Homework Marking System

Arthur LANG
School of Engineering
Jönköping University
Jönköping, Sweden
laar20vx@student.ju.se

Madalina Aldea
School of Engineering
Jönköping University
Jönköping, Sweden
alma20px@student.ju.se

***Abstract*—** Homework is an important part of the learning process. It helps the student to better fix knowledges. But if homework gives work to students, there is also a certain amount of work for the teachers, who have to correct all the submissions. This represents 3 hours of work every week on average [1]. We think this time can be reduced by automated the process of correction. This paper proposes a method for automatically correct homework using AI. For simplification reasons, we decided to focus on submission in primary schools.

Keywords*— *Homework marking, Machine learning, Computer vision.

I Introduction

An AI cannot completely replace a teacher, many of the attributes that make good teachers are the very things that AI or other technology fails to emulate: inspiring students, building positive school and class climates, resolving conflicts, creating connection and belonging, seeing the world from the perspective of individual students, and mentoring and coaching students [1].

But teachers spend time on a subject that can be optimized, as the homework evaluation. This time gain is estimated a bit less than 50% with automation of the correction [1].

II The problem

The risk when you talk about homework correction is the diversity of the format, subjects, and exercises. We had to decide a scope for our project and focus on a simple exercise. The first thing that comes to mind was the math. Indeed, a math exercise is, most of the time, true or false. It removes all the interpretation parts that could be present in a literature exercise. Moreover, we wanted to focus only on the result part for the moment. It could be looking oversimplified, but our main concern was the difficulty to recognize children's handwriting that is more various and unsure than adults' one.

That why our focus with this project was the children's handwritten recognition through math assignment correction.

We took a certain amount of time to define a clear workflow that will be followed by the teacher. We had several problems: first, we needed to have a version of the homework with the correct answers, to be able to correct the rest. Secondly, we do not want the teacher to select for every children’s submission the area of each answer because this approach would not substantially reduce the workload. On the other hand, we were limited in terms of time to add the search of the answer by the algorithm.

We finally agreed on the following approach: the teacher hand in a paper with the correct answers and uses the program to mark the position of each answer (we call that: Regions Of Interest or ROI). Our idea was to limit the number of data to treat. Despite interpreting the subject and try to guess the answers, the user can directly specify where the answers are positioned. Moreover, when a teacher gives an exercise to students, all the subjects are the same, so the children's answers are in the same area.

The ROIs given by the teacher are crop on each submission. Then, we need to identify the digit in the teacher's paper and the one in the children's work. For that, we will need to train a model for handwritten recognition.

At the end of the program, we wanted to give intuitive feedback to the teacher, by marking with a green box the correct answer and with an orange box the wrong one. We also wanted to output the digit recognize by our algorithm, to be able to check if the recognition went well or not. In the future, this feature can be remove for the correct answer but will be kept for the wrong one.

III Approach

a. Possible datasets

We spent a lot of time searching for a good dataset, but unfortunately, we were not able to find one that is perfectly suited for our problem. That is why we first used the MNIST dataset. It is the “hello world” of handwriting recognition, a dataset composed of 70.000 (60.000 for training and 10.000 for the test) pictures with digits between 0 and 9.

We also found another dataset, an extension of MNIST, that expend pictures from 0 to 99. This dataset allows us to go around the problem of numbers. With only MNIST, you have to split each number in digits. For example, 12 have to be split into 1 and 2.

Finally, we get in touch with a teacher in primary school who helped us by handing students some worksheets on which children wrote digits and solved equations. On the first task, the children were supposed to write 5 times each number between 0 and 10 in this way we obtained a dataset that was used for fine-tuning the model. In total, we had 340 samples of digits (the worksheets were scanned and the quality for some digits was poor, making them impossible to use). The worksheets needed to be processed in order to be able to use the information, each digit was cropped and labeled accordingly. Moreover, to improve the size of the dataset we created, we did a dataset augmentation, but note that this process was done at the late part of the project.

b. Chosen technologies.

We used python as a language for several reasons: lots of machine learning libraries and tools are implemented in python and this language was the best master by both of us.

In addition, we used the NumPy library, which allows a great and optimized way of manipulating arrays in python.

