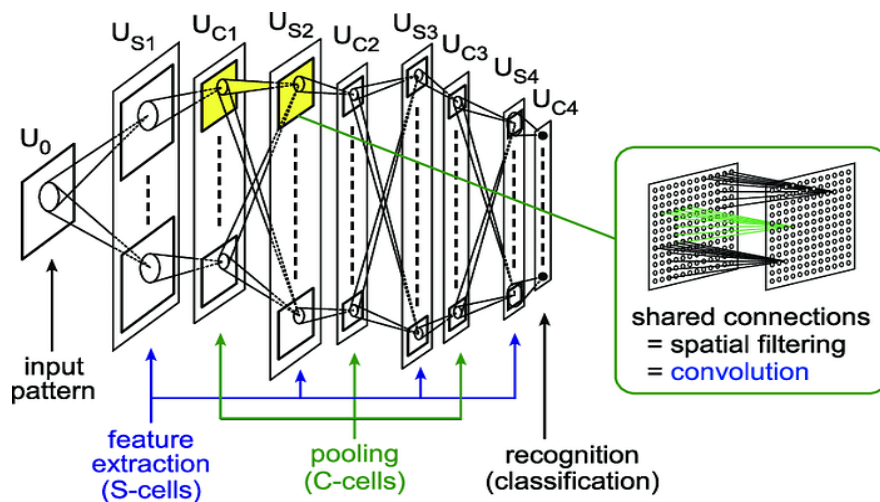


Figure 3-23 The architecture of Neocognitron network.

b. LeNet.

In 1986, the Neocognitron structure was adapted to incorporate weight sharing among neurons in the same layer, utilising formal learning principles through gradient optimisation, specifically gradient retropropagation (BP) [76]. Thus, the structural underpinnings of convolutional networks were established through the initial formalisation of a learning algorithm. In 1998, LeCun and colleagues introduced LeNet-5, a convolutional network structure [77]. **Figure III.24** depicts the standard design of a LeNet CNN-5.

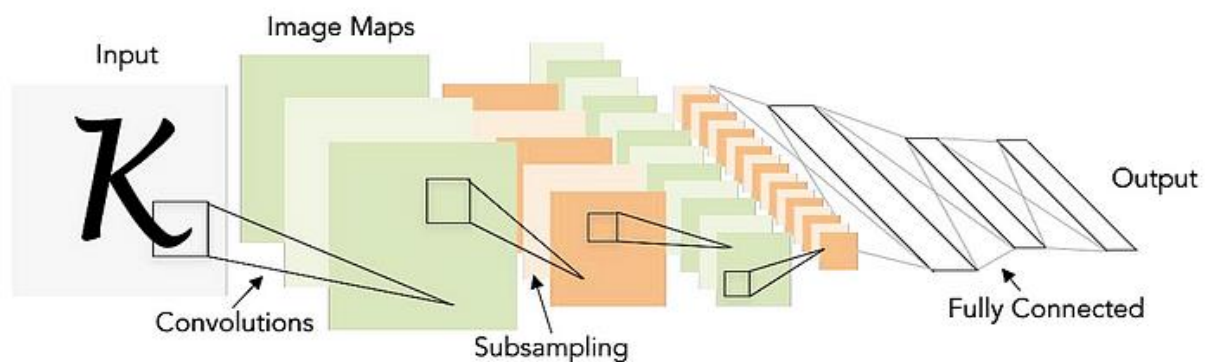


Figure 3-24 The architecture of LeNet-5.

Yann LeCun et al. As previously mentioned, designed the LeNet, which gave rise to the term "convolutional neural networks" [78]. It was primarily developed between 1989 to 1998 for the recognition of handwritten digits' challenge. The general architecture was [CONV-POOL-CONV-POOL-FC-FC]. It employed 5x5 filters of convolution with a stride of one. The pooling of (subsampling) layers measured 2x2 and had a stride of 2. It contains around 60 K parameters.

c. AlexNet, the Deep Learning Architecture that pioneered CNN

Alex et al.[79] collaborated to develop AlexNet. AlexNet's architecture closely resembled that of LeNet, featuring increased depth and size achieved by stacking Convolutional Layers on top of each other. AlexNet, the initial large-scale Convolutional Neural Network (CNN), achieved victory in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012.

The AlexNet architecture was designed for use with large-scale image datasets, and it achieved state-of-the-art results upon its release. AlexNet consists of five convolutional layers, max-pooling layers, three fully connected layers, and two dropout layers, as shown in **Figure III.25**. All layers utilise the Relu activation function. The activation function utilised in the output layer is Softmax. The total number of parameters in this model is approximately 60 million.

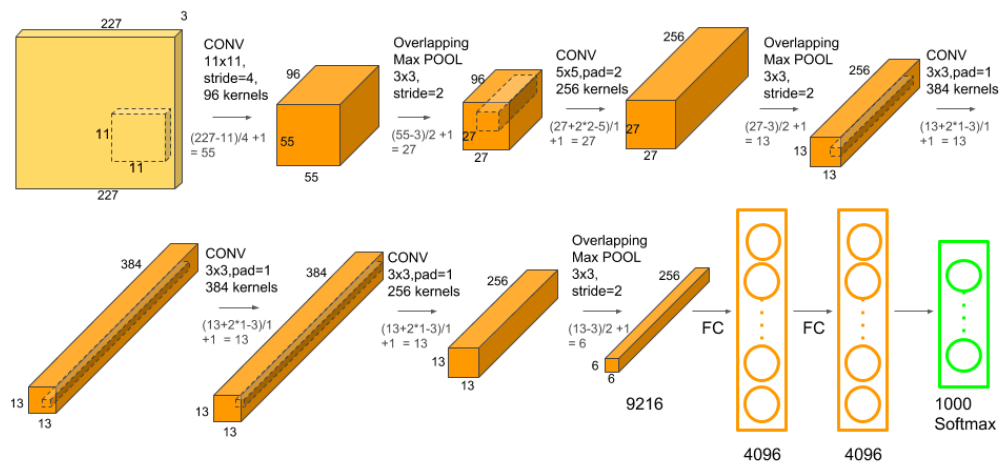


Figure 3-25 AlexNet architecture [79].

Other famous architectures are introduced in the Literature Review Chapter, especially those used in steel inspection field.

d. AutoML

Model selection entails evaluating aspects such as dataset size, kind, and hardware constraints. Model assessment requires tuning hyperparameters such as learning rate and optimizer, as well as altering model architectural factors like layer number and operations.

In machine learning, these parameters are usually established by human configuration. Transfer learning (TL) can be used to decrease the time needed to alter parameters if training results are not optimal. Hyperparameter tuning may be

automated by artificial intelligence techniques, whereas AutoML leverages artificial intelligence to automate machine learning processes.

Figure III.26 shows how the AutoML leverages AI to help automate ML process.

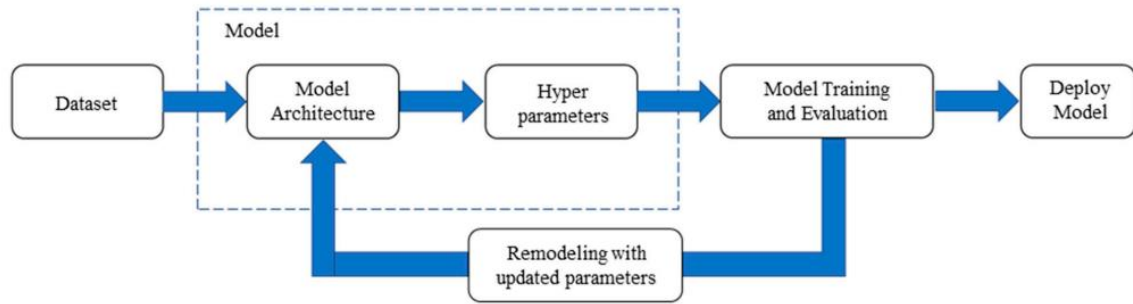


Figure 3-26 ML-related training steps.

e. Network Architecture Search NASNet

NASNet, or Neural Search Architecture (NAS) Network, is a recent machine learning model. Its basic ideas differ from common models such as GoogleNet and are expected to result in a significant breakthrough in AI shortly.

Neural Architecture Search (NAS) automates network architecture design. It is a technique that looks for the optimal method to execute a specific task. The standard setup of NAS has three components (**Figure III.27**).

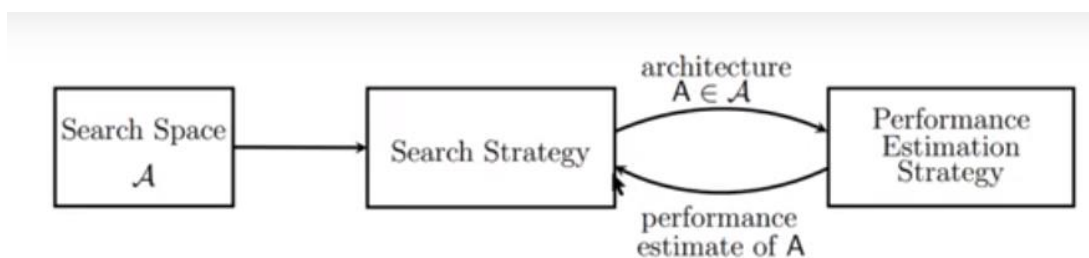


Figure 3-27 Main NAS setups

✓ Search space

The search space is the set of all feasible designs for a particular problem, which includes neural networks as functions that change input variables. There are two types

of search spaces (**Figure III.28**): global search spaces, which allow for a wide range of operations, and cell-based search spaces, which focus on identifying individual cells that may be joined to form the full neural network. Cell-based search spaces produce smaller, more efficient structures that may be combined into bigger ones.

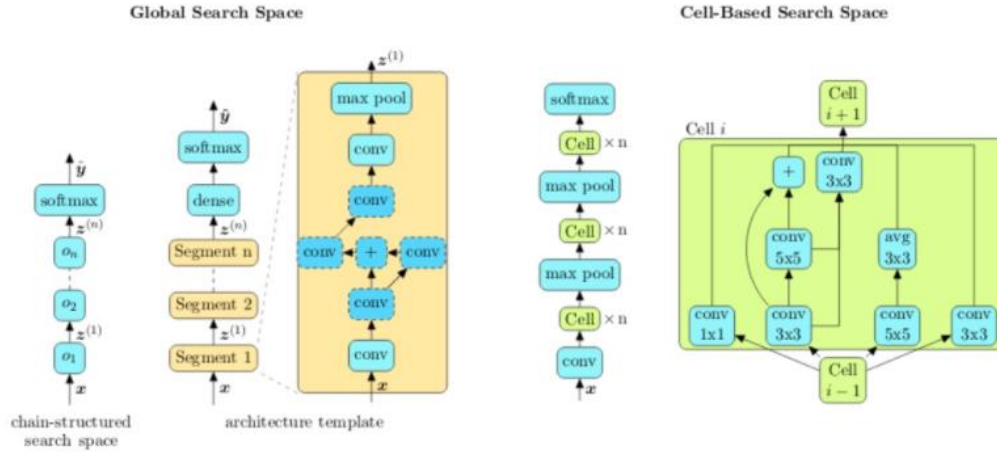


Figure 3-28 Global search space (left) and cell-based search space.

✓ Search strategy

NAS employs a variety of search algorithms to determine the optimum design for predicting performance while avoiding testing ineffective ones. Grid and random search, gradient-based search, evolutionary algorithms, and reinforcement learning are some of the available methodologies. Grid and random search are easy and effective for small search spaces, but they fail for larger search spaces. Gradient-based search is simpler to define, but it requires converting a discrete architectural space into a continuous space. Evolutionary algorithms use biological mechanisms to create optimal designs under specified limitations. Reinforcement learning (RL) is used in NAS to propose and evaluate child model designs.

The "REINFORCE" method is used to train the controller network (**Figure III.29**), resulting in a variable-length sequence of tokens for network architectural configuration.

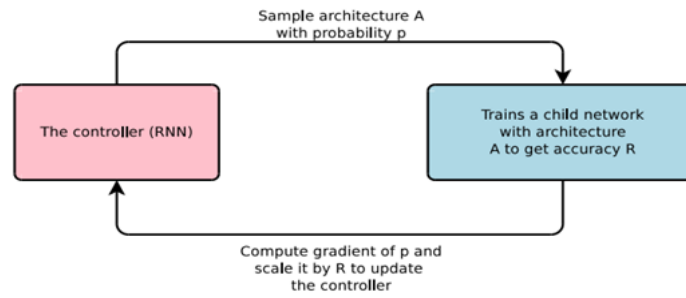


Figure 3-29 Reinforce method to train controller network.

✓ Performance estimation strategy

Performance estimation is an approach for evaluating a neural network's performance based on its architecture rather than developing and training it. It returns a number that represents the model architecture's performance or accuracy when applied to a test dataset. This can aid in passing results to the next iteration or improvising from the start.

➤ Transfer Learning

Transfer Learning is a machine learning technique that involves reusing a pre-trained model as the basis for a new model designed for a different problem. Reusing a model trained on one job for a second, similar task is a form of optimisation that enables quicker modelling of the second task. Utilising transfer learning for a novel task has the potential to yield significantly higher performance compared to training with a limited dataset.

Transfer learning is a common practice in training models for image or natural language processing tasks, making it rare to start training a model from scratch. Researchers and data scientists typically opt to start with a pre-trained model that is already capable of identifying objects and has acquired general knowledge such as recognising edges and shapes in images.

ImageNet, AlexNet, and Inception exemplify models that utilise transfer learning. **Figure III.30** shows the difference between traditional ML and Transfer Learning.

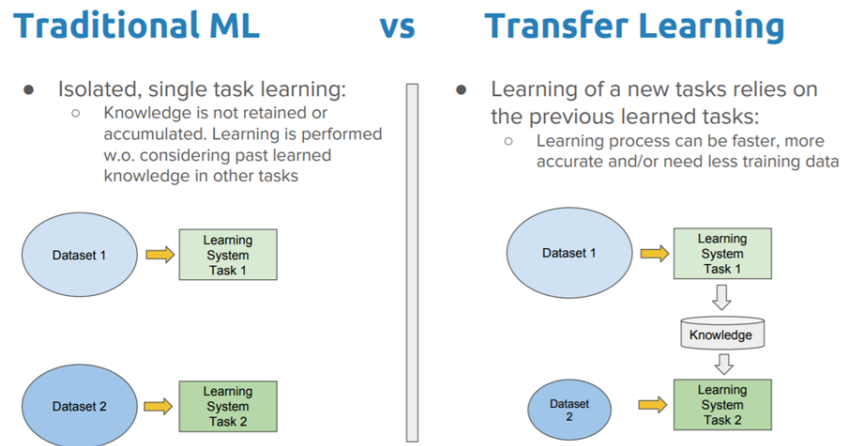


Figure 3-30 Difference between Traditional ML and Transfer Learning.

Transfer learning models use information obtained by addressing one problem with a related problem. Pre-trained models may be utilized as starting points for training, offering a dependable architecture while saving time and resources. This strategy is appropriate for scenarios with little data or for obtaining speedier results. The process involves selecting a source model that is similar to the target domain, modifying it prior to transferring knowledge, and then training the source model to adapt to the target model. Common strategies include fine-tuning the higher-level layers and freezing the lower levels. Overfitting can arise in jobs with limited data if the source and target models are too similar. To avoid this, tune the learning rate, freeze some layers, or add linear classifiers when training the target model, which can help reduce overfitting.

3.4 Conclusion

In this chapter, we introduced the most important aspects of Convolution Neural Network, including its definition, importance, subfields, and application in the real world. Here are the main conclusions for this chapter.

In our thesis, we are more interested in CNNs, as they are the backbone for image classification tasks. A Convolution Neural Network (CNN) architecture is made up of layers, including input layers, layers for convolution, activation layers, and pooling layers. The convolutional layer collects important features from the input picture, while the activation layer introduces nonlinearity. The pooling layer decreases computation size and memory. The fully connected layer provides the final prediction by learning optimum filters using backpropagation and gradient descent.

