

# Proposal for the project: system of linear equations solver with deep learning

## Problem

Over the course of the last months e-learning gets more and more to the focus of the broad population. The Covid-19 pandemic forces nearly every country in the world to apply restrictions on the number of people allowed to gather in public spaces including schools. One of the major effects of these lockdowns is the breakdown of public education as we know it. Students, teachers and parents have to turn to modern and innovative approaches for teaching on a remote basis.

This trend results in a rising demand for digital solutions in the education sector enabling students to learn at home and at their individual pace. Furthermore these solutions enable children in less developed countries to catch up with their peers in the so-called First and Second World without having access to a widespread and sophisticated public education infrastructure.

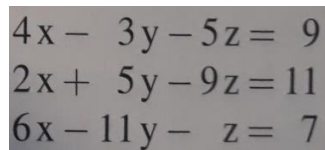
Mathematics is one of the essential domains in the course of every child's formal education providing the necessary skills to thrive in the modern technical and digitalized world. Systems of linear equations (SLE) are a central subject of every Mathematics curriculum. Enabling to solve problems ranging from flows in complex systems to analytical geometry.

There are already different projects out there tackling these problems. Google DeepMind released a paper in 2019 (Saxton et. al, 2019) stating a poor performance with establishing a neural net answering high-school math problems. Especially with problems of higher complexity like solving a 2D system of linear equations the three trained models output the right answer with a probability of 55% - 90%. System of linear equations in 3D were not examined at all. The DeepMind project used questions in free text format that have not been parsed before being used as input for the model. For Example:

What is  $g(h(f(x)))$ , where  $f(x) = 2x + 3$ ,  $g(x) = 7x - 4$ , and  $h(x) = -5x - 8$ ?

Additionally models without any mathematical knowledge implemented were used.

Other industry approaches are already in use like the photomath Mobile App (photomath, 2020). These programs reduce some of the complexity by allowing only non-free-text examples like a well formatted equation or a SLE. But on the other hand they add some complexity by allowing users to scan and crop handwritten or printed mathematical tasks.


$$\begin{array}{rcl} 4x - 3y - 5z & = & 9 \\ 2x + 5y - 9z & = & 11 \\ 6x - 11y - z & = & 7 \end{array}$$

Officially released accuracy scores for those apps are not available but self-performed test show scores of nearly 100%. Most probably these apps parse the given information to lines and even single mathematical symbols and solve it with the given mathematical tools. So deep learning is only used for image recognition and parsing.

This Project tries to combine both worlds and implement a program that allows to input a scanned pre-cropped image of either handwritten or printed systems of linear equations and retain the solution by parsing each single symbol from the equation, passing it to an image classifier and calculating the solution with the help of the inverse matrix.

## Data

As a first approach in reducing complexity we restrict these equations to the common mathematical variable identifiers x, y and z as well as integer numbers consisting of the literals 0, 1, ...9 as operands and -, + and = as operators.

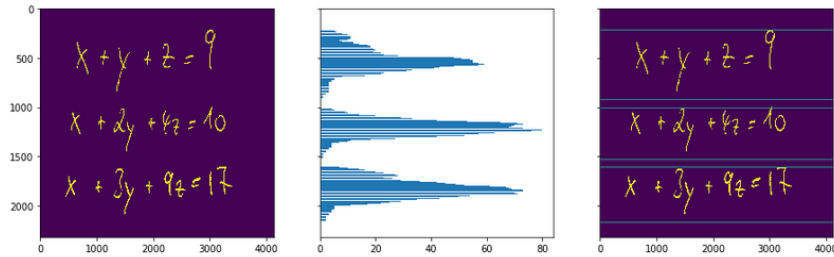
Either a manually or synthetically created dataset of images representing scanned systems of linear equations are needed for evaluation of the part that parses images to the single mathematical symbols. For identifying the symbols itself a dataset consisting of images of the above mentioned symbols is needed.