The model was built using Keras, a deep learning API written in Python, running on top of the machine learning platform TensorFlow[2]. Keras has a user-friendly and intuitive design which makes it easy to use. Also, each model can take advantage of the pre-built functions for compiling, fitting, and evaluating it. Keras also provides methods for saving or loading models.

Since we were manipulating pictures, we used OpenCV. This library is famous for image manipulation and provides a lot of functions needed for our project. It is implemented also in python, which fits with our technical stack and is compatible with NumPy. Note that OpenCV also integrated some machine learning tools.

IV Results

a. Choosing a model

During this project, we tried the following models.

a.1 The first model

```
Model: "sequential"
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
conv2d (Conv2D) (None, 26, 26, 32) 320
_________________________________________________________________
conv2d_1 (Conv2D) (None, 24, 24, 64) 18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 64) 0
_________________________________________________________________
dropout (Dropout) (None, 12, 12, 64) 0
_________________________________________________________________
flatten (Flatten) (None, 9216) 0
_________________________________________________________________
dense (Dense) (None, 128) 1179776
_________________________________________________________________
dropout_1 (Dropout) (None, 128) 0
_________________________________________________________________
dense_1 (Dense) (None, 10) 1290
=================================================================
Total params: 1,199,882 Trainable params: 1,199,882 Non-trainable
params: 0
_________________________________________________________________
```

Fig. 1. Architecture of the first model

To detail a bit the layers of the first model, the convolutional layers and the MaxPool layer are used for feature extraction and down sampling the input by taking the maximum value over a defined window. The dropout layers are used to prevent overfitting and the flattening layer is used for reshaping the data for the dense layers. Dense layers implement the operation:

output = activation(dot(input, kernel) + bias)

where activation is the element-wise activation function passed as the activation argument, the kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer [3] [4].

