LEVERAGING TEXT EXTRACTION AND LANGUAGE MODELS FOR COIN IMAGE CLASSIFICATION

BSc Santeri Vähämäki

MSc thesis
May 2025

Reviewers:
Dr. Luca Zelioli
MSc Adrian Borzyszkowski
Prof. Henri Nyberg

DEPARTMENT OF MATHEMATICS AND STATISTICS

UNIVERSITY OF TURKU
Department of Mathematics and Statistics

Vähämäki, Santeri: Leveraging text extraction and language models for coin image classification
MSc Thesis, 68 pages, 12 appendix pages
Statistics
May 2025

The process and classification of historical coins includes significant challenges. Despite advances in digital technology for cultural heritage (CH) artifacts, reliable automated methods for their identification are not perfectly effective. This thesis investigates the application of machine learning (ML) and deep learning (DL) techniques, and their numerical and statistical methods to enhance historical coin recognition and classification, aiming to improve authenticity verification and documentation.

This research uses a combination of classical pattern recognition (CPR) techniques, DL models, and natural language processing (NLP) approaches to analyze coin images. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are applied for feature extraction and classification, while Optical Character Recognition (OCR) models, such as Keras OCR and Convolutional Recurrent Neural Network (CRNN), support in text extraction. Furthermore, generative language models like GPT-3 or ChatGPT refine text recognition output and support coin classification.

The results indicate that CNN-based feature extraction enhances text visibility compared to traditional methods, but additional steps, including OCR and sequence modeling, are necessary for precise recognition. The CRNN model performs effectively on structured datasets, but struggles with real-world coin images due to background complexity and curved texts. Keras OCR improves text detection, particularly when preprocessing techniques such as random rotation are applied to straighten text regions. In addition, NLP-based correction using GPT-3.5 enhances the accuracy of extracted text and contributes to improved coin classification.

Keywords: Pattern Recognition, Classical Pattern Recognition, Local Binary Pattern, Edge Detection, Deep Learning, Feature Extraction, Convolutional Neural Network, Recurrent Neural Network, Text Extraction, Convolutional Recurrent Neural Network, Keras OCR, Generative Language Models.

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

Vähämäki, Santeri: Tekstin erottelun ja kielimallien hyödyntäminen kolikkokuvien luokittelussa
Pro gradu, 68 pages, 12 liites.
Tilastotiede
Toukokuu 2025

Historiallisten kolikoiden prosessointi ja luokittelu sisältävät merkittäviä haasteita. Kulttuuriperintöön (CH) liittyvän digitaaliteknologian kehityksestä huolimatta, ei ole vielä olemassa täysin tehokkaita ja luotettavia automatisoituja menetelmiä näiden tunnistamiseen. Tämä opinnäytetyö tutkii koneoppimisen (ML) ja syväoppimisen (DL) sovelluksia sekä näiden numeerisia ja tilastollisia menetelmiä parantaakseen historiallisten kolikoiden tunnistamista ja luokittelua, tavoitteena kehittää aitouden varmistamista ja dokumentointia.

Tutkimus käyttää klassisten kuviontunnistusmenetelmien (CPR), syväoppimismallien (DL) ja luonnollisen kielen käsittelyn (NLP) lähestymistapojen yhdistelmää kolikkojen kuvien analysointiin. Konvoluutioneuroverkkoja (CNN) ja toistuvia neuroverkkoja (RNN) sovelletaan piirteiden erotteluun ja luokitteluun, kun taas optisen merkintunnistuksen (OCR) mallit, kuten Keras OCR ja konvoluutio-toistuva neuroverkko (CRNN), tukevat tekstin erottelua. Lisäksi generatiiviset kielimallit, kuten GPT-3 tai ChatGPT, tarkentavat tekstintunnistuksen tuloksia ja tukevat kolikkojen luokittelua.

Tulokset osoittavat, että CNN-pohjainen piirteiden erottelu parantaa tekstin näkyvyyttä verrattuna perinteisiin menetelmiin, mutta lisävaiheet, kuten OCR:n ja sekvenssimallinnus ovat tarpeellisia tarkkaan tunnistukseen. CRNN-malli toimii tehokkaasti rakenteellisissa aineistoissa, mutta kohtaa haasteita reaalimaailman kolikkokuvissa taustan monimutkaisuuden ja kaarevien tekstien vuoksi. Keras OCR parantaa tekstin tunnistusta erityisesti silloin, kun esikäsittelytekniikoita, kuten satunnaista kiertoa, sovelletaan suoristamaan tekstialueita. Lisäksi NLP-pohjainen korjaus GPT-3.5-mallin avulla parantaa poimitun tekstin tarkkuutta ja edistää kolikoiden luokittelun kehitystä.

Avainsanat: kuviontunnistus, klassinen kuviontunnistus, paikallinen binaarinen kuvio (LBP), reunojen tunnistus, syväoppiminen, piirteiden erottelu, konvoluutioneuroverkko, toistuva neuroverkko, tekstin erottelu, konvoluutio-toistuva neuroverkko, Keras OCR, generatiiviset kielimallit.

Contents

Abbreviations

AI Artificial Intelligence

BPTT Backpropagation Through Time

BRNN Bidirectional Recurrent Neural Network

CDF Cumultative density function

CRAFT Character Region Awareness for Text Detection

CH Cultural Heritage

CLAHE Contrast Limited Adaptive Histogram Equalization

CNN Convolutional Neural Network

CTC Connectionist Temporal Classification

CRNN Convolutional Recurrent Neural Network

CPR Classical pattern recognition

DCNN Deep Convolutional Neural Network

DL Deep learning

GPT Generative Pre-trained Transformer

LBP Local Binary Pattern

LSTM Long Short-Term Memory

ML Machine Learning

MLP Multilayer preceptron

NLP Natural Language Processing

OCR Optical Character Recognition

PCR Principal component analysis

PDF Probability density function

PR Pattern recognition

ReLU Rectified Linear Unit

ROI Region of the Interest

RTN Rotation Transformation Network

RNN Recurrent Neural Network

SS Selective search

SGD Stochastic gradient decent

1 Introduction

In recent years, the investigation of cultural and historical goods has become popular due to the availability through digital libraries and the internet [1]. This field became relevant in preventing the illicit trafficking of Cultural Heritage (CH) goods. Among these goods, historical coins represent a particularly rich and complex category. They include not only monetary or economic value but also substantial cultural information in their inscriptions, materials, and symbols. Consequently, the digital analysis of historical coins plays an important role in heritage preservation and academic research. [1][2]

Machine learning (ML), and especially Deep learning (DL), has become an essential tool in addressing complex classification and Pattern Recognition (PR) in image data and they can be analyzed using statistical techniques. Models like Convolutional Neural Networks (CNNs) [2][3][11] and Convolutional Recurrent Neural Networks (CRNNs) [4][5] are widely used to extract and classify features from images. Additionally, Natural Language Processing (NLP) has introduced generative language models such as GPT-3 and ChatGPT which are capable of interpreting, generating, and correcting text in a variety of domains and classify images based on them.

The combination of these methods offers new possibilities for analyzing historical artifacts, particularly for image classification and text extraction from ancient coins. However, several challenges remain. Historical and cultural coin images contain complexity and unclear features that are difficult to recognize [2]. Their inscriptions are frequently curved, fragmented, or obscured, which reduces the effectiveness of standard PR [11][16] and Optical Character Recognition (OCR) [7][8][12][13] tools. Although DL models have achieved remarkable results on image datasets, their performance in real-world cultural heritage scenarios remains insufficiently explored. Moreover, the ability of generative language models to enhance or correct the extracted text produced by DL models is an open question.

This thesis investigates how modern DL methods and NLP models can be applied to the classification and text recognition of historical coin images. The focus is especially on whether modern OCR models and language models improve text extraction from degraded historical coin images. In addition, the study explores how well DL models (e.g., CNN, CRNN) perform in classifying real-world coin images compared to Classical Pattern Recognition (CPR) methods, and whether GPT-based correction can improve the accuracy of OCR results on historical coins. To guide the research, the research questions are formulated as follows:

- RQ1: How do DL models (e.g. CNN, CRNN) perform in classifying real-world coin images compared to Classical Pattern Recognition (CPR) methods?
- RQ2: Can OCR models improve text extraction from degraded historical coin images?
- RQ3: Does GPT-based correction improve the accuracy of OCR results on historical coins?

To answer these questions, the work focuses on evaluating the performance of DL models in feature recognition, classification, and text extraction tasks and evaluating

whether generative language models can improve the quality of text extracted from degraded coin inscriptions. Specific attention is paid to the challenge of curved and fragmented text and how pre-processing techniques such as rotation [8][11][15] or contrast enhancement may improve OCR accuracy. The evaluation includes the use of metrics such as classification accuracy and cosine similarity to compare textual output.

The main point of this work lies in demonstrating the practical effectiveness of combining visual DL models with modern NLP techniques in a context of real-world cultural heritage. In particular, it explores how GPT-based models may act as a post-processing layer to refine and improve OCR outputs when dealing with historical coin images. The findings offer valuable information for museums and researchers working with digitized historical objects, as well as for the broader field of AI-driven heritage analysis.

1.1 Motivation of thesis

The primary motivation for this research is to investigate useful methods to detect counterfeit coins and track stolen cultural artifacts. Many illicit transactions occur through online marketplaces, where traditional verification methods are insufficient. By integrating image-processing techniques, text extraction, and generative language models with statistical methods, this study aims to develop an advanced PR system capable of distinguishing authentic historical coins from forgeries and interpreting their text symbols.

Moreover, existing approaches to coin classification often rely on hand-crafted feature extraction methods, which may not generalize well across different datasets. DL models like CNNs, CRNNs and Keras OCR offer a more developed solution to analyze coin images. Furthermore, NLP techniques, including cosine similarity and language models such as GPT-3 or ChatGPT, can support text recognition and interpretation in cases where inscriptions are unclear or fragmented.

The significance of this research extends beyond academia. Museums, numismatists, historians, and law enforcement agencies can benefit from an improved methodology for coin identification and authentication. By refining image preprocessing techniques and leveraging state-of-the-art DL algorithms, this study aims to contribute to ongoing efforts in digital cultural heritage preservation and security.

Ultimately, this thesis aims to explore the potential of combining CPR methods with DL and NLP models to improve the classification and identification of historical coins. By doing so, it aims to advance technological solutions for cultural heritage protection and facilitate more effective digital documentation and analysis of historical artifacts.

1.2 Thesis structure

This thesis consists of 6 chapters. Chapter 1 describes a brief introduction and the motivation of the thesis. Chapter 2 includes the theoretical background of PR and DL methods. Chapter 3 reports our study of the coin image. Details of the coin image dataset are also described in this section as well. The chapter also describes

the role of PR and DL methods. Chapter 4 summaries the results made in Chapter 3. Chapters 5 and 6 give a brief summary of the thesis and the discussion and future improvements of the study.

2 Pattern recognition

PR plays a central role in the analysis of image data, allowing machines to identify, categorize, and interpret complex patterns such as symbols, texts, and objects in images. This chapter explores the principles of PR and its applications in the context of cultural heritage image analysis. The discussion begins by outlining the general structure of PR systems, including data collection, pre-processing, feature extraction, and classification. Both classical and modern approaches are presented: CPR relies on statistical and texture-based feature methods, while DL methods, such as Convolutional and Recurrent Neural Networks, offer more effective alternatives that improve accuracy and scalability. In particular, advanced DL techniques such as CRNN and Keras Optical Character Recognition (Keras OCR) are highlighted for their effectiveness in extracting and recognizing text from complex image data. Special attention is paid to the extraction and classification of symbolic patterns such as numerical digits and text from coin images, where robust and interpretable solutions are required. By integrating theoretical foundations with practical techniques, this Chapter aims to provide a comprehensive overview of PR methodologies and their significance in supporting digital analysis of historical and cultural artifacts.

2.1 What is Pattern Recognition?

PR is essential for real-world applications, such as medical analysis, image recognition, and classification. This field focuses on developing an algorithm that tests to identify features and regularities in the data. PR can be combined with statistical analysis methods to discover patterns and regularities due to rough discoveries in the data. For example, decision trees, nearest neighbors, or neural networks can be analyzed to discover patterns and regularities in the data. In addition, PR systems may act as feature extractor and selector. Recognizing symbols and texts in images requires robust and efficient techniques. [11][16]

PR is used in tasks including object classification, semantic segmentation, and event recognition. In image analysis object classification focuses on determining the class to which an image belongs, and in object segmentation, the algorithm may isolate objects of interest by identifying their pixel boundaries. Advanced PR methods, particularly region-based approaches, are highly effective in extracting text and symbols from images. These techniques focus on analyzing specific areas within an image by identifying regions of interest (ROIs) enclosed in bounding boxes, which highlight detected objects for further analysis. [11]

Moreover, the extracted features can be used as inputs for other models. PR shares many similarities with statistics. Statistical pattern recognition uses statistical methods to learn the structures from observed data based on extracted features, which are then used to create a model and the probability distribution to other patterns based on the observed patterns. To classify unseen patterns, the created model identifies relationships between new data points and previously observed data points. [11][16]

PR includes versatile tools for identifying and categorizing patterns such as objects and events. Objects such as text characters and numerical symbols are tangible

entities characterized by their specific shape, form, and structure. PR methods are effective in recognizing these entities, which makes them widely applicable in text and number recognition tasks, such as OCR [13]. Events are defined as occurrences or happenings that involve interactions between multiple objects. While text and number recognition primarily focuses on static objects, PR techniques used for event detections may analyze temporal sequences to identify patterns of interaction. These distinctions highlight the adaptability of PR methods across different application domains that can be analyzed to detect text symbols in images. [11]

2.2 Classical PR Methods

CPR follows a systematic approach in which the raw image data is processed to extract suitable features. The goals of feature extraction in CPR is to identify the most informative data, to remove redundancy, and to create classes that have small variations between the class members. This process relies on a predefined set of features that are formed to the specific investigation. Feature engineering is not a straightforward process, and it varies significantly depending on the dataset and the application. The effectiveness of CPR often depends on the quality and relevance of the selected features of the images. [11] Based on these, the CPR is based on the process shown in Figure 1.

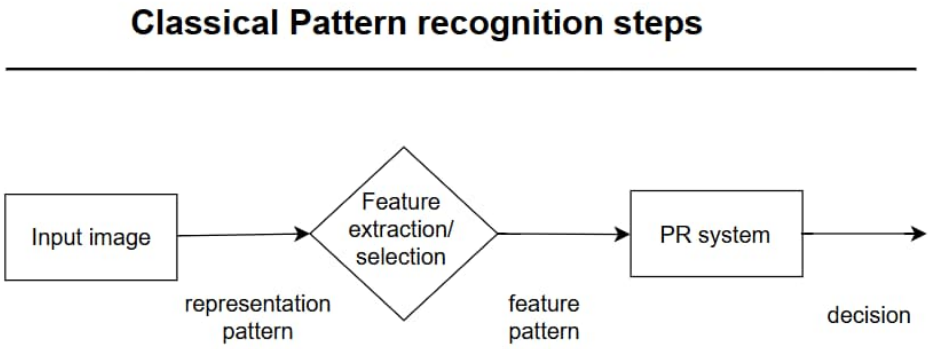


Figure 1: The steps of classical pattern recognition (CPR).

In CPR features are typically extracted using statistical or texture-based methods (in Figure 2). In the statistical case, the useful methods in feature extraction are, for example, the mean, variance, standard deviation, skewness, kurtosis and median. These statistical measures provide information about the distribution and characteristics of the input data. The mean represents the average intensity of the pixel values in a given region. The variance shows how much the values deviate from the mean. The standard deviation indicates how spread the values are. The skewness specifies the asymmetry of the distribution, showing whether the values tend to cluster more on one side of the mean. Kurtosis tells how peaked or flat the distribution is. The median is the middle value in the data. In addition, different spatial filters, such as wavelengths, can be analyzed to create features, particularly in image-based recognition. These fundamental methods offer simple and effective ways to understand the nature of images and support classification tasks. [11]

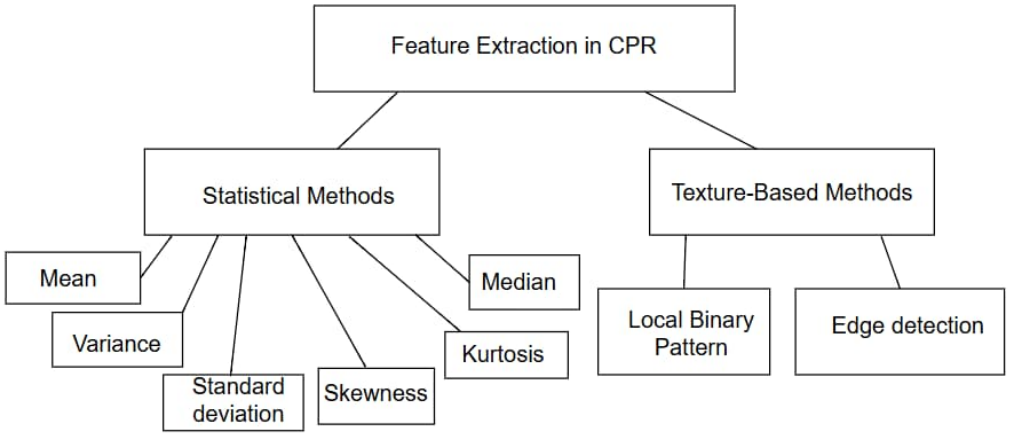


Figure 2: Example methods to extract features from images.

The above statistical features may be sufficient for simple applications. However, there are cases where they are not as effective. For example, in coin images, more sophisticated features like *texture features* are necessary when the goal is to extract text symbols. A texture refers to the spatial arrangement and distribution of patterns in the input image [11]. Texture features are particularly valuable in image processing as they capture the regularity or repetition of patterns within an image. These features can be analyzed to derive properties such as (a) contrast, which measures the intensity differences between neighboring pixels

$$\Sigma_i^{N_g}\Sigma_j^{N_g}(i-j)^2P(i,j),$$

(b) correlation, which examines the relationship between different portions of the image

$$\Sigma_i^{N_g}\Sigma_j^{N_g}\frac{(i-\mu_i)(j-\mu_j)P(i,j)}{\sigma_i\sigma_j},$$

(c) the similarity of neighboring pixels

$$\Sigma_i^{N_g}\Sigma j^{N_g}\frac{P(i,j)}{1+|i-j|}$$

and (d) the uncertainty of neighbouring pixels

$$-\Sigma_i^{N_g}\Sigma_j^{N_g}P(i,j)log\ P(i,j).$$

Here $P(i,j)$ represents the Gray Level Co-occurence Matrix (GLCM) value of the image in the row i and column j, N_g the number of shades of gray, μ_i and μ_j the means of the marginal distributions of the GLCM and σ_i and σ_j their standard deviations. [11][24]

One of the popular texture extraction methods used in this study is the Local Binary Pattern (LBP). It is used due to the computational simplicity of the process.

