Ukrainian Catholic University

Faculty of Applied Sciences

Business Analytics & Computer Science Programmes

Photo to LaTex converter

Linear Algebra final project report

Authors:

Bohdan Pelekh

Fedir Zhurba

Maksym Mykhasyuta

16 May2023

1. Introduction

The project aims to develop a simple handwritten text recognition system based on computer vision and the usage of linear algebra. This system should correctly recognize simple mathematical expressions and symbols and translate them into LaTex.

Having a system that can convert handwritten text to LaTeX has several important benefits:

1. Improved Efficiency: Converting handwritten text to LaTeX automatically saves time and effort compared to manually transcribing or typing mathematical expressions. It eliminates the need for individuals to spend significant amounts of time translating handwritten equations into LaTeX code.
2. Enhanced Accessibility: Handwriting is a natural and intuitive way for many people to express mathematical ideas. By providing a system that can recognize and convert handwritten text to LaTeX, it enables individuals who prefer to write by hand to easily communicate and share their mathematical expressions with others who use LaTeX for typesetting.
3. Standardization and Consistency: LaTeX is widely used in academia and scientific publishing for its ability to produce high-quality mathematical typesetting. By converting handwritten text to LaTeX, the system ensures that the resulting expressions adhere to a standardized format, allowing for consistent and professional-looking mathematical notation across different documents and platforms.
4. Integration with Digital Tools: LaTeX is compatible with various digital tools and platforms used in mathematics, such as typesetting software, equation editors, and document preparation systems. By converting handwritten text to LaTeX, the system enables seamless integration with these tools, facilitating the incorporation of handwritten mathematical expressions into digital documents, presentations, and online platforms.

5. Collaboration and Sharing: LaTeX is a commonly used format for sharing mathematical content among researchers, students, and educators. By converting handwritten text to LaTeX, the system enables individuals to easily share their handwritten work in a format compatible with LaTeX-based platforms, fostering collaboration, feedback, and knowledge exchange within the mathematical community.

The research area of converting hand-written text to LaTex from images is an active and interdisciplinary field that combines computer vision, mathematics, machine learning, and natural language processing. Our work concentrates on introductory linear algebra and computer vision approaches that allow us to develop sophisticated and correct machine-learning models.
The input to the system will be an image of a handwritten simple mathematical expression or symbol, which may contain characters such as numbers, letters, operators (e.g., plus, minus), and special symbols.
The system must be able to correctly recognize and interpret these characters and symbols, even if they are written in different styles or orientations. To accomplish this, our system will use computer vision techniques to preprocess the image and extract the relevant features.

2. Problem formulation

In our project, there are several problems to be solved. In the first step, it is necessary to preprocess the image to make it possible to work with. The next important step is to separate the written formula on the segments to recognize each symbol separately. The following step is to find an efficient way to extract features of the symbol, which will be used in the classification of symbols. Finally, we perform the classification of symbols to determine the symbol class and relate it to the corresponding LaTex character.

3. Steps implementation

1. Preprocessing

    1.1. Grayscaling

    Grayscaling is essential in the project as it simplifies the image data, standardizes the representation, improves the visibility of the text, reduces computational complexity, and enhances compatibility with existing algorithms. These benefits contribute to more accurate and efficient handwritten text recognition, ultimately improving the overall performance of the system in handwritten text recognition.

    1.2. Denoising

    Gaussian blur is a widely used image filtering technique in image processing and computer vision. It applies a convolution operation using a Gaussian kernel to the image, resulting in a blurring or smoothing effect. The Gaussian kernel is a 2D matrix representing a Gaussian distribution, with the center having the highest weight and decreasing weights towards the edges.

    Gaussian blur effectively reduces noise in images, especially high-frequency noise or pixel-level variations. By blurring the image, random noise or unwanted artifacts are smoothed out, resulting in a cleaner and more visually pleasing image. This approach can effectively reduce the impact of noise and enhance the quality of the denoised output.

    1.3. Thresholding

    The result of thresholding is a binary image, where pixels are either black or white, representing foreground and background, respectively.

We have implemented adaptive thresholding (Otsu's method) to automatically determine suitable threshold values based on local image statistics. This adaptive thresholding approach can adapt to variations in lighting conditions, background noise, or individual writing styles, improving the robustness and accuracy of your handwritten text recognition system.

