People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University of 8 May 1945-Guelma-

Faculty of Mathematics, Computer Science and Science of Matter
Department of Computer Science

Master Thesis

Specialty: Computer Science

Option:

Science and technology of information and communication

Theme

Automated Classification of Noise in Historical Document Images

Presented by: Wala Salah Eddine Bouregba

Jury Members:

N	Full name	Quality
1	Chemesse Ennehar BENCHERIET	Chairman
2	Abderrahmane KEFALI	Supervisor
3	Hassina BOURSSACE	Examiner

June 2024.

ملخص

رقمنة الوثائق التاريخية أمر بالغ الأهمية للحفاظ على السجلات القيمة للأجيال القادمة. ومع ذلك، فإن الحصول على نسخ رقمية عالية الجودة من هذه الوثائق يشكل تحديًا بسبب حالتها المتدهورة في كثير من الأحيان، التي تتميز بانخفاض التباين والآثار التالفة الناتجة عن التقدم في العمر والعوامل البيئية والتعامل. تجعل هذه التدهورات من الصعب قراءة الوثائق التاريخية واستخدامها بشكل فعال. تركزت الأبحاث الحديثة على استعادة وتحسين جودة هذه الوثائق المتدهورة لضمان بقائها متاحة وقابلة للاستخدام.

يعد التصنيف الدقيق لأنواع الضوضاء في الوثائق أمرًا ضروريًا لتطبيق طرق الاستعادة الصحيحة لتحويل الوثائق إلى وثائق خالية من الضوضاء. تتطلب الأنواع المختلفة من الضوضاء تقنيات معالجة مسبقة محددة للإزالة الفعالة، مما يجعل تحديد الضوضاء بدقة خطوة حاسمة في عملية الاستعادة.

يهدف عملنا إلى بناء نموذج تصنيف آلي قوي يحدد بدقة نوع الضوضاء في صور الوثائق. من خلال استغلال قدرات التعلم والتعميم للشبكات العصبية الالتفافية (CNNs)، يصنف نموذجنا خمسة أنواع رئيسية من الضوضاء: "صورة نظيفة، تلف الورق، الشفافية، الصورة الباهتة، والبقع". يتيح التصنيف الدقيق اختيار الطريقة المناسبة لإزالة الضوضاء. يُحسن هذا التصنيف بشكل كبير من كفاءة وفعالية عمليات استعادة الوثائق. تسلط النتائج الواعدة من تجاربنا الضوء على إمكانيات التعلم العميق في تحسين جودة الوثائق التاريخية الرقمية وضمان سهولة الوصول إليها على المدى الطويل.

كلمات مفتاحية: تصنيف التدهور، صور الوثائق التاريخية، الضوضاء، شبكات الأعصاب العصبية التصنيفية، تحليل صور الوثائق.

Abstract

The digitalization of historical documents is crucial for preserving valuable records for future generations. However, obtaining high-quality digital versions of these documents is challenging due to their often degraded state, characterized by low contrast and corrupted artifacts resulting from aging, environmental factors, and handling. These degradations make it difficult to read and utilize historical documents effectively. Recent research has focused on restoring and improving the quality of these degraded documents to ensure they remain accessible and usable.

Accurate classification of noise types in documents is essential for applying the correct restoration methods to change the documents to a noise free document. Different types of noise require specific preprocessing techniques for effective removal, making precise noise identification a critical step in the restoration process.

Our work aims to build a robust automated classification model that accurately identifies the type of noise in document images. By leveraging the learning and generalization capabilities of Convolutional Neural Networks (CNNs), our model classifies five main noise types 'clean image, paper damage, transparency, faded image, spots', accurate classification enables the selection of appropriate noise removal method. This classification significantly improves the efficiency and effectiveness of document restoration processes. The promising results from our experiments highlight the potential of deep learning in enhancing the quality of digitized historical documents and ensuring their long-term accessibility.

Keywords: degradation classification, historical document images, noise, CNN, document image analysis.

Acknowledgments

First and foremost, Alhamdulillah for his blessings and grace and for blessing me with good health and gifts through this long journey.

I am profoundly indebted to my beloved family, especially my mom and dad, for their unwavering emotional and financial support. Their encouragement and sacrifices have been the cornerstone of my success, and for that, I am eternally grateful.

I am indebted to my esteemed supervisor Dr. Abderrahmane Kefali for his invaluable guidance and profound wisdom throughout this endeavor. His mentorship has been instrumental in shaping my understanding and refining my work.

Table of Contents

List of Figures

List of Tables

General Introduction

The digitization of documents has become essential for preserving valuable document especially historical and archival records, ensuring they remain accessible for future generations. However, this process poses numerous challenges. These challenges can result in degraded or noisy document images, as digital degraded image by some types of digital noise such as blur, poor color quality, and salt-and-pepper artifacts frequently emerge during the digitization process itself. Beyond these digital challenges, physical degradations also affect documents over time. Factors like aging, moisture damage, and physical trauma contribute to physical degraded papers by noise types such as stains, tears, and faint text. The integrity of the document can be significantly compromised by its initial condition, environmental factors, and handling practices during digitization.

Despite the advancements in digital technology, the primary issue lies in the insufficiency of data that accurately represents the variety and complexity of noise types found in document images. This lack of comprehensive data hampers the development of effective noise classification systems. The noises we will take into consideration for our work include paper damage, clean image, transparency, faded text, and spots.

Our primary objective is to develop a robust noise classification system capable of accurately identifying and classifying the most significant types of noise affecting historical document images. This classification not only aids in improving the accuracy of document processing tasks but also helps in preserving and restoring historical documents by identifying the prevalent types of degradation. Accurate noise classification enables more precise and effective document processing, as specific types of noise can be addressed with tailored preprocessing steps. Moreover, identifying and classifying different noise types can significantly aid in developing targeted strategies for document conservation. Ultimately, our work aims to create a reliable tool that supports the ongoing efforts to maintain the integrity and accessibility of historical documents, ensuring their longevity and usability for future generations.

This thesis is structured into four main chapters.

Chapter 1: Noise and Noise Classification in Document Images

This chapter introduces the key concepts and definitions related to noise and degradation in document images. It presents an overview of relevant literature concerning noise classification in document images, highlighting various approaches and methodologies used to classify and mitigate noise in digitized documents.

Chapter 2: Neural Networks

This chapter explores the fundamentals of neural networks, detailing their structure, activation functions, and architectures. It emphasizes the role of neural networks in processing and classifying images, highlighting both their advantages and limitations.

Chapter 3: Conception

This chapter explores the conceptual framework of our study, detailing our objectives, the types of noise considered in our work, and the datasets utilized. It describes the methodology we employed, with a specific emphasis on integrating diverse models to enhance classification accuracy.

Chapter 4: Implementation and Results

The practical implementation of the proposed noise classification system is presented in this chapter along with the details of experiments carried out on different models of deep learning to assess their performance. The experiments results, including model examinations as well as performance assessments, are analyzed in detail in this chapter.

Chapter 1: Noise and Noise Classification in Document Images

1. Introduction

Document digitization offers a contemporary solution for preserving documents, extending their longevity, and safeguarding them from damage. It enhances accessibility, facilitating ease of use for researchers. However, the digitization process can encounter various challenges, resulting in corrupt, degraded, or noisy outcomes. Yet, beyond these digital challenges lie the physical degradations that documents may face over time. These physical deteriorations, such as aging, moisture damage, or physical trauma, significantly impact the document's integrity. Factors contributing to physical degradation include the document's initial condition, environmental factors, and handling practices during digitization. Understanding and addressing both digital and physical degradation are paramount for ensuring the long-term preservation and usability of documents.

In this chapter, we commence by defining "document," "noise," and "Degradation". Subsequently, we explore noise categories and their respective types. Finally, we delve into noise classification and denoising techniques.

2. Term definition

In this section, we aim to provide a comprehensive understanding by introducing key definitions essential to grasp the concepts discussed within this domain. These definitions serve as foundational knowledge, elucidating fundamental terms pivotal for navigating through the subject matter effectively.

2.1. Document

Suzanne Briet [BRI06] document definition is “A document is any material, visual, or digital entity that is intentionally created or produced to convey a representation, a message, or information, with the purpose of being conserved, communicated, or utilized as a reference.”

2.2. Noise

Image noise refers to the unwanted random variations of brightness or color information in images. These variations can result from the electronic sensor and circuitry of a camera, environmental conditions, or the inherent characteristics of the imaging process. Common types of image noise include Gaussian noise, salt-and-pepper noise, and Poisson noise [GON02].

2.3. Degradation

F. DRIRA [DRI07] defines a degradation as any cumulative unfavorable impacts that make it more difficult to read, process, or preserve images, referred to as degradation. There are several sources of degradation, and it is challenging to distinguish between them due to the buildup of flaws. Compared to other so-called "natural" images, document images are more severely damaged by degradation. Preserving it’s historic significance and integrity are utilized in noise reduction efforts.

3. Noise categories

It's important to note that there isn't a single, universally accepted categorization of noise types. Therefore, it would be prudent to mention the used classifications along with their respective references.

RABEUX V [RAB13] proposed two categories of degradation: Degradations relative to the original documents and degradations resulting from poor digitization.

DRIRA F [DRI07] established two classes of degradation: *Uniform degradation*, these are degradations that affect the entire document uniformly, and *non-uniform degradation*: refer to all alterations that vary depending on the position within the image.

The several types of noise in document images will be shown in the following section, and they will be divided into two groups: noise resulting from the document digitalization process we will call **digital noise** and noise resulting from physical document degradation we will refer to as **physical noise**.

3.1. Digital noise types

RABEUX V [RAB13] defined poor digitization as:" We speak of poor digitization when the resulting document image is not sufficiently faithful to the original page. This can stem from incorrect settings (focus, white balance, color adjustment, etc.), mishandling (page movements during scanning, improper placement of the glass, etc.), changes in external scanner parameters (such as changes in room lighting), or a mismatch between the scanner's capabilities and the type of document." There are various types of digital noise that affect ancient documents during the numeration process. We present here some types mentioned by RABEUX V [RAB13], FIRAS AJ [JAS13] and ZIAD A [QAD18] :

3.1.1. Document's aperture

The document's aperture refers to the maximum angle between two pages. When the binding is too tight, it becomes difficult to lay the document flat. This can lead to focusing issues and black areas in the binding. There are also scanners capable of reaching into the margins of the document. [RAB13]

3.1.2. Blur

[RAB13] stated that blur can arise from several suboptimal procedures or configurations. The scanning operator, for instance, adjusts the focus on manual scanners. But certain ancient books are quite heavy, and after scanning a certain number of pages, the scanner needs to be readjusted. The entire page exhibits the blur defect. In other situations, the blur is localized because stemming from deep binding or movements of the document during scanning.

3.1.3. Poor color quality

Incorrect scanner settings can lead to documents with poor color quality (white balance, color chart) [RAB13]. It is frequently necessary to modify these settings in response to outside events, including variations in ambient illumination.

3.1.4. Inclination

The document's orientation may not be respected, resulting in an inclination in the resulting scanned image [RAB13]. The inclination arises from improper positioning of the document on the scanner glass.

3.1.5. Marginal noise

Marginal noise in document images refers to the unwanted artifacts or distortions that appear along the edges or margins of a document. These noises can be introduced during the digitization process, such as scanning or photocopying, and may include dark borders, light streaks, or other irregularities that obscure the true content of the document [SPI97].

3.1.6. Salt-and-pepper noise

Salt-and-pepper noise, characterized by sporadic white and black pixels, occasionally appears in images [QAD18]. This type of noise typically arises from sensor defects in cameras, software failures, or hardware issues during image capture or transmission [JAS13].

Figure 1.1 provides an example of each digital noise.

3.2. Physical noise

The physical noise category is a degradation that encompasses all defects that primarily stem from the document's condition. In most cases, these degradations cannot be mitigated [RAB13]. Here are some types of physical noise:

3.2.1. Show-through (bleed through or back-to-front interference)

On excessively thin papers, the ink on the back is visible through the front. The transparency can show through or not, depending on how much ink seeps through [RAB13].

3.2.2. Stains

While moisture is a common cause of stains, incorrect handling can also lead to them [RAB13].

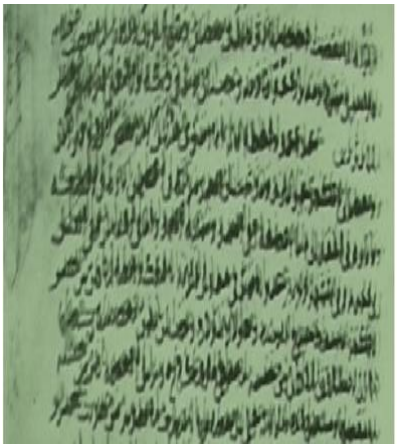
3.2.3. Tears and holes

Tears and holes are less frequent degradations that can arise from a variety of sources, include wear and tear, misuse, insects, etc. [RAB13].

3.2.4. Shape Degradations

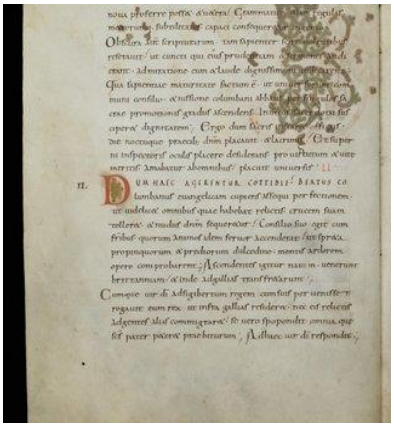
We refer to shape degradations as those that alter the continuity of lines and the topology of objects. Different types of degradations may appear on characters: breaks in lines, truncated

characters, gaps in lines, character mergers, stair-step effect... Additionally, the appearance of transparency effects or holes may render certain characters unreadable. These degradations significantly impair text readability and automatic shape recognition [DRI07].

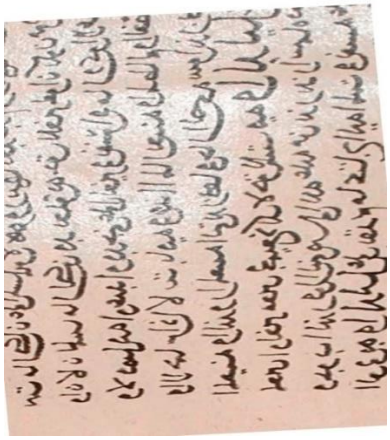


(a) Blurred paper

(b) poor color quality;



(c) marginal noise



(d) Wrong Orientation

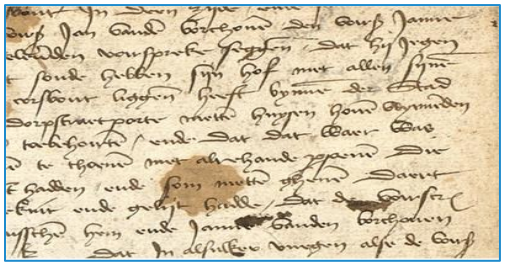
(e) salt and pepper noise;

Figure 1.1. Examples of degradations related to the digitization process.

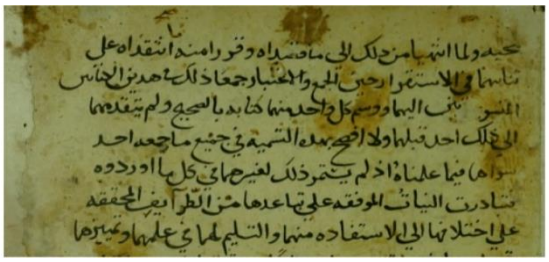
3.2.5. Faint Text and Low Contrast

A document image with faded or faint text is called a faint-text image. The text may be missing or corrupted in this state. Furthermore, this deterioration produces a low-contrast image where the text and backdrop have very little contrast. The writing becomes difficult to read against the background due to this kind of degradation; As a result, recovering text from such degradation will pose a significant challenge. This type of deterioration complicates the differentiation between text and background, thereby hindering the extraction process [ARN21].

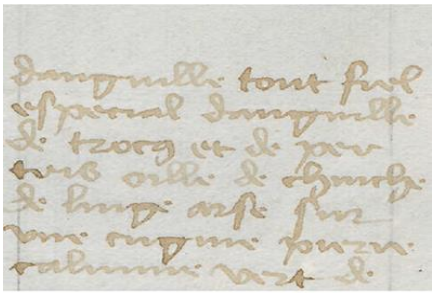
Figure 1.2 illustrates an example of each type of physical noise.



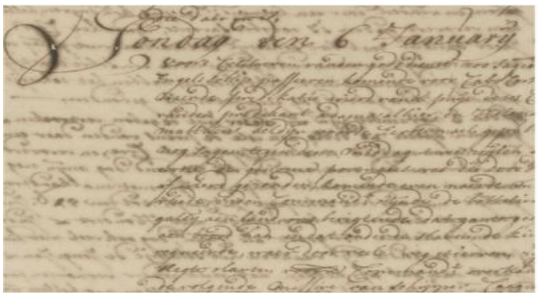
(a) Stains



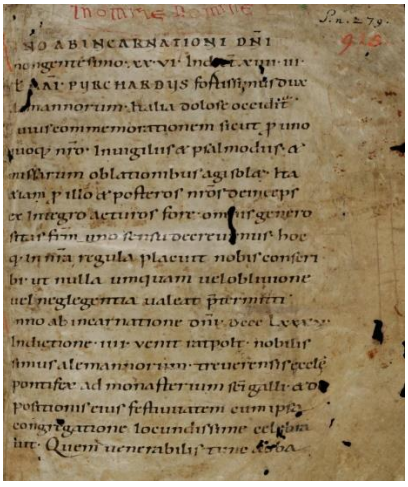
(b) Shape degradation



(c) Fainted text



(d) Show-through



(e) Tears and holes

Figure 1.2. Examples of degradations affecting old documents.

4. Noise identification

In anticipation of a comprehensive review of literature on noise identification in document images, it's prudent to first establish a robust conceptual framework and underscore its practical importance. This section aims to provide an essential exploration into the significance of noise identification, highlighting its pivotal role in advancing methodologies for document image analysis. By elucidating this concept, the groundwork is laid for a deeper understanding of its relevance and application within the field.

Despite its importance, the field of noise identification in document images has received limited attention. In this section, we provide an overview of significant works in noise identification tailored specifically for document images.

Some works focus solely on detecting specific types of noise within images, while others aim to classify images based on the types of noise present, categorizing them into distinct classes.

The works presented here are categorized into two main approaches: classical methods and machine learning methods.

4.1. Classical methods

Classical methods rely on traditional algorithms and heuristic techniques for noise detection. These methods often involve the application of well-established signal processing techniques, such as median filtering, Fourier analysis, or morphological operations. Classical methods are typically designed to detect specific types of noise based on predefined characteristics or thresholds. They may struggle with complex or variable noise patterns and may require fine-tuning for optimal performance.

4.1.1. Fitri Arnia Method (Noise Characterization in Ancient Document Images Based on DCT Coefficient Distribution)

[ARN15] introduced a novel method for classifying degraded images, which utilizes the discrete cosine transform coefficient distribution (DCT coefficient distribution). The images used in this method are sourced from Acehnese ancient manuscripts that have undergone digitalization and preservation. The noise-containing areas within these images are standardized to dimensions of 256×256 pixels, resulting in the generation of 46 cropped images. Ground-truthing involves the localization and cropping of noisy areas, followed by binarization using appropriate techniques. DCT coefficient distributions are then computed from both the cropped noisy and clean images. By analyzing the standard deviation of these distributions, three distinct noise types are identified: *fox*, *background spots*, and *uneven background*. This method offers a unique perspective on image classification, leveraging DCT coefficients to distinguish various types of noise within historical document images.

Figure 1.3 illustrates the global architecture of this method.