The goal of the LBP is to represent the local texture of the image. Before using the LPB, the image must be converted to a grayscale image, because it helps to discover the intensity of each pixel of the input image. LBP specifically focuses on describing local textures by defining a threshold and comparing the intensity of each pixel within a circular neighborhood to the central pixel. If the intensity of a neighboring pixel is greater than the central pixel, marked as 1. Otherwise, it is marked as 0. This process is called binary notation. The binary values of all neighbors are then concatenated and converted into a decimal number, which serves as the local texture pattern of the central pixel. This procedure is repeated for every pixel in the image, effectively capturing its local texture features. [9][11]

Edge detection methods are used to identify the textures of the images. One of these methods in this study is *Gabor filters*, which serve to reveal distinctive content in specific directions. Gabor filters are particularly effective in analyzing the spatial and frequency characteristics of input data. These filters can be used as convolutional kernels to reveal unique features within an image, making them especially useful for texture analysis. As the number of Gabor filters applied to an image increases, so does the ability to discover a wider variety of features. Furthermore, Gabor wavelets can be utilized to capture both spatial and frequency information simultaneously, providing detailed insights into the scale and orientation decomposition of an image. [10][11]

2.2.1 Local Binary Pattern

The LBP operator is a theoretically simple and powerful method for analyzing the textures of images [18]. As mentioned above, it is used because of the computational simplicity of the process and the goal is to represent the local texture of the image. The LBP operator is computationally complex and includes traditionally different statistical and structural models to extract the features of the images. This section shows in more detail how the LBP combines aspects of statistical and structural texture analysis and why it is called a unifying approach. The main point is to investigate the local neighborhood pixels of the images and many other extensions. [9][11][18]

The input data must be preprocessed before applying the LBP algorithm. This transformation enables the analysis of local spatial patterns and grayscale contrast. As mentioned above, the original LBP operator forms labels for the image pixels by thresholding the 3×3 neighborhood of each pixel with the center value and considering the result as a binary number [18]. It can be extended to use also neighborhoods of different sizes surrounding the center pixel. The useful parameters to define this are a circular neighborhood (the number of surrounding points) p and a radius r. [9][18]

If the fixed 3×3 neighborhood pixel matrix is

$$X = \begin{pmatrix} 47 & 51 & 65 \\ 62 & 70 & 70 \\ 80 & 83 & 78 \end{pmatrix}$$

and it is completed LBP image, the first step is to take the center pixel (70) and threshold it against its neighborhood of 8 pixels. If the intensity of the center pixel

is greater than or equal to its neighbor, then we set the value to 1 and otherwise set it to 0. After this the pixel matrix is

$$X = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 70 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

After this, the LBP value is calculated for the center pixel (70) by multiplying by powers of two and sum as follows: $1 \cdot 1 + 1 \cdot 2 + 1 \cdot 4 + 0 \cdot 16 + 0 \cdot 32 + 0 \cdot 64 + 0 \cdot 128 = 15$. The center pixel (70) is then replaced by value 15. More formally the LBP operator for a pixel (x, y) is defined as:

$$LBP_{P,R}(x, y) = \Sigma_{p=0}^{P-1} s(g_p - g_c) \cdot 2^p, \tag{1}$$

where g_c is a gray-level value of the center pixel, g_p are gray-level values of the P neighbors on a circle of radius R on a circle of radius R and $s(x)$ is the thresholding function [9][18][19]:

$$s(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \tag{2}$$

After computing the LBP values for each pixel in the pixel matrix, a histogram can be constructed to investigate the texture distribution of the image. The histogram is given by:

$$H_i = \Sigma_{x,y} I\{f_L(x, y) = i\}, i = 0, ..., n - 1, \tag{3}$$

where n is the number of different labels produced by the LBP operator and indicator function $I\{A\}$ is 1 if A is true and 0 otherwise. When comparing texture histograms from different image regions, histogram normalization is necessary to ensure a consistent representation:

$$N_i = \frac{H_i}{\Sigma_{j=0}^{n-1} H_j} \tag{4}$$

This normalized histogram provides a rotation- and scale-invariant texture description, making LBP highly effective for texture investigation. Since in the original image 3×3 neighborhood has $2^8 = 256$ possible patterns, the LBP 2D array thus has a minimum value of 0 and a maximum value of 255. This means that the constructed histogram of the LBP image contains 256 bins. [9][18]

The original LBP method can capture exact and fine-grained details in images that contain different textures and patterns. There are anyway certain images that contain small-scale details such as patterns. This is one of the biggest drawbacks in LBP method, because it can capture details only on the fixed 3×3 scale, not on varying scales. This might appear in the historical and cultural coin images, where some text patterns might be unclear and at small scales. To solve this problem, it is useful to handle the neighborhood size and radius parameter mentioned above, which allows to account for different scales in the images. [9]

2.2.2 Edge detection (Canny)

Edge detection methods offer a powerful tool for texture analysis by capturing both spatial and frequency characteristics of an image. For example, Gabor filters have the ability to highlight specific orientations and frequencies, which makes them highly effective in detecting edges, patterns, and structural details. This is useful in coin image investigation, for example, when access is needed to clarify the text symbols and patterns to extract them clearly. This section focuses on mathematical foundation of Gabor filters, their implementation in image processing and their role in feature extraction in CPR. By adjusting key parameters such as wavelength, orientation, and phase offset, it is possible to fine-tune the filters to enhance specific image features. Furthermore, the combination of multiple Gabor filters allows for a more comprehensive representation of textures, enabling a more detailed image analysis. [10][11]

Gabor filters are mathematically formulated as sinusoidal plane waves. This formulation ensures that filters are localized in both spatial and frequency areas, which aligns the properties of early visual processing in the human visual system. The two-dimensional Gabor function can be expressed as:

$$g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2})\ cos(2\pi\frac{x'}{\lambda} + \psi) \tag{5}$$

where $x' = x\ cos(\theta) + y\ sin(\theta)$ and $y' = -x\ sin(\theta) + y\ cos(\theta)$. In the above equation, the parameters have the following roles:

- λ (wavelength) controls the frequency of the sinusoidal component.
- θ (orientation) determines the angle of the filter.
- ψ (phase offset) shifts the sinusoidal function.
- σ (standard deviation) defines the spatial extent of the Gaussian envelope.
- γ (spatial aspect ratio) adjusts the ellipticity of the filter.

By tuning these parameters, Gabor filters can be adapted to detect specific textual features, such as edges, ridges, and frequency variations in different orientations. Their ability to decompose an image into multiple frequency and orientation components makes them particularly effective in texture segmentation tasks. [10][20]

Gabor filters can be used as part of DL models when they are used as convolutional kernels to reveal features from images and classify them. The ability of Gabor filters to distinguish key features, such as texture and edge characteristics, makes them valuable for enhancing feature extraction for CNNs. Since deep CNN models require a large number of training samples to prevent overfitting, the use of Gabor filters can reduce this need by providing pre-extracted discriminative features. This helps CNN models and other DL models focus on higher-level feature analysis, improving generalization and classification accuracy, especially in cases where training samples are limited. [11][21]

2.3 Deep learning in PR

One of the most effective methods that can be analyzed to automate the PR problem in text and symbol detection is DL. Among the most widely used DL architectures are CNNs and its variants, which rely on convolutional layers to process and analyze input data. These layers use complex filters to generate feature maps that represent both the location and accuracy of the detected objects. Depending on the task to be tackled, different DL methods can be applied. For example, the Recurrent Neural Network (RNN) can be widely used for text generation and sequence prediction tasks, such as handwriting recognition and machine translation [4][5][22][21]. Similarly, Keras OCR can be effectively applied to text extraction from images using CRNN [6][12][13]. By automating feature extraction and classification, DL offers the fastest way to recognize and extract features of the images compared to manual PR methods. [2][11]

DL models are inspired by biological learning, in which information moves through different cell structures called neurons. Many of these algorithms can easily discover different features from the images using numerical and statistical methods such as filters, activation functions, and principal component analysis (PCA), and compare them with each other. The application of DL models and statistical analysis enables not only the identification of objects from images, but also the extraction of unique features such as digits, letters, and symbols which can support historical and provenance research. For example, in the case of coin images, DL models can identify specific features such as text symbols and patterns that are not easy to recognize with traditional PR methods. This approach improves the accuracy of categorization and contributes to a greater understanding of the cultural heritage. [2][11]

DL can perform many different types of tasks such as object localization, semantic segmentation, and object classification [2][3][11]. In semantic segmentation, the algorithm predicts the category of each image pixel, dividing the images into multiple classes of segments [11]. It is often based on auto-encoders, which encode and decode image features for precise segmentation. This method enables precise boundary detection in region-based approaches, studies patterns in images, and reveal regions of images that contain specific objects. Since semantic segmentation assigns a specific label to each pixel, it allows for a detailed classification of image content into predefined categories. These detected regions are often enclosed by bounding boxes, and the method is widely used in complex DL models. [11]

Object classification is a fundamental task in DL, where a model assigns an image or its specific regions to predefined categories. Unlike semantic segmentation, which classifies each pixel, object classification focuses on identifying and labeling entire objects within an image. CNNs have demonstrated good performance in classification tasks by extracting hierarchical features from input images. These models process visual information through multiple layers by identifying edges, textures, and complex structures that distinguish different classes of objects. Object classification plays a crucial role in various applications, including medical diagnosis, automated quality control, and autonomous systems, where precise recognition of objects is essential for decision making and further analysis. [2][3][11]

2.3.1 CNN

In object localization and classification tasks, one of the most popular and effective DL models is CNN, whose purpose is to process and analyze image data by identifying hierarchies and features within the image. The structure of the CNN is more complex compared to multilayer perceptron (MLP) neural networks and other pattern recognition models, and there is an input layer and an output layer and between them multiple hidden layers. In addition, the hidden layers are connected by convolutional layers and fully connected layers to learn complex patterns from images. As seen in Figure 3, the convolutional layers apply filters to the input images capturing essential features, and they can be followed by the batch normalization layer and the pooling layer. Thanks to these hierarchical approaches, the DL can perform well in image-related tasks and achieve high accuracy in recognizing and classifying objects within images. In addition of object detection, it has ability to demonstrate superior performance across various applications including audio classification, time series analysis, signal data analysis, etc. [2][3][11]

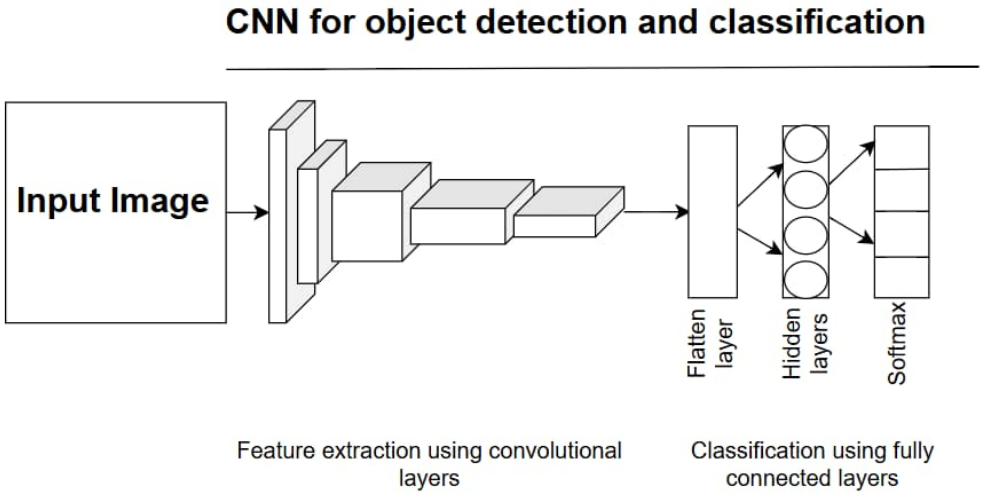


Figure 3: A classical Convolutional Neural Network (CNN) architecture used for classification purposes.

Compared to CPR tasks, CNNs can learn many details when the number of hidden layers and neurons is applied accordingly. This method is called HP tuning, where multiple parameters are tested until the best possible combination is found that accurately improves the result. In this case, each layer takes input from the previous layer and passes information to the next layer, allowing the network to capture increasingly complex patterns. One of the characteristics of CNN is that it can capture details such as textures, edge variations, and shapes that are characteristics of specific objects, which distinguish coins with similar designs by recognizing even slight differences. In addition, CNN can reduce the impact of variations in lighting, rotation, and background noise, which are common in coin images. [2][3][11][12]

If used properly, CNN is useful to tackle the image recognition problem with a high classification accuracy value. During CNN training, the neurons of the layers have weights that are statistically adjusted when the model learns new features of

the images and improves the classification accuracy. CNNs can also learn the most important features from the images and manual feature extraction is not needed like in the CPR models. Instead, they utilize their deep structure to identify relevant features during the training process. Thanks to these abilities, neural network models can be used in any other pattern and image recognition application such as gesture recognition, face recognition, object classification, and generating scene description. The success of this recognition rate is, for example, due to the algorithms developed, the large number of hidden layers and neurons, and effective statistical activation functions. The number of alternative CNN models is big based on different hyperparameter values and many other architectures, and these can affect their feature extraction and image recognition abilities. [2][3][12]

One of the useful applications where the CNN model can be applied in the investigation of cultural and historical coin images is an OCR task. OCR can extract texts from difficult and complex images. The feature extraction methods of this type of CNN is to gather letters and numbers from input images. OCR is useful in extracting textual patterns from cultural heritage goods [13]. Some of the effective CNN architectures used in OCR tasks include LetNet and AlexNet, which employ various functions and algorithms to optimize model performance. [12]

It has also been developed pre-trained models that are already trained with some big image data and must be fine-tuned with smaller data. This advancement is useful in the classification of coin images, especially if the amount of coin image data with their labels and the classification accuracy value of the model is too small. [2][3]

2.3.2 CNN architecture

A reliable CNN architecture includes one or more inputs, a combination of hidden layers, and one or more outputs. In image classification, a Softmax activation layer is usually included, whereas in object detection, a linear activation regressor is used. In a CNN, hidden layers are the layers between the input and the output. They are responsible for extracting and transforming features from the input data (usually images) as the data moves deeper through the network. These layers learn increasingly abstract and complex patterns. In addition, the convolutional layer can be followed by a batch normalization layer and a pooling layer [11]. There exists also a useful activation functions that the convolutional and fully connected layers can be analyzed to access useful features and a good accuracy rate in challenge coin image classification. Examples of activation functions that can be used in the model are Max pooling, ReLU, Softmax and Sigmoid. These methods are the most useful key operations especially in coin image classification. [3][12]

Convolutional layers apply filters (e.g. 3×3 or 5×5 matrix) to input matrices to detect local features such as edges or textures. For inputs of shape $H^l \times W^l \times D^l$, filters of the same size slide spatially to produce a single output value per location by element-wise multiplication and summation [2][11][14]. *Batch normalization* normalizes intermediate outputs to stabilize training and accelerate convergence. It adjusts values after convolution and activation layers to reduce internal covariate shift and complements pooling layers that summarize features through max or average operations [11]. *Max pooling* selects the maximum value from $k \times k$ window, emphasizing dominant features and reducing noise. This is statistically similar to

non-parametric feature selection or the maximum likelihood principle. It can be expressed as:

$$X_{out} = max(X_{ij}) \tag{6}$$

where X_{ij} represents the pixels within the k x k window [2][3][12][14]. *Rectified Linear Unit (ReLU)* is non-linear transformation, which removes non-informative (negative) signals, aiding in efficient learning. The mathematical formula can be represented as:

$$f(x) = max(0, x) \tag{7}$$

where x is the input. [2][3][12][14] *Softmax* converts CNN output into a probability distribution over multiple classes. It ensures that the sum of probabilities in all classes is equal to 1. For a vector z of class scores, Softmax is defined as:

$$Softmax(z_i) = \frac{e^{z_i}}{\Sigma_j e^{z_j}} \tag{8}$$

This transforms the unnormalized logit values z_i into probabilities where e^{z_i} is the exponent of each class score. [2][3][12][14] *Sigmoid* is useful in binary classification cases, mapping inputs to (0, 1):

$$y = \sigma(x) = \frac{1}{1 + exp(-x)}, \tag{9}$$

where the output y lies within the range $0 < y < 1$. Its derivative $\frac{dy}{dx} = y(1 - y)$ limits the size of the gradient, leading to vanishing gradients in deep networks. This is one reason why the ReLU activation function has become the preferred in modern CNN [14]. The example of batch normalization, max pooling and Relu is illustrated in Figure 4.

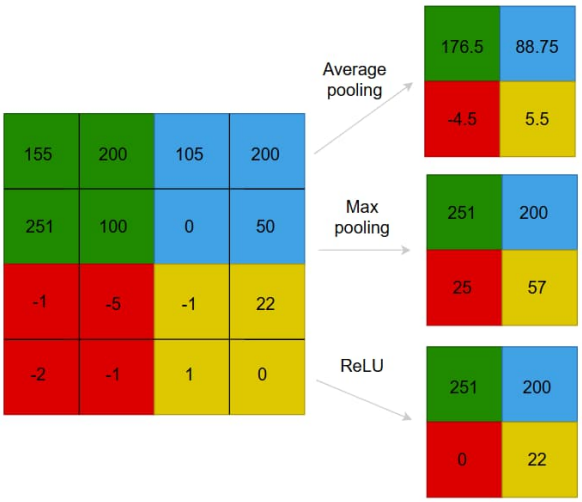


Figure 4: Operation of the Average Pooling layer, Max pooling and Rectified Linear Unit (ReLU) to reduce the dimensionality.

With advancements in DL, modern models have increased significantly in depth and accuracy, enabling them to handle a growing number of classification tasks and recognize features more effectively. These models are capable of extracting features at multiple hierarchical levels: (1) Low-level features involve basic elements such as edges, colors, and simple shapes. (2) Mid-level features capture more complex structures such as patterns or combinations of shapes. (3) High-level features represent abstract concepts, such as identifiable objects or symbols, built upon the outputs of previous layers. [2][3]

However, excessively DL can lead to diminishing returns in performance and may suffer from overfitting. To mitigate this and improve accuracy, various techniques are employed, including filter concatenation, residual learning, and leveraging probability distributions across possible classes. Additionally, the use of descriptive statistics, such as the mean and standard deviation, helps in evaluating the model's output more efficiently. [2][3]

In image classification, the images are converted to numerical pixel values. Statistically, image pixel values can be described as a matrix $X \in \mathbb{R}^{HxW}$, where H is the number of rows and W is the number of columns. In addition, the images are set to the input layer in the CNN and the model can start to check the images statistically. The convolution operation is the beginning of CNN and involves sliding filters over the input image to detect the patterns and features of the image. In the images, the patterns and features can be, for example, textures and shapes. Color information is also very important in many image-based learning and recognition problems, and the dimensions of the image pixel matrices can be different depending on their colors. For grayscale images, a 2D input vector is used, while RGB images are processed across three channels (representing different colors), which helps the network to work with color information. This means that the grayscale image pixel values must be converted as matrix $X \in \mathbb{R}^{H \times W}$ and RGB images as matrix $X \in \mathbb{R}^{H \times W \times D}$. Because RGB images have three colors (R, G, B), $D = 3$ in this case. [2][14]

CNN can find different patterns from the images. These include 1) *Selective search* (SS) and 2) *Sliding Window* approach that allows CNN to recognize different image objects better by focusing on identifying initial candidate regions [11]. The details of these methods are described below:

Selective search (SS)

- In SS the CNN groups pixels of the input image based on how similar their features are (bottom-up approach). The similarity depends on, for example, the colors and texture, and the regions in the same cluster group are similar based on these features. The generated region groups have been formed; they are combined in a top-down process to create a hierarchy of the candidate regions. During each iteration of the algorithm, the quality of each candidate region is evaluated and a score is produced. If the score achieves a certain grade of accuracy, the candidate region becomes a Region of the Interest (ROI), and it can be compared to regions from pre-trained models to improve prediction. [11]

Sliding window

- In the Sliding window the CNN scans the entire image by moving a window across it, scoring each section as a potential ROI and adjusting the window size to detect objects of various dimensions, ensuring comprehensive coverage but with high computational cost. Because the window can be of different sizes, it can scale the input image to detect objects of varying sizes. The ones with better scores assume the value of objects of interest. In order to achieve better performances and fit objects of different dimensions, the input is re-scaled, and windows of different dimensions are applied. The Sliding Window approach ensures that all parts of the input are covered. [11]

The filters that CNN includes are called also kernels and during the training they can learn the patterns of the images by adjusting their weights during the training data. The convolution operation involves multiplying the input pixel values by weight kernels and scanning the entire image using a sliding window approach. This process allows the model to extract higher-level features from images, which are essential for classification and it can be written mathematically as follows:

$$Z = X * f \tag{10}$$

where X is the input pixel matrix of the image and f presents the weights of the filter layer. This product can be used with the Max Pooling (6), ReLU (7), or Softmax (8) activation functions to learn more specific features. [2]

The convolutional layers act as feature extraction blocks to analyze and extract hierarchical image features across multiple layers. There are different types of layer, such as convolutional layers, pooling layers, and dropout layers. Thanks to these structures, it can get into the deep details of images and collect useful patterns and characteristics. As the data progress through additional convolutional layers, more complex shapes, contours, and structures are extracted. Because the input pixel values of the images are multidimensional, a sliding method is used with specific size filters and layers to form a one-dimensional vector representation of the image. The filter passes through all values in the input pixel matrix and outputs the image pixels multiplied by a weight matrix (10). The model must also define how much it must jump forward to capture new values. After this the model summary the input pixel matrix to a new lower dimensional matrix or feature map. During training with the training data, each filter allows the model to refine feature detection better. [2][11][12]

At this point the input pixel matrix is converted to the one-dimensional vector using convolutional layers, pooling layers, and dropout layers the *embedding* of the specific image has been formed. This vector can be used as an input value in the fully connected layer part when the interest is to examine the category of the image. These are also called feature convolutional layers and fully connected layers. The fully connected layer is called as a classification section where the formed embedding vector is passed through the next hidden layers to learn complex relationships and correlation between features. The most popular and useful layer in the fully connected layer that provides probabilities for each class label based on the learned features and relationships is the Softmax layer with its activation function (8). In the

case of complex and difficult images, it is recommended to use a sufficient amount of hidden layers and neurons to get a good accuracy rate. [3][11][12]

