University of Brasília

Exact Sciences Institute
Computer Science Department

Optical Music Recognition with Transformers

Hevelyn Sthefany Lima de Carvalho

Monograph submitted in partial fullfilment of
the requirements to Bachelor Degree in Computer Engineering

Advisor
Mr. Dr. Vinícius Ruela Pereira Borges

Brasília
2022

University of Brasília

Exact Sciences Institute
Computer Science Department

Optical Music Recognition with Transformers

Hevelyn Sthefany Lima de Carvalho

Monograph submitted in partial fullfilment of
the requirements to Bachelor Degree in Computer Engineering

Brasília, May 05, 2022

Dedication

I dedicate this work to my father, my dear dad, the most incredible musician I have ever met. The idea for this work came to me while I was watching him composing music in his office for a long time, writing perfectly drawn notes in pencil and editing music on his computer. Although he is no longer in this world, he still teaches me through my memories. I am grateful that I was able to share the subject of this work with him while he was still alive, that I see his joy and have his support. I am grateful to have had the honor of being his daughter. Your love, your laughter, the sound of your trumpet, and the movements of your arms as you conducted your orchestra will always be in my memory.

Acknowledgements

First, I thank God, my good Father, for allowing me to go through all the stages and leading me here. I feel Your care and love every day and my faith will always be in You. Thank you, Lord. I thank my family who are my structure, my rock and my best friends. You have given me much support and encouragement. They have been instrumental in making this work a success. I also thank my friends who have also encouraged me. I am especially grateful for those who shared the beautiful trajectory we had in college, and for those in which I was able to exchange words of encouragement during this work.

I would like to thank all the professors of this educational institution who have taught me so much and who have contributed a great deal to the realization of this work. Teachers who made my academic education possible with their teachings. I am especially grateful to my advisor, Vinicius Borges, who has guided me along the way to this course final paper. I thank you for your friendship and for the time and attention you have given me in our work together, especially in conducting this research. If I have made it this far, it is because I have stood on the shoulders of giants. And finally, I would like to thank all the staff of this institution who have contributed to my development and supported me throughout the course.

Much love and thanks to all of you.

Abstract

Optical Music Recognition (OMR) is a research field concerned with the analysis of music notations on documents or digital surfaces. It is divided into two approaches: offline OMR, which deals with the analysis of digitised handwritten scores, and online OMR, which involves the analysis of musical notation written on a digital surface. Although these tasks have been explored in the last years, both approaches still offer challenges and research opportunities. The most promising results so far have been obtained by using convolutional neural networks to classify musical symbols. However, taking advantage of the recent development of self-attention architectures, this work presents a method for recognising musical symbols in online and offline data using Transformers for image classification. Experiments were performed in order to validate the proposed method on six publicly available standard datasets, namely Handwritten Online Music Symbols (HOMUS), Seoul National University (SNU) Dataset for Online Music Symbol Recognition, Capitan_Score_Uniform, Capitan_Score_Nonuniform, Rebelo_real, and Fornés. Three pre-trained transformer models were tested with the datasets and their performances compared: BEiT from the Microsoft team, ViT from the Google team, and DEiT from the Facebook team. The proposed method achieves recognition accuracy that is close to the state-of-the-art researches, and thus proves to be a promising approach. In general, the proposed method provided results above 98%, with the model using the DEiT architecture showing the best performance in most cases. For example, DEiT achieved 99.12% of the F1 score, exceeding the 97.48% score of the ensemble method proposed by Paul [1] with Homus data.

Keywords: optical music recognition, omr, transformer, self attention, music, music notation

Resumo

O Reconhecimento Óptico de Música (OMR) é um campo de pesquisa voltado para a análise de notações musicais em documentos ou superfícies digitais. Divide-se em duas abordagens: OMR offline, que trata da análise de partituras manuscritas digitalizadas, e OMR online, que envolve a análise da notação musical escrita em uma superfície digital. Embora essas tarefas tenham sido exploradas nos últimos anos, ambas as abordagens ainda oferecem desafios e oportunidades de pesquisa. Os resultados mais promissores até então foram por meio de redes neurais convolucionais para classificar símbolos musicais. No entanto, aproveitando o recente desenvolvimento de arquiteturas de autoatenção, este trabalho apresenta um método para reconhecimento de símbolos musicais em dados online e offline utilizando Transformers para classificação de imagens. Experimentos foram realizados para validar o método proposto em seis conjuntos de dados padrão disponíveis publicamente, a saber, Handwritten Online Music Symbols (HOMUS), conjunto Seoul National University (SNU) para Reconhecimento de Símbolos de Música Online, Capitan_Score_Uniform, Capitan_Score_Nonuniform, Rebelo_real e Fornés. Três modelos de transformadores pré-treinados foram testados com os conjuntos de dados e seus desempenhos comparados: BEiT da equipe da Microsoft, ViT da equipe da Google e DEiT da equipe do Facebook. O método proposto alcança uma precisão de reconhecimento próxima das pesquias do estado da arte e, portanto, mostra-se uma abordagem promissora. Em geral, o método proposto apresentou resultados acima de 98%, sendo que o modelo utilizando a arquitetura DEiT apresentou o melhor desempenho na maioria dos casos. Por exemplo, o DEiT obteve 99,12% de pontuação F1, superando a pontuação F1 de 97,48% do método ensemble proposto por Paul [1] com dados do Homus.

Palavras-chave: reconhecimento óptico de música, omr, transformador, auto-atenção, música, notação musical

Contents

List of Figures

List of Tables

Glossary

ANN	Artificial Neural Networks.
ASHA	Async Successive Halving Algorithm.
BEiT	Bidirectional Encoder representation of Image Transformers.
BERT	Bidirectional Encoder Representations from Transformers.
BOHB	Bayesian Optimization and Hyperband.
BRIEF	Binary Robust Independent Elementary Features.
CNN	Convolutional Neural Network.
CPU	Central Process Unit.
DEiT	Data-Efficient Image Transformers.
DTW	Dynamic Time Warping.
FAST	Features from Accelerated Segment Test.
FFC	Freeman Chain Code.
GPU	Graphics Processing Unit.
HMM	Hidden Markov Models.
HOG	Oriented Gradient Histogram.
HOMUS	Handwritten Online Music Symbols.
ILSVRC	ImageNet Large Scale Visual Recognition Challenge.
KNN	K-Nearest Neighbor.

MIDI Musical Instrument Digital Interface.

MLP Multi-layer Perceptron.

NB Naïve Bayes.

OCR Optical Character Recognition.

OMR Optical Music Recognition.

ORB Oriented FAST and Rotated BRIEF.

PBT Population Based Training.

RBF Radial Basis Function.

RF Random Forest.

SIFT Scale-invariant Feature Transform.

SMO Sequential Minimum Optimization.

SNU Seoul National University.

SURF Speeded Up Robust Features.

SVC C-Support Vector Classification.

SVM Support Vector Machine.

TPE Tree-Structured Parzen Estimator.

VAE Variational Autoencoder.

ViT Vision Transformer.

Chapter 1

Introduction

Traditionally, music can be read and written using musical notation, like the modern Western [1] [2] music notation known today, which can be considered the universal language of music. From using paper to computers, music can be written in music composition software using the mouse or an instrument connected to the computer. Despite the complexity and robustness that such software has gained over time, many composers still prefer to express their new musical compositions using pen and paper for simplicity and practicality. Sticking to a limited menu, predefined styles, and point-and-click mouse actions can be tedious for artists. Consequently, the need for automatic recognition of handwritten musical symbols led to the emergence of Optical Music Recognition (OMR) as a research field for analyzing musical notations in documents [10].

Automatic analysis of musical notation images has been a long-standing task, as shown by Blostein and Baird in [11], who presented an overview and critical analysis of the problems and proposed solutions in the processing of music notation images from 1966 to 1990. There are two types of input data for OMR approaches and systems, referred to in the scientific community as offline data and online data. The taxonomy of these two input types is the main difference between the two domains. While one operates on static images (offline), the other operates on input data from an electronic pen that is captured over the time (online).

Typically, a OMR offline approach is divided into a four-stage pipeline [12]: (1) pre-processing and binarization, (2) staff removal, (3) symbol localization and detection, and (4) notation reconstruction. With the recognition of musical symbols and semantic recovery of the set of individual elements of the recovered musical text, it is possible to create

[1]Western: Concerning European and European-colonized countries
[2]How did music notation actually begin, accessed: May 17, 2022

digital files in the format MIDI [3], which can be processed, for example, by music software. Some researchers have created small datasets proposing offline OMR methods, such as Fornés et al. [13], who used Dynamic Time Warping (DTW) algorithm to classify 2128 handwritten musical symbols with seven clefs from modern and old (19th century) scores of different authors. The same dataset was used by Nawade et al. [14, 15], in which the recognition was performed by the K-Nearest Neighbor (KNN) algorithm. Chanda et al. [16] extracted Freeman code histograms and Zernike moment-based features from offline music notation data and classified them using Support Vector Machine (SVM). In another work, Malakar et al. [8] used the texture-based feature descriptor Daisy.

To take advantage of the new tablet technologies, the OMR study area have also been established in the online form, which processes musical notation as it is written on a digital surface using an electronic pen to allow its representation in a digital format. This process is similar to a continuous digitization of writing, converting the drawn symbols into a computer-readable format. Such a research area is very similar to the field of Optical Character Recognition (OCR), in which pen-based character processing has been proposed by Plamondon and Srihari [17], Swethalakshmi et al. [18] and Zhang et al. [19]. OMR online is also a long-standing [20] area of study. Jorge [21] proposed a pen-based online recognition method for 20 music symbols using a multi-layer perceptron classifier, achieving a recognition accuracy around 80% on unseen test data. Miyao and Maruyama [22] describe an online music symbol recognition method in which they used Freeman string code to represent time series data and Dynamic Programming for the model-based symbol recognition process. Lee et al. [23] proposed a approach using local and global resources along with the Hidden Markov Models (HMM) classifier for a data set with 8 different musical symbols.

In the absence of online notation data sets, Calvo-Zaragoza and Oncina [24] created HOMUS (Handwritten Online Music Symbols) and applied different methods online and offline, comparing the recognition results, which can be dependent or not on the symbols's writer. Oh et al. [25] classified online music notation data in two levels: recognition of strokes separately and recognition of symbols by combining the prediction of features and other features of symbols. The features extracted from tracks categorized by the authors include magnitude information, histogram of directional movement angles, and the histogram of undirected movement angles. Particularly, the last two mentioned features are based on the Oriented Gradient Histogram (HOG) [26]. The features extracted from a symbol, combined with the recognition of the strokes, contain information about the location and size of its composing strokes. Furthermore, Calvo-Zaragoza and Oncina [27]

[3]File MIDI is a file format that provides a standardized way to store, transport, and open musical sequences on other systems. These files contain information such as note values, time, and track names. Think of the MIDI file format as the musical version of a ASCII text file.

proposed another OMR method using finite state machines and dissimilarity measures. A finite set of feature primitives was considered and then labeled data were used to learn a language describing musical symbols obtained from isolated feature primitives. The traces obtained as input were mapped to the probability of representing each of the primitives considered. In addition, the authors also used a semantic model describing which musical sequences are formally acceptable. Using both sources, the underlying image (offline data) and the traces (online data), the approach proposed by Sober-Mira et al. [7] was able to achieve an error of less than 4% in recognizing symbols with a Convolutional Neural Network.

The development of OMR approaches or systems continues to be fraught with challenges, including poor quality notation, complex scores, overlapping symbols, and structural complexity [28]. In such a discussion, it is clear that music notation is a complex system and requires specific approaches for its recognition, which makes it difficult to obtain efficient approaches in addition to good accuracy. OMR is a rich and challenging research area. Moreover, to the best of my knowledge, there are no studies in the literature that use transformers for classifying musical symbols. The transformer uses a self-attention architecture, a state-of-the-art model proposed by Vaswani et al. [6] which revolutionized the field of machine learning by eliminating recursion and convolution. For these reasons, I was motivated to get to know and understand this field better in order to contribute with new approach.

Considering the two data formats, this work proposes an approach for recognizing musical symbols written with a pen on a digital surface (online data) and musical symbols derived from digitized handwritten scores (offline data). The online data, defined by sets of coordinates of the symbol traces, goes through a process to generate the images of the symbols and then follows the same pipeline as the offline data. In other words, the method does not consider the real-time acquisition of online data, but a set already prepared for study purposes that follows the offline approaches when converted into a set of images.

1.1 Goals

This work aims to study the optical recognition of musical notes using transformer models and to contribute with a method that can achieve promising results that can overcome the recognition accuracy of the last generation techniques. In specific, this work aims to:

1. Study the field of optical recognition of musical notation symbols from online and offline data;

2. Explore the transformer model for OMR, which has proven successful in natural language processing and has recently been explored for images;
3. Identify improvements and new solutions to the task of optical recognition of musical notation;
4. Propose an OMR approach that processes online and offline notation data, and learns symbols using transfer learning with pre-trained transformer models "*BERT Pre-Training of Image Transformers*" (BEiT) [9] inspired by the famous BERT transformer [29], "*Vision Transformer*" (ViT) [30] and "*Data-Efficient Image Transformers*" (DeiT) [31];
5. Test the models on six data sets of different nature and evaluate the recognition performance.

In summary, the contributions of this research are:

1. A study on OMR and how transformer-based models can contribute to this research area;
2. Contribute to OMR research by proposing a method with good performance for classifying musical notation;
3. Propose an approach to musical symbol recognition that can be combined with other OMR systems to create a complete OMR system.

1.2 Structure of the document

This document is organized as follows. Chapter 2 introduces important concepts for understanding this Final Paper. Chapter 3 briefly presents some recent work related on symbol recognition and the employed transformer model. Chapter 4 describes the proposed method and its constituting steps. Chapter 5 presents the experiments performed to validate the proposed method and discusses the obtained results. Finally, Chapter 6 concludes the work and presents suggestions for future research.

Chapter 2

Fundamentals

Musical notation is a writing system with well-defined rules and an extensive vocabulary of definitions. Therefore, this chapter summarizes the most important basic definitions that are required for understanding this work. In addition to musical notation, important concepts of visual representation are also introduced, including the type of data in one of the steps of the proposed method, as well as an explanation of the transformer model, which is the core for image transformation and classification.

2.1 Musical Notation

Musical notation is a system of graphical representation of music that allows a musician to perform an arrangement as envisioned by the writer. The symbols (notes and other musical representations) are written on a set of five lines - the pentagram - and form a score.

2.1.1 Musical note

A musical note is an isolated sound event that is usually employed in combination with other events to form a musical text. The most important attributes of the note that can be withdrawn in the score are:

- height: represents the oscillation frequency of the sound, which is higher (high sound) or lower (low sound). Pitches are grouped and arranged in an ascending order called the scale, and are represented by the position of the note on the staff;
- duration: duration is the temporal dimension of notes that describes the duration (time length) or in what units of time the vibration occurs. The combination of notes and rests with different durations in the staff forms patterns, i.e. the rhythm

in music. The time lengths are represented by different symbols as depicted in Table 2.1;

- Intensity: is the parameter of sound that refers to the amplitude of the deflection of the vibrating body or sound source. Intensity is represented by a gradation ranging from molto pianissimo (minimum, almost inaudible sound intensity) to molto fortissimo (maximum sound intensity that can be achieved without damaging the voice or instrument), and is represented by abbreviations in the music sheets.

Table 2.1: Notes and Pauses corresponding to each duration.