In this project a dataset provided by Xai Nano on Kaggle is used. It consists of images for 82 different handwritten symbols. So only a subset is used. The given data can be used to create a set of synthetically created images of systems of linear equations used in the image parsing task. Additionally, the single images are used to train the deep learning image classification model.

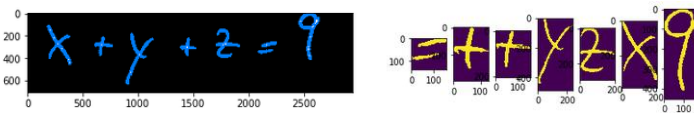
## Solution

The solution to the given problem consists of two major tasks.

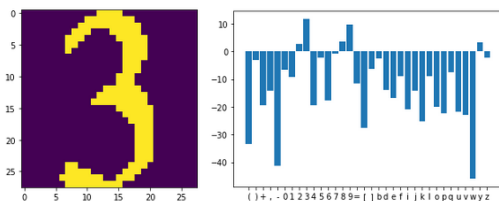
The first part handles the user images provided by the user. They have to be parsed to the single lines first. This can be done by using the mean pixel density in one dimension.



Single mathematical symbols are then parsed by using computer vision structures known as contours. Contours are lines indicating the boundaries of areas of same pixel intensity.



The deep learning image recognition model forms the basis of the second task. It has to be trained on a set of preprocessed images of single mathematical symbols and evaluated on a holdout set. It will then be fed with the preprocessed image snippets and outputs a digital representation of the given symbols. This data is used to create a matrix (A) representation of the SLE.



In a last step the solution (x) of the SLE is calculated by using the mathematical relationship:

$$A \times x = b$$

$$x = A^{-1} \times b$$

The inverse matrix  $A^{-1}$  given A can be calculated with the help of the Gauss-Jordan method.

The aim of this project is to establish a local program for example a command line tool that accepts an image of a SLE and outputs the solution.

## Evaluation

Given the two-part layout of the program, evaluation must be split too.

The user-uploaded image segmentation task can be evaluated with the help of the synthetically created dataset of SLEs. The difference of number of lines and symbols parsed and given by the test image can be used as a metric for evaluation.

The deep learning model evaluation is based on the accuracy score of right classified images. A benchmark for the used image subset could not be found publicly so the matrices derived from different deep learning models trained on the MNIST dataset (LeCun et al.) serve as a benchmark for the single symbol image classifier.

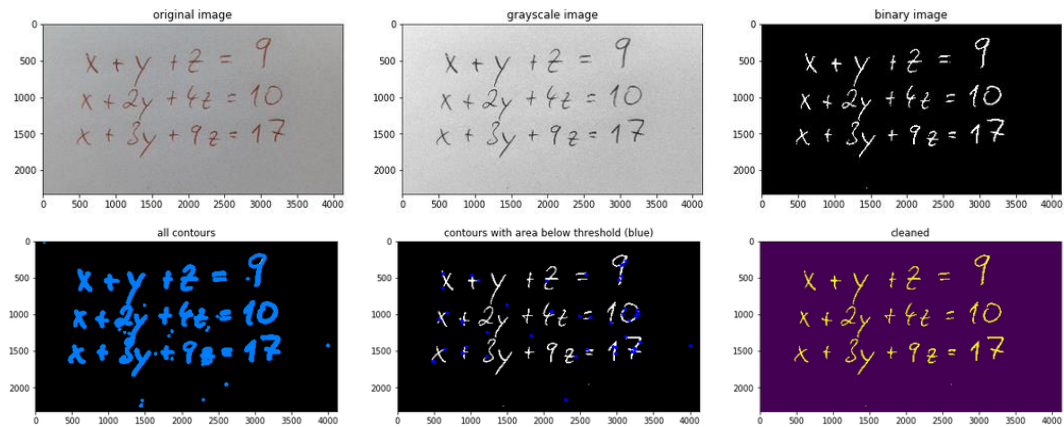
Additionally, an over-all evaluation can be performed by calculating accuracy on providing the right solution for the scanned SLE.