2.3.3 RNN

In order to extract text and symbols from the images in a more accurate way, the RNN is designed to handle this kind of operations. This makes RNNs particularly effective in tasks that require context awareness, such as text recognition. RNN models can be used successfully especially in NLP tasks such as language modeling, text detection, and word embedding extraction. In addition, RNN are used in image-based sequence recognition tasks such as scene text recognition, and text extraction, which are useful in real-world application tasks and coin image investigation [4][5]. To do this, the access is to use statistical conditional probabilities which use the previous words to predict the next word. [4][5][22][23]

The RNN encoder-decoder model is the foundational architecture in sequence-to-sequence learning, commonly used for tasks such as machine translation, text summarization, and speech recognition. It consists of two RNNs: the encoder and the decoder. The encoder RNN processes the input sequence step by step, compressing the information into a fixed-size context vector that captures the sequence's most relevant features. This context vector is then passed to the decoder RNN, which generates the output sequence, one element at a time, based on the encoded information. This architecture is effective for handling variable-length input and output sequences, as it allows the model to learn temporal dependencies and relationships within sequences. However, its performance may degrade on long sequences due to the fixed size bottleneck of the context vector, which later models like attention mechanisms were introduced to overcome. [4][5][22][23]

This method is applied in both NLP tasks and image investigation. In NLP tasks, for example, the encoder forms an input sequence into a fixed length vector representation, which is called embeddings, and decoder forms a sequence from this representation. This model is valuable in many tasks where the access is to investigate text symbols and words. By jointly training the encoder and decoder to maximize the conditional probability of a target sequence given a source sequence, this architecture enhances the ability to capture linguistic regularities, making it a powerful tool in NLP applications. [23]

Like CNN model, RNN consists of input layers, hidden layers, and output layers. The difference compared to the CNN model is that it doesn't use feed forward NN architectures like CNNs. Instead of this, it uses cycles called recurrent in hidden layers, where information flows back and provides a form of memory of previous inputs $(x_1, ..., x_{t-1})$ as seen in figure 5. This means that the model learns a conditional probability distribution on a sequence $p(x_t|x_{t-1}, ..., x_1)$ to predict the next symbol in a sequence. It works as a time series model, where the next value is modeled as a linear combination of its own past values. The hidden layer contains a hidden state $\mathbf{h}$ and the output layers $\mathbf{y}$ operate on a sequence pf variable $\mathbf{x} = (x_1, ..., x_T)$. Like in time series model, at each time step t, the hidden state $\mathbf{h}_{(t)}$ of RNN is updated as follows:

$$\mathbf{h_{(t)}} = f(\mathbf{h}_{t-1}, x_t) \tag{11}$$

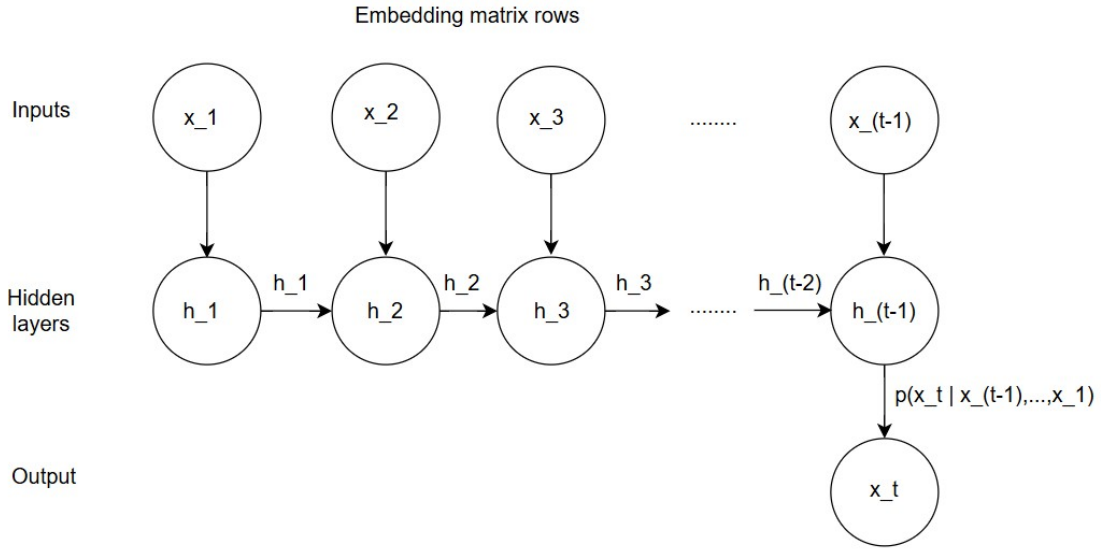


Figure 5: Recurrent Neural Network (RNN) Encoder-Decoder architecture for sequential data, in the Natural language processing (NLP) task, where the access is to predict the next text symbol x_t using previous text symbols $x_1, ..., x_{t-1}$.

where f is a non-linear activation function, $\mathbf{h_{(t-1)}}$ previous hidden state and x_t the input at time t. [5][22][23]

There exist different activation functions between different layers that can be used in the RNN model. One of the simple activation functions is the logistig sigmoid function (9) as shown in the previous section. During training, the RNN learns a probability distribution over a sequence. To predict the output of the next symbol in the sequence in timestep t, the useful activation function is softmax (8) as in the CNN model. Using the current and previous hidden states $(\mathbf{h}_{(t)}, \mathbf{h}_{(t-1)}, ..., \mathbf{h}_{(1)})$ and learned weight matrices $(\mathbf{w}_{(K)}, \mathbf{w}_{(K-1)}, ..., \mathbf{w}_{(1)})$, the softmax can be written as:

$$p(x_t|x_{t-1}, ..., x_1) = \frac{exp(\mathbf{w}_j \mathbf{h}_{(t)})}{\sum_{j'=1}^{K} exp(\mathbf{w}_{j'} \mathbf{h}_{(t)})} \tag{12}$$

for all possible text symbols $j = 1, ..., K$. By combining these probabilities, it can be be computed the probability of the sequence $\mathbf{x}$ as follows:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t|x_{t-1}, ..., x_1) \tag{13}$$

From this learned distribution it can be sample a new symbol at each time step from the most probable symbols. [22][23]

Depending on the task, there are different structural RNN models in which the number of outputs may vary. The more complex approach involves decoding a fixed-length vector representation back into a variable-length sequence. This is achieved by utilizing the conditional distribution over a variable-length sequence, conditioned

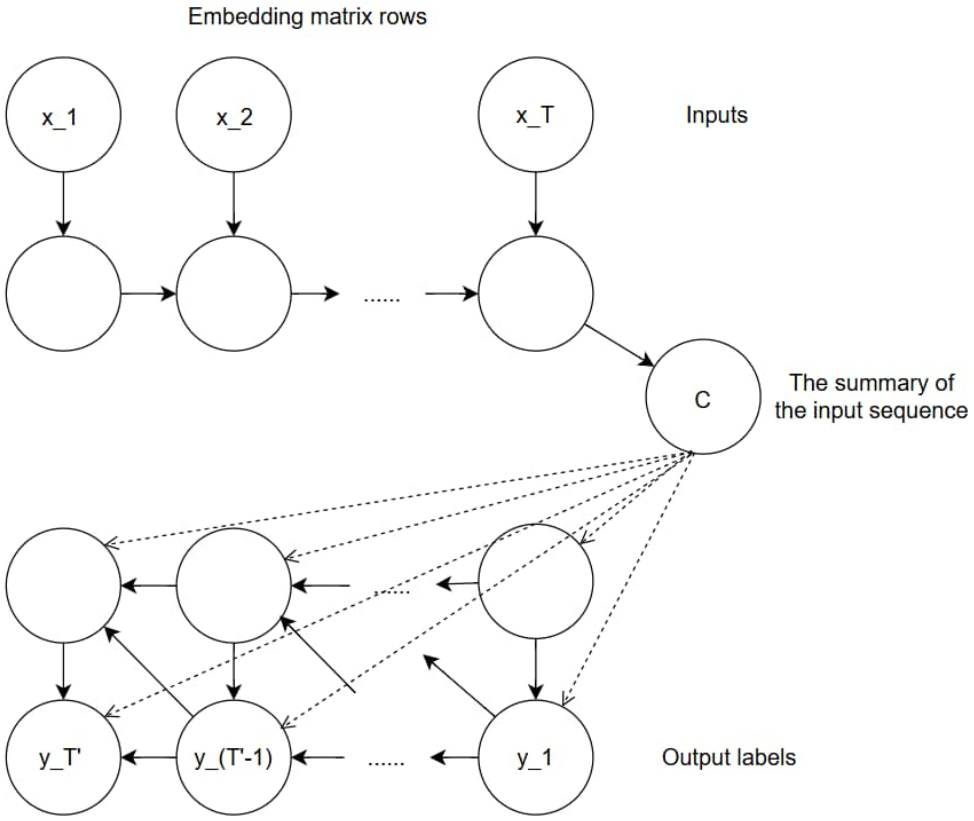


Figure 6: Another Recurrent Neural Network (RNN) Encoder-Decoder used to study sequential data. In Natural language processing (NLP) task, where the access is to decode a given fixed-length vector representation $(x_1, ..., x_{T'})$ back into variable length sequence $(y_1, ..., y_{T'})$.

on the input sequence length, using conditional log-likelihood as follows:

$$p(y_1, ..., y_{T'}|x_1, ..., x_T) = max_\theta \frac{1}{N} \sum_{n=1}^{N} log\ p_\theta(\mathbf{y}_n|\mathbf{x}_n) \tag{14}$$

where θ represents the set of model parameters and each $(\mathbf{x}_n, \mathbf{y}_n)$ consists of input sequence and corresponding output sequence from training set. Unlike in the model in Figure 5, in this model both y_t and the hidden state $\mathbf{h}_t$ are also conditioned on the previous label y_{t-1} and the summary of the input sequence $\mathbf{c}$. Based on these the hidden state of the decoder at time t is determined as follows:

$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, y_{t-1}, \mathbf{c}) \tag{15}$$

and the conditional distribution of the next symbol as follows:

$$P(y_t|y_{(t-1)}, ..., y_1) = g(\mathbf{h}_{(t-1)}, y_{t-1}, \mathbf{c}) \tag{16}$$

for given activation functions f and g. The architecture of this RNN Encoder-Decoder architecture can be seen in Figure 6. This RNN Encoder-Decoder architecture can be applied, for example, in text translation, speech recognition, text summarization, etc. [22]

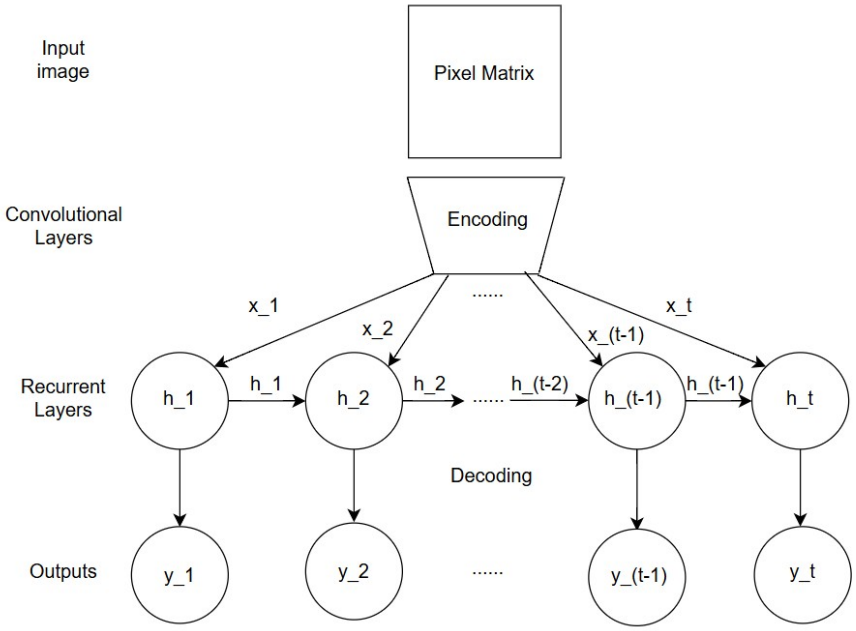


Figure 7: Standard Recurrent Neural Network (RNN) Encoder-Decoder architecture for image-based sequence recognition, where the access is to extract text symbols for example using encoded inputs $x_1, ..., x_t$.

The RNN Encoder-Decoder model works very similar in the image-based sequence recognition. The difference compared to the NLP tasks is that it is used Convolutional Layers as encoding part as in the Deep Convolutional Neural Network (DCNN) model, which learns features from images and converts them to the lower dimensional subspace. This preprocessing step converts an input object image into a sequence of image features. The Recurrent Layers work as decoding part as in NLP tasks. The architecture of this model is quite similar to the previously mentioned RNN Encoder-Decoder architecture, as seen in Figure 7. Here, a series of object labels is predicted instead of a single label. There are several image-based sequence recognition tasks such as scene text recognition, optical music recognition, etc. where this architecture can be applied. The specific image-based sequence recognition task used with coin images in this study is to extract text symbols from images. [4][5]

When dealing with sequential data, standard RNNs may face certain limitations and challenges. For instance, when processing long text sequences, they can suffer from short-term memory issues, meaning the model may not effectively retain all relevant inputs. This happens because the impact of an input on the hidden layer and consequently on the network's output either diminishes or amplifies exponentially as it propagates through the recurrent connections of the network [23]. This issue, known as the vanishing gradient problem [5][23], significantly restricts the amount of contextual information that standard RNNs can leverage, making it difficult to establish connections between relevant inputs and target outputs [23].

To overcome this limitation, architectures such as Long Short-Term Memory (LSTM) [23] networks are used. LSTMs introduce memory cells within the hidden layers, along with three key multiplicative gates: input, forget, and output gates.

These gates control the flow of information, enabling the network to retain essential data for extended periods. The memory cell preserves the context of the past, while the input and output gates facilitate long-term storage of information [5]. Meanwhile, the forget gate allows the model to reset stored data when necessary. By selectively storing and retrieving information, LSTMs effectively mitigate the vanishing gradient problem and enhance the model's ability to capture long-range dependencies, which are particularly important in image-based sequences [5][23].

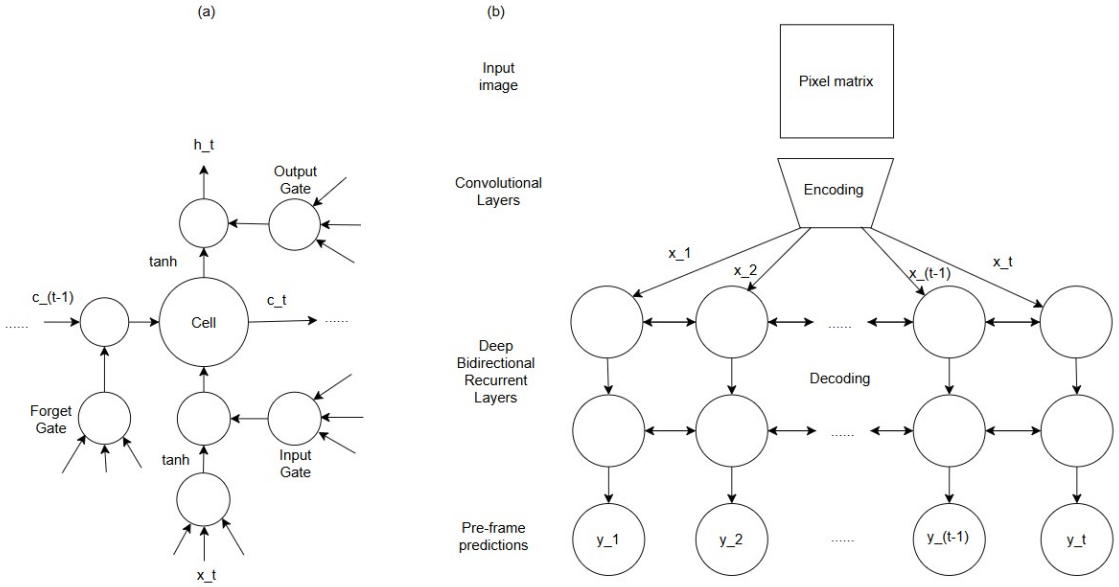


Figure 8: (a) The structure of a basic Long Short-Term Memory (LSTM) unit, which consists of a cell module and three gates, namely the input gate, the output gate and the forget gate. (b) The structure of deep bidirectional LSTM Encoder-Decoder used in image-based sequence. Combining a forward (left to right) and a backward (right to left) LSTMs results in a bidirectional LSTM. Stacking multiple bidirectional LSTM results in a deep bidirectional LSTM.

In many applications, it is required both past and future contextual understanding. For these cases *Bidirectional Recurrent Neural Networks* (BRNNs) [23] has been developed, which are used with LSTM structures. BRNNs process sequences in both forward and backward directions, providing the output layer with richer contextual information. A more advanced variant, Bidirectional LSTM (BLSTM), combines the benefits of both LSTMs and BRNNs, leading to improved performance in sequence learning tasks such as phoneme recognition and handwriting recognition. These enhancements make LSTMs and BLSTMs powerful alternatives to standard RNNs, particularly in tasks requiring long-range dependency modeling. For example, in image-based sequences, contexts from both directions are useful and complementary to each other [5]. Furthermore, it has developed multiple bidirectional LSTMs that can be stacked, resulting in a deep bidirectional LSTM architecture, as seen in Figure 8. [5][23]

To extract text symbols from images, the deep bidirectional LSTM Encoder-Decoder model alone is not good enough to predict clear text. One major limitation is that the model can predict text symbols at each time step but struggles to generate

properly structured words and sequences. To address this, a transcription layer is added after the Recurrent Layers to make the model more deep and effective. As mentioned above, the convolutional layers automatically extract feature sequence from each input image and the recurrent layer makes a prediction for each frame of the feature sequence, outputted by the convolutional layers. The task of the transcription layer is to convert pre-frame predictions made by RNN into a label sequence. Mathematically, transcription aims to find the label sequence with the highest probability conditioned on pre-frame predictions $(y_1, ..., y_t)$. This model is called CRNN and it consists of three components, including the convolutional layers, the recurrent layers and transcription, as seen in Figure 9. [4][5][6]

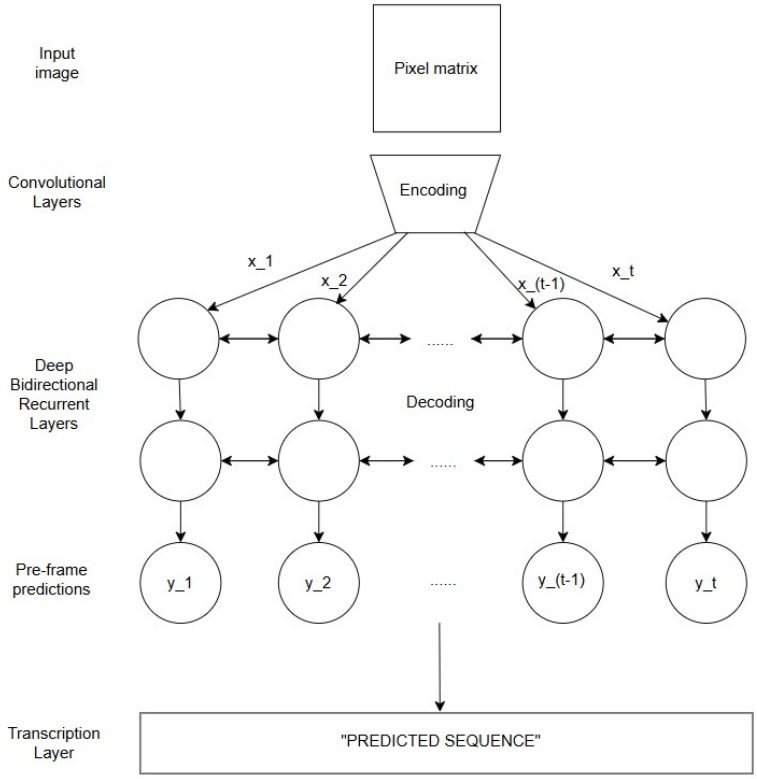


Figure 9: The architecture of Convolutional Recurrent Neural Network (CRNN) model for text extraction from images.