One of the major issues faced in the DL algorithm is *overfitting* and *underfitting*; they are the leading reasons for low machine learning performance. Overfitting happens when the model performs well in the training process but has low accuracy in the test dataset. Underfitting happens when the model fails to extract features from the training data. To get a good fit, training should end at an ideal moment, minimizing under-observing features or acquiring extraneous information and noise in the training set.

We also saw that model evaluation is crucial in Machine Learning, and performance metrics like accuracy, precision, recall, and auc-roc are used to evaluate a model's effectiveness. The AUC-ROC curve is a key metric in machine learning, providing insight into the model's performance in classification tasks.

Also, we have a general idea about the different types of CNN used today in image classification tasks. Convolutional neural networks have progressed from LeNet to MobileNets, which are more efficient. Each design, from AlexNet to ResNet and beyond, has added new features that have expanded our knowledge and use of CNNs. They have pushed the limits of what is feasible in picture identification and classification tasks, with extensive applications in a variety of fields. The future of CNNs presents both great prospects and difficulties, promising to further change our digital ecosystem.

The CNN that is going to be used in this thesis is NASnet-Mobile. We will dive into its explanation in the experimentation chapter. Yet, we got a general understanding of NAS architecture. NAS has created deep neural networks and architectures for image processing and computer vision that use RL and evolutionary algorithms. However, further research is needed to make NAS more general because of its high cost for actual applications. Search efficiency is an issue, especially for RL-based NAS systems, which take thousands of GPU-days of searching and training to get ideal computer vision results.

Finally, we defined transfer learning at the end of this chapter. Transfer Learning is a method of machine learning in which a pre-trained model serves as the foundation for a new assignment, allowing for faster development and greater performance. In image and natural language processing applications, researchers often favor pre-trained models with object categorization and generic characteristics.

In the next chapter, we will talk about steel as a dominant metal, its production, its quality inspection techniques, different defects that can occur on steel surface during production line, manual and automatic quality inspection, advantages and disadvantages of each method.

4 Chapter IV

STEEL MAKING PROCESS AND DEFECT INSPECTION IN STEEL SHEET

4.1 Introduction

Steel has had a profound influence on our world unlike any other metal. A new high-performance steel allows a jet pilot to reach greater altitudes or a surgeon to conduct intricate procedures. The shuttle's solid rocket booster casings allow astronauts to venture into uncharted territories, similar to a child enjoying a roller coaster ride. Every steel piece we manufacture is meticulously crafted to meet precise specifications. This industry measures product quantities in millions of tons and quality in millionths of an inch.

This chapter provides a comprehensive explanation of steel, insights into its production in the industry, and various types of steel surfaces. Next, we observe the primary imperfections that may arise on steel sheets during the steelmaking process, such as scratches, pitting, and so on. Discuss various inspection techniques, steel defect recognition algorithms, and the primary challenges encountered in the inspection process.

4.2 Definition of Steel

Steel is a special alloy composed of iron, carbon, and other elements. There are several chemically distinct varieties of steel. Most steels have more than 95% iron, with trace elements of carbon, manganese, silicon, phosphorus, and sulfur. Steel is made by refining (removing impurities from iron) and alloying it with additional metals and minerals. Steel can be hard or soft, rigid or flexible, rusty or stainless.

Steel is currently the world's most significant engineering and building material. It is utilized in the manufacture of automobiles, household appliances, ships and trains, bridges, tools and medical equipment, weaponry, skyscrapers, and many other products. Steel is an exact alloy composed of iron, carbon, and other elements.

There are several chemically distinct varieties of steel. Most steels have more than 95% iron, with trace elements of carbon, manganese, silicon, phosphorus, and sulfur. Steel is made by refining (removing impurities from iron) and alloying it with additional metals and minerals. Steel can be hard or soft, rigid or flexible, rusty or stainless. Steel is currently the world's most significant engineering and building material. It is utilized in the manufacture of automobiles, household appliances, ships and trains, bridges, tools and medical equipment, weaponry, skyscrapers, and many other products.

4.3 Importance Of Steel Surface Production

Steel is an important and versatile material that has had a huge impact on our lives, from automobiles to homes. It is utilized in power plants, natural gas pipes, machine tools, and military weaponry. Steel's flexibility is demonstrated by its numerous applications, including hot and cold formability, machinability, hardness, wear resistance, corrosion resistance, and heat resistance. Steel has lower production costs than other materials, with the energy required for iron extraction being around 25% of that required for aluminum. It is also recyclable, making it an ecologically responsible option. Steel output is 20 times more than nonferrous metals combined. The steel industry has created new technologies to increase the material's strength and adaptability. There are around 3500 kinds of steel, including 1500 considered high-grade steels.

4.4 Economic and Industrial Impact of Steel Production

According to a recent report, the American Iron and Steel Institute (AISI) is a vital element of the US economy, accounting for more than \$520 billion in economic production and about two million jobs in 2017. These workers received more than \$130 billion in pay and benefits. Overall, the business collected \$56 billion in federal, state, and municipal taxes.

In the industrial side, steel is an important metal for manufacturing and consumption, with output surpassing 1,582 million metric tons in recent years. There are about 3,500 grades of steel, with flat steel goods accounting for over half of commerce. An integrated iron and steel factory uses blast furnaces to manufacture liquid iron from iron ore, coke, sinter, and flux. This is transformed into liquid steel by primary and secondary processes, which is subsequently cast into slabs and billets. Slabs have a rectangular cross-section with dimensions of 1,600 mm broad, 250 mm thick, and 12,000 mm long, whereas billets are square and measure 150 x 150 mm and 12,000 mm long. Slabs are subsequently rolled into hot and cold strips, and billets are formed into structural forms.

Figure IV.1 shows a clear flowchart of the steel-making operations.

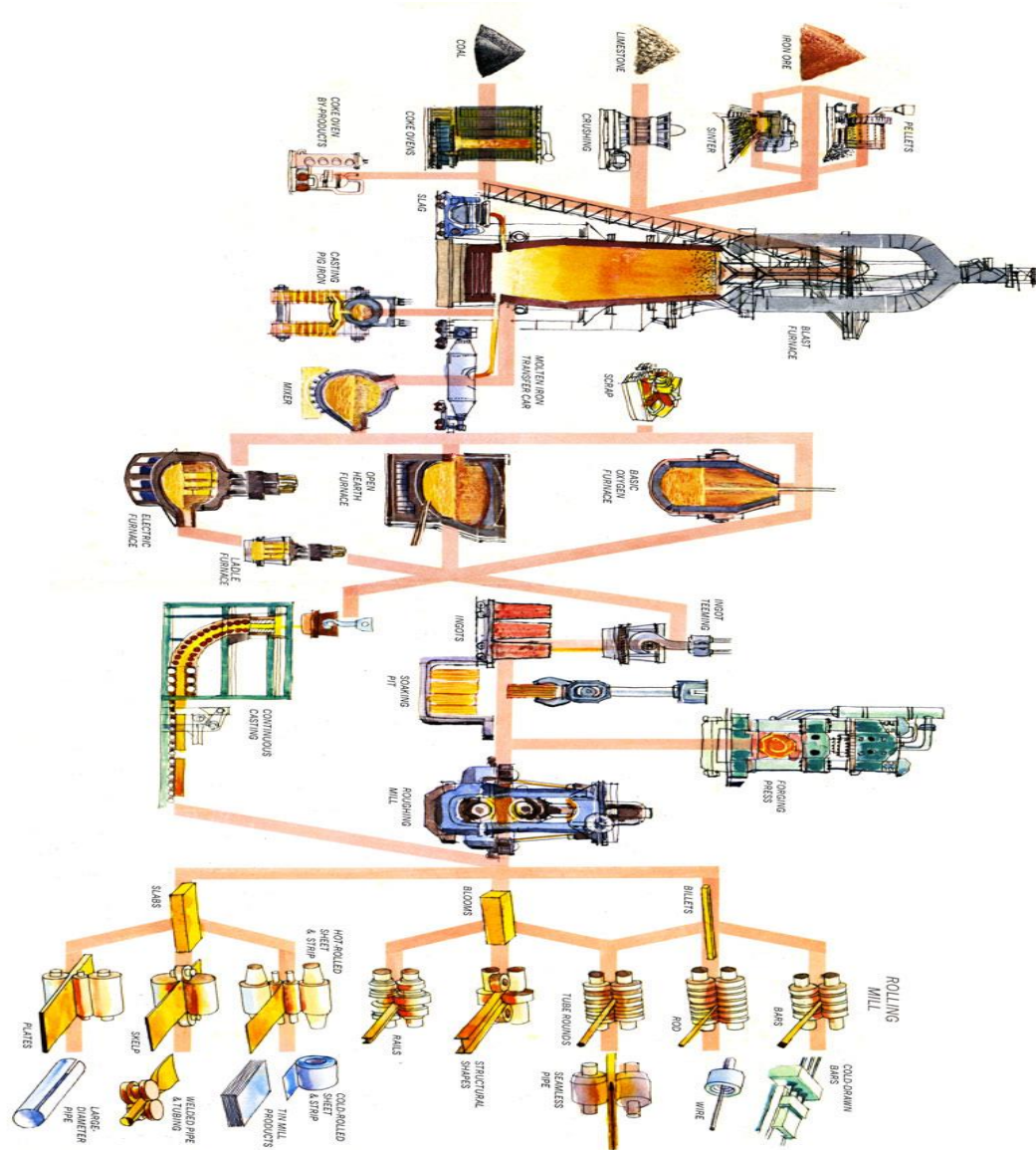


Figure 4-1 Steel making process.

4.5 Steel making process

Steel is commonly manufactured through two primary methods: blast furnaces and electric arc furnaces. The blast furnace, which marked the beginning of the steel production process from iron oxides, was first implemented in the 14th century with a daily output of one tonne. Pig iron is typically manufactured in the blast furnace through the combination of coke, iron ore, and limestone. Coal is frequently used in the coke-making procedure, where it is crushed and ground into a powder before being heated to 1800°F without the presence of oxygen. The coal is subsequently cooled and filtered into particles with dimensions ranging from one to four inches.

Coke is a porous, dense black rock composed of concentrated carbon and possessing a high energy content. The process involves providing the necessary permeability, heat, and gases to facilitate the reduction and melting of iron ore, pellets, and sinter. Currently, there is a growing trend of utilising natural gas in blast furnaces as a substitute for coke to reduce carbon emissions.

During the late 19th century, electric arc furnaces, also known as EAFs, were invented. The utilisation of Electric Arc Furnaces (EAFs) has experienced significant growth and currently represents more than 70 percent of steel production in the United States. The Electric Arc Furnace (EAF) is distinguished from the blast furnace by its utilisation of an electrical current to transform scrap steel, direct reduction iron, and/or pig iron into molten steel.

4.5.1 Steel Processing And Finishing

The following are the main processing and finishing phases for steel:
Casting : Two casting methods are utilised in the industry: batchwise casting, which involves ingots, and continuous casting, which is used for slabs, blooms, and billets. Continuous casting has been found to be more energy-efficient.

Rolling: Rolling involves hot roll mills that reduce the thickness of the metal billet, which is subsequently cold rolled into sheets, bars, and rods. The cold drawing of a steel rod results in the production of wire.

Pickling: It is a process used to eliminate oxides and scale from the surface of steel prior to cold rolling or drawing. The process is frequently carried out using sulfuric or hydrochloric acid.

Coating: Metal coating procedures encompass zinc (galvanising), zinc/aluminium alloy, and painting. Steel is often passivated with zinc, aluminium, or alloy coatings to prevent surface corrosion during storage. Passivation is commonly carried out using a chromate solution, which contains hexavalent chromium, a recognised carcinogen. Various alternative coating processes have been developed and are utilised to varying extents across different product categories.



Figure 4-2 Rolled Metal Products : Steel Profiles and Tubes

4.5.2 Different steel surface categories

Steel surfaces can be classified into flat or long (see **Figure IV.2, Figure IV.3 and Figure IV.4**).

Flat product surfaces can be categorized as follows:

- Slabs and billets are manufactured by continuous casting from liquid steel and have identical surface and interior characteristics. The surface is scale-covered and more gritty.
- Plates are made by reheating a slab at 1,250°C and rolling it. The surface is oxidized and even compared to the slab.
- Cold strips are made by rolling hot strips in a cold rolling machine after pickling to remove the oxide layer and polish the surface. Cold strips have a smooth surface due to the strong rolling pressures utilized during the cold deformation process, preventing oxidation.
- Coated strip (galvanized, tinned) or polished stainless strip surfaces are extremely reflective.

Long product surfaces can be categorized as follows:

- Rods/bars: Hot rolling produces rods and bars with oxidized surfaces from billets. The surface is not flat, resulting in non-uniform picture intensity as the angle of reflection varies towards the perimeter

- Billet/bloom is used to make long items such as angles, channels, structural steel, and rails. These complicated cross-sections demand specialized lighting and camera setups.

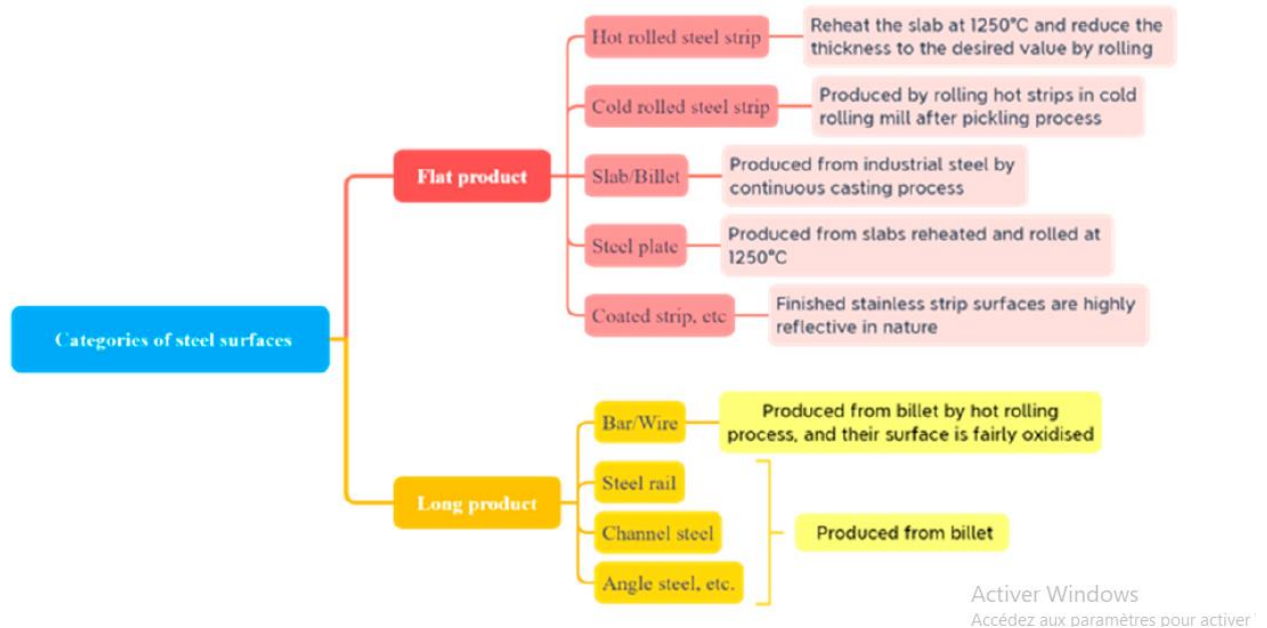


Figure 4-3 Different types of steel products [31].

The categorization provided above does not include products made with materials other than steel or those produced outside of the steel mill, such as the remelting of steel ingots. The category comprises items that have undergone galvanization or other treatments to prevent surface corrosion. This category excludes painted objects, except for steel that has been painted to prevent surface corrosion during storage and transportation.