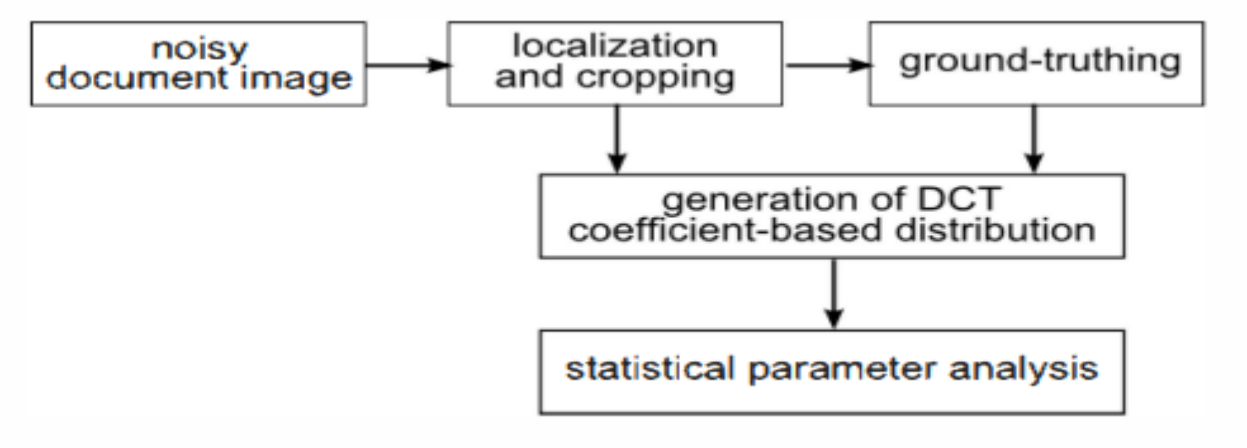


Figure 1.3. General architecture of the Fitri Arnia method [ARN15]

4.1.2. Maryam Shamqoli Method (Border Detection of Document Images Scanned from Large Books):

The method outlined in [SHA13] is designed to identify and eliminate dark borders from document images. The algorithm (as shown in Figure 1.4) begins by applying an adaptive Wiener filter to the grayscale input images to enhance their quality. Subsequently, an edge detection process is performed using the Prewitt operator to highlight the borders. Horizontal and vertical histograms are then computed to detect noisy black borders on the upper, lower, left, and right sides of the image. Any pixels outside the detected bounds are removed in the initial noise removal procedure. Finally, the median of a limited histogram and quadratic Detection techniques are employed to refine the detection of bounds for improved accuracy. The method was evaluated using 159 document images from Persian books, with 150 images found to be noise-free and 9 yielding unsatisfactory results. This approach offers a systematic method for detecting and removing dark borders from document images, contributing to improved image quality and readability.

4.1.3. Tan lee Method (Bayesian damage recognition in document images based on a joint global and local homogeneity model)

Tan Lee [TAN21] presents an innovative method for detecting damages in document images, emphasizing Textural Homogeneity (TH) patterns' importance in identifying regularities. This method intricately explores homogeneity measures, crucial for discerning anomalies. It adopts a dual perspective: global and local. Globally, it constructs a neighborhood graph model based on connected components (CCs) from the document image. Weights, derived from probabilistic homogeneity measures between neighboring CCs, reveal a novel global homogeneity measure named Probabilistic Text Homogeneity (PTH), aiding in pinpointing potential damage areas. Locally, it introduces a pixel-wise approach termed Propagation of Wavelet Approximation (PWA), adept at capturing neighborhood transition volatility due to damages via wavelet multi-resolution analysis. The fusion of global and local measures through Bayesian methods results in

a sophisticated framework for probabilistic recognition of damaged pixels. This framework not only ensures robust detection but also facilitates precise classification of damaged regions within document images, significantly enhancing damage assessment efficiency and accuracy.

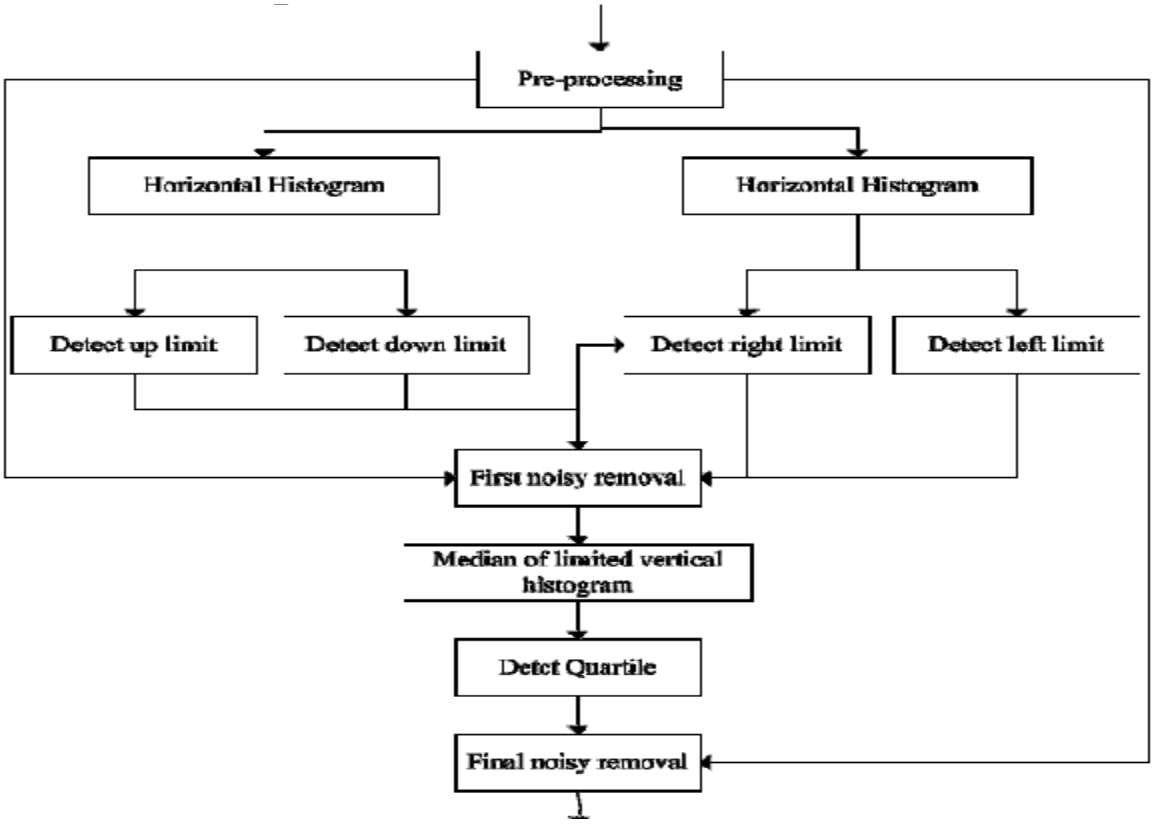


Figure 1.4. General architecture of the Maryam Shamqoli method [SHA13]

4.1.4. Ben mesaud method (New method for the selection of binarization parameters based on noise features of historical documents)

In [BEN11], Ben Messaoud proposed an approach for detecting and classifying image noise, aimed at developing an automated parameterization methodology for binarization techniques. The approach is tailored to identify three main noise classes: transparency effect, foreground/background similarity, and variable background. A fourth class is also introduced to account for other types of noise or the absence thereof. Detection of the first two noise classes relies on analyzing the image's grayscale histogram. The transparency effect is identified when the histogram shows two peaks in the background region. If the distance between these peaks is less than a predetermined threshold, it signifies similarity between the foreground and background. Lastly, by examining local variances across multiple windows, the presence of a variable background in the document can be inferred.

4.2. Machine learning methods

Machine learning methods utilize automated classifiers to perform image classification based on learned patterns. These methods use algorithms that can autonomously learn from labeled data to identify and classify noise within document images. Machine learning approaches offer the advantage of adaptability and scalability, as they can effectively handle diverse and evolving noise patterns without the need for explicit rule-based programming. By training on large datasets, machine learning models can learn intricate patterns and nuances in document images, leading to more accurate and robust noise identification.

4.2.1. Lins Method (Automatically Detecting and Classifying Noises in Document Images)

Rafael D. Lins [LIN10] proposed an automated classifier capable of detecting and classifying noise into four classes: *Orientation noise*, *Border noise*, *Back-to-front noise*, and *Skew noise*. For the latter, Lins divided the Back-to-Front noise further into three sub-classes as it is shown is Figure 1.5: *Strong Back-to-Front noise*, *Medium Back-to-Front noise*, and *Weak Back-to-front noise*. The classifier utilized in this study is *Random Forest*, implemented using Weka, an open-source tool for statistical analysis developed at the University of Waikato, New Zealand. The dataset comprised 69,935 images, with 64,584 images allocated for testing and 5,351 images for training. Each image in the dataset underwent feature extraction, with a total of nine features considered crucial for the classifier's success. The accuracy of Lins' method reached 96%; however, the detection of upside-down documents accounted for most of the incorrect orientations observed in the classifier's results.

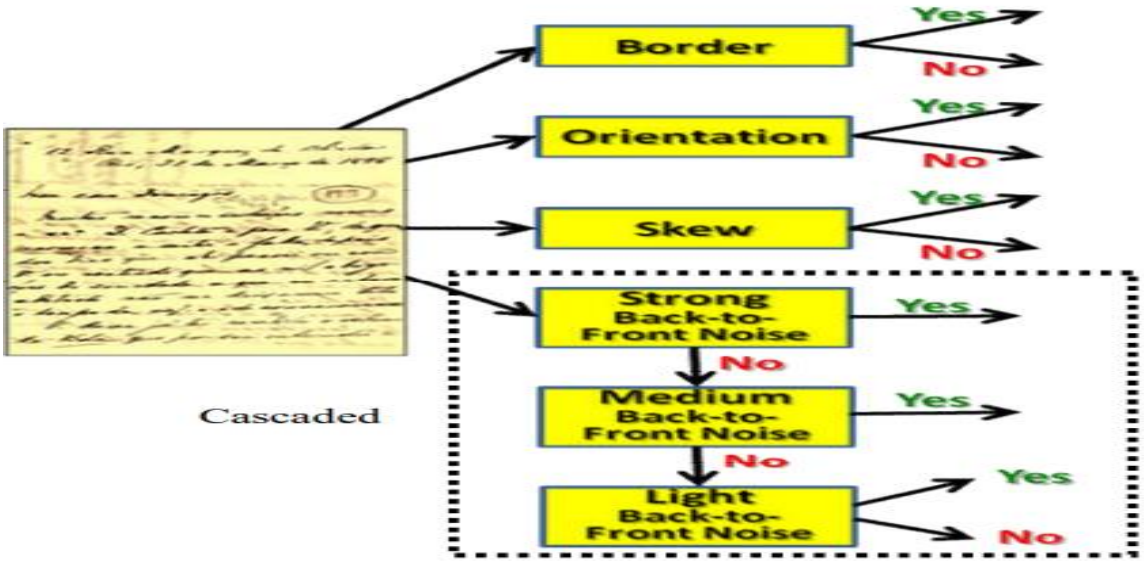


Figure 1.5. General architecture of the Rafael D. Lins method [LIN10]

4.2.2. Pawar p Method (Deep Learning Approach for the Detection of Noise Type in Ancient Images)

Pawar's method [PAW22] employs deep learning to detect and categorize four types of digital noise: Gaussian noise, impulse noise, Poisson noise, and speckle noise. This approach utilizes the Indian Heritage Image Retrieval Dataset, using its images as reference points. To simulate different types of noise, such as impulse, Gaussian, speckle, and Poisson noise, these reference images are intentionally corrupted with noise variations.

A Convolutional Neural Network (CNN) architecture is chosen for its high output capacity. The CNN takes a colored-labeled reference image as input, and the wavelet transform (WT) extracts features such as energy, mean, homogeneity, contrast, correlation, variance, skewness, kurtosis, and entropy. These features are then reduced to decrease system complexity. The reduced features serve as input for an 11×11 convolution layer to enhance classification accuracy. The output from this layer goes to 3×3 pooling layers using the max pooling technique to preserve prominent features. The final CNN layers, fully connected, categorize features extracted by the convolutional and pooling layers. These feature maps are flattened into a single 1D vector before being input into the neural network for classification. Figure 1.6 illustrates the architecture of the used CNN:

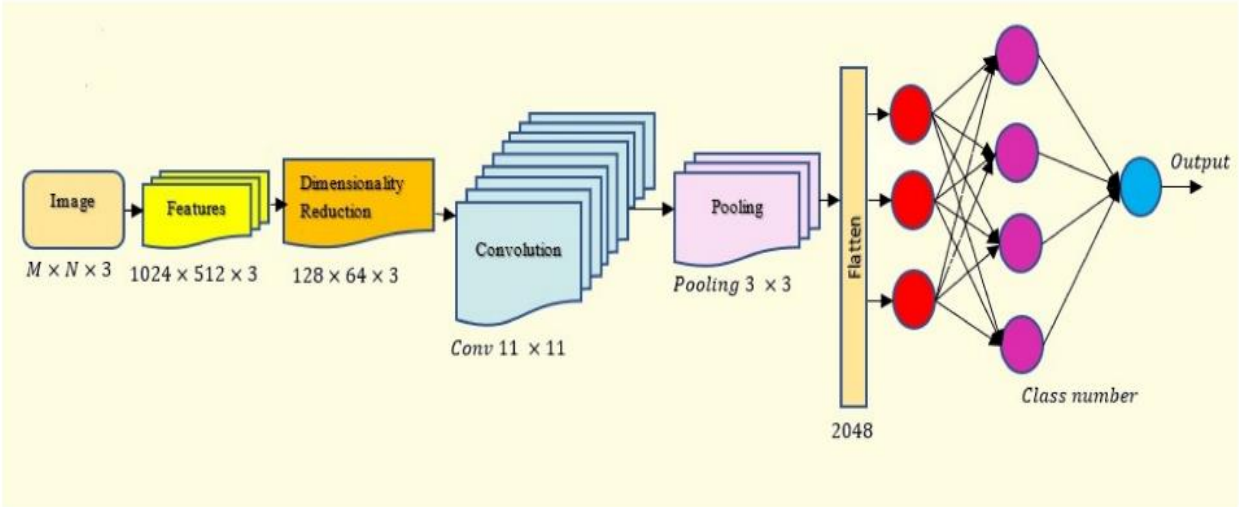


Figure 1.6. General architecture of the Pawar p method [PAW22]

4.2.3. Fitri Arnia Method (DCNet: Noise-Robust Convolutional Neural Networks for Degradation Classification on Ancient Documents)

Fitri Arina [ARN21] proposed leveraging a deeper neural network architecture to tackle the classification challenge posed by ancient documents. Generating 30,000 image patches sized at 224 × 224 pixels, [ARN21] sourced noisy images from the DIBCO dataset [GAT09, PRA17], the PHIBD dataset [AYA13], and the private JAWI dataset [SAD15, SAD17]. From these datasets, four types of image deterioration were identified, namely *bleed-through/show-*

through/ink-bleed, *faint-text and low contrast*, *smear-spot-stain*, and *uniform degradation*. Initially, the input image is trained in parallel using MobileNetV2 and ShuffleNet blocks, known for their efficiency and accuracy in image categorization [MA18, XIE17]. These blocks are concatenated and activated through the SWISH activation layer. Following this, a ResNet block stage is introduced, leveraging residual layers to enhance performance in image classification and object detection [HE16]. The architecture is further augmented with a transition layer proposed by [ARN21], culminating in the final SoftMax and classification layers (see Figure 1.7).

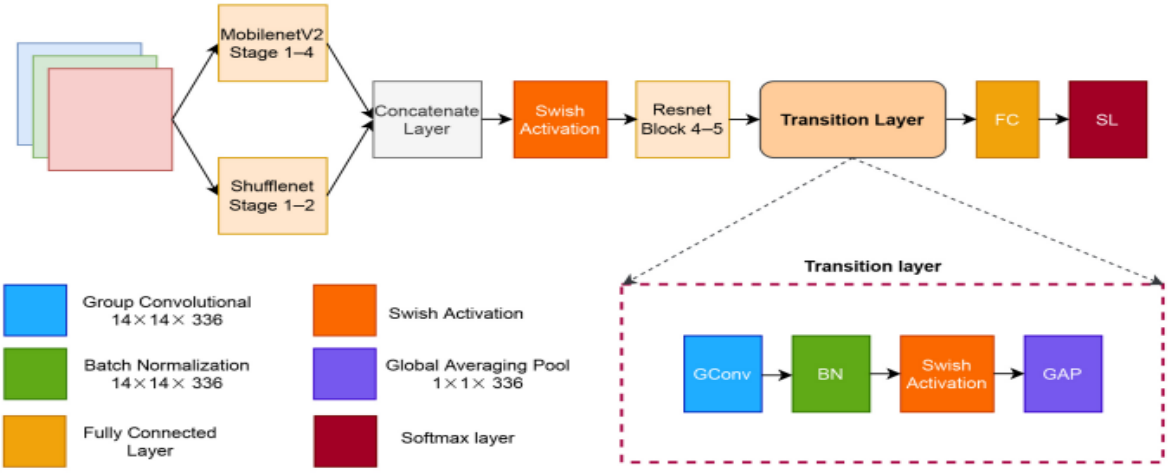


Figure 1.7. General architecture of the Fitri Arina method [ARN21]

4.2.4. Kazuki endo Method (Classifying Degraded Images Over Various Levels of Degradation)

The method proposed by [END20] focuses on identifying the level of degradation and restoring deteriorated images rather than classifying specific types of degraded images. [END20] introduces a network comprising four different types: a restoration network, classification networks, an estimation network of degradation levels, and an estimation network of ensemble weights. To train these networks, [END20] utilized four datasets: yang91 [JIA08], Urban100 [HUA15], General100 [DON16], and CIFAR [KRI12]. The first three datasets were employed to train both the restoration network and the degradation level estimation network. From each image, [END20] generated 64 × 64 patches and enriched the dataset through transpose, horizontal, and vertical flips. These patches were then compressed using the JPEG algorithm. For training the classification and ensemble weight estimation networks, CIFAR datasets were used. CIFAR images underwent data augmentation, including zooming, shearing, horizontal flipping, rotating, and vertical and horizontal shifting, before applying JPEG compression. The [END20] approach leverages degraded images recovered by the restoration network, supplying them to two classification networks: one trained with clean images and another trained with restored images. Additionally, degraded images are input into the network that estimates degradation

levels. Estimated degradation levels are then utilized in an ensemble weight estimation network, where weights range from 0 to 1 and sum to one. The expected probability is computed using weighted averaging. The global outline of this approach is summarized in Figure 1.8.

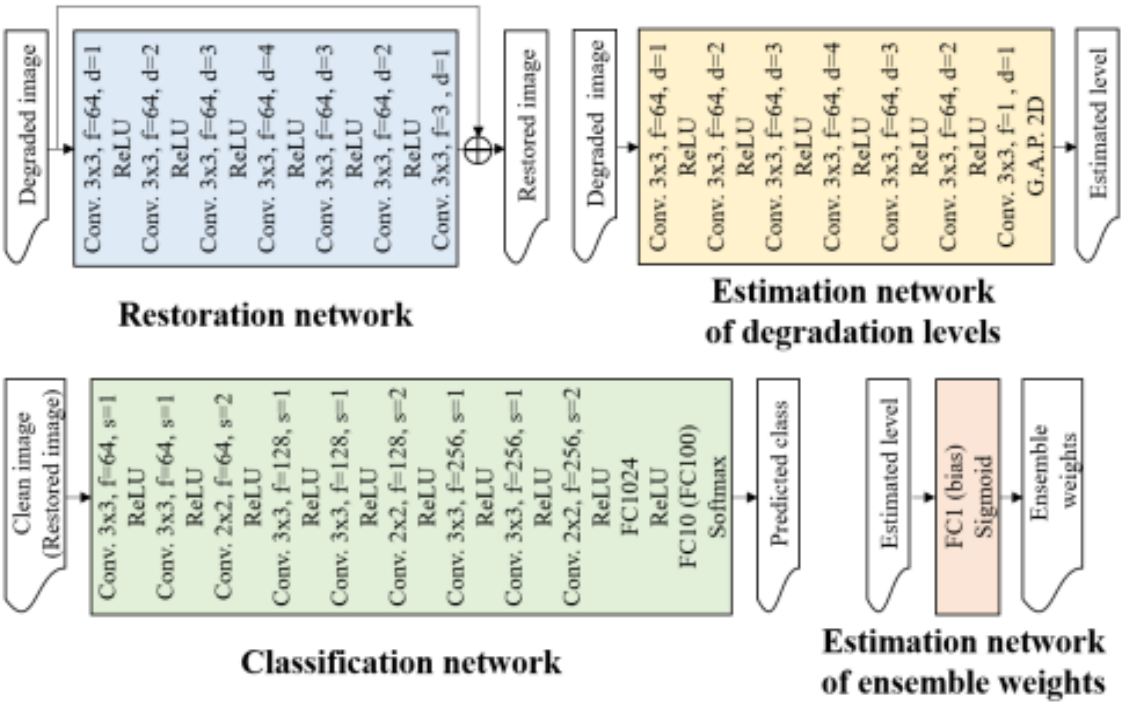


Figure 1.8. General architecture of the Kazuki endo method [END20]

4.2.5. Amal GHOMRASSI Method (Restoration of Ancient Colored Documents: Foreground/ Background Separation)

This method [GHO15] comprises three primary steps:

1) **Initial Separation:** The first step aims to distinguish between the foreground and background of the image. This process begins with converting the RGB image to LAB color space [GHO15], chosen for its capability to separate the luminance and color components effectively. Subsequently, image contrast is enhanced to facilitate foreground-background differentiation.
2) **Secondary Separation:** In the second step, the foreground is further separated from the remaining background or transparency pixels. This phase involves two main stages. Firstly, the initial foreground undergoes a pixel-oriented analysis to extract its features, a process known as Characterization of the initial foreground. Principal Components Analysis (PCA) is then applied to optimize information extraction while minimizing data redundancy and complexity. Secondly, the Foreground/non-foreground classification utilizes a Fuzzy C-Means (FCM) classifier to predict the class of each pixel in the foreground. The labels assigned by the FCM classifier are integrated into a Maximum Likelihood (ML) approach to refine segmentation results by rectifying misidentified pixels.