Conditional probability in the transcription layer uses a Connectionist Temporal Classification (CTC) algorithm. It can be analyzed to train deep neural networks in speech recognition, handwriting recognition, and other sequence problems. The CTC algorithm defines the probability of a label sequence $\mathbf{l}$ conditioned on the pre-frame predictions $\mathbf{y} = (y_1, ..., y_t)$ while ignoring the exact positions of the labels. This eliminates the need for explicit alignments between inputs and output labels, simplifying the training process. The key idea is to introduce a sequence-to-sequence mapping function, which removes repeated labels and blank tokens from intermediate predictions. For example, a predicted sequence like "–hh-e-l-ll-oo–" (where '-' represents a blank label) would be mapped to the final output "hello". Mathematically, the conditional probability $p(\mathbf{l}|\mathbf{y})$ is computed by summing over all possible

intermediate sequences π that can be mapped by β to $\mathbf{l}$ as follows:

$$p(\mathbf{l}|\mathbf{y}) = \sum_{\pi:\beta(\pi)=1} p(\pi|\mathbf{y}) \tag{17}$$

where the probability of π is defined as $p(\pi|\mathbf{y}) = \prod_{t=1}^{T} y_{\pi_t}^t$ ($y_{\pi_t}^t$ is the probability of having label π_t at time stamp t). [4][5][6]

The above equation (17) is computationally infeasible due to the exponential number of possible sequences, so the forward-backward algorithm is used for efficient optimization. In a lexicon-free transcription setup, the most probable sequence $\mathbf{l}$ is determined approximately by selecting the highest probability label at each time step and applying the sequence-to-sequence mapping function to obtain the final output as follows:

$$\mathbf{l}^* = arg\ max\ p(\mathbf{l}|\mathbf{y}) \tag{18}$$

This approach significantly improves text recognition accuracy by making the model robust to variations in input length and character positioning. [5][6]

To train the CRNN model effectively, the model is optimized by minimizing the negative log-likelihood of conditional probability of the ground truth label sequence, given an input image. For a training dataset $\mathbf{X} = \{I_i, \mathbf{l}_i\}_i$, where I_i is training image and $\mathbf{l}_i$ is the ground truth label sequence, the objective function is defined as:

$$O = -\sum_{(I_i,\mathbf{l}_i)\in X} log\ p(\mathbf{l}_i|\mathbf{y}_i) \tag{19}$$

where $\mathbf{y}_i$ represents the sequence of predictions generated by the convolutional and recurrent layers from the input image I_i. This objective function enables end-to-end mode training without requiring manual annotation of individual text components in training images. The model is trained using stochastic gradient decent (SGD), with gradients computed via backpropagation. In the transcription layer, error differentials are propagated using the forward-backward algorithm, while in the recurrent layers, Backpropagation Through Time (BPTT) is applied to adjust the model parameters effectively. To improve optimization, the ADADELTA algorithm is used for adaptive learning rate adjustment. Unlike traditional momentum-based methods, ADADELTA eliminates the need to manually set a learning rate and accelerates convergence, making it a robust choice for training deep sequence-to-sequence models such as CRNN. [5]

2.3.4 Keras OCR

Another and more enhanced DL technique to handle sequential data is Keras OCR method [7][8][12][13]. This method is effective with complex and difficult images where text symbols are not easy to detect with the standard CRNN model [4][5]. These images include, for example, curved, handwritten and printed texts, and contain noise and other patterns. As mentioned in Section 2.3.1, this method is a useful application in which the CNN model can be applied to investigate cultural and historical coin images. Like the CRNN model, the access of this is especially

to extract text symbols from images and generate properly structured words and sequences. [7][8][12][13]

Keras OCR method is pre-trained on a diverse dataset and used for automatic recognition of image text into an analyzable format. Printed text is generally easy to recognize because the characters have clearly defined sizes and shapes. However, recognizing handwritten text is more challenging if each person's handwriting is unique, and the and shape of characters are not consistent. Keras OCR is widely applied in text image investigation and the DL methods, such as CNNs have significantly improved its performance. It is useful to analyze historical materials, such as coin images and their texts. These methods achieve high accuracy, particularly in printed text recognition, but there are challenges, for example, with handwritten or unclear text. [7][12][13]

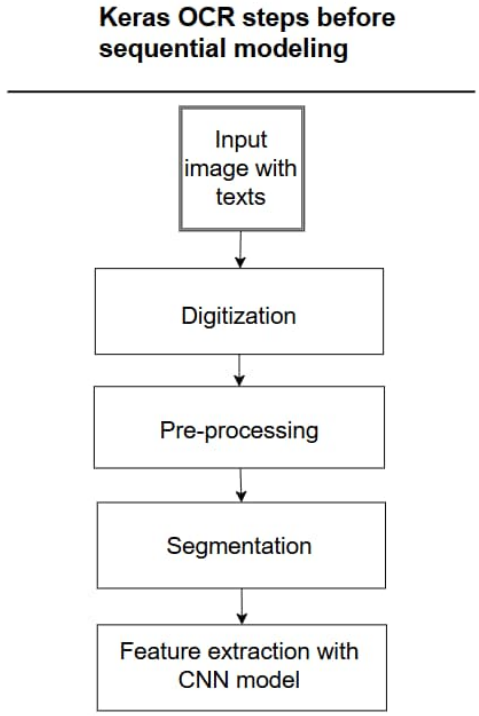


Figure 10: The steps of Keras Optical Character Recognition (OCR) to recognize text patterns.

The key steps of the Keras OCR approach before sequential modeling are digitization, preprocessing, segmentation, and feature extraction, as seen in Figure 10. *Digitization* begins the process where text is transformed into an electronic format by scanning documents or capturing images. This step ensures that the original content is preserved as a digital image. Following this, the *preprocessing* phase involves enhancing image quality through operations such as grayscale conversion, normalization, noise removal, and contrast adjustment. These techniques aim to improve the clarity of the image and support accurate text recognition in the subsequent steps. After preprocessing, *segmentation* takes place, where the image is divided into individual characters or words. The effectiveness of this stage significantly impacts the accuracy of text recognition tools like Keras OCR. Challenges such as overlapping

characters, diverse handwriting styles, and varying font sizes can complicate segmentation. Finally, *feature extraction* involves analyzing segmented characters by identifying attributes such as line direction, shape, and curvature. These features are statistically evaluated and used in CNN models for character classification and recognition, following the convolutional processing steps discussed in Section 2.3.2. [7][8][12][13]

After the above steps, the useful DL method for predicting text symbols and words based on the extracted features is CRNN with its statistical sequential modeling, probability methods, and loss functions, as mentioned in Section 2.3.3 [4][5]. CRNN is particularly effective in sequence-based text recognition, as it combines CNNs for feature extraction, RNN for sequence modeling, and the CTC loss function to map predictions to text sequences. CRNN is a powerful method for recognizing text with Keras OCR, especially when dealing with complex images with handwritten, circular, and different texts. [4][5][13]

To accurately localize text in images [13] and to improve the accuracy rate of the CRNN model bounding boxes play an important role in text detection [25]. Traditional pre-trained Keras OCR systems can apply word-level bounding boxes, which define a rectangular area around a detected word and and they can be applied as inputs in the CRNN model. However, these bounding boxes can struggle with complex cases, such as curved, arbitrarily shaped, or extremely long texts. To address this limitation, DL-based text detection methods utilize multiple bounding boxes, where bounding character-level annotations enhance detection flexibility. [7][25]

AN approach to creating the bounding boxes in the images is the Character Region Awareness for Text Detection (CRAFT) model, which applies the CNN methods to produce text region scores and affinity scores. The region score is used to locate individual characters in the image and the affinity score is used to group each character into a single instance [25]. It uses a Gaussian heatmap distribution to identify character centers and their connections. Since most datasets provide only word -level annotations, the model employs weakly supervised learning where the interim predictions generate character-level boxes. Their reliability is assessed using the following confidence score:

$$S_{conf}(w) = \frac{l(w) - min(l(w), |l(w) - l_c(w)|)}{l(w)} \tag{20}$$

where $l(w)$ is the ground truth word length, and $l_c(w)$ is the estimated number of characters. The pixel-wise confidence map S_c for an image is computed as follows:

$$S_c(p) = \begin{cases} S_{conf}(w), \ p \in R(w) \\ 1, \ otherwise \end{cases} \tag{21}$$

where p denotes the pixel in the region $R(w)$. The model is trained with a loss function that minimizes the squared difference between predicted and the ground truth values:

$$L = \sum_{p} S_c(p)(||S_r(p) - S_r^*(p)||_2^2 + ||S_a(p) - S_a^*(p)||_2^2) \tag{22}$$

where $S_r(p)$ and $S_a(p)$ are predicted region score and affinity score , and $S_r^*(p)$ and $S_a^*(p)$ are the ground truth region and affinity scores. Bounding boxes with low confidence ($S_{conf}(w) < 0.5$) are discarded to prevent training errors. This statistical approach improves text detection for long and distorted text. [25]

By combining CRNN with advanced bounding-box techniques, Keras OCR models have achieved greater flexibility and accuracy in text recognition tasks, especially with complex images. Whether applied to handwritten documents, historical images, or scene text in complex environments, these methods provide robust solutions for extracting text information in a structured and analyzable format. [4][5][13][25]

2.3.5 Combination with random rotation and Deep Learning

The coin images can be different and may contain various features even if they belong to the same category. To train effective CNN models, a huge amount of labelled data is needed that are used for training. That is why CNN models and other DL models are called data-hungry. [2][3]

Data augmentation includes different transformative techniques that can be analyzed to the images the original data include. These techniques include, for example, random rotation, random flipping, and random zoom, and with the help of these, it is possible to enhance existing features the images include. [11]

Random flipping and rotation are powerful data augmentation and pre-processing techniques, and they enhance the generalization ability of the CNN and other DL models. The random flipping involves mirroring the image horizontally, vertically, or both, and it can change the size of the image content. Random rotation rotates the image content by a randomly chosen angle θ to different positions. This allows the CNN model to observe the image from different orientations, and it adds the ability of the model to handle different images. Thanks to these methods, the CNN model can be trained more effectively because they add more variation to the images. This can make the CNN model more reliable in recognizing and detecting features in the images. [11]

Due to the position of the text in the coin images, mostly in a curved shape, a useful augmentation method in coin image investigation is random rotation. It directly addresses the challenge of achieving rotation invariance in CNN models. Coins are often presented at arbitrary angles, and ordinary CNN models struggle to recognize coins in different orientations. Using random rotation, the model can be trained to recognize features and classify coins regardless of their orientation. [8][15]

Keras OCR also benefits significantly from data augmentation techniques, such as random rotation in the case of coin images. As mentioned in the previous section, text localization, text processing, and text recognition are well-known challenges that OCR systems may face [8]. The OCR models can better recognize horizontal or straight texts. The upside-down text may remain unrecognizable or detected in the wrong order. In addition, trained OCR models assume that the text proceeds from left to right and top to bottom. Since coin images often contain text at various angles, random rotation helps the Keras OCR model recognize characters in different orientations. This augmentation enhances the accuracy of text recognition by exposing the model to a wider range of positional variations during training. [8]

The image rotation uses rotation transformation network (RTN) method [15]. Assuming that θ is the angle at which an image is to be rotated, a given image can be rotated clockwise at angle θ using the following rotation matrix form:

$$\begin{pmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{23}$$

If (x_0, y_0) is the center of the image before rotation and (x_1, y_1) is the center of the image after rotation, the relationship between these points can be written as follows:

$$\begin{pmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} = \begin{pmatrix} x_0 cos(\theta) + y_0 sin(\theta) \\ -x_0 sin(\theta) + y_0 cos(\theta) \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \tag{24}$$

The rotation of the image can be seen as a combination of a rotation matrix and a translation matrix, and it can be performed by translating (x_1, y_1) to (x_0, y_0). Let a and b be the translated values of x_1 and y_1 in their respective directions, leading to the following formula:

$$\begin{cases} a = x_0 - x_1 \\ b = y_0 - y_1 \end{cases} \tag{25}$$

By substituting equation (24) into (25), it is achieved the following formula:

$$\begin{cases} a = x_0 - x_0 cos(\theta) - y_0 sin(\theta) \\ b = y_0 + x_0 sin(\theta) - y_0 cos(\theta) \end{cases} \tag{26}$$

Finally it is obtained the following transformation matrix $T_\theta(I)$ that rotates the image I around its own center with θ:

$$T_\theta(I) = \begin{pmatrix} cos(\theta) & sin(\theta) & x_0 - x_0 cos(\theta) - y_0 sin(\theta) \\ -sin(\theta) & cos(\theta) & y_0 + x_0 sin(\theta) - y_0 cos(\theta) \\ 0 & 0 & 1 \end{pmatrix} \tag{27}$$

The resulted matrix ensures that the image remains centered and that random rotation θ will be implemented with precision. [15]

Data augmentation can somewhat address this issue by exposing the model to different rotations during training, but this method alone is not reliable enough for coin recognition. [15]

2.4 Machine learning application to Cultural Heritage

In recognition problems patterns such as texts and digits in coin images may involve complex structures that are best understood from a hierarchical perspective. The syntactic PR approach involves the use of descriptors that capture the structure of patterns by viewing them as compositions of simpler subpatterns. These descriptors enable the construction of complex observation structures from simpler patterns, helping to define the uniqueness of each pattern in the images. [11][16]

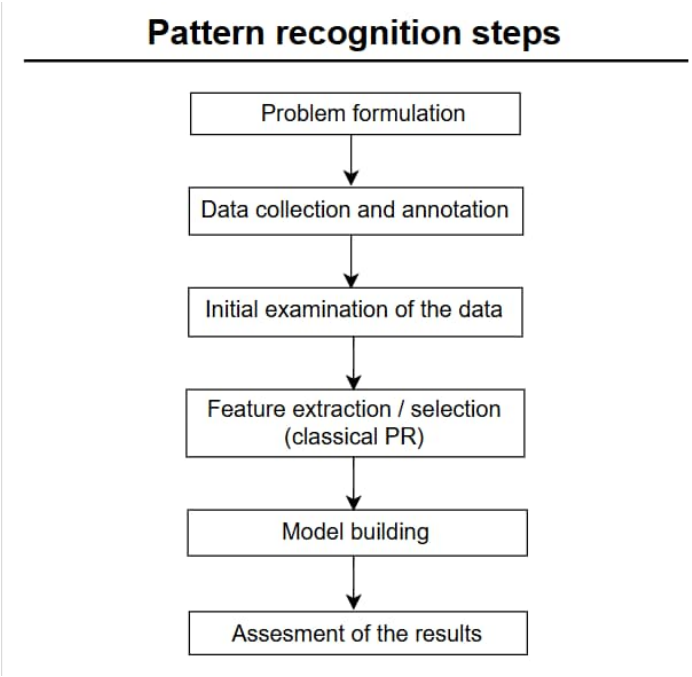


Figure 11: The summarize of Pattern Recognition steps.

To implement the PR task for the investigation of the coin image, there are specific steps that must be completed as seen in Figure 11. The first step is the *problem formulation* that aims to understand which type of investigation should be addressed to solve the problem. In the second step, must *collect the data* that contain specific coin images and annotate them if necessary. *Pre-processing steps*, such as outlier handling, normalization, and missing data management, ensure data quality. Relevant features are then extracted to optimize classification or clustering tasks with model building following to train algorithms for supervised or unsupervised learning. CPR and DL approaches are employed such as LBP [9][18], edge detection [10] and CNN to improve accuracy and generalization. Finally, *model performance* is evaluated using techniques such as cross-validation to ensure robust results across unseen data. PR applications thus rely on a systematic approach to transform raw data into actionable insights. [11]

3 Experiment

This study aims to build PR systems for coin classification analysis. In addition, this chapter focuses to develop systems that recognize and extract patterns from coin images. PR techniques such as LPB and edge detection are developed in order to extract features from coin images. In addition, modern approaches (DL) are introduced as well with the ability to take textual patterns, and in the end, these textual patterns are sent for NLP model to compute cultural analysis.

3.1 Set up

All experiments are implemented with Python in Jupyter Notebook and the results were recorded and visualized using Matplotlib and Seaborn libraries. For image pre-processing and feature extraction, LBP and edge detection techniques were employed. DL models, including CNN and CRNN were implemented using TensorFlow and Keras. OCR was performed using the Keras OCR library, while language generation and reasoning tasks were assisted by OpenAI's GPT-3.5 model via the OpenAI API.

3.2 Data preparation

The data of this study includes historical and cultural U.S. coin images from 20th century and it is available on Kaggle (Saharovskiy, 2023) [27]. These images contain information about 3,749 coin images belonging to three different categories. The dataset includes three classes of coins, as summarized in Table 1.

Coin Type	**Year Range**	**Nuber of Images**
Jefferson Nickels	from 1938 to present	1,214
Lincoln Cents	from 1909 to present	1,285
Washington Quarters	from 1932 to 1998	1,250

Table 1: Number of Coins per Class in the Dataset.

Figure 12: Example of coin images from each three classes.

This ensures a balanced representation of the coin types for training ML models. Before performing the modeling step, coin images are rearranged (normalized) in order to keep the data in the same interval (0, 1) (see Chapter 2 Section 2.1).

3.3 Implementing classical PR

Classical PR method is first used to extract features of the coin images. In order to recognize features and all text symbols as well as possible it is important to check the clarity of the images. Many of the dataset items contain artifacts such as black, unclear figures and text that may hinder the process of the feature extraction. To address these challenges mathematical and statistical preprocessing techniques can be applied to enhance the clarity of the images and improve text extraction accuracy. These techniques are for example histogram equalization, LBP methods, filtering and edge detections.

One of the alternative PR method to improve the quality of the images in this study is first to convert the images to grayscale where the color space is transformed from BGR to grayscale. In addition, all images are enhanced using histogram equalization which works by redistributing the intensity values of the image so that they are spread more evenly across the range of possible values. The result is an improvement in the image's clarity and contrast, as the darker regions become brighter and details that were previously hidden become more visible.

Figure 13 shows an example of coin image in its original gray image form and enhanced grayscale image form, and Figure 14 their histograms. The clarity of the image has slightly improved compared to the original gray image and its pixel matrix values are distributed more evenly. The patterns and text symbols can stand out better when there is more lighting in the image. There are still some dark points in the edges of the enhanced image also and some unstable points in its histogram. That is why this doesn't look like to be the best method to enhance the quality of the images. CPR methods must also be tested.

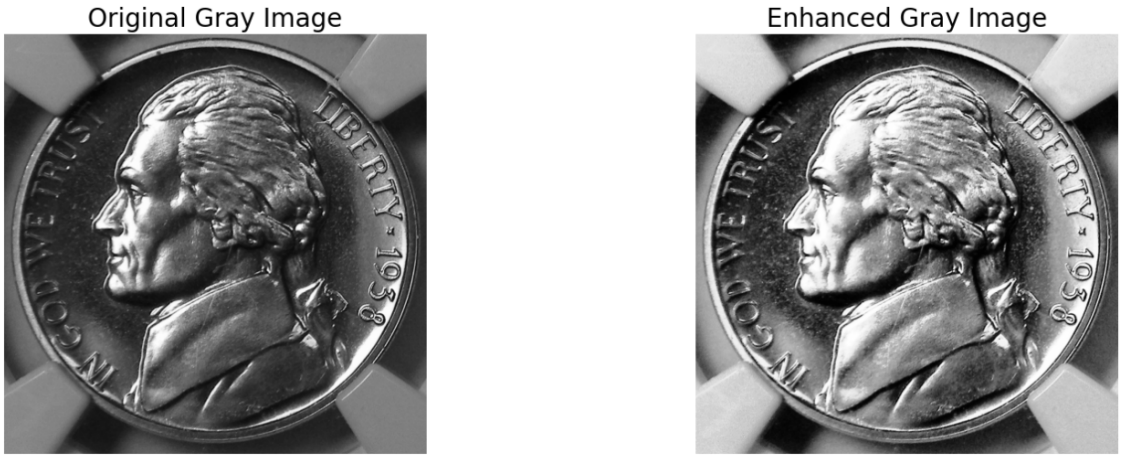


Figure 13: An example grayscale image before and after histogram equalization.

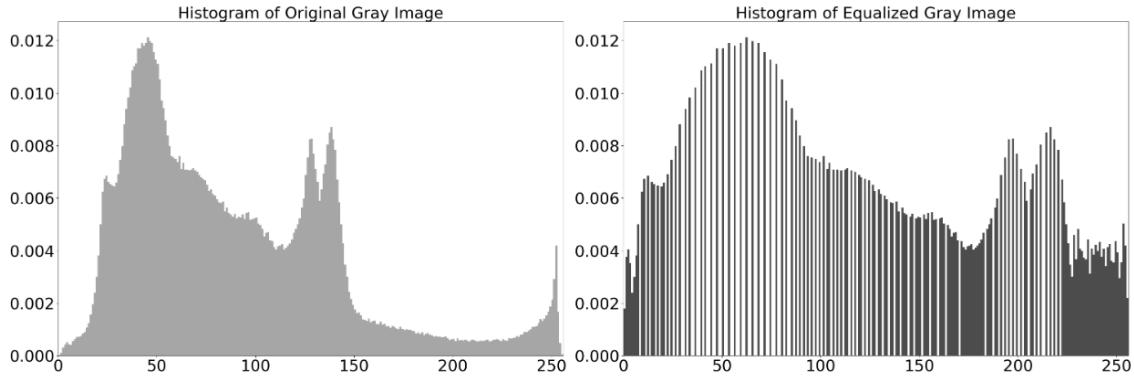


Figure 14: The pixel value distributions of the original gray scale image and the enhanced gray scale image.

