Final Project: Classification and Detection with Convolutional Neural Networks

Description

For this topic you will design a digit detection and recognition system which takes in a single image and returns any sequence of digits visible in that image. For example, if the input image contains a home address 123 Main Street, you algorithm should return “123”. One step in your processing pipeline must be a Convolutional Neural Network (CNN) implemented in TensorFlow or PyTorch . If you choose this topic, you will need to perform additional research about CNNs. Note that the sequences of numbers may have varying scales, orientations, and fonts, and may be arbitrarily positioned in a noisy image.

Sample Dataset: http://ufldl.stanford.edu/housenumbers/
Related Lectures (not exhaustive): 8A-8C, 9A-9B

Problem Overview

Methods to be used: Implement a Convolutional Neural Network-based method that is capable of detecting and recognizing any sequence of digits visible in an image.

RULES:

- **Don’t use external libraries for core functionality**
  You may use TensorFlow and Pytorch and are even required to use pretrained models as part of your pipeline.
- However, you will receive a low score if the main functionality of your code is provided via an external library.
- **Don’t copy code from the internet**
  The course honor code is still in effect during the final project. All of the code you submit must be your own. You may consult tutorials for libraries you are unfamiliar with, but your final project submission must be your own work.
- **Don’t use pre-trained machine learning pipelines**
  If you choose a topic that requires the use of machine learning techniques, you are expected to do your own training. Downloading and submitting a pre-trained models that does all the work is not acceptable for this assignment. For the section on reusing pre-trained weights you expected to use a network trained for another classification task and re-train it for this one.
- **Don’t rely on a single source**
  We want to see that you performed research on your chosen topic and incorporated ideas from multiple sources in your final results. Your project must not be based on a single research paper and definitely must not be based on a single online tutorial.

Please do not use absolute paths in your submission code. All paths must be relative to the submission

directory. Any submissions with absolute paths are in danger of receiving a penalty!

Programming Instructions

In order to work with Convolutional Neural Networks we are providing a conda environment description with the versions of the libraries that the TA will use in the grading environment in canvas->files->Project files. This environment includes PyTorch, Tensorflow, Scikit-learn, and SciPy. You may use any of these. It is your responsibility to use versions of libraries that are compatible with those in the environment. It is also up to you to organize your files and determine the code's structure. **The only requirement is that the grader must only run one file to get your results**. This, however, does not prevent the use of helper files linked to this main script. The grader will not open and run multiple files. Include a README.md file with usage instructions that are clear for the grader to run your code.

Write-up instructions

The report must be a PDF of 3-6 pages including images and references. Not following this requirement will incur in a significant penalty and the content will be graded up to page 6. **Note that the report will be graded subject to a working code.** There will be no report templates provided with the project materials.

The report must contain:

1) A clear and concise description of the algorithms you implemented. This description must include references to recently published computer vision research and show a deep understanding of your chosen topic.
2) Results from applying your algorithm to images or video. Both positive and negative results must be shown in the report and you must explain why your algorithm works on some images, but not others.

You report must be written to show off your work and demonstrate a deep understanding or your chosen topic. The discussion in your report must be technical and quantitative wherever possible.

How to submit

Similar to the class assignments, **you will submit the code and the report to Gradescope (note: there will be no autograder part).** Find the appropriate project and make your submission into the correct project.
Important: Submissions sent to Email, Piazza or anything that is not Gradescope will not be graded.

Grading

The report will be graded following the scheme below:

- Code (30%): We will verify that the methods and rules indicated above have been followed.
- Report (70%): Subject to a working code.
  - Description of existing methods published in recent computer vision research.
  - Description of the method you implemented.
  - Results obtained from applying your algorithms to images or videos.
  - Analysis on why your method works on some images and not on others. (with images)
  - References and citations.

Assignment Overview

This project requires you to research how Convolutional Neural Networks work and their application to number detection and recognition. This is not to be a replica of a tutorial found online. Keep in mind this content is not widely covered in this course lectures and resources. The main objective of this assignment is to demonstrate your understanding of how these tools work. We allow you to use a very powerful training framework that helps you to avoid many of the time-consuming implementation details because the emphasis of this project will be on the robustness of your implementation and in-depth understanding of the tools you are using.

1 Installation and Compatibility

The provided environment yml description gives you with the versions of the libraries the TA's will during grading. We recommend you use conda to install the environment. Make sure the forward pass of your pipeline runs in a reasonable amount of time when using only a CPU as some TA's do not have a GPU.

Windows Users Warning:

Be warned that TA's grade exclusively on linux machines. Thus, it is your responsibility to make sure that your code is platform independent. This is particularly important when using paths to files. If your code doesn't run during grading due to some incompatibility you will incur a heavy penalty.

