Report on Photomath assignment

Sanja Deur

January 3, 2022

1 Dataset

- Source: `https://www.kaggle.com/xainano/handwrittenmathsymbols`
- Dataset consists of .jpg images with resolution of 45×45.

Table 1: Distribution of original dataset

Label	**Number of examples**
0	6,914
1	26,520
2	26,141
3	10,909
4	7,396
5	3,545
6	3,118
7	2,909
8	3,068
9	3,737
+	25,112
-	33,997
x	26,594
/	199
(	14,294
)	14,355
Total	208,808

- Since the dataset is not homogeneous (distribution of labels is not uniform), I decided to keep 3,000 examples for every character.
- Code: `scripts/homogenise_data.py`
- Most of the labels have its examples reduced to 3,000, whereas '7' and '/' have some of its examples repeated to reach 3,000.
- The idea is to mostly remove excess examples, and to not repeat too much.
- Dataset is divided into training and test set, by 85% and 15% respectively.

Table 2: Distribution of homogeneous dataset

Label	Training set	Test set	Total
0	2,550	450	3,000
1	2,550	450	3,000
2	2,550	450	3,000
3	2,550	450	3,000
4	2,550	450	3,000
5	2,550	450	3,000
6	2,550	450	3,000
7	2,550	450	3,000
8	2,550	450	3,000
9	2,550	450	3,000
+	2,550	450	3,000
-	2,550	450	3,000
x	2,550	450	3,000
/	2,550	450	3,000
(	2,550	450	3,000
)	2,550	450	3,000
Total	40,800	7,200	48,000

2 Implementation

- Programming is done exclusively in Python.
- The main imported libraries: `tensorflow`, `opencv-python`, `numpy`, `scikit-learn`, and `h5py`.

2.1 Handwritten Character Detector

- Code: `src/detector.py`
- Input: photo of a math expression
- Output: extracted characters in given resolution (in accordance with dataset, 45×45)
- Used library: `https://opencv.org/`
- Characters are extracted only if they are of certain width and height. This is implemented in order to avoid detection of dots or other small noises in photos.
  - In Figure 1 wrong character (wave) is detected, whereas dot is not even considered a character.
  - Wave is later on classified with a very small certainty (37.815%), which indicates it is probably not a valid character.

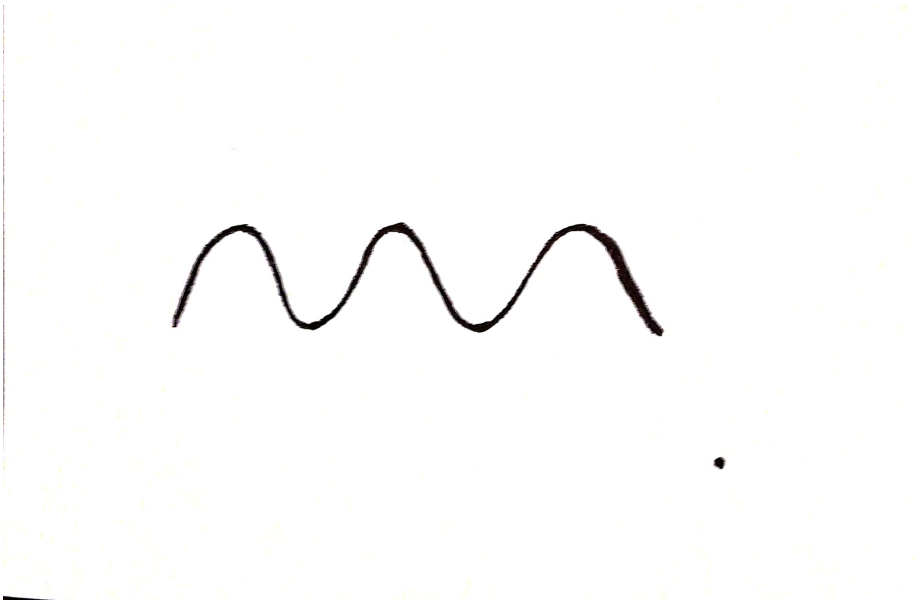

Figure 1: Expression containing invalid character and noise

2.2 Handwritten Character Classifier

- Characters are mapped in the following way: digits are mapped to 0-9, and '+', '-', 'x', '/', '(', and ')' are mapped to 10-15, respectively.
- Photos are converted to numpy arrays of shape (45, 45).
- Values in the array are divided by 255, in order to get values 0-1.