a.2 The second model

After performing some trials, we decided to update our model, following the structure shown in fig. 2. Extraction of the features and down sampling are done twice using two blocks with convolutional layers and a MaxPooling2D module. Also, two new modules were integrated (BatchNormalization()) for speeding up the training and for avoiding internal covariate shift which appears because of the changes in network parameters during training phase.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 32)        9248
_________________________________________________________________
batch_normalization (BatchNo (None, 24, 24, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 128)       36992
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
batch_normalization_1 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 256)               524544
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 11)                2827
=================================================================
Total params: 722,155
Trainable params: 721,835
Non-trainable params: 320
_________________________________________________________________
```

Fig. 2. Architecture of the second model

b. Evaluation of the models

Regarding the evaluation of the models, we first used the accuracy indicator to know about the efficiency of our models. Accuracy is quite a good indicator in our case because the MNIST dataset is balanced. We also did some tests from our own writing and writing samples we collected from children, to allow us to see how the models perform on new data. The data we collected from children was balanced: each child had the same subject and write the same number of digits. Even if we skipped few digits, less than ten, from our collection, it represents only 0.02% of the total dataset. It is too few to unbalance the dataset.

Model / result	Accuracy
Model 1	99 %
Model 2	99.4 %

Fig. 3. Accuracy got from models after training

However, this method had a problem: the accuracy is computed on the data the model was trained on. We confront this problem at the beginning, though model 1 has good accuracy, it seems to be less efficient than model 2 to recognize new data. Unfortunately, we had no proper results about this last fact. That why come the motivation for a new evaluation method, where we would like to know how models interact with new data.

We split this task into two parts: evaluation of the model and evaluation of the dataset.

The first evaluation consists of training the two models on MNIST. Then, we test the models using the crossvalidation method on the dataset composed of children digits and end with a global confusion matrix for all the different folds.

The result of this evaluation is in the Fig. 4.

Model	Confusion Matrix
Model 1	[**31** 5 1 0 1 0 0 0 1 0] [0 **31** 1 0 0 0 1 0 0 0] [2 1 **22** 0 0 0 0 0 3 0] [0 2 0 **26** 0 1 0 1 0 1] [0 3 0 0 **26** 0 1 0 0 0] [1 1 0 0 0 **28** 0 0 0 0] [7 0 0 0 0 4 **20** 0 0 0] [0 0 0 0 0 1 0 **24** 0 0] [2 0 1 2 0 0 0 0 **28** 0] [2 0 0 5 1 1 0 1 0 **21**]
Model 2	[**35** 0 1 0 0 0 0 0 1 2 0] [0 **32** 0 0 0 0 0 0 0 0 1] [0 1 **25** 0 0 0 1 0 1 0 0] [0 3 0 **28** 0 0 0 0 0 0 0] [0 3 0 0 **25** 0 1 1 0 0 0] [0 2 0 0 0 **27** 1 0 0 0 0] [5 0 0 0 0 0 **25** 0 0 0 1] [0 2 0 1 3 0 0 **17** 2 0 0] [0 0 0 0 0 0 3 0 **30** 0 0] [1 1 1 0 0 1 0 0 2 **25** 0] [0 1 0 0 0 0 0 0 0 0 **28**]

Fig. 4. Result from first evaluation

We can then deduce that model 2 seems to be more efficient, especially on a small dataset.