3) **Reconstruction:** The final step involves reconstructing the restored image by combining the final foreground with the mean of the final background. The resulting image is transformed back to RGB for the ultimate outcome.

The [GHO15] method employs the M-SAGE database as its dataset, consisting of 500 historical sub-images with varying degrees of recto/verso transparency degradation. This comprehensive approach addresses foreground/background separation in historical document images, contributing to the restoration and preservation of cultural heritage.

5. Conclusion

In this chapter, we explored the complexities of noise and degradation in document images. Key terms like "document," "noise," and "degradation" were defined to establish a foundational understanding. We categorized noise into digital (e.g., blur, poor color quality) and physical types (e.g., stains, shape degradation), identifying their origins from digitization processes and the document's condition, respectively. In the rest of the chapter, we discussed two main noise identification methods, classical methods and advanced machine learning approaches.

Chapter 2: Neural Networks and Deep Learning

1. Introduction

Neural networks have emerged as a powerful tool in the field of artificial intelligence, revolutionizing various industries by enabling machines to learn from data and make predictions or decisions. Inspired by the structure and function of the human brain, neural networks consist of interconnected nodes, or neurons, organized into layers. Through a process known as training, neural networks can adjust their parameters based on input data to perform tasks such as classification, regression, and pattern recognition with remarkable accuracy.

This chapter provides an overview of neural networks, covering fundamental concepts, architectures, and applications. Drawing upon a comprehensive array of scholarly sources and references, including seminal works and recent research papers, we delve into the underlying principles of neural network operation and explore state-of-the-art techniques and advancements. By synthesizing insights from a diverse range of literature, we aim to provide readers with a comprehensive understanding of neural networks and their significance in modern computing.

2. What is an artificial Neural network?

According to Margaret Rouse [MAR23], "*An artificial neuron network (neural network) is a computational model that mimics the way nerve cells work in the human brain.*

Artificial neural networks (ANNs) use learning algorithms that can independently make adjustments – or learn, in a sense – as they receive new input. This makes them a very effective tool for non-linear statistical data modeling".

3. Classical Neural networks

3.1. Neural network structure

3.1.1. Formal neuron

Formal neuron is a non-linear and bounded algebraic function, whose value depends on parameters called coefficients or weights.

The variables in this function are commonly referred to as the 'inputs' of the neuron, while the result of the function is termed its 'output.' Thus, a neuron fundamentally acts as a mathematical operator, whose numerical value can be calculated by a few lines of software [DJE17].

Based on observations of biological neurons, the model of the formal neuron was proposed by W. M. Culloch and W. Pitts in 1943. It is illustrated by the following Figure:

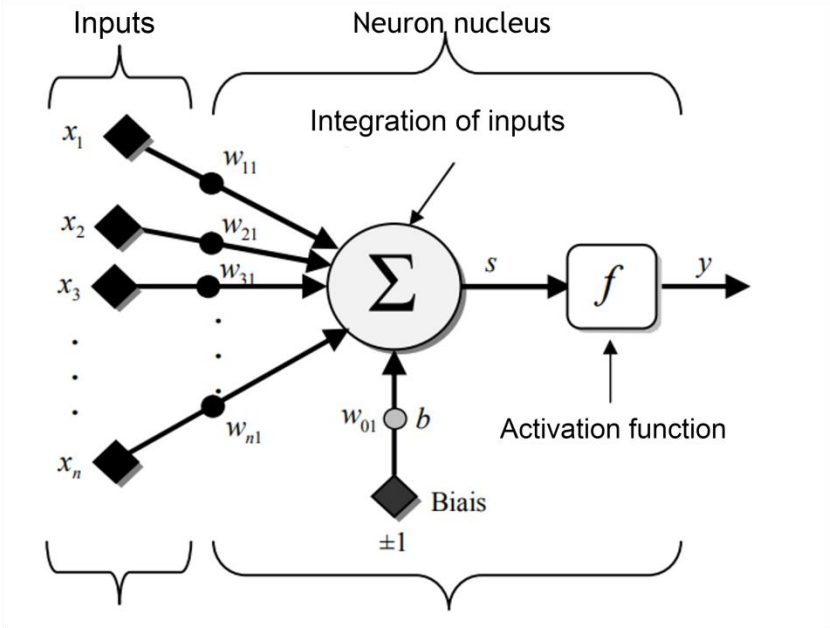


Figure 2.1. General architecture of formal neuron [DJE17]

The x_i represent input vectors, which come either from the outputs of other neurons or from sensory stimuli (visual sensor, auditory sensor, etc.).

The w_{ij} are the synaptic weights of neuron j. They correspond to the synaptic efficacy in biological neurons ($w_{ij} > 0$) excitatory synapse; $w_{ij} < 0$) inhibitory synapse). These weights weigh the inputs and can be modified through learning.

Bias: Input often takes values of -1 or +1, which adds flexibility to the network by allowing the adjustment of the neuron's firing threshold through the adjustment of weights and bias during learning.

Kernel: Integrates all inputs and the bias and calculates the neuron's output according to an activation function, which is often nonlinear to provide greater learning flexibility.

3.1.2. Activation functions

An activation function in a neural network is a mathematical function that determines the output of a neural network node given an input or set of inputs. The activation function introduces non-linearities into the network, enabling it to learn and model complex data patterns. Without activation functions, a neural network would simply be a linear model, limiting its ability to solve non-linear problems [GOO16].

Most commonly used activation functions in neural networks are summarized in the following table:

Function name	Equation
1. ReLU (Rectified Linear Unit)	$f(x) = \max(0, x)$
2. Sigmoid	$f(x) = \frac{1}{(1 + e^{-x})}$
3. Tanh (Hyperbolic Tangent)	$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$

Table 2.1. Some commonly used activation functions

3.1.3. Neural Network Architecture

The neural network's architecture defines how neurons are organized and connected, outlining their arrangement and structure within the network. Essentially, it describes the structure and layout of the network. While most networks use standard neurons, some rare architectures may feature specialized ones. The specific architecture of a neural network is influenced by the task it is designed to learn. Neural networks generally comprise layers of neurons, spanning from input to output. The main types of architectures are feedforward neural networks and recurrent neural networks [DJE17].

a) Feedforward neural networks

Feedforward neural networks perform algebraic functions on their inputs by composing the functions performed by each of their neurons. Graphically, they are represented as a set of neurons "connected" to each other, with information flowing from inputs to outputs without feedback. If we represent the network as a graph where nodes are neurons and edges are the "connections" between them, the graph of a feedforward network is acyclic [DJE17].

a.1) Single-Layer Neural Networks

The structure of a single-layer network is such that neurons organized as input are fully connected to other neurons organized as output through a modifiable layer of weights [DJE17] as shown in Figure 2.2.

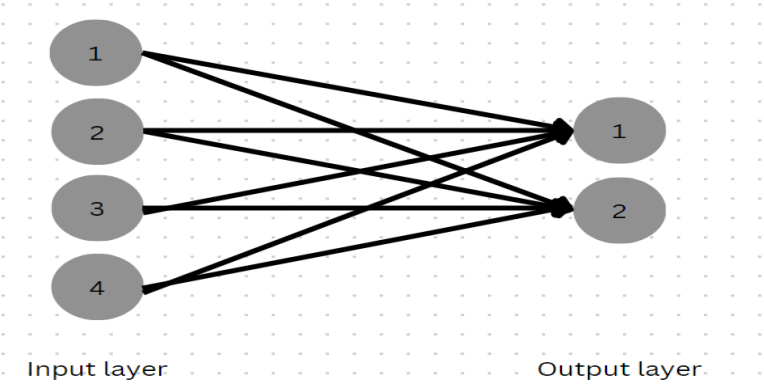


Figure 2.2. General architecture of Single-Layer Neural Network

a.2) Multilayer neural networks

In a neural network, connections between neurons do not exist within the same layer; connections are exclusively made with neurons in subsequent layers. Usually, each neuron in one layer connects to every neuron in the next layer, and only to that layer. This arrangement introduces the concept of the direction of information flow (activation) within the network, defining input and output neurons. The layer containing input neurons is termed the input layer, while the layer containing output neurons is known as the output layer. Layers that mediate between input and output without direct external interaction are referred to as hidden layers.

It is also noted that multilayer networks are much more powerful than single-layer networks. By using two layers (one hidden layer and one output layer), and employing a sigmoid activation function in the hidden layer, a network can be trained to approximate most functions with arbitrary precision, although this may require a large number of neurons in the hidden layer. Except in rare cases, artificial neural networks typically utilize two or three layers [DJE17].

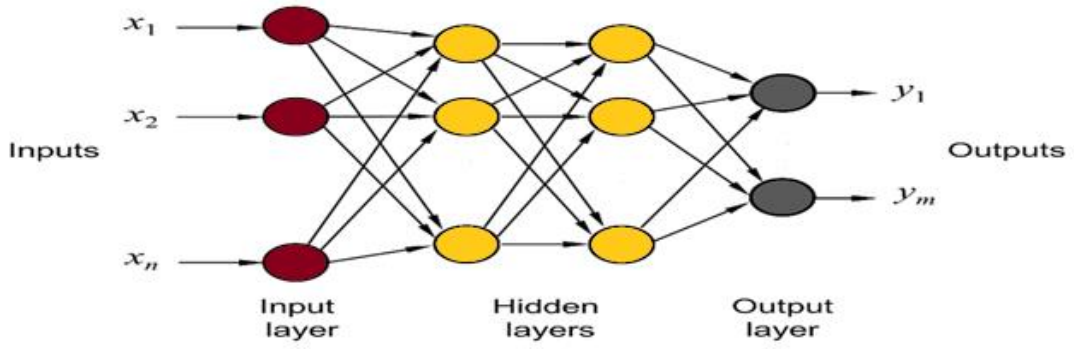


Figure 2.3. General architecture of Multilayer neural networks

b) Recurrent neural networks

Unlike non-looped neural networks, whose connection graph is acyclic, looped or recurrent neural networks can have any connection topology, including loops that feed back the value of one or more outputs to the inputs. For such a system to be causal, each loop must have an associated delay, making a looped neural network a dynamic system governed by differential equations. Since the vast majority of applications are implemented by computer programs, discrete-time systems are considered, where differential equations are replaced by difference equations. A looped neural network in discrete time is governed by one or more nonlinear difference equations resulting from the composition of functions performed by each neuron and the delays associated with each connection [DJE17].

Figure 2.4. Illustrate the General architecture of recurrent neural networks:

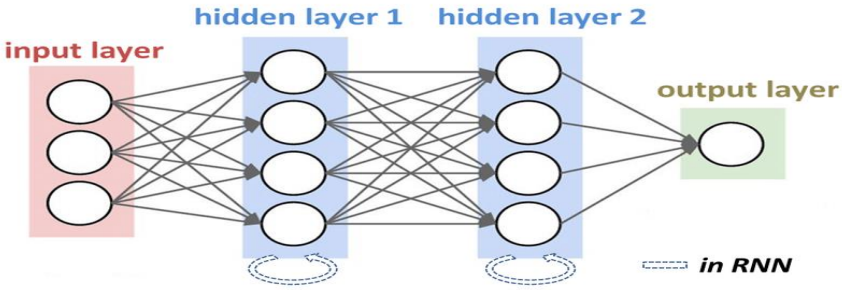


Figure 2.4. General architecture of recurrent neural networks [RNN]

3.2. Neural Network Models

3.1.1. Hopfield Model

The Hopfield model, introduced in 1982, is a straightforward model based on the principle of associative memories, earning its designation as an associative network due to its resemblance to memory retrieval mechanisms. The model adopts a fully connected and recurrent network architecture, with undirected connections where each neuron does not influence itself. The outputs of the network are determined by both the inputs and the network's previous state [DJE17].

3.1.2. Kohonen Model

T. Kohonen introduced this model in 1982, drawing inspiration from biological observations. Its purpose is to represent complex data, typically existing in a discrete space of large dimensions, with a topology constrained to one or two dimensions. Kohonen maps are constructed using a two-layer network: one for input and one for output. It's important to note that the neurons of the input layer are fully connected to the output layer. Neurons in the output layer are arranged in a one- or two-dimensional space, with each neuron typically having neighboring neurons in this spatial arrangement. Additionally, each neuron in the output layer forms recurrent lateral

connections within its layer, where distant neurons are inhibited while neighboring neurons are allowed to interact [DJE17].

3.1.3. Perceptron Model

The perceptron mechanism was pioneered by psychologist F. Rosenblatt in the late 1950s as an attempt to elucidate fundamental properties of intelligent systems. In this model, the network comprises three layers: an input layer, which supplies data to an intermediate layer responsible for computations by summing the impulses from connected cells and typically responding based on a defined law with a threshold. This intermediate layer is then connected to the output layer, known as the decision layer, which represents the examples to be memorized. Only the output layer sends signals to the intermediate layer until their connections stabilize [DJE17]

3.1.4. ADALINE Model

Widrow and Hoff's ADALINE is a three-layer network consisting of one input layer, one hidden layer, and one output layer. This model shares similarities with the perceptron model, with the main distinction lying in the transfer function, which changes but remains linear. It's important to note that the neuron models employed in both the perceptron and ADALINE are linear models [DJE17].

3.1.5. Multilayer Perceptron (MLP)

To surpass linear separations, additional hidden layers are introduced to construct multilayer neural networks. Within a layer, neurons are not interconnected. By incorporating hidden layers, a multilayer network extends the capabilities of the perceptron. Perceptrons themselves are a type of feedforward neural network. In a typical three-layer network, there are three inputs and one output. The input layer distributes inputs, the hidden layer serves as the network's core for processing, and the output layer furnishes the final result. Layers are numbered from input to output; for instance, a network with three neurons in the hidden layer is denoted as 3-3-1 [DJE17].

3.3. Training

3.3.1. Term definitions

a) Training

Training is a dynamic and iterative process that enables the modification of a network's parameters in response to stimuli from its environment. The type of training is dictated by how parameter changes occur.

This definition suggests that a network must be stimulated by its environment, undergo changes in response to this stimulation, and subsequently generate new responses to the environment in the future. Through this iterative process, the network can enhance its performance over time [DJE17].

b) Overfitting

Overfitting arises when a machine learning model captures noise within the training data rather than learning the underlying distribution. As a consequence, the model exhibits exceptional performance on the training data but performs poorly when applied to unseen test data. This occurs because the model becomes excessively complex, effectively memorizing the intricacies and noise within the training data to such an extent that it detrimentally affects its ability to generalize to new data [YU18].

c) Underfitting

Underfitting happens when a machine learning model is too basic and fails to capture the underlying patterns in the data. This leads to poor performance not only on the training set but also on new data. Such a model struggles to recognize the inherent patterns in the training data, often resulting in high bias and low variance [YU18].

3.3.2. Training strategy

Training within neural networks is contingent upon both the network's architecture and the specific problem it aims to address. While principles such as Hebb's and Widrow's dictate the adjustment of neuron weights, they lack universal applicability, as each network possesses unique requirements. Unlike deductive logic, which derives new knowledge from established facts, learning in neural networks involves generalizing from limited data. This process encompasses memorization, where examples are densely stored, and generalization, which entails processing new, similar data. Striking a balance between memorization and generalization is crucial, as prioritizing one can impede the other. Statistical learning systems tend to prioritize generalization. Discovering an appropriate learning coefficient necessitates experimentation [DJE17].

3.3.3. Learning modes

a) Supervised learning

"Supervised learning" is distinguished by the presence of a "teacher" who possesses a comprehensive understanding of the environment within which the neural network operates. In practical terms, the knowledge of this "teacher" is encapsulated in a set of input-output vector pairs, denoted as $\{((x_1, d_1), (x_1, d_1) \ldots\ldots (x_\eth, d_\eth))\}$, where (x_i) represents a stimulus (input) and (d_i) represents the target output for that stimulus, i.e., the desired outputs of the network.

Each pair (x_i, d_i) corresponds to a case specifying what the network should produce (the target) for a given stimulus. Hence, supervised learning is also commonly referred to as learning from examples [DJE17].

One of the most well-known algorithms for supervised learning is backpropagation. It is commonly used to train neural networks, specifically multilayer perceptrons (MLPs), by

iteratively adjusting the network's parameters to minimize the difference between predicted and actual outputs.

a.1) Error Gradient Back-Propagation Algorithm

One widely adopted algorithm in neural networks is known as the "back-propagation" algorithm. It functions by modifying the weights of a fixed-structure network upon the presentation of each example (x_i, y_i). This modification is intended to minimize the discrepancy between the desired output and the network's response to the input (x_i). The algorithm leverages the technique of gradient descent, where, in each cycle, the input signal traverses the network from input to output, generating an output. The error between this output and the expected result is then computed. Back-propagation is employed to calculate the errors associated with intermediate layers, particularly the hidden layer, facilitating adjustments to the weights (w_{ij}) of these layers [DJE17].

The error back-propagation algorithm involves two phases:

1. *Propagation:* At each step, an input example is presented to the network. This input propagates to the output layer.
2. *Correction:* Typically, the network output won't precisely match expectations. Thus, an error is computed (usually the mean squared error across all output neurons) and propagated back through the network. This process halts once the overall error is deemed sufficiently minimized.

b) Reinforcement learning

Reinforcement learning presents an alternative approach to address certain limitations of supervised learning. While it shares similarities with supervised learning, it operates with a scalar satisfaction index instead of a vector error signal. In practice, the implementation of reinforcement learning is complex, so we will not explore networks that utilize it in detail. Nevertheless, it is crucial to grasp the distinction between reinforcement learning and supervised learning [DJE17]

c) Unsupervised learning

The third type of learning is known as "unsupervised" or "self-organized" learning. Unlike supervised learning, which has a teacher providing error signals, or reinforcement learning, which uses a satisfaction index, unsupervised learning operates without any such guidance. Only environmental stimuli are present, and the network learns independently without external intervention.

In this approach, the network models the internal state of stimuli, optimizing its synaptic weights based on a defined quality measure. Ultimately, the network forms internal representations that encode the characteristics of stimuli, automatically creating similar stimulus classes. Unsupervised learning relies on a competitive process to develop a model where synaptic

weights represent prototypes of stimuli. These prototypes are evaluated using a metric that measures the distance between the stimuli and the prototypes [DJE17].

3.4. Advantages and disadvantages of neural network

As with any technique, neural networks possess a range of advantages and disadvantages that must be carefully considered in their application and implementation [MIJ21].

3.4.1. Advantages

- **Entire network information storage:** Unlike traditional programming where data is stored in databases, an artificial neural network (ANN) stores information across its entire network. This decentralized storage means the network can still function even if some information is lost in one area.
- **Handling incomplete knowledge:** Post-training, an ANN can still produce outputs with incomplete data. The degree of performance degradation depends on the significance of the missing information.
- **Fault tolerance:** Even if one or more cells within an ANN are corrupted, it can still generate outputs, showcasing its fault tolerance.
- **Distributed memory:** To facilitate learning, ANNs require exposure to examples that demonstrate the desired outputs. The network's effectiveness correlates directly with the quality of these examples; incomplete or inadequate exposure may lead to inaccurate outputs.
- **Gradual degradation:** Over time, an ANN may slow down and experience diminishing performance, but these issues typically manifest gradually rather than abruptly.
- **Machine learning capability:** ANNs learn from events and make decisions based on patterns identified in similar events.
- **Parallel processing:** ANNs possess the computational capability to execute multiple tasks concurrently, leveraging their inherent parallel processing capabilities.

3.4.2. Disadvantages

- **Hardware dependence:** Artificial neural networks require processors with parallel processing power, in accordance with their structure. For this reason, the realization of the equipment is dependent.
- **Unexplained behavior of the network:** This is the most important problem of ANN. When ANN produces a probing solution, it does not give a clue as to why and how. This reduces trust in the network.
- **Determination of proper network structure:** There is no specific rule for determining the structure of artificial neural networks. Appropriate network structure is achieved through experience and trial and error.
- **Difficulty of showing the problem to the network:** ANNs can work with numerical information. Problems have to be translated into numerical values before being introduced to

ANN. The display mechanism to be determined here will directly influence the performance of the network. This depends on the user's ability.