3.3.1 Applying Local Binary Pattern

LBP also offers an improved quality of the image by computing a local representation of texture. The LBP algorithm considers pixel and its surroundings. These surroundings are typically defined as neighborhood $H \times W$ with the pixel in exams in the center. If the value of the pixel its greater or equal to the pixel in the center, a value of 1 is assigned. In the opposite case, a value of 0 is assigned. When the process is completed, the image becomes a sequence of binary numbers. Each $H \times W$ block of binary values is then converted back to its corresponding decimal number.

LBP works properly when correct neighborhood $H \times W$ pixels is selected. After various tests, the proper radius for this case study is decided. In this study, one of the best chosen values for parameters are $radius = 1$ and $n\_points = radius * 8 = 8$. Testing these parameter values to the enhanced grayscale image created in the previous phase, it can be seen if the algorithm can analyze and extract texture features from the coin image better and recognize the distribution of the image patterns in histogram as shown in Figures 15 and 16.

Figure 15: The enhanced image and its and Local Binary Pattern (LBP) form.

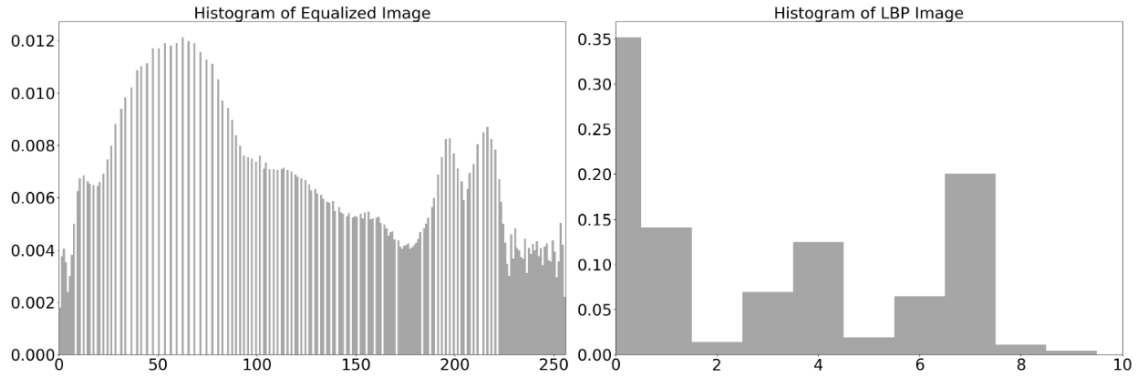


Figure 16: The histograms of enhanced and LBP image.

Based on the histogram distribution of the LBP image in Figure 16, the LBP algorithm with chosen parameter values has effectively captured texture patterns, particulary those corresponding to uniform and high frequency structures and also text symbols can be recognized well. The concentration of specific intensity values in the histogram suggests that the algorithm differentiates between smooth and edge-rich regions, indicating successful feature extraction for texture classification. The reason for this difficulty might be their small scale. Although the LBP can capture patterns in the image well there are still some drawbacks. Text symbols and other patterns don't stand out as well as in the enhanced grayscale image.

To improve image clarity, the image is cropped and masked to limit the analysis to the central region. This process enhances both grayscale and LBP images by focusing only the relevant area where important patterns or texts appears. Moreover, the Contrast Limited Adaptive Histogram Equalization (CLAHE) [26], which improves standard histogram equalization by limiting the contrast amplification in small regions to prevent overenhancement and noise artifacts. This method focuses on masking the image to emphasize only the relevant central region while eliminating irrelevant outer areas. This method ensures that only the central portion of the image, where relevant patterns or text are most likely located, is considered in further analysis. By eliminating extraneous details near the edges, the clarity of the region of interest is improved and computational resources can be focused more efficiently. This method keeps the pixel matrix values in the region of interest intact while setting all other pixel matrix values to zero.

The goal is to preprocess the input image by applying a circular mask and before extracting texture features using LBP algorithm. First, the center coordinates of the enhanced grayscale image are determinated. A circular mask with a fixed radius (400 pixels in the example image) is then created and drawn on a black background of the same dimensions as the input image. The circle is filled with white to define the region of interest while excluding areas outside its boundary. This mask is then applied to the enhanced grayscale image, resulting a masked output where only the circular region of interest remains visible, while surrounding areas are blacked out.

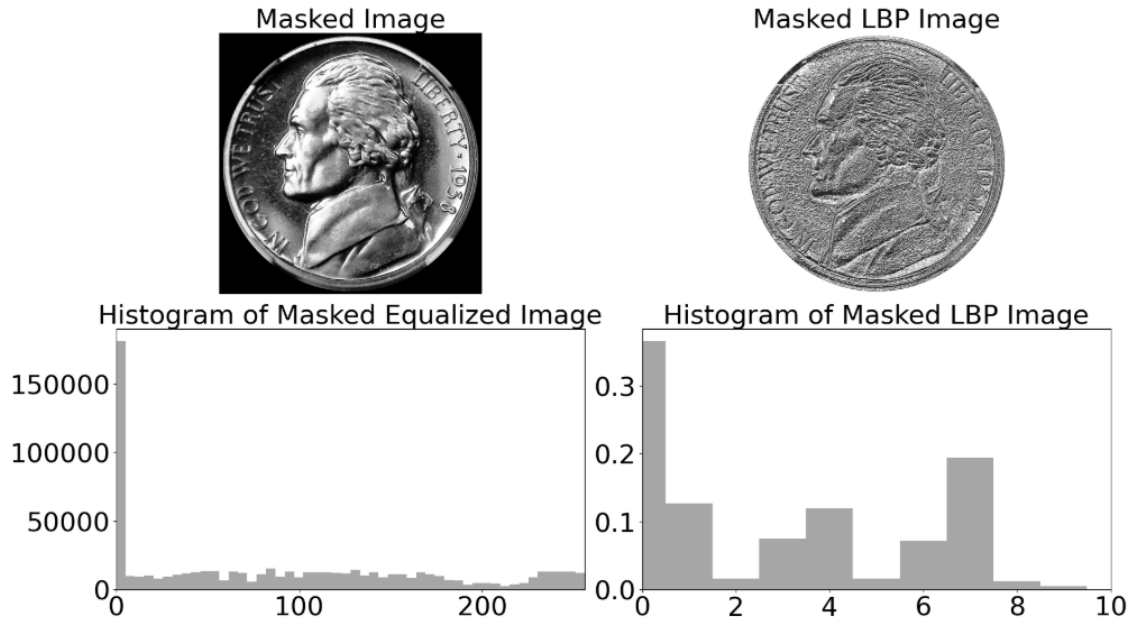


Figure 17: Example of masked coin images in enhanced grayscale image and Local Binary Pattern (LBP) image forms and their pixel distributions.

The patterns within the images are now more clear, and their distributions appear more stable, as distracting background elements have been largely removed. In the image in the Figure 17 text symbols and other patterns looks like to stand out more clearly in the equalized grayscale image than in the LBP image. Another advantages is the masking technique that can reduce noise and irrelevant details near the edges of the image. This helps to focus the analysis on the most critical areas, where image texts and patterns locates. Furthermore, LBP may reduce computational load, as fewer pixels are analyzed compared to the full image.

However, a significant drawback of the masked image is that this approach may exclude potentially relevant details that are located near or outside the defined region of interest. For instance, if patterns or text symbols extend beyond the circular mask, they will not be visible in the processed image, potentially leading to incomplete analysis. Additionally, the use of a fixed radius for the mask may not be optimal for all images, as the region of interest can vary in size or position depending on the dataset.

The LBP algorithm performs well in enhancing patterns with consistent texture but may struggle with regions that have uneven lighting or variations. The text symbols in particular may not always achieve the desired clarity because the binary thresholding used in the LBP can obscure finer details. To address these challenges, alternative method such as histogram equalization, edge detection techniques or DL approaches could be explored to complement or replace LBP in specific scenarios.

3.3.2 Applying Edge detection (Canny)

Edge detection is useful to find object inside an image by studying their boundaries. The boundaries are discovered by sliding the density feature region pixels. In edge detection, the density feature of region pixels refers to the proportion or concentra-

tion of edge pixels within a specific region of an image. This measure helps quantify the presence of edges in a given area and is useful for analyzing texture, detecting objects, and improving segmentation techniques. It can be calculated using the following formula:

$$D = \frac{N_e}{N_t} \tag{28}$$

where D represent the edge density, N_e is the number of edge pixels in the region and N_t is the total number of pixels in the region. In addition, edge detection can be used as increasing their feature analysis. In this experiment it is used Garbor filter method which is used in texture analysis, feature extraction, etc. The input is converted to grayscale and equalized to improve the lighting.

When the images are processed using the Garbor filters the patterns can be easily highlighted. The Garbor filter is a linear filter used to detect frequency components and orientations within an image. It is based on a Gaussian function modulated by a sinusoidal plane wave, which enables it to capture both local spatial frequency and phase information. This makes the Garbor filter particularly effective in texture analysis and edge detection, as it can highlight patterns at different scales and orientations.

The Gabor filter works by convolving the image with a set of kernels. Each kernel detects features that align with its specific properties, such as horizontal or vertical edges, as well as more complex textures. The convolution process essentially enhances the features of interest while suppressing others that do not match the filter's characteristics. This allows the Gabor filter to highlight key patterns, such as text symbols and structural details, which are crucial for further image analysis and classification.

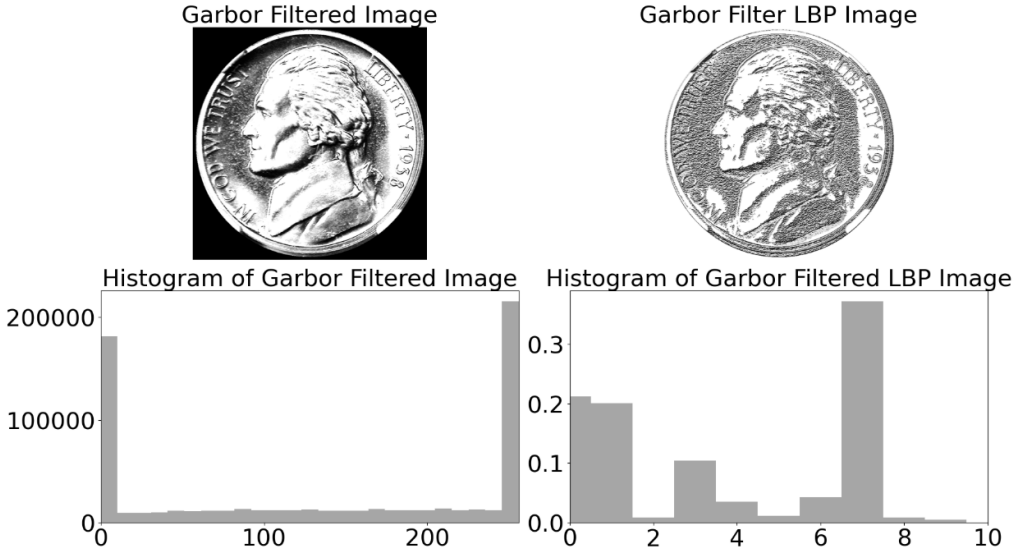


Figure 18: Example of masked coin images in enhanced grayscale Gabor Filter image and its Local Binary Pattern LBP processed image with their pixel value distributions.

In this experiment the Gabor filter is applied to extract texture features and

enhance the structural details of the image. To get as good results as possible, it is used masked enhanced grayscale image where the region of interest is limited (Figure 17). The parameters that are used are kernel size, Gaussian standard deviation σ. orientation θ, wavelength λ, aspect ratio γ and phase offset ψ as mentioned in Section 2.1.2. These parameters control the sensitivity of the filter to specify text symbols and patterns in the image. The resulting filtered image is further processed using the LBP algorithm to extract additional texture-based features. After this it is created Gabor filtered image and its LBP processed image with their pixel value distributions as in Figure 18.

In the Figure 18, it can be seen that the masked enhanced grayscale image is a bit more clear when it is added the Gabor filter algorithm in the previous results. In addition, text symbols and patterns stands out a little better when they are highlighted and the used parameter values are set sensibly in the algorithm. Based on its histogram, it can pay attention to the areas, where text symbols locate.

3.4 Implementing Deep learning

DL offers more straightforward methods to extract image features, and it can perform other operations such as image classification, object location, and semantic segmentation. To enable the CNN model to recognize and classify coin features accurately there are steps that must be done before the training of the model. The images must be preprocessed to ensure uniformity in their sizes and qualities. This step ensures that the model focuses solely on the relevant features such as digits, text, and symbols without being affected by variations in image dimensions. In addition, data augmentation is applied in order to train the model with more data.

3.4.1 Applying random rotation in CNN

Figure 19: Example coin image and its rotated form.

The used and recommend augmentation method for coin images in this study is random rotation. Random rotation is a data augmentation technique that creates new images by rotating the original one. The rotation can be applied in a tick-wise manner or randomly. In coin analysis, especially when the task assigned to CNN is to extract text from images, tick-wise operation may be used to extract letters from it. Using random rotation the CNN model can observe the coin images from different orientations and it adds the ability of the model to handle different images. Figure 19 shows one of the example coin image and its example rotated form.

3.4.2 Applying data splitting in CNN

In order to well train a model, the data need to be in a section used to train the model and another section used to test the model, which should not be seen in the training phases. Normally the data split in training, validation and test is achieved by random sampling the data. Because the size of the training data must be as large as possible especially in the DL models a split of 80% for training and 20% for testing is the most popular and recommended choice. In addition, the testing data is further divided into 10% for validation and 10% for final testing.

All training, validation and test set contain representative samples from all classes. This balance prevents class bias, which could skew the performance of the model. The 80/20 split is recognized as a best choice in machine learning, because it provides enough data for training while keeping a portion aside to evaluate the model accurately. To ensure the model can be fine-tuned without affecting the reliability of the final performance assessment the data need to be divided into all training, validation and test set.

3.4.3 CNN modeling and testing

This study uses a CNN with three hidden layers (see chapter 2 section 2.2.1) composed by (Convolutional layer batch normalization and max pooling) followed by three fully connected layers. Last layer uses Softmax (see Chapter 2 section 2.2.1) as final activation. Softmax is used to perform a classification operation. In this study, three convolutional layers and three fully connected layers are used in the CNN model.

After the CNN model is created, the coin image data is combined with its rotated images, which helps to ensure the robustness and reliability of the CNN model. In addition, the training data is preprocessed using ImageDataGenerator class, which is usually used to feed the CNN with augmented data. During the training, early stopping is used to prevent overfitting, where training stops automatically if validation loss does not improve anymore. The model is trained for up to 20 epochs with a batch size of 64 and validation data is used to estimate the performance of the model during training. These techniques contribute to an efficient and stable training process, as it can be seen in the Figures 20a and 20b.

(a) Training and validation accuracies.

(b) Training and validation losses.

Figure 20: Model performance during training.

By investigating the above Figures it can be seen, how well the CNN model learns to recognize the features of the coin images and classify them to the correct classes. In the beginning of the training, the classification accuracy increase deeply but slows down after two epochs. The validation accuracy increase also in the beginning of the training but starts to stay stable after two epochs. The same can be seen in the loss plot where the decreasing of the loss value slows down after two epochs both in training and validation sets and doesn't improve significantly after this. This is why the CNN model stops the training at epoch 10 to avoid the overfitting (early stopping). Including the Table 2 the estimated accuracy of the training dataset is about 99.6% and its loss value is about 0.014. When the model is tested on the testing dataset the accuracy value is about 98.4% and loss value is about 0.9.

	Training Data	**Validation Data**
Loss	0.014	0.996
Accuracy	0.087	0.984

Table 2: Train data and test data accuracies and losses.

To better understand how many images are correctly classified in each class, a confusion matrix is used. The confusion matrix makes it possible to identify which of the three classes has the highest number of correctly predicted images. In Figure 21, it is seen what kind of confusion matrix results the used CNN model has given in the coin image classification. The horizontal lines represents the predicted classes and vertical lines true classes.

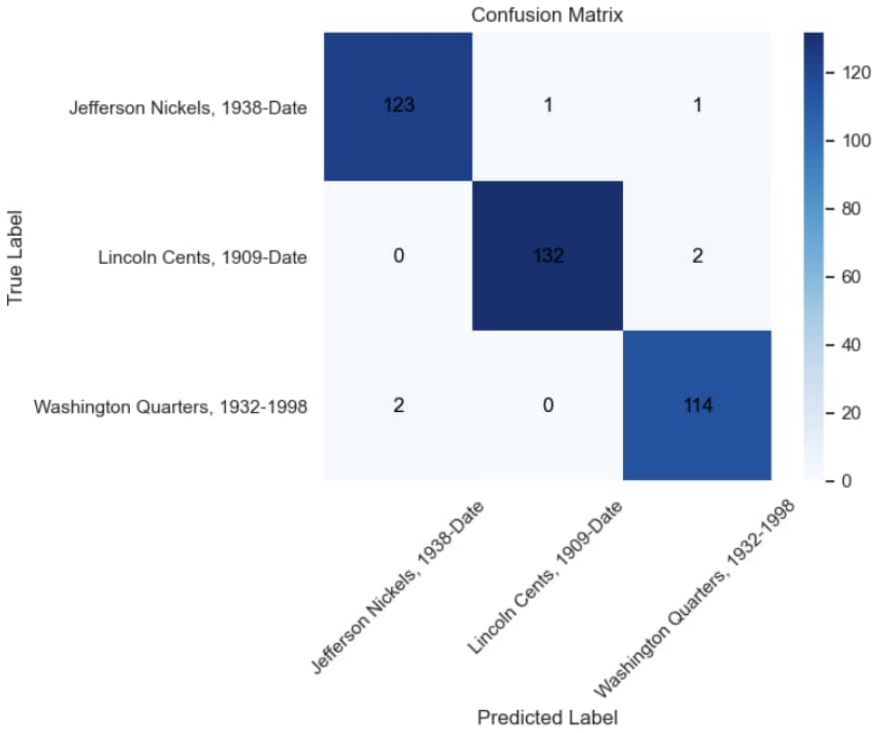


Figure 21: Test data confusion matrix results.

Including the results of the confusion matrix, 123 of the test coin images have correctly classified to the Jefferson Nickels class, 132 have correctly classified to the Lincoln Cents class, and 114 have correctly classified to the Washington Quarters class. Based on this, it has obtained the following class accuracies:

- Jefferson Nicels accuracy: $\frac{123}{123+0+2} = 0.984$
- Lincoln Cents accuracy: $\frac{132}{132+1+0} = 0.992$
- Washington Quarters accuracy: $\frac{114}{114+2+1} = 0.974$

Based on these results, the CNN model with chosen hyperparameters is good to classify U.S coin images in all three classes.

3.4.4 Applying Convolutional Recurrent Neural Network

Using a CNN model alone is insufficient for reading or recognizing text patterns from images. Instead, more comprehensive DL models are required, with CNNs playing only a minor role in feature extraction. This task is particularly challenging due to the diversity and variability of text patterns, as well as the presence of unpredictable backgrounds in images. As mentioned earlier, some images may contain dark or light backgrounds, while others contain unclear or complex visual elements. In certain cases, the font and background colors are nearly identical, with only a thin outline distinguishing the text. In addition, variations in font style, orientation, and text length further complicate recognition. Given these challenges, the key question is whether it is possible to develop DL models capable of accurately recognizing

text in a wide range of images, regardless of font type, orientation, or background conditions.

One of the useful and comprehensive DL models to test this challenge is to use CRNN. CRNN is the combination of Deep Convolutional Neural Networks (DCNN) and RNN which constructs an end-to-end system for sequence recognition utilizing statistical indexing and computations. Three key components of this model are convolutional layers, recurrent layers, and a transcription layer. Each of these components has their own tasks, as mentioned in Section 2.3.3, and they add more versatile structure to the model.

The convolutional layers extract features of the images and they work as encoder parts of the model. These features encode spatial information from the image, allowing the network to focus on meaningful patterns such as the shapes and textures of the text regions. This component can also extract a sequential feature representation ($x = x_1, ..., x_T$) and all these features are summarized to the lower dimension. In this CRNN model the component of convolutional layers is constructed by taking the convolutional and max-pooling layers.