The second evaluation was done by variating the training dataset. We train model 1 and 2 with two different datasets: MNIST (with 10 labels) and a mix of MNIST and augmented images from the dataset composed of children images. We drew a confusion matrix of prediction on only children's pictures (new for the algorithm in all cases).

Note that the dataset contained 340 images. The result is below, in the fig. 5.

As we can see, the first model trained with the MNIST dataset had some strange behaviors. We did not have the time to investigate too much. This model correctly classified only 30 images from a total of 340.

The first model trained on the MNIST dataset enriched with an augmented version of children’s dataset and the performance was better, the model correctly classified 289 digits from a total of 340.

Model 2 trained only on MNIST dataset performed better than Model 1. It correctly classified 242 images from a total of 340.

Model 2 trained on the MNIST dataset and an augmented version of children dataset was the best performer, with a correct classification of 297 images from 340.

c. *Utilities around*

Having a functional model was not our only goal, but we wanted to have a minimal program that can be used by a teacher or easily integrated with a Graphical User Interface.

Model + dataset	Confusion Matrix
Model 1 + Mnist	[**0** 0 0 0 0 0 0 0 0 39 0] [0 **0** 0 0 0 0 0 0 0 33 0] [0 0 **0** 0 0 0 0 1 0 27 0] [0 0 0 **0** 0 0 0 0 0 31 0] [0 0 0 0 **0** 0 0 0 0 30 0] [0 0 0 0 0 **0** 0 0 0 30 0] [0 0 0 0 0 0 **0** 0 0 31 0] [0 0 0 0 0 0 0 **0** 0 25 0] [0 0 0 0 0 0 0 0 **0** 33 0] [0 0 0 0 0 0 0 1 0 **30** 0] [0 0 0 0 0 0 0 0 0 29 **0**]
Model 1 + Mnist and augmented children	[**26** 0 0 0 1 0 0 0 0 12 0] [0 **29** 0 0 3 0 0 1 0 0 0] [0 0 **25** 0 0 0 0 0 3 0 0] [0 0 1 **29** 0 1 0 0 0 0 0] [1 1 0 0 **25** 0 1 2 0 0 0] [0 0 0 0 0 **30** 0 0 0 0 0] [2 0 0 0 0 0 **25** 0 2 2 0] [0 0 0 4 0 0 0 **17** 1 0 3] [1 0 1 0 0 0 0 0 **31** 0 0] [1 0 1 1 0 0 1 1 3 **23** 0] [0 0 0 0 0 0 0 0 0 0 **29**]
Model 2 + Mnist	[**19** 0 0 0 0 0 1 0 0 19 0] [0 **29** 0 0 1 0 1 1 0 0 1] [0 0 **25** 0 1 0 0 0 2 0 0] [0 4 0 **26** 0 1 0 0 0 0 0] [0 0 0 0 **24** 0 1 4 0 1 0] [0 0 1 0 0 **22** 0 2 0 5 0] [2 0 0 0 0 0 **24** 0 1 4 0] [0 1 1 4 2 0 0 **16** 1 0 0] [0 0 0 0 0 2 2 0 **29** 0 0] [0 2 2 1 0 1 0 0 1 **24** 0] [2 3 0 0 5 0 12 0 3 0 **4**]
Model 2 + augmented children	[**26** 0 0 0 1 0 0 0 0 12 0] [0 **29** 0 0 3 0 0 1 0 0 0] [0 0 **25** 0 0 0 0 0 3 0 0] [0 0 1 **29** 0 1 0 0 0 0 0] [1 1 0 0 **25** 0 1 2 0 0 0] [0 0 0 0 0 **30** 0 0 0 0 0] [2 0 0 0 0 0 **25** 0 2 2 0] [0 0 0 4 0 0 0 **17** 1 0 3] [1 0 1 0 0 0 0 0 **31** 0 0] [1 0 1 1 0 0 1 1 3 **23** 0] [0 0 0 0 0 0 0 0 0 0 **29**]

Fig. 5. Result from second evaluation

The system architecture can be found in fig.6. HomeworkMarking is the main object that links all the processes. It is inspired by the Mediator Design Pattern [6], but still contains some logic, to avoid the multiplication of objects.

ROIDetection is the object that manages all the ROIs. Note that an ROI is defined as an array of 4 integers, representing the coordinates x and y of the upper left point and bottom right point of the ROI, which is a rectangle.

The fig.7 is an example of the output given by the program.

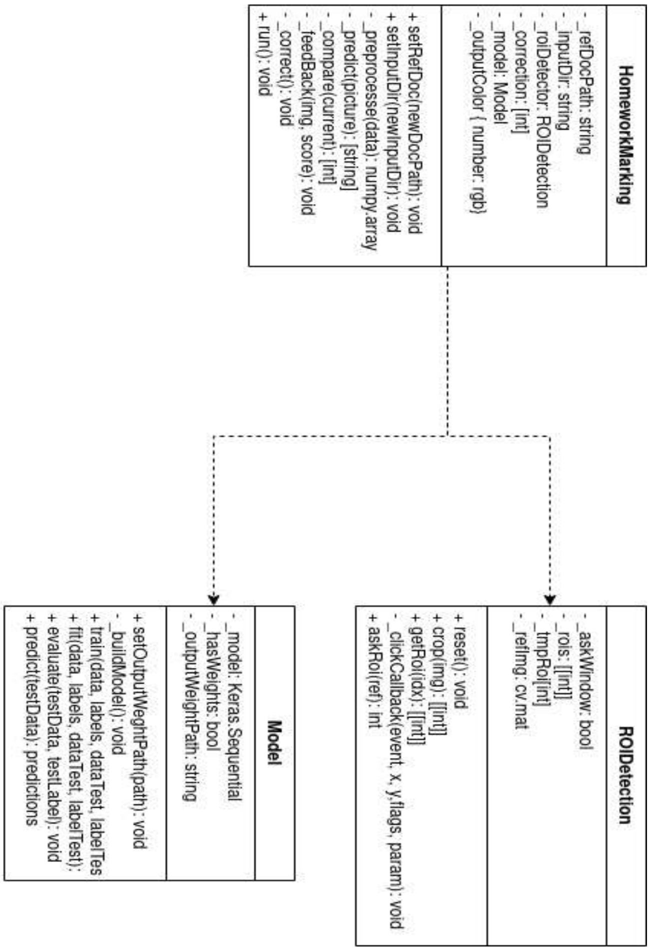
HomeworkMarking
- _refDocPath: string
- _inputDir: string
- _roiDetector: ROIDetection
- _correction: [int]
- _model: Model
- _outputColor (number: rgb)
+ setRefDoc(newDocPath): void
