Universität Stuttgart

CNN for plot function detection

Applied Machine Learning Group Project

Philipp Fürst (3383656)
Wasim Essbai (3649361)

Contents

List of Figures

List of Tables

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.

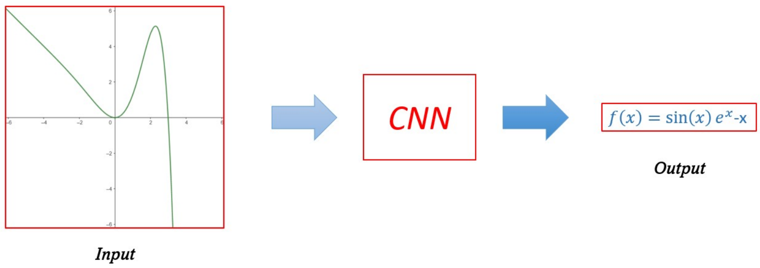

Figure 1: Final result to achieve

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

Neural Networks models need a large amount of training data. The first challenge was to retrieve that data.
For this project we developed an algorithm to generate and classify data, since no graph function datasets were available. This algorithm allowed to retrieve an arbitrary large amount of data. It is described in the following section.

2.1 Algorithm

The algorithm is based on two main groups. A list of *basic functions* and a list of *binary operators* to combine them. They are shown in table 1.

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

Table 1: Basic functions and binary operators

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

Representation
The functions are represented as *Computational Graphs*. Fig. 2 gives an example of such a

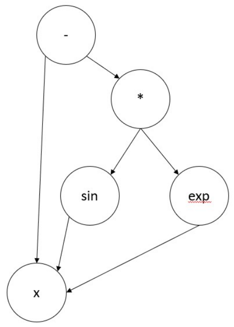

Figure 2: Computational graph example

representation. It refers to the function $f(x) = \sin(x) \cdot e^x - x$ of Fig. 1.

Simplifications

The first simplification is referred to the maximum number of children at each node, which is two. The result is a binary graph.

The second simplification is the graph depth. This limits the possible number of functions that can be generated. It is kept less or equal to 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
$f(x) = \sin(x) + x$	$f(x) = \tan(\sin(x) + x)$	$f(x) = \tan(\sin(x) + x)x^2$
$f(x) = \log(\sin(x))$	$f(x) = \log(\sin(x)x^2)$	$f(x) = \frac{\log(\sin(x))}{x}$
$f(x) = \frac{e^x}{x^2}$	$f(x) = \lvert\frac{e^x}{\sin(x)}\rvert$	$f(x) = \lvert\frac{e^x}{x} - \frac{a}{x}\rvert$

Table 2: Complexity Levels

Generation

The generation is made with different levels 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 further 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 konstant. These simple solutions were filtered out to reduce the redundancy of our dataset.

An different problem was given by equivalent representations, e.g. $f(x) = x + x$ and $f(x) = 2x$. They are the same functions with different representation. This could lead to confusion in the model. We were not able to deleted all these combinations but we strongly mitigated them.

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.

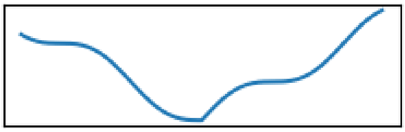

Figure 3: Generated image example

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. "$6, 12, 9$" means "application of function 12 to function 9". Here 12 stands for $\log(x)$ and 9 for $\sin(x)$, so it translates into "$\log(\sin(x))$".

$a+b$	$a-b$	$\frac{a}{b}$	$a \cdot b$	a^b	a	x	1	$\exp(x)$	$\sin(x)$	x^2	$\tan(x)$	$\log(x)$	$\lvert x \rvert$
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Table 3: Labeling of vector elements

With this choices, each vector has a maximum length of 15, which is reached with a full binary graph of depth 3. Each entry of the vector can have a number between 0 and 13 corresponing to Tabel 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. Thereby it is reached that all vectors posses 15 entries. The following is an example of such a labeling.

$$f(x) = \sin(x) + |x|$$
$$\Downarrow$$
$$x, 1, x, 1, x, 1, x, 1, x, 1, x, 1, +, \sin(x), |x|$$
$$\Downarrow$$
$$(13, 9, 0, 7, 6, 7, 6, 7, 6, 7, 6, 7, 6, 7, 6)$$

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

3 CNN model

3.1 Architecture

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

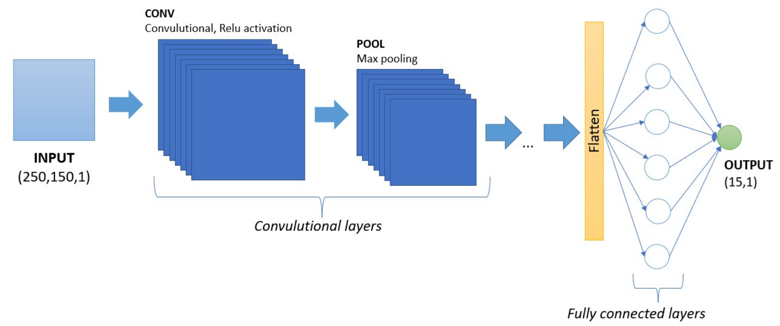

Figure 4: CNN's basic architecture.

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 code.

```
self.network = Sequential(
        Conv2d(numChannels, 16, kernel_size=2, padding=1),
        ReLU(),
        MaxPool2d(2, 2),
        Conv2d(16, 32, kernel_size=2, padding=1),
        ReLU(),
        MaxPool2d(2, 2),
        Conv2d(32, 64, kernel_size=2, padding=0),
        ReLU(),
        MaxPool2d(2, 2),

        Flatten(),
        Linear(23808, 1012),
        ReLU(),
        Linear(1012, 512),
        ReLU(),
        Linear(512, output_size)
)
```

4 Data Processing & Training

4.1 Data Processing

Before training the model, a simple pre-processing phase is done. This consists in the conversion of images from 4-channel to grey scale. The main reason is make the training process faster.

Furthermore, the labels are converted from integer values to float, for compatiblity reasons. The labels are also scaled by a factor of 1000000, to get more margin between each code.

4.2 Training

The training is done by splitting the dataset into 70% for train set, 15% validation and 15% test. The training execution was GPU-based on Google Colab.

A first training was done on the smaller data set consisting of 2000 images, to get a first look on the perfomances and change the model parameters to achieve the best performances with the model used. After this preliminary tunings it was used the full dataset of 50000 samples. The loss for each epoch is showed in figure 5.

(a) Training loss vs epochs

(b) Validation loss vs epochs

Figure 5: Loss vs epochs

The validation loss was used to tune the hyperparameters, that are learning rate, number of epochs and batch size. The best emerged values are learning rate 0.001, 10 epochs and batch size 500.

5 Experimental Results

From the obtained model, the accuracy is 0.3% for both training and validation data. The perfomances are very poor, but doing more analysis it can be seen that the performances increases a lot comparing sub-vectors. More precisely, the comparison is made for all components except the first n_shift entries, that could be considered less relevant, since more deeper in the computational graph.

This was done on test data, with output size 7, that were never used before, obtaining the values in table 4.

n_{shift}	Accuracy
0	0.4%
2	25.89%
4	86.41%

Table 4: Accuracy vs shift comparison

It's possible to see that the network learned how to classify images except the first two entries.

After a review of the work done, two main candidate reasons came out:

1. Image representation: the main problem is the presence of different ways to express the same mathematical function that are not managed yet, e.g. $f(x) = \log(e^x)$ and $f(x) = x$ are equivalent but with different encoding.

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.