Recognition of Offline Handwritten Mathematical Expressions Using Convolutional Neural Network

Filip Bajraktari
Mathematical Grammar School, Belgrade, Serbia
email: bajraktarifilip@gmail.com

***Abstract*—This paper tackles the problem of handwritten mathematical expressions recognition using an algorithm from the field of optical character recognition. The segmentation of mathematical symbols was done with the help of numerical methods of vertical and horizontal histogram projections while for the classification of characters, a convolutional neural network was used. Measurements were done on CROHME's 2011 competition database, but the neural network was trained using Kaggle's database of math symbols. The success rate of the algorithm appears in Levenshtein distance and in the number of correct predictions. The neural network reached a success rate of 18%, while on the same database without symbols like sin, cos, tan, i, j, ..., and $\div$ 30% was reached.**

I. Introduction

The advancement of technology in the last 100 years has been substantial, and with that it has become cheaper and more accessible for the majority of people. We have smartphones and personal computers which have allowed us to acquire any information we would need in a matter of a couple of clicks. A new problem has arisen following this advancement, starting in the latter half of the 20th century, and that was the transfer of data from analog storage to digital. It was evident that it was impossible for us to transfer everything manually, it would take too long, so people came up with certain algorithms which will make a load of this problem lighter. In contrast to text, recognition of handwritten expressions is noticeably more difficult. First of all, the number of possible characters in an expression is much larger than the number of letters in the alphabet (there are around 40 characters that can appear whereas the number of mathematical symbols exceeds 100). Also, words do not have multidimensional nestings like exponents and indices. It represents a problem because every symbol should be first segmented and then classified in the program. Due to these reasons, the problem of expression recognition is currently the most difficult problem in computer vision.

Considering the difficulty of the problem, it has been worked on for the last couple of decades. Heuristics that stands behind mathematical laws and syntax is the reason why an algorithm, which would be simple enough for computers to execute and complex enough to recognize a plethora of mathematical expressions, was so hard for researchers to come up with. So far, the most successful methods were shown in the paper [3,4]. One of the methods was convolutional neural network which was used as a classifier for mathematical symbols and another one was to enrich the data set in order to further increase the precision of the system.

The focus of this paper was not to train a neural network but to tackle the problem of segmentation using two nested recursive functions and the heuristics of grouping the segmented symbols. Two types of measurements were done in this paper. The first one was done regularly, while the other was done on an enriched data set. The results showed that the latter had noticeably better precision which was within expectations because the width of characters is not uniform due to the variety found in different handwritings.

II. Method

The complexity of the problem implies the complexity of the algorithm for its execution. Two key steps of this algorithm are segmenting and regrouping of the segmented symbols into latex. In order for segmentation to be done, some preparation is necessary. Firstly, we need to translate the image from RGB to Grayscale and then remove the noise. One of the most crucial subalgorithms is noise reduction because if some noise stays, it will be considered as a foreground pixel, a pixel that carries information, and as such it will be classified as a math symbol which is not present in the data set and make our result meaningless.

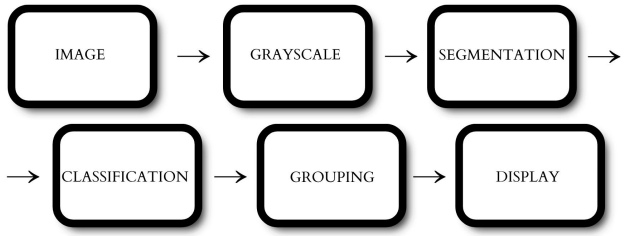

Fig. 1. Algorithm scheme for recognition of handwritten mathematical expressions

Apart from this, data set enrichment was used. In the database which was used for training the neural network, symbols are only one pixel wide. Primarily, from experience, we know that handwritings heavily differ, and so do the pencils we use which means we need a system that will be invariant to these different styles and different interpretations of the same symbol. The solution that was decided upon was to use $dilate$ function from the library $OpenCV$ with a kernel K.

$$K = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Using this kernel we can be sure that each symbol will be widened by only one pixel.

A. Image processing

Image processing consists of all modifications applied to an image which make it easier for further processing. This part can be divided into 3 smaller segments: RGB to Grayscale conversion (images that exclusively have shades of gray), noise reduction, and image binarization. If we worked with an RGB image, we would have to worry about the amount of red, green, and blue present in each pixel. Due to this fact, we simplify the image by converting three different values into a unique one (Grayscale value) using $cvtColor$ function from the library $OpenCV$.