Note	Name	Rest	Count
𝅝	whole	𝄻	4
𝅗𝅥	half	𝄼	2
𝅘𝅥	quarter	𝄽	1
𝅘𝅥𝅮	eighth	𝄾	$\frac{1}{2}$
𝅘𝅥𝅯	sixteenth	𝄿	$\frac{1}{4}$
𝅘𝅥𝅰	thirty-second	𝅀	$\frac{1}{8}$

2.1.2 Time Signature

The staff is divided into measures by vertical lines (bars) and determines the rhythmic structure of the music. The time signature is represented by a formula in which the denominator indicates the number of parts into which a whole measure must be divided to obtain a single measure unit, in other words, it indicates whether a beat consists of a half note (2), a quarter note (4), an eighth note (8), or a sixteenth note (16), and the numerator indicates the number of beats that the measure contains. The Figure 2.1 shows that there is a “four by four” measure, meaning that the time unit has a duration of 1/4 of the half note and the measure has 4 beats. In this case, a half note takes up the entire measure.

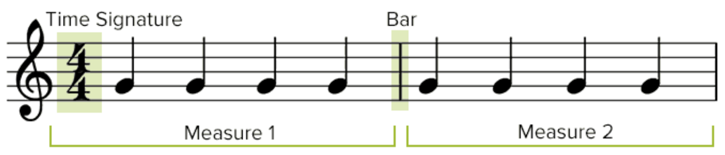


Figure 2.1: 4/4 time [2]: the time unit has a duration of 1/4 of the whole note and each measure must contain 4 beats. Thus, one beat corresponds to a quarter note, and four quarter notes make a measure. The combination of different notes would result in different rhythms.

Musical notation is a complex writing system, comprising a set of graphic signs that symbolize a cadence of sounds. The data sets used in this work contain several other symbols. For a deeper understanding, a dictionary of musical symbols can be found on the Dolmetsch platform [1]. However, the definitions that were presented here are sufficient to understand the data and the proposed method. In one of the processing steps of the proposed approach, the strokes drawn by the pen are converted into images by the digital surface. The next section details this process.

2.2 Image

Pixel, a combination of the terms *"picture"* and *"element"*, i.e. *"picture element"*, is the smallest unit in a digital image. Each pixel of the image has a bit depth, which is related to the number of color channels. The more bits per pixel, the more color possibilities and higher color accuracy are possible. Example:

- An image with a bit depth of 1 has pixels with two possible values: black and white.
- An image with a bit depth of 8 has 28 or 256 possible values.
- Images in grayscale mode with a depth of 8 have 256 possible gray values.
- RGB images consist of three color channels. An RGB image presenting 8 bits per pixel has 256 possible values for each channel, i.e. more than 16 million possible color values.

An image can be represented by a matrix that contains the intensity values and whose size is defined by the width and height of the image. In other words, if the size is 1920×1080 (Full HD), there are 1920 pixels horizontally and 1080 vertically. This means that there is a total of $(1920 \cdot 1080) = 2073600$ pixels. Generally, a color image has 3 color channels: Red, Green and Blue, and each pixel is assigned 3 values as depicted in Figure 2.2. In this sense, an RGB image can display up to 16 million different colors. In addition to RGB, the alpha channel can be considered the fourth element of color and determines a pixel's transparency. A grayscale image is represented by a 2D matrix, in which each pixels is denoted by a single intensity value between zero and 255 (or between zero and one for normalized values).

The number of pixels that defines an image, also called resolution, determines its quality, and the more pixels, the higher the quality. For image processing tasks, however, working with high resolution images would result in high computational processing.

[1]Music dictionary, accessed: May 17, 2022

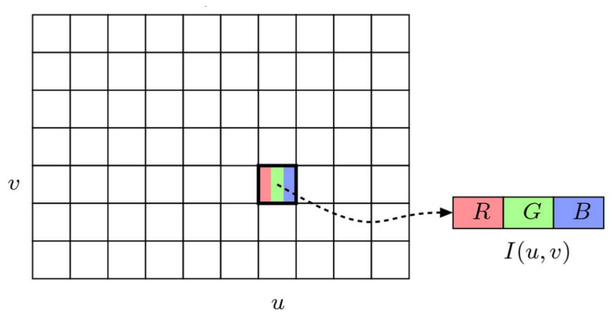


Figure 2.2: Vector of one-pixel color channels in an RGB image [3].

Nevertheless, a high resolution is not necessary, because the shape of the image content, especially in minimalistic and high-contrast images, can already be expressed by a few pixels. For a better representation of the image, the the features extracted from the images can also be used as input to a classifier, instead of its raw form. This process is called "feature extraction".

2.3 Extraction of image features

Image feature extraction consists of determining the most compact and informative feature sets for efficient image processing or storage. The most common and convenient way to represent images in classification and regression tasks is to create feature vectors, where each element represents a feature or attribute resulting from a quantitative or qualitative measurement [32]. In this way, the sample is summarized in a table where each row corresponds to an image and the columns represent the attributes. Attributes can describe texture, contours, convexity, solidity, and other visual properties of the image. It is desirable that the extracted features from an input image present invariance to rotation, translation, and scaling so that classification is not affected by these transformations. The main approaches to feature extraction are presented below.

2.3.1 Contours and Corners

Contours are curves that connect all contiguous points (along the boundary of an object in the image) that have the same color or intensity, which in turn supports the shape identification and the estimation of the object's size. Therefore, contour extraction is useful in object detection and recognition tasks. However, it may be useful and sufficient to extract only the corners of objects that can be identified by the coincidence of two edges, or other methods such as the Moravec detector [33] and the Harris detector [34].

Alternatively, it is also possible to identify only the most important corners, called key points, which can be found by the detectors Scale-invariant Feature Transform (SIFT) [35], Speeded Up Robust Features (SURF) [36], which is an accelerated version of SIFT, and Oriented FAST and Rotated BRIEF (ORB) [37] based on the Features from Accelerated Segment Test (FAST) [38] and Binary Robust Independent Elementary Features (BRIEF) [39] detectors.

2.3.2 Direction

Directional or gradient features can also be extracted from images. The magnitude of the gradients in an image is very large at edges and corners due to the abrupt changes in intensity. Therefore, the gradient of an image also contains relevant information about the object's shape. Directional features can also be extracted from the contour strokes of an object by encoding them according to some predefined directions and grouping them into a "directional chain". HOG [26] is an example of a descriptor that computes the oriented gradient histogram of an image. The Freeman Chain Code (FFC) [40] computes the representativeness of the boundary by a contiguous sequence of straight line segments of certain length and direction, based on 4 or 8 predefined directions (codes), as shown in Figure 2.3.

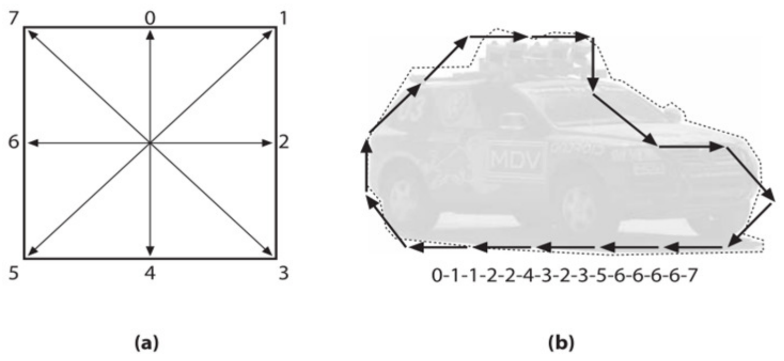


Figure 2.3: Freeman Chain Code, (a) starting from 0 to 7 and (b) code generated covering up the image from the bumper back of the vehicle [4].

2.3.3 Region

Region-based descriptors have also been proposed for characterizing silhouettes of regions and objects within an image. One of the simplest descriptors are the basic geometric features, such as area (number of pixels in the shape), perimeter (number of pixels at the edge of the shape), eccentricity (ratio between the length of the longest chord of the shape and the longest chord that is perpendicular to it), elongation (ratio between the height and width of one of the smallest rectangles that the shape fits into) and rectangularity [41]. The topology of the object can also extract information from the shape, which is a property of the shape that does not change even if the parts of the shape are torn or joined. A well-known topological property is Euler's number E: the number of connected components minus the number of holes H. Shape properties can be extracted by measuring the number or size of concavities in the shape. To do this, the shape of the convex hull itself is subtracted, resulting in holes ("pond") or concavities ("bays"). This can be useful, for example, when trying to distinguish the letter "O" from "C" in handwritten images.

2.4 Image classification

The human perceptual system allows to efficiently recognize objects, regions, and features in an image in new situations based on previous experience. For a computer, however, this image is denoted as a matrix of pixels. Therefore, Computer Vision has become one of the hottest research areas and represents the biggest challenge for machines to understand and make sense of digital images. In this sense, image classification is the task of categorizing images into one of several predefined labels.

2.5 Artificial Neural Networks

Artificial Neural Networks (ANN) are a computational model loosely inspired by the brain and the way humans learn. The perceptron, introduced by Frank Rosenblatt [42], is a neural network in its simplest form, containing only 1 artificial neuron. The perceptron, as shown in Equation 2.1 and Figure 2.4, consists of a layer of input data x_i (in green), a layer of weights w_i (in lilac), a weighted sum that add all the multiplied values, and an activation function. The layes also contais the bias x_0, w_0 (in yellow). The activation function bounds the output amplitude of the neuron, i.e., the value obtained from the sum is normalized within a closed interval, e.g., $[-1, 1]$ or $[0, 1]$ and represents the probability $\hat{y}$ of the outcome. Furthermore, the curve of the activation function is centered by an appropriate bias value.

$$\hat{y} = f(\sum_{i=0}^{n} w_i x_i + b) \tag{2.1}$$

There are several activation functions, such as the ReLU, softmax, tanh, sigmoid, among others, shown in Table 2.2. The sigmoid function looks like an S-shape and is used to calculate the probability of an output, since each probability is in the range [0,1]. However, the ANN may not converge during training because the probability values are

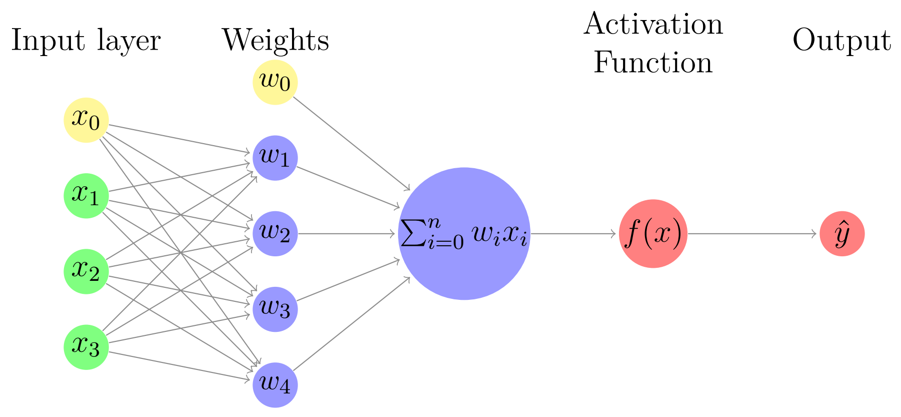
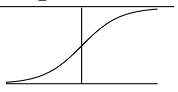
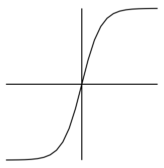
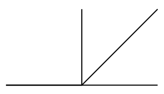


Figure 2.4: Perceptron ANN: a single neuron.

independent. Therefore, the softmax function is more recommended because it calculates the relative probabilities. That is, the softmax function returns the probability values for the membership of the data in each class. The tangent function is like the sigmoid function, but maps the result to the range [-1,1], taking negative entries into account. The ReLU function, on the other hand, maps the negative values to zero, which can have a negative effect on training the data, but has the advantage of being computationally more favourable.

Table 2.2: Non-linear activation functions.

Name	Function	Figure
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	
Tanh	$\sigma(x) = \frac{e^x - e^{-x}}{e^z + e^{-z}}$	
ReLU	$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0. \end{cases}$	
Softmax	$f(x) = \frac{e^x}{\sum_i e^x}$	

Later, Rosenblatt introduced the Multi-layer Perceptron ANN (MLP) [43], which combines neurons in hidden layers. In this topology, shown in Figure 2.5, an input layer receives the input data. Then, in the hidden layer, the outputs of the neurons serve as input to the neurons in the next layer and the output layer that contains the outcome probabilities (y_i). This left-to-right process characterizes the network as a **feedforward** network. In the hidden layers, an activation function activates the neuron or not, i.e., it determines whether the information the neuron receives is relevant to the information provided or should be ignored.

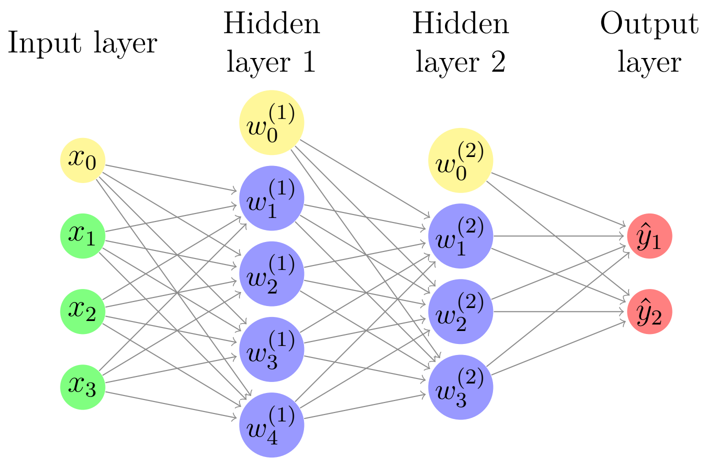


Figure 2.5: An example with a two hidden layers multilayer perceptron and input data with three attributes (x_i) and two labels ($\hat{y}_i$)

First, the weights are initialized randomly. Then David Rumelhart et al. [44] presented the backpropagation algorithm, which adjusts the network weights based on the obtained and expected results. Thus, the training of the network proceeds as follows: after backpropagation, the predicted output is evaluated in relation to the expected output using a cost function, which may be the mean square error function or the cross entropy. The cost determines how much the weights must be adjusted to make the output approach the expected output. Therefore, in the backpropagation step, the updates are propagated from right to left in the hidden layers. In other words, backpropagation aims to minimize the cost function by adjusting the weights and biases of the network.

The extent of network parameter updates depends on the learning rate, a configurable hyperparameter used in neural network training that controls how quickly the model adapts to the problem. Lower learning rates require more training epochs due to the small changes made to the weights at each update and training may not converge, while higher learning rates result in faster changes, require fewer training epochs, and may also lead to the convergence of a suboptimal solution.

Network training occurs in learning cycles. Each cycle (forward propagation followed by backward propagation) is called an epoch. For better generalization of the network, the dataset is divided into batches. The batch size is the number of samples given to the network at once. It also is a hyperparameter that must be tested and adjusted based on the performance of the model during training.

For efficient image classification using neural networks, the Convolutional Neural Network (CNN) was first developed by Fukushima [45], which introduces convolutional layers responsible for identifying features of an image. These layers can be seen as filters that pass through the image, each layer looking for a particular feature (edges, texture, curvatures, colors, etc.). Basically, a CNN consists of convolutional layers interspersed with pooling layers and a regular neural network for classification purposes, as shown in Figure 2.6. The pooling layer reduces the data dimension, extracts dominant features that are rotation invariant and position dependent, and suppresses noise. The most commonly used technique is max pooling [46], in which the maximum value in each patch of each resource card is calculated, thus retaining the most available resource in the patch.