The recurrent layers are designed to make sequence-labeling predictions for each frame of the extracted feature sequence. These layers capture temporal dependencies and sequence information, which is crucial for processing text in images, especially when the text is sequential or has varying lengths and orientations. The recurrent layers typically uses architectures like LSTM networks to handle these dependencies effectively and to avoid a memory problem. In addition, it is used a deep Bidirectional RNN structure because it it is observed that in image-based sequence, contexts from both directions are useful and complementary to each other. The recurrent layers predict a label distribution y_t for each frame x_t in the feature sequence $x = x_1, ..., x_T$.

The transcription layer, usually located at the end of the model, translates the predictions from the recurrent layers into a final label sequence. This transcription process is implemented using the Connectionist Temporal Classification (CTC) algorithm, which is designed to handle the challenges of sequence-to-sequence mapping tasks, such as text recognition. After the features of the image $X = [x_1, ..., x_T]$ are extracted and their correspondig output sequences $Y = [y_1, ..., y_U]$ (text labels) predicted, one of the primary challenges is that both X and Y can vary in length and there is no explicit alignment between the elements of the input and output sequences. The CTC algorithm addresses these issues by providing an output distribution over all possible Y -sequences for a given X. This allows the model to infer statistically the most likely output sequence without requiring a strict one-to-one correspondence between the input and output. For instance, the algorithm can correctly decode text from input sequences even when the spacing or alignment between characters in the image is irregular.

Because the coin image dataset does not contain labels indicating the text content in the images, an additional dataset is required for training the text recognition model. For this purpose, a separate dataset called the "MJ Synthetic Word Dataset" is used. This dataset contains thousands of word images along with their corresponding ground truth text labels. The dataset is widely used in OCR research and is publicly available on the Visual Geometry Group's website at the University of Ox-

ford (Visual Geometry Group, 2024) [28]. Figure 22 includes some example images of this dataset. A sample of over 200,000 word images is selected, each having a corresponding text label with a character length between 4 and 12 characters.

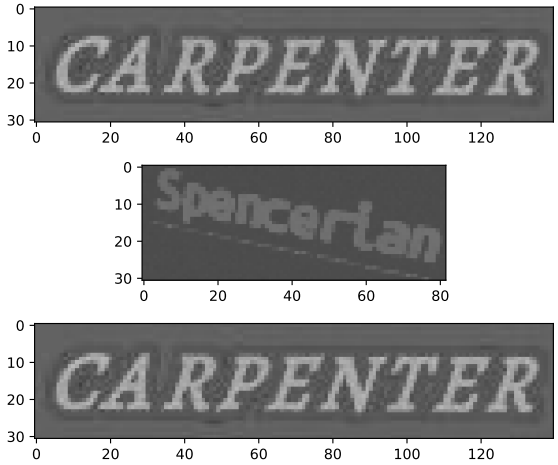


Figure 22: Example images of the training dataset of Convolutional Recurrent Neural Network (CRNN) model.

3.4.5 Image preprocessing and analyzing before Convolutional Recurrent Neural Network

Before training, it is recommended to analyze the features of the images and compare the training images with coin images. A comparison between Figures 12 and 22 reveals that the coin images have a square format, where as the training images are rectangular. Additionally, the text patterns in the coin images are not aligned in a straight line, unlike those in the training images. To address this discrepancy, the coin images are rotated so that the text appears in straight lines, as shown in Figure 23.

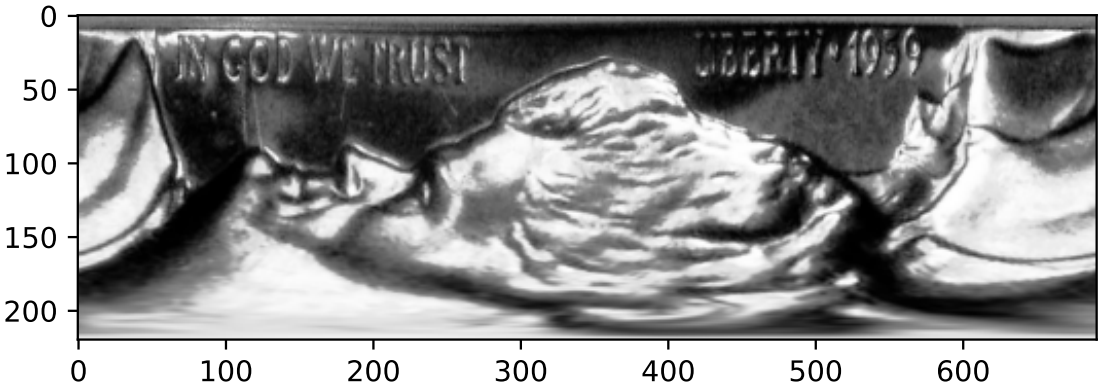


Figure 23: Preprocessed coin image example.

After this, the images can be resized accordingly without any loss of data, and

the model can be trained on the resized images. Using statistical methods, results are obtained as seen in Tables 3, 4 and 5.

Summary Statistics

Metric	Height	Weight
count	100000.00 px	100000.00 px
mean	115.64 px	31.04 px
sdt	39.73 px	0.36 px
min	1.00 px	9.00 px
25%	88.00 px	31.00 px
50%	109.00 px	31.00 px
75%	136.00 px	31.00 px
max	608.00 px	32.00 px

Table 3: The summary of height and width of the training images (in pixels).

Summary Statistics

Metric	Height	Weight
count	3748.00 px	3748.00 px
mean	254.31 px	798.04 px
sdt	118.15 px	371.20 px
min	35.00 px	110.00 px
25%	146.00 px	459.00 px
50%	290.00 px	911.00 px
75%	350.00 px	1100.00 px
max	668.00 px	2099.00 px

Table 4: The summary of height and width of the coin images (in pixels).

The name of the value	**Value**
Train Images Height 90 percentile	31.0
Train Images Height 99 percentile	32.0
Train Images Width 90 percentile	167.0
Train Images Width 99 percentile	240.0
Validation Images Height 90 percentile	31.0
Validation Images Height 99 percentile	32.0
Validation Images Width 90 percentile	167.0
Validation Images Width 99 percentile	239.0
Test Images Height 90 percentile	384.0
Test Images Height 99 percentile	508.649
Test Images Width 90 percentile	1206.0
Test Images Width 99 percentile	1597.949

Table 5: Summary of the 90 and 99 percentile values of the image heights and widths (in pixels) in the training, validation and test data.

Based on the analysis of the above results, Table 3 shows that the average height of the training images is 31 pixels, with a standard deviation of 0.36 pixels, indicating that the height remains relatively consistent across the dataset. In contrast, the average width of the training images is approximately 116 pixels, with a significantly higher standard deviation of 39.73 pixels, suggesting greater variability in image width. This variation in width may impact the model's ability to generalize,

as large differences in aspect ratios could affect feature extraction and recognition performance. Based on these, most of the training images have height between 30 and 32 and there is more variation in their width. In addition, in Table 5 it can be observed that there is no difference in the 90th and 99th percentile values for training and validation image heights. There is a considerable difference in the 90th and 99th percentile values for training and validation image widths, which can assume that most of the training and validation images have a height of 32 pixels and a width of 170 pixels. Very few images have have a width more than 200 pixels.

Table 4 shows that the mean height in the preprocessed coin images is around 254 pixels and the mean of the width is around 800 pixels. Additionally, the height and width values exhibit a significantly higher standard deviation compared to the training images, indicating greater variability in image dimensions. Furthermore, differences can be observed in the 90th and 99th percentile values, suggesting that the extreme cases in the dataset deviate more from the median dimensions than in the training set. This increased variation may impact model performance, particularly in handling images with uncommon aspect ratios. Based on these, it can be assumed that most of the preprocessed coin images have a height of 384 pixels and a width of 1200 pixels. Very few images have a width greater than 1200 pixels.

Because text labels are the output that our model predicts given an input image containing the text it is also useful to analyze the length of the labels. The length of labels plays an important role in this problem, and this makes an interesting case to explore the lengths of labels in the training data. To investigate these lengths, the probability density function (PDF) and cumulative density function (CDF) plots are used, as seen in Figures 24 and 25.

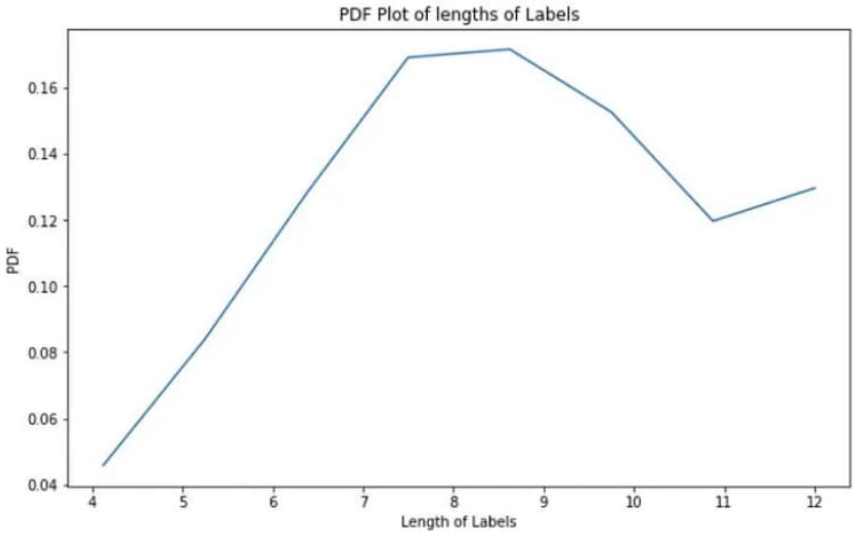


Figure 24: Probability density function (PDF) plot represents the distribution of length of labels in the training data.

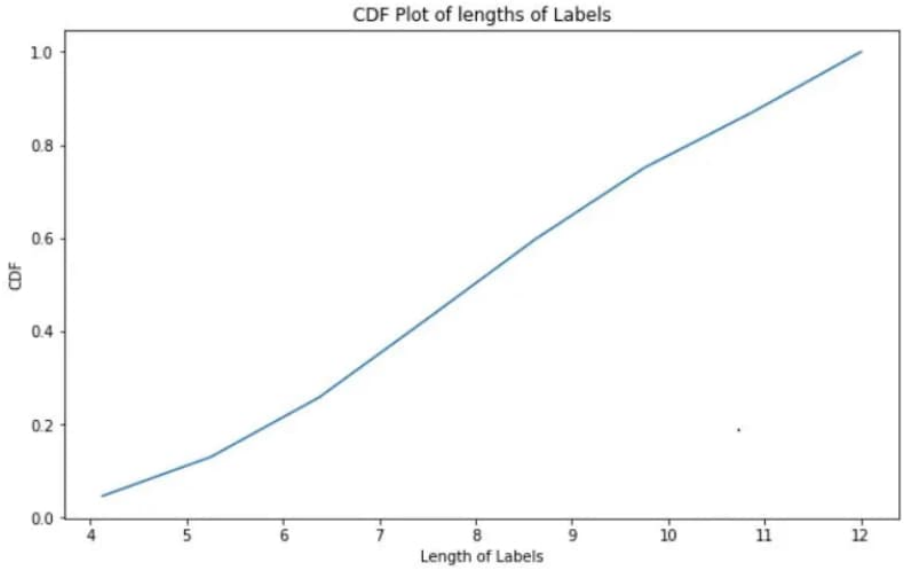


Figure 25: Cumultative density function (CDF) plot represents the probability that length of labels in the training data takes on a value less than or equal to a given point.

In Figure 24, the PDF plot of labels gives a good view of how the label lengths are distributed in the data. It can be seen that most of the labels have lengths 7, 8, 9 and 10 text characters. There are fewer labels with lengths 4 and 5 text characters. In Figure 25, from the CDF plot it can be seen that labels with lengths 4 and 5 are very less comprising comprising of nearly 10% and almost 80% of the data have a label length of 10 or less.

3.4.6 CRNN modeling and testing

The model used in this context employs a CNN as its backbone, followed by a Bidirectional LSTM for sequence modeling. The optimization process is carried out using the Adam optimizer to minimize the loss function. In addition, the letter and digits considered present in the text labels are the following strings: "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ".

The training stage takes input image, text labels (encoded as integers), input length (time-steps length), label length, and then the algorithm outputs CTC (Connectionist Temporal Classification) loss. The CTC loss is a special method that is used, for example, to recognize text and designed specifically for situations where sequence input (for example, image pixels) and the corresponding output (such as text) do not have precise alignment. The CTC loss is minimized by maximizing the probability of the correct sequence: $CTCLoss = -log(P(Y|X))$. The prediction stage takes the input image and outputs a matrix of dimensions 42×32 where 42 is the number of time steps of RNN and 32 is the length of letters + 1 character for the ctc blank.

The model is trained on 200000 images and validated on 12000 images for 20 epochs with early stopping. After 16 epochs the validation loss value doesn't improve so it stops at epoch 18. Using the best weights the model has created during training,

the plot is obtained, as seen in Figure 26 and Table 6.

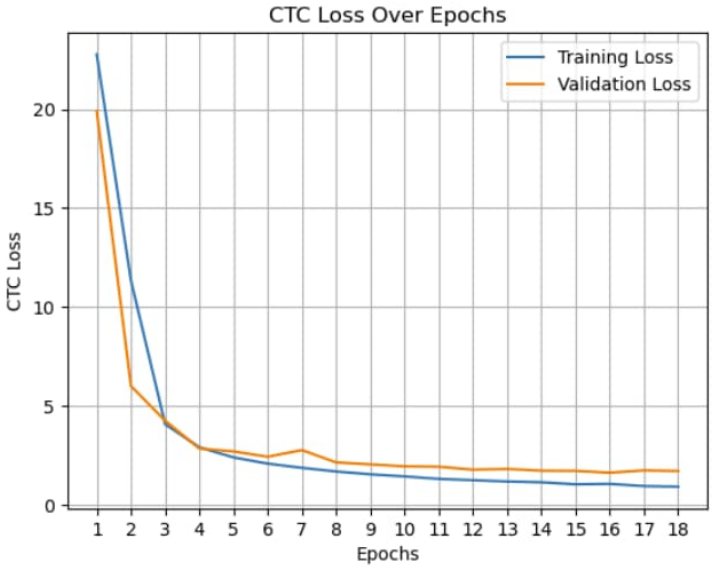


Figure 26: Train and validation loss plot.

Epoch 16/20	
Train Average Accuracy	89.32 %
Train Average Letter Accuracy	95.29 %
Train Loss	1.0615
Validation Average Accuracy	81.26 %
Validation Average Letter Accuracy	91.51 %
Validation Loss	1.6261
Epoch 17/20	
Train Average Accuracy	86.38 %
Train Average Letter Accuracy	94.49 %
Train Loss	0.9512
Validation Average Accuracy	78.23 %
Validation Average Letter Accuracy	90.59 %
Validation Loss	1.7446
Epoch 18/20	
Train Average Accuracy	88.80 %
Train Average Letter Accuracy	95.24 %
Train Loss	0.9202
Validation Average Accuracy	81.02 %
Validation Average Letter Accuracy	91.46 %
Validation Loss	1.7144

Table 6: Train and validation letter accuracies, average accuracies and CTC loss values in the last epochs.

Including Figure 26 the trained CRNN model performs reasonably well for images it is used for training and validation. It can be seen how much training CTC loss and validation CTC loss values decrease after each epoch during the training. The greatest improvement in the loss value occurs at the beginning of the training, and it slows down after each epoch. In the table 6, the best validation CTC loss value is reached at epoch 16 when the training CTC loss is around 1.1 and validation CTC loss around 1.6. The model trains for two more epochs after this and the validation CTC loss doesn't improve so it stops at epoch 18.

Testing the rotated coin images to the model it can be gotten the example text predictions as seen in Figures 27 and 28:

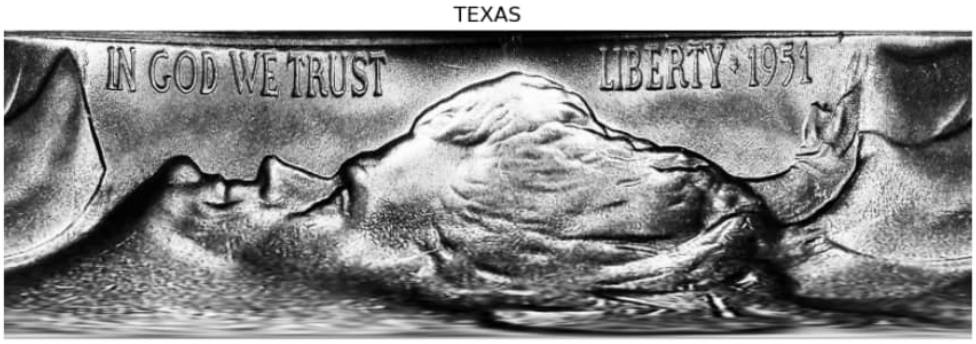


Figure 27: Example coin image text prediction.

Figure 28: Example coin image text prediction.

The used CRNN model predicts some text symbols, but they are not close the true texts of the coin images. Although the used CRNN model gives good CTC loss values in training and validation images, it is not effective for coin images because their features and sizes are so different. By comparing the images used for training and validating in Figure 22 and the coin images, it can be seen that the coin images include also other patterns that do not represent text symbols. The text symbols in the coin images are also difficult to separate, because they are the same color as their background and other patterns. In addition, in the training and validation images, text symbols are bigger and separates from the other environment of the image clearly. It is also possible that the used model is not effective enough for complex and difficult images or hyperparameter selections are not correct. Because it is difficult to find coin image data that includes annotated text labels, it can be

reasonable to try better DL models that do not need any annotation for training and can handle even complex images.

3.4.7 Applying Keras OCR with random rotation

A more specialized CNN in reading characters from text or images is Keras OCR that offer a better alternative than CRNN. This is primarily due to the limited versatility of the training dataset, which doesn't sufficiently cover variations in lighting, font styles, text orientations and other patterns. In addition, it focuses only to recognize texts, but doesn't find text areas effectively. To address these limitations, it is reasonable to explore a pre-trained DL method that incorporates both text detection and recognition. Using a model trained on a diverse dataset, we can improve the ability to locate and extract text more effectively before recognition, potentially leading to more robust and accurate results.

An effective method to fix this problem, which integrates both text detection and recognition, is Keras OCR. The key advantage of this approach is that it first localizes text regions before performing recognition, improving the accuracy in complex backgrounds such as coin images. In this study, the grayscale image is first converted to a three-channel color format to ensure compatibility with the Keras OCR algorithm. The algorithm itself is pre-trained on a diverse dataset, allowing it to generalize well to different text appearances. The method extracts text regions and outputs both the detected text and its corresponding bounding-box coordinates.

In order to enhance text recognition results, useful preprocessing methods are used. The preprocessing methods used in the classical pattern recognition phases are gray scale converting, the lighting enhancing, image limiting and edge detections. Depending on the clarity of the image, it is not necessary to use all of these methods to clarify if the image is already clear enough. For example, in the coin image dataset used in this study, some images have light background and some images have dark background.

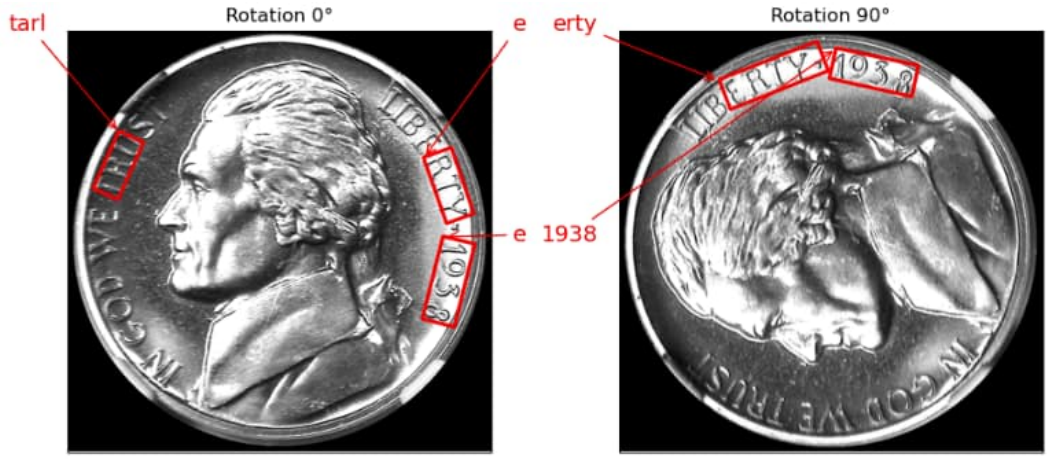


Figure 29: Keras OCR results in different rotational positions.