- **The duration of the network is unknown:** The network is reduced to a certain value of the error on the sample means that the training has been completed. This value does not give us optimum results.

Artificial neural networks, which emerged in the mid-20th century, continue to advance swiftly. Today, we have thoroughly explored both the benefits and challenges associated with their application. It is important to note that as this evolving branch of science addresses its drawbacks, the advantages of artificial neural networks are progressively enhancing. This trajectory suggests that artificial neural networks are destined to become integral to our lives, growing in significance over time.

4. Deep learning

In this section we will try some basic understanding for deep learning.

4.1. What's Deep learning?

Deep learning, categorized under machine learning, employs algorithms that aim to extract abstract features from data through multiple layers of processing. These layers comprise intricate structures or numerous nonlinear transformations. In contrast to shallow learning models like Support Vector Machines and Logistic Regression, introduced in the 1990s, represents shallow machine learning models characterized by having either no hidden layers or only one layer of nodes, as depicted in Figure 2.2. In contrast, deep learning relies on multi-layer neural networks as depicted in Figure 2.3 where each layer uses the output from the preceding layer as its input. This approach enables deep learning to effectively learn and extract highly abstract features from data [HAO19].

4.2. Overview of the Primary Deep Learning Algorithm

4.2.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of deep neural network mainly used for processing structured grid data, such as images. They employ convolutional layers to automatically and adaptively learn spatial hierarchies of features from input images, which makes them particularly effective for image recognition and classification tasks [KRI10].

a) CNN Architecture

CNN consists of several key architectural elements that sequentially transform input data into meaningful outputs (Figure 2.5).

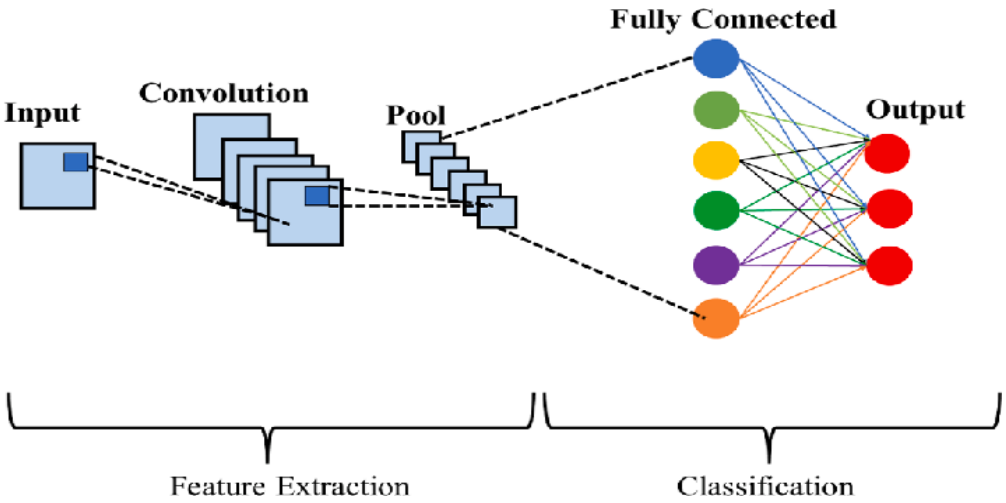


Figure 2.5. Schematic diagram of a basic CNN architecture [CNN]

a.1) Input Layer

The input layer receives the raw pixel values of the image. For instance, an RGB image with a size of 224 × 224 pixels has an input dimension of 224 × 224 × 3 [KRI10].

a.2) Convolutional Layers

These layers utilize convolutional filters on the input image, helping to extract local features such as edges, textures, and patterns. Each filter slides across the image to produce feature maps. The convolution operation involves calculating the dot product between the filter matrix and the current patch of input it overlays. Activation functions like ReLU (Rectified Linear Unit) are often used to introduce non-linearity [KRI10].

a.3) Pooling Layers

Pooling layers, frequently employing max pooling, are employed for downsampling, effectively reducing the spatial dimensions of the feature maps. This helps in decreasing the number of parameters and computations within the network, while also assisting in controlling overfitting [KRI10].

a.4) Fully Connected Layers

After multiple convolutional and pooling layers, the neural network's higher-level reasoning takes place via fully connected layers. In these layers, each neuron establishes connections with all activations from the previous layer, mirroring the structure of neurons in a conventional feedforward neural network [KRI10].

a.5) Output Layer

The final fully connected layer often employs a softmax activation function for classification tasks, enabling the output of probabilities for different classes [KRI10].

b) Domains of Application

b.1) Computer Vision

- Image Classification: Assigning a label to an entire image.
- Object Detection: Identifying and locating objects within an image.
- Segmentation: Dividing an image into parts and classifying each part.
- Facial Recognition: Identifying and verifying faces in images.
- Medical Imaging: Analyzing medical scans such as MRI and CT images for diagnosis.

b.2) Robotics

- Object Manipulation: Recognizing and interacting with objects.
- Navigation: Understanding and navigating the environment.

b.3) Autonomous Vehicles

- Obstacle Detection: Identifying obstacles in the path of the vehicle.
- Lane Detection: Recognizing lane boundaries on roads.

4.2.2. Generative Adversarial Network

The Generative Adversarial Network (GAN) was introduced in 2014, employing two distinct models: a generative model and a discriminative model. The discriminative model's task is to discern whether a given image is real or generated, while the generative model aims to produce images that resemble real ones as closely as possible. The generated images are crafted to deceive the discriminative model, which in turn learns to distinguish between real and generated images. Both models are trained simultaneously, enhancing their performance through iterative confrontation until they reach equilibrium. GANs are highly versatile, applicable not only to image generation and discrimination but also to various other types of data [HAO19].

a) GAN architecture

Generative Adversarial Networks (GANs) comprise two main components: a generative model and a discriminative model, which work in tandem to produce and evaluate synthetic data samples (Figure 2.6).

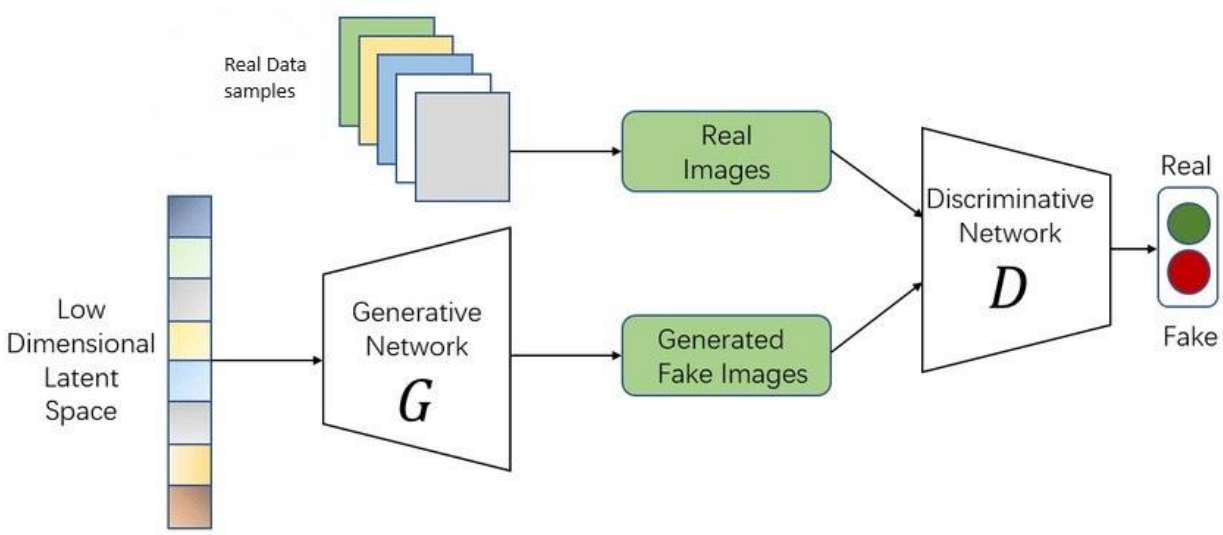


Figure 2.6. GAN architecture

a.1) Generative Model

The generative model aims to generate synthetic data samples that resemble real data. It typically consists of a neural network that takes random noise as input and generates data samples (e.g., images) from this noise [GOO14].

a.2) Discriminative Model

The discriminative model, often referred to as the discriminator, is tasked with distinguishing between real data samples (e.g., real images from a dataset) and fake data samples generated by the generative model. It is implemented as another neural network trained to classify whether a given input is real or generated [GOO14].

a.3) Training Process

During training, the generative model and the discriminative model are trained simultaneously in a competitive manner. The generative model improves its ability to generate realistic data samples to fool the discriminator, while the discriminator enhances its ability to accurately distinguish between real and generated samples.

This adversarial process leads both models to iteratively improve until an equilibrium is reached, where the generative model produces realistic samples that are indistinguishable from real data [GOO14].

b) Domains of Applications

GANs have found applications beyond image generation, including data synthesis, image-to-image translation, style transfer, and more recently, in natural language processing and drug discovery [GOO14].

4.3. Advantages and Disadvantages of Deep Learning

Deep learning has some advantages and challenges that influence its application and development. Here are the principal ones [HAO19]:

4.3.1. Advantages

- **Performance:** Deep learning often outperforms traditional neural networks, especially in tasks like image classification.
- **Efficiency:** Once a deep neural network is trained and properly adjusted, it saves significant computational effort and can handle large workloads quickly.
- **Flexibility:** Unlike traditional algorithms, which may require extensive code changes for model adjustments, deep learning models can be adjusted by simply modifying parameters within a pre-determined network framework.
- **Continuous Improvement:** The deep learning framework can be continuously improved, approaching an almost perfect state over time.
- **Generality:** Deep learning models can be tailored to various problems and are not limited to specific issues.

4.3.2. Disadvantages

- **High Training Cost:** Training deep learning models can be expensive due to the need for high-performance computing modules, despite improvements in computer hardware. Simple neural networks can be trained on common computing modules, but complex networks require costly high-performance modules.
- **Data Requirements:** Deep learning requires large amounts of data for training to achieve satisfactory performance, which can be challenging to obtain.
- **Manual Feature Marking:** Most deep learning frameworks rely on manual feature marking for training, leading to a significant workload when dealing with large-scale datasets.
- **Lack of Theoretical Support:** Despite the practical success of deep learning, there is still a lack of comprehensive and rigorous theoretical explanations for its models, limiting further study and improvement.

4.4. Deep Learning Applications

Deep learning has revolutionized various domains of data processing, demonstrating remarkable capabilities in both image and audio applications. Here are its principal applications [HAO19]:

4.4.1. Image Processing

- Manual feature selection is laborious and time-consuming, often leading to instability. Deep learning allows computers to automatically learn features, improving efficiency and consistency.

- Utilizes multi-layer neural networks for pre-processing, feature extraction, and processing.
- Neural networks can process large amounts of image data more efficiently than manual recognition, even though they currently do not match the accuracy of the human eye.
- Deep learning advances face recognition technology, a biometric recognition based on human facial features.
- Widely used in various fields like finance, justice, military, public security, and more. Technology maturity and social recognition will lead to broader applications and promising development prospects.

4.4.2. Audio Data Processing

- Extracts and enhances speech signals, performing pre-processing, cleaning, and feature extraction.
- Differentiates features into distinct sounds using deep learning techniques.
- Combines sounds into words and sentences, utilizing various neural model-based architectures for classification.
- Enhanced efficiency and accuracy across all levels due to deep learning methods.

4.5. Complexity Control

Complexity control in deep learning is essential for optimizing model performance and ensuring robustness. It involves managing the intricacy of neural network architectures to prevent overfitting and enhance generalization capabilities.

4.5.1. Regularization

In elementary networks, a simple option to prevent **overfitting** consists of introducing a penalty term or regularization, as in ridge regression, into the optimization criterion. This criterion then becomes: $Q(\theta) + \gamma||\theta||^2$. The higher the value of the parameter γ (decay), the less chaotic the input weights of the neurons can become, thus helping to limit the risks of overfitting [DJE17].

4.5.2. Parameter Selection

Therefore, the user must determine:

1. The input variables and the output variable; subject them, as with all statistical methods, to possible transformations and normalizations.
2. The network architecture: the number of hidden layers, which corresponds to an ability to handle non-linearity problems, and the number of neurons per hidden layer. These two choices directly influence the number of parameters (weights) to be estimated and thus the complexity of the model. They contribute to finding a good bias/variance trade-off, meaning the balance between learning quality and prediction quality.
3. Three other parameters also affect this trade-off: the maximum number of iterations, the maximum tolerated error, and a possible term of ridge regularization (decay).

4. The learning rate as well as a possible strategy for its evolution.
5. The size of the observation sets or batches considered at each iteration.

In practice, all these parameters cannot be adjusted simultaneously by the user. The user is faced with choices mainly concerning the control of overfitting: limiting the number of neurons or the duration of training or increasing the penalty coefficient of the parameter norm. This requires determining an error estimation mode: *validation sample* or *test*, *cross-validation*, or *bootstrap*. A simple and likely effective strategy is to introduce a rather large number of neurons and then optimize only the regularization parameter (decay) through cross-validation.

5. Conclusion

In this chapter we explored the fundamentals of neural networks, covering their structure, learning methods, and key models like Hopfield and MLP. We highlighted the advantages and disadvantages of neural network. We also covered deep learning, emphasizing its role in advancing neural networks, and discussed both Generative Adversarial Network (GAN) and Convolutional Neural Networks (CNNs), which are pivotal for image and pattern recognition tasks. We also highlighted the advantages and disadvantages of deep learning and its application. This overview underscores the significance of neural networks and deep learning in artificial intelligence and their potential for future advancements.

Chapter 3 : Conception

1. Introduction

In this chapter, we will present our contribution to solving the problem of noise classification in historical document images. As discussed in Chapter 1, these documents are very important as they represent the collective memory of a society. However, they are often affected by various types of noise that complicate their processing at different levels.

Our methodology relies on leveraging the remarkable advances in deep learning and artificial intelligence techniques to identify and classify the different types of noise present in these documents. We will discuss the various stages of our approach, from data preparation to model implementation, including the challenges encountered and the solutions provided.

The rest of this chapter is structured as follows...

2. Objective of our work

Our primary objective is to develop a robust noise classification system capable of accurately identifying and classifying the most significant types of noise affecting historical document images. This classification not only aids in improving the accuracy of document processing tasks but also helps in preserving and restoring historical documents by identifying the prevalent types of degradation.

Accurate noise classification is crucial because it enables more precise and effective document processing. For instance, knowing the specific type of noise present in an image allows for tailored preprocessing steps, such as specialized filters or algorithms designed to handle that particular type of degradation. This leads to better results in optical character recognition (OCR) and other downstream tasks, even though these are not our primary focus.

Moreover, the ability to identify and classify different types of noise can significantly aid in the preservation and restoration of historical documents. By recognizing patterns of paper damage, transparency issues, spots, and fading, conservators can develop targeted strategies to mitigate these effects. For example, documents with significant paper damage might benefit from physical repairs or digital reconstruction techniques, while those affected by fading could be digitally enhanced to improve readability.

Additionally, our noise classification system provides valuable insights into the common types of degradation affecting historical documents. This information is essential for researchers studying the aging processes of these materials and for archivists looking to implement better storage and handling practices to prevent future damage. By focusing on four specific types of noise—paper damage, transparency issues, spots, and fading pictures—our system helps prioritize and streamline the efforts of those involved in document preservation.

Ultimately, our work aims to create a reliable and effective tool that supports the ongoing efforts to maintain the integrity and accessibility of historical documents. By ensuring these documents

are accurately classified and appropriately handled, we contribute to their longevity and usability for future generations.

3. Noise types considered in our study

Although there are many types of noise and degradation that can affect historical document images, our study focuses on a specific subset of these. As discussed in Chapter 1, document images can be subject to various sources of noise, including environmental factors, handling, and the natural aging process. For the purposes of our research, we have selected four types of noise to study: 'paper damage', 'transparency', 'spots', and 'fading picture'.

We chose these specific noise types for several reasons:

1. ***Prevalence***: These noise types are commonly found in historical documents. Paper damage and fading are frequent due to the age and handling of documents, while spots and transparency issues can occur due to environmental conditions and storage methods.
2. ***Impact on Readability:*** Each of these noise types significantly affects the readability and usability of the documents. For instance, paper damage can lead to missing information, while fading can make text difficult to decipher.
3. ***Technical Challenges:*** These types of noise present unique challenges for image processing and classification, making them suitable for testing the robustness and effectiveness of our methodologies.

Based on these considerations, our classification methodology includes five different classes:

1. ***Paper Damage:*** This class includes images with physical damage to the paper, such as tears, holes, or extensive wear.
2. ***Transparency:*** This class covers images where text or graphics from the opposite side of the page or underlying layers are visible.
3. ***Spots:*** This class contains images with spots or stains caused by ink, water, mold, or other contaminants.
4. ***Fading Picture:*** This class includes images where the ink has faded over time, making the text or graphics less visible.
5. ***Clean image:*** This class is for images that do not exhibit any of the above noise types and are considered clean.

4. Used Dataset

Datasets are fundamental to any machine learning task and classification problem, serving as the cornerstone for training and evaluating models. Faced with the need for a suitable dataset, we had two options: either construct our own dataset or search for a public and freely available one that would be suitable for our specific task.

4.1. Searching for a suitable freely available dataset

Creating a dataset from scratch was a difficult undertaking, particularly gathering ancient records, which may be fairly challenging. As first attempt, we tried to find an existing dataset suitable for our situation. So we conducted an extensive search for a public dataset that is both large and specifically tailored for noise classification purposes. Unfortunately, the only dataset proposed in the literature for this purpose is the MHDID dataset [SHA18]. However; the MHDID dataset is private and not readily accessible to researchers. Despite our efforts to contact the authors and request access to the dataset for academic and research purposes, we did not receive any response from them.

4.2. Creation of our own dataset

As a result of the non-availability of a freely available public dataset suitable for noise classification, we were compelled to prepare our own dataset. Building such a dataset requires meticulous collection and annotation of images to ensure they are representative of the wide array of noise types encountered.

In Figure 3.1, we illustrate the sequential steps involved in the creation of our dataset. We outline these steps below.

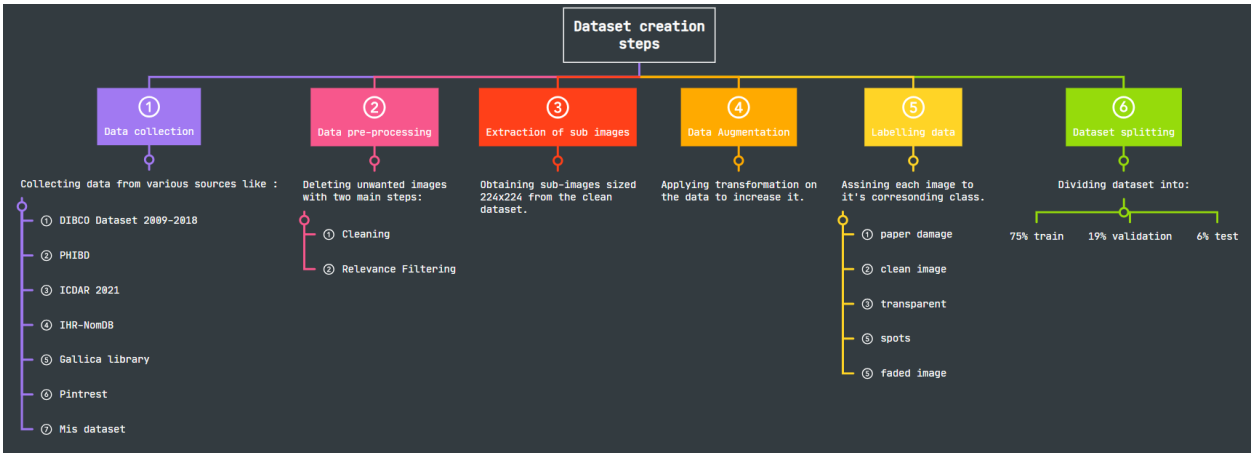


Figure 3.1. The sequential steps involved in the creation of our dataset.

4.2.1. Data collection

The majority of the data we acquired came from publicly available datasets such as DIBCO 2009-2018 [GAT09, PRA10, PRA11, PRA12, PRA13, NTI14, PRA16, PRA17, PRA18], ICDAR 2021 [JOS21], and PHIBD [AYA13] , IHR-NomDB [VU21] and MIS dataset [HED13]. These datasets are publicly available, and they were proposed in the context of international competitions for various document processing and analysis tasks, such as binarization, tabular data localization, writer identification, etc., in order to provide a base for research and development in these areas.