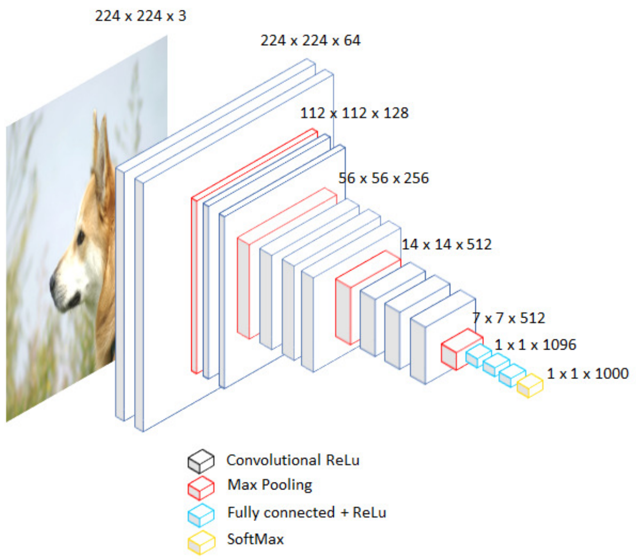


Figure 2.6: Example of a CNN architecture [5].

The number of convolutional layers is proportional to the number of features extracted from an image. However, this increases the cost of memory and processing and requires the use of GPUs and parallel processing to speed up the training of the network. Indeed, one of the drawbacks of CNNs is that a large amount of labeled data is needed to extract the patterns. Therefore, there are researchers who prefer to extract features from images using a computationally cheaper method and then use the result as input to a classifier with a less complex structure, such as C-Support Vector Classification (SVC) or a simple ANN. Another way to take advantage of CNNs success is to use architectures that have already been trained on large sets of images for a particular task. This process is called "transfer learning".

2.6 Transfer learning

According to psychologist Judd [47], *"Every experience has in it the possibilities of generalization"*, and therefore transfer learning is the result of generalizing experience. Transfer learning consists of transferring knowledge between domains, and is a promising machine learning method to solve the problem that for effective training of a model, a large database is needed [48]. For example, knowledge from a model trained to classify images can be used to classify images of vegetables. Using a model that already has some "intelligence" when faced with a new and similar task helps to overcome the problem of depending on large data for machine learning, which is not always possible due to the scarcity or privatization of data. In this way, it is possible to use much less data for the current task, speed up the training process and achieve accurate and effective results.

It is also important to note that transfer learning makes sophisticated and complex models that require large resources, data, time, and computing power more accessible. Therefore, in this work, this practice is applied because there are efficient pre-trained models provided by consolidated research groups, some of which belong to Google [30] and Facebook [31].

2.7 Transformer

To contribute to sequence modeling and transduction tasks, Vaswani et al. [6] proposed the transformer, a model architecture that relies on an attention mechanism to draw global dependencies between input and output, described in his famous paper "Attention Is All You Need".

The transformer model is based on an encoder-decoder architecture with two or three sublayers, as show in Figure 2.7. The encoder consists of a stack of $N = 6$ identical layers. Each layer has a sublayer with a multi-headed self-attention mechanism and a sublayer defined by a simple, fully connected feed-forward network. There is a residual connection and normalization between the sublayers. The decoder also consists of a stack of $N = 6$ identical layers that, in addition to the multi-headed auto-attention mechanism and feed-forward network, have a sub-layer that directs the attention of multiple heads to the output of the stack encoder. Residual connections and normalization are also present in the decoder.

2.7.1 Self-attention sublayer

In this layer, three vectors are computed from an input: Query, Key, and Value, and an attention function maps these vectors to an output, as shown in Figure 2.8. In practice, the attention function is computed simultaneously in a set of queries packed into an array Q, where the keys and values are also packed into matrices K and V, as in the Equation 2.2, where d_k is the dimension of the query and key vectors and T denotes the transverse matrix.

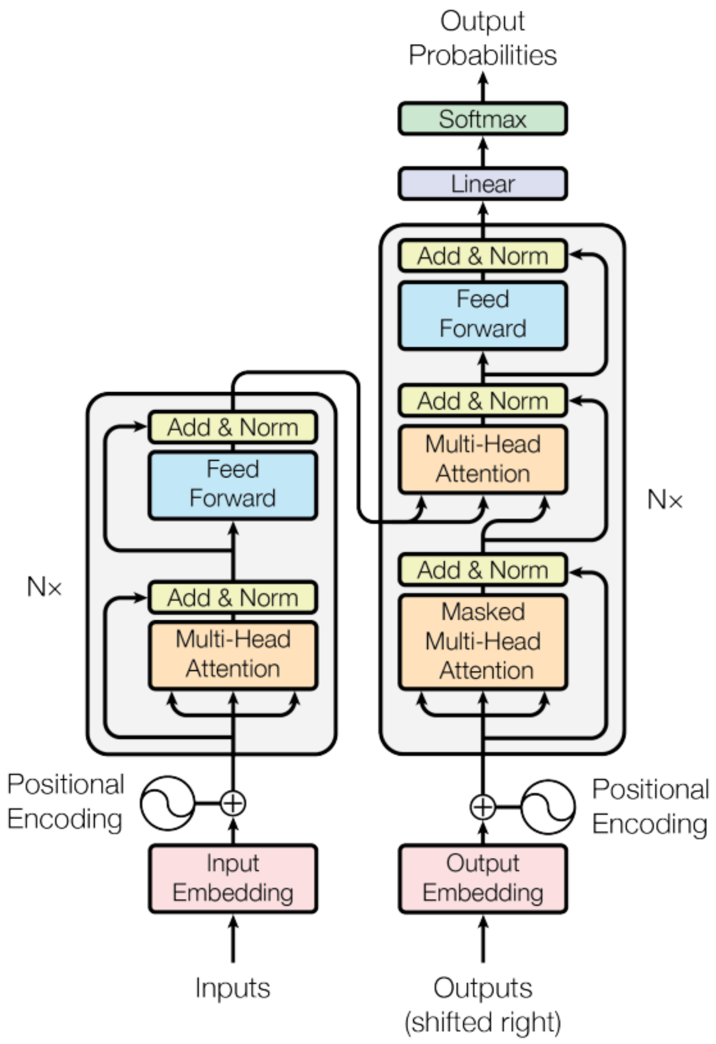


Figure 2.7: Transformer model architecture [6].

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QKQ^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

The query vector is the representation of a single piece of data (e.g., a word in text data) for which self-attention is calculated. The key vector represents each part in the sequence and is used to match the query of the part for which self-attention is calculated. The value vector represents the real part. The dot product between the query vector and the key vector gives the score that reflects the importance of each value, i.e., each data unit, in the self-attention vector.

2.7.2 Multiple Heads

Rather than running a single attention function, the authors found it beneficial to run the described process multiple times with different weight matrices. In each of these designed versions of queries, keys, and values, the attention function is executed in parallel. The outputs are concatenated and re-projected, resulting in the final values, as shown in Figure 2.8 and by the Equation 2.3. Each projection is called a head, and each head learns information from different representative subspaces at different positions in a given sequence.

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(head_1, ..., head_h h)W^O \\ \text{where } head_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.3}$$

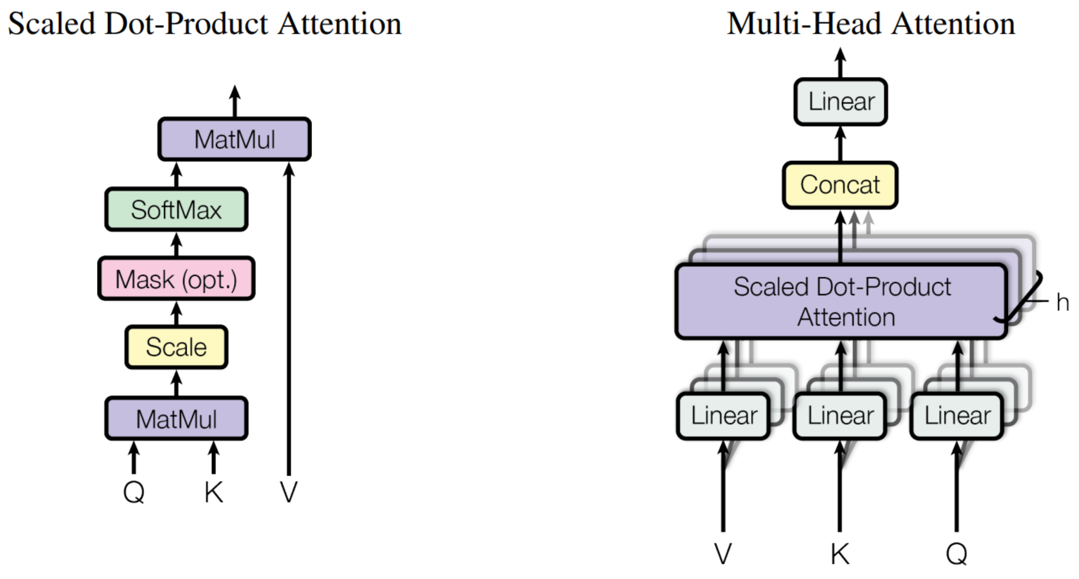


Figure 2.8: (left) Product scales - Attention Layer. (right) Multi-Head Attention with multiple layers of attention working in parallel [6].

2.7.3 Feed-Forward

Each feed-forward network consists of two linear layers with a ReLU function 2.5 between them, as shown in Equation 2.4. The weights and offsets W_1, W_2, b_1, and b_2 are applied identically to each position. However, each encoder and decoder layer has its own feed-forward layer:

$$\text{FFN}(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.4}$$

where W_i represents the weight layers, x the input and b_i the bias values of the i layers.

2.7.4 Positional Encoding and Mask

Since the model does not contain any recursion or convolution where it is possible to know the order of the input sequence, a positional encoding is added to the input embeddings in the encoder and decoder, computed by the sine and cosine functions. Also, some positions in the decoder input are masked by keeping only the words up to the current positions that the decoder knows, since it cannot know the future positions. The authors Wasvani et al. use the sine and cosine functions with different frequencies, presented in Equation 2.5,

$$\begin{aligned} PE_{(pos,2i)} &= \sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= \cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.5}$$

in which pos is the position, i is the dimension, and d_{model} is the dimensionality of input and output. Thus, each dimension of positional encoding corresponds to a sinusoid. The authors chose these functions based on the assumption that the model can account for relative positions, since PE_{pos+k} can be represented as a linear function of PE_{pos} for any fixed displacement k. In addition, the sine function allows the model to work with larger sequences than those found during training.

2.8 Evaluation of Classification Performance

An important step after training a classification model is to evaluate its performance using a set of unknown images to the model, also referred as testing set. Usually such evaluation is performed by measuring the amount of correct predictions in relation to total number of images and its constituting labels. However, there are effective quality assessment metrics for evaluating model performance. I will discuss some of them that have been used in this work.

2.8.1 Confusion Matrix

Confusion matrices are used to evaluate a model's performance on categorical data. Each row of the confusion matrix refers to the predicted label and each column of the matrix represents the actual label. As in Table 2.3, the cells indicate (by numbers or a color legend) the number of occurrences the model generated for each of the labels and provide information about True Negatives (TN), False Negatives (FN), False Positives (FP), and True Positives (TP). For multi-class problems, compute the values for each class: Let i and j be the rows and columns of the matrix, respectively. Each element E_{ij} of the matrix would be the number of items with true class i classified as belonging to label j, as shown in Table 2.3, with labels A, B and C.

Table 2.3: Confusion Matrix for binary and multi-class classification tasks, respectively.

		Actual	
		positive	negative
Predicted	positive	TP	FP
	negative	FN	TN

		Actual		
		A	B	C
Predicted	A	TP_A	E_{BA}	E_{CA}
	B	E_{AB}	TP_B	E_{CB}
	C	E_{AE}	E_{BC}	TP_C

2.8.2 Precision

Precision evaluates the number of true positives compared to the sum of all positives, with false positives receiving more attention:

$$\text{precision} = \frac{TP}{TP + FP}. \tag{2.6}$$

2.8.3 Recall

Recall is a metric that evaluates whether the model successfully detects true positives amongst all real positive labels:

$$\text{recall} = \frac{TP}{TP + FN} \tag{2.7}$$

2.8.4 F1 score

The F1 score is defined by the harmonic mean between Precision and Recall and therefore combines both metrics. A low F1 score reflects low Precision or Recall:

$$\text{f1 score} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.8}$$

For multi-class classification problems, precision, recall and F1 score can be calculated by the following types of averaging performed on the data:

- micro: Metrics are calculated globally by counting the sum of true positives, false negatives, and false positives;
- macro: The metrics are calculated for each label and their unweighted average is obtained. This type of average does not take into account the imbalance of the labels;

- weighted: The metrics are calculated for each label and their weighted average is determined by the support (the number of true instances for each label). This changes the 'macro' to account for label imbalance and may result in an F-score that is not between accuracy and recovery.

2.9 Final considerations

This chapter discussed the fundamentals that are required to study and tackle image classification tasks using computer vision techniques for OMR. First, the basic concepts of music notation and strokes were presented once that is the topic covered by this final paper. After that, possibilities for representing music symbol in digital images and shape-based feature extraction techniques were discussed as a traditional approach for the underlying recognition task. Finally, the deep learning was also presented, with focus on CNNs networks and transformer that have largely been employed in the recent days for object recognition in images.

The techniques and models explored in this chapter showed the wide range of possibilities for the recognition of music symbols in digital images when considering the online and offline approaches. As deep learning is constantly receiving novel contributions, this research explores an opportunity of studying transformer for OMR. It is also important to review previous research to understand the progress of OMR studies and identify gaps. Therefore, studies were selected to support this work and are discussed in the following chapter.

Chapter 3

Literature review

A literature review as defined by Siddaway et al. [49], was conducted to discuss a selection of the most relevant and recent studies on the recognition of symbols in OMR and application of transformers in computer vision presented after the year 2010. The following keywords were initially considered for the article search: “omr", “optical music recognition", “freeman chain code", “online music recognition", “offline music recognition", “transformer" and “self-attention", as well as references to previously selected articles. Capes Journal Portal [1] and Google Scholar [2] tools were used to search for articles, and the papers were selected after three filters in sequence, considering the abstract, the introduction along with the results, and the similarity of the proposed method to the method presented in this work. This chapter presents the main selected articles that have helped in the study of this work.

3.1 Recognition of Pen-Based Music Notation: the-HOMUS dataset

This work is one of the pioneers regarding the task of OMR, especially online OMR . Since OMR had been little explored until then and the absence of datasets for comparative experiments, Calvo-Zaragoza and Oncina [24] created an online dataset for music symbols to create a reference corpus. The dataset was created by 100 musicians and contains 32 music symbols distributed in 15200 text files. These files contain one or more lines, each with a single stroke, where each stroke represents a series of 2D points (coordinates of the path traced with a digital pen). The paper also presents some classification techniques for online and offline (images) modes. The considered online techniques include a neural

[1] `https://www-periodicos-capes-gov-br.ezl.periodicos.capes.gov.br/index.php?`
[2] `https://scholar.google.com/`

network, in which two versions of feature vectors have been defined for calculating distances: FFC [40] and DTW [50], and the HMM statistical model, in which a continuous left-to-right topology was used and the models were trained with the Baum-Welch algorithm [51]. Feature extraction was performed as described by Lee in [23] and included the following features: Ink directions (from the pen), spatial relationships between subregions of the symbol divided into a grid, and stroke information defined between pen-down and pen-up sessions. These resources provided good results for online recognition of musical symbols. Offline techniques include k-Nearest Neighbor, where the dissimilarity between two samples was given by the Euclidean distance, ANN [52] with the MLP topology, and Support Vector Machines [53] considering the Radial Basis Function (RBF) and Polynomial kernel function and trained by Sequential Minimum Optimization Algorithm (SMO) [54].

