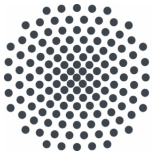

Universität Stuttgart

CNN for plot function detection

Applied Machine Learning Group Project

Philipp Fürst (3383656)
Wasim Essbai (3649361)

Contents

List of Figures

1 Introduction

The aim of this project is to design a machine learning model able to recognize a mathematical function from its graph. This is showed in Fig. 1.

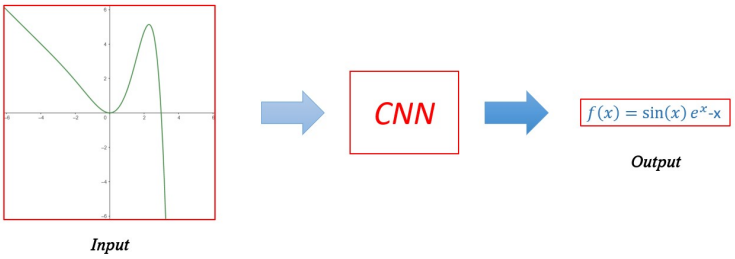

Figure 1: Final result to achieve.

This problem can be approached as an image classification task. A common method for solving this type of problem is to use Convolutional Neural Networks (CNN).

2 Data Generation

For Neural Networks models training a large amount of data is necessary. The first challenge was then to retrieve the data.

For this project it has been developed an algorithm to generate and classify data, since no graph functions datasets available. This algorithm allowed to retrieve an arbitrary large amount of data and it is going to be described in this section.

2.1 Algorithm

The algorithm is based on two main groups that are a list of *basic functions* and list of *binary operators* to combine them. This two lists are showed in table 1.

Basic functions	1	$\exp(x)$	$\sin(x)$	x^2	$\tan(x)$	$\log(x)$	$\lvert x \rvert$	a	x
Basic operators	$a+b$	$a-b$	$a \cdot b$	$a \div b$	a^b				

Table 1: Basic functions and binary operators.

The basic idea is to choose basic functions and combine them with some operators. But this could lead infinite possibilities, for this reason it was necessary to do some design choices.

Rappresentation

The functions are represented as *Computational Graphs*. In figure 2 is given an example of such representation, that refers to Fig. 1.

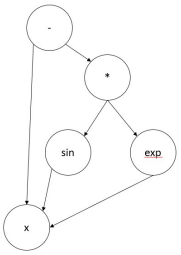

Figure 2: Computational graph example.

Semplifications

The first semplification is referred to the maximum number of children at each node that is two. Then the result is a binary graph. This does not limit the possible functions that can be generated.

The second semplification is the graph depth, that is kept less or equal than three. The depth is directly proportional to the function's complexity. Here are some examples for different complexity levels.

Complexity level 1	Complexity level 2	Complexity level 3
$f(x) = sin(x) + x$	$f(x) = tan(sin(x) + x)$	$f(x) = tan(sin(x) + x)x^2$
$f(x) = log(sin(x))$	$f(x) = log(sin(x)x^2)$	$f(x) = \frac{log(asin(x))}{x}$
$f(x) = \frac{e^x}{x^2}$	$f(x) = \lvert\frac{e^x}{sin(x)}\rvert$	$f(x) = \lvert\frac{e^x}{x} - \frac{a}{x}\rvert$

Generation

The generation is made with different level of complexity to guarantee a fair distribution in the dataset. More precisely, 10% is depth 1, 30% depth 2 and 60% depth 3. The precentage is proportional to the number of possible resulting functions.

Two datasets are generated, a reduced one with 2000 samples to do preliminary training and a larger one with 50000 samples.

Adjustments

Some more adjustments were made to the algorithm in order to get better datasets. The first problem was the huge presence of constant functions, e.g. a function like "$f(x) = sin(x) - sin(x)$" is just a 0. Therefore, this kind of situations are managed.

Another problem was given by equivalent representations, e.g. "$f(x) = x + x$" and "$f(x) = 2x$" are the same functions but with different representation and this could lead to confusion in the model. This is not deleted at all but strongly mitigated.

2.2 Plots

The plots form the input to the Convolutional Neural Network. The mathematical functions are always plotted in the same frames. In particular, they are evaluated in the interval $x \in [-8, 8]$, the y-axis is automatically scaled to the corresponding function

values, so that relevant data is not cut off. Each plot is saved as a fixed size image (250 x 100 pixel) and read in by the CNN. The axes, as well as the grid, are removed within the plot because they are scaled differently depending on the function. Their removal creates a uniform basis and reduces the dimension of the input space, since the CNN does not have to learn this additionally. An actual input to the CNN of a function randomly selected from the dataset $f(x) = \sin(x) + |x|$ is shown in Fig. 3.