Since hardware is something man-made, it is logical that it will not always work as fully as intended. Hardware faults are called noise. These flaws are becoming less common due to advancements in electronics, but they are still prevalent and might cause issues during further processing. As Gaussian noise is the most common in images, we will only deal with its isolation, while we will consider others negligible. The method used in the experiment for removing noise is utterly simple: we will replace the color of a certain pixel with the average value of other pixels from its environment. To do so, we will need to discover pixels that are similar. They are not required to border with a specific pixel. Consider the case of repeating patterns. As a result, scanning a proportionally big region of the image to discover as many pixels that resemble the original pixel as possible is critical. Two pixels resemble each other if they are approximately equal. To do this, a function from the $OpenCV$ library called $fastNLMeansDenoising$ was used.

Binarization is a process in which each Grayscale pixel value is assigned one of two possible values, as the name alludes, those values are 0 or 255. In the experiment, pixels with a value of 255 will be foreground pixels, while those with a value of 0 will be background pixels. In order to do that, it is necessary to establish a threshold for each pixel that allows us to assign an adequate value. As in our case, the brightness of the image will not be perfect, finding the local limit values would enable us to separate the expressions more precisely. Therefore, the experiment used the adaptive threshold function from $OpenCV$ library and for the adaptive method, $adaptiveThreshGaussianC$ was used.

B. Segmentation

Observing a simple neural network, we can notice that it has a number of neurons at its output, where each neuron represents the probability with which the neural network can assess whether a particular object is in the image or not. Consider now an example of mathematical expressions. The set of all potential formulas is determined by all possible combinations of mathematical symbols, which makes the recognition of whole expressions impossible, but the set of mathematical symbols is finite and due to that fact segmentation is performed.

The previous paragraph introduced the importance of noise reduction and in this one, it will be further expanded upon. To begin with, we should introduce a numerical method called histogram projection, which is used to examine whether a part of the expression is in a given column or row where image row and column refer to the row and column of pixels that make up the image. In the given experiment, two different histogram projections were used: vertical and horizontal. The vertical histogram projection calculates the number of foreground pixels for each column in the image. The horizontal histogram projection is defined analogously [4].

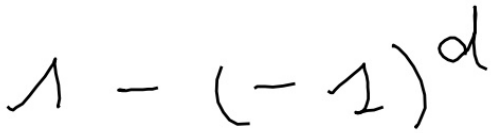

Fig. 2. An example of a mathematical expression

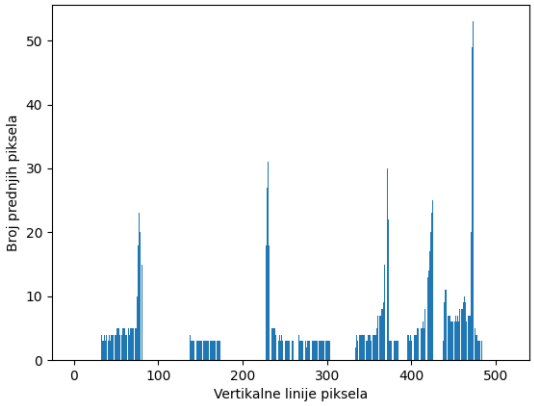

Fig. 3. Vertical histogram projection

If we take the example in figure 2 into observation and look at its horizontal histogram projection, we can come to a conclusion that there is only one expression because we can notice only one peak on the diagram. A peak is a series of contiguous rows in the picture such that each row contains at least one front pixel. Similarly, by observing the vertical histogram projection, seven peaks can be observed that directly correspond to the seven symbols of the mathematical expression. In this example, one horizontal and one vertical projection were enough to find all the symbols in the image.