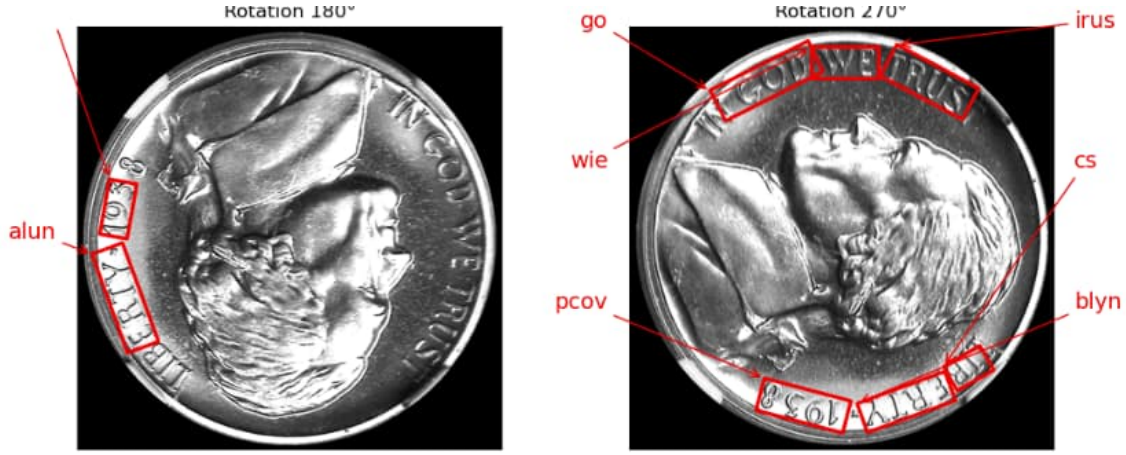


Figure 30: Keras OCR results in different rotational positions.

As it can be seen in Figures 29 and 30, Keras OCR can extract some text pieces well from the coin images. The model can find text symbols at the pixel level and group them into blocks (red rectangles in the above images), allowing it to recognize curves and informal text as well. The detected text blocks are used as inputs in the pre-trained CRNN model that can detect text symbols in the images. The model can even detect unclear fonts when the CRNN model combines convolutional and RNN neural networks using statistical functions and hyperparameters and utilizes the CTC method in texts of different lengths.

Although the used model is effective, it can't detect all text symbols correctly, as seen in Figures 29 and 30. The reason for this is the curved form of the texts when part of the text symbols are upside down. Curved text breaks text letters into non-standard places, which makes it difficult to combine them into the right word. Many of the OCR models have been trained mainly for horizontal or slightly tilted text. The upside-down text may remain unrecognizable or detected in the wrong order. In addition, trained OCR models assume that the text proceeds from left to right and top to bottom.

To solve these problems and improve the detection results, preprocessing techniques are needed to straighten the texts. As seen in Figures 29 and 30, the text detection results depend on the rotation position of the coin image. Based on this, one way is to rotate the image into different positions and check which text patterns the model can detect. However, there are text symbols that the model can't detect correctly. Another and better way is to straighten the images like in Figure 23, to get text symbols to the straight line. When the mentioned Keras OCR steps are completed for these preprocessed coin images, the results are obtained, as seen in Figure 31. To straighten the text in the image, the center of the image must first be cropped. Then, the image is rotated 90 degrees counterclockwise to better align the text. Afterward, the image was remapped into polar coordinates, which helps to preserve the text as a continuous block, despite the original curve. Finally, the image was rotated back 90 degrees to restore the proper orientation of the text before passing it through the OCR model.

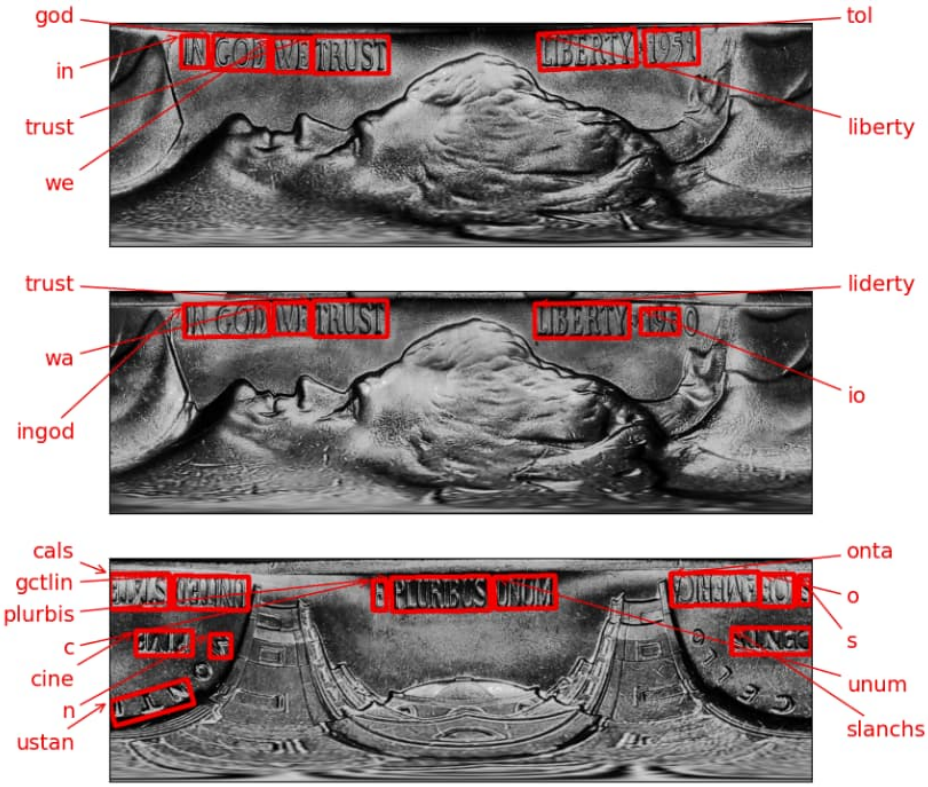


Figure 31: Keras OCR results with straighted images.

Comparing the results in Figure 31 with Figures 29 and 30, it can be seen that the OCR model can detect text symbols much better. In the first two example images, the model can recognize almost all words and text symbols correctly despite some small errors, such as year numbers. Therefore, it is possible that more preprocessing techniques are needed to improve the clarity of the images, or the used OCR model must be fine-tuned with some variety of annotated data to get better results. Although the texts are in the straight there are still a few symbols the model can't recognize. In addition, there are some images in the coin dataset, where it is not easy to rotate the images so that all the text is straight in the right direction.

3.5 Implementing Generative language models

As mentioned above, an effective approach to correct misidentified words in text extraction results is leveraging NLP models like GPT-3 and other generative language models. These models can help refine extracted text by predicting the most likely corrections based on context, thereby improving the overall accuracy of text recognition. Additionally, NLP models can be utilized to compare texts with each other using mathematical and statistical measurements and by comparing texts to reference data. By integrating DL-based text detection with advanced NLP techniques, the extracted text can be optimized for downstream tasks such as document analysis, classification, and information retrieval.

3.5.1 Applying GPT-3.5 and similarity measurement

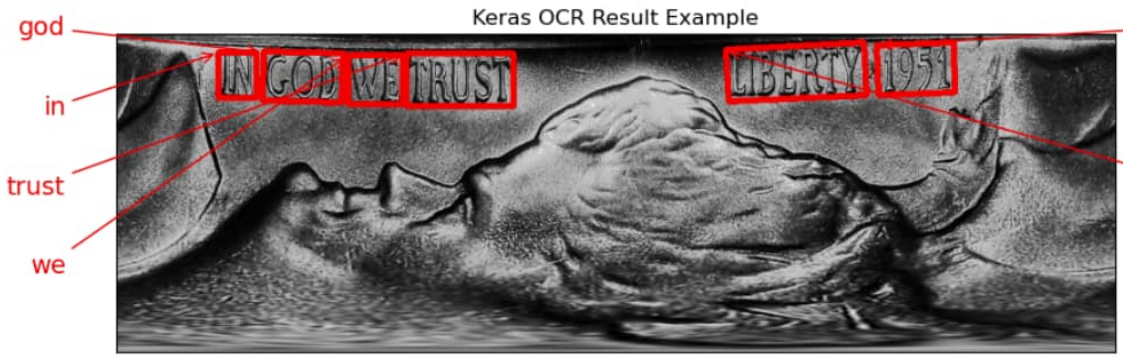
A combination of DL and NLP models is utilized to recognize and classify text symbols on U.S coins. First, images are pre-processed using PR techniques, such as grayscale conversation, histogram equalization, and circular masking. As mentioned earlier, these steps help to remove background noise and enhance text contrast. In addition, the images were transformed into polar coordinates to preserve the curvature of the coin's text.

After the Keras OCR model is employed for text recognition of the preprocessed images, the extracted text is then analyzed using NLP models. Especially, the GPT-3.5 model is used in comparing and categorizing the recognized text symbols into predefined coin classes (Jefferson Nickels, 1938-Date, Lincoln Cents, 1909-Date and Washington Quarters, 1932-1998). It is used to correct predicted text symbols, estimate the coin's category, dimensions, material, and symbolic representation, enabling a more precise and context-aware interpretation of OCR results.

Additionally, mathematical and statistical measurements such as cosine similarity are applied to measure how similar the predicted text with Keras OCR and the corrected text with GPT-3.5 are to each other. To do this, both texts must be converted to numerical vectors. In this study, the TF-IDF method is used for this. After the vectors are formed the cosine similarity can be calculated using the following equation:

$$\frac{A \cdot B}{||A|| \cdot ||B||}, \tag{29}$$

where A and B are vectorized texts. The value can be between 0 and 1. Value 1.0 means that texts are identical and value 0.0 means that texts are not similar at all.



{"id": "chatcmpl-AzhhvTqRvhPD5pXehE4CuFeOiZayN", "choices": [{"finish_reason": "stop", "index": 0, "logprobs": null, "mess
ust, liberty for all.", "role": "assistant", "function_call": null, "tool_calls": null, "refusal": null}}], "created": 173
o-0125", "object": "chat.completion", "service_tier": "default", "system_fingerprint": null, "usage": {"completion_tokens"
l_tokens": 36, "prompt_tokens_details": {"cached_tokens": 0, "audio_tokens": 0}, "completion_tokens_details": {"reasoning_
"accepted_prediction_tokens": 0, "rejected_prediction_tokens": 0}}}

Cosine Similarity: 0.6327904583679949

GPT-3 Similarity: In God we trust, liberty for all.

Figure 32: Correcting the extracted texts with GPT-3.5 model.

{"id": "chatcmpl-Azhtl9dgup3eMZvBFUIvXAq6fGhqY", "choices": [{"finish_reason": "stop", "index": 0, "logprobs": null, "mess
text \"In God we trust, liberty for all,\" the coin likely belongs to the category of 'Washington Quarters, 1932-1998'. Wa
ture the motto \"In God We Trust\" and symbols of liberty such as Lady Liberty and the American bald eagle. ", "role": "as
l, "tool_calls": null, "refusal": null}}], "created": 1739270117, "model": "gpt-3.5-turbo-0125", "object": "chat.completic
"system_fingerprint": null, "usage": {"completion_tokens": 62, "prompt_tokens": 98, "total_tokens": 160, "prompt_tokens_de
udio_tokens": 0}, "completion_tokens_details": {"reasoning_tokens": 0, "audio_tokens": 0, "accepted_prediction_tokens": 0,
0}}}

Cosine Similarity: 0.40458082617324087

GPT-3 Similarity: Based on the text "In God we trust, liberty for all," the coin likely belongs to the category of 'Washir
ington Quarters typically feature the motto "In God We Trust" and symbols of liberty such as Lady Liberty and the Americar

Figure 33: Categorization based on the corrected text.

The results are shown in Figures 32 and 33. By combining Keras OCR techniques with DL-based NLP analysis, it can be seen that this method can make the result more accurate and context sensitive in the recognition and classification of coin texts. This shows how generative language models can be utilized for interpreting and analyzing text on historical artifacts, such as coins. The used model can't detect the year numbers so well, because Keras OCR method couldn't extract them correctly. This is why the GPT-3.5 model categorizes the image to the "Washington Quarters" category instead of the "Jefferson Nichels" category. Because the cosine similarity: 0.63 is greater than 0.5, the predicted text that Keras OCR has extracted is quite close to the correct result.

4 Results

This section presents the key results of this thesis. This chapter is organized in the following subsections: covering CPR techniques, CNN-based feature extraction and classification, CRNN text recognition, Keras OCR, and NLP-based refinement.

4.1 Classical Pattern Recognition Results

LBP, edge detection (Canny), and histogram analysis were tested to improve text visibility and separate desired patterns from others. The results showed that the LBP and edge detection techniques were able to enhance the clarity of patterns and produce clear text regions, but failed in more complex and low-contrast coin images. Histogram analysis proved useful for highlighting the brightness distribution in the text regions, however, it was not sufficient on its own to effectively separate text from complex backgrounds. Using these techniques, the results were obtained in Figure 34 with the example image.

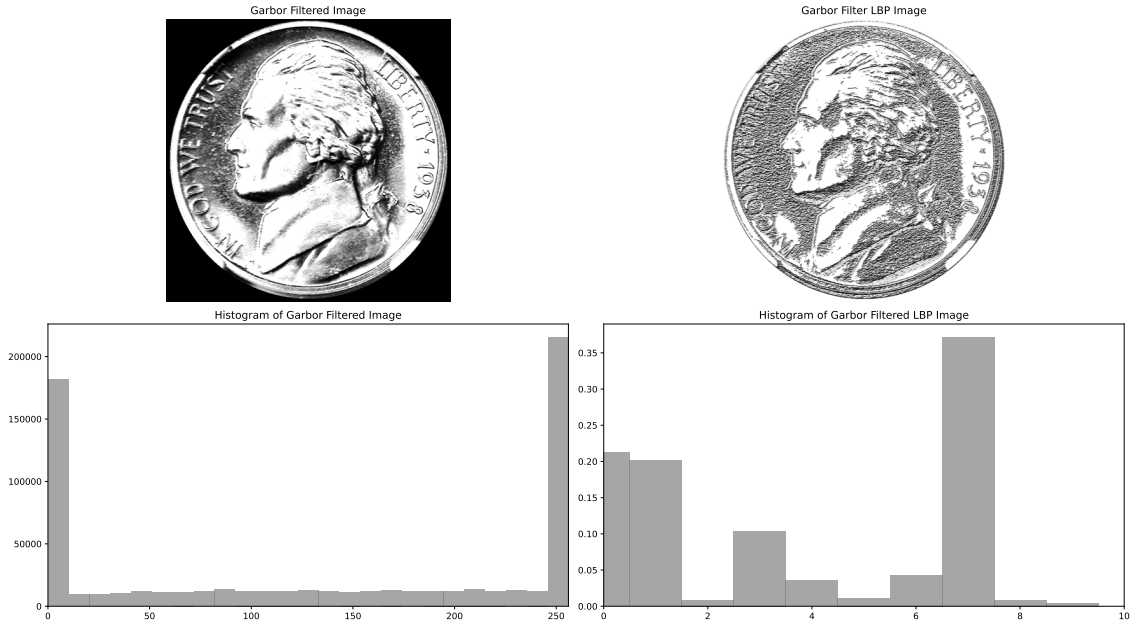


Figure 34: These result shows how texture-based CPR results like LBP and edge detection can improve the quality and of the coin image.

4.2 CNN-Based Feature Extraction and Classification Results

CNN models extracted features and classified image regions with improved clarity compared to classical methods. The loss and accuracy values, in Table 7, and the confusion matrix, in Figure 35 showed high precision in all classes. Rotated coin images 36 were also used to improve robustness.

	Training Data	Validation Data
Loss	0.014	0.996
Accuracy	0.087	0.984

Table 7: Train data and test data accuracies and losses in CNN model.

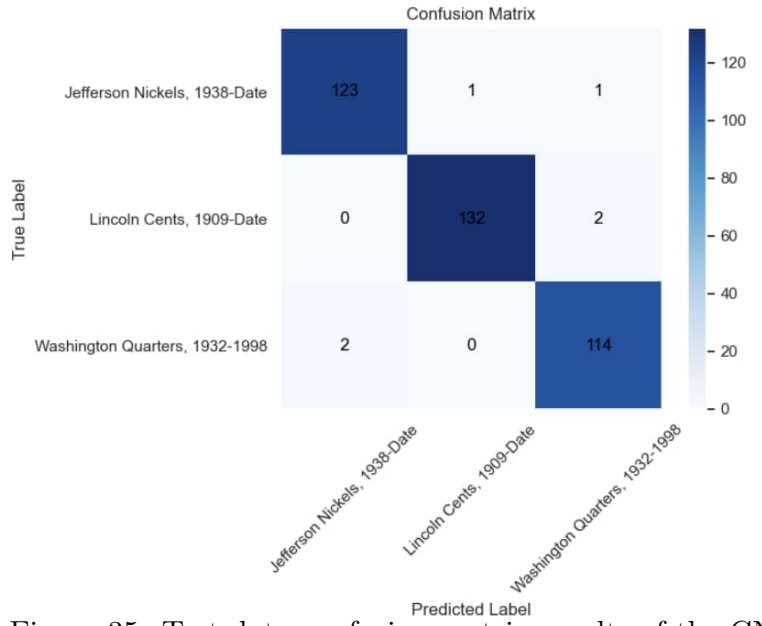


Figure 35: Test data confusion matrix results of the CNN model.

Figure 36: Example coin image and its rotated form.

4.3 CRNN Model Performance Results

The CRNN model, trained on 200,000 images and validated on 12,000 images, achieved its best validation loss at epoch 16, after which no significant improvements were observed as seen in Figure 37. The model performed well in structured datasets, achieving reasonable CTC loss values during training and validation. However, when applied to rotated coin images in Figures 38 and 39, the predictions were not accurate.

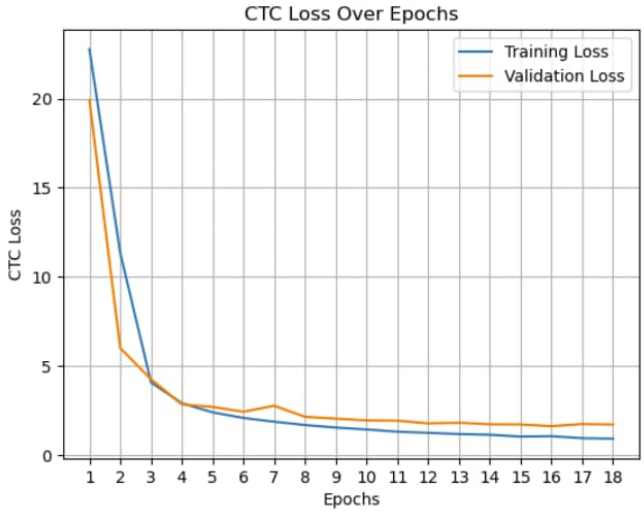


Figure 37: This plot show how well the CRNN model can recognize text symbols from images used for training.

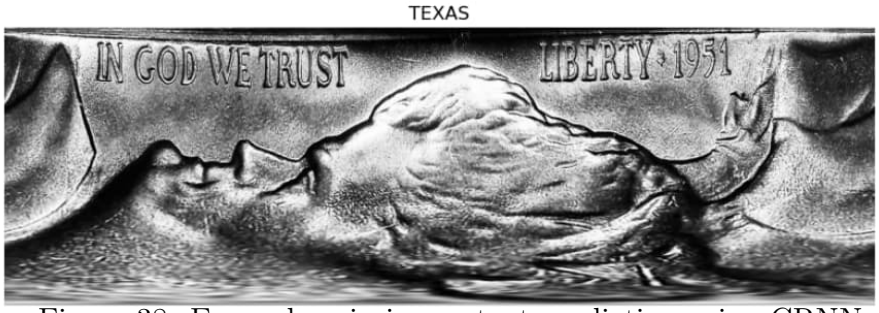


Figure 38: Example coin image text prediction using CRNN model.

Figure 39: Example coin image text prediction using CRNN model.

4.4 Keras OCR Results

To improve text recognition, Keras OCR was tested on coin images. Unlike the CRNN model, Keras OCR first detects text regions before recognizing the characters, leading to better results. The method was able to identify and extract text from coin images more effectively, even when dealing with curved or irregularly placed text. Orientation had a major impact on accuracy: straightened images gave better results but there appeared numeric misclassifications also as seen in Figure 40.