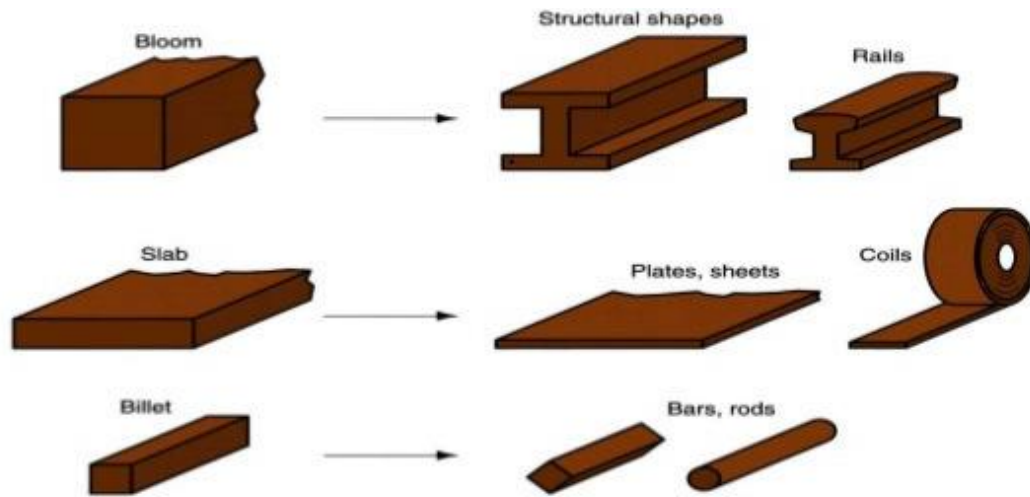


Figure 4-4 Some final steel products made from rolling.

4.5.3 Quality inspection in steel surfaces

Surface quality of steel products, specifically cold-rolled steel, has risen in importance since the 1980s as a result of automotive industry needs. The quality of hot strip surface and structural items such as rods and bars has also increased. Traditionally, the surface quality of flat steel goods is determined manually by cutting around 30 meters of random coils in a batch and evaluating it by an expert. However, because to high line speeds, tiredness, and other negative variables, manual inspection is insufficient to ensure defect-free surfaces. In the beginning of the 1980s, nine steel and three aluminum businesses in the United States initiated a research effort on vision-based steel inspection of surfaces in partnership with two commercial enterprises.

Prototype system was created and tested in many steel factories in 1987, and systematic study into the surface inspection of steel items began in the second half of the 1980s. Many well-known businesses now make vision-based Automated Surface Inspection Systems (ASIS), and since 2006, a partnership of manufacturers and others has convened an annual International Surface Inspection Summit (ISIS). Real-time assessment of steel surfaces presents various obstacles, including hazardous areas, high operating speeds, and differences in surface flaws across steel products.

Flat steel products require two separate sets of inspection systems, one for the top and one for the bottom surface, each with three to four cameras to cover the whole

width of the strip. Multiple cameras are required on the periphery of long steel items to ensure full coverage. Gathering photos and analyzing them in real time is a tough undertaking since the variety of surface flaws in different steel products, as well as differences in manufacturing processes, make ensuring defect-free surfaces challenging.

4.5.3.1 Key Components of the Automatic Surface Inspection System's Hardware Setup

Figure IV.5 depicts the fundamental hardware framework for ASIS. It comprises several light sources, one or more cameras (bright fields or both bright as well as dark field), a high-speed image processor, a server, and an operator interface.

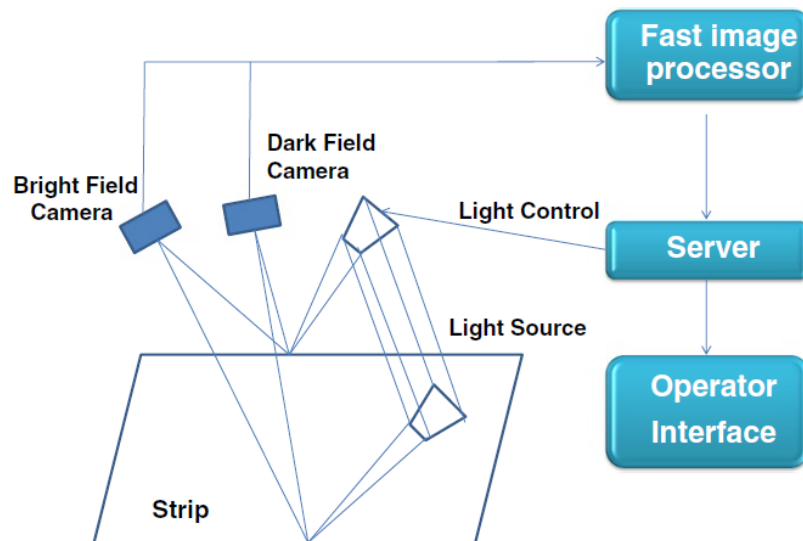


Figure 4-5 ASIS basic hardware structure [1].

A. Image Acquisition

Surface image quality is critical for proper analysis, and high-quality lighting lowers the computing load of image processing. There are two types of lighting approaches for metallic surfaces: intensity imaging and range imaging. Range imaging gives height information, which highlights 3D flaws, although it is not comparable with intensity imaging. Intensity imaging is divided into two distinct categories: bright field and dark field. Bright field illumination collects the majority of the immediately reflected light, yielding a bright surface and darker fault features. Dark field

illumination needs more powerful lighting, approximately eight times greater than brilliant field lighting.

Nevertheless, not all surface flaws occur in both fields. To solve this, several systems employ two separate sets of cameras that cover both fields of vision. For example, at a Chinese iron and steel facility, 20 charge-coupled device (CCD) area scan cameras were utilized to collect surface pictures of hot-rolled strips in both bright and dark field modes. However, due to maintenance concerns and system complexity, most systems position the cameras between the two areas.

B. Source of Light

When shooting at rapid speeds, the camera's exposure time is limited, therefore good fill light is essential for producing clear shots. A good lighting system can increase the inspection system's efficiency and accuracy. Fluorescent, incandescent, xenon, and light-emitting diodes (LEDs) are common light sources used in machine vision. LEDs are the most commonly employed in steel defect identification because to their extended illumination cycle, low heat, low power consumption, consistent brightness, and range of colors. It can be tuned to different irradiation angles and has a quick reaction time, achieving maximum brightness in 10 microseconds or less. The LED light source is powered by an external trigger, computer-controllable, and has minimal running costs.

The instructions for setting up the LED light source may be obtained from [80]. To lessen brightness from metal samples and eliminate specular reflections on flat steel surfaces, diffusion devices can be incorporated to the lighting system.

The light source should give homogeneous, ripple-free light as far as practicable. Ripple-free lighting requires specialized power supplies [81], but constant intensity is sometimes impossible due to the utilization of several light sources. Commonly utilized light sources include wide-spectrum tungsten, fluorescent tubes, halogen, Xenon, and LED.

C. Camera Resolution

Industrial cameras may be divided into two different kinds: analog and digital. Analog cameras, which produce standard video signals, requires a specialized image capture card to convert them to digital data. They are widely used in TV cameras and surveillance because of their flexibility and inexpensive cost. However, they have lesser resolution, a slower acquisition speed, and are prone to noise interference, resulting in picture quality loss. Digital cameras, on the opposite hand, include an inbuilt A/D conversion circuit that transforms analog picture signals directly into digital data, reducing interference and producing higher-quality photos. They also feature higher resolution, frame rate, smaller size, and lower power consumption needs, making them more suited for quick steel production lines and complicated situations in actual steel mills.

When divided by chip type, industrial cameras are classified into charge coupled device (CCD) and also complementary metal oxide semiconductor (CMOS) cameras. CCD cameras use a small number of output electrodes to convert light into electrical signals, whereas CMOS cameras use individual charge-to-voltage conversions for each picture element. CCD cameras have better picture quality because they can adjust to a wider brightness range, but CMOS cameras have a simpler construction, lower power consumption, and lower cost. CMOS sensors' image quality and processing speed have increased as a result of the introduction of a "active image sensitive unit," which reduces noise. CMOS sensors are likely to be the primary sensor technology for machine vision in the future.

Because of its quick processing and capacity to transform electrons into voltage signals, CMOS cameras are better suited for defect detection systems on steel mill production lines. Their high frame rates of over a thousand frames make them perfect for high-speed manufacturing lines. As a result, picture capturing in steel mill manufacturing lines is best done with CMOS digital cameras.

Many scan cameras typically have a resolution of 1,024 (cross web) \times 1 (down web) or 2,048 \times 1 pixels. Yazdchi et al. [82] reported using a 4,096 \times 1 pixel camera. Manufacturers often employ 1,024/2,048/4,096 \times 1 pixel. Popat et al. [83] reported an area scan of 600 \times 400 pixels. In [84], 4,096 \times 1,000 pixels were utilized for slabs.

D. Image Processing Computer Hardware

A rapid, parallel processing unit that is close to the camera receives images taken by a charge-coupled device (CCD) camera [85]. The real-time operation is facilitated by the parallel processing system, which is capable of analysing extensive visual data to detect and save regions of interest (RoI). The parallel processing system could potentially be integrated within the camera, an FPGA processor, or a general-purpose CPU with dedicated hardware. The components of the system play a critical role in both real-time operation and precise flaw identification and categorization. Subsequently, a server with substantial backup memory is utilised for further processing along with the operator's interface (**Figure IV.6**).

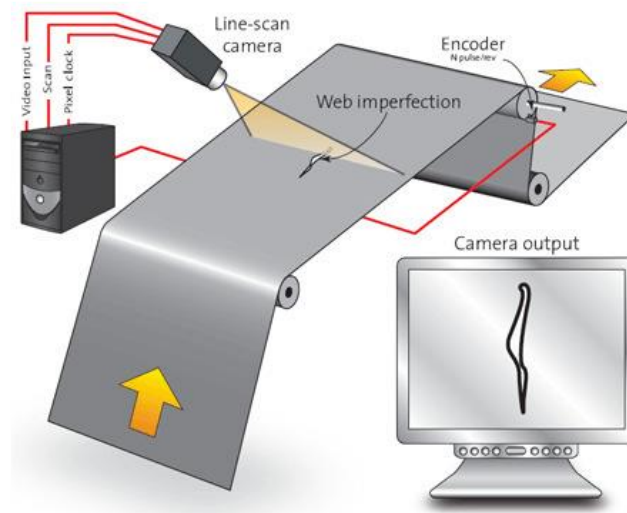


Figure 4-6 Surface inspection system for steel strips based on machine vision.

Image resolution : Several studies [82, 84-87] have reported on image resolution dimensions. The cross-web resolution ranges from 0.17 mm to 1 mm, whereas the stated down-web resolution ranges from 0.25 to 1.25 mm.

4.5.3.2 Different Types of Defects in Industrial Steel Surfaces

Different kinds of steel have different flaws on their surfaces, and there isn't a set standard for how to classify them. This means that there is a chance that defects will be put into the wrong categories, which makes them harder to find. Within this study, the prevalent defect categories identified in numerous sources are consolidated, with the defect catalogue provided by Verlag Stahleisen GmbH [88] serving as the

framework for delineating the defect categories present on steel surfaces. The specific defect categories are detailed in **table IV.1** and **figure IV.7** below.

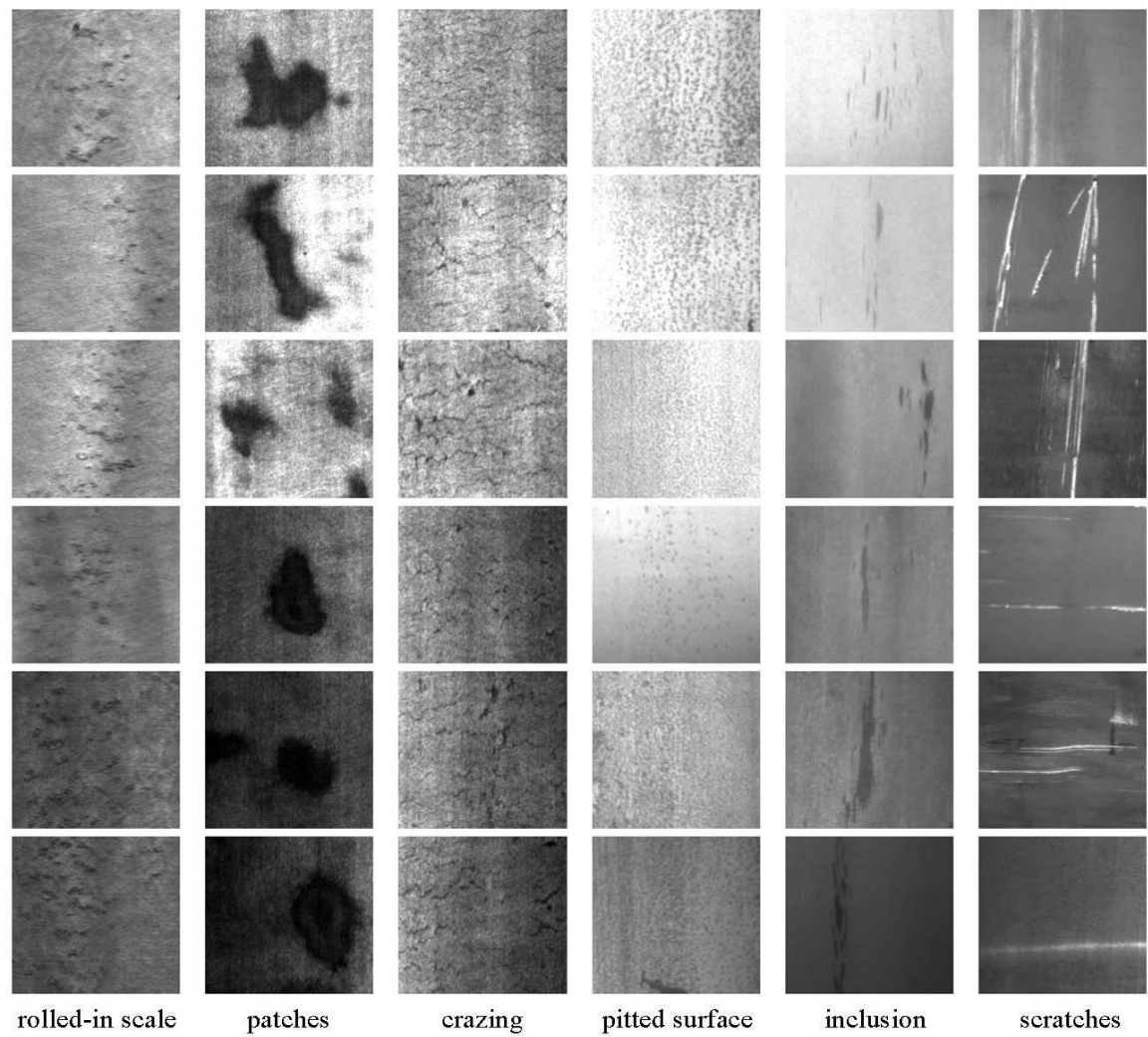


Figure 4-7 Sample photos of six types of typical surface defects.

Table 4-1 Different defect categories according to steel surface types.

Steel Surface Types	Defect Category
Slab	Crack, scratches, scarfing, pitting.
Plate	Crack, seam, scratch.
Billet	Corner crack, scratch, line defect.
Hot rolled steel strip	Hole, rolled in scale, scratch, pits/ scab, crack, edge defect/coil break, sliver, lamination, shell
Colled rolled steel strip	Lamination, hole, roll mark, oil spot, dark, fold, heat buckle, rust, inclusion, scale, sliver, edge, scratch, etc.
Stainless steel	Hole, scratch, scale, roll mark, inclusion, blowhole, shell,
Wire/Bar	Dark line, spot, crack, seam, overfill, lap, scratch, etc.

4.5.3.3 Defect inspection algorithms of steel surface

Steel surface defect identification methods can be categorised into two types: traditional machine learning and deep learning-based algorithms. Three types of traditional machine learning algorithms are categorised as texture-based methods, colour-based methods, and shape-based approaches. Deep learning methods can be categorised as supervised, unsupervised, or weakly supervised. Deep learning methodologies can be categorised based on the specific neural network chosen for tasks such as image categorization or object identification.