By incorporating thresholding techniques, we enhance the quality of the input images, segment text regions effectively, and enable more accurate character recognition, leading to improved performance and reliability of the text recognition system.

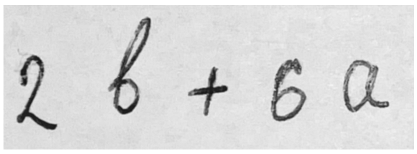
2. Segmentation

2.1. Component Connection Analysis

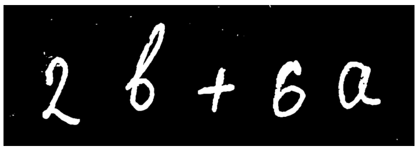
Using CCA we separate symbols in our binary matrix, performing a common technique of separating components of 1’s. In our implementation, we use 8 connectivity, when we try to connect pixel not only with its row and column neighbors but also with diagonal neighbors. This way we get the coordinates of groups of points. After that, we reject small groups of points as a way to filter noise and receive only valuable components.

2.2. Boundary box

After performing Component Connection Analysis, we get a list of components, their heights, and widths. Then we detect the edges of each component (symbol) in our case. This way, we get a rectangular matrix of each separate symbol, which is transferred to the next step.



Initial image + Gaussian blur



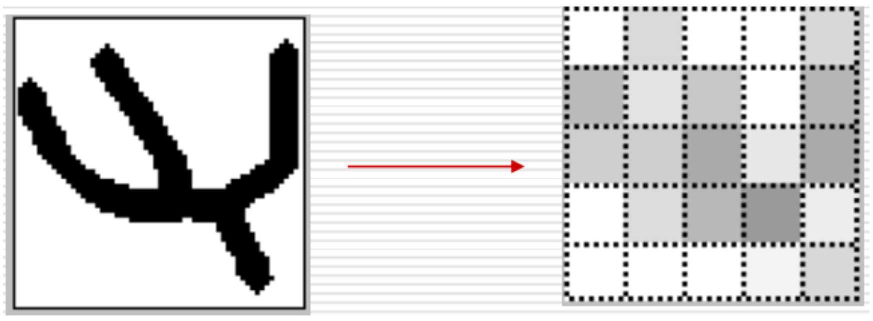
Adaptive thresholding, we can see some noise here.

Symbols received after CCa

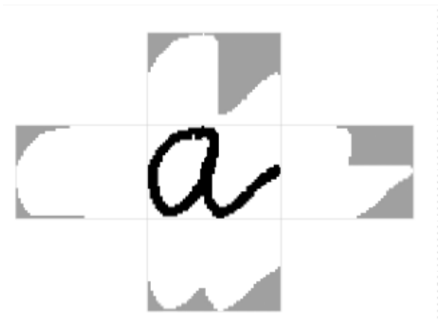
3. Feature extraction

   For the classification algorithm, it is important to determine the universal way to receive features from the separated symbols in the boundary boxes. One of the most useful and efficient ways to do it, having a matrix of non-standard size is separating the matrix on the determined number of squares (49 in our case).

   Then we calculate the relation of black pixels to the general number of pixels in the square. As a result of feature extraction, we get the 7*7 matrix of black pixel density. This matrix is a symbol feature that is used for further classification.



Second feature extraction method is a profile. The profile counts the number of pixels (distance) between the bounding box of the character image and the edge of the character. The profiles describe well the external shapes of characters and allow to distinguish between a great number of letters, such as “p” and “q”.



4. Classification

There are a lot of classifiers, beginning from the most naive and simple: Binary Tree Classifier or Random Forest, which are quite easy to understand and implement. However, attempts to use those methods were not so successful in case of their generality and simplicity. According to this reason we need to use more specialized and complex algorithms.

The next step was trying to use the Naive Bayes algorithm, which showed some results. However, Naive Bayes assumes that the features are independent, which may not be true in practice and can result in suboptimal performance in some cases. So, we need algorithms that can handle it.