+ setInputDir(newInputDir): void
- _preprocesse(data): numpy.array
- _predict(picture): [string]
- _compare(current): [int]
- _feedBack(img, score): void
- _correct(): void
+ run(): void
ROIDetection
- _askWindow: bool
- _rois: [[int]]
- _tmpRoi[int]
- _refImg: cv.mat
+ reset(): void
+ crop(img): [[int]]
+ getRoi(idx): [[int]]
- _clickCallback(event, x, y,flags, param): void
+ askRoi(ref): int
Model
- _model: Keras.Sequential
- _hasWeights: bool
- _outputWeightPath: string
+ setOutputWeightPath(path): void
- _buildModel(): void
+ train(data, labels, dataTest, labelTes
+ fit(data, labels, dataTest, labelTest):
+ evaluate(testData, testLabel): void
+ predict(testData): predictions

Fig. 6. System architecture

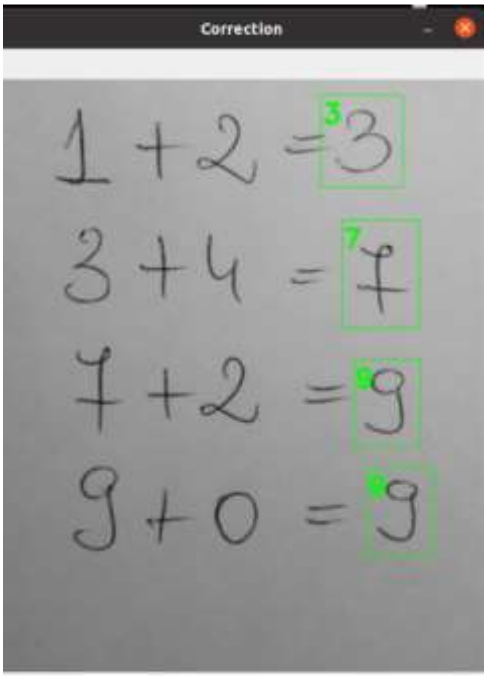

Fig. 7. Sample of correction

V. Discussion

During our work, we noticed that training the models only on MNIST gives poor results on children datasets. The answer could be the difference between digits from MNIST and the pictures we had from children. For example, the MNIST dataset only has centered digits, that was not our case. Moreover, children's writings are more unsure, weirdly formed.

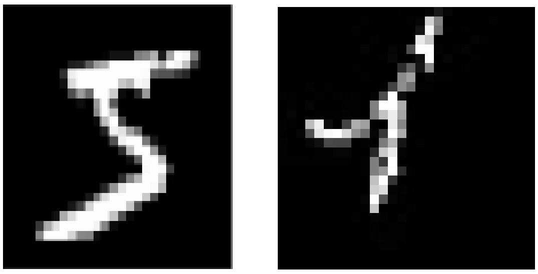

Fig. 8. Sample of digits : left - mnist sample, right - sample from digits written by children

Another problem we were aware of, is the case of numbers like 12, 43 or 938. Our first approach was to split a number like 12 in digits: 1 and 2. Then we could identify each digit and merge the result at the end. But after several research and trials, we were not able to separate them perfectly. Our main problem was when the digits were too close, when they "touch" each other or when the shape of the digit is not perfectly defined. To avoid the separation in digits, we found a dataset, a sort of augmented MNIST[5] to generate all the digits between 0 and 99. Unfortunately, we were not able to continue in this way, due to a lack of time and data from children.

Our model is not perfect, sometimes it mismatches digits. Some reasons for this result could not be the model but the custom dataset processing for the following reasons: the cropped digit is not always centered as we talked about above, not like the MNIST dataset, and the cropped digit must be resized in 28x28, so that could alter the image of the digit.

VI. Disclosure of Contribution

Find bellow a table with all our tasks and how much time we spent on each.

TABLE I. DISCLOSURE OF CONTRIBUTION

Task / Member	Mădălina	Arthur
Research	15h	12h
Dataset work	12h	8h
Building Model	9h	10h
Training Model	20h	17h
Utilities development	15h	20h
Evaluation	6h	4h
Write the report	5h	10h
Documenting	1h	2h
Lab Sessions	4h	4h
Total	87h	87h

VII. CONCLUSION

Classification of children's writing gave us more challenges than expected, and we conclude that a bigger dataset is needed to have a real-life applicable solution. But we are still confident about the result we got, and we think that the bases we put on this project could be easily expanded. Moreover, several facets can be reworked, like different methods of scaling or dataset augmentation.

REFERENCES

[1]https://www.mckinsey.com/industries/public-and-social-sector/our-insights/how-artificial-intelligence-will-impact-k-12-teachers
[2] https://keras.io/about/
[3]Hyper-Parameter Optimization: A Review of Algorithms and Applications Tong Yu, Hong Zhu
[4] https://keras.io/api/layers/core_layers/dense/
[5]https://github.com/shaohua0116/MultiDigitMNIST
[6] https://refactoring.guru/design-patterns/mediator