Conventional machine learning algorithms, which depend on the creation of human feature extraction rules, tend to be less robust and generalizable, leading to decreased detection accuracy. The system does not perform flaw localization or segmentation, indicating that it is incomplete. Machine learning algorithms used for the classification of steel surface defects can be categorised into three groups: texture

feature-based approaches, shape feature-based methods, and colour feature-based methods.

Deep learning algorithms have gained popularity in defect identification tasks because of their enhanced computational capabilities and exceptional performance. Deep learning methodologies have the ability to extract more intricate and abstract visual features through techniques like convolution and pooling. This allows for enhanced characterization capabilities and facilitates the creation of end-to-end models. Convolutional neural networks extract features from input images through convolutional operations, enabling the capture of different levels of semantic information and enhancing the model's ability to generalise. Convolutional neural networks (CNNs) incorporating pooling layers and sparse connections have been shown to reduce the number of model parameters without compromising computational efficiency or network performance.

4.5.3.4 Challenges and Solutions

Deep learning can tackle many classification problems with high accuracy in less time compared with other conventional machine learning algorithms; however, it faces some challenges during its implementation. The following are some of the limitations that are faced:

A. Insufficient data samples

Deep learning algorithms perform better at higher levels, but they take a lot of data to train, which can lead to overfitting owing to the enormous number of parameters in deep networks. However, the quality of real-world datasets is often poor, and the number of types is incomplete, limiting their effectiveness in fault identification. To address this problem, four successful approaches may be combined: transfer learning methods, unsupervised or poorly supervised methods, data augmentation strategies, and network structure optimization.

Transfer learning is using pre-trained models built on large datasets to solve the object identification problem, which often includes a smaller training dataset. However, because of the huge gap between the original training dataset and the

photos used for defect identification, this may have an impact on the final algorithm's performance. Unsupervised or poorly supervised algorithms can help address the issue of insufficient and incomplete defect samples in the dataset. Data augmentation strategies can be used during the dataset's preparation phase or automatically during training.

Network structure optimization, such as the suggested NASNet-mobile network architecture, can address the issue of insufficient data sampling. The next chapter has an in-depth discussion of the algorithm.

B. Unbalanced Data Samples

Deep learning-based defect detection methods frequently encounter data sample imbalances during training models. This imbalance happens when the dataset has the most defect-free samples and the fewest faulty samples. This imbalance may result in algorithms focusing more on categories with more samples and having less recognition capacity for categories with less data volume. To alleviate this issue, approaches like as data augmentation, data resampling, and GAN networks can be employed to alter the sample distribution of the training set. At the model level, attention to tiny samples may be modified by assigning appropriate weights to each sample in the training set, with smaller sample categories receiving heavier weights and larger ones receiving less weights.

For small classes of incorrectly classified samples, we can increase the loss term to maximize the objective function. An anomaly detection technique may be used to create a single classifier for anomalies, resolving the data sample imbalance issue.

4.6 Conclusion

In this chapter, we aim to give a general understanding of what steel is. We have reviewed different types of steel products, their processes, their defects, and their inspection techniques and algorithms. Here is the summary of this chapter.

An end-to-end steel surface defect inspection system has three components: image classification, defect recognition, and quality inspection. Image acquisition involves measuring and capturing picture information from an object using an optical system, which includes a camera or analog camera and lighting. The camera captures the picture of the item to be recognized, while the light source helps the camera record a higher-quality image, which improves detection accuracy and training efficiency. The software program implements the fault recognition procedure; a more detailed explanation of the algorithm implemented in our thesis is provided in the experimental section.

In addition, steel is a key material in industrial production, and surface defect identification technology helps ensure high-quality manufacturing. Therefore, the study of steel surface fault identification technologies is crucial worldwide. This chapter also stated the most effective defect identification methods published recently, with the majority released during the last 10 years.

This chapter also covered the challenges faced when identifying steel surface defects, including insufficient data samples and unbalanced data samples. Finally, automated visual identification of steel surface defects aims to respond to changing global industrial competition patterns and gain an advantage over the competition. To improve the surface quality of industrial steel planar materials, it's important to integrate and collaborate on multiple technologies. Developing high-accuracy, real-time, and stable defect detection algorithms and instruments is crucial for automated control and process management.

5 Chapter V

EXPERIMENTATIONS, RESULTS AND DISCUSSION

5.1 Introduction

The main goal of our thesis is to classify images using intelligent surveillance systems and, hence, to recognize defects in the surfaces of steel sheets. In a nutshell, this chapter's contributions are comprised of different steps: First, a steel surface dataset of 1800 samples is proposed from the NEU Kaggle Competition for steel surface identification, which was established three years ago. The dataset comprises about 8,000 snapshots of steel faults. In this study, we chose to utilize only a small number of images (300 images per category) to evaluate the efficiency of our proposed technique. Secondly, a deep learning-based algorithm called NasNet-Mobile is used to classify six types of faults in steel surface sheets using improved methodologies explained in detail. NASNet-Mobile was chosen because it is a basic transfer learning model with just 5.3 million parameters, making it computationally efficient and quick to run. Furthermore, it strikes a nice compromise between acceptable performance and low-cost computations, making it an easy transfer learning model to implement. Finally, changing the hyper-parameters and fine-tuning the model allow for testing the method's viability. The obtained results are presented and analyzed at the end of the chapter.

5.2 Approach Description

Our plan includes four major steps:

Step 1: We preprocess the data and categorize it into six sorts of flaws (patches, crazing, pitted surface, scratches, rolling in size, and inclusion). This dataset is accessible at the SEVERSTAL Steel Detection Competition website [30].

Step 2: We utilize the pre-trained CNN NASNet-Mobile as the model's backbone to extract picture features; the top layers are frozen to use the ImageNet stored weights. The last block is then completely wiped and rebuilt with a brand new one (global average pooling, dropout, exponential linear unit (ELU), which represents the dense layers, and a Softmax function for the prediction and classification layer).

Step 3: We fine-tune the model using the resulting weights and switch between optimizers (ADAM optimizer and ADAMAX optimizer) to achieve the best results.

Step 4: We perform a comparison study to choose the best fine-tuned model.

5.2.1 Dataset Preparation

Northeastern University (NEU) [89] built a surface defect dataset that contains six typical surface defects of hot-rolled steel strips : rolling scale (RS), patches (Pa), cracking (Cr), pitting surface (PS), inclusions (In), and scratches. The dataset consists of about 8000 grayscale photos representing six various kinds of typical surface flaws, in our study, we took only 1800 images divided into six groups of steel defects. Each group contains 300 samples of one defect (**Figure V.1**). For defect classification tasks, the dataset includes annotations that specify the defect category in each image.

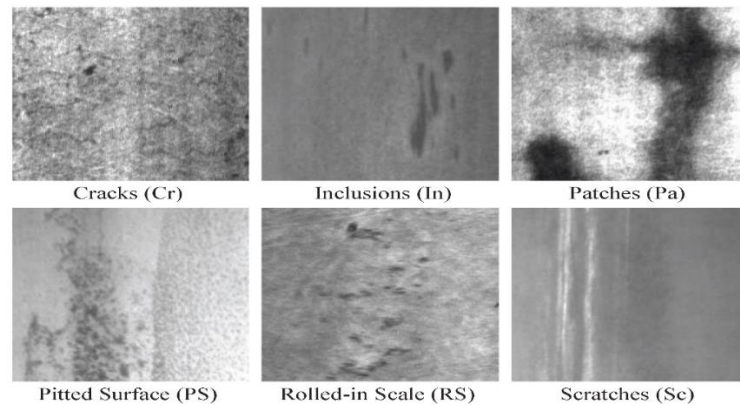


Figure 5-1 Some images from the NEU-CLS dataset [90].

5.2.2 Image Preprocessing

Both pre-processing and experimental data analysis are critical steps in understanding the nature of the data. This key quality allows variables and characteristics to be viewed. Steel is one of the most crucial components of infrastructure. This project has the potential to improve steel manufacturing efficiency by automating the problem identification process.

Data was acquired from Kaggle for data pre-processing and exploration data analysis purposes. The collection includes 1800 photos of defective and non-defective steel sheets. All of the photographs were shot with high-infrared cameras. There are 300 photos that are defect-free, and the rest are defective. All of the photos are tagged with classifications. Each class represents a certain fault type. The research revealed that the data is balanced, with 300 photos for each defect category. The collection only comprises photos that have a single defect type. The Python library was used to complete the implementation in Google Collaboratory [91]. Additional implementation information is provided in the implementation section.

5.2.3 Image Data Generator and Data Augmentation

The training data was created by randomly selecting 80% of the data (240 images for each fault class) from the NEU dataset. The remaining twenty percent is utilized to confirm the network's categorization. All data was enhanced with the Tensorflow [92] and Keras [93] libraries using "Image-Data-Generator" function (**Figure V.2** and **Figure V.3**). Rotation (0, 45, 90, 180 degrees), horizontal flipping, shearing (0.2), and zooming (0.2). Before feeding each image into our network, the pixel values were modified to lie between an interval of [-1; 1].

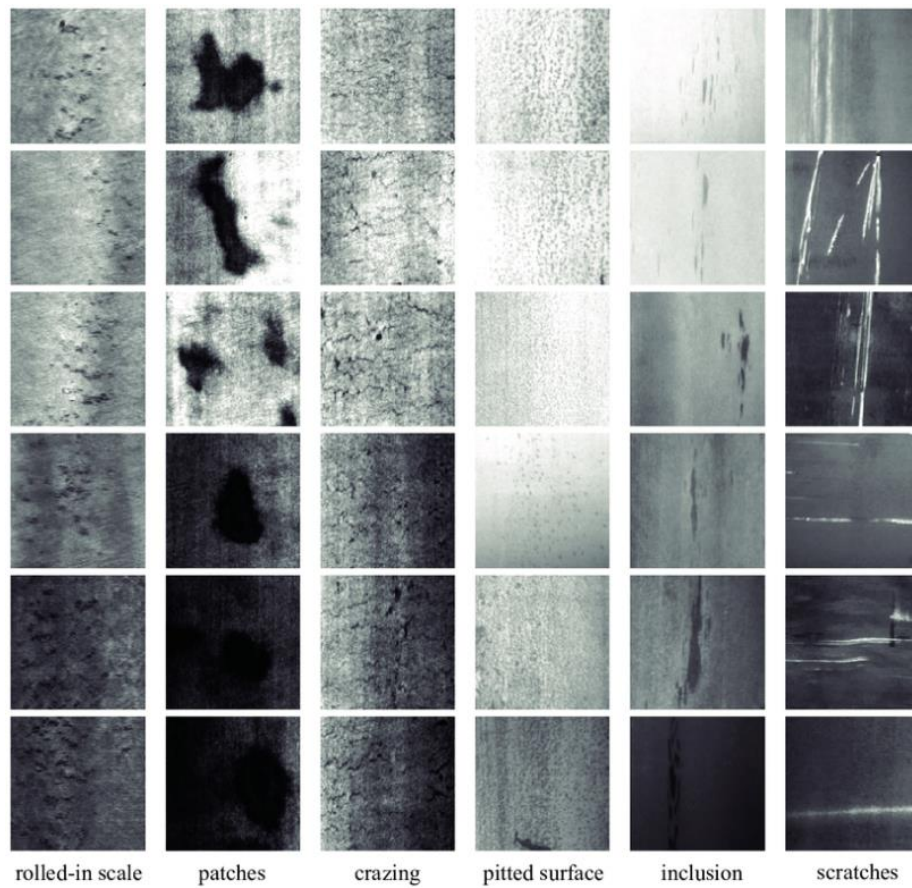


Figure 5-2 Data augmentation of different defect types

In data preprocessing as in **Figure V.3**, we flip the picture horizontally and vertically. Then we rotate each photo 45 degrees independently. Finally, we randomly crop different areas of the image and Add a Gaussian noise to the image.

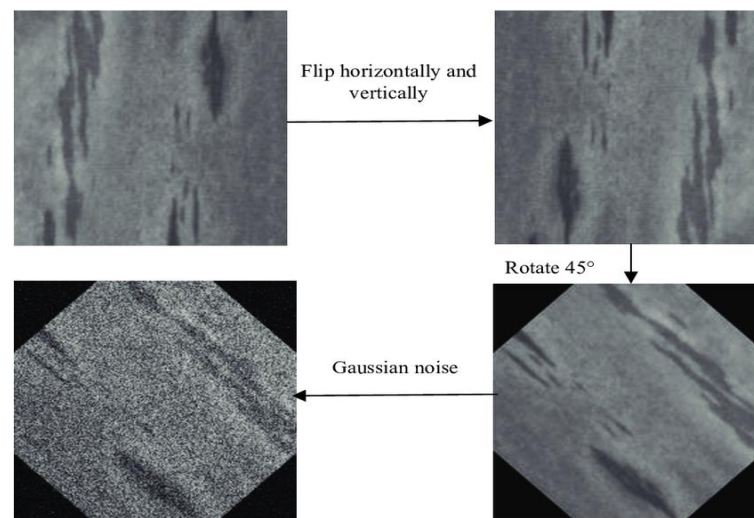


Figure 5-3 Data preprocessing using flipping, rotating and Gaussian Noise.

5.2.4 Classification model – Improved NASNet-Mobile

5.2.4.1 MobileNet

MobileNet is a considerably quicker CNN architecture with a smaller model that employs a new type of convolutional layer called as Depthwise Separable convolution. Because of their compact size, these types are thought to be ideal for usage in mobile and embedded applications. Hence the moniker MobileNet. The primary distinction between two-dimensional convolutions and Depthwise Convolution consists of the fact that 2D convolutions are conducted over all/multiple input channels, while Depthwise convolution keeps each channel independent.

➤ Depthwise Separable Convolution

Depthwise separable convolution involves using kernels that cannot be "factored" into smaller ones. As a result, it is increasingly commonly used. A depthwise separable convolution divides a kernel into two distinct kernels, each with two convolutions: depthwise and pointwise (Figure V.4 and Figure V.7).

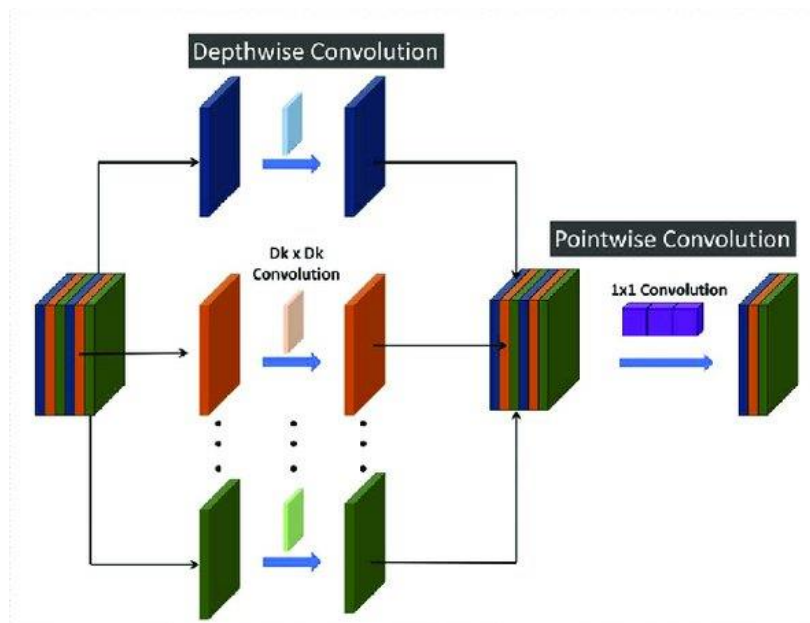


Figure 5-4 Depthwise Separable Convolution

A. Depthwise Convolution

Depthwise convolution involves applying a single convolutional filter to each input channel. The filter can be as deep as the input in standard 2D convolution over many input channels, allowing us to combine channels arbitrarily to generate each element in the output (Figure V.5).