Two types of experiments were conducted: one to evaluate the difficulty when recognizing symbols from an unknown user, in which tests consisted of samples of each musician isolated from the entire data set. The second setting evaluated how classification results are affected when samples of the same musician are found in the training set. The first experiment setting yielded to error rates of more than 15%. For online symbol recognition, DTW had the lowest error rate (15.2%), while SVM with RBF kernel achieved the best performance (26%) among the offline techniques. The second form was able to achieve better error rates and outperformed FFC with an average error of 7%. In both experiment forms, the algorithms that used the online nature of the data had the best performance, while those that explored the offline modality had higher error rates. Therefore, the authors especially encourage the study of OMR in offline mode.

3.2 Pen-Based Music Document Transcription with Convolutional Neural Networks

With data sets for field research, it is possible to propose models for the online system to be used in a future music writing application, as well as for the offline system that recognizes scores that have already been written. Sober-Mira et al. [7] had the idea of using online optical recognition to recognize the musical notation of old handwritten scores, using human interaction to transcribe the musical document. This involves writing the notes with a digital pen over a score displayed on the screen, i.e., copying the handwritten symbols. However, the proposed model can also be used in a scenario where the musician wants to write new compositions. The system receives a multimodal signal: a sequence of coordinates of the path traveled by the pen on the digital surface, and the image of the score below the drawn symbol containing the original drawn symbol. Then, the image

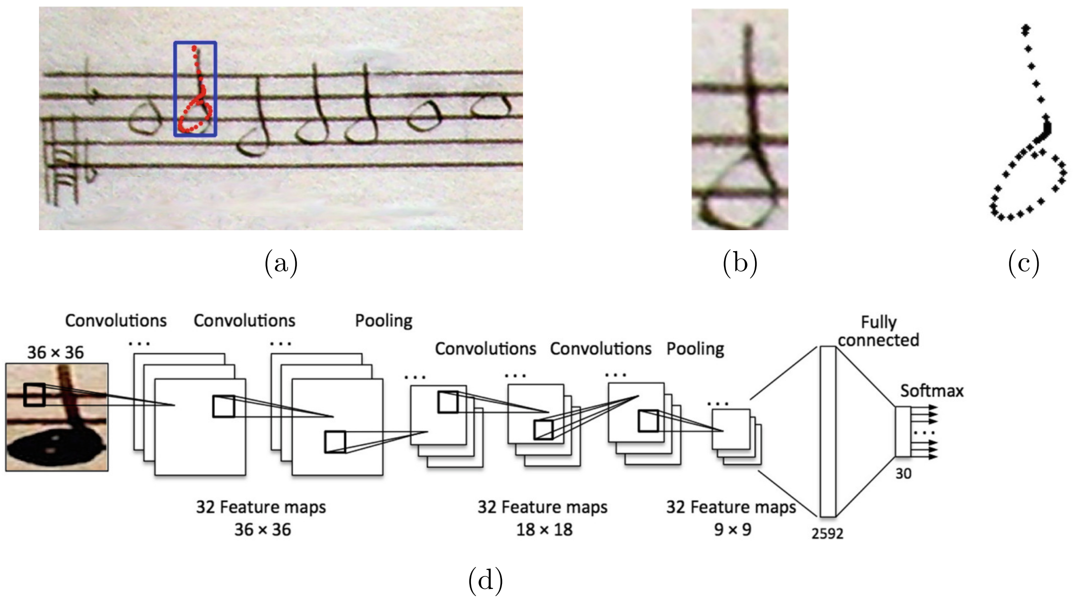


Figure 3.1: Process of image acquisition and CNN architecture used: (a) tracking process. (b) Offline data. (c) Online data. (d) CNN architecture used for both offline and online data [7].

of the drawn symbol is generated in two ways: by rendering the segments between the pairs of consecutive points and by cropping the original image of the score considering the bounding box of the pen strokes. For the classification task, the authors used a CNN with the following configuration:

$$\text{Conv(32, 3)} \rightarrow \text{Conv(32, 3)} \rightarrow \text{MaxPool(2)} \rightarrow \text{Conv(32, 3)} \rightarrow \text{Conv(32, 3)} \rightarrow \text{MaxPool(2)}$$

in which MaxPool is the max pooling operation and Conv(c, k) represents a spatial convolutional layer with kernel size $k \times k$ and number of filters c, with Rectified Linear Unit (ReLU) [55] activation. Figure 3.1 illustrates the process for a single symbol and CNN architecture.

Experiments were performed in a dataset consisting of 60 pages of handwritten documents in white Spanish mensural notation (ca. 16th and 17th centuries), containing 10150 symbols from 30 different classes, using a 5-fold cross-validation scheme. The best results were obtained by combining the two input forms:

- Intermediate fusion: the two images were fed into the input layer of two CNN s and ended up in a single output layer - error $= 3.6 \pm 0.5$;
- Late fusion: where the decisions of each CNN were interpreted in terms of probabilities and combined into a single decision - error $= 3.5 \pm 0.7$.

3.3 Offline music symbol recognition using Daisy feature and quantum Gray wolf optimization based feature selection

Malakar et al. [8] also followed the pen-based strategy for classifying symbols in offline mode. The authors used the descriptor Daisy [56] to extract the features of music symbol images from different databases. Since the extracted feature vector present high dimension, they applied a FS strategy called QGWO [57]. The methodology is shown in Figure 3.2. The following datasets were used to validate the proposed method: HOMUS (offline version only), Capitan score database (Capitan_score_uniform and Capitan_score_non-uniform) and Fornés dataset.

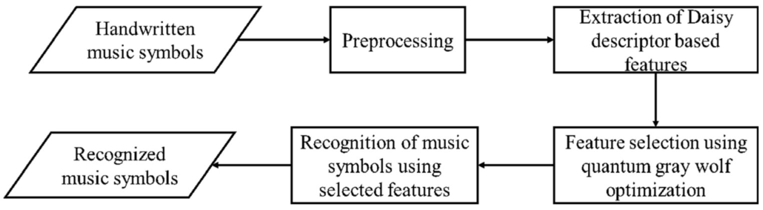

Handwritten music symbols
Preprocessing
Extraction of Daisy descriptor based features
Feature selection using quantum gray wolf optimization
Recognition of music symbols using selected features
Recognized music symbols

Figure 3.2: Block diagram of the proposed model [8].

Five common classifiers were tested on a portion of the HOMUS dataset to select the best performing model and use it for the rest of the experiment: MLP, KNN, Naïve Bayes (NB), Random Forest (RF), and SMO. SMO achieved the best result and was selected as the classifier. In addition, the authors conducted experiments with and without the use of FS, and it became clear that the use of this technique could improve the symbol recognition accuracy of all datasets. The experimental results showed that on average a recognition accuracy of almost 99% was achieved, except in the HOMUS dataset with an average accuracy of 92.64%. The authors also compared their method with some traditional techniques in the research field and the results showed that the proposed method performed better than its predecessors.

3.4 An ensemble of deep transfer learning models for handwritten music symbol recognition

Paul et al. [1] applied three pre-trained deep learning models to different offline datasets of images with music symbols, using SVM as an aggregator. Figure 3.3 illustrates the general process used in their work. First, images of music symbols were collected from five publicly available standard datasets, namely Handwritten Online Music Symbols (HOMUS), Capitan_Score_Uniform, Capitan_Score_Nonuniform, Rebelo_real, and Forne's. Then, three pre-trained models, ResNet50, GoogleNet, and DenseNet161, were fitted to the HOMUS set and the hyperparameters found were applied to the experiments in the remaining data. The confidence score output vectors of each model were linked and used to train the SVM classifier to predict the final class. Thus, the length of the input vector for the SVM classifier is three times the number of classes considered for classification.

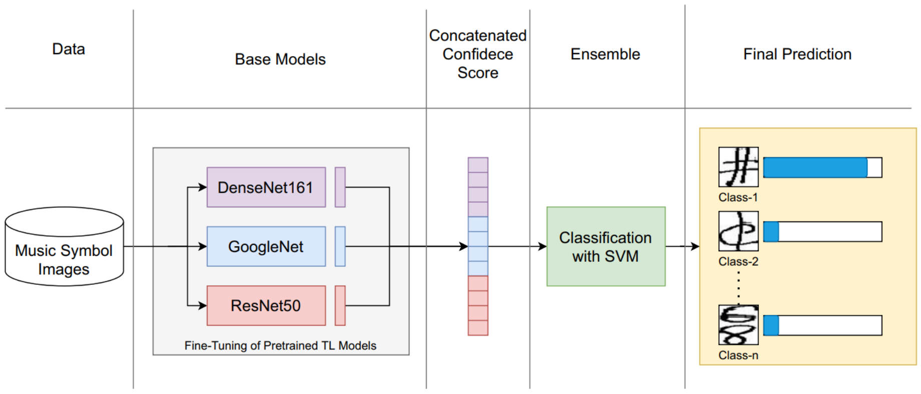


Figure 3.3: A graphical description of the Ashis Paul et al. proposed model [1].

The SVM classifier was trained using a weighted sampling method that assigns a constant weight to samples with multiple unique predictions. Up to now, the authors obtained the best results for all these datasets with an average accuracy of 97.42% for the HOMUS set and over 99% for the other datasets.

3.5 BEiT : BERT Pre-Training of Image Transformers

With the success of the large visual Imagenet dataset, The ImageNet dataset, a very large collection of photographs with human annotations designed by academics, and the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), an annual competition using subsets of the ImageNet dataset, have enabled the development and benchmarking of state-of-the-art computer vision algorithms. In particular, convolutional neural networks have indeed managed to achieve satisfactory results on image comprehension tasks, such as the work described above. However, following the success of attention-based models in natural language processing, many researchers have proposed architectures that insert the idea of the transformer into convolutional networks for visual tasks and achieve promising performance in computer vision [30, 31]. However, to achieve good performance, large datasets are required, even more data than CNN s would demand. Self-supervised pre-training is a promising solution to overcome this problem. Therefore, Bao et al. [9]

presented the model Bidirectional Encoder representation of Image Transformers (BEiT) based on the model BERT.

As shown in Figure 3.4, an input image is divided into a grid of fields (analogous to words in text input data). The original image is also tokenized with latent Variational Autoencoder (VAE) codes [58]. In the pre-training phase, before the transformer is turned on, some areas are randomly masked (gray areas in Figure 3.4). The model then learns to retrieve the visual tokens from the original image through self-supervised learning. Once the pre-training is complete, the image is reconstructed using the predicted tokens and can be applied to downstream tasks by adjusting parameters such as classification, segmentation, or intermediate tuning.

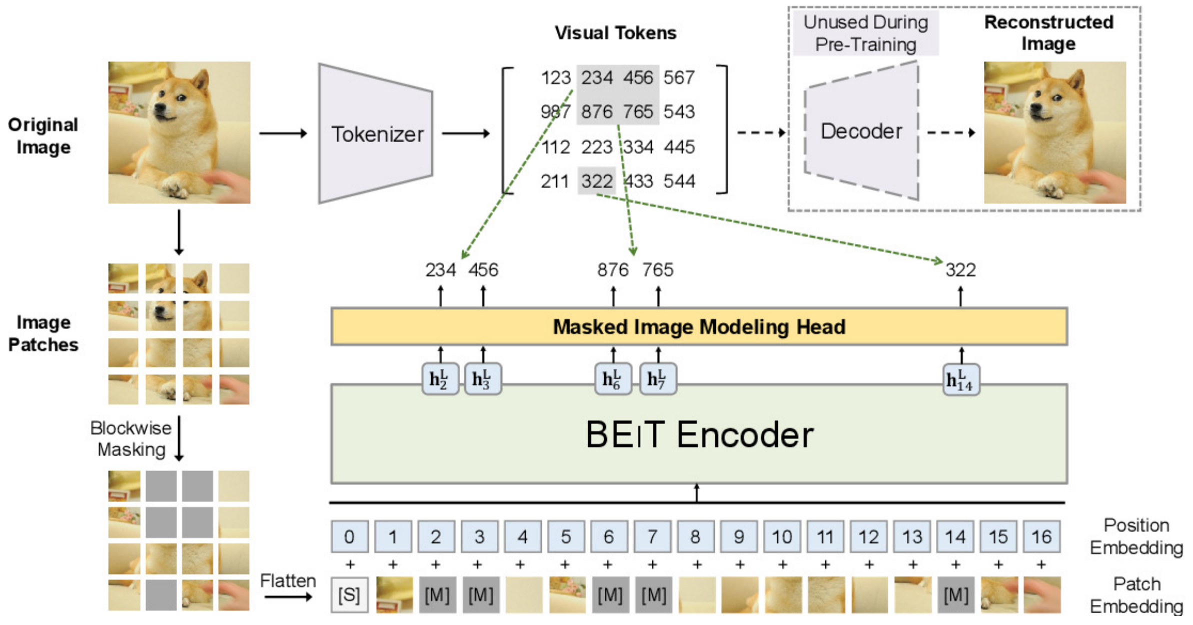


Figure 3.4: Overview of BEiT pre-training [9].

In the image classification task, a simple linear classifier is added to the transformer. The authors employed a pooling layer to aggregate the representations and pass them to a softmax classifier. The pooling layer is used to reduce the input dimension and, consequently, reduce the number of parameters to be learned and the computational cost. The layer includes grouping and a filter applied to the feature maps. The most common grouping methods are average and maximum pooling, in which the average and maximum values are calculated for each field in the resource map, respectively. The result is a summary of the input features that is invariant to small changes in the position of the input features (invariant to local translation). The BEiT classifier uses average pooling. The probabilities of the categories are calculated by Equation 3.1.

$$\mathrm{softmax}(\mathrm{avg}({h_i^L}_{i=1}^{N} W_c)), \tag{3.1}$$

in which h_i^L is the final encoding vector of the i-th image patch, $W_c \in \mathbb{R}^{D \times C}$ is an array of parameters, and C is the number of labels. The probability of labeled data is maximized by updating the BEiT parameters and the softmax classifier.

BEiT achieved an accuracy of 91.8 in the CIFAR-100 dataset and 83.2 in the ImageNet dataset. Thus, BEiT is an efficient transformer model for images, as well as other models, with "*Vision Transformer*" (ViT) porposed by Dosovitskiy et al. [30] and "*Data-Efficient Image Transformers*" (DEiT) porposed by Touvron et al. [31].

ViT is the first model to successfully train a transformer encoder on images, in particular ImageNet, with very good results compared to known convolutional architectures. The model follows a simplified pipeline compared to BEiT, without the use of tokens:

1. splitting an image into patches;
2. flatten the patches;
3. creating low-dimensional linear embeddings;
4. adding position encoding vectors to embeddings;
5. feed the sequence as input to a standard transformer encoder;
6. pre-train the model in a supervised manner;
7. fine-tune the downstream dataset for image classification.

ViT models must be trained on expensive infrastructure to achieve good accuracies. To solve this problem, Touvron et al. has proposed DEiT [31], which requires much less data and much less computational resources compared to ViT . DEiT contains a distillation token that is learned by backpropagation from the pre-trained convolutional RegNety [59] and interacts with data labels and image patch tokens through self-attention layers.

3.6 Final considerations

The literature review showed some researches that provides various online and offline notation datasets. Moreover, some promising studies in OMR employed different computer vision techniques, and most importantly, presented successful applications using transformers. Hence, this scenario is appropriate to explore and study an approach to OMR based on transformer. Since the datasets are not large, the method presented in this work has as one of the objects of study the transformer through three pre-trained models: BEiT, ViT and DEiT, which do not require a large database to obtain satisfactory results.

The details of the method are described in the next chapter.

Chapter 4

Methodology

This work proposes a method for recognizing music notation, more specifically for classifying music symbols from online images (drawings with a pen on a digital surface) and offline images, using a transformer-based classification model. Each step of the proposed method is detailed in the next sections. Section 4.1 describes the image acquisition. Section 4.2 describes the preprocessing of images. Section 4.3 explains the transformation and classification of images. Finally, Section 4.4 describes the evaluation strategy of the proposed method.