## Design

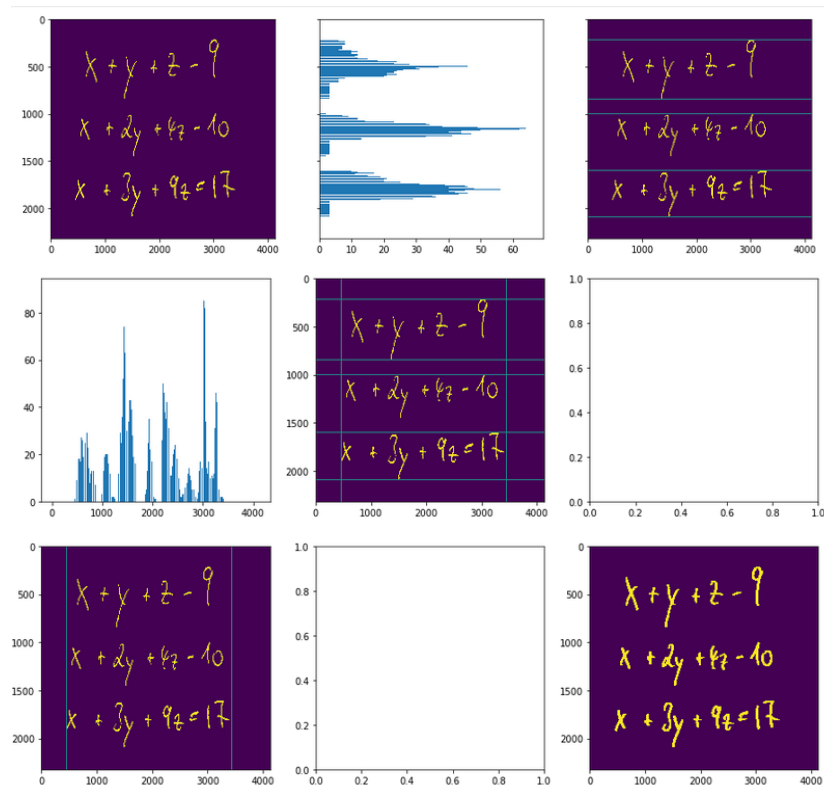
To clarify the architecture of this project a standard flow of data is shown through the program.

1. User scans image of SLE (for evaluation purpose a synthetic image is generated from the math symbol dataset) and passes it as an argument to a Python command line tool via argparse.  

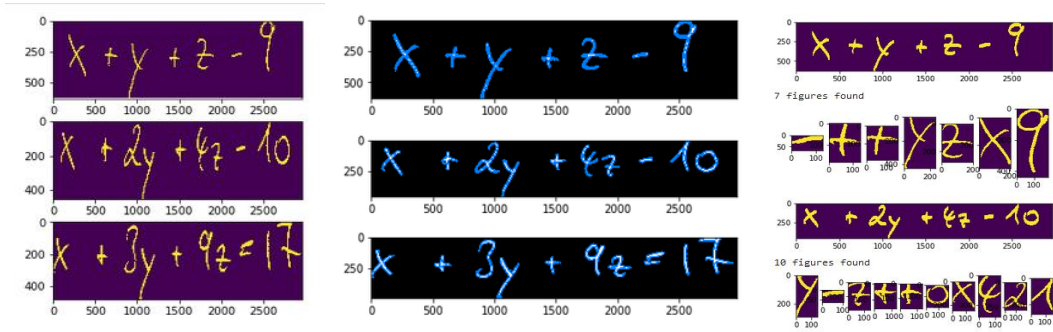
```
>>> python firstprogram.py --path_to_scanned_image
```
2. Image is preprocessed to reduce noise and prepare for follow up steps with the help of the popular Open CV and Pillow libraries in Python. In a first step the image is converted to a single-channel binary image. Only pixels with a value above a given threshold are used, otherwise set to an empty value to reduce noise in the image. In the next step contours (lines indicating the boundaries of areas of same pixel intensity) are generated representing the different shapes in the image. Shapes with an area below a given threshold are discarded because they most probably are only noise too.