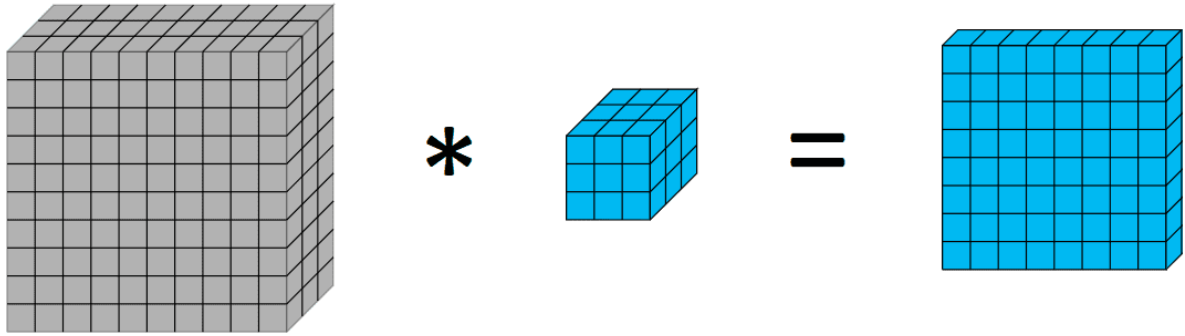


Figure 5-5 Depthwise Convolution

B. Pointwise Convolution

Pointwise Convolution is a type of convolution that uses a 1x1 kernel and iterates over each and every point. This kernel's depth equals the total number of channels in the input image. It may be adopted with depthwise convolutions to produce Depthwise Separable Convolutions, a valuable sort of convolution (Figure V.6).

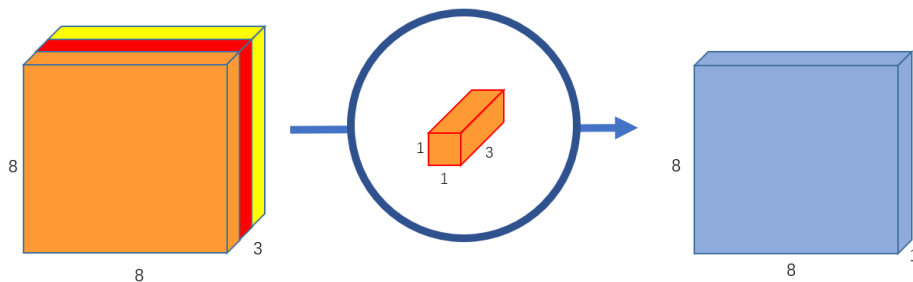


Figure 5-6 Pointwise convolution transforms a three-channel image into a one-channel image.

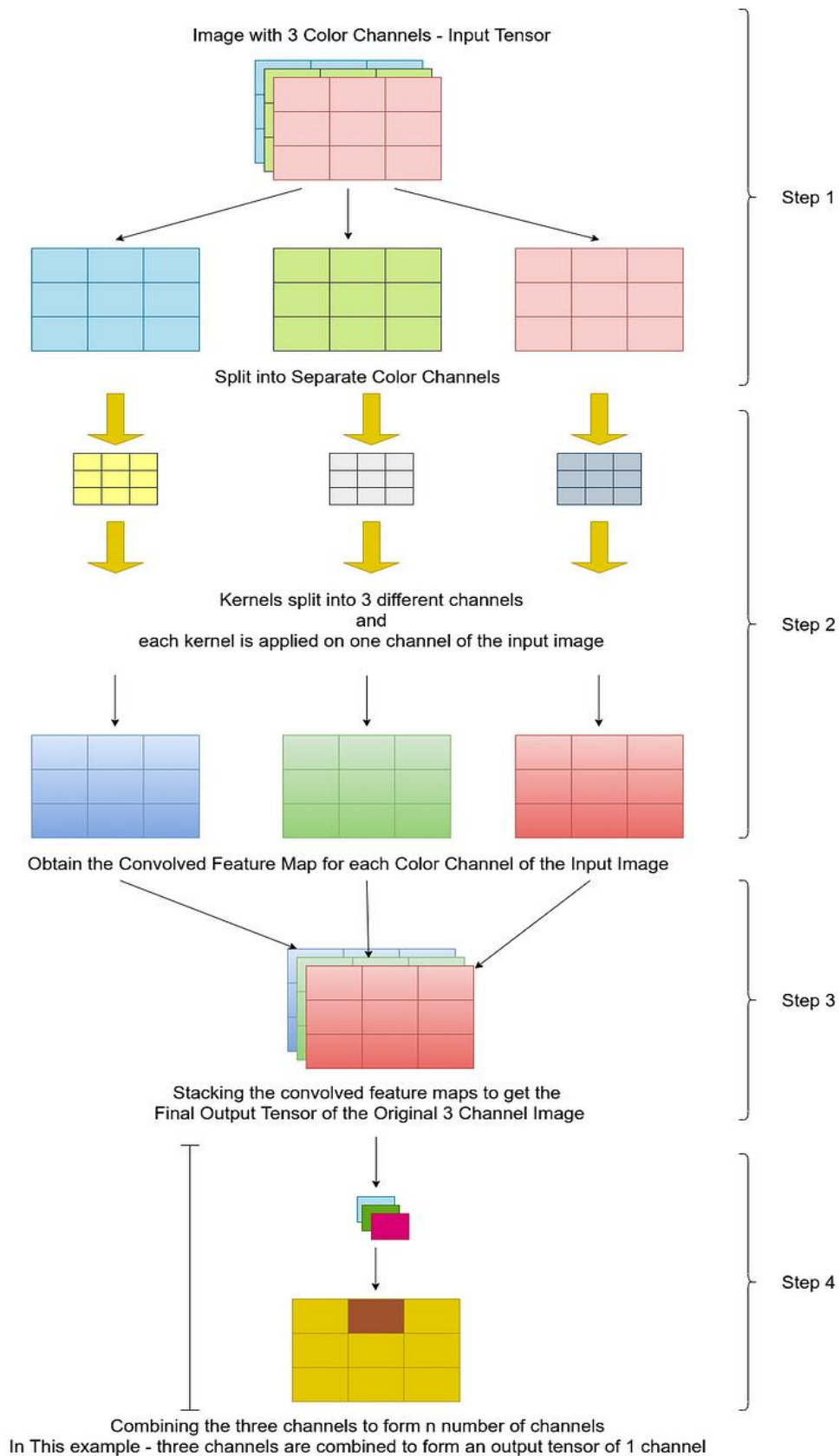


Figure 5-7 Diagrammatic explanation of Depthwise Separable Convolutions.

5.2.4.2 NASNet Framework

NASNet is a scalable CNN framework that uses reinforcement learning to optimize its fundamental building pieces (cells) [94]. A cell is made up of a few processes (e.g., Depthwise and pointwise separable convolutions and pooling) that are performed several times to meet network capacity requirements (see details in Chapter I).

5.2.4.3 Developed NASNet-Mobile for Steel Sheet Inspection

The basic NASNet-Mobile model is pre-trained with 1,056 ImageNet [95] recognition output channels. The main exploration in this design revolves upon the number of normal cells in the model. In our modified NasNet-Mobile architecture, we combined three reduction cells with three normal cells (**Figure V.8**). The total amount of characteristics is 4,376,022, with only 106,306 (2.42%) trainable and the remainder frozen. [96]

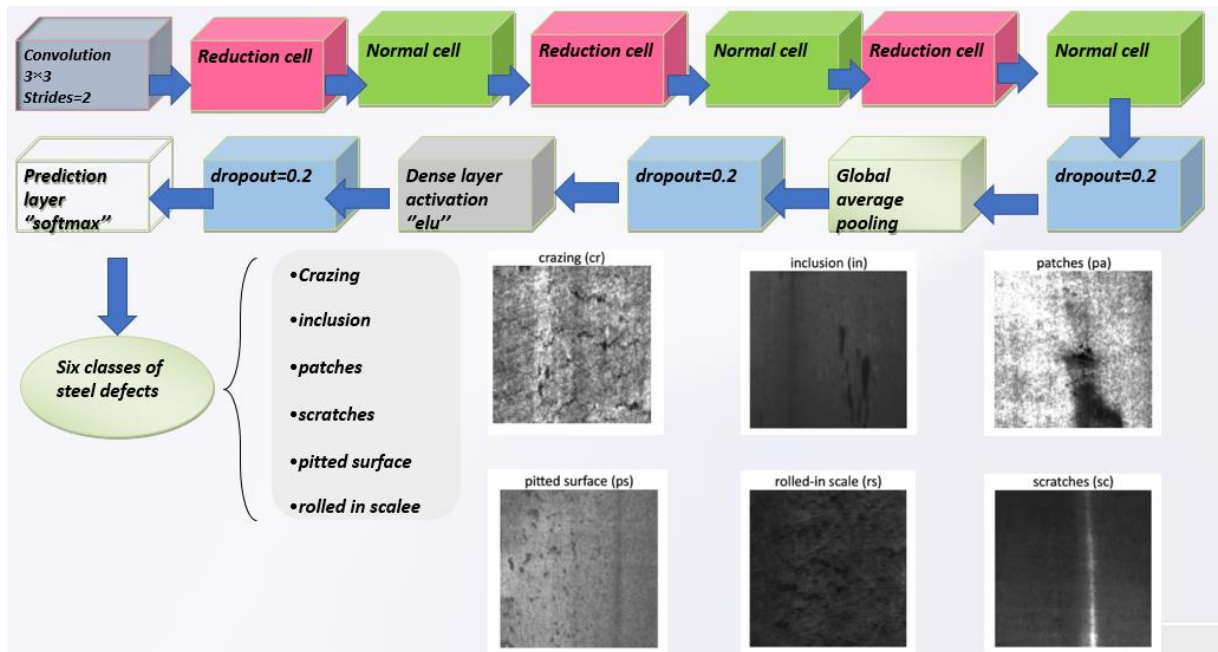


Figure 5-8 General structure of the proposed approach [97]

The pre-trained NasNet-Mobile framework is used as the backbone building design, which consists of six cells (reduced and normal), followed by a newly built defect classification block which includes a convolution layer, dense, dropout, and global average pooling. In the first dense layer, the activation function is "ELU" rather

than "ReLU," [98] which is a function that converges cost to a zero value faster and produces more accurate results [99]. ELU has an additional alpha constant that must be positive, as shown in Equation (5.1).

$$R(z) = \begin{cases} z & \text{when } z > 0 \\ \alpha \cdot (\exp(z) - 1) & \text{when } z < 0 \end{cases} \quad (5.1)$$

➤ Exponential Linear Unit activation function (ELU)

ELU is quite similar to ReLU, except for the negative inputs. They are both in the identity function form for non-negative inputs. In contrast, ELU smoothies gradually until it reaches $-\alpha$, whereas ReLU smoothies significantly (**Figure V.9**). ELU is preferred over ReLU as an activation function because it smooths out gradually till reaching α , while ReLU smooths out suddenly. Furthermore, unlike ReLU, ELU can generate negative outputs [100].

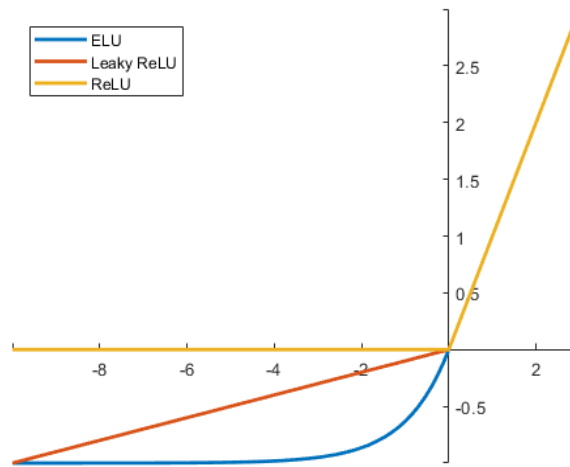


Figure 5-9 Graph demonstrating the distinction between ELU (green) and ReLU (red) activation functions.

➤ Advantages of exponential linear unit ELU

The ELU is a continuous as well as a differentiable activation function that provides quicker training durations than other linear non-saturating functions, such as ReLU and its several variants (Leaky-ReLU (LReLU) and Parameterized-ReLU). It does not suffer from dying neurons, exploding or vanishing gradients, and it achieves

higher accuracy than other activation functions like as ReLU, Sigmoid, and Hyperbolic Tangent.

Steel surface defect classifier parameters may be updated by lowering a multi-class loss function referred to as Categorical crossentropy (Equation 5.2).

$$LOSS = - \sum_{i=1}^{outputsize} y_i * \log \hat{y}_i \quad (5.2)$$

With, y_i is the i -th scalar value in the model output, y_i indicates the equivalent goal value, and output size refers to the number of scalar values in the output of the model.

5.2.4.4 Model Optimization for Steel Surface Defect Classification

Following the creation of the fundamental NASNet-Mobile algorithm for steel surface inspection, we present a number of possible solutions for increasing accuracy and lowering execution time. Initially data augmentation is utilized to increase the number of characteristics that the model learns. The second step a new block, which we have previously specified, is added to the bottom of the model for the prediction component; this block will aid enhance accuracy while reducing model parameters and execution time. Third, we try different optimizers to see which one works best (ADAM and ADAMAX). Finally, we lower the learning rate using exponential decay as in Equation 5.3 and we apply early stopping when the model's accuracy cannot increase any more. The model recovers the optimal weights (eq 5.3).

$$y = a(1 - b)^x \quad (5.3)$$

With y represents the final value (in this case, the learning rate after a certain time), a represents the initial value (starting learning rate), b represents the decay factor (a constant that determines how quickly the learning rate decreases), in addition x is the value of time that has elapsed (the number of iterations that have elapsed).

This equation 5.3 represents exponential decay, which is commonly used in deep learning to decrease the learning rate over time. The idea is that as training

progresses, we make smaller updates to the model weights to fine-tune the learning process.

At the beginning ($x=0$), the learning rate is at its initial value a . As training progresses (x increases), the term $(1 - b)^x$ causes the learning rate to shrink. The larger the decay factor b , the faster the learning rate decreases over time. So this prevents the model from making large, unstable updates when it is close to convergence.

5.2.4.5 Model Implementation

Our technique bases itself on the publicly accessible Python framework from Google Colaboratory [40]. Tesnorflow [31], Keras, Matplotlib, NumPy, and Glob are the primary libraries utilized in this implementation. We used 80% of the photographs from the NEU collection as training data (240 images for each fault category) and 20% as validation data. Performance is measured both before and after the network has been fine-tuned. Table V.1 displays the values of the hyper-parameters used to train the CNN.

The experiments were carried out with Windows 10 Professional on the Intel® Core (TM) -i5 7200U, 64-bit platform with 8GB RAM and NVIDIA RTX 2070, taking use of the free accessible GPU on Google Colab Platform. The training using the surface defect dataset was really quick. The model was trained in 3414 seconds (56 minutes and 54 seconds) before fine-tuning, and in 528 seconds (8 minutes and 48 seconds) after.

Table 5-1 Hyper-parameters used to train NASNet-Mobile Convolutional Neural Network.

Hyper-parameters	Before fine-tuning	After fine-tuning
Number of epochs	20	100
Steps per epoch	6	6
Number of trainable characteristics	106,306	4,376,022
Learning rate mode	Max	Exponential Decay
Restore best weights	True	True
Learning rate value	Min = 1 e-8 Max = 0.01 Steps = 6 Factor = 25%	Min = 0.01 Max = 0.1 Steps = 20 Factor 50%

	Patience = 3	Patience = 10
Loss	Categorical Crossentropy	Categorical Crossentropy
Optimizer	Adam	AdaMax

5.3 Results And Discussions

This section will provide the main results obtained during the experiments, graphs, tables, and figures will be analysed and discussed as well.

5.3.1 Model Evaluation

We ran our model twice, first without fine-tuning the parameters and then with them fine-tuned. The following deep learning metrics are used to evaluate the model: accuracy, loss, recall, AUC, FP, FN, TP, TN, and precision. We compare these metrics before and after fine-tuning, using different datasets for training and validation.