Of course, this does not have to be the general case with mathematical expressions. On the example of a fraction in any expression, it can clearly be seen that after the vertical and horizontal projection most symbols would have been successfully segmented, but with fractions, the numerator,

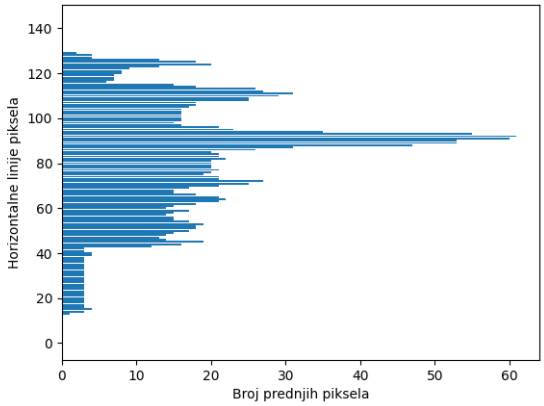

Fig. 4. Horizontal histogram projection

denominator, and the fractional line will be recognized as a single block. Intuitively, a new idea is introduced. We add another horizontal projection. This will improve the algorithm, but it is possible that either the numerator or the denominator will contain a new mathematical expression. Theoretically, such a sequence can go on indefinitely. For the simple reason, a finite number of horizontal and vertical projections would be just an approximation of the problem, so the possibility of recursive programming in the Python programming language was used in this experiment. As for any recursion, it is necessary to define the base case, the case when the recursion is interrupted. The first idea that arises is to stop the recursion when we cannot get new smaller block images, i.e. when only one peak appears on the graph of both histograms. This would work in laboratory-perfect conditions, assuming that each symbol is perfectly written. If we take the expression a^y, we can notice that it is very likely that some part of y will cross over the vertical projection of a. In that case, the recursion would be stopped even though the segmentation is not completed. As a correction for this problem, at the end of each recursion, a flood fill algorithm was performed. The flood fill algorithm converts an image matrix block into an unweighted undirected graph and determines the number of connected components on it and at the same time marks pixels from the image with the corresponding ordinal number of the graph component. This algorithm was implemented using the $label$ function from the library $SciPy$.

C. Classification

A convolutional neural network (CNN) was used as the classification system. CNN is one of the classes of deep neural networks and one of the many deep learning algorithms that allow us to use self-optimizing weights and biases to identify certain objects, i.e. to enable the computer to have the same perception as we do. The main difference between CNN and regular neural networks is the presence of convolution. Convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other. In the case of CNN, the convolution serves us to make a feature map from the input data using certain filters and kernels. In the examples with numbers, these features can be certain curves or strict sharp angles,...

The computer nervous system is very similar to ours. It consists of a network that contains a huge number of neurons. Based on the impulse received by the neuron and the significance of each impulse, the neuron decides whether to activate and forward the impulse further through the network or to keep the axon closed and thus prevent the flow. In short, each neuron can be viewed as a set of inputs, a set of weights, and an activation function.

There are different types of neurons and they depend on the stage (layer) of the network in which they are located. The three basic layers in each neural network are the input, hidden, and output layers, with only one input and output layer, while the hidden layers may be more depending on the network structure and the type of the problem. In the case of CNN, there are convolutional, activational, pooling, and fully connected layers. The architecture used in the experiment is $[Conv - Relu - Pool] \times 2 - [FC - Relu] \times 2 - [FC]$. Also, 2×2 $max\ pooling$ with a step of length 2 was used and the algorithm was implemented in the $PyTorch$ library.

D. Grouping

Although a meaningful recognition of all symbols from the image and their grouping into an expression is an easy task for humans, it turns out not to be the simplest process for a computer for two reasons. Firstly, not all symbols are written without taking your hand from the paper. Secondly, a vast majority of the expressions have more than one dimension, i.e. there are exponents, indices, fractions,...

As we have said many times so far, mathematical expressions are characterized by multidimensional nesting, so we memorized the distance for each symbol we inputted into the neural network from the axis that goes through the center of mass of the previous symbol. Based on this line, we determine whether the current symbol is a continuation of a mathematical expression or an exponent, or an index of the previous symbol. We find the width and the height of the symbols using the method of connected components, remembering the northernmost, southernmost, westernmost, and easternmost pixels of each symbol and then calculate their differences. If the axis of the previous symbol intersects the current symbol, we say that the current symbol is a continuation of the mathematical expression. If the axis of the previous symbol is below or above the current symbol, we say that the current symbol is an exponent or an index of the previous symbol.

Considering the problem of disconnected symbols, the situation is more difficult. For the explanatory purpose, we can assume that the expression has no multidimensional nesting, i.e. the baseline crosses every symbol in the expression. The following are potential statements that need to be corrected:

1) sign of equality, in case we come across two minus signs whose centers of mass are in the range of the other (by

x-axis) and differ by y-axis, we can connect them and look at them as a sign of equality because we know that a series of two minuses is an incorrect mathematical notation and cannot appear in any expression.