3. One image per equation is parsed from the cleaned image by computing the upper/lower and left/right boundaries with help of the pixel density in either dimension. For example, an upper border is a row with a mean pixel density below a certain threshold with a follow up row with a mean above this threshold.

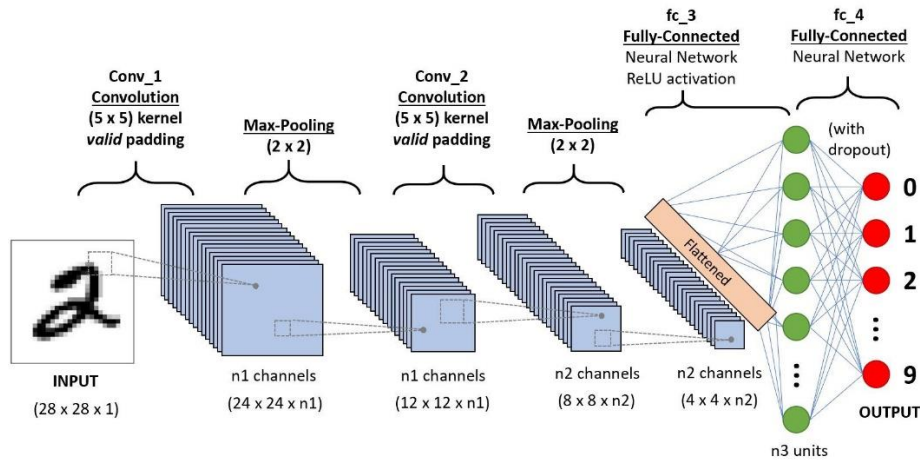


4. Each of these images is parsed to retrieve one image per mathematical symbol with the help of contours again.

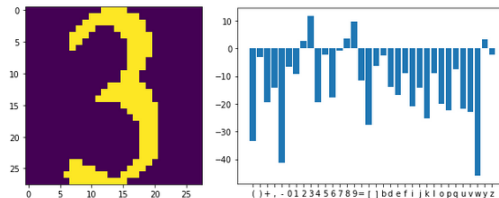


All steps up to this one do not rely on a classical machine learning approach using a model. Thresholds used were set manually to fit a broad range of input images. The quality of the preprocessing can be evaluated with the help of the synthetic SLE images set. For these images the number of lines and symbols in each line are known and can be compared to the outcome of step 4. For example parsing the equal sign may pose a challenge because both '=' are found as separate contours completely detached from each other. Dilation is used extending the shape until both parts of the equal sign overlap and can be parsed as one contour.

The following steps rely on a deep learning model most probably consisting of a Pytorch Convolution Neural net as these models are providing high scores in image recognition especially in models classifying the MNIST handwritten image dataset that has features similar to the Kaggle dataset used in this project. The model is trained on data in the Kaggle data subset and evaluated on the accuracy score calculated with the help of an holdout set from this data.



- Images are fed to the neural net and the prediction is used as a digital representation of the image. The following example is parsed to the number '3'.



- Now the equations are fully digitalized from the images in the form of the matrix A and the column vector b.

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 9 \\ 10 \\ 17 \end{pmatrix}$$

- The inverse matrix of A is calculated with the help of numpy and used in  $x = A^{-1} \times b$  to retrieve the solution of the SLE. If the SLE has a unique solution this information will be outputted to the user otherwise the user will be prompted to rescan and crop the image.

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{pmatrix}^{-1} \times \begin{pmatrix} 9 \\ 10 \\ 17 \end{pmatrix} = \text{solution}$$

#### References:

image of a convolutional neural network: Sumit Saha (<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>)

photomath, <https://photomath.app/en/>, 17.10.2020

Saxton, D. et al. (2019), <https://arxiv.org/abs/1904.01557>

Xai Nano, <https://www.kaggle.com/xainano/handwrittenmathsymbols>, 18.10.2020

LeCun et al., <http://yann.lecun.com/exdb/mnist/>, 17.10.2020