A. Performance of The Model Before Fine-Tuning

The results are shown in the tables and graphs below, with an analysis of each. The metrics in Table 3 reveal highly promising results for both the training and validation datasets. We can see a tiny decrease between them because the model learns from the training data, making it more trustworthy, but validation data shows that it only contains 20% of the entire data and that the model has never learned from it. Because assessing the model on the training dataset may yield biased results, it is evaluated using held-out sampling to provide an impartial assessment of its competence. Model fine-tuning and dataset augmentation, which allow the model to learn more characteristics, are two strategies that may be used to offset the performance gap.

Figure V.10 shows the training and validation curves for the NasNet-Mobile optimization produced using the previously stated dataset of 1800 pictures augmented using Image Data Generator. The training lasted 20 epochs, with a break on the 12th.

We can see a decreasing trend as the number of epochs increases, followed by validation loss and training loss.

During the learning phase, the model appears to recognize the visual prominence of both the reference and candidate images. As a result, the loss incurred during exercise tends to reduce. The photographs were selected at random during the testing period.

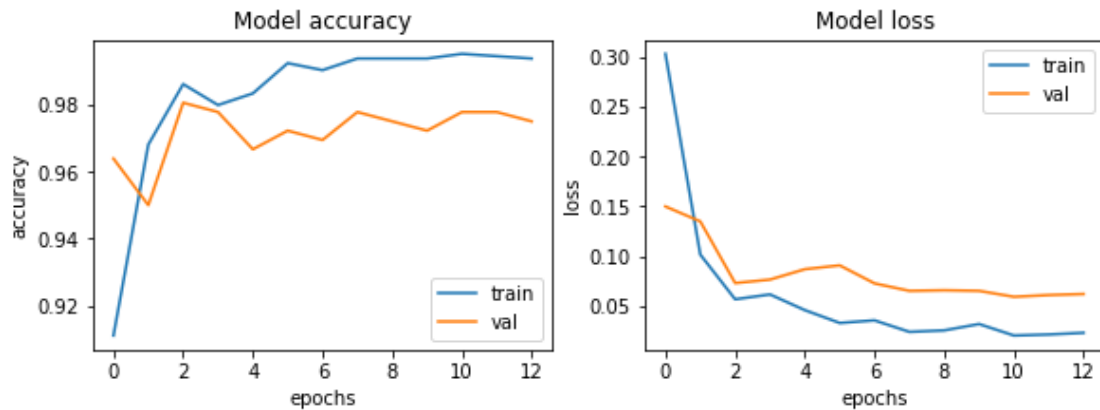


Figure 5-10 Accuracy and loss curves before fine-tuning the NASNet-Mobile

These photos come from a separate class and have never been given to the network throughout training. As a result, as the training steps advance, we see that the accuracy of the set to be tested approaches that of the training set while being only slightly lower. This demonstrates that the algorithm, which was trained on examples from the training set, accurately predicts cases that were not in the training set.

As we see in **Figure V.11** and **Table V.2**, the best-achieved Accuracy is around 99.51% in training and 98.6% in validation. The Recall is around 99.51% for training and 97.78% for validation datasets. These findings were obtained prior to fine-tuning.

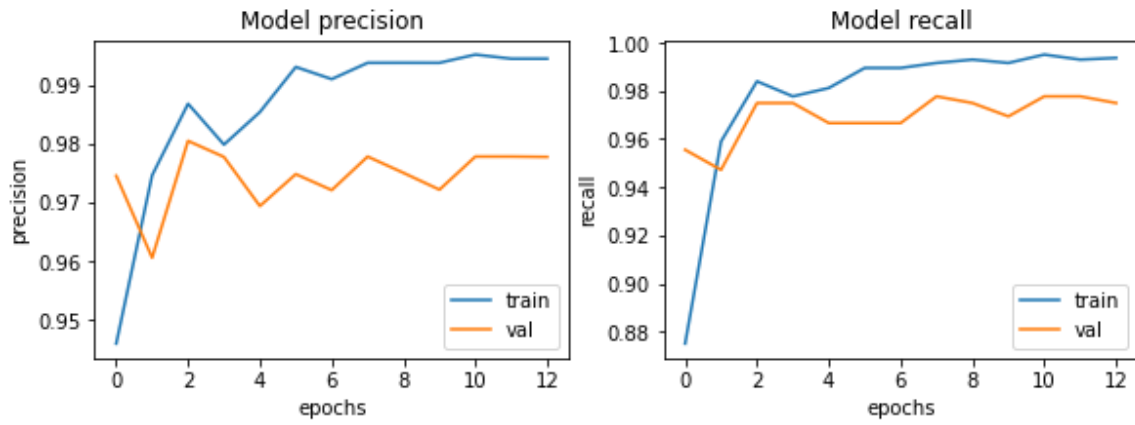


Figure 5-11 Precision and accuracy curves before fine-tuning the NASNet-Mobile.

Table 5-2 Performance metrics of NASNet-Mobile in the training and validation dataset before fine-tuning

Metrics	Training Dataset	Validation Dataset
Accuracy	99.51%	97.78%
Loss	0.028	0.064
Precision	99.51%	98.60%
Recall	99.51%	97.78%
Area under the curve AUC	100%	99.96%

B. Performance of the model after fine-tuning

Table 5-3 Performance metrics of NASNet-Mobile in the training and validation dataset after fine-tuning

Metrics	Training dataset	Validation dataset
Accuracy	100%	98.06%
Loss	0.0245	0.0729
Precision	100%	98.04%
Recall	100%	97.50%
Area under the curve AUC	100%	99.44%

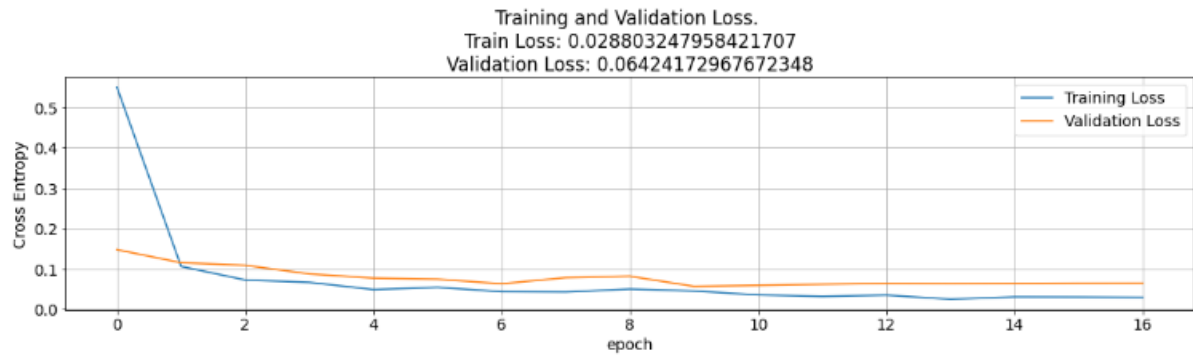


Figure 5-12 Training and validation Loss

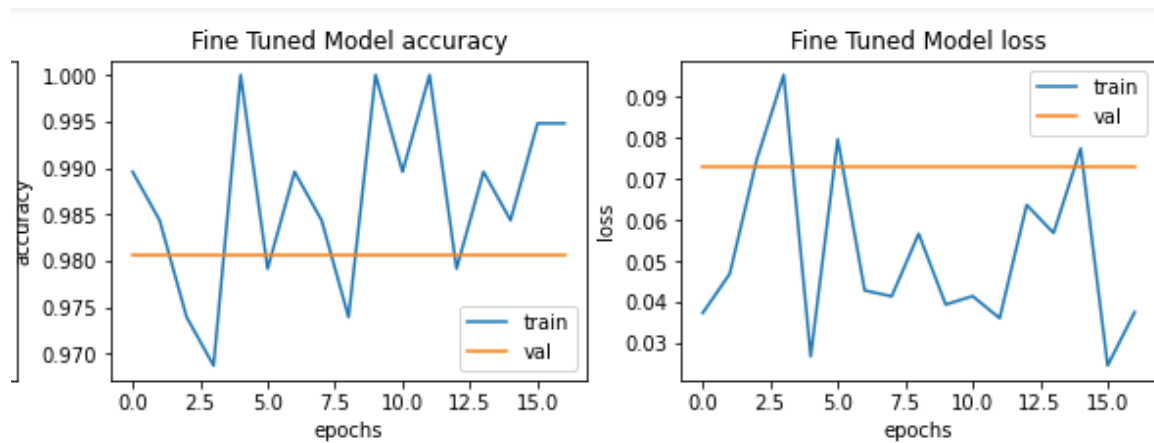


Figure 5-13 accuracy and loss after fine-tuning the Model.

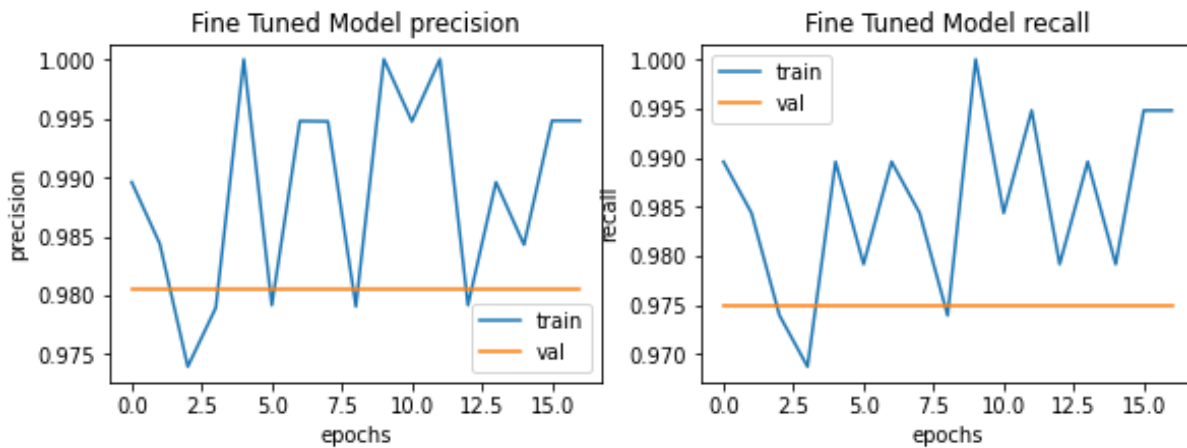


Figure 5-14 Precision and recall loss in the fine-tuned Model.

Training procedure was early stopped, as shown in **Figure V.12**, since there was no progress during three consecutive epochs. We can clearly see the instability of both accuracy and loss in the training and validation datasets, as shown in **Figure V.13** and **Figure V.14**, despite the fact that the model's accuracy can reach 100% (**Table V.3**),

which is very good considering the small amount of data (only 300 images per class, 240 for training, and 60 for validation). This instability is induced by the lack of batch normalization in the studies. Batch normalization improves the training process by decreasing internal covariate changes, increasing stability, and optimizing the model. It also enhances generalization by normalizing layer activations, eliminating overfitting, and decreasing the sensitivity of initial weight.

5.3.1.1 Test results visualization

Here we want to test the ability of defect recognition by the model. The data were not seen before and were stacked in the test dataset.

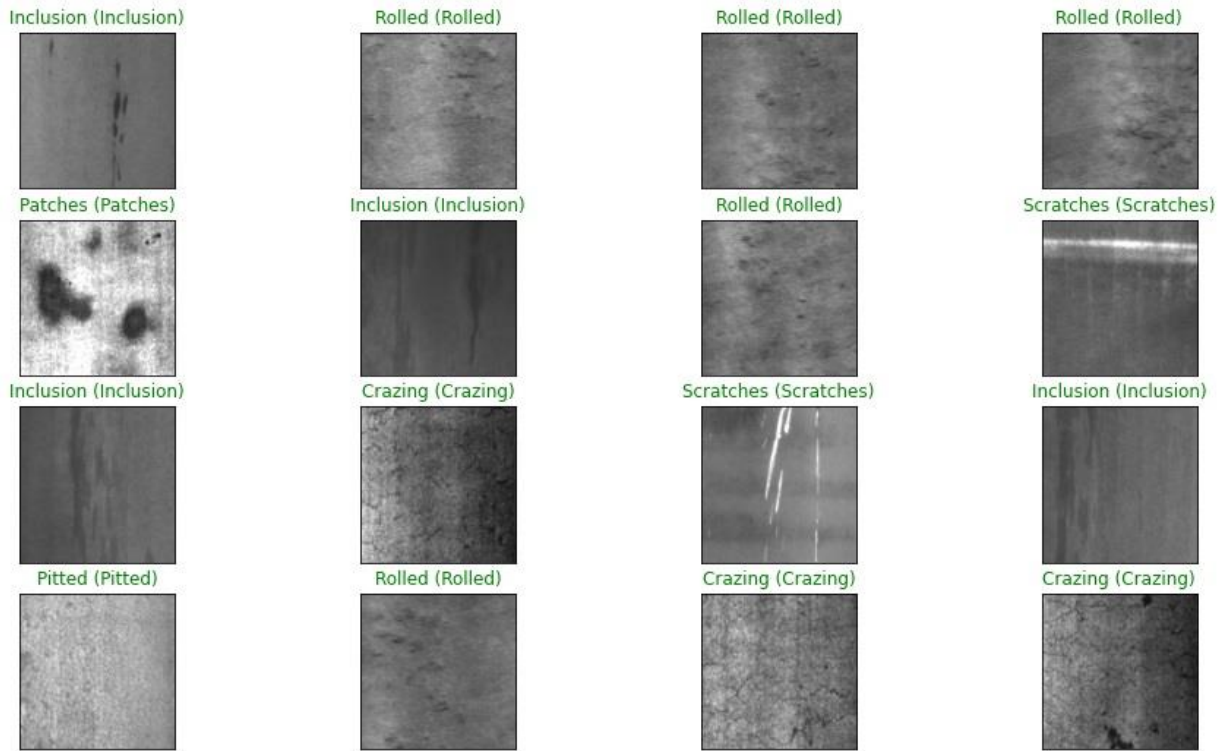


Figure 5-15 The model predicts the defects correctly for unseen data.

Figure V.15 shows that the model correctly classified defects in steel sheets ; however, it can make mistakes in defect recognition when the faults are similar, like in **Figure V.16**. Therefore, further modifications can be adopted to improve the training of the network.

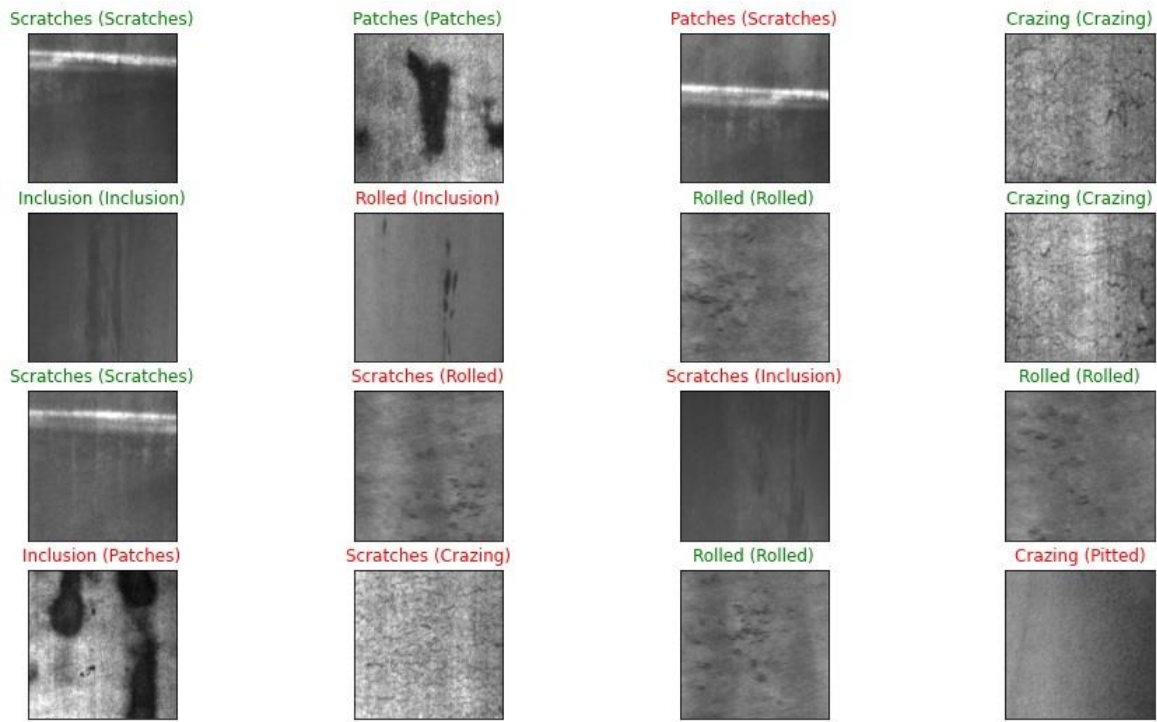


Figure 5-16 The model predicts some faults incorrectly (red) and correctly (green).