In addition to these primary sources, we also utilized data extracted from internet sources such as Gallica and Pinterest. These platforms offered a diverse range of historical document images, enriching our dataset with a variety of noise types and document styles.

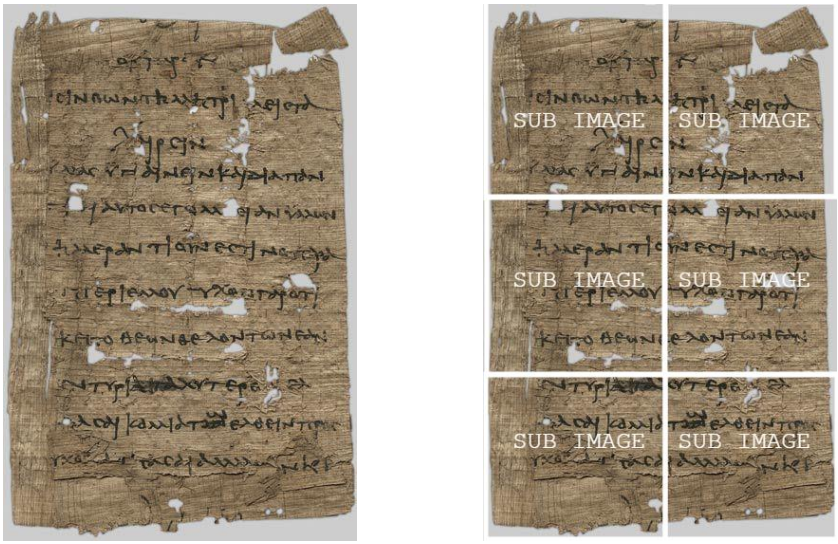
4.2.2. Data preprocessing

Data preprocessing is a critical step that involves cleaning the collected data and removing images that are irrelevant to our problem [REH12]. This ensures that the dataset used for training and evaluation is of high quality and specifically tailored to the task of noise classification in historical document images.

1. **Cleaning:** This step involves inspecting each image to ensure it meets the necessary criteria for inclusion. Images that are blurry, corrupted, or do not contain any historical document content are discarded.
2. **Relevance Filtering:** We manually reviewed the dataset to remove images that do not contain noise types pertinent to our classification task. This included eliminating images with modern prints, non-document content, or noise types that fall outside the scope of our classification criteria.

4.2.3. Extraction of Sub-Images

After the data preprocessing step, we proceeded with the extraction of sub-images from the cleaned dataset. Each sub-image measures 224 × 224 pixels and contains a single type of noise as it is shown in Figure 3.2.



A) Original image *B) sub-images from the original image*

Figure 3.2. Example of sub-image extraction.

This step is crucial for multiple reasons. First, by focusing on sub-images with only one type of noise, we ensure that the classification model learns to accurately identify and differentiate between specific types of degradation, thereby enhancing its reliability. Second, standardizing

the size of the images to 224 × 224 pixels ensures consistency in the input to the neural network, which is vital for the performance of deep learning models. Lastly, this process increases the size of the dataset, as multiple sub-images can be extracted from a single document image. This augmentation of the data helps to reduce the risk of overfitting during model training.

The extraction process involves segmenting each historical document image into multiple 224 × 224 pixel sub-images. We then filter these sub-images to ensure that each one contains a single type of noise, discarding those with mixed noise types or no noise at all.

4.2.4. Data augmentation

Before proceeding to data labeling, we implemented a data augmentation step to enhance the robustness and generalizability of our classification. Data augmentation involves generating additional training samples from the existing dataset through various transformations (Figure 3.3). This step helps to prevent overfitting by exposing the model to a broader range of variations within each class and increases the overall size of the dataset, which is particularly beneficial when the original dataset is limited in size.

In this step, and in order to enhance the diversity and volume of our dataset, we implemented two types of transformations on our existing data: rotation and flipping. These transformations were chosen to augment our dataset effectively, enabling our model to learn from a broader range of image variations and orientations [MIK18].

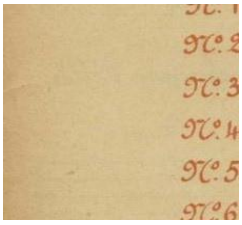
A) Original image *B) Augmented image*

Figure 3.3. Example of Image augmentation technique

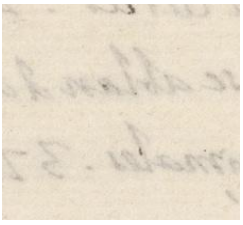
4.2.5. Data labelling

After the process of sub-image extraction, our dataset consisted of a set of sub-images, each containing a single noise type. The next step was to categorize our dataset into the five classes previously mentioned for our classification problem: 'paper damage', 'clean image', 'transparent', 'spots', and 'fading picture'. We meticulously tagged each sub-image with its relevant class, ensuring that our dataset was appropriately labeled and ready for training our image classification models.

Figure 3.4 illustrates examples of each noise type from our dataset:

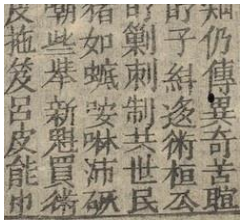


A) Clean image

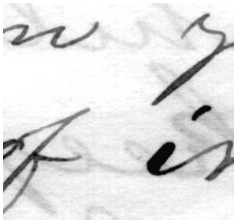


B) Faded text

C) Paper damage



D) Spots



E) Transparency from the back

Figure 3.4. Example Images from Each Class of the Dataset

4.2.6. Dataset Splitting

The final step in preparing our dataset involved dividing it into three subsets: training, validation, and testing. We allocated 75% of the data to the training set, 19% to the validation set, and another 6% to the testing set.

- **Training Set**: This subset, was used to train our noise classification models. A larger training set allows the model to learn from a diverse range of examples and variations within each class, improving its ability to generalize to unseen data.
- **Validation Set**: The validation set, was utilized to tune hyperparameters and monitor the performance of the model during training. By evaluating the model on a separate validation set, we could make adjustments to the model architecture or training process to optimize performance.
- **Testing Set**: The remaining of the dataset was reserved for the testing set, which served as an independent evaluation dataset. This subset was used to assess the final performance of the trained model on unseen data, providing an objective measure of its effectiveness in classifying noise types in historical document images.

5. Methodology

In developing our noise classification system for degraded document images, we sought to leverage the advancements in deep learning due to its demonstrated success in various image classification tasks. Our approach involved defining and training one or several deep learning

models on our dataset to automatically learn and extract relevant features from historical document images.

By harnessing the power of deep learning, we aimed to build a robust classification system capable of accurately identifying and categorizing different types of noise present in these images.

However, we explored different scenarios, ranging from using pretrained existing models to the proposition of a customized model tailored specifically to our dataset and task at hand. Furthermore, we investigated the possibility of combining the outputs of multiple models to improve classification accuracy. This comprehensive exploration allowed us to assess the effectiveness of each approach and determine the most suitable strategy for our noise classification system.

The methodology may be summarized by Figure 3.5, which illustrates the various stages involved in our approach to developing the noise classification system.

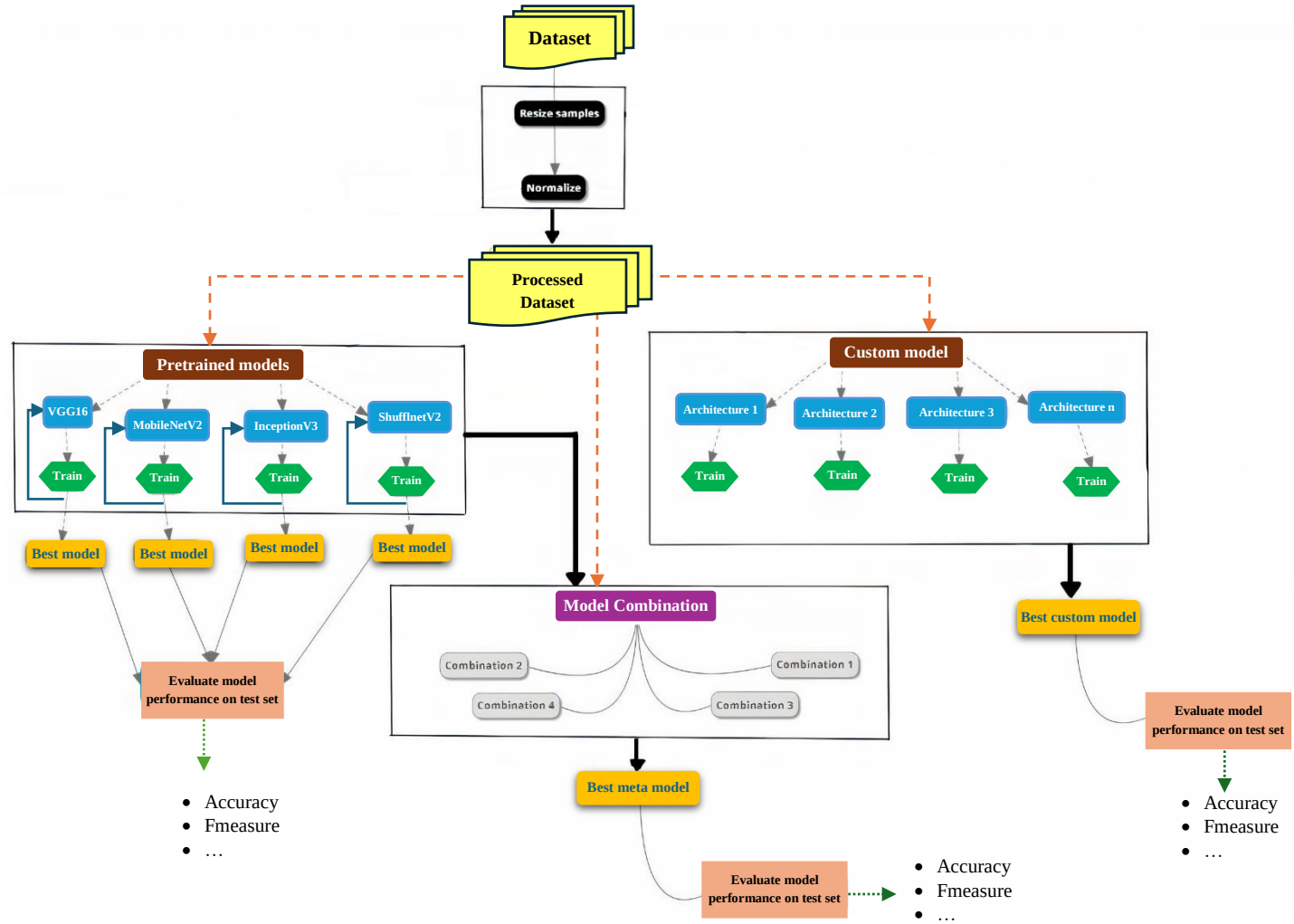


Figure 3.5. The various stages involved in our approach to developing the noise classification system.

5.1. Preprocessing the dataset

We preprocessed the dataset before feeding it into the deep learning models to guarantee the best possible performance. Each image was resized as part of the preprocessing to a constant size of 224 × 224 pixels (if not already resized), which is the minimum input size needed for the pretrained deep learning models. We also normalized the image pixel values to fall between 0 and 1 and transformed the images to the RGB color space. This procedure is crucial because it guarantees consistency and standardizes the input data, which enhances the model's performance and convergence during training.

The code used for performing preprocessing is illustrated in Figure 3.6.

```
        class_dir = os.path.join(dataset_dir, class_name)
        for image_name in os.listdir(class_dir):
            image_path = os.path.join(class_dir, image_name)
            image = cv2.imread(image_path)
            image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            # Resizing the images
            image = cv2.resize(image, (224, 224))
            X.append(image)
            y.append(i)
    return np.array(X), np.array(y)
# Load the dataset
X, y = load_dataset(dataset_dir)
# Normalize the images
X = X.astype('float32') / 255.0
```

Figure 3.6. Preprocessing Code for Dataset

5.2. Exploring pretrained models

In this project, we experimented with several pre-trained models including VGG16 [SIM15], InceptionV3 [SZE16] , ShuffleNetV2 [MA18] , and MobileNetV2 [SAN18] to determine their effectiveness in addressing our image classification problem.

We adapted these models by replacing their final layers with layers that are compatible with our specific classification problem. These models, renowned for their effectiveness in a wide range of computer vision tasks, offer different architectures and complexities. By examining and adapting these models to our specific problem, we aim to determine which architecture yields the best results among these pre-trained models when applied to our dataset.

5.2.1. VGG16 model

a) Model architecture

The VGG16 model, introduced by Simonyan and Zisserman in 2014, is a convolutional neural network architecture renowned for its simplicity and effectiveness in image recognition tasks. This model consists of 13 convolutional layers and 3 fully connected layers. The architecture

includes an input layer that takes an image of size (224, 224, 3). The convolutional layers are responsible for extracting features from the input image, with each convolutional layer followed by a ReLU activation function and a max-pooling layer. Following the convolutional layers, there are 3 fully connected layers. The output of the last fully connected layer is passed through a SoftMax activation function to obtain the final classification probabilities [SIM15].

In our adaptation of the VGG16 model for our classification problem, we made the following modifications (Figure 3.7): Firstly, we loaded the pre-trained VGG16 model using the VGG16 function from '`tensorflow.keras.applications`'. Then, we replaced the original output layer of the VGG16 model with a new fully connected layer and a ReLU activation function, followed by a dropout layer with a dropout rate of 0.5. The choice of 512 neurons in the fully connected layer was determined through experimentation. We tested several values to find the optimal number of neurons that balance model complexity and performance. Too few neurons could lead to underfitting, where the model does not learn enough to make accurate predictions. Conversely, too many neurons could lead to overfitting, where the model performs well on training data but poorly on unseen data. After extensive trials, we found that 512 neurons provided the best performance in terms of both training efficiency and classification accuracy, making it the optimal choice for our specific dataset and noise classification task. The purpose of adding the dropout layer is to prevent overfitting by randomly dropping out 50% of the neurons during training, forcing the network to learn more robust features. Finally, we added a dense layer with the number of units equal to the number of classes in our classification problem (5 in our case) and a SoftMax activation function.

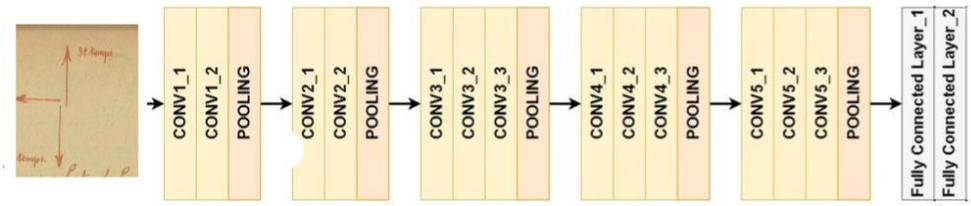


Figure 3.7. VGG16 custom model architecture

b) Training the model

As previously mentioned, the dataset is split into three subsets: 75% for training, 19% for validation, and 6% for testing.

Training is conducted by first loading the preprocessed dataset and feeding it into the modified VGG16 model. Additionally, we employed early stopping and model checkpointing techniques to monitor the validation loss during training. Early stopping helps prevent overfitting by halting training when the validation loss stops improving for a specified number of epochs, while model checkpointing saves the best model weights based on validation performance.

The training process involved iterating over the dataset in batches, allowing the model to update its weights incrementally. We trained the model for a maximum of 50 epochs, but early stopping typically reduced the number of epochs needed.

By the end of the training process, the model was fine-tuned to classify the different types of noise in historical document images.

With this setup, we achieved an accuracy of 97% after 25 epochs.

Later, we adopted a different approach by employing a cross-validation technique called ***K-Fold Cross-Validation***. In this technique, the part of the dataset used for the training and validation, known as the ***Training-Validation set*** (comprising 75% + 19% of the dataset) is randomly partitioned into K equal-sized subsets or "folds". The model is trained on $K-1$ folds and tested on the remaining one fold. This process is repeated K times, with each fold being used exactly once as the test set. The performance metric (e.g., accuracy, precision, recall) is averaged over the K iterations to provide a more robust estimate of the model's performance.

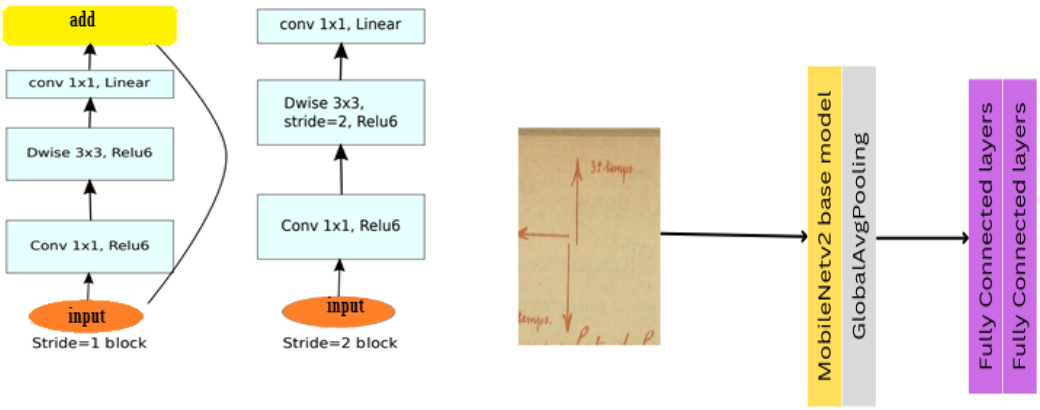
However, we divided our training-validation set into 80% for training and 20% for validation and opted for 5 folds in total. Each fold underwent 10 epochs of training. The mean accuracy across all folds was 96%.

5.2.2. MobileNetV2

a) Model architecture

The MobileNetV2 model is a convolutional neural network architecture designed to be efficient both in terms of computational complexity and memory usage. It consists of a series of building blocks that perform depth-wise separable convolutions and inverted residual blocks [SAN18].

In our adaptation of the MobileNetV2 model for our classification problem, we loaded the pre-trained MobileNetV2 model and used the same parameters as the VGG16 model, including a fully connected layer with 512 units, a dropout rate of 0.5, and a SoftMax activation function as it is shown in Figure 3.8 As with the VGG16 model, the hyperparameters of the custom MobileNetV2 model were chosen through experimentation.



A) MobileNetV2 base model *B) Custom model architecture*

Figure 3.8. MobileNetV2 custom model architecture

b) Training the model

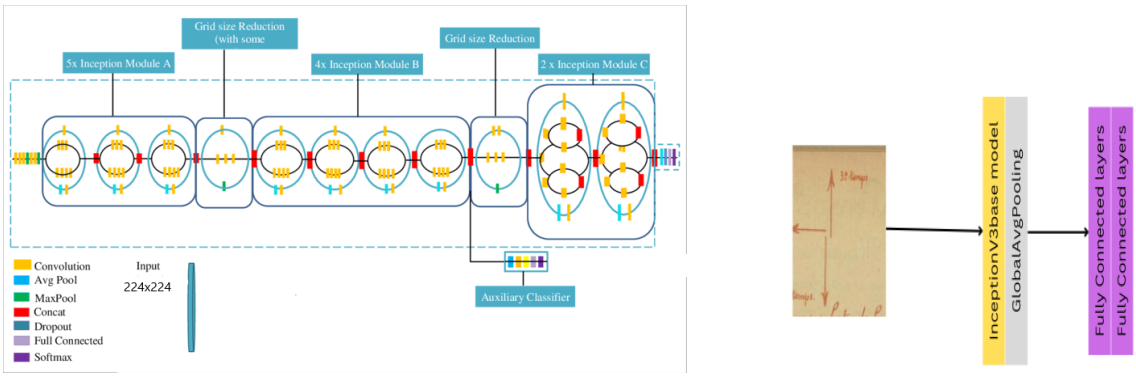
For this model, we also used early stopping and model checkpointing techniques to monitor the validation loss in addition to the training loss at each epoch, in order to identify the optimal stopping point for training. We employed the same cross-validation technique, ***K-Fold Cross-Validation***. We split the dataset in the same manner as the VGG16 model, with 80% for training and 20% for validation, using 5 folds and training for 10 epochs for each fold. The mean accuracy across all folds was **97.97**%.

5.2.3. InceptionV3

a) Model architecture

The InceptionV3 model is a convolutional neural network architecture designed to achieve high accuracy with relatively low computational cost. It employs a series of inception modules, which perform convolutions of multiple sizes and pooling operations in parallel, allowing the network to capture information at different scales and significantly improve performance. Additionally, the model utilizes batch normalization and factorized convolutions to optimize the efficiency of the architecture [SZE16].