The data set passes through a pipeline defined by a sequence of transformations, as shown in the flowchart in Figure 4.1.

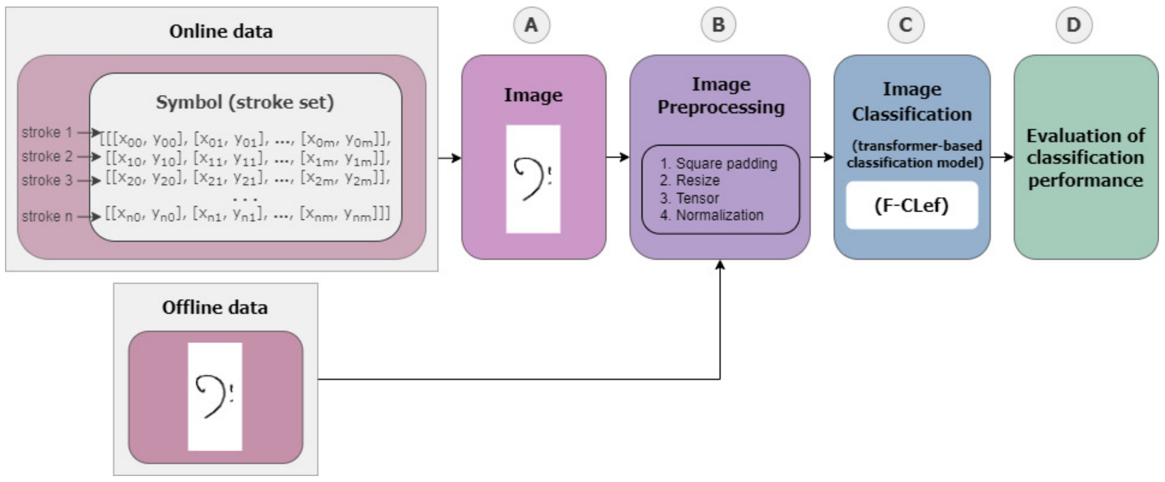
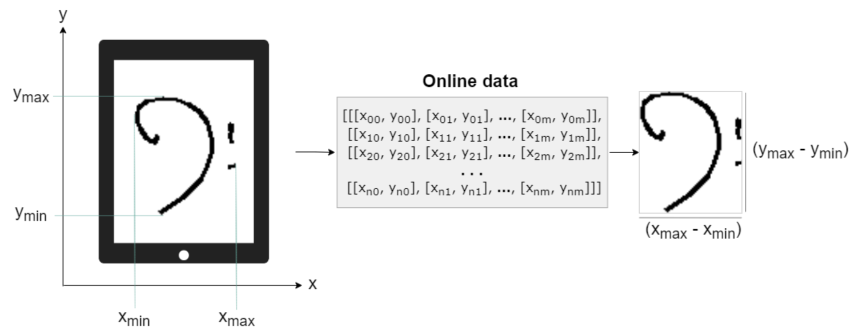

Online data
Symbol (stroke set)
stroke 1
stroke 2
stroke 3
stroke n
[[[x00, y00], [x01, y01], ..., [x0m, y0m]],
[[x10, y10], [x11, y11], ..., [x1m, y1m]],
[[x20, y20], [x21, y21], ..., [x2m, y2m]],
...
[[xn0, yn0], [xn1, yn1], ..., [xnm, ynm]]]
Offline data
A
Image
B
Image Preprocessing
1. Square padding
2. Resize
3. Tensor
4. Normalization
C
Image Classification
(transformer-based classification model)
(F-CLef)
D
Evaluation of classification performance

Figure 4.1: Flowchart describing the proposed method.

4.1 Step A: Obtaining the images of online data

In a real situation, the musician draws his composition on a digital surface (tablet) over time. A system can capture the coordinates of the pen's contact with the surface at a certain frequency, and in this way group the coordinates into tracks, which in turn are grouped into different musical symbols, depending on the capture rate. Thus, the proposed method receives as input a sequence of sequences of coordinate pairs, in which a novel image is created regarding the corresponding symbol.

Converting online data to offline data consists of creating an image of the input symbol by drawing a line between each coordinate pair in turn on a “blank screen”. This is done for all pairs of coordinates so that the final symbol is drawn. The final image is defined by the bounding box around the symbol, according to the maximum and minimum points of each dimension of the coordinates (x_{max}, x_{min}, y_{max} and y_{min}), as show in Figure 4.2.



y
y_{max}
y_{min}
x
x_{min}
x_{max}
Online data
[[[x_{00}, y_{00}], [x_{01}, y_{01}], ..., [x_{0m}, y_{0m}]],
[[x_{10}, y_{10}], [x_{11}, y_{11}], ..., [x_{1m}, y_{1m}]],
[[x_{20}, y_{20}], [x_{21}, y_{21}], ..., [x_{2m}, y_{2m}]],
...
[[x_{n0}, y_{n0}], [x_{n1}, y_{n1}], ..., [x_{nm}, y_{nm}]]]
(y_{max} - y_{min})
(x_{max} - x_{min})

Figure 4.2: Creating the symbol image without pad.

4.2 Step B: Image Preprocessing

This phase involves the transformation of images, both those generated from online data in step A and those from offline datasets that already have this format. As discussed in Section 2.3, it is desirable that the model presents invariance to rotation, translation and scaling. As explained earlier, in musical notation the symbols are written on top of the staff, and without this the symbols have no meaning. Therefore, writing follows well-defined rules, i.e., there is no possibility of having images of rotated symbols, and thus the proposed method does not need to be present invariance to rotation.

Each transformer model, as mentioned earlier, has been pre-trained on images with specific sizes. Thus, the images must be resized to that size in order to serve as input to these models. However, when the images are resized, the symbols lose their structure, such as the height and width properties. Since the recognition task is sensitive to the shape of the symbol in the image, a fill is added to the image so that it becomes square, and then the size is changed. Thus the proportions of the objects are preserved. The resize process is performed by interpolation [60], which is a function for finding new data points based on the range of a discrete set of known data points. The resize value is defined according to the size of the images used in the pre-training of the transformers models. In this work, the nearest neighbor interpolation algorithm [61] was used, which selects the value of the nearest point and disregards the values of neighboring points. This approach was chosen because it is best suited for image classification tasks, since the pixel values that are used as input to the function remain exactly the same.

Another common and important step to consider in image classification tasks is to define the image as a tensor, i.e., an n-dimensional matrix that can be run on the GPU, and normalize it, since this guarantees that each pixel has a distribution of similar data. This speeds up convergence when training the classifier. An image is normalized to mean and standard deviation [62] as in Equation 4.1.

$$\text{normalized[channel]} = (\text{input[channel]} - \text{mean[channel]})/\text{std[channel]} \tag{4.1}$$

4.3 Step C: Image Classification

Three pre-trained transformer models were selected from a large external database: ViT [30], BEiT [9] and DEiT [31]. The transformer architecture contains as its final layer a simple linear classifier configured according to the number of labels (symbol types) of the input data to categorize images, i.e., to identify the music symbol corresponding to the image. As there are no large music notation datasets in the online approach, the method of this work considers transfer learning by taking into account the reasons described in Section 2.6.

4.4 Step D: Evaluation of the classification performance

To verify that the model can correctly classify the unknown images, the accuracy, recognition, and F1 score are analyzed 2.8. Furthermore, to get an overview of the recognition of the symbols, the confusion matrix and the graphics depicting the accuracy values of each class are analyzed. To test the proposed method, experiments were performed on different image sets with music notation and will be evaluated in the next chapter.

4.5 Final considerations

This chapter discussed the steps that constitute the proposed method. These include the generation of images from online data, preprocessing and classification of the images. Finally, the next step is concerned to the evaluation of the proposed method in order to evaluate the classification performances on different image sets of handwritten musical notes were employed. The experiments and the discussion of the results are presented in the next chapter.

Chapter 5

Experimental results

This chapter describes the experiments to validate the proposed method, which were conducted using different data sets. The source code for this work is available in a repository on GitHub [1].

5.1 Data sets

The experiments considered six data sets. Two sets of online data: HOMUS [24] [2] and SNU [25] [3], and four data sets: Capitan_Score_Uniform and Capitan_Score_Non-uniform [63] [4], Fornés [13] [5] and Rebelo_real [64] [6], which are discussed in more detail below.

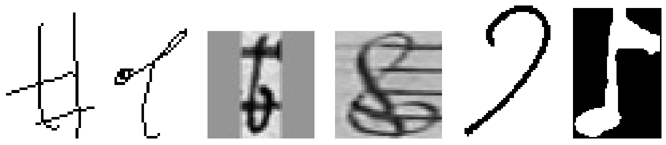


Figure 5.1: Symbols examples of HOMUS (*Sharp*), SNU (*Eighth-Rest*), Capitan_Score_Uniform (*flat*), Capitan_Score_Non-uniform *(g-clef)*, Fornés (*CLEF_Bass*) and Rebelo_real (*notesFlags*) datasets, respectively.

5.1.1 Handwritten Online Music Symbols dataset

HOMUS has 15200 samples with 32 different symbols and was created by 100 musicians with much, little or no experience in musical composition from the Escuela de Educandos Asociación Musical l'Avanç music schools (El Campello, Spain) and the Superior Music

[1] https://github.com/hevslc/OMR-with-Transformers
[2] https://grfia.dlsi.ua.es/homus/
[3] http://mipal.snu.ac.kr/index.php/SNU_Dataset_for_Online_Music_Symbol_Recognition
[4] http://grfia.dlsi.ua.es/musicdocs/Capitan.zip
[5] http://www.cvc.uab.es/~afornes/
[6] https://github.com/apacha/OMR-Datasets

Conservatory of Murcia "Manuel Massotti Littell" (Murcia, Spain). The symbols were written according to the writing style of each musician, with the eighth notes, sixteenth notes, thirty-second notes, and sixty-fourth notes written twice: straight and inverted. Writing was performed on a Samsung Galaxy Note 10.1 device with a resolution of 1280×800 (149 ppi), a sampling rate of 16 ms (60 fps), and using the S Pen stylus. The Table 5.1 shows the symbols included in this dataset. Figure 5.2 presents the distribution of labels.

Table 5.1: Symbols from the HOMUS dataset.

Note	whole, half, quarter, eithght, sixteenth, thirty-second, sixty-fourth
Rest	whole/half, quarter, eithght, sixteenth, thirty-second, sixty-fourth
Accidentals	flat, sharp, natural, double sharp
Time signatures	common time, cut time, 4-4, 2-2, 2-4, 3-4, 3-8, 6,8, 9-8, 12-8
Clef	G-clef, C-clef, F-clef
Others	dot, barline

The symbols are organized in text files distributed in independent directories for each musician. Each text file contains:

- Example label;
- One or more lines representing a dash;
- Each dash contains a list of 2D points, where dashes are separated by a semicolon and the dimensions of a point are separated by a comma.

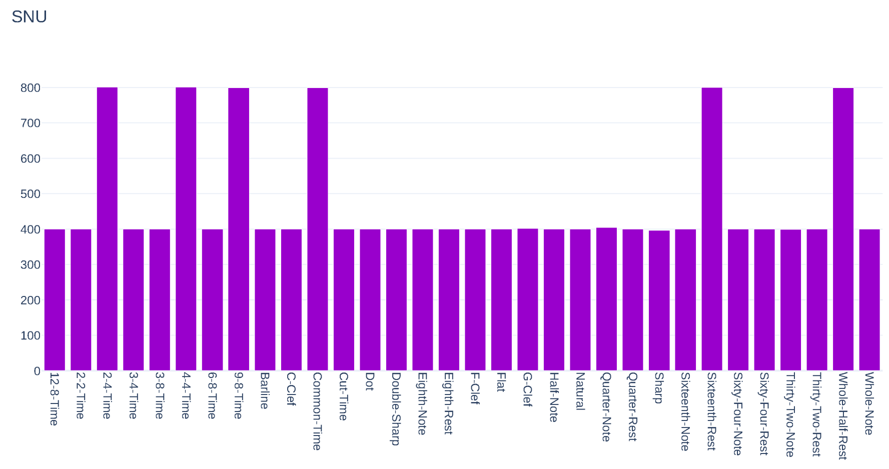


Figure 5.2: Distribution of labels in the HOMUS dataset.

5.1.2 Seoul National University Dataset for Online Music Symbol Recognition

The SNU dataset, created by Oh et al. [25], similarly to the HOMUS dataset (and considered by the authors to be a subset of it), contains 1716 samples with 16 different symbols written by 18 musicians. The images were collected using a digital device and a stylus pen. Table 5.2 shows the types of symbols the set contains, and the organization of the data is the same as in the HOMUS set. Figure 5.3 presents the distribution of labels.

Table 5.2: Símbolos do conjunto de dados SNU .

Note	whole, half, quarter, eighth, sixteenth
Rest	whole/half, quarter, eighth, sixteenth
Accidentals	flat, sharp, natural
Time signatures	common time, cut time, 4-4, 2-2, 2-4, 3-4, 3-8, 6,8, 9-8, 12-8
Clef	G-clef, F-clef
Others	dot, barline

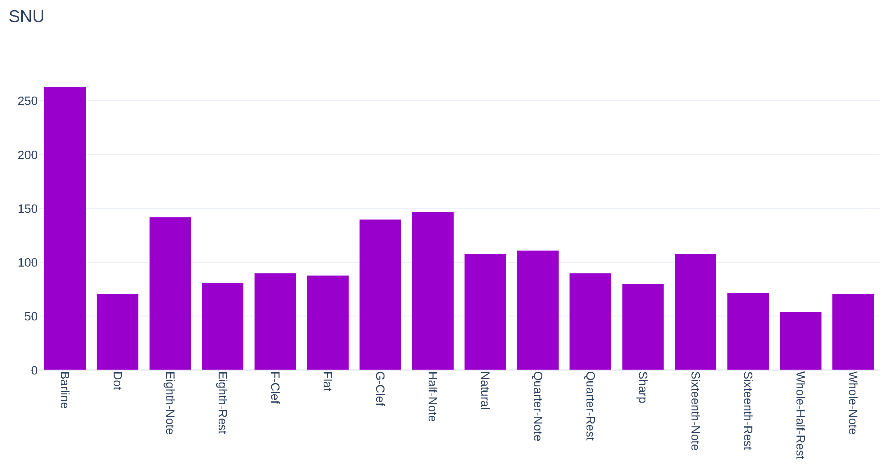


Figure 5.3: Distribution of labels in the SNU dataset.

5.1.3 Capitan dataset

The construction of this dataset was coordinated by Calvo-Zaragoza et al. [63], where 10230 musical symbols, represented in Table 5.3, were found in old manuscripts from the 16th to 18th centuries [65]. The image set contains information about both the stroke captured by the pen on a digital surface and the writing field below the stroke itself. Only online data were used in this work. This set contains two types of samples with the names proposed by Malakar et al. [8]: "Capitan_Score_Uniform" has a uniform boundary outside the music symbol content and “Capitan_Score_Non-uniform” presents non-uniform boundaries. Figure 5.4 presents the distribution of labels for both types of samples.

Table 5.3: Symbols of Bimodal music symbols from Early notation dataset.