2.2.1 Train

- Code: `scripts/train.py`
- Input: directory containing homogenised characters
- Output: trained model in H5 file format
- Used library: `https://www.tensorflow.org/`
- The model is consisted of three convolutional layers and 2 max pooling layers as can be seen in Figure 2.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 43, 43, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 21, 21, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 19, 19, 64)        18496
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 17, 17, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 100)               409700
_________________________________________________________________
dense_1 (Dense)              (None, 16)                1616
=================================================================
Total params: 467,060
Trainable params: 467,060
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: CNN model summary

- Stochastic gradient descent optimisation is applied, with learning rate equal to 0.01, and momentum equal to 0.9.
- Categorical cross entropy is utilised as loss function.
- Model is trained for 10 epochs, with batch size of 32.
- Trained model is saved to a H5 file, so it can be easily loaded for testing and prediction later on.

2.2.2 Test

- Code: `scripts/test.py`
- Input: directory with extracted characters and trained model
- Output: accuracy achieved on the test set
- Accuracy: 98.306%

2.2.3 Classify

- Code: `src/classifier.py`
- Input: directory with extracted characters and trained model
- Output: math expression in string format
- Program goes through all of the extracted characters and predicts the label with certain accuracy. If the accuracy is below 50%, a warning is printed about uncertainty of that prediction.

2.3 Solver

- Code: `src/solver.py`
- Input: math expression in string format
- Output: final result
- Program operates in the following steps:
  1. Expression is normalised (e.g. '+ 0 3' to ['3'], and '- (3 - 2)' to ['-1', '*', '(', '3', '-', '2', ')'])
  2. Expression is validated, i.e. it is made sure that expression is in valid infix notation.
  3. The shunting-yard algorithm is used to build abstract syntax tree (AST) in order to parse math expressions specified in infix notation.
  4. AST is evaluated taking care of operators' precedence and associativity.
- Program is thoroughly tested: `test/test_detector.py`.

2.4 Photomath Application

- Code: `src/photomath.py`
- Input: photo of a math expression
- Output: final result
- Photomath application calls functions `detect`, `classify`, and `solve` in order to obtain final result.
- The application prints appropriate messages and intermediate results when necessary.
- The application also takes care of reporting error messages, such as invalid path or division by zero.
- Options are shown in Figure 3.

Photomath

```
python src/photomath.py -p <photo_path> [options ...]

  options:
    -p, --photo_path <str>
      required
      path to the photo of a math expression
    --width <int>
      default: 45
      width of the photos in the dataset
    --height <int>
      default: 45
      height of the photos in the dataset
    --dest <str>
      default: 'resources/extracted_characters'
      path to the directory where extracted characters will be stored
    --model_path <str>
      default: 'final_model.h5'
      path of the trained model
    --verbose <int>
      default: 1
      choose 1 for displaying the prediction information, 0 otherwise

NOTE: Please run scripts/train.py first in case there is no saved model.
```

Figure 3: Photomath application options

2.5 Remarks and Future Work

- Even though this program avoids small noises, such as dots and thin lines, there could be possible improvements to avoid even more noise.
- Test accuracy of 98.306% is remarkable, but it can probably be even better when defining a different model or parameters.
- Some characters, such as '1' and '/' are similar, so they sometimes get mixed up by the network. One reason could also be only 299 '/' in the dataset, so there should probably be more examples.
- The program could include even more operators in the future, or even equations.