In our adaptation of the InceptionV3 model for our classification problem, we loaded the pre-trained InceptionV3 model and used the same parameters as the VGG16 and MobileNTV2 models, including a fully connected layer with 512 units, a dropout rate of 0.5, and a SoftMax activation function.



A) InceptionV3 base model *B) custom model architecture*

Figure 3.9. InceptionV3 custom model architecture

b) Training the model

For this model, we also used early stopping and model checkpointing techniques to monitor the validation loss in addition to the training loss at each epoch, in order to identify the optimal stopping point for training. We employed the same cross-validation technique, ***K-Fold Cross-Validation***. We split the dataset in the same manner as the other models, with 80% for training and 20% for validation, using 5 folds and training for 10 epochs for each fold. The mean accuracy across all folds was **97.15%**

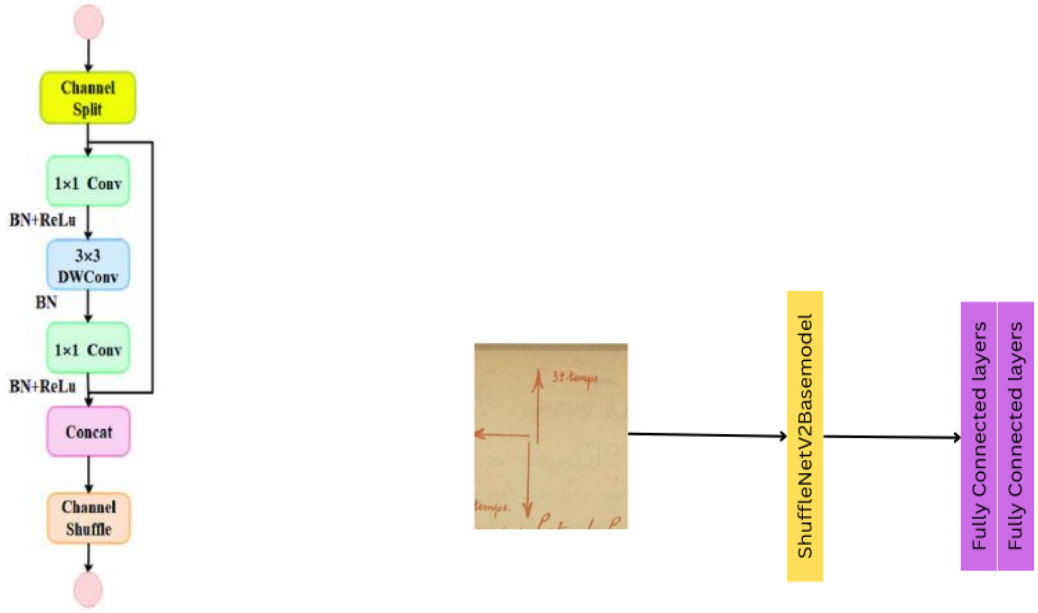
5.2.4. ShuffleNetV2

a) Model architecture

The ShuffleNetV2 model is a convolutional neural network architecture designed to be efficient both in terms of computational complexity and memory usage. It employs a series of building blocks that perform pointwise group convolutions, channel shuffle operations, and depthwise separable convolutions. This design allows for a significant reduction in computational cost while maintaining high accuracy [MA18].

In our adaptation of the ShuffleNetV2 model for our classification problem, we loaded the pre-trained ShuffleNetV2 model and used the same parameters as the VGG16 model, including a fully connected layer with 512 units, a dropout rate of 0.5, and a SoftMax activation function.

Note that, the hyperparameters of the custom ShuffleNetV2 model were chosen through experimentation, similar to the previous models.



A) ShuffleNetV2 base model *B) custom model architecture*

Figure 3.10. ShuffleNetV2 custom model architecture

b) Training the model

For this model, we also used early stopping and model checkpointing techniques to monitor the validation loss in addition to the training loss at each epoch, in order to identify the optimal stopping point for training. We employed the same cross-validation technique, ***K-Fold Cross-Validation***. We split the dataset in the same manner as the other models, with 80% for training and 20% for validation, using 5 folds and training for 10 epochs for each fold. The mean accuracy across all folds was **99%.**

5.3. Creating our own deep learning model

In the second experiment we explored involved creating a deep learning model from scratch. Our rationale for pursuing this approach stemmed from a critical observation regarding pretrained models. We recognized that models like VGG16, MobileNetV2, and ShuffleNetV2 are typically pretrained on datasets such as ImageNet, which predominantly consist of natural scene images. These pretrained models are initialized and their parameters are tuned based on the characteristics of this dataset.

However, historical document images, the focus of our problem, possess distinct characteristics that set them apart from natural scene images. They often exhibit unique patterns of noise and degradation, which may not be adequately represented in datasets like ImageNet. Consequently, pretrained models may not be optimally suited for our specific task of noise classification in historical document images.

Given this disparity between the training data of pretrained models and the target domain of historical document images, we reasoned that fine-tuning these models for our task might necessitate substantial modifications or even complete retraining. Such endeavors could be both time-consuming and resource-intensive, potentially diminishing the efficiency and effectiveness of the models.

To address these challenges and capitalize on the domain-specific nature of historical document images, we opted to create our own deep learning model from scratch. This approach afforded us the opportunity to tailor the architecture directly to the intricacies of historical document noise, without being constrained by pretrained model architectures designed for different domains.

In this section, we will discuss the architecture of the deep learning model we experimented with, the challenges we encountered.

5.3.1. Architecture of the Proposed Model

We experimented with several architectures and hyperparameters to develop a model that effectively captures and classifies noise in historical document images. Through rigorous experimentation, we aimed to identify the most suitable architecture and parameters for our specific task. The performances of various experimented architectures will be presented in Chapter 4.

However, the most suitable CNN model, which provided the best performances, designed for image classification, consists of four convolutional layers and two fully connected layers. Each convolutional layer uses ReLU activation, followed by batch normalization, max pooling, and dropout for regularization (Figure 3.11).

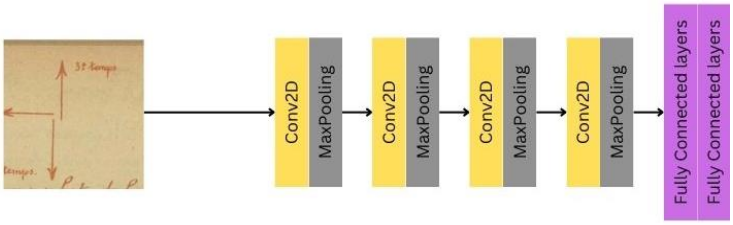


Figure 3.11. Custom model architecture

5.3.2. Training the proposed model

The training of the proposed model followed a similar procedure as that used for pretrained models, employing cross-validation, early stopping, and model checkpointing techniques.

During training, the model only reached a validation accuracy of 70% but this validation accuracy was the result of the validation accuracy fluctuating which means model is giving completely random predictions even though the train accuracy was improving steadily which was a sign that the model was suffering on the unseen data or in another term the model was overfitting. The classification reports and confusion matrices for both the test and validation sets

revealed substantial misclassifications across all classes. This clearly showed the model's difficulty in accurately distinguishing between the different types of image damage.

5.3.3. Efforts to Improve Model Performance

Various efforts were made to improve the model's performance, including experimenting with different architectures, changing the number of layers, adjusting the number of neurons, modifying dropout rates, and tweaking learning rates. Despite these attempts, the model's performance remained disappointing.

5.4. Models' combination

In the third experiment we explored involved combining multiple models in an ensemble approach. Ensemble learning has proven to be an effective strategy for improving model performance by leveraging the diversity of multiple models.

5.4.1. Ensemble Learning

Ensemble learning is a powerful technique in machine learning that aims to improve the performance and robustness of models by combining multiple base learners. This approach leverages the strengths of different algorithms to reduce errors and enhance predictive accuracy.

There are several popular methods for creating ensemble models:

a) Bagging (Bootstrap Aggregating)

Bagging involves training multiple versions of a predictor on different bootstrap samples (random samples with replacement) from the original dataset and then averaging their predictions (for regression) or using majority voting (for classification). The process generates multiple datasets by sampling with replacement from the training data, with each dataset used to train a separate model, typically of the same type. The final prediction is made by aggregating the predictions of these individual models. Bagging aims to reduce variance and prevent overfitting, making models more stable and reliable [MIE22].

a.1) Example Algorithms

- **AdaBoost**: Adjusts the weights of incorrectly classified instances so that subsequent models focus more on them.
- **Gradient Boosting**: Builds models sequentially, each new model trained to predict the residuals (errors) of the previous models.
- **XGBoost**: An efficient and scalable implementation of gradient boosting that includes regularization to prevent overfitting.
- **LightGBM**: A gradient boosting framework that uses tree-based learning algorithms, optimized for speed and efficiency.

- **CatBoost**: A gradient boosting algorithm that handles categorical features effectively without the need for extensive preprocessing.

a.2) Advantages

- Produces highly accurate models.
- Effective in handling various types of data.

a.3) Limitations

- Can be prone to overfitting if not properly regularized.
- Computationally expensive due to sequential training.

b) Boosting

Boosting is an ensemble technique that converts weak learners into strong ones by training them sequentially, each new model correcting the errors of its predecessors. In the boosting process, each model in the sequence is trained to emphasize the training instances that previous models misclassified, with the predictions combined through weighted voting or averaging. The primary purpose of boosting is to reduce bias and improve accuracy by iteratively correcting errors [MIE22].

b.1) Example Algorithms

- **Random Forest**: An ensemble of decision trees, each trained on a different bootstrap sample, and using random subsets of features for further randomness.
- **Extra Trees Classifier**: Like Random Forest, but with further randomness in node splits.

b.2) Advantages

- Reduces variance without increasing bias.
- Introduces diversity in the input data due to bootstrapping.

b.3) Limitations

- Computationally intensive for large datasets.
- Decreases interpretability due to the use of multiple models.

c) Stacking

Stacking, also known as Stacked generalization, is an ensemble learning technique that combines the predictions of multiple base models using a meta-model. Unlike bagging, where base models operate independently, stacking leverages the outputs of base models as features for training a higher-level model, called a meta-classifier. Stacking is particularly effective when the base models exhibit complementary strengths and weaknesses [WOL92].

c.1) Example Techniques

- **Super Ensemble**: Uses various types of base models and a meta-learner to combine their predictions.
- **Blending**: A simpler form of stacking where a holdout dataset is used to train the meta-learner.
- **SVM**: SVM is known for its robustness in high-dimensional spaces and its effectiveness in both classification and regression tasks.

c.2) Advantages

- Can achieve higher predictive performance by combining different model types.
- Flexible in the choice of base and meta-learners.

c.3) Limitations

- More complex to implement and tune.
- Requires careful consideration of model diversity and the risk of overfitting the meta-learner.

d) The technique we chose

We chose the 'stacking' technique as it is known to be one of the most effective ensemble learning methods. Stacking involves combining the predictions of multiple base models (VGG16, MobileNet, InceptionV3, and ShuffleNet) and using them as features to train a meta-model. This approach is powerful because it learns to correct the errors of the base models using a higher-level model.

d.1) Stacking technique results

In our approach, we first trained four base models (VGG16, MobileNet, InceptionV3, and ShuffleNet) individually on the training data. Next, we collected the predictions of these base models on a validation set and used them as input features to train a ***logistic regression*** meta-classifier (Figure 3.12). After conducting several experiments with different types of meta-classifiers, including decision trees, support vector machines, and neural networks, we found that logistic regression consistently provided the best performance in terms of accuracy, interpretability, and computational efficiency. Therefore, we chose logistic regression as our meta-classifier to combine the predictions of the base models effectively.

This meta-classifier learns how to best combine the predictions from the base models to improve overall predictive performance. In the models part, we explored various combinations to identify the optimal ensemble that could yield the highest performance. However, through experimentation and evaluation, we discovered that the combination of these two base models InceptionV3, and ShuffleNet proved to be the most effective. The stacking technique achieved a test accuracy of **98%**.

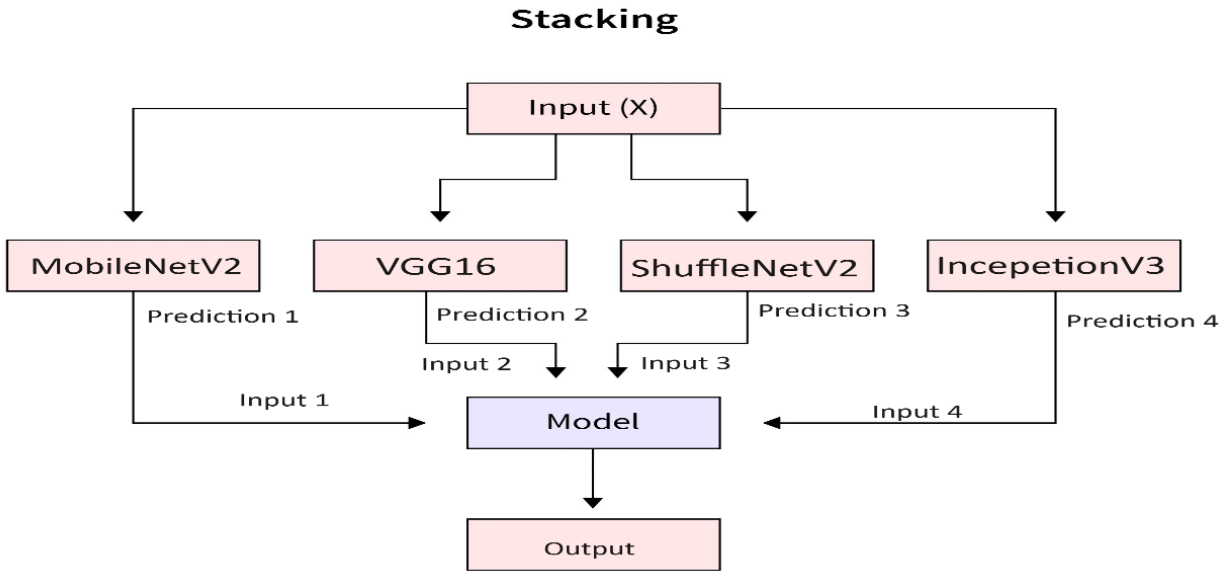


Figure 3.12. Stacking technique

6. Conclusion

In conclusion, this chapter synthesizes the efforts made and the methodology adopted to resolve the problem of identifying noise types in document images. We compared the effectiveness of four deep learning architectures: VGG16, InceptionV3, ShuffleNetV2, and MobileNetV2 for our image classification task. Additionally, we experimented with creating a custom architecture tailored to our problem, but the results were underwhelming. Ultimately, we combined the pre-trained models to leverage their strengths. By analyzing datasets containing images affected by various types of noise, we aimed to identify the most suitable model for our purposes. Armed with these insights, we are better equipped to refine our image classification methods and address real-world challenges more effectively.

Chapter 4: Implementation and Results

1. Introduction

In this chapter, we detail the implementation process and present our results. We start with an overview, followed by a description of the development environment, including the programming language and the libraries used. Next, we showcase the application, highlighting its features and functionality. Finally, we present our experimental results, analyzing key performance metrics and discussing their implications.

2. Development environment

In this section, we outline the language, libraries and tools used for implementing the project

2.1. Programming Language

The primary programming language was Python, chosen for its versatility and extensive library support.

2.2. Libraries

TensorFlow [MAR15] and ***PyTorch*** [PAS19] libraries were utilized, providing robust frameworks for model development and training.

For the user interface, we chose the ***tkinter*** library in Python for its ease of use and versatility.

2.3. Tools

The initial coding and experimentation were conducted on Kaggle and Google Colab, leveraging their powerful CPU, RAM, and GPU resources to handle computationally intensive tasks efficiently. These platforms facilitated seamless development and testing, ensuring optimal performance and scalability of the application.

3. Presentation of the application

Our interface is like this:

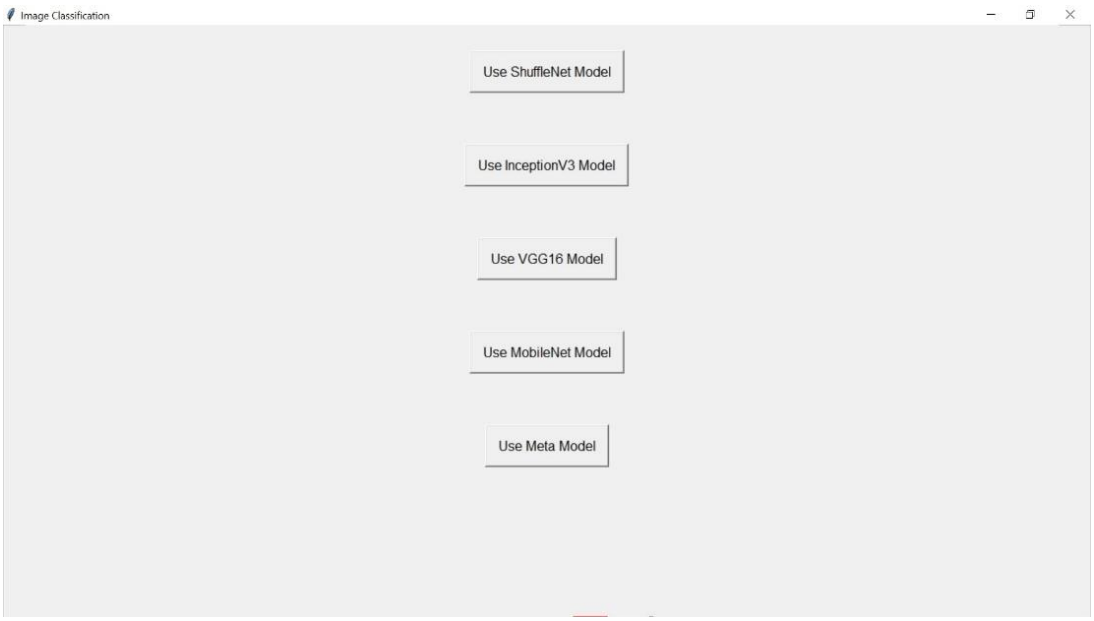


Figure 4.1. Initial interface

In the initial interface, we have five buttons each button allows us to choose which model we want to use.

After choosing the model another interface will pop up and the first one will be closed. In the figure 4.2 is presentation of the second interface:

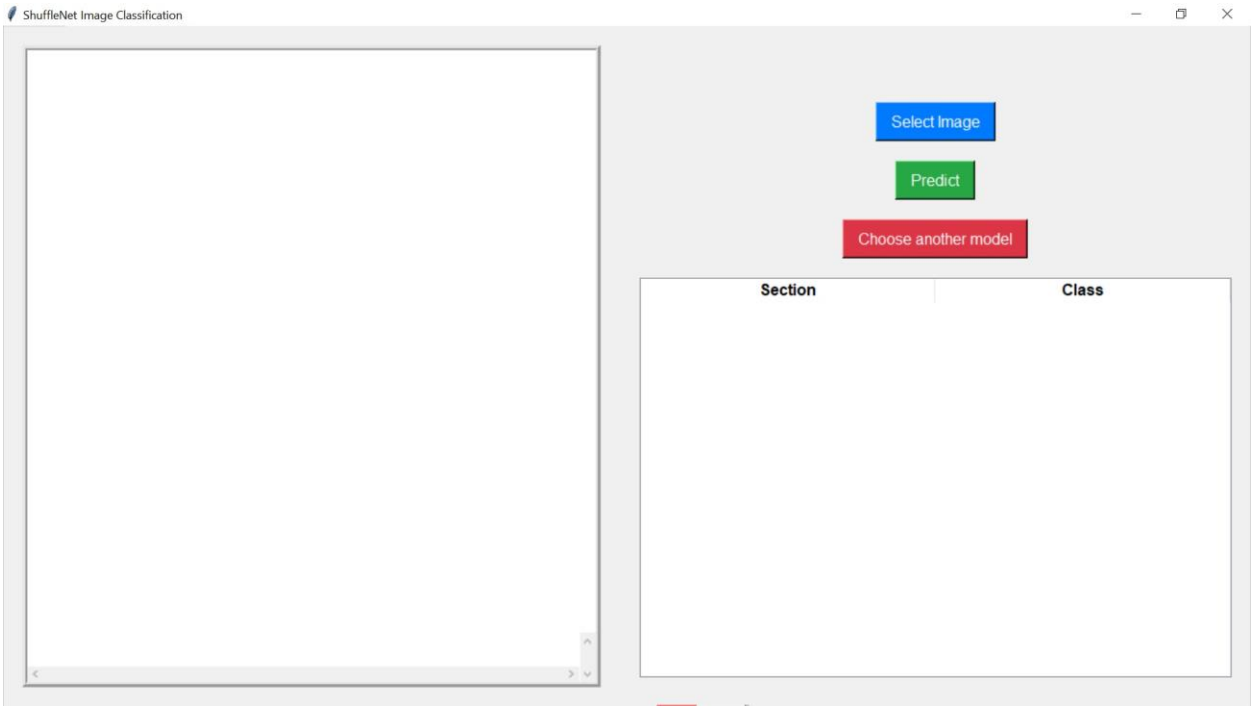


Figure 4.2. Second interface

In the second interface we have 3 main buttons that allows us to choose an image and predict what type of noise is in that image and a button that will take us to the initial interface:

- Select image button allows us to select image from our local machine.
- Predict button allows us to predict the type of noise in the selected image.
- Choose another model button allows us to go back to the initial interface and choose different model.