The next possible algorithm was K-Nearest Neighbours, which is quite universal and can handle both linear and non-linear decision boundaries. However, it can be too sensitive to the method of feature evaluation and also be too computationally expensive on our dataset. Finally, we have chosen the SVM algorithm as one of the most flexible and at the same time precise algorithms of binary classification. Overall, SVM is a powerful and versatile algorithm that can be applied to a wide range of computer vision tasks and has been shown to achieve state-of-the-art performance in many applications. So, it is more than suitable for our project task. What is more, it is widely connected with linear algebra, which gives us the possibility to use and improve our knowledge.

4. SVM general description

SVM is a machine learning algorithm that is used for classifying data into two categories. It works by finding a line (or a plane) that separates the data into two groups as well as possible. The idea is to maximize the distance between the separating line and the closest points from both groups. This is called the "margin", and the larger the margin, the better the separation between the two groups.

If the data is not easily separated by a straight line (or a plane), SVM can still work by transforming the data into a higher-dimensional space using a mathematical function called a "kernel". This allows SVM to find a more complex separation boundary to better distinguish between the two groups.

Advantages:

- SVM works well with high-dimensional data and can handle many features without overfitting.
- SVM is effective in cases where the number of features is much greater than the number of data points.
- SVM can handle both linearly separable and non-linearly separable data by using the kernel trick to map the data into a higher-dimensional space.
- SVM has a regularizing effect, which helps to prevent overfitting and improves generalization performance.
- SVM can be used for both binary classification and multi-class classification problems.

Disadvantages:

● SVM can be sensitive to the choice of kernel function and its hyperparameters, which can affect the performance of the algorithm.

● SVM can be computationally expensive to train on large datasets, especially when using non-linear kernels.

● SVM can be difficult to interpret the results, making it less suitable for applications where interpretability is important.

● SVM can be prone to overfitting when the data is noisy.

● SVM can be sensitive to the presence of outliers, which can affect the placement of the decision boundary.

As we can see, SVM is a powerful tool for data classification, which can help to avoid overfitting the model, which is extremely useful in our case. However, when the dataset increases, we may face the problem of too complicated and long computations of the kernel function. But the popularity of SVM in the field of computer vision and text classification proves that it can give great results if the data preprocessing is done correctly.

5. SVM theoretical basis

Support Vector Machines (SVMs) are a type of supervised learning algorithm that can be used for both classification and regression tasks. SVMs are based on the idea of finding the hyperplane that maximally separates the data points of different classes.

The hyperplane that maximizes the margin between the classes is known as the maximum margin hyperplane, and it is the solution of the optimization problem that defines the SVM. Suppose we have a set of training data points, $(x1, y1), (x2, y2), ..., (xn, yn)$, where xi is a d-dimensional feature vector, and y_i is the corresponding class label (either +1 or -1).

The goal of the SVM is to find a hyperplane of the form $w^T x + b = 0$ that separates the positive and negative data points with the largest margin, where w is the weight vector and b is the bias term.

The margin is defined as the distance between the hyperplane and the closest data points from each class, and it can be computed as $2/||w||$, where $||w||$ is the L2-norm of the weight vector.

The optimization problem that defines the SVM can be written as:

minimize $1/2 * ||w||^2$

subject to $y_i(w^T x_i + b) \leq 1$ for all i

This is a convex optimization problem, which can be solved using a variety of techniques, such as quadratic programming or gradient descent. The solution of the optimization problem gives us the weight vector w and the bias term b, which define the maximum margin hyperplane.

However, in practice, the data may not be linearly separable, meaning that there may not exist a hyperplane that separates the positive and negative data points without any errors. In this case, we can introduce a slack variable

The optimization problem becomes:

minimize $1/2 * ||w||^2 + C * \sum(\xi i)$

subject to $y_i(w^T x_i + b) >= 1 - \xi i$ for all i

$\xi i \geq 0$ for all i,

where C is a hyperparameter that controls the trade-off between maximizing the margin and minimizing the errors.

The solution to this optimization problem gives us a hyperplane that separates the data points with a margin that is as large as possible while still allowing for some errors. In addition to the linear SVM, there are also kernel SVMs, which can handle non-linearly separable data by mapping the data points into a higher-dimensional feature space, where they can be separated by a hyperplane.