Note	(coloured) brevis, longa, (coloured) minima (whole/half in HOMUS), (coloured) semibrevis (whole in HOMUS), (coloured) semiminima (eighth in HOMUS)
Rest	brevis, longa, whole/minima (whole/half in HOMUS), seminima (eighth in HOMUS)
Accidentals	flat, sharp
Time signatures	common time, cut time
Clef	C-clef, F-clef 1, F-clef 2, G-clef
Others	dot, barline, double barline, custos, fermata, beam, proportio minor/maior

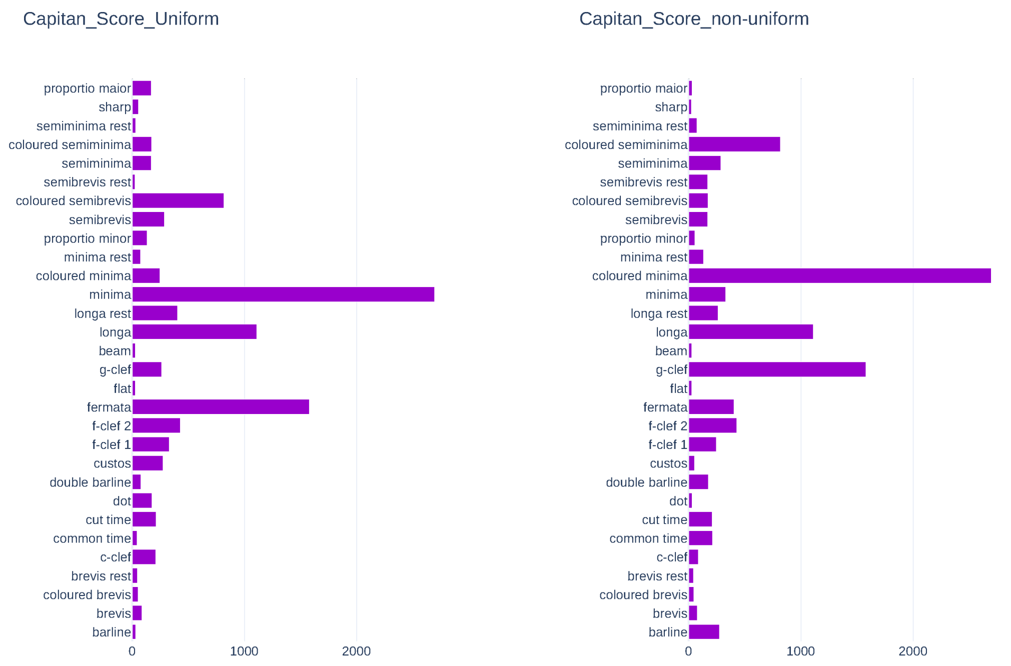


Figure 5.4: Distribution of labels in the Capitan dataset.

5.1.4 Fornés dataset

The construction of this image set was coordinated by Fornés et al. [13], where 2128 clefs and 1970 accidentals, represented in the Table 5.4, were found in modern and old manuscripts from the from the 19th century. The The symbols have been written by 24 different Portuguese musicians. Figure 5.5 presents the distribution of labels for both types of samples.

Table 5.4: Symbols of Fornés dataset.

ACCIDENTAL_Sharp
CLEF_Alto
CLEF_Bass
CLEF_Trebble

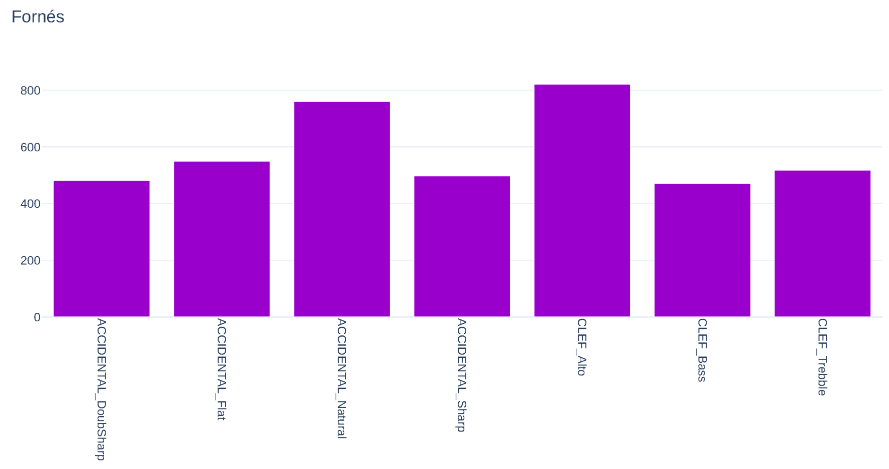


Figure 5.5: Distribution of labels in the Fornés dataset.

5.1.5 Rebelo_real dataset

This image set was provided by Rebelo et al. [64] and contains about 3500 of 14 different handwritten musical symbols 5.5 from early music scores, probably written by 5 different Portuguese musicians between the 17th and 19th centuries. Figure 5.6 presents the distribution of labels for both types of samples.

Table 5.5: Symbols of Rebelo_real dataset.

accent	bassClef
beams	flat
naturals	notes
notesFlags	notesOpen
rests1	rests2
staccatissimo	trebleClef
sharps	unknown

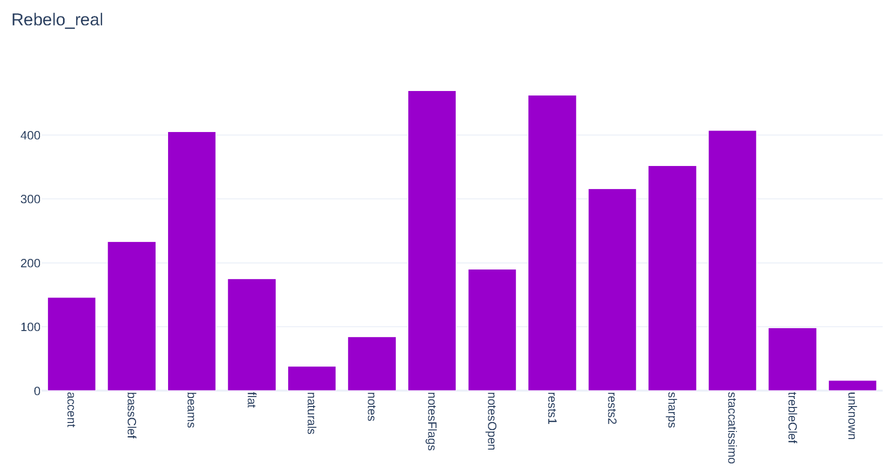


Figure 5.6: Distribution of labels in the Rebelo_real dataset.

5.2 Experimental setup

To perform the experiments, the development environment and data sets were configured as follows:

- The experiments were performed in a Linux environment and with an Intel(R) Xeon(R) Gold 5220R CPU @ 2.20GHz.
- The method was coded in Python 3.8 using the following main libraries for support:

  image preprocessing: torchvision [7],

  transformer: huggingface to obtain the transformer model, training model and hyperparameter optimization model [8];
- The BEiT model was pre-trained in a self-supervised manner on the large ImageNet 22k image collection (14 million images, 21,841 labels) in 224 × 224 resolution and fitted in a supervised manner on the ImageNet subset ILSVRC2012 [66] (1 million images, 1000 labels), also presenting 224 × 224 resolution.
- The ViT model was pre-trained on ImageNet-21k (14 million images, 21,843 labels) at resolution 224 × 224, and fine-tuned on ImageNet 2012 (1 million images, 1,000 labels) at resolution 224 × 224.
- The DEiT model was pre-trained and fine-tuned on ImageNet-1k (1 million images, 1,000 labels) at resolution 224 × 224.

[7] `https://pytorch.org/vision/stable/transforms.html`
[8] `https://huggingface.co/`

- All data sets used in this paper were divided according to the holdout method and the Pareto principle (also known as the 80/20 rule), with 80% for the training set and 10% for the validation and test sets. The split returns stratified data, i.e, each set contains approximately the same percentage of samples of each target class as the complete set.

Setting up architectures for machine learning models requires careful tuning. For complex architectures, manual configurations or brute force methods are not feasible, as such processes can be extremely time consuming. Fortunately, automatic hyperparameter optimization methods exist today, such as Async Successive Halving Algorithm (ASHA) [67], Population Based Training (PBT) [68], Bayesian Optimization and Hyperband (BOHB) [21] and and BlendSearch [69], some of which are specifically designed for parallel computing.

Therefore, in this work, the hyperparameters of the models were optimized by parallel computations using Tree-Structured Parzen Estimator (TPE) [70]. TPE worked with the ASHA [67] scheduler based on the original HyperBand [71] scheduler, which can terminate bad tests early, pause tests, clone tests, and change hyperparameters of a running test. Thirty tests runs were defined, each running on 1 CPU and 1 GPU. Optimization took place via the BEiT transformer in a subset of HOMUS (half of the data set). The hyperparameter values found are presented in Table 5.6 and were replicated for all transformer models and all data sets.

Table 5.6: Hyperparameter values obtained after the optimization process on the validation set.

learning rate	train batch size	eval batch size	train epochs
$2.428e-05$	4	4	3

5.3 Quantitative assessment

Table 5.7 describes the performances in terms of accuracy, recall and F1 score 2.8. The weighted average of the scores of each class for these metrics was considered. Results of over 98% were obtained. It can be concluded that the proposed method performs well in all metrics and datasets. DEiT showed the best performance in most cases, which could be due to the fact that it requires much less data and much less computational resources to build a powerful image classification model.

In Table 5.9, the original results of this work were compared with those obtained by recent and best-performing works in literature, such as the researches by Oh et al.

Table 5.7: Summary of the results of the recognition of musical symbols of the datasets with different transformer models.

Dataset	Classifier (transformer)	Precision (in %)	Recall (in %)	F1 score (in %)
HOMUS	BEiT	98.66	98.66	98.66
	ViT	98.95	98.95	98.95
	DEiT	**99.13**	**99.12**	**99.12**
SNU	BEiT	99.67	99.65	99.65
	ViT	99.54	99.53	99.53
	DEiT	**100.00**	**100.00**	**100.00**
Capitan_Score_Non-uniform	BEiT	99.75	99.75	99.75
	ViT	99.73	99.73	99.73
	DEiT	**99.86**	**99.86**	**99.86**
Capitan_Score_Uniform	BEiT	99.75	99.75	99.75
	ViT	99.71	99.71	99.71
	DEiT	**99.86**	**99.86**	**99.86**
Fornés	BEiT	**100.00**	**100.00**	**100.00**
	ViT	99.95	99.95	99.95
	DEiT	98.95	99.95	99.95
Rebelo_real	BEiT	98.79	98.76	98.76
	ViT	98.89	98.88	98.87
	DEiT	**99.30**	**99.29**	**99.29**

[25] on the SNU dataset, and by Paul et al. [1] on the other datasets. Regardless of differences in methodology, as show in Table 5.8, the method used in this work, manages to achieve state-of-the-art results for all datasets evaluated, except for the Rebelo_real and the Fornés sets, in which there is a small difference of points for the latter.

Table 5.8: Training configuration of each method compared.

Method	Training
OMR with Transformer	80-10-10 batch=4
Ensemble [1]	5-fold cross-validation validation: 15% of the train samples
Oh's et al. [25]	10-fold crossvalidation

Table 5.9: Comparison with methods of the most recent works.

Dataset	Method	F1 score (in %)
HOMUS	Ensemble [1]	97.48
	OMR with Transformer (DEiT)	**99.12**
SNU	Oh's et al. [25]	93.65
	OMR with Transformer (DEiT)	**100.00**
Capitan_Score_Non-uniform	Ensemble [1]	99.64
	OMR with Transformer (DEiT)	**99.86**
Capitan_Score_Uniform	Ensemble [1]	98.90
	OMR with Transformer (DEiT)	**99.86**
Fornés	Ensemble [1]	**100.00**
	OMR with Transformer (DEiT)	98.95
Rebelo_real	Ensemble [1]	**99.56**
	OMR with Transformer (DEiT)	99.29

5.4 Visual assessment

From the diagram in Figure 5.7, it can be seen that the lowest scores are related to the *Eighth-Rest*, *Sixteenth-Rest* and *Natural* symbols. This is due to the fact that they are similar symbols and can be misclassified among them, as shown by the confusion matrix in Figure 5.8. The example of these symbols, depicted in Figure 5.9, shows the similarity between *Thirty-Two-Note* and *Sixty-Four-Note* or between *Thirty-Two-Rest* and *Sixty-Four-Rest*.

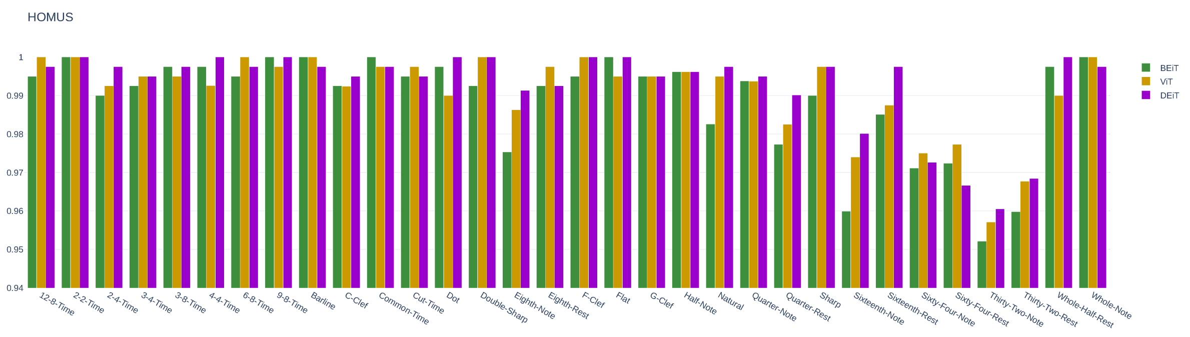


Figure 5.7: F1 score for HOMUS symbols in each transformer model.

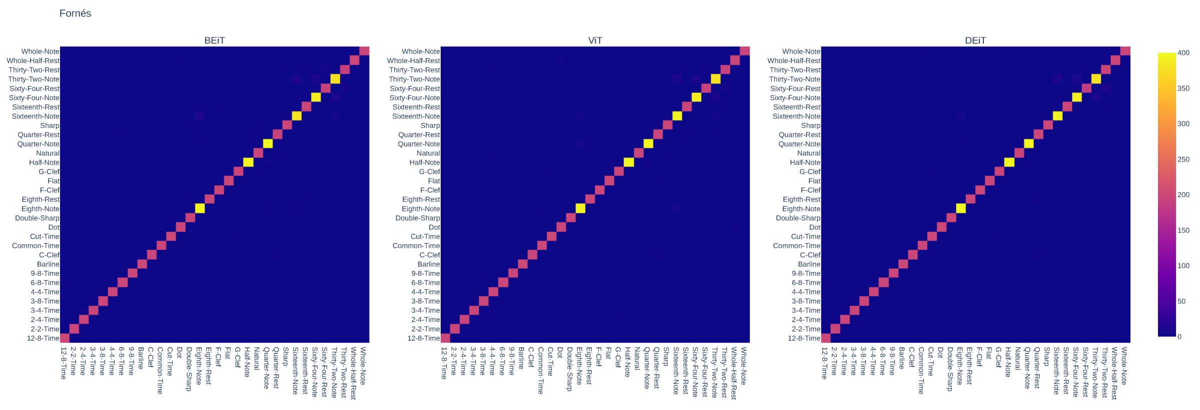


Figure 5.8: Confusion matrix for HOMUS symbols in each transformer model.

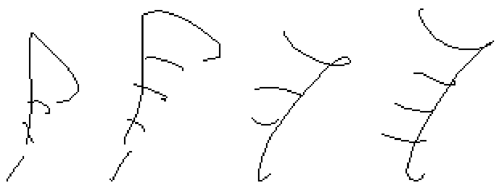


Figure 5.9: Symbols from the HOMUS dataset: *Thirty-Two-Note*, *Sixty-Four-Note*, *Thirty-Two-Rest* and *Sixty-Four-Rest*, respectively.

