Image Analysis Final Project

Arthur Bricq
Anne-Aimée Bernard
Jonas Ulbrich

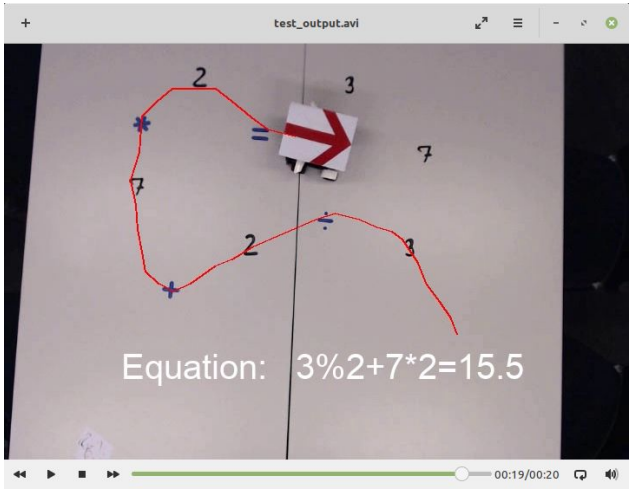
May 29, 2020

École polytechnique fédérale de Lausanne

The main tasks

Overview of the main task

1. Analyse and edit a video in Python
2. Extract the position and the meaning of the objects on the image
   a. Digits
   b. Operators
   c. Robot
3. Robustness against
   a. Rotation
   b. Color
4. Solve an equation as a String
5. Assess the efficiency and robustness of our code



Handling video in Python

A **video** is an array of frames (images), with a given frame rate (FPS).

Free and open-source command line tool for video edition: **ffmpeg**

- **ffmpeg** -i INPUT -f image2 images/image%03d.png
- **ffmpeg** -f image2 *-framerate 2* -i images/image%03d.png OUTPUT

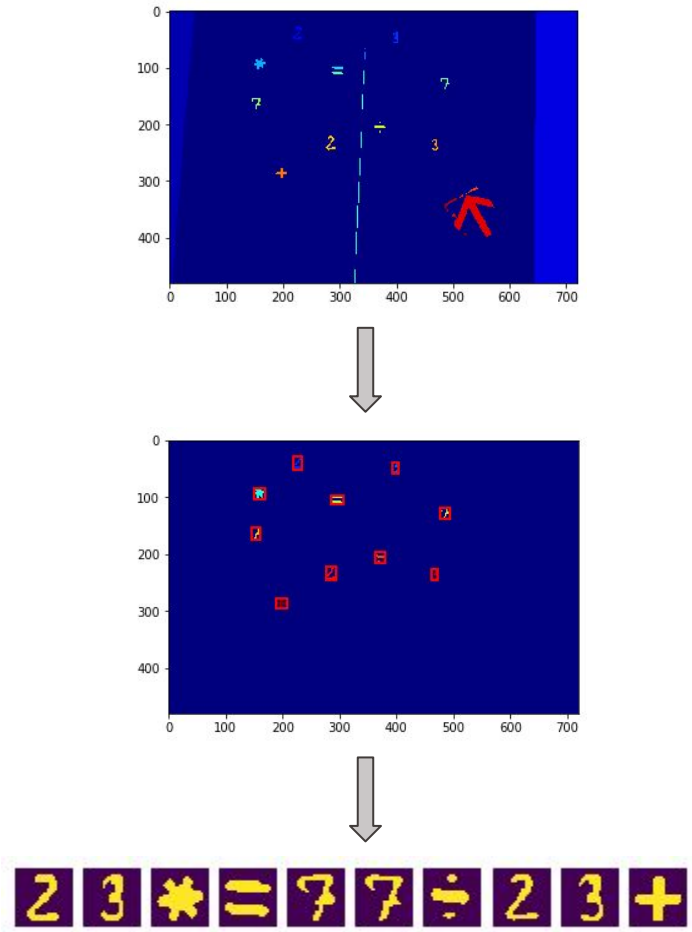
Can be used in Python with package ‘os’

- os.system(“ffmpeg_command”)

Object extraction

From the first image of the video:

- Threshold to remove the background
- **Object labeling** algorithm (*lab01*) applied successively with selection of the obtained regions to remove unwanted objects
- Reshaping of final objects to have definite shape: 28x28 pixels