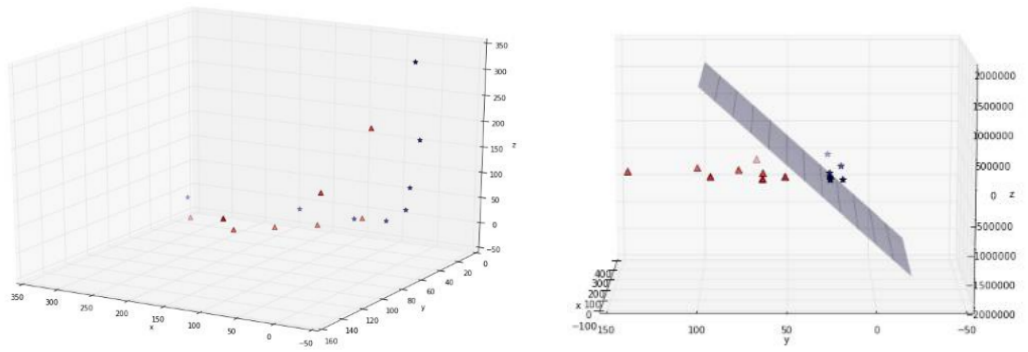
The mapping is performed by a kernel function, which computes the inner product of the feature vectors in the higher-dimensional space without actually computing the feature vectors themselves. Some common kernel functions include the polynomial kernel, the Gaussian (or RBF) kernel, and the sigmoid kernel.

6. More about kernels

6.1. General idea

Even if the original data is in two dimensions, we can transform it before feeding it into the SVM. One possible transformation would be, for instance, to transform every two-dimensional vector (x_1, x_2) into a three-dimensional vector. For example, it is possible to perform a polynomial mapping by applying the function $\phi : \mathbb{R}^2 \to \mathbb{R}^3$ defined by:

$$\phi(x_1, x_2) = (x_1^2,\ \sqrt{2}x_1 x_2, x_2^2)$$



Example of transforming non-linear separatable data to a higher dimensional space with further separation.

The vectors x_1 and x_2 belong to $\mathbb{R}^2$. The kernel function computes their dot product as if they have been transformed into vectors belonging to $\mathbb{R}^3$, and it does that without doing the transformation, and without computing their dot product! To sum up: a kernel is a function that returns the result of a dot product performed in another space. More formally, we can write:

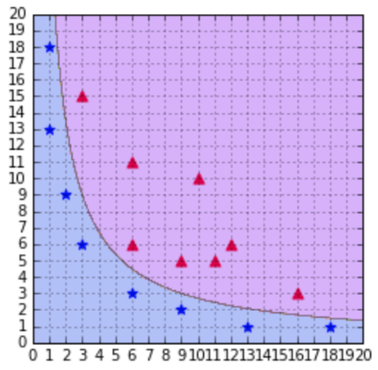
Given a mapping function $\phi : \chi \to \vartheta$, we call the function $K : \chi \to \mathbb{R}$ defined by $K(x, x\prime) = \langle \phi(x), \phi(x\prime) \rangle \vartheta$, where $\langle \cdot\ ,\ \cdot \rangle \vartheta$ denotes an inner product in ϑ, a kernel function.

The simplest kernel function is a linear kernel function, which is usually defined as a simple inner product. Despite its simplicity, the performance of this method is not so high and has a lot of limitations, so in the next sections, we will look closer at more advanced types of kernel functions.

6.2. Polynomial kernel

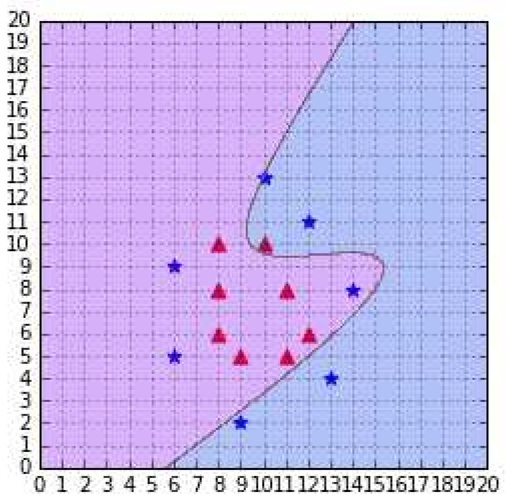
Generally, the polynomial kernel function approach is a more advanced version of the linear function. It has two parameters: c, which represents a constant term, and d, which represents the degree of the kernel.

$$K(x, x') = (x - x' + c)^d$$



In this example, we can see how polynomial functions work with linearly non-separatable data. In most general cases this type of kernels perform quite good, however, they are not universal.