Figure 5.10 illustrates that the lowest scores are related to the labels *Eighth-Rest*, *Sixteenth-Rest* and *Natural*. As in the HOMUS set, the similarity of the symbols, as shown in Figure 5.12, can lead to misclassification. In Figure 5.12, both symbols consist of a vertical bar and one or two horizontal bars to the left of it.

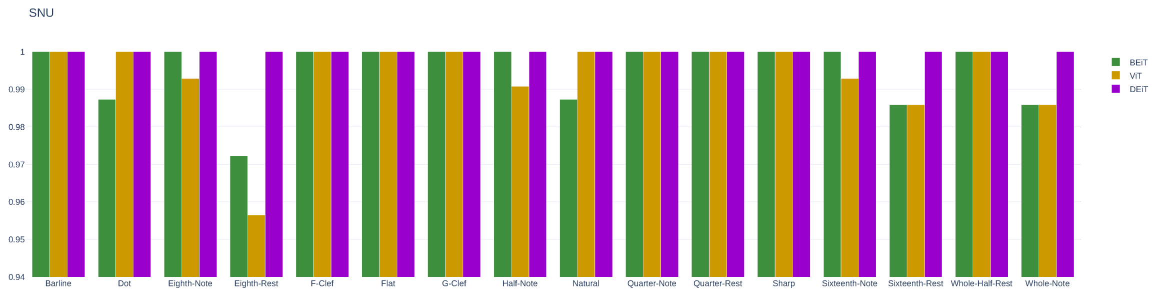


Figure 5.10: F1 score for SNU symbols in each transformer model.

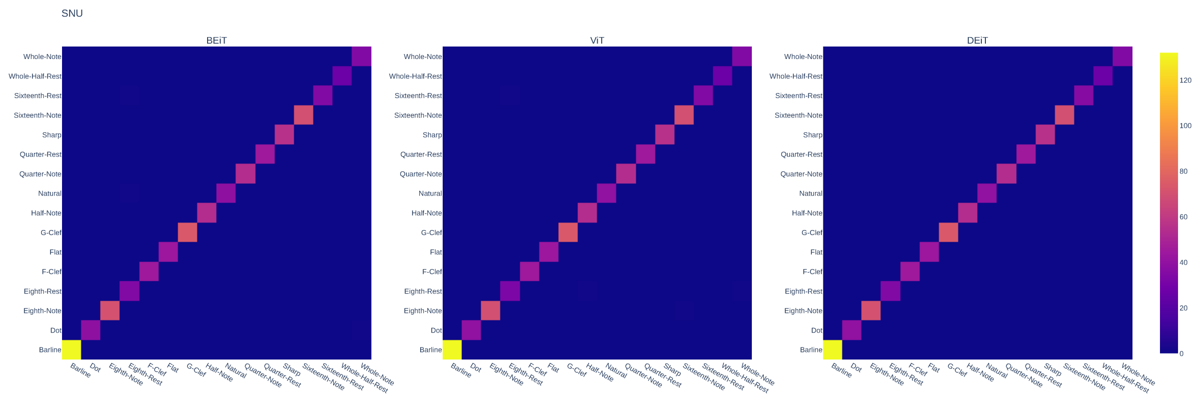


Figure 5.11: Confusion matrix for SNU symbols in each transformer model.

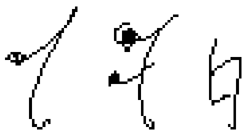


Figure 5.12: Symbols from the SNU dataset: *Eighth-Rest*, *Sixteenth-Rest*, and *Natural*, respectively.

For the other data sets, it is not possible to conclude from the confusion matrices, in Figures 5.11, 5.14, 5.15, 5.16 and 5.17, that the lower scores are due to the similarity of some symbols. Therefore, it is possible that the symbols with the lowest scores have an extremely minimalist design with simple slashes or circles that can be confused with other symbols, as show in Figure 5.13.

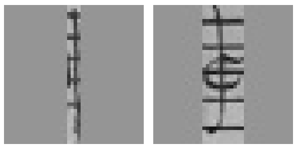


Figure 5.13: *Barline* on the left and *Cut time* on the right from Capitan_Score_Non-uniform set.

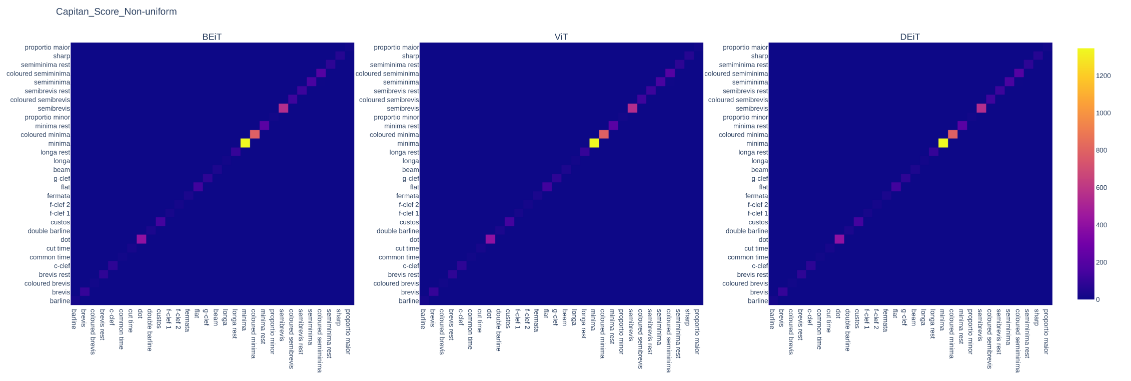


Figure 5.14: Confusion matrix for Capitan_Score_Non-uniform set in each transformer model.

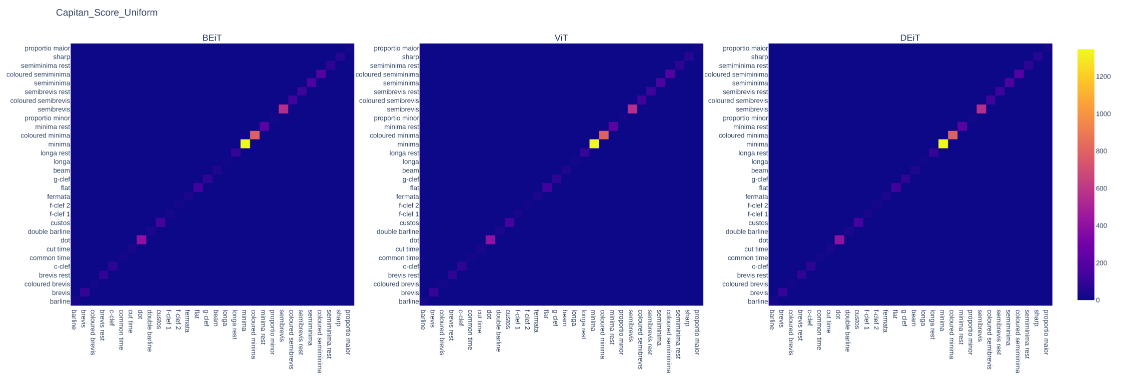


Figure 5.15: Confusion matrix for Capitan_Score_Uniform symbols set in each transformer model.

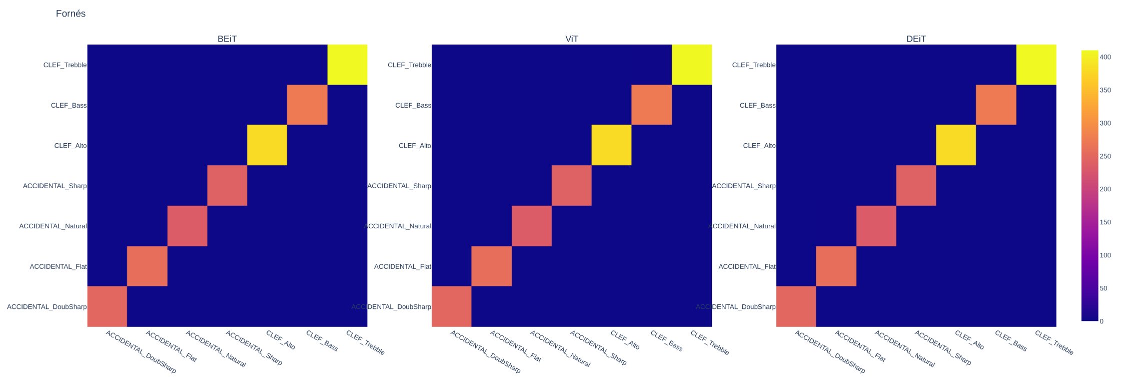


Figure 5.16: Confusion matrix for Fornés set in each transformer model.

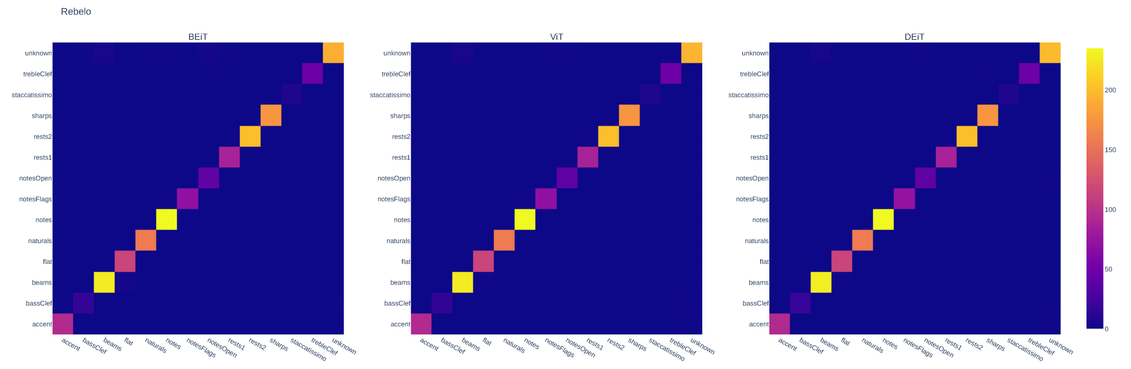


Figure 5.17: Confusion matrix for Rebelo_real set in each transformer model.

The Figures 5.19, 5.18, 5.20 and 5.21 present the scores of each symbol class for the data sets, respectively. Note that DEiT continues to perform best for most symbols. This could be due to the fact that DEiT uses a distillation token. Distillation is the process by which one neural network (the student) learns from the output of another network (the teacher). In DEiT, the distillation token is a learned vector that flows through the network along with the transformed image data.

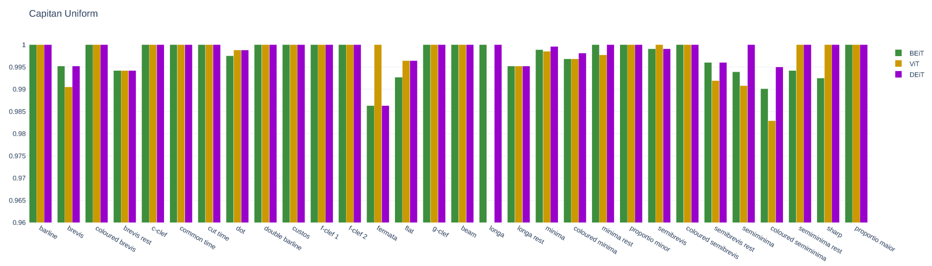


Figure 5.18: F1 score for Capitan_Score_Uniform symbols set in each transformer model.

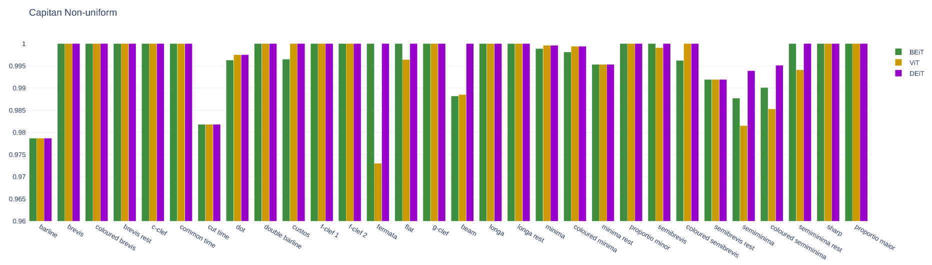


Figure 5.19: F1 score for Capitan_Score_Non-uniform symbols set in each transformer model.

The factors contributing to the higher initial scores on the image labels can be properly analyzed according to the attention visualization technique proposed by Chefer et al. [72]. In this technique, the attention matrices of a layer are multiplied by the gradient of the

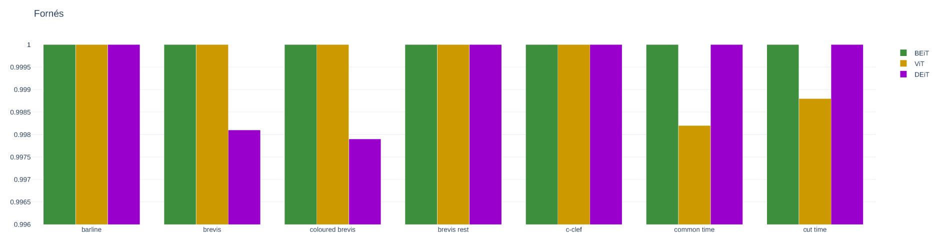


Figure 5.20: F1 score for Fornés set in each transformer model.

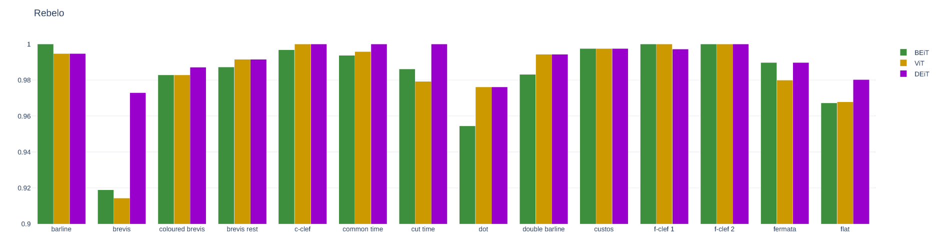


Figure 5.21: F1 score for Rebelo_real set in each transformer model.

target class. Since each transformation layer has multiple heads and each head focuses at different image regions, the information from the heads was combined by computing the average between them. As a result, the output image enhances only the main attentions that contribute to a higher final precision. Figure 5.22 shows the contributions of the attentions of the last layer of the DEiT transformer for the Sixty-Four-Note and Quarter-Note labels of the HOMUS set after training. It is shown that the structure of the attention regions actually corresponds to the structure of the symbol in question, i.e., the model pays attention to the regions that follow the format of the class symbol.

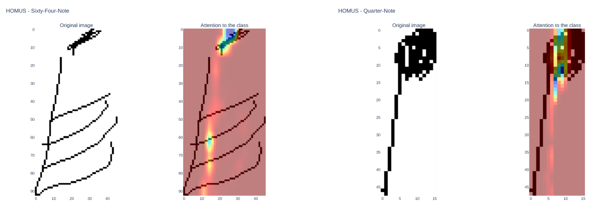


Figure 5.22: Contributions of the attentions of the last layer of DEiT transformer for the *Sixty-Four-Note* class to the left and *Quarter-Note* class on the right, after training. Brighter regions mean greater attention.

5.5 Final considerations

This chapter presented the implementation of the experiments with the sets HOMUS, SNU, Capitan_Uniform, Capitan_Non-uniform, Fórnes, and Rebelo using the three pre-trained transformer models BEiT, ViT, and DEiT. The content of each set, the quantitative results, including classifier estimates and comparison with recent research techniques, and the visual results, including graphs showing the accuracy of each label and the resulting confusion matrix for each classifier, were presented. In addition, some visual results of the attention maps of the DEiT model were presented.