5.3.1.2 Comparative Study

In this section, we'll compare our model to popular steel surface inspection methods that are already in use. In addition to accuracy, we consider execution time, model lightness (number of parameters), and data size.

According to **Table V.4**, our model clearly outperforms the aforementioned models in terms of accuracy, with a score of 99.51%, which is greater than the other accuracies. The suggested NASNet-Mobile achieves extremely satisfying results with fewer model parameters and running time, and it does not require significant amounts of data: 48 times lower than [34], 7 times lower than [33], and 14 times lower than [41]. Our model appears to be lighter (just 5.3 million parameters) compared to DeCAF's model [41] with 60 million parameters and the residual neural network's 25.6 million parameters [22].

Table 5-4 The classification accuracy (%) for multiple state-of-the-art steel surface defect classifiers, taking into consideration the triplet model lightness, execution time, and data size.

Method	Accuracy	Model Lightness (number of characteristics)	Time (ms) Per Inference step	Data Size
[34]	96.91%	25.6 million	58.2	87704
[33]	98.56%	25.6 million	77.1	12568
[41]	97.27%	60 million	10.3	26334
Our method (Developed NASNet- Mobile)	99.51%	5.3 million	27.0	1800

Our model has the lowest error rate when compared to other models. In **Table V.5**, our model has an error rate of 0.028, which is 22 times lower than the CNN approach with LU's activation function. This means that our model can learn with fewer mistakes and more precision. Finally, our network has a faster execution time than the others. As can be seen, it is three times smaller than the [33] model and 2.5 times smaller than the [22] model.

Table 5-5 Evaluation of Classification errors depending on the activation function, the model employs the name of the function enclosed in parenthesis.

Method	Error Rate (%)
CNNs (Sigmoid)	12.8721
CNNs (Hyperbolic Tangent)	18.1129
CNNs (ReLU) Recall	9.5018
CNN (LUs)	0.6292
Our method (ELUs)	0.028

Finally, our network has a faster execution time than the others. As can be seen, it is three times smaller than the [33] model and 2.5 times smaller than the [22] model.

5.4 Conclusion

This chapter introduces a novel method for categorizing defects in steel sheets by utilizing a pre-trained NASNet-Mobile CNN. Here are the main conclusions drawn from the experimentation and results section:

The NASNet-Mobile network is employed to tackle the challenge of precise classification in scenarios with limited memory, given its lower parameter count compared to other CNNs (5.3 million characteristics). The upper layers of this CNN were frozen to reduce memory usage and computational load while preserving weight. By leveraging the free GPU provided on the Google Colab Platform, the issue of long execution time has been resolved. This study effectively tackles the issue of dataset scarcity by leveraging the capabilities of the CNN model. With a substantial depth of 389 layers, the CNN can capture essential image characteristics and extract more features even with a limited dataset size.

After fine-tuning the hyperparameters, there was an improvement in the model's performance. We found that the AdaMax optimizer performs better than the ADAM optimizer in our modified NASNet-Mobile architecture. Both accuracy and error rate exhibited slight enhancements. Explaining the Adam optimizer, it adjusts weights based on the scaled L2 norm of previous gradients, while AdaMax takes this a step further by considering the infinite norm of previous gradients.

The model has been altered, with the final block being entirely removed and substituted with a new one. Various modifications were implemented to address the requirements of the steel image. The convolution layers were implemented with an ELU activation function. We planned to implement this activation function to leverage its benefits. This activation function helps address the dying RELU problem, which arises when the gradient becomes 0 on the negative side of the graph. Due to this, the weights and biases of certain neurons remain constant throughout the backpropagation process. This may result in inactive neurons that do not transmit signals. Addressing the RELU dying problem involves incorporating a log curve for negative input values. It assists the network in adjusting weights and biases appropriately.

Utilizing the ELU activation function enhances model performance and network robustness. It gradually becomes smoother until it reaches zero, while RELU

smooths out more noticeably. An additional dropout of 0.2 was included following each fully connected layer, and a global average pooling step was added before the dense layer to optimize time and memory usage while constructing the model. Adjusting the learning rate over time enables us to take larger steps initially and then gradually decrease the step size as the weights get closer to their optimal values.

There are opportunities for further enhancements through gathering additional training data, enhancing the network's architecture, and optimizing its hyperparameters instead of simply extending the training epochs within the existing framework to avoid overfitting.

Overall, it can be deduced that the suggested method could be applied in image processing tasks like image classification to address the three main challenges: time consumption, memory inadequacy, and limited data encountered by small mills and non-powerful operators.

GENERAL CONCLUSION AND PERSPECTIVES

Monitoring the industrial environment is crucial for avoiding unexpected repairs and shutdowns and detecting faulty goods that can lead to significant losses. Real-time system tracking is now possible because of data-driven methodologies and advances in sensor technology, such as the Internet of Things (IoT). QC measures can continually check a product's health throughout its production duration. Quality inspection evaluates and determines if a product is acceptable or rejected. Visual inspection, often known as final inspection, includes a human operator viewing a product to determine its state.

Visual inspection is a key aspect of quality control in manufacturing. Here, operators try to visually check the product's status at various stages to determine whether it may proceed to the next operation. Sinclair offered a four-step visual inspection process that included presenting the product, looking for potential faults or defects, making a judgment based on the evaluation, and taking action. Wang and Drury defined the visual inspection process as a series of sub-tasks, including orienting the item, looking for faults, identifying and classifying defects, reaching a judgment, dispatching the item, and documenting any information. The purpose is to find faults rapidly and correctly, allowing you to make educated judgments.

However, a number of factors influence the visual inspection process, resulting in an industry-wide inspection accuracy of around 80%. Manual visual assessment can be both time-consuming and expensive when attempting to achieve 100% inspection in sophisticated manufacturing processes. Computer Vision (CV)-based algorithms have assisted in automating certain aspects of the visual inspection process, but there are still unresolved issues.

Because of the multiple elements that might influence an operator during the visual inspection process, data-driven techniques are becoming more popular for detecting faulty items. Traditional ML approaches sometimes need domain expertise during the feature creation or feature engineering process, whereas DL methods may automatically choose and train abstract features. CNN, autoencoders, and Recurrent

Neural Networks (RNN)-based deep learning algorithms outperform a wide range of inspection applications.

These DL-based methods are used in a variety of domains. In our case, the industrial domain, we aimed to inspect the quality of industrial steel surfaces. As it is known, steel is an important metal for industrial civilization, and its use is a good measure of a country's growth. The World Steel Association claimed in 2013 that crude steel output had reached 1,582 million tons, exceeding all other metals combined. There are about 3,500 grades of steel, with flat steel goods accounting for over half of commerce. An integrated iron and steel factory uses blast furnaces to manufacture liquid iron from iron ore, coke, sinter, and flux. This liquid steel is then formed into slabs and billets, the former having a rectangular cross section and the latter having a square cross section. These slabs are subsequently rolled into hot and cold strips, and billets are formed into structural components. The surface quality of metals, especially cold-rolled steel, has grown in importance since the 1980s, owing mostly to the needs of automobile manufacturers. Hot strip surface quality and architectural product surface quality are also becoming more essential.

In the early 1980s, nine American steel and aluminum companies worked together on a vision-based steel surface assessment initiative. In 1987, a prototype system was created and tested at many steel factories. European businesses have also begun to develop this technology. Since then, systematic research into the surface inspection of steel products has begun, with several credible businesses producing vision-based automated surface inspection systems (ASIS). The technology for vision-based automated examination of steel items has advanced.

A good, or, let's say, an excellent, alternative for human inspection when we talk about image classification in the concept of defect recognition—in our case, the recognition of six types of defects in steel sheet—are Convolution Neural Networks CNNs. This work described an AI-based approach to visual inspection that employs deep learning algorithms. The technique consists of a customized Convolutional Neural Network (CNN) called NASNet-Mobile.

NASNet-Mobile is a convolutional neural network trained using more than one million photos from the ImageNet collection. The network can identify photos into 1000 item categories, including keyboards, mice, pencils, and other animals. As a

consequence, the network has learned detailed feature representations for a diverse set of pictures. The network's image input size is 224 by 224.

In this thesis, we developed the NASNet-Mobile using fine-tuning techniques to enhance its performance, like transfer learning, modifying layers, and replacing the ReLU activation function with the Exponential Linear Unit (ELU). ELU helped solve the problem of dying neurons. We also switched between optimizers ADAM and ADAMAX to better optimize the model. More improvements were applied to the data using preprocessing techniques like data collection, selection, resizing, and data augmentation.

The proposed algorithm was evaluated using different metrics, including the overall Accuracy, Precision, Recall, and Confusion Matrix. These metrics helped demonstrate the results of the inspection procedure. The suggested model has an inspection accuracy of 99.51% based on image data from the Open-access NEU dataset before model fine-tuning. In addition, our model achieved a 100% after fine-tuning.

➤ **Answering the Questions of the Problem Statement Section**

1. *Can we develop a surface defect classification technique using computer vision and deep learning algorithms?*

Yes, surface defect classification techniques were developed using Convolution Neural Network CNNs and Network Architecture Search. These two DL methods combined gave us NASNet-Mobile for flaw classification of steel flats.

2. *How can we tackle the problem of the long execution time of the algorithm, which in fact delays the inspection time in the production line?*

The problem of long execution time was solved in this work using a light-weight model with fewer computations and easy and fast model building.

3. *What are the alternative settings when the number of defected steel images is not sufficient, which is so familiar in small industries that do not own very sophisticated cameras?*

In the case of insufficient data, we can use some data preprocessing techniques that help enhance the quality of the image. As well as its ability to generate more data. Data Augmentation is employed in our work.

4. *Can we find a CNN architecture for defect inspection that gives satisfying accuracy even though the processing units are powerless?*

Yes, and this is what was adopted in this thesis. The use of a small neural network architecture with fewer characteristics (5.3 million) and fine-tuning the hyperparameters and using improvement techniques like transfer learning, regularization, model block modifications, and switching between optimizers are all good alternatives that can address the problem of small and powerless processing units.

In this thesis, we developed a defect identification technique for steel surfaces that allowed for the classification of six types of faults in steel flats. The dataset was selected and preprocessed using Image-Data-Generator from Tensorflow and Keras; this preprocessing allowed the model to learn to extract deeper features from different images. The third objective that was reached is to design a CNN architecture for steel inspection operations, considering severe conditions like illumination and a lack of dataset.

The main objective was to classify defects in steel sheet images automatically using advanced deep learning algorithms and image preprocessing techniques. Convolutional neural networks were trained and evaluated on publicly available datasets containing diverse types of steel defects. Optimization, transfer learning, regularization, and preprocessing techniques were used to improve the accuracy of the models. This study demonstrated the potential of collaboration between industry and academia in overcoming hardware advancements and improving quality inspection operations.

➤ **Future Perspectives**

The work between our hands is not perfect and looks for future enhancements to get better results. It is suggested to work on a larger dataset to let the model get

more features and learn to generalize better. The model can be further fine-tuned according to the task to be treated and the dataset in question. Also, more data sophistication and preprocessing techniques can be adopted, like data generation using Generative Adversarial Networks GANs or similar approaches.

Better coordination between CNNs makes the model benefit from their full advantages.

References

- [1] N. Neogi, D. K. Mohanta, and P. K. Dutta, "Review of vision-based steel surface inspection systems," *EURASIP Journal on Image and Video Processing*, vol. 2014, no. 1, p. 50, 2014/11/13 2014, doi: 10.1186/1687-5281-2014-50.
- [2] S. Ghorai, A. Mukherjee, M. Gangadaran, and P. Dutta, "Automatic Defect Detection on Hot-Rolled Flat Steel Products," *Instrumentation and Measurement, IEEE Transactions on*, vol. 62, pp. 612-621, 03/01 2013, doi: 10.1109/TIM.2012.2218677.
- [3] K.-C. Song and Y. Yan, "A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects," *Applied Surface Science*, vol. 285, pp. 858-864, 11/01 2013, doi: 10.1016/j.apsusc.2013.09.002.
- [4] R. Usamentiaga, F. D. Garcia, J. Molleda, F. Bulnes, and G. Bonet, *Vibrations in Steel Strips: Effects on Flatness Measurement and Filtering*. 2013, pp. 1-10.
- [5] H. Di, X. Ke, Z. Peng, and D. Zhou, "Surface defect classification of steels with a new semi-supervised learning method," vol. 117, pp. 40-48, 06/01 2019, doi: 10.1016/j.optlaseng.2019.01.011.
- [6] S. Sakib and M. Uddin, "Digital Image Restoration And Image Classification Using Deep Learning," 2019.
- [7] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.
- [8] M. W. Ashour, F. Khalid, A. Abdul Halin, L. N. Abdullah, and S. H. Darwish, "Surface Defects Classification of Hot-Rolled Steel Strips Using Multi-directional Shearlet Features," *Arabian Journal for Science and Engineering*, vol. 44, no. 4, pp. 2925-2932, 2019/04/01 2019, doi: 10.1007/s13369-018-3329-5.
- [9] X. Wu *et al.*, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1-37, 2008/01/01 2008, doi: 10.1007/s10115-007-0114-2.
- [10] B. Baharudin, L. H. Lee, K. Khan, and A. Khan, "A Review of Machine Learning Algorithms for Text-Documents Classification," *Journal of Advances in Information Technology*, vol. 1, 02/01 2010, doi: 10.4304/jait.1.1.4-20.
- [11] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer New York, 2013.
- [12] H. Hu, Y. Liu, M. Liu, and L. Nie, "Surface Defect Classification in Large-scale Strip Steel Image Collection via Hybrid Chromosome Genetic Algorithm," *Neurocomputing*, vol. 181, 12/01 2015, doi: 10.1016/j.neucom.2015.05.134.
- [13] X.-W. Zhang, Y.-Q. Ding, Y.-y. Lv, A.-y. Shi, and R.-y. Liang, "A vision inspection system for the surface defects of strongly reflected metal based on multi-class SVM," *Expert Syst. Appl.*, vol. 38, pp. 5930-5939, 05/01 2011, doi: 10.1016/j.eswa.2010.11.030.
- [14] K. Agarwal, R. Shivpuri, Y. Zhu, T.-S. Chang, and H. Huang, "Process knowledge based multi-class support vector classification (PK-MSVM) approach for surface defects in hot rolling," *Expert Syst. Appl.*, vol. 38, pp. 7251-7262, 06/01 2011, doi: 10.1016/j.eswa.2010.12.026.
- [15] R. Gong, C. Wu, and M. Chu, "Steel surface defect classification using multiple hyper-spheres support vector machine with additional information," *Chemometrics and Intelligent Laboratory Systems*, vol. 172, pp. 109-117, 2018/01/15/ 2018, doi: <https://doi.org/10.1016/j.chemolab.2017.11.018>.