After choosing the image and clicking on the button predict the chosen model will predict the type of noise in that image as shown in the Figure 4.3:

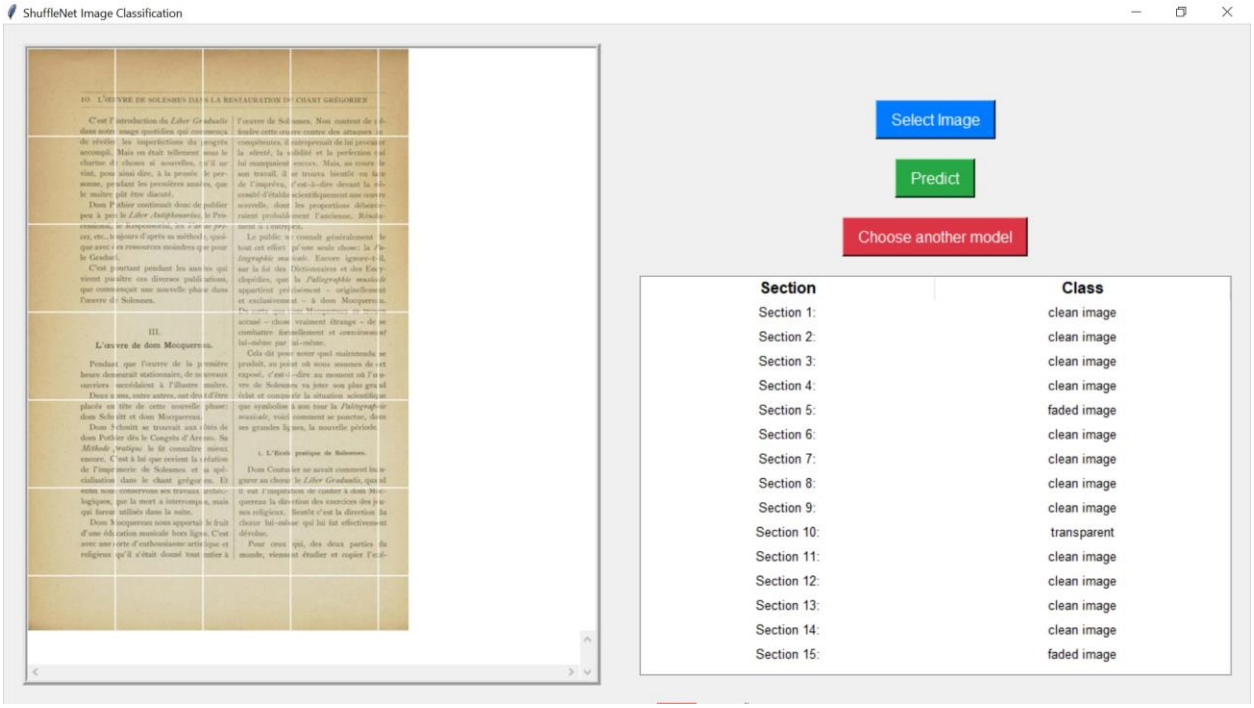


Figure 4.3. Third interface

In Figure 4.3, after selecting an image sized 763x1093 pixels, the image is divided into smaller sections of 224 × 224 pixels. If any of these sections are smaller than 224 × 224 pixels, they are resized to fit that size. Each section is then processed by the model, which identifies the type of noise present in each section along with its probability.

4. Experiments and results

In this section, we will present the experiments conducted to choose the different model architectures and evaluate the performance of our noise classification models. Our experiments aimed to rigorously assess the effectiveness of various model architectures, training techniques, and hyperparameters in classifying noise in historical document images.

We will provide a detailed overview of the experimental setup, including the dataset used for training, validation, and testing, as well as the evaluation metrics employed to quantify the

models' performance. Additionally, we will present the results of our experiments, showcasing the performance of each model and ensemble configuration across different classes of noise.

4.1. Dataset Summary

Our dataset is composed of 4285 images. Each of the five classes, 'paper damage', 'Clean image', 'transparent', 'spots', and 'fading picture', had 857 images. The dataset is split into three sets: training, validation, and testing.

Table 4.1 illustrates the distribution of images between the training validation, and testing sets.

	Train	**Validation**	**Testing**	**Total**
Percentage	75%	19%	6%	100%
Number of images	3228	807	250	4285

Table 4.1. The distribution of images between the training, validation, and testing sets.

4.2. Evaluation measurements

To evaluate the performance of our noise classification models, we employed a range of evaluation metrics that provide a comprehensive understanding of each model's strengths and weaknesses. These metrics include:

- **Accuracy**: The proportion of correctly classified images out of the total number of images. We can calculate the **accuracy** by using this formula.

$$Acc = \frac{Number\ of\ Correct\ Predictions}{Total\ number\ of\ predictions} \qquad \text{[DEN09]}$$

- **Precision**: The ratio of true positive classifications to the sum of true positive and false positive classifications for each class. This metric indicates the accuracy of the model in identifying each specific class. We can calculate the **Precision** by using this formula.

$$Pre = \frac{True\ Positive}{(True\ Positive + False\ Positive)} \qquad \text{[CIR11]}$$

- **Recall**: The ratio of true positive classifications to the sum of true positive and false negative classifications for each class. This metric reflects the model's ability to correctly identify all instances of a given class. We can calculate the **Recall** by using this formula.

$$Rec = \frac{True\ Positive}{(True\ Positive + False\ Negative)} \qquad \text{[SIM14]}$$

- **F1-Score**: The harmonic mean of precision and recall, providing a single metric that balances both considerations. We can calculate the **F1-score** by using this formula.

$$F1 = 2 * \frac{Pre * Rec}{Pre + Rec} \quad \text{[KRI12]}$$

- **Confusion Matrix**: A matrix that shows the true versus predicted classifications, allowing for a detailed examination of which classes are being confused by the model.
- **Training Time**: The time taken to train the model, which is important for assessing the practicality of the model in real-world applications.
- **True Positive**: A true positive occurs when the model correctly predicts the positive class.
- **False Positive:** A false positive occurs when the model incorrectly predicts the positive class.
- **False Negative:** A false negative occurs when the model incorrectly predicts the negative class for an instance that is actually positive.

4.3. Experiment 1: Searching for the best architectures of custom pretrained models

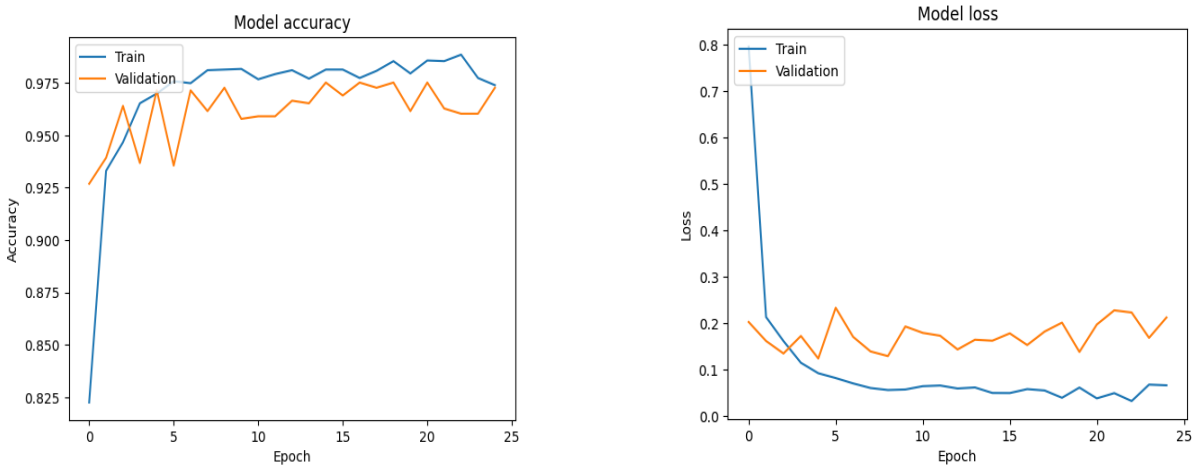
In this section, we describe the experiments conducted to determine the optimal architectures for the pretrained models explored and integrated into our system. We focused on fine-tuning these models to enhance their performance on our specific task of noise classification in historical document images.

4.3.1. Assessing VGG 16

Several experiments were conducted to select the best architecture for our custom VGG16 model. We tested the model with and without cross-validation to evaluate the benefits of cross-validation for our specific problem.

a) Without cross-validation

The training results for the parameters indicated in chapter 3 were as follows:



A *B*

Figure 4.4. Model Training Progress: Accuracy, Loss, Validation Accuracy and Validation Loss per Epoch

Figure 4.4.A illustrates the training progress of the model over 25 epochs. The training accuracy steadily increases over epochs, reaching approximately 97.40% accuracy by the end of training. The validation accuracy also increases over epochs, peaking at around 97.27% accuracy.

Figure 4.4.B displays the training and validation loss values over 25 epochs. The training loss steadily decreases over epochs, indicating improved model performance. Similarly, the validation loss decreases initially but starts to fluctuate after around 15 epochs.

The confusion matrix for these parameters is presented in the following table:

	Clean image	**faded image**	**Spots**	**transparent**	**paper damage**	**Total**
Clean image	174	0	0	0	0	**174**
faded image	0	164	1	4	1	**170**
Spots	0	5	153	4	3	**165**
Transparent	0	2	0	148	1	**151**
paper damage	0	0	0	1	146	**147**
Total	**174**	**171**	**154**	**157**	**151**	**807**

Table 4.2. Confusion Matrix for the first customed VGG16 Model.

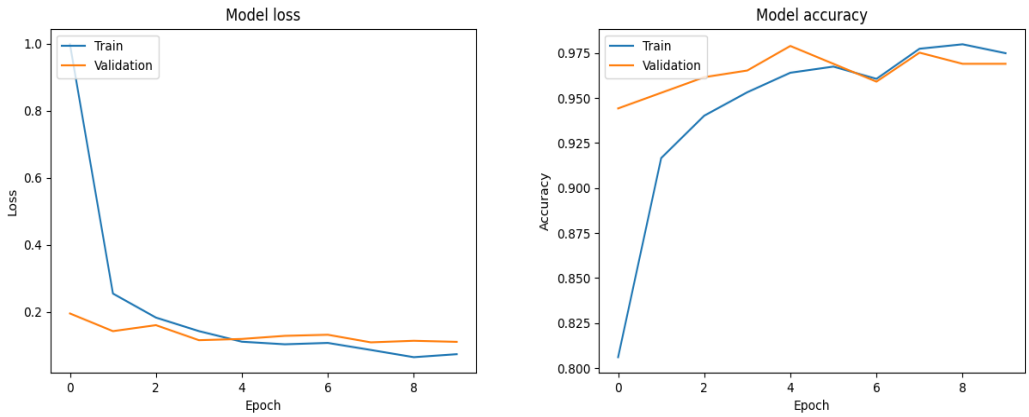
We obtained precision, recall, and F1-score for each class from the confusion matrix. The results are summarized in the table below:

	Clean image	faded image	Spots	transparent	paper damage	Average
precision	100%	96%	99%	94%	97%	97,2%
recall	100%	96%	93%	98%	99%	97,2%
f1-score	100%	96%	96%	96%	98%	97,2%

Table 4.3. Classification Performance by Class

b) With cross-validation

We employed a 5-fold cross-validation approach to better evaluate the performance of our VGG16 model. The results for each fold were averaged to provide a comprehensive assessment of the model's performance.



A *B*

Figure 4.5. Model Training Progress: Accuracy, Loss, Validation Accuracy and Validation Loss per Epoch

Figure 4.5.A illustrates the training progress of the model over 10 epochs. The training accuracy steadily increases over epochs, reaching approximately 97.50% accuracy by the end of training. The validation accuracy also increases over epochs, peaking at around 97.80% accuracy.

Figure 4.5.B displays the training and validation loss values over 10 epochs. The training loss steadily decreases over epochs, indicating improved model performance. Similarly, the validation loss decreases initially but starts to fluctuate.

The confusion matrix for these parameters is presented in the following table:

	Clean image	faded image	Spots	transparent	paper damage	Total
Clean image	159	2	0	0	0	**161**
faded image	0	160	0	1	0	**161**
Spots	0	2	156	4	0	**162**
Transparent	0	6	1	154	0	**161**
paper damage	0	8	0	1	153	**162**
Total	**159**	**178**	**157**	**160**	**153**	**807**

Table 4.4. Confusion Matrix for the cross validation VGG16 Model.

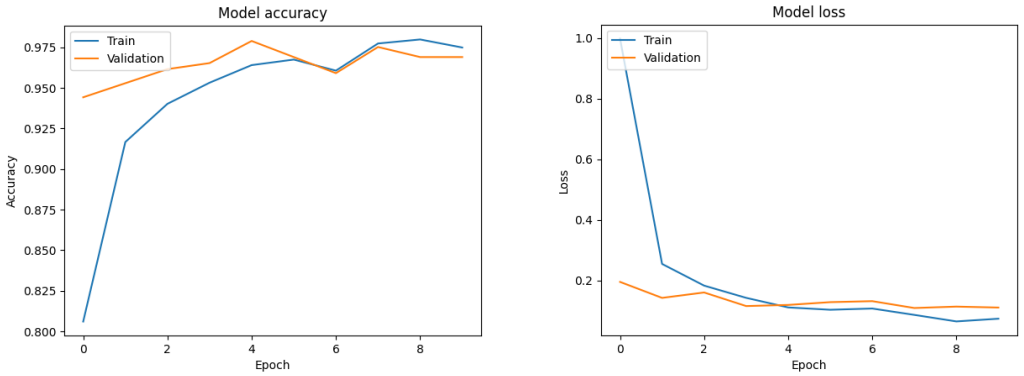
We obtained precision, recall, and F1-score for each class from the confusion matrix. The results are summarized in the table below:

	Clean image	faded image	Spots	transparent	paper damage	Average
precision	100%	90%	99%	96%	100%	97%
recall	99%	99%	96%	96%	94%	96,8%
f1-score	99%	94%	98%	96%	97%	96,8%

Table 4.5. Classification Performance by Class

4.3.2. Assessing MobileNetV2

To determine the optimal hyperparameters for our custom MobileNetV2 model, we conducted a series of experiments. We aimed to find the configuration that yielded the best performance. Using cross-validation, we systematically evaluated different setups. For fold number 2, the training results were as follows:



A) *B)*

Figure 4.6. Model Training Progress: Accuracy, Loss, Validation Accuracy and Validation Loss per Epoch

Figure 4.6.A illustrates the training progress of the model over 10 epochs. The training accuracy steadily increases over epochs, reaching approximately 99% accuracy by the end of training. The validation accuracy also increases over epochs, peaking at around 99,30% accuracy.

Figure 4.6.B displays the training and validation loss values over 10 epochs. The training loss steadily decreases over epochs, indicating improved model performance. Similarly, the validation loss decreases initially but starts to fluctuate.

The confusion matrix for these parameters is presented in the following table:

	Clean image	**faded image**	**Spots**	**transparent**	**paper damage**	**Total**
Clean image	161	0	0	0	0	**161**
faded image	0	161	0	0	0	**161**
Spots	0	3	158	1	0	**162**
transparent	1	3	5	151	1	**161**
paper damage	0	2	0	0	160	**162**
Total	**162**	**169**	**163**	**152**	**161**	**807**

Table 4.6. Confusion Matrix for the cross validation MobileNetV2 Model.

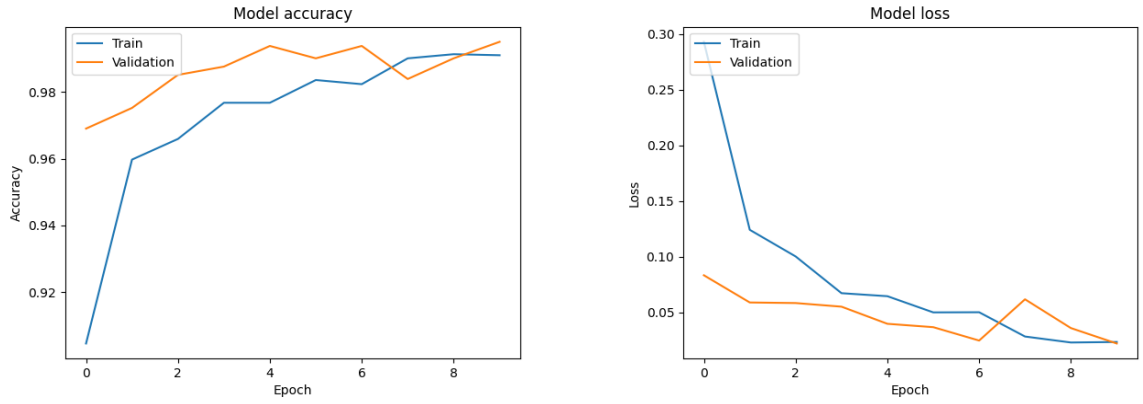
We obtained precision, recall, and F1-score for each class from the confusion matrix. The results are summarized in the table below:

	Clean image	faded image	Spots	transparent	paper damage	Average
precision	100%	95%	97%	99%	99%	98%
recall	100%	100%	98%	94%	99%	98%
f1-score	100%	98%	97%	96%	99%	98%

Table 4.7. Classification Performance by Class

4.3.3. Assessing InceptionV3

Similar to our approach with the VGG16 and MobileNetV2 models, we conducted a series of tests to determine the best architecture and hyperparameters for our custom InceptionV3 model. Using cross-validation, we systematically evaluated different configurations. For fold number 2, the training results were as follows:



A *B*

Figure 4.7. Model Training Progress: Accuracy, Loss, Validation Accuracy and Validation Loss per Epoch

Figure 4.7.A illustrates the training progress of the model over 10 epochs. The training accuracy steadily increases over epochs, reaching approximately 98% accuracy by the end of training. The validation accuracy also increases over epochs, peaking at around 99% accuracy.

Figure 4.7.B displays the training and validation loss values over 10 epochs. The training loss steadily decreases over epochs, indicating improved model performance. Similarly, the validation loss decreases initially but starts to fluctuate.

The confusion matrix for these parameters is presented in the following table:

	Clean image	faded image	Spots	transparent	paper damage	Total
Clean image	162	0	0	0	0	**162**
faded image	0	160	0	1	0	**161**
Spots	0	2	155	5	0	**162**
Transparent	0	1	0	160	0	**161**
paper damage	0	0	0	0	161	**161**
Total	**162**	**163**	**155**	**166**	**161**	**807**

Table 4.8. Confusion Matrix for the cross validation InceptionV3 Model.

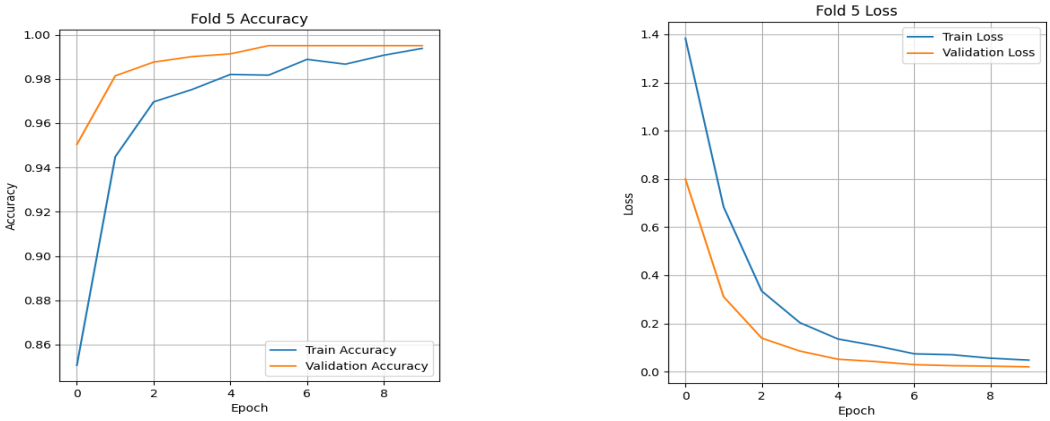
We obtained precision, recall, and F1-score for each class from the confusion matrix. The results are summarized in the table below:

	Clean image	faded image	Spots	transparent	paper damage	Average
precision	100%	98%	100%	96%	100%	98,6%
recall	100%	99%	96%	99%	100%	98,6%
f1-score	100%	99%	98%	98%	100%	99%

Table 4.9. Classification Performance by Class

4.3.4. Assessing ShuffleNetV2

The last pretrained model we explored was ShuffleNetV2. Through a series of tests and evaluations, we identified the optimal architecture for ShuffleNetV2, which has been described in detail in Chapter 3. Here, we present the results of our cross-validation experiments to assess its performance.