The pre-trained transformers were able to classify the symbols satisfactorily, with accuracy values above 98%, even for data sets classified as small. The lower values may be due to either the similarity of the symbols or the diversity of the writing style. Nevertheless, most accuracy scores exceeded the state of the art, which includes CNNs such as Paul's approach et al. [1]. The proposed method was promising and offers opportunities for improvement. Some suggestions for future work are recommended in the following chapter.

Chapter 6

Conclusion

A method for recognition of online and offline handwritten musical symbols was proposed using a transformer model with transfer learning as a feature extractor and classifier. The method was evaluated on six data sets: HOMUS and SNU (online), Capitan_Score_Non-uniform, Capitan_Score_Uniform, Fornés and Rebelo_real (offline), using three pre-trained transformers, BEiT, ViT, and DEiT.

First, the images were preprocessed by padding them to avoid losing the original structure of the symbol, resizing them, transforming them into tensors, and normalizing them. To obtain the best configuration of the trainings, the hyperparameters were optimized by parallel computations with the TPE and with the ASHA scheduler.

The models showed a satisfactory classification performance, outperforming state-of-the-art methods for most datasets. In particular, for the HOMUS dataset, which has higher complexity due to the high structural similarity of the symbols compared to the other datasets, the transformers achieved a significant improvement in classifications compared to other work. For example, DEiT achieved 99.12% of the F1 score, exceeding the 97.48% score of the ensemble method compared in this work. In general, the proposed method provided results above 98%, with the model using the DEiT architecture showing the best performance in most cases. Therefore, it is concluded that transformers have a great contribution in the OMR field.

The proposed method can be combined with other approaches that include other subsets of OMR, such as online acquisition of music symbols, segmentation of symbols in images of handwritten scores, and reconstruction of MIDI files to create a complete OMR system. Therefore, the method proposed here is promising as it makes a strong contribution to OMR and offers opportunities for improvement.

6.1 Future work

The promising results in this research provides some possibilities for further steps on the OMR research field. The internal parameters of the transformers have not been optimized, so this task will be extended to a future work. In addition, the execution times can also be studied and improved, especially in terms of generating images of the online data. A more detailed study of the visualizations of the attentions in relation to the precisions can be done as well. An investigation of the proposed method can also be performed by merging the databases whose characteristics of the images, such as background color and symbol color, are similar, and performing an intersection of the symbol types present in each set.

In addition, using the contribution of the ensemble method of Paul et al. [1], a combination of two or more transformers can be studied as a method to improve the in this work. Or the last classification layer of the transformer architecture can be improved to increase the classification accuracy.

References

[1] Paul, Ashis, Rishav Pramanik, Samir Malakar, and Ram Sarkar: *An ensemble of deep transfer learning models for handwritten music symbol recognition.* Neural Computing and Applications, pages 1–19, 2021. v, vi, ix, 25, 26, 40, 41, 47, 49

[2] Lobl, Phyl: *Not just noise: Plainspeak sessions, music teaching program, session 3: Top figure of the time signatures*, 2019. `https://phyllobl.net/education/plainspeak-sessions/part2-session3/`, Accessed on May 17, 2022. ix, 8

[3] Agu, Emmanuel: *Digital image processing (cs/ece 545) lecture 9: Color images (part 2) & introduction to spectral techniques (fourier transform, dft, dct).* `https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture09.pdf`, Accessed on May 17, 2022. ix, 10

[4] Bradski, Gary and Adrian Kaehler: *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008. ix, 11

[5] Roman, Victor: *Convolutional neural networks in practice: Develop and implement your first cnn with keras!*, 2020. `https://towardsdatascience.com/convolutional-neural-networks-in-practice-406426c6c19a`, Accessed on May 17, 2022. ix, 15

[6] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin: *Attention is all you need.* Advances in neural information processing systems, 30, 2017. ix, 5, 16, 17, 18

[7] Sober-Mira, Javier, Jorge Calvo-Zaragoza, David Rizo, and José M Iñesta: *Pen-based music document transcription with convolutional neural networks.* In *International Workshop on Graphics Recognition*, pages 71–80. Springer, 2017. ix, 5, 23, 24

[8] Malakar, Samir, Manosij Ghosh, Agneet Chaterjee, Showmik Bhowmik, and Ram Sarkar: *Offline music symbol recognition using daisy feature and quantum grey wolf optimization based feature selection.* Multimedia Tools and Applications, 79(43):32011–32036, 2020. ix, 4, 25, 35

[9] Bao, Hangbo, Li Dong, and Furu Wei: *Beit: Bert pre-training of image transformers.* arXiv preprint arXiv:2106.08254, 2021. ix, 6, 26, 27, 31

[10] Calvo-Zaragoza, Jorge, Jan Hajič Jr, and Alexander Pacha: *Understanding optical music recognition.* ACM Computing Surveys (CSUR), 53(4):1–35, 2020. 3

[11] Blostein, Dorothea and Henry S Baird: *A critical survey of music image analysis.* In *Structured document image analysis*, pages 405–434. Springer, 1992. 3

[12] Rebelo, Ana, Ichiro Fujinaga, Filipe Paszkiewicz, Andre RS Marcal, Carlos Guedes, and Jaime S Cardoso: *Optical music recognition: state-of-the-art and open issues.* International Journal of Multimedia Information Retrieval, 1(3):173–190, 2012. 3

[13] Fornés, Alicia, Josep Lladós, and Gemma Sánchez: *Old handwritten musical symbol classification by a dynamic time warping based method.* In *International Workshop on Graphics Recognition*, pages 51–60. Springer, 2007. 4, 33, 36

[14] Nawade, Savitri Apparao, Mallikarjun Hangarge, Chitra Dhawale, Mamun Bin Ibne Reaz, Rajmohan Pardeshi, and Norhana Arsad: *Old handwritten music symbol recognition using directional multi-resolution spatial features.* In *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, pages 1–4. IEEE, 2018. 4

[15] Nawade, Savitri Apparao, Rajmohan Pardeshi, Chitra Dhawale, and Mallikarjun Hangarge: *Old handwritten music symbol recognition using the combination of foreground and background projection profiles.* In *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pages 1–4. IEEE, 2018. 4

[16] Chanda, Sukalpa, Debleena Das, Umapada Pal, and Fumitaka Kimura: *Offline handwritten musical symbol recognition.* In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 405–410. IEEE, 2014. 4

[17] Plamondon, Réjean and Sargur N Srihari: *Online and off-line handwriting recognition: a comprehensive survey.* IEEE Transactions on pattern analysis and machine intelligence, 22(1):63–84, 2000. 4

[18] Swethalakshmi, Hariharan, Anitha Jayaraman, V Srinivasa Chakravarthy, and C Chandra Sekhar: *Online handwritten character recognition of devanagari and telugu characters using support vector machines.* In *Tenth international workshop on Frontiers in handwriting recognition.* Suvisoft, 2006. 4

[19] Zhang, Xu Yao, Yoshua Bengio, and Cheng Lin Liu: *Online and offline handwritten chinese character recognition: A comprehensive study and new benchmark.* Pattern Recognition, 61:348–360, 2017. 4

[20] Anstice, Jamie, Tim Bell, Andy Cockburn, and Martin Setchell: *The design of a pen-based musical input system.* In *Proceedings Sixth Australian Conference on Computer-Human Interaction*, pages 260–267. IEEE, 1996. 4

[21] George, Susan E: *Online pen-based recognition of music notation with artificial neural networks.* Computer Music Journal, 27(2):70–79, 2003. 4, 39

[22] Miyao, Hidetoshi and Minoru Maruyama: *An online handwritten music score recognition system.* In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 1, pages 461–464. IEEE, 2004. 4

[23] Lee, Kian Chin, Somnuk Phon-Amnuaisuk, and Choo Yee Ting: *Handwritten music notation recognition using hmm—a non-gestural approach.* In *2010 International Conference on Information Retrieval & Knowledge Management (CAMP)*, pages 255–259. IEEE, 2010. 4, 23

[24] Calvo-Zaragoza, Jorge and Jose Oncina: *Recognition of pen-based music notation: the homus dataset.* In *2014 22nd International Conference on Pattern Recognition*, pages 3038–3043. IEEE, 2014. 4, 22, 33

[25] Oh, Jiyong, Sung Joon Son, Sangkuk Lee, Ji Won Kwon, and Nojun Kwak: *Online recognition of handwritten music symbols.* International Journal on Document Analysis and Recognition (IJDAR), 20(2):79–89, 2017. 4, 33, 35, 40, 41

[26] Dalal, Navneet and Bill Triggs: *Histograms of oriented gradients for human detection.* In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005. 4, 11

[27] Calvo-Zaragoza, Jorge and Jose Oncina: *Recognition of pen-based music notation with finite-state machines.* Expert Systems with Applications, 72:395–406, 2017. 4

[28] Shatri, Elona and György Fazekas: *Optical music recognition: State of the art and major challenges.* arXiv preprint arXiv:2006.07885, 2020. 5

[29] Devlin, Jacob, Ming Wei Chang, Kenton Lee, and Kristina Toutanova: *Bert: Pre-training of deep bidirectional transformers for language understanding.* arXiv preprint arXiv:1810.04805, 2018. 6

[30] Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, *et al.*: *An image is worth 16x16 words: Transformers for image recognition at scale.* arXiv preprint arXiv:2010.11929, 2020. 6, 16, 26, 28, 31

[31] Touvron, Hugo, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou: *Training data-efficient image transformers & distillation through attention.* In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 6, 16, 26, 28, 31

[32] Guyon, Isabelle, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh: *Feature extraction: foundations and applications*, volume 207. Springer, 2008. 10

[33] Moravec, Hans Peter: *Obstacle avoidance and navigation in the real world by a seeing robot rover.* PhD thesis, Stanford University, 1980. 10

[34] Harris, Chris, Mike Stephens, *et al.*: *A combined corner and edge detector.* In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988. 10

[35] Lowe, David G: *Distinctive image features from scale-invariant keypoints.* International journal of computer vision, 60(2):91–110, 2004. 11

[36] Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool: *Surf: Speeded up robust features.* In *European conference on computer vision*, pages 404–417. Springer, 2006. 11

[37] Rublee, Ethan, Vincent Rabaud, Kurt Konolige, and Gary Bradski: *Orb: An efficient alternative to sift or surf.* In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011. 11

[38] Rosten, Edward and Tom Drummond: *Machine learning for high-speed corner detection.* In *European conference on computer vision*, pages 430–443. Springer, 2006. 11

[39] Calonder, Michael, Vincent Lepetit, Christoph Strecha, and Pascal Fua: *Brief: Binary robust independent elementary features.* In *European conference on computer vision*, pages 778–792. Springer, 2010. 11

[40] Freeman, Herbert: *On the encoding of arbitrary geometric configurations.* IRE Transactions on Electronic Computers, EC-10(2):260–268, 1961. 11, 23

[41] Zhang, Dengsheng and Guojun Lu: *Review of shape representation and description techniques.* Pattern Recognition, 37(1):1–19, 2004. 12

[42] Rosenblatt, Frank: *The perceptron: a probabilistic model for information storage and organization in the brain.* Psychological review, 65(6):386, 1958. 12

[43] Rosenblatt, Frank: *Principles of neurodynamics. perceptrons and the theory of brain mechanisms.* Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961. 13

[44] Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams: *Learning representations by back-propagating errors.* nature, 323(6088):533–536, 1986. 14

[45] Fukushima, Kunihiko and Sei Miyake: *Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition.* In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982. 14

[46] Chollet, Francois: *Deep learning with Python.* Simon and Schuster, 2021. 15

[47] Judd, Charles H: *The relation of special training and general intelligence.* Educational review, 36:28–42, 1908. 15

[48] Zhuang, Fuzhen, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He: *A comprehensive survey on transfer learning.* Proceedings of the IEEE, 109(1):43–76, 2020. 16

[49] Siddaway, Andy P, Alex M Wood, and Larry V Hedges: *How to do a systematic review: a best practice guide for conducting and reporting narrative reviews, meta-analyses, and meta-syntheses.* Annual review of psychology, 70:747–770, 2019. 22

[50] Sakoe, Hiroaki and Seibi Chiba: *Dynamic programming algorithm optimization for spoken word recognition.* IEEE transactions on acoustics, speech, and signal processing, 26(1):43–49, 1978. 23

[51] Jelinek, Frederick: *Statistical methods for speech recognition.* MIT press, 1998. 23

[52] Graupe, Daniel: *Principles of artificial neural networks*, volume 7. World Scientific, 2013. 23

[53] Vapnik, Vladimir: *The nature of statistical learning theory.* Springer science & business media, 1999. 23

[54] Platt, John C: *Advances in kernel methods. chapter fast training of support vector machines using sequential minimal optimization.* MIT Press, Cambridge, MA, USA, 3:185–208, 1999. 23

[55] Glorot, Xavier, Antoine Bordes, and Yoshua Bengio: *Deep sparse rectifier neural networks.* In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011. 24

[56] Tola, Engin, Vincent Lepetit, and Pascal Fua: *Daisy: An efficient dense descriptor applied to wide-baseline stereo.* IEEE transactions on pattern analysis and machine intelligence, 32(5):815–830, 2009. 25

[57] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis: *Grey wolf optimizer.* Advances in engineering software, 69:46–61, 2014. 25

[58] Ramesh, Aditya, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever: *Zero-shot text-to-image generation.* In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021. 27

[59] Radosavovic, Ilija, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár: *Designing network design spaces.* In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 28

[60] Steffensen, Johan Frederik: *Interpolation.* Courier Corporation, 2006. 30

[61] Rukundo, Olivier and Hanqiang Cao: *Nearest neighbor value interpolation.* arXiv preprint arXiv:1211.1768, 2012. 31

[62] Bland, J Martin and Douglas G Altman: *Statistics notes: measurement error.* Bmj, 312(7047):1654, 1996. 31

[63] Calvo-Zaragoza, Jorge, David Rizo, and José Manuel Inesta Quereda: *Two (note) heads are better than one: Pen-based multimodal interaction with music scores.* In *ISMIR*, pages 509–514, 2016. 33, 35

[64] Rebelo, Ana, G Capela, and Jaime S Cardoso: *Optical recognition of music symbols.* International Journal on Document Analysis and Recognition (IJDAR), 13(1):19–31, 2010. 33, 37

[65] Esteban, Antonio Ezquerro: *Música de la catedral de barcelona a la biblioteca de catalunya.* Biblioteca de Catalunya, Barcelona, 2001. 35

[66] Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei: *ImageNet Large Scale Visual Recognition Challenge.* International Journal of Computer Vision (IJCV), 115(3):211–252, 2015. 38

[67] Li, Liam, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-Tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar: *A system for massively parallel hyperparameter tuning.* Proceedings of Machine Learning and Systems, 2:230–246, 2020. 39

[68] Jaderberg, Max, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, *et al.*: *Population based training of neural networks.* arXiv preprint arXiv:1711.09846, 2017. 39

[69] Wang, Chi, Qingyun Wu, Silu Huang, and Amin Saied: *Economic hyperparameter optimization with blended search strategy.* In *International Conference on Learning Representations*, 2020. 39

[70] Bergstra, James, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl: *Algorithms for hyper-parameter optimization.* Advances in neural information processing systems, 24, 2011. 39

[71] Li, Lisha, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar: *Hyperband: A novel bandit-based approach to hyperparameter optimization.* The Journal of Machine Learning Research, 18(1):6765–6816, 2017. 39

[72] Chefer, Hila, Shir Gur, and Lior Wolf: *Transformer interpretability beyond attention visualization.* In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 782–791, 2021. 45