2) trigonometric functions, if we came across a series of characters such as c, o, s we could combine them into cos, the same goes for tan, log,...
3) sin, for sine we have a specific case because i cannot be written without lifting the pen off the paper, so in that case the classification string would look like s, $|$, n. In that case, we would consider this combination as sin.
4) $\leq$ i $\geq$, if there were a string of characters $<$, $-$ or $>$, $-$, we would classify them as $\leq$ and $\geq$. The reverse order also applies.
5) $\pm$, if there were a string of characters $+$ i $-$, or vice versa, they would be merged into the $\pm$.

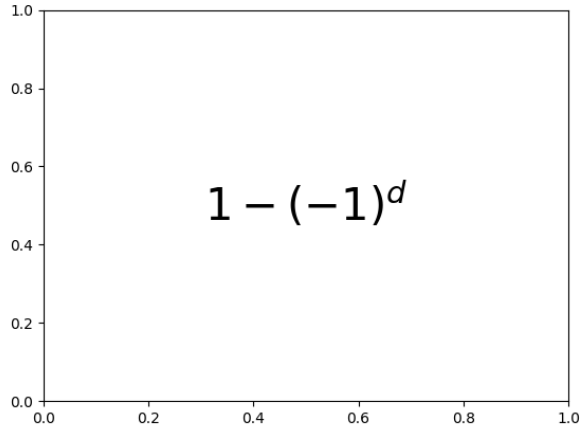

Fig. 5. The final look of the expression after grouping

III. Results

The first part of the research refers to the difference in the accuracy of the system when classifying mathematical expressions written using all eighty-two possible symbols from the used database and mathematical expressions that do not contain certain symbols. It is important to mention that samples in the database are given in the form of XTML files [1], so it was necessary to first convert those files to jpg and then perform the previously described algorithm.

!	(	)	+	,	-	=	[	]	{
}	0	1	2	3	4	5	6	7	8
9	A	α	|	b	β	C	$\cos^*$	d	δ
$\div^*$	e	$\exists$	f	$\forall$	/	G	γ	$\geq$	$>$
H	i^*	$\in$	∞	$\int$	j^*	k	l	λ	$\ldots^*$
$\leq$	$\lim^*$	log	$<$	M	μ	N	$\neq$	o	p
ϕ	π	$\pm$	$\prime$	q	R	$\rightarrow$	S	σ	$\sin^*$
$\sqrt{x}$	$\sum$	T	$\tan^*$	θ	$\times$	u	v	w	x
y	z								

The table represents all the symbols that are in the database of mathematical symbols used in this experiment, while the asterisk indicates those symbols that were not used in the second part of the research.

Results		
enrichment/symbols	all symbols	selected symbols
with enrichment	17.29%	29.90%
without enrichment	8.45%	13.17%

Although the accuracy of neural networks trained with widened and regular symbols is approximately equal, 98% and 97% respectively, a drastic difference in accuracy is observed when recognizing complete mathematical expressions. First of all, in the original database, the thickness of each symbol is only one pixel. This would mean that the expressions have to be taken from a great distance in order to make the original expressions thinner, which is not the case in the database of written mathematical expressions. The expressions in this database are represented in a such way that they occupy a proportionally large part of the image. In this case, we can expect that the expressions are actually magnified and therefore bold. By doubling the symbol database and widening it by exactly one pixel, we allow the neural network to be invariant to the thickness of the symbols, and therefore invariant to the distances from which the images were taken. The small difference in the accuracy of the neural networks themselves is reflected in the number of training examples from which the neural network learned. As the neural network both trained for three epochs, none of them reached saturation because there was no drop in the accuracy of the neural network during all three epochs, so the network with more training examples proved to be better. As both percentages are quite high, we cannot say that a difference of 1% had an impact on the difference in the accuracy of the whole system.

The second part of the research is about how certain symbols make a difference in the final precision of the system. Eight symbols, which are often used in mathematics and which present potential problems due to their complexity, have been removed from the entire database. In both cases (with or without symbol widening) a difference in the accuracy of the system is observed. This is due to the fact that trigonometric functions can be written in many different ways. For example, some people write $\sin$ so that all three letters are connected,

some separate the words si and n, and some write all three-letter separately. If some people write in small print, it is possible to associate sin with '('. In this case, the neural network will not be able to accurately recognize the expression sin(because it does not exist in the symbol database. It is also important to emphasize that the overlap of symbols does not necessarily have to be related to trigonometric functions. Actually, relatively low results are mostly due to the problem of overlapping symbols, because the computer cannot distinguish whether a series of the intertwined lines belongs to one symbol or to some set of symbols.