Object classification

2 classifiers available

Rotation invariant

- Makes use of the colors
- ‘%’ and ‘=’ are classified with number of regions.
- Uses skeleton-based method for classifying operators (counting the number of endings)
- **CNN** (keras) trained with **MNIST** for the numbers

Rotation and color invariant

- Doesn’t use the colors
- ‘%’ and ‘=’ are classified with number of regions.
- **CNN** trained with a larger dataset: **MNIST** with operators ‘+’ from the **CROHME** dataset (we didn’t included ‘-’ to avoid confusion with 1)
- Heuristic rules are used to detect operators (*really low certitude means it’s an operator*)

Rotation invariant CNN

About our CNN:

- Done with Keras
- Trained with 5 epochs
- Data-augmentation
- Class balancing

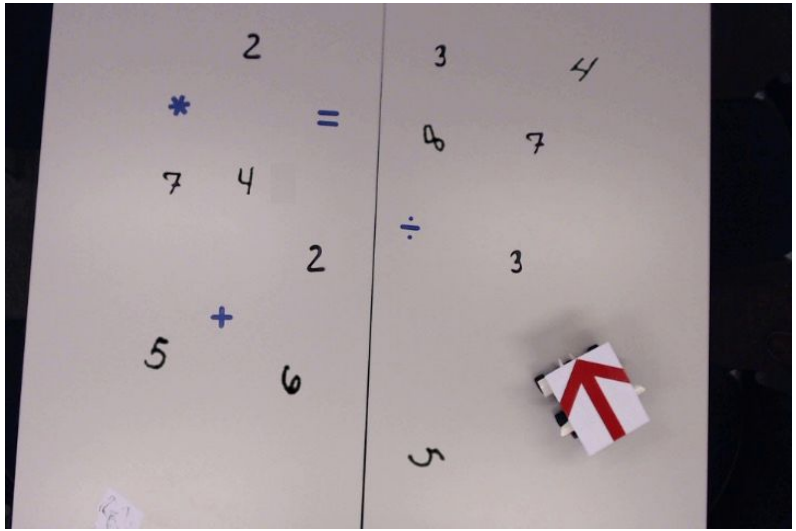
How to make a CNN rotation invariant ?

- Data augmentation during training the dataset (takes about 5 minutes)
- Classification rule uses 36-folds averaged results

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 26, 26, 32)	320
activation_13 (Activation)	(None, 26, 26, 32)	0
conv2d_10 (Conv2D)	(None, 24, 24, 32)	9248
activation_14 (Activation)	(None, 24, 24, 32)	0
max_pooling2d_5 (MaxPooling2	(None, 12, 12, 32)	0
conv2d_11 (Conv2D)	(None, 10, 10, 64)	18496
activation_15 (Activation)	(None, 10, 10, 64)	0
conv2d_12 (Conv2D)	(None, 8, 8, 64)	36928
activation_16 (Activation)	(None, 8, 8, 64)	0
max_pooling2d_6 (MaxPooling2	(None, 4, 4, 64)	0
flatten_3 (Flatten)	(None, 1024)	0
dense_5 (Dense)	(None, 512)	524800
activation_17 (Activation)	(None, 512)	0
dropout_3 (Dropout)	(None, 512)	0
dense_6 (Dense)	(None, 10)	5130
activation_18 (Activation)	(None, 10)	0

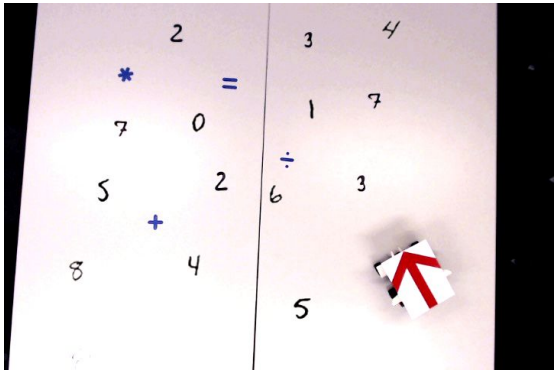
Total params: 594,922
Trainable params: 594,922
Non-trainable params: 0

Efficiency assessment



We used GIMP to create fake images and assess our code the best we could

- Adding more numbers
- Adding more rotations
- Changing light settings



Demo Time

Let’s see if it works !