A *B*

Figure 4.8. Model Training Progress: Accuracy, Loss, Validation Accuracy and Validation Loss per Epoch

Figure 4.8.A illustrates the training progress of the model over 10 epochs. The training accuracy steadily increases over epochs, reaching approximately 99% accuracy by the end of training. The validation accuracy also increases over epochs, peaking at around 99% accuracy.

Figure 4.8.B displays the training and validation loss values over 10 epochs. The training loss steadily decreases over epochs, indicating improved model performance. Similarly, the validation loss decreases initially but starts to fluctuate.

The confusion matrix for these parameters is presented in the following table:

	Clean image	**faded image**	**Spots**	**transparent**	**paper damage**	**Total**
Clean image	162	0	0	0	0	**162**
faded image	0	161	0	0	0	**161**
Spots	0	0	162	0	0	**162**
transparent	0	2	0	156	3	**161**
paper damage	0	1	0	0	160	**161**
Total	**162**	**164**	**162**	**156**	**164**	**807**

Table 4.10. Confusion Matrix for the cross-validation Shuffle-Net Model.

We obtained precision, recall, and F1-score for each class from the confusion matrix. The results are summarized in the table below:

	Clean image	faded image	Spots	transparent	paper damage	Average
precision	100%	98%	100%	100%	98%	99,2%
recall	100%	100%	100%	97%	99%	99,2%
f1-score	100%	99%	100%	98%	99%	99,2%

Table 4.11. Classification Performance by Class

4.4. Experiment 2: Searching for the best architectures of our own deep learning model

In this experiment, we focused on developing and optimizing our own deep learning model tailored specifically for the noise classification task in historical document images. To identify the most effective architecture, we tested several configurations and parameters before adopting the final architecture detailed in Chapter 3. The following table summarizes some of the tested architectures and their corresponding performance on the testing set:

Architecture number	Convolutional layers number	Fully connected layers number	Mean Train accuracy	Mean Validation accuracy	Test accuracy
1	4	2	59%	40%	35%
2	4	3	66%	40%	36%
3	5	2	66%	42%	37%
4	6	2	85%	59%	54%
5	7	2	86%	64%	59%
6	10	2	69%	50%	46%
7	10	3	73%	54%	49%

Table 4.12. Deep learning custom architectures and their performance.

From analyzing the results of various architectures, we identified a common issue of overfitting across all models. Some architectures exhibited severe overfitting, while only one, architecture number 1, showed comparatively better results, though it still suffered from overfitting. Despite

its slight improvement, the train, validation, and test results were underwhelming. The reason that architecture number 1 is better than the others is that, although other architectures may have shown better results in training accuracy and validation accuracy, their validation accuracy fluctuated excessively. For instance, in one epoch, the validation accuracy might reach 80%, but then drop back to 20% in another epoch. This inconsistency indicates poor generalization to new data. We attempted different approaches to mitigate the overfitting problem, including incorporating dropout and L2 regularization, applying wavelet transforms to the data, and adjusting the models' hyperparameters and parameters. Unfortunately, these efforts were not successful.

4.5. Experiment 3: assessing the model combination procedure

In this experiment, we evaluated the efficacy of combining the predictions from multiple deep learning models to enhance the overall classification performance. Simultaneously, we explored the identification of the optimal classifier to serve as the meta-model, responsible for combining the outputs of individual models.

4.5.1. Stacking Models using Logistic Regression as Meta-Classifier

For the stacking technique, we combined predictions from multiple base models, including VGG16, MobileNet, InceptionV3, and ShuffleNet. Additionally, we utilized a logistic regression classifier as the meta-model. The obtained results are summarized in the following table:

Model combination	Validation accuracy
Vgg16, InceptionV3	98.5%
Vgg16,MobileNet	99%
Vgg16,ShuflleNet	99.2%
MobileNet,InceptionV3	99%
MobileNet,ShuffleNet	99.5%
InceptionV3, ShuffleNet	99.2%
InceptionV3,Vgg16,MobileNet	99.13%
ShuffleNet,Vgg16,MobileNet	99%
ShuffleNet, InceptionV3,MobileNet	99.2%
ShuffleNet, InceptionV3, Vgg16	99.1%
ShuffleNet, InceptionV3, Vgg16, MobileNet	99.5%

Table 4.13. Models Combination Used for Stacking Technique with Logistic Regression Classifier and Their Results

4.5.2. Stacking Models using SVM as Meta-Classifier

We also combined predictions from multiple base models, including VGG16, MobileNet, InceptionV3, and ShuffleNet using a SVM as the meta-model.

Model combination	Validation accuracy
Vgg16, InceptionV3	98.8%
Vgg16,MobileNet	99.5%
Vgg16,ShuflleNet	99.01%
MobileNet,InceptionV3	99.3%
MobileNet,ShuffleNet	99.26%
InceptionV3, ShuffleNet	99.5%
InceptionV3,Vgg16,MobileNet	99.26%
ShuffleNet,Vgg16,MobileNet	99.27%
ShuffleNet, InceptionV3,MobileNet	99.4%
ShuffleNet, InceptionV3, Vgg16	99.38%
ShuffleNet, InceptionV3, Vgg16, MobileNet	99.52%

Table 4.14. models Combination Used for Stacking Technique with SVM Classifier and Their Results

4.6. Performances evaluation

To evaluate the performance of our noise classification models, we utilized the testing set separated before, which serves as an independent dataset unseen by the models during training and validation. This testing set consists of 250 images in total, with 50 images for each class.

We employed a range of evaluation metrics, including training time, accuracy, precision, recall, and F1-score, to quantify the models' performance across different classes of noise. These metrics provide valuable insights into the models' ability to correctly classify historical document images based on the presence of various types of noise.

Table 4.15. shows the performance of each model on the test set.

The model	precision	F1-score	Recall	Testing accuracy
MobileNet	90,45%	89,06%	89,20%	89,20%
VGG16	88,26%	87,41%	87,60%	87,60%
ShuffleNet	93,86%	93,14%	93,20%	93,20%
InceptionV3	88,27%	88,12%	88,40%	88,40%
CustomModel	35,7%	34,8%	35,6%	35,6%
Combination	98.03%	98%	98%	98%

Table 4.15. Performance of each model on the test set

The examination of the results obtained highlights a clear disparity between the performance of established pre-trained models like MobileNet, VGG16, ShuffleNet, and InceptionV3, and our custom model. While the pre-trained models demonstrate notable precision, F1-score, recall, and testing accuracy, ranging from 87.60% to 93.20%, our custom model falls significantly short with a testing accuracy of only 35.6%. This substantial difference underscores the superior effectiveness and robustness of the pre-trained models in comparison to our custom architecture.

As for the model's combination, the significant improvement observed is due to the stacking technique, which leverages the strengths of multiple base models to create a more robust and accurate meta model. The meta model can capitalize on the diverse strengths of the base models and compensate for their individual weaknesses.

5. Conclusion

In this chapter we investigated three main areas: the first one is the development environment where we discussed the programming language, libraries and tools we used. The main interface of our application and the functions of each button were demonstrated and explained in the second section. The third section contains the results of our experiments with various pretrained, custom, and combined models, each of which was assessed on a test set to determine each model's performance. Each section provided valuable insights into each model's performance and effectiveness.

General Conclusion and Perspectives

Conclusion

This thesis focuses on developing an automated robust approach for classifying noise types in document images using deep learning techniques, particularly Convolutional Neural Networks (CNNs). Accurate classification of noise types is essential to apply the correct preprocessing techniques for restoring degraded documents. For our classifier we addressed five specific noise categories: transparency, faded image, paper damage, spots and clean images.

By training a CNN on our dataset, the network learned to distinguish between these different types of noise effectively. Our extensive experiments optimized the CNN's architecture and parameters, resulting in high classification accuracy. This approach significantly reduces the reliance on manual visual inspection, making the noise classification process more efficient.

The primary goal of this research is to build a robust automated classification model that accurately identifies the type of noise in document images, allowing for the application of the appropriate noise removal methods to achieve a noise-free image. Automating this process contributes significantly to the preservation of valuable documents by enhancing the efficiency and effectiveness of noise removal and image restoration techniques.

Perspectives

To further improve and expand upon our current noise classification model, the following enhancements can be considered:

- Expand the Dataset size so the model can learn more and give accurate results.
- Include Additional Noise Classes.
- Implement noise removal technique for each noise type
- Experiment with Different Neural Network Architectures.
- Implementing a multi-label classification system to handle cases where multiple types of noise affect the same document image simultaneously.
- improving our own deep learning architecture.

References

References

[ARN15] Arnia, F., Fardian, Muchallil, S., & Khairul, M. (2015). Noise characterization in ancient document images based on DCT coefficient distribution. *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, (pp. 971-975).

[ARN21] Arnia, F., Saddami, K., & Munadi, K. (2021, July). DCNet: Noise-Robust Convolutional Neural Networks for Degradation Classification on Ancient Documents. *Journal of Imaging*, 114.

[AYA13] Ayatollahi, S., & Nafchi, H. (2013). Persian heritage image binarization competition (PHIBC 2012). *2013 First Iranian Conference on Pattern Recognition and Image Analysis (PRIA)* (pp. 1-4). Birjand, Iran: IEEE.

[BEN11] Messaoud, I. B., El Abed, H., Amiri, H., & Märgner, V. (2011, September). New method for the selection of binarization parameters based on noise features of historical documents. In Proceedings of the 2011 Joint Workshop on Multilingual OCR and Analytics for Noisy Unstructured Text Data (pp. 1-8)

[BRI06] Briet, S., Day, R. E., Martinet, L., & Anghelescu, H. G. B. (2006). *What is Documentation?: English Translation of the Classic French Text*. Scarecrow Press. https://books.google.dz/books?id=2yMQZJWeL2IC

[CIR11] Ciresan, D., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2011). Flexible, High Performance Convolutional Neural Networks for Image Classification. In *International Joint Conference on Artificial Intelligence IJCAI-2011* (p. 1242). https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-210

[CNN] A lightweight CORONA-NET for COVID-19 detection in X-ray images - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Schematic-of-Convolutional-Neural-Network-CNN_fig4_369963460 [accessed 9 Jun, 2024].

[DEN09] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[DJE17] Djeriri, Y. (2017). Les Réseaux de Neurones Artificiels.

[DON16] Dong, C., Chen Change, L., & Tang, X. (2016). Accelerating the Super-Resolution Convolutional Neural Network. *ECCV 2016 (European Conference on Computer Vision).* Springer.

[DRI07] Drira, F. (2007). *Contribution à la restauration des images de documents anciens.* Institut National des Sciences Appliquées de Lyon, École Doctorale Informatique et Information pour la Société (EDIIS).

[END20] Endo, K., Tanaka, M., & Okutomi, M. (2020). Classifying Degraded Images Over Various Levels Of Degradation. *2020 IEEE International Conference on Image Processing (ICIP)* (pp. 1691-1695). Abu Dhabi, United Arab Emirates: IEEE.

[GAT09] Gatos, B., Ntirogiannis, K., & Pratikakis, I. (2009). ICDAR 2009 document image binarization contest (DIBCO 2009). *2009 10th International Conference on Document Analysis and Recognition* (pp. 1375–1382). Barcelona,Spain: IEEE.

References

[GHO15] Ghomrassi, A., Charrada, Mohamed, A., Essoukri Ben Amara, & Najoua. (2015). Restoration of ancient colored documents foreground/background separation. *2015 IEEE 12th International Multi-Conference on Systems, Signals & Devices (SSD15)*, (pp. 1-6).

[GON02] Gonzalez, R., & Woods, R. (2002). Digital Image Processing. *Prentice Hall*.

[GOO14] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks. *Advances in Neural Information Processing Systems*, *3*. https://doi.org/10.1145/3422622

[GOO16] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[HAO19] Hao, Z. (2019). Deep learning review and discussion of its future development. *MATEC Web of Conferences*, *277*, 02035. https://doi.org/10.1051/matecconf/201927702035

[HE16] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 770–778). Las Vegas, NV, USA.

[HED13] Hedjam, R., & Cheriet, M. (2013). Historical document image restoration using multispectral imaging system. Pattern Recognition, 46(8), 2297-2312.

[HUA15] Huang, J.-B., Singh, A., & Ahuja, N. (2015). Single image super-resolution from transformed self-exemplars. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 5197-5206).

[JAS13] Jassim, F. A. (2013). Kriging Interpolation Filter to Reduce High Density Salt and Pepper Noise. *World of Computer Science and Information Technology Journal (WCSIT)*, *3*(1), 8-14.

[JIA08] Jianchao, Y., Wright, John, Huang, Thomas, & Yi Ma. (2008). Image super-resolution as sparse representation of raw image patches. *2008 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1-8). IEEE.

[JOS21] Josep, L., Daniel, L., & Seiichi , U. (2021). 16th International Conference on Document Analysis and Recognition, ICDAR 2021, Lausanne, Switzerland, September 5-10, 2021, Proceedings, Part II. *12822*. Springer.

[KRI12] Krizhevsky, A. (2012, May). Learning Multiple Layers of Features from Tiny Images. *University of Toronto*.

[KRI10] Krizhevsky, A., Sutskever, I., & Geoffrey, E. (2010). ImageNet Classification with Deep Convolutional Neural Networks. *In Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*, (pp. 1097-1105).

[LIN10] Lins, R., Banerjee, S., & Thielo, M. (2010). Automatically detecting and classifying noises in document images., (pp. 33-39).

[MA18] Ma, N., Zhang, X., Zheng, H., & Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient CNN architecture design. *European Conference on Computer Vision (ECCV)*, (pp. 116–131). Munich, Germany.

[MAR15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, … Xiaoqiang Zheng. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. https://www.tensorflow.org/

References

[MAR23] Margaret, R. (2023, June 17). *Artificial Neural Network*. Retrieved from Techopedia: https://www.techopedia.com/definition/5967/artificial-neural-network-ann

[MIE22] Mienye, D., & Sun, Y. (2022). A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects. *IEEE Access, PP*, 1–1. https://doi.org/10.1109/ACCESS.2022.3207287

[MIJ21] Mijwil, M. (2021, Aout). Artificial Neural Networks: Advantages and Disadvantages. *Mesopotamian Journal of Big Data*, 29-31.

[MIK18] Mikołajczyk, A., & Grochowski, M. (2018). Data augmentation for improving deep learning in image classification problem. *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, 117–122. https://doi.org/10.1109/IIPHDW.2018.8388338

[NIT14] Ntirogiannis, K., Gatos, B., & Pratikakis, I. (2014). ICFHR2014 Competition on Handwritten Document Image Binarization (H-DIBCO 2014). *2014 14th International Conference on Frontiers in Handwriting Recognition*, 809–813. https://doi.org/10.1109/ICFHR.2014.141

[PAS19] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[PAW22] Pawar, P., Ainapure, B., Rashid, M., Ahmad, N., Alotaibi, A., & Alshamrani, S. (2022, September). Deep Learning Approach for the Detection of Noise Type in Ancient Images. *Sustainability, 14*.

[PRA10] Pratikakis, I., Gatos, B., & Ntirogiannis, K. (2010). H-DIBCO 2010—Handwritten Document Image Binarization Competition. *2010 12th International Conference on Frontiers in Handwriting Recognition*, 727–732. https://doi.org/10.1109/ICFHR.2010.118

[PRA11] Pratikakis, I., Gatos, B., & Ntirogiannis, K. (2011). ICDAR 2011 Document Image Binarization Contest (DIBCO 2011). *2011 International Conference on Document Analysis and Recognition*, 1506–1510. https://doi.org/10.1109/ICDAR.2011.299

[PRA12] Pratikakis, I., Gatos, B., & Ntirogiannis, K. (2012). ICFHR 2012 Competition on Handwritten Document Image Binarization (H-DIBCO 2012). *2012 International Conference on Frontiers in Handwriting Recognition*, 817–822. https://doi.org/10.1109/ICFHR.2012.216

[PRA13] Pratikakis, I., Gatos, B., & Ntirogiannis, K. (2013). ICDAR 2013 Document Image Binarization Contest (DIBCO 2013). *2013 12th International Conference on Document Analysis and Recognition*, 1471–1476. https://doi.org/10.1109/ICDAR.2013.219

[PRA16] Pratikakis, I., Zagoris, K., Barlas, G., & Gatos, B. (2016). ICFHR2016 Handwritten Document Image Binarization Contest (H-DIBCO 2016). *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 619–623. https://doi.org/10.1109/ICFHR.2016.0118

[PRA17] Pratikakis, I., Zagoris, K., Barlas, G., & Gatos, B. (2017). ICDAR2017 competition on document image binarization (DIBCO 2017). *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)* (pp. 1395–1403). Kyoto, Japan: IEEE.

References

[PRA18] Pratikakis, I., Zagori, K., Kaddas, P., & Gatos, B. (2018). ICFHR 2018 Competition on Handwritten Document Image Binarization (H-DIBCO 2018). *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 489–493. https://doi.org/10.1109/ICFHR-2018.2018.00091

[QAD18] Qadi, Z. A. (2018). Salt and Pepper Noise: Effects and Removal. *International Journal on Electrical Engineering and Informatics*.

[RAB13] Rabeux, V. (2013). *Évaluation de la qualité des documents anciens numérisés.* Université de Bordeaux 1, Labri (Laboratoire Bordelais de Recherche en Informatique).

[REH12] Rehman, A., & Saba, T. (2012). Document image preprocessing: a brief survey. In 2012 International Conference on Multitopic Conference (INMIC) (pp. 193-196). IEEE.

[RNN] Classification and prediction of wave chaotic systems with machine learning techniques - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Generalized-recurrent-neural-network-architecture-with-two-hidden-layers-The-NN_fig3_335159004 [accessed 9 Jun, 2024] .

[SAD15] Saddami, K., Munadi, K., & Arnia, F. (2015). A database of printed Jawi character image. *2015 Third International Conference on Image Information Processing (ICIIP)* (pp. 56–59). Waknaghat, India: IEEE.

[SAD17] Saddami, K., Munadi, K., Away, Y., & Arnia, F. (2017). DHJ: A database of handwritten Jawi for recognition research. *2017 International Conference on Electrical Engineering and Informatics (ICELTICs)* (pp. 292–296). Banda Aceh, Indonesia: IEEE.

[SAN18] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 4510–4520. https://doi.org/10.1109/CVPR.2018.00474

[SHA13] Shamqoli, M., & Khosravi, H. (2013). Border detection of document images scanned from large books. *2013 8th Iranian Conference on Machine Vision and Image Processing (MVIP)*, (pp. 84-88).

[SHA18] Shahkolaei, A., Beghdadi, A., Al-maadeed, S., & Cheriet, M. (2018). MHDID: A Multi-distortion Historical Document Image Database. *2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR)*, 156–160. https://doi.org/10.1109/ASAR.2018.8480372

[SIM14] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv 1409.1556*.

[SIM15] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition.

[SPI97] Spitz, A. L. (1997). Determination of the script and language content of document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(3), 235–245.

[SZE16] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016, June). *Rethinking the Inception Architecture for Computer Vision*. https://doi.org/10.1109/CVPR.2016.308

[TAN21] Tan, L., & Ann, D. (2021). Bayesian damage recognition in document images based on a joint and local homogeneity model. *Pattern Recognition, ISSN 0031-3203, 118*, 108034.

References

[VU21] Vu, M., Le, V., & Marie, B.-A. (2021). *IHR-NomDB: The Old Degraded Vietnamese Handwritten Script Archive Database* (pp. 85–99). https://doi.org/10.1007/978-3-030-86334-0_6

[WOL92] Wolpert, & H, D. (1992). Stacked generalization. *Neural networks*, 241-259.

[XIE17] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 1492–1500). Honolulu, HI, USA.

[YU18] Yu, Z., & Qiang, Y. (2018). A Survey on Multi-Task Learning. *In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, (pp. 2296-2302).