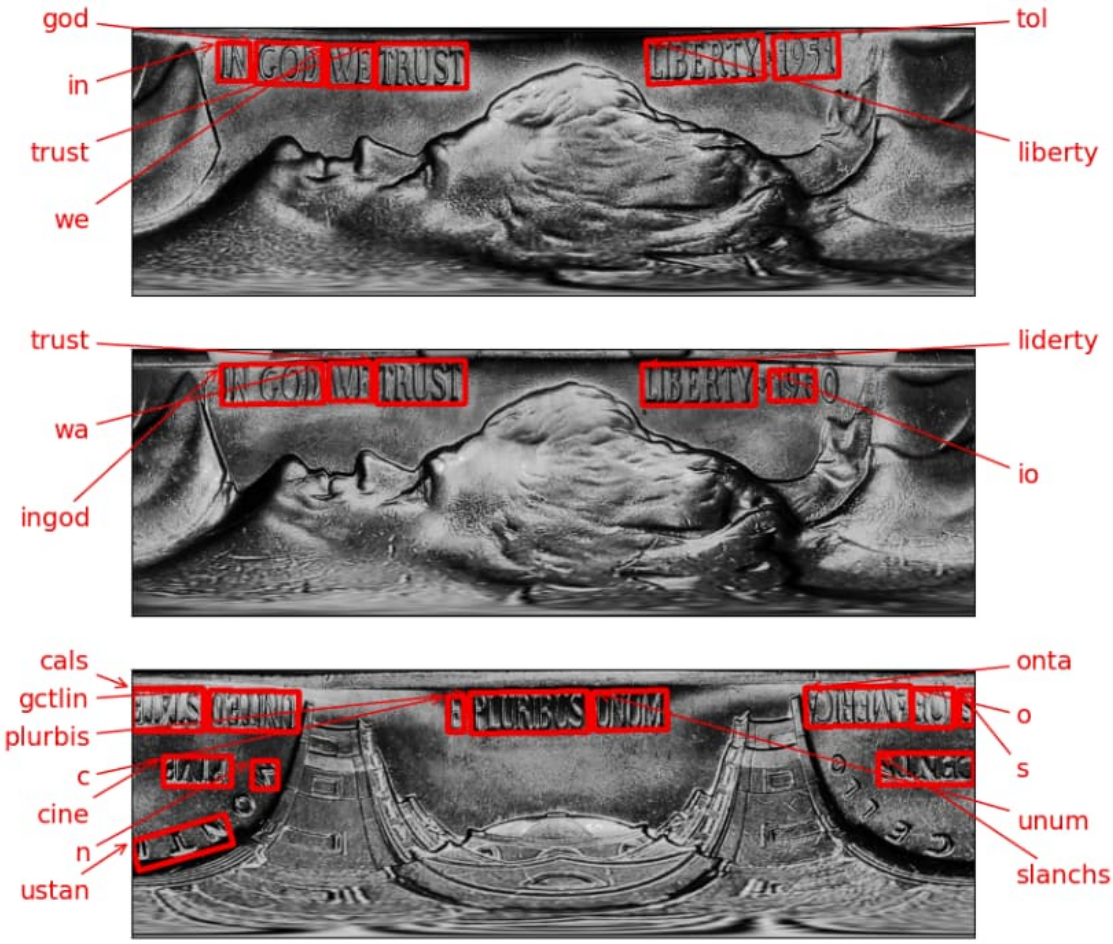
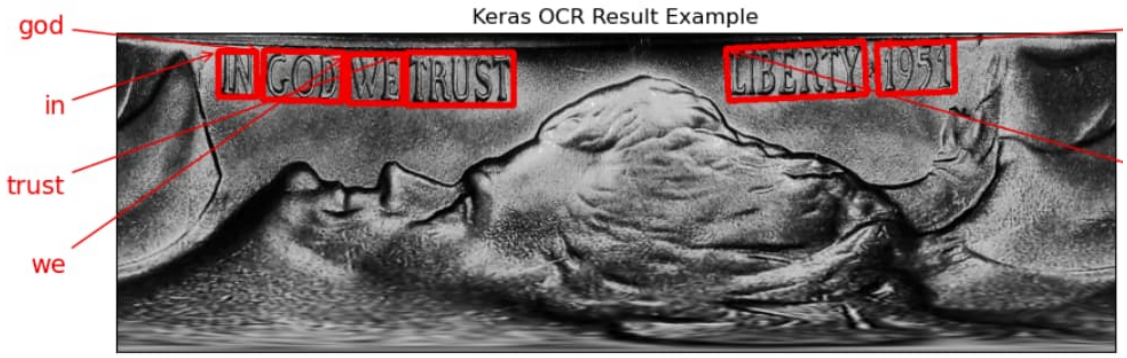


Figure 40: These example results illustrates how Keras OCR results with straighted images improves text detection compared to the CRNN model.

4.5 NLP-Based Text Correction Results

To further improve text recognition, a language model (GPT-3.5) was used to correct the OCR-extracted text and categorize coins in Figures 41 and 42. Cosine similarity analysis was used to measure the difference between the raw OCR output and the corrected text. A similarity score of 0.63 indicated that the extracted text was fairly close to the correct result, indicating a moderate improvement.



{"id": "chatcmpl-AzhhvTqRvhPD5pXehE4CuFeOiZayN", "choices": [{"finish_reason": "stop", "index": 0, "logprobs": null, "mess
ust, liberty for all.", "role": "assistant", "function_call": null, "tool_calls": null, "refusal": null}}], "created": 173
o-0125", "object": "chat.completion", "service_tier": "default", "system_fingerprint": null, "usage": {"completion_tokens"
l_tokens": 36, "prompt_tokens_details": {"cached_tokens": 0, "audio_tokens": 0}, "completion_tokens_details": {"reasoning_
"accepted_prediction_tokens": 0, "rejected_prediction_tokens": 0}}}

Cosine Similarity: 0.6327904583679949

GPT-3 Similarity: In God we trust, liberty for all.

Figure 41: Correcting the extracted texts with GPT-3.5 model.

{"id": "chatcmpl-Azht19dgup3eMZvBFUIvXAq6fGhqY", "choices": [{"finish_reason": "stop", "index": 0, "logprobs": null, "mess
text \"In God we trust, liberty for all,\" the coin likely belongs to the category of 'Washington Quarters, 1932-1998'. Wa
ture the motto \"In God We Trust\" and symbols of liberty such as Lady Liberty and the American bald eagle. ", "role": "as
l, "tool_calls": null, "refusal": null}}], "created": 1739270117, "model": "gpt-3.5-turbo-0125", "object": "chat.completic
"system_fingerprint": null, "usage": {"completion_tokens": 62, "prompt_tokens": 98, "total_tokens": 160, "prompt_tokens_de
udio_tokens": 0}, "completion_tokens_details": {"reasoning_tokens": 0, "audio_tokens": 0, "accepted_prediction_tokens": 0,
0}}}

Cosine Similarity: 0.40458082617324087

GPT-3 Similarity: Based on the text "In God we trust, liberty for all," the coin likely belongs to the category of 'Washir
ington Quarters typically feature the motto "In God We Trust" and symbols of liberty such as Lady Liberty and the Americar

Figure 42: Categorization based on the corrected text with GPT-3.5 model.

5 Discussion

This chapter discusses the key findings of the study in relation to the research questions presented in the Introduction. The results reveal clear performance differences between CPR methods and modern DL approaches, particularly in the context of degraded historical coin images. In addition, the ability of generative language models to analyze coin image texts and to improve the accuracy of OCR output of historical coins is discussed.

5.1 Interpretation and Analysis of Key Findings

RQ1: How do DL models (e.g. CNN, CRNN) perform in classifying real-world coin images compared to Classical Pattern Recognition (CPR) methods?

The results showed clear performance differences between the CPR and DL models. CPR techniques such as LBP and edge detection were useful in ideal conditions, but their performance decreased significantly in the coin images. LBP, for example, struggles with uneven lighting and fails to recognize text patterns, especially in low-contrast regions or where text is small or partially degraded. Edge detection is effective in highlighting boundaries, but sensitive to noise and parameter settings such as kernel orientation or wavelength, which can make it unreliable across varying image conditions. These limitations restrict the effectiveness of CPR techniques in more complex and degraded historical images.

CNN models significantly outperformed CPR techniques by successfully extracting and classifying visual features, even from rotated or noisy coin images. Their ability to learn spatial hierarchies and abstract visual patterns made them more robust in identifying relevant features and classifying coin images into correct classes. For example, CNNs maintained high accuracy and recall across multiple classes and were not as sensitive to noise or orientation as classical methods.

The CRNN model further improved performance by incorporating recurrent layers, allowing it to recognize sequential patterns and generate coherent text outputs. It performed well on structured datasets and straight inscriptions. However, its accuracy decreased on coins with long or curved texts, revealing a structural limitation. CRNN assumes a fixed reading order, which fails when the text layout is irregular or circular.

Conculsion for RQ1: DL models, especially CNN and CRNN, provided a significant improvement over classical CPR methods in classifying and interpreting coin images. However, CRNN performance was limited when dealing with complex and non-linear text layouts commonly found on historical coins.

RQ2: Can modern OCR models improve text extraction from degraded historical coin images?

To address the limitations of sequential models such as CRNN, this study evaluated the performance of pre-trained Keras OCR model, which first detects text regions and then recognizes individual characters. Keras OCR is an effective method to

recognize text symbols and characters in diverse and complex images. Compared to the single CRNN model, it can find smaller text regions, which processes entire image input rather than focusing on local text areas. This combination makes Keras OCR an effective tool in different text extraction challenges, and it can recognize texts in different formats. In addition, Keras OCR offers the straightforward method for text extraction, because it has been pre-trained with several complex images. Therefore, it is not necessary to annotate any new data.

Image orientation played a major role in text recognition accuracy in the case of coin images. Straightened coin images led to significantly better results, indicating that pre-processing steps such as image rotation can enhance performance. Despite these improvements, some challenges remained, such as occasional confusion between numbers and letters.

Conculsion for RQ2: Modern OCR systems such as Keras OCR significantly enhance text extraction from complex and degraded coin images by utilizing modular architectures and robust detection mechanisms. Their performance is further improved with appropriate pre-processing.

RQ3: Does GPT-based correction improve the accuracy of OCR output on historical coins?

Based on the results, the GPT-3.5 model used in this study provides significant advantages in correcting the text extracted from the images and categorizing the images according to the refined text. One of its key strengths is its ability to understand context and infer missing or incorrect words, which is particularly useful when dealing with OCR-generated text that may contain errors due to image noise or poor lighting conditions. By utilizing DL and extensive pre-training on large text corpora, the model effectively transforms raw OCR outputs into more coherent and semantically meaningful descriptions. Additionally, the model's generative capabilities enable it to provide detailed and structured category predictions for images based on the refined text, enhancing classification accuracy.

For example, when OCR produced partial dates or broken fragments of legends, GPT-3.5 often inferred likely completions or corrections, yielding more semantically consistent results. Cosine similarity analysis between the raw and corrected text in example image showed a similarity score of 0.63, indicating a moderate but meaningful improvement in output quality.

Conculsion for RQ3: The integration of GPT-3.5 as a language-based correction tool significantly enhanced the accuracy of OCR outputs, especially in cases where initial recognition was uncertain or incomplete. This demonstrates the efficacy of incorporating contextual understanding into the OCR pipeline for historical data.

5.2 Limitations

While the results of this study demonstrate clear advantages of DL models and modern OCR methods over CPR methods in the analysis of historical coin images, several limitations were identified that affect the generalizability and robustness of the approaches.

Limitations of DL Approaches (CNN, CRNN)

Although CNNs demonstrated strong performance in learning features from coin images and classifying them, the CRNN model exhibited structural constraints due to its sequential nature. It struggled with non-linear text layouts, such as curved inscriptions, which are common in historical coins. Moreover, CRNN relies on the CTC loss, which is highly sensitive to hyperparameter tuning. In addition, the need for large, well-labeled training datasets presents a challenge, particularly in historical contexts where annotated data is rare. CRNN also failed to generalize well from synthetic training data to real-world coin images with more diverse text styles and noise.

Limitations of Keras OCR

Despite many advantages, Keras OCR can include some challenges. Because Keras OCR is one of the DL methods, in the case of large images, it may need computing power. Although the model is good for complex images, curved, diagonal, or upside-down texts can be challenging and cause detection errors. In addition, there can be poor quality images, which include dark points, for example. These cases require certain image pre-processing steps. Pre-processing steps, like random rotation, is not easy in the case of all images.

Limitations of GPT-3.5

The use of GPT-3.5 as a correction tool showed promise, but came with its own set of limitations. The model may introduce biases or incorrect assumptions when correcting text, especially in cases where domain-specific terminology or rare words are involved. Furthermore, while cosine similarity can quantify how well the corrected text aligns with the original OCR output, it does not always capture the true semantic improvements made by the model. Another drawback is the model's dependency on input quality. If the extracted text is highly incomplete, the correction process may produce inaccurate results. Additionally, while the model can classify images based on textual descriptions, it lacks direct image understanding, which may lead to misclassifications when text alone is insufficient for accurate categorization. In general, GPT-3.5 offers a powerful tool for refining OCR-extracted text and assisting in image classification, but its performance is influenced by factors such as input text quality, context comprehension, and inherent biases in training data.

5.3 Future improvements

Based on the results of this study, several areas for future improvements have been identified. One crucial aspect involves improving preprocessing techniques to better handle challenges related to curved and low-context text in coin images. Improving lighting adjustments, contrast enhancement, and noise reduction methods could significantly enhance text visibility and facilitate more accurate recognition.

Another potential improvement involves exploring OCR models specifically trained to handle irregular or rotated text. Although the models used in this study showed

promising results, their performance was limited by the unique characteristics of historical coins, such as curved inscriptions and low contrast between text and background. Developing or fine-tuning OCR systems to account for these features could lead to more accurate results.

Additionally, fine-tuning the NLP model with a dataset containing historical coin inscriptions may improve its ability to correct extracted text and enhance recognition and classification accuracy. Since OCR misclassifications often involve numbers and rare symbols, incorporating domain-specific training data could refine the text correction process and improve the reliability of the classification system.

Further work could also explore the integration of additional DL architectures, such as transformer-based models, to enhance both text detection and PR. Combining these with existing CNN and CRNN models may lead to more comprehensive and adaptive recognition. In general, these advances could contribute to the development of more robust and accurate systems to identify and analyze historical coin images, ultimately supporting broader efforts in the investigation of cultural heritage.

6 Conclusion

6.1 Summary of the thesis

The Introduction in Chapter 1 started with a brief introduction about the importance of text extraction and the relevant methods for this task. It also discussed what kind of PR, DL and developed NLP models can be applied and how these methods can be used statistically. In particular, the investigation of cultural and historical goods has received attention because of their accessibility through digital libraries and the internet. This has become increasingly relevant in recent years to prevent the illict trafficking of CH artifacts. Three significant research questions of these thesis were introduced as well

In Chapter 2, a theoretical background of this study was reviewed. The chapter presented the main ideas of PR and different PR techniques, including CPR and DL methods, which are useful in image recognition and feature extraction. It was introduced the concept of PR and its ability to real-world applications such as image classification and text recognition. The fundamental process of PR were explained, including feature extraction, feature selection, and classification. In addition, statistical and texture-based PR methods were explained, demonstrating how PR can be applied to analyze patterns and textures in images. The texture-based methods such as LBP and Edge Detections are assumed to be more useful in the case of complex images such as coin images. Furthermore, DL techniques such as CNNs, RNNs and OCR were introduced as effective methods for automating PR tasks.

Chapter 3 covered the experiments of the CPR and DL methods introduced in Chapter 2 with the coin image data used in this study. The section also described and visualized the results of the investigation and their effectiveness, especially in extracting features and recognizing text symbols. The first investigation was based on textual-based CPR methods such as LBP and Edge Detection. These methods were used to enhance the clarity of the image and extract text symbols by improving contrast, reducing noise, and highlighting key patterns in the images. The second investigation focused on the DL methods such as CNN model, CRNN and Keras OCR, and visualized their statistical accuracies to recognize features and extract text symbols. In addition, different random rotation techniques were applied to improve the accuracy of the results. At the end of this chapter, the GPT-3.5 language model was applied to correct misidentified words in text extraction results and to compare extracted texts with each other using mathematical and statistical measurements such as cosine similarity.

Chapter 4 presented the summaries of the experiment results of Chapter 3. For each method used in the investigation, the ideas of the methods and their achievements were introduced.

References

[1] Patias, Petros, and Charalampos Georgiadis. "Fighting Illicit Trafficking of Cultural Goods—The ENIGMA Project." Remote Sensing 15.10 (2023): 2579.

[2] Manzoor, Sehrish, et al. "Ancient coin classification based on recent trends of deep learning." VIPERC. 2022.

[3] Sharma, Neha, Vibhor Jain, and Anju Mishra. "An analysis of convolutional neural networks for image classification." Procedia computer science 132 (2018): 377-384.

[4] Image Text Recognition Using CNN and RNN. (2020). Analytics Vidhya on Medium. https://medium.com/analytics-vidhya/image-text-recognition-738a368368f5

[5] Shi, Baoguang, Xiang Bai, and Cong Yao. "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition." IEEE transactions on pattern analysis and machine intelligence 39.11 (2016): 2298-2304.

[6] Hannun, Awni. "Sequence modeling with ctc." Distill 2.11 (2017): e8.

[7] Deepa, R., et al. "An enhanced machine learning technique for text detection using keras sequential model." 2023 Second International Conference on Electronics and Renewable Systems (ICEARS). IEEE, 2023.

[8] Bajić, Filip, and Josip Job. "Data extraction of circular-shaped and grid-like chart images." Journal of imaging 8.5 (2022): 136.

[9] Rosebrock, A. (2015). Local Binary Patterns with Python & OpenCV. PyImageSearch. https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv

[10] Shah, A. (2018). Through the Eyes of Gabor Filter. Exploring Neurons on Medium. https://medium.com/@anuj_shah/through-the-eyes-of-gabor-filter-17d1fdb3ac97

[11] Zelioli, Luca. "LEVERAGING MACHINE LEARNING FOR MARITIME OBJECT DETECTION AND PEATLAND CLASSIFICATION."

[12] Sarika, Naragudem, Nageswararao Sirisala, and Muni Sekhar Velpuru. "CNN based optical character recognition and applications." 2021 6th International conference on inventive computation technologies (ICICT). IEEE, 2021.

[13] Ahlawat, Savita, et al. "Improved handwritten digit recognition using convolutional neural networks (CNN)." Sensors 20.12 (2020): 3344.

[14] Wu, Jianxin. "Introduction to convolutional neural networks." National Key Lab for Novel Software Technology. Nanjing University. China 5.23 (2017): 495.

[15] He, Wenze, and Hao Wu. "Ancient And Modern Coin Recogintion Using a Novel Rotation-Invariant Convolutional Neural Network." 2021 6th International Symposium on Computer and Information Processing Technology (ISCIPT). IEEE, 2021.

[16] Jain, Anil K., Robert P. W. Duin, and Jianchang Mao. "Statistical pattern recognition: A review." IEEE Transactions on pattern analysis and machine intelligence 22.1 (2000): 4-37.

[17] Mendlovic, David, Zeev Zalevsky, and Haldun M. Ozaktas. "Applications of the fractional Fourier transform to optical pattern recognition." Optical pattern recognition (1998): 89-125.

[18] Pietikäinen, Matti. "Local binary patterns." Scholarpedia 5.3 (2010): 9775.

[19] Mendlovic, David, Zeev Zalevsky, and Haldun M. Ozaktas. "Applications of the fractional Fourier transform to optical pattern recognition." Optical pattern recognition (1998): 89-125.

[20] Fogel, Itzhak, and Dov Sagi. "Gabor filters as texture discriminator." Biological cybernetics 61.2 (1989): 103-113.

[21] Chen, Yushi, et al. "Hyperspectral images classification with Gabor filtering and convolutional neural network." IEEE Geoscience and Remote Sensing Letters 14.12 (2017): 2355-2359.

[22] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

[23] Graves, Alex, et al. "A novel connectionist system for unconstrained handwriting recognition." IEEE transactions on pattern analysis and machine intelligence 31.5 (2008): 855-868.

[24] Haralick, Robert M., Karthikeyan Shanmugam, and Itshak Dinstein. "Textural features for image classification." IEEE Transactions on systems, man, and cybernetics 6 (1973): 610-621.

[25] Baek, Youngmin, et al. "Character region awareness for text detection." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.

[26] Reza, Ali M. "Realization of the contrast limited adaptive histogram equalization (CLAHE) for real-time image enhancement." Journal of VLSI signal processing systems for signal, image and video technology 38 (2004): 35-44.

[27] Saharovskiy, S. (2023). US Coins Dataset. Kaggle. https://www.kaggle.com/datasets/sergiosaharovskiy/uscoins/data

[28] Visual Geometry Group. (2014). Text spotting data. University of Oxford. https://www.robots.ox.ac.uk/ vgg/data/text/