2 Classifier Requirements

Your classification pipeline must be robust in the following ways:

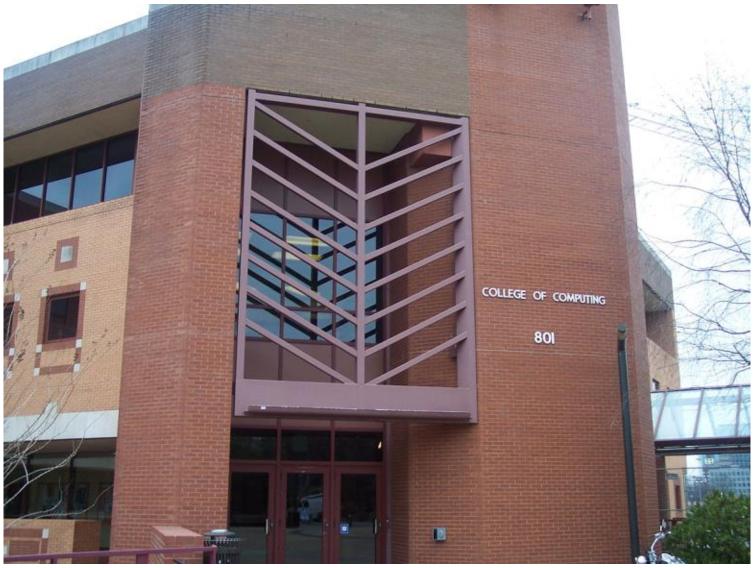
1. **Scale Invariance:**
   The scale of the sequence of numbers in an image in vary.
2. **Location Invariance:**
   The location of the sequence of numbers in the image may vary.
3. **Font Invariance:**
   You are expected to detect numbers despite their fonts.
4. **Pose Invariance:**
   The sequence of numbers can be at any angle with respect to the frame of the image.
5. **Lighting Invariance:**
   We expect robustness to the lighting conditions in which the image was taken.
6. **Noise Invariance:**
   Make sure that your pipeline is able to handle gaussian noise in the image.

3 Pipeline Overview

The final pipeline should incorporate the following preprocessing and classification components. We expect you to clearly explain in your report what you did at each stage and why.

3.1 Preprocessing

Your pipeline should start from receiving an image like this:



Notice that this is not the type of image your classification network trained on. You will have to do some preprocessing to correctly detect the number sequence in this image.
In the preprocessing stage your algorithm should take as input an image like the one above and return region of interest. Those ROI will be regions in the image where there is a digit. In order to perform this preprocessing step you will need to use the MSER algorithm (see https://docs.opencv.org/4.1.0/d3/d28/classcv_1_1MSER.html).
Note: The region proposal stage has to be separated from the classification stage. For this project we will use MSER to detect the ROI. This means that one-stage approaches (detection + classification) such as YOLO are not allowed.

3.1.1 Noise Management

We expect to see you handle gaussian noise and varying lighting conditions in the image. Please explain what you do in order to handle these types of perturbations and still have your classifier work.

3.1.2 Location Invariance

Since you don’t know where the numbers will appear on the image you will have to search for them using a sliding window method.

3.1.3 Scale Invariance

Make sure to implement an image pyramid with non-maxima suppression to detect numbers at any scale.

3.1.4 Performance Considerations

Running your full classifier through a sliding window can be very expensive. Did you do anything to mitigate forward pass runtime?

3.2 Classification

This section is concerned with the implementation of a number classifier based on the sample dataset.

3.2.1 Model Variation

There are several approaches to implementing a classifier and we want you get exposure to all of them:

1. Make your own architecture and train it from scratch. (https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html) (without pre-trained weights).
2. Use a VGG 16 implementation and train it with pre-trained weights. (Note: Final Linear layer will have 11 classes, https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html#finetuning-the-convnet)

Make sure you mention in your report what changes you made to the VGG16 model in order to use it for your particular classification task. What weights did you reuse and why? Did you train over the pre-trained weights?

3.2.2 Training Variation

We want you to have some familiarity with stochastic gradient descent. For this reason we want you to explain your choice of loss function during training. We also want an explanation for your choice of batch size and learning rate. In the report we expect a definition of these parameters and an explanation of why you chose the numbers you did. We also want to see how you decided to stop the training procedure.

3.2.3 Evaluating Performance

In order to evaluate the performance of your learning model we expect you to include training curves with validation, training and test set errors. When you compare the performance of each model we also want you include tables with the test set performance of the each model. We want to see a discussion of your performance in each of the models outlined above and we want to see empirical data demonstrating which is better. Your final pipeline should use the model and training that empirically demonstrates better performance.

4 Final Results

4.1 Image Classification Results

During grading, TAs expect to be able to run a python 3 file named **run.py** that writes five images to a **graded_images** folder in the current directory. The images should be named 1.png, 2.png, 3.png, 4.png and 5.png. You can pick these images; however, across the five of them we will be checking that you demonstrate following:

1. Correct classification at different scales
2. Correct classification at different orientations
3. Correct classification at different locations within the image.
4. Correct classification with different lighting conditions.

Notice, that since we allow you to pick the images, we expect good results.
In addition, add extra images showing failure cases of your implementation. Analyse and comment why your algorithm is failing on those images.