- [16] K. Liu, A. Li, X. Wen, H. Chen, and P. Yang, *Steel Surface Defect Detection Using GAN and One-Class Classifier*. 2019, pp. 1-6.
- [17] I. Goodfellow *et al.*, "Generative Adversarial Networks," *Advances in Neural Information Processing Systems*, vol. 3, 06/10 2014, doi: 10.1145/3422622.
- [18] D. Wettschereck, D. W. Aha, and T. Mohri, "A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms," *Artificial Intelligence Review*, vol. 11, no. 1, pp. 273-314, 1997/02/01 1997, doi: 10.1023/A:1006593614256.
- [19] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, "Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification," in *Computer Vision – ECCV 2018*, Cham, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., 2018// 2018: Springer International Publishing, pp. 318-335.
- [20] F. Dupont, C. Odet, and M. Cartont, "Optimization of the recognition of defects in flat steel products with the cost matrices theory," *Ndt & E International - NDT E INT*, vol. 30, pp. 3-10, 02/01 1997, doi: 10.1016/S0963-8695(96)00045-X.
- [21] Q. Luo, Y. Sun, P. Li, O. Simpson, L. Tian, and H. Yigang, "Generalized Completed Local Binary Patterns for Time-Efficient Steel Surface Defect Classification," *IEEE Transactions on Instrumentation and Measurement*, vol. PP, pp. 1-13, 07/27 2018, doi: 10.1109/TIM.2018.2852918.
- [22] J. Zhao, Y. Peng, and Y. Yan, "Steel Surface Defect Classification Based on Discriminant Manifold Regularized Local Descriptor," *IEEE Access*, vol. PP, pp. 1-1, 11/19 2018, doi: 10.1109/ACCESS.2018.2881962.
- [23] R. Medina *et al.*, "Automated visual classification of frequent defects in flat steel coils," *The International Journal of Advanced Manufacturing Technology*, vol. 57, pp. 1087-1097, 12/01 2011, doi: 10.1007/s00170-011-3352-0.
- [24] P. Caleb-Solly and M. Steuer, *Classification of surface defects on hot rolled steel using adaptive learning methods*. 2000, pp. 103-108 vol.1.
- [25] Q. Luo *et al.*, "Automated Visual Defect Classification for Flat Steel Surface: A Survey," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 12, pp. 9329-9349, 2020, doi: 10.1109/TIM.2020.3030167.
- [26] T. Czimmermann *et al.*, "Visual-Based Defect Detection and Classification Approaches for Industrial Applications – A SURVEY," *Sensors*, vol. 20, p. 1459, 03/06 2020, doi: 10.3390/s20051459.
- [27] J. Wang, Y. Ma, L. Zhang, R. Gao, and D. Wu, "Deep Learning for Smart Manufacturing: Methods and Applications," *Journal of Manufacturing Systems*, vol. 48, pp. 144-156, 01/08 2018, doi: 10.1016/j.jmsy.2018.01.003.
- [28] G. Psuj, "Multi-Sensor Data Integration Using Deep Learning for Characterization of Defects in Steel Elements," *Sensors (Basel, Switzerland)*, vol. 18, 01/19 2018, doi: 10.3390/s18010292.
- [29] Y.-H. Shih, Pan, J. Ma, X. Zhu, and Y. Zhang, "A High-Performance Deep Learning Algorithm for the Automated Optical Inspection of Laser Welding," *Applied Sciences*, vol. 10, p. 933, 01/31 2020, doi: 10.3390/app10030933.
- [30] D. Tabernik, S. Šela, J. Skvarc, and D. Skočaj, "Segmentation-based deep-learning approach for surface-defect detection," *Journal of Intelligent Manufacturing*, vol. 31, 03/01 2020, doi: 10.1007/s10845-019-01476-x.
- [31] X. Wen, J. Shan, Y. He, and K. Song, "Steel Surface Defect Recognition: A Survey," *Coatings*, vol. 13, no. 1, doi: 10.3390/coatings13010017.

- [32] G. Fu *et al.*, "A deep-learning-based approach for fast and robust steel surface defects classification," *Optics and Lasers in Engineering*, vol. 121, pp. 397-405, 2019-10-01 2019, doi: 10.1016/j.optlaseng.2019.05.005.
- [33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," vol. 7, 12/10 2015.
- [34] D. He, K. Xu, and P. Zhou, "Defect detection of hot rolled steels with a new object detection framework called classification priority network," *Computers & Industrial Engineering*, vol. 128, pp. 290-297, 2019/02/01/ 2019, doi: <https://doi.org/10.1016/j.cie.2018.12.043>.
- [35] Y. Kateb, H. Megloul, and A. Khebli, "Steel Surface Defect Detection Using Convolutional Neural Network," *Algerian Journal of Signals and Systems*, vol. 5, no. 4, pp. 203-208, 2020-12-15 2020, doi: 10.51485/ajss.v5i4.122.
- [36] I. Konovalenko, P. Maruschak, J. Brezinová, J. Viňáš, and J. Brezina, "Steel Surface Defect Classification Using Deep Residual Neural Network," *Metals*, vol. 10, no. 6, doi: 10.3390/met10060846.
- [37] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv 1409.1556*, 09/04 2014.
- [38] X. Wan, X. Zhang, and L. Liu, "An Improved VGG19 Transfer Learning Strip Steel Surface Defect Recognition Deep Neural Network Based on Few Samples and Imbalanced Datasets," *Applied Sciences*, vol. 11, no. 6, doi: 10.3390/app11062606.
- [39] S. Guan, M. Lei, and H. Lu, "A Steel Surface Defect Recognition Algorithm Based on Improved Deep Learning Network Model Using Feature Visualization and Quality Evaluation," *IEEE Access*, vol. PP, pp. 1-1, 03/10 2020, doi: 10.1109/ACCESS.2020.2979755.
- [40] X. Feng, X. Gao, and L. Luo, "X-SDD: A New Benchmark for Hot Rolled Steel Strip Surface Defects Detection," *Symmetry*, vol. 13, no. 4, doi: 10.3390/sym13040706.
- [41] S. Wen, Z. Chen, and C. Li, "Vision-Based Surface Inspection System for Bearing Rollers Using Convolutional Neural Networks," *Applied Sciences*, vol. 8, no. 12, p. 2565, 2018-12-11 2018, doi: 10.3390/app8122565.
- [42] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 21-26 July 2017 2017, pp. 1800-1807, doi: 10.1109/CVPR.2017.195. [Online]. Available: <http://doi.ieeecomputersociety.org/10.1109/CVPR.2017.195>
- [43] O. Ronneberger, P. Fischer, and T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015, pp. 234-241.
- [44] A. Boikov, V. Payor, R. Savelev, and A. Kolesnikov, "Synthetic Data Generation for Steel Defect Detection and Classification Using Deep Learning," *Symmetry*, vol. 13, no. 7, doi: 10.3390/sym13071176.
- [45] S. N, N. Ramasamy, and K. Ramar, "Localization and segmentation of metal cracks using deep learning," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, 03/01 2021, doi: 10.1007/s12652-020-01803-8.
- [46] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

- [47] Y.-C. Huang, K.-C. Hung, and J.-C. Lin, "Automated Machine Learning System for Defect Detection on Cylindrical Metal Surfaces," *Sensors*, vol. 22, no. 24, p. 9783, 2022-12-13 2022, doi: 10.3390/s22249783.
- [48] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016-04-13 2016, doi: 10.1098/rsta.2015.0202.
- [49] S. Youkachen, M. Ruchanurucks, T. Phatrapornnant, and H. Kaneko, *Defect Segmentation of Hot-rolled Steel Strip Surface by using Convolutional Auto-Encoder and Conventional Image processing*. 2019, pp. 1-5.
- [50] M. Niu, K.-C. Song, L. Huang, q. wang, Y. Yan, and Q. Meng, "Unsupervised Saliency Detection of Rail Surface Defects Using Stereoscopic Images," *IEEE Transactions on Industrial Informatics*, vol. PP, pp. 1-1, 06/23 2020, doi: 10.1109/TII.2020.3004397.
- [51] G.-W. Kang and H.-B. Liu, *Surface defects inspection of cold rolled strips based on neural network*. 2005, pp. 5034-5037 Vol. 8.
- [52] L. A. O. Martins, F. L. C. Pádua, and P. E. M. Almeida, "Automatic detection of surface defects on rolled steel using Computer Vision and Artificial Neural Networks," in *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, 7-10 Nov. 2010 2010, pp. 1081-1086, doi: 10.1109/IECON.2010.5675519.
- [53] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, *MixMatch: A Holistic Approach to Semi-Supervised Learning*. 2019.
- [54] Y. He, K.-C. Song, H. Dong, and Y. Yan, "Semi-supervised defect classification of steel surface based on multi-training and generative adversarial network," *Optics and Lasers in Engineering*, vol. 122, pp. 294-302, 11/01 2019, doi: 10.1016/j.optlaseng.2019.06.020.
- [55] J. Haugeland, *Artificial intelligence: the very idea*. Massachusetts Institute of Technology, 1985.
- [56] P. H. Winston, *Artificial intelligence (3rd ed.)*. Addison-Wesley Longman Publishing Co., Inc., 1992.
- [57] M. Abadi, "Réalisation d'un réseau de neurones "SOM" sur une architecture matérielle adaptable et extensible à base de réseaux sur puce "NoC". (Neural Network Implementation on an Adaptable and Scalable Hardware Architecture based-on Network-on-Chip)," 2018. [Online]. Available: <https://tel.archives-ouvertes.fr/tel-01868313>
- [58] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115-133, 1943/12/01 1943, doi: 10.1007/BF02478259.
- [59] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386-408, 1958, doi: 10.1037/h0042519.
- [60] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: perceptron, Madaline, and backpropagation," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415-1442, 1990, doi: 10.1109/5.58323.
- [61] M. Minsky and S. Papert, *Perceptrons*. 1969.

- [62] T. Kohonen, "Kohonen, T.: Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics* 43(1), 59-69," *Biological Cybernetics*, vol. 43, pp. 59-69, 01/01 1982, doi: 10.1007/BF00337288.
- [63] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554-2558, 1982/04/01 1982, doi: 10.1073/pnas.79.8.2554.
- [64] G. E. Hinton and T. J. Sejnowski, "Optimal perceptual inference," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 1983, vol. 448: Citeseer, pp. 448-453.
- [65] A. Khebli, "Représentations D'images Pour La Recherche Et La Classification D'images " PhD, Automatique Appliquée et Traitement du Signal, Université M'hamed Bougara - Boumerdes ALGERIA, 2021. [Online]. Available: <http://dlibrary.univ-boumerdes.dz:8080/handle/123456789/6951>
- [66] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303-314, 1989-12-01 1989, doi: 10.1007/bf02551274.
- [67] Andryi and Borokov, "Image Classification with Deep Learning," master, Fakultät für Mathematik, Informatik und Naturwissenschaften, universität hamburg, 2017. [Online]. Available: <https://www.physik.uni-hamburg.de/en/iexp/theses-pp-and-dd/pdfs/masterarbeit-andriy-borovkov.pdf>
- [68] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," *Under Review of ICLR2016 (1997)*, 11/23 2015.
- [69] F. Rosenblatt, *The Perceptron, a Perceiving and Recognizing Automaton Project Para.* Cornell Aeronautical Laboratory, 1957.
- [70] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533-536, 1986-10-01 1986, doi: 10.1038/323533a0.
- [71] C. Bishop, "Pattern Recognition and Machine Learning," vol. 16, 2006, pp. 140-155.
- [72] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [73] V. Veeramsetty, G. Singal, and T. Badal, "Coinnet: platform independent application to recognize Indian currency notes using deep learning techniques," *Multimedia Tools and Applications*, vol. 79, 08/01 2020, doi: 10.1007/s11042-020-09031-0.
- [74] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," 02/10 2015.
- [75] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193-202, 1980/04/01 1980, doi: 10.1007/BF00344251.
- [76] D. E. Rumelhart, G. E. Hinton, R. J. Williams, D. E. Rumelhart, J. L. McClelland, and P. R. Group, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing, Volume 1: Explorations in the Microstructure of Cognition: Foundations*: The MIT Press, 1986, p. 0.

- [77] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998, doi: 10.1109/5.726791.
- [78] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278-2324, 12/01 1998, doi: 10.1109/5.726791.
- [79] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Neural Information Processing Systems*, vol. 25, 01/01 2012, doi: 10.1145/3065386.
- [80] S. Shreya, C. Priya, and G. Rajeshware, *Design of machine vision system for high speed manufacturing environments*. 2016, pp. 1-7.
- [81] J. Järvinen and J. Rauhamaa, "Real-time inspection of steel strip."
- [82] M. Yazdchi, A. Mahyari, and A. Nazeri, *Detection and Classification of Surface Defects of Cold Rolling Mill Steel Using Morphology and Neural Network*. 2009, pp. 1071-1076.
- [83] M. Popat and S. Barai, "Defect Detection And Classification Using Machine Learning Classifier," 01/01 2004.
- [84] J. Yun, D. Jung, and C. Park, "Surface Defect Inspection System for Hot Slabs," *Journal of Institute of Control, Robotics and Systems*, vol. 22, pp. 627-632, 08/01 2016, doi: 10.5302/J.ICROS.2016.16.0048.
- [85] G. Wu, H. Kwak, S. Jang, K. Xu, and J. Xu, *Design of Online Surface Inspection System of Hot Rolled Strips*. 2008, pp. 2291-2295.
- [86] T. Sasaki, H. Takada, and Y. Tomura, "Automatic surface inspection system for tin mill black plate (TMBP)," pp. 60-63, 03/01 2007.
- [87] T. Mäenpää, "Surface quality assessment with advanced texture analysis techniques," 01/01 2006.
- [88] Verlag Stahleisen GmbH. www.stahleisen.de (accessed 07/02/2024, 2024).
- [89] K. Song and Y. Yan. *Steel Surface: NEU-CLS*,
- [90] S. Li, C. Wu, and N. Xiong, "Hybrid Architecture Based on CNN and Transformer for Strip Steel Surface Defect Classification," *Electronics*, vol. 11, p. 1200, 04/09 2022, doi: 10.3390/electronics11081200.
- [91] E. Bisong, "Google Colaboratory," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, E. Bisong Ed. Berkeley, CA: Apress, 2019, pp. 59-64.
- [92] {TensorFlow}: *Large-Scale Machine Learning on Heterogeneous Systems*. (2015). [Online]. Available: tensorflow.org
- [93] Keras. *GitHub*. (2015). *GitHub*. [Online]. Available: <https://github.com/fchollet/keras>. 2015
- [94] B. Zoph, V. Vasudevan, J. Shlens, and Q. Le, "Learning Transferable Architectures for Scalable Image Recognition," 07/21 2017.
- [95] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, *ImageNet: a Large-Scale Hierarchical Image Database*. 2009, pp. 248-255.
- [96] H. Ide and T. Kurita, *Improvement of learning for CNN with ReLU activation by sparse regularization*. 2017, pp. 2684-2691.
- [97] Y. Kateb, A. Khebli, H. Megloul, S. Aguib, and M. S. Khelifi Touhami, "Classifying Surface Fault in Steel Strips Using a Customized NasNet-Mobile

- CNN and Small Dataset " (in English), Journal of Electrical Systems, vol. 20, no. 1, pp. 52-67, 2024.
- [98] I. Konovalenko, P. Maruschak, V. Brevus, and O. Prentkovskis, "Recognition of Scratches and Abrasions on Metal Surfaces Using a Classifier Based on a Convolutional Neural Network," *Metals*, vol. 11, no. 4, doi: 10.3390/met11040549.
- [99] P. Damacharla, V. Achuth, J. Ringenberg, and A. Javaid, TLU-Net: A Deep Learning Approach for Automatic Steel Surface Defect Detection. 2021, pp. 1-6.
- [100] W. Zeng, Z. You, M. Huang, Z. Kong, Y. Yu, and X. Le, Steel Sheet Defect Detection Based on Deep Learning Method. 2019, pp. 152-157.