The last part of the research is related to the percentage accuracy of recognition. In the first two parts, it was observed only the final product was compared to whether the output model fully corresponds to the label. In this part, Levenshtein distance was used to compare how much the outputs of the system differ from the given exact value, ie. Levenshtein distance calculates the minimum required number of changes to the output latex code in order to match the corresponding label [2].

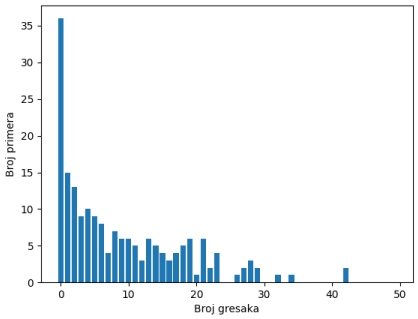

Fig. 6. Levenshtein distance on the dataset with all symbols

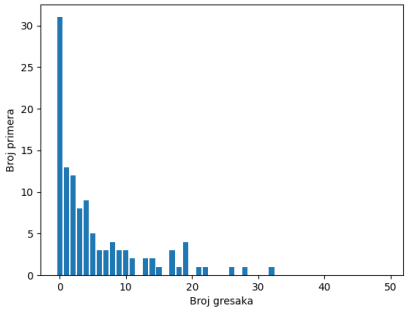

Fig. 7. Levenshtein distance on the dataset with selected symbols

From the given graphs it can be noticed that although the general accuracy is relatively low, the number of examples decreases exponentially with the increase in the number of errors per example. Also, the average percentage accuracy of recognition for one mathematical expression is 75%, which shows that the use of this system would drastically accelerate the mentioned digitalization.

IV. Conclusion

Testing neural networks over small samples reveal small differences in their accuracy. This is explained by the unique characteristics of each mathematical symbol, which allows the neural network to accurately classify. A convolutional neural network with two $covn - pooling$ layers and three fully connected layers was used in all experiments. The activation function was $ReLU$. Three different measurements were performed in this paper. The first measurement refers to the difference in the classification accuracy of the system in relation to whether or not widened symbols were used during neural network training. A neural network system trained with widened symbols proved to be twice as good at classifying mathematical expressions as a neural network system trained with the symbols of constant width. These results are justified because we cannot expect different people to have the same writing style nor to have the same pens. The second measurement refers to the classification precision of the system in relation to which mathematical symbols were used in the expressions. It has been found that symbols that have a complex spatial structure, as well as a complicated $latex$ notation, undermine the precision of both systems. This result is in accordance with the results achieved in the reference work. In this paper, it was shown using Levenshtein distance that the accuracy of the system at the level of individual mathematical expressions is 75%, which shows that the use of this system would drastically accelerate the mentioned digitalization. Also, it was noticed that the main problem is the existence of overlapping symbols. Therefore, as far as further improvements are concerned, by developing an algorithm that would effectively separate the two overlapping symbols, we could increase the classification accuracy of the system.

Acknowledgment

First of all, I would like to express my deepest appreciation to my mentor Filip Parag who successfully guided me through this journey and taught me a lot. Furthermore, I could not undertake this journey without the Science and Engineering Center "PFE" which even during the time of the pandemic organized seminars online. Also, I would like to thank Mathematical Grammar School for the excellent knowledge I gathered during these four years of my high school education and to Senior Research Associate Nikola Zogovic who helped me in writing the paper.

References

[1] Ink markup language, http://www.w3.org/TR/InkML/
[2] Levenshtein distance, https://en.wikipedia.org/wiki/Levenshtein_distance
[3] Catherine Lu and Karanveer Mohan, *Recognition of Online Handwritten Mathematical Expressions Using Convolutional Neural Network*, cs231n project report Stanford 2015
[4] Md Bipul Hossain, Feroza Naznin, Y.A. Joarder, Md Zahidul Islam , Md Jashin Uddin, *Recognition and Solution for Handwritten Equation Using Convolutional Neural Network*, 2018