Sometimes polynomial kernels are not sophisticated enough to work like such a dataset:



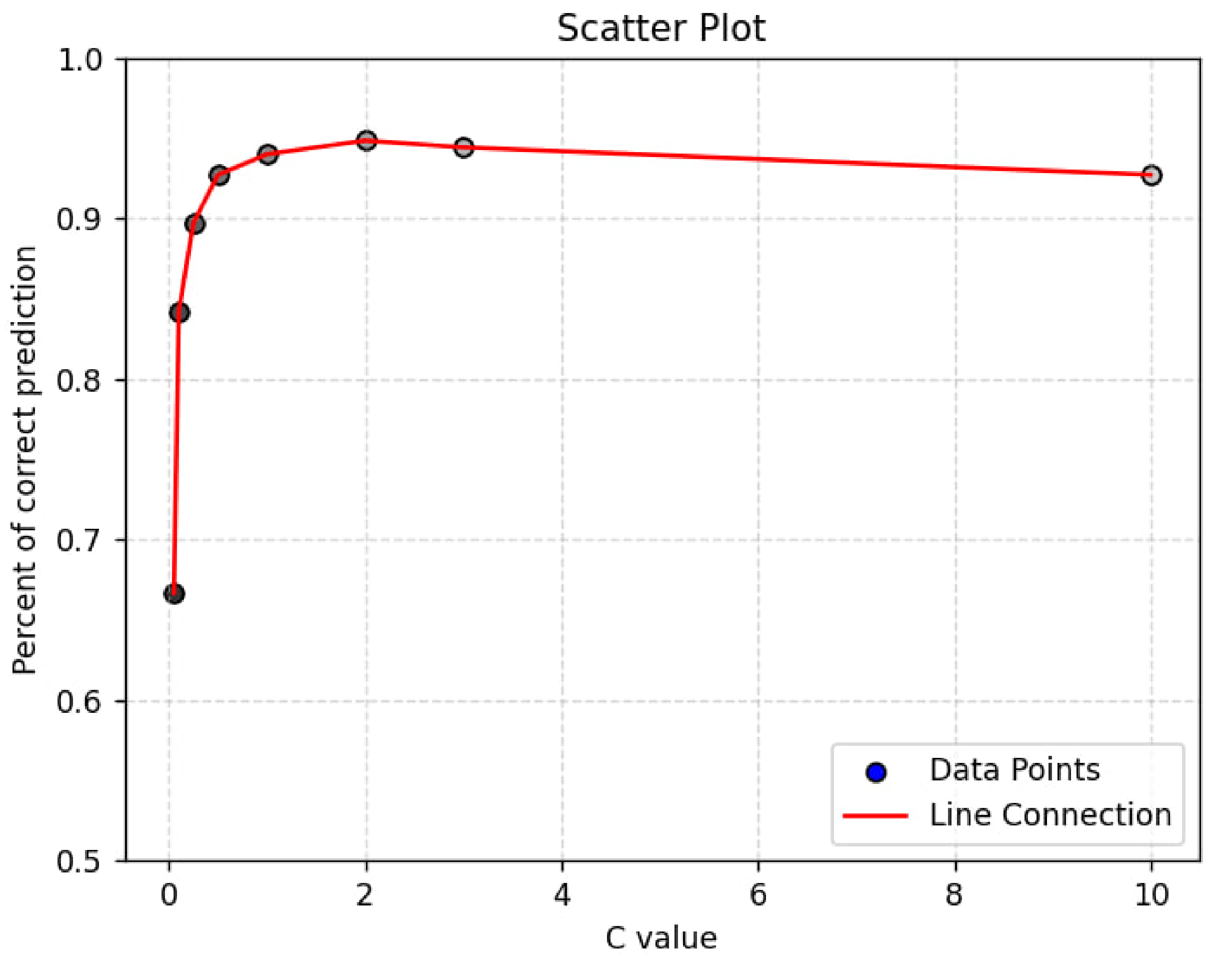
6.3. RBF (Gaussian) kernel

As mentioned in the previous section polynomial kernels can give bad results with extraordinary data. When we have a complicated dataset, this type of kernel will show its limitation. This case calls for us to use another, more complicated, kernel: the Gaussian kernel. It is also named RBF kernel, where RBF stands for Radial Basis Function. A radial basis function is a function whose value depends only on the distance from the origin or some point.