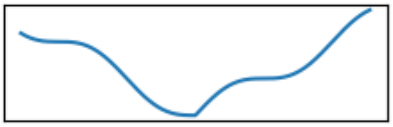

Figure 3: Generated image example.

2.3 Labeling

Another important point is to choose how to label the images. In order to be able to classify mathematical functions unambiguously by means of CNN, it is necessary to label them correctly and without loss of information. The solution chosen is based on the Polish notation. First of all, a numerical value is assigned to each basic function and each basic operator in order to encode the mathematical functions as vectors.

To get a unique encoding, a new operator is introduced and it's the one with code 6. This operator represent the function application, e.g. 6|12|7, that means "application of function 12 to function 7".

With this choices, each vector has a maximum length of 15, which is reached with a full binary graph of depth 3. Functions with less entries are stretched to the length of 15 by simply adding a padding. This is given by sequences of 6 and 7 towards the end of the vector, where 7 is the code for "identity function". This does not change the actual function, however, thereby it is reached that all vectors posses 15 entries. The following is an example of such a labeling.

$$\begin{gathered} f(x) = \sin(x) + |x| \\ \Downarrow \\ x, 1, x, 1, x, 1, x, 1, x, 1, x, 1, +, sin(x), |x| \\ \Downarrow \\ (13, 9, 0, 7, 6, 7, 6, 7, 6, 7, 6, 7, 6, 7, 6) \end{gathered}$$

However, from the data generation it was noticed that the maximum length could be never achieved, adding some useless cells. For this reason the output dimension is detected dynamically during the generation and saved in file at the end.

3 CNN model

3.1 Architecture

As anticipated at the beginning, a CNN is used. The architecture of the network is showed in figure 4.

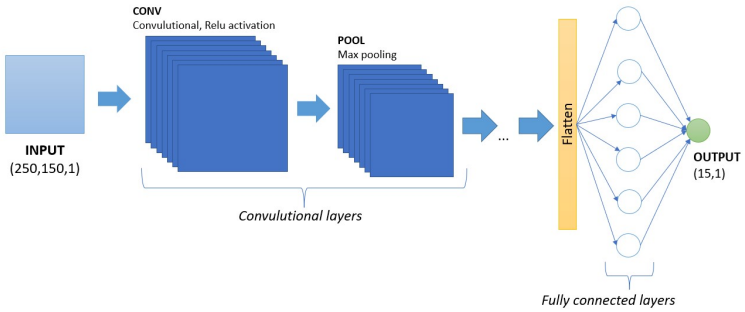

Figure 4: CNN's basic architecture.

There is a first convulutional part, formed by three sets of $CON \Rightarrow RELU \Rightarrow MaxPool$. The second part is the classification, given by three fully connected layers activated by RELU.

3.2 Implementation

The implementation of this CNN is made in Python using the framework Pytorch. The definition is showed in the following snippet code:

```
1         self.network = Sequential(
2                 Conv2d(numChannels, 16, kernel_size=2, padding=1),
3                 ReLU(),
4                 MaxPool2d(2, 2),
5                 Conv2d(16, 32, kernel_size=2, padding=1),
6                 ReLU(),
7                 MaxPool2d(2, 2),
8                 Conv2d(32, 64, kernel_size=2, padding=0),
9                 ReLU(),
10                MaxPool2d(2, 2),
11
12                Flatten(),
13                Linear(23808, 1012),
14                ReLU(),
15                Linear(1012, 512),
16                ReLU(),
17                Linear(512, output_size)
18        )
```

4 Training

First we created a smaller data set consisting of 2000 function plots and the corresponding function vectors. This was again split into 70% training data, 15% validation data and 15% test data. Based on this we tested different sizes of kernels, epochs and learning rates, as there the execution time remained manageable.

We achieved the highest performance with 2x2 kernels, padding of the first two convolutional layers, 15 epochs, 0.1% learning rate, and a scaling factor of 1000000, which further separates the closely spaced numbers of vector entries, allowing easier learning of the entries.

With these parameters found, we could train the CNN on our large dataset (50000). The execution of the training was done GPU-based on Google Colab.

5 Experimental Results

learning curve of loss over epochs and accuracy over epochs, for full vector length and not full vector length (both on the big data set)

6 Conclusions

Our general idea works. It is possible to recognize functions from a given plot and to classify their mathematical function using our trained CNN. The performance is still improvable.

Among other things, this is due to the fact that there are redundancies within our data sets, which are currently not filtered out. For example, one and the same function can be generated by different mathematical functions $f(x) = |x^2| = x^2$. Eliminating these would be a first step to have a more meaningful dataset.

Additionally, due to lack of computer resources, we trained our CNN on only 50000 samples. This is very small in the dimension of machine learning and therefore still expandable. However, the approach is promising if already with these small data sets accuracies of X.XX% are achieved when comparing all but one vector entries of the mathematical function.