The RBF kernel function is $K(x, x') = \exp\left(-\|x - x'\|^2\right)$.

In our project, we decided to use the RBF kernel because it is the best choice for non-specialist models. This approach is more flexible and allows to avoid limitations according to data distribution.

While using the RBF implementation in our program we have faced the problem of detecting best value of constant C. After the research and building the plot, mentioned below, we have received the best value C=2, for values lower than it our model is underfitted, and for greater values - overfitted. As we can see, SVM deals good with overfitting, which is one of the main advantages of this algorithm.



7. Clustering of the model

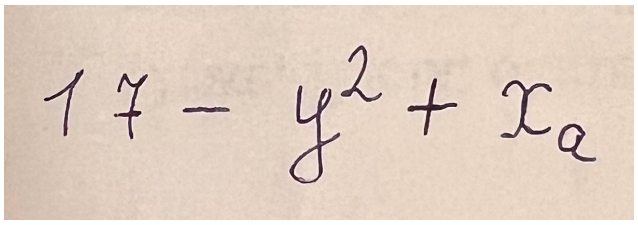
Our SVM model uses one-to-one strategies while working with multiclassing. Such an approach has quadratic complexity to increase the number of classes for classification. So, it needs a lot of time to compare each pair of classes with each other. We considered the possibility of clustering some classes by their mean values. In this way, we will be able not to compare all classes with each other, conducting classification only in the selected cluster. Although such method leads to improving time performance, it can harm of quality of classification.

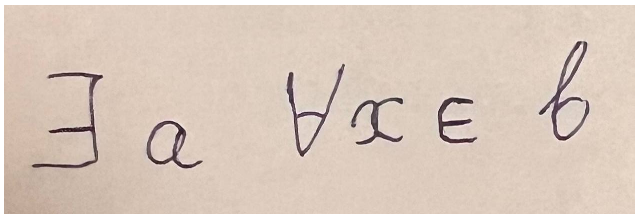
time	accuracy%	No. clusters
6.010516881942749	0.8418803418803419	3
6.301044464111328	0.8846153846153846	2
6.739951133728027	0.9487179487179487	1

We can see that although this method helps to improve time performance, the damage to quality is more significant.

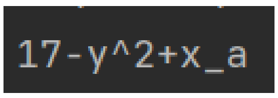
8. Demonstration

Before the project presentation, we are planning to roll out an application that does the conversion just like the cli version but with a user-friendly interface. We are not sure if PlayMarket will accept our app in the nearest future, but if so, we'll provide ulr to download it and try it for yourself.

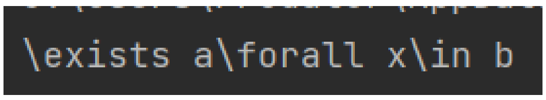




As a demonstration, we took 2 formulas. We've run Python script for both of them and received LaTex results:



```
17-y^2+x_a
```



```
\exists a\forall x\in b
```

9. Conclusion

As a result of this project, we have successfully implemented a basic photo-to-LaTex conversion algorithm, using various techniques of image preprocessing, symbol separation, and feature extraction. We had an experience with different algorithms of classification and learned to work with SVM algorithms, based on kernel functions. Overall, we had earned useful experience in creating a project and going through all steps of design, implementation, and presentation.

Link to our GitHub repository: *https://github.com/Bohdan213/LA_project*

Literature:

1. *Alexandre Kowalczyk - Support Vector Machines Succinctly (2017)*
2. *Pulak Purkait, JRF ECSU, Indian Statistical Institute - Feature Extraction Methods for Handwritten Character Recognition (9th North-East Workshop on Computational Information Processing)*
3. *Andreas Christmann, Ingo Steinwart - Support Vector Machines(2008)*