Image and Vision Computing Mini-Project

October 2022

Abstract

This describes the Image and Vision Computing assessed practical. The main goal is to compare sensitivities of a classical and a deep learning method as applied to image classification. The core issue here is the robustness of the classification. There are 3 stages to this assignment, and you must do all 3 for full marks. You must do this project with a partner.

Please put your group details here: https://docs.google.com/spreadsheets/d/1iugr-alpLoUhe5cs2ahUcbyZ7oEVggcg6t0JQmHONkg/edit#gid=0

Task Background

The dataset consists of 3035 images belonging to one of 8 classes. The classes are as follows: Black Widow, Captain America, Doctor Strange, Hulk, Ironman, Loki, Spiderman, and Thanos. The dataset is already split into training and validation folders for you. You will need to train two models, one "classical" and one based on "deep learning" to classify these images. Try to make both models as small as possible while keeping accuracy as high as possible. There is no expected F1 score and mostly your approach to the problem will be assessed.

The assignment is to:

1. Implement 2 classifiers, one based on "deep learning", and the other based on a classical method. (You are allowed to use pre-trained model weights for this).
2. Train both classifiers to perform multi-class classification on the dataset and calculate the F1 score for each using the validation data provided. (You may use image augmentation techniques as you see fit to increase your F1 scores).
3. Compare the robustness of the classifiers as progressively stronger perturbations are applied to the test images.
4. Report the results of evaluation of robustness.

Resources

The data for this can be found at:
https://www.kaggle.com/datasets/hchen13/marvel-heroes

A link to a resized dataset will be available in Learn soon.

In the meantime, try writing your own code to resize all the images.

Classifier Details

A possible classical method can be based on the HoG and/or SIFT feature extraction combined with an SVM classifier. The "deep learning" method can be a CNN such as ResNet.

The work below can be done using either Matlab or Python (or any other language that you are familiar with). You are expected to find implementations of the key technical components from public resources but connect the components together yourself for doing the robustness exploration. You are not expected to code up the algorithms yourself. Below you will find a starting point for 2 possible classifiers that you could use for the project: The ResNet-18 model and the SVM. However, you are free to explore any other classical or deep learning approach to classifying these images.

ResNet-18

The ResNet18 classifier is an end-to-end classifier that has an image as input and, in this case, has 8 outputs. You will use a 'pretrained' network but will need to do some additional re-training so it classifies just the 8 characters. Ideally, this means 'freezing' the convolutional layers, replacing the fully connected layers for 8 classes, and then retraining just the fully connected layers. However, don't worry if you cannot 'freeze' the convolutional layers, and have to tune the whole pre-trained network.

You can do data augmentation to improve the training. This should be limited to translation, rotation, flipping, and scale variations. You might explore some of the values of the hyperparameters associated with the fully connected layers, such as the number of layers, size of layers, learning rate, etc. You should split the training data into training and validation, but not touch the test data until the inference stage.

There are several sources for pretrained ResNet18 in either Matlab or Python: matlab: https://www.mathworks.com/help/deeplearning/ref/resnet18.html and pytorch: https://pytorch.org/docs/stable/torchvision/models.html or pytorch: https://docs.fast.ai/quick start.html. You are welcome to use other sources.

SVM

For the input feature to the SVM. Here we use HoG and/or SIFT feature detection in OpenCV and from this create a Bag of Visual Words (BoVW) for each image which we then use to train the SVM model.

Python code for HOG can be seen in the link below, similar MATLAB code can also be found: https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html

Some Matlab SVM code can be found in the following links listed below: https://uk.mathworks.com/help/stats/support-vector-machinesfor-binaryclassification.html or https://uk.mathworks.com/discovery/support-vector-machine.html .

Some Python SVM code is at: https://scikit-learn.org/stable/modules/svm.html.

As with the ResNet18 training, one should use the training and validation data samples to tune the algorithm, before computing test performance with the independent test data subset.

Robustness exploration

Once the two image classification approaches are trained and working well, there are a set of robustness evaluation experiments that need to be run using the trained classifiers. The central idea is that a variety of perturbations will be applied to the images, and then the classification rate will be re-calculated for both of the classifiers. You will produce a plot of classification accuracy *versus* amount of perturbation for each of the perturbations. Don't retrain the classifiers, just evaluate their performance on the perturbed data.

The evaluation metric is mean classification accuracy on the test set. Evaluation will use increasing levels of perturbation applied to the test set. The plots will show the change in accuracy with respect to the amount of perturbation. In total, there are 8 perturbations resulting in 8 plots. Show examples of each perturbation in your report.

Gaussian pixel noise

To each pixel, add a Gaussian distributed random number with 10 increasing standard deviations from {0, 2, 4, 6, 8, 10, 12, 14, 16, 18 }. Make sure that the pixel values are integers in the range 0..255 (e.g. replace negative numbers by 0, values > 255 by 255).

Gaussian blurring

Create test images by blurring the original image by 3x3 mask:

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Repeatedly convolve the image with the mask 0 times, 1 time, 2 times, ... 9 times. This approximates Gaussian blurring with increasingly larger standard deviations.

Image Contrast Increase

Create test images by multiplying each pixel by { 1.0, 1.01, 1.02, 1.03, 1.04, 1.05, 1.1, 1.15, 1.20, 1.25 }. Make sure that the pixel values are integers in the range 0..255 (e.g. replace > 255 values by 255).

Image Contrast Decrease

Create test images by multiplying each pixel by { 1.0, 0.95, 0.90, 0.85, 0.80, 0.60, 0.40, 0.30, 0.20, 0.10 }.

Image Brightness Increase

Create test images by adding to each pixel: { 0, 5, 10, 15, 20, 25, 30, 35, 40, 45 }. Make sure that the pixel values are integers in the range 0..255 (e.g. replace > 255 values by 255).

Image Brightness Decrease

Create test images by subtracting from each pixel: { 0, 5, 10, 15, 20, 25, 30, 35, 40, 45 }. Make sure that the pixel values are integers in the range 0..255 (e.g. replace < 0 values by 0).

Occlusion of the Image Increase

In each test image, replace a randomly placed square region of the image by black pixels with square edge length of { 0, 5, 10, 15, 20, 25, 30, 35, 40, 45 }.

Salt and Pepper Noise

To each test image, add salt and pepper noise of increasing strength. Essentially replace the amount in skimage.util.random_noise(...) with {0.00, 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18}

Your Report

Each team writes and submits a single report (10 pages long plus code in an appendix) that describes:

- The components that you selected and connected together for each of the two classifiers.
- How you trained each of the classifiers on the ‘clean’ dataset.
- The algorithms that you used to perturb the test images in each exploration.

- Plots of the classification performance as a function of increasing perturbation.
- Discussion of the robustness of the 2 classification approaches to the different types of perturbation.
- As an appendix, add the code that your team wrote. Do not include the code that was downloaded from other web sites, but include a statement about what code was used and where it came from.

Other Comments

The assignment is estimated to take 30 hours coding/test and 5 hours report writing per person, resulting in a 10 page report plus the code appendix.

You must do this assignment in teams of 2. Please submit your group details here:

https://docs.google.com/spreadsheets/d/1iugr-alpLoUhe5cs2ahUcbyZ7oEVggcg6t0JQmHONkg/edit#gid=0

A single, joint, report is to be submitted. Split the work so that each partner can do most work independently (i.e. share the work rather than duplicate it).

Assignment Submission

The deadline for submission is __25/11/2022_. Please make your submission anonymous (ie. no identifying information in the PDF). Name the submitted PDF file as: <student-number-1> <studentnumber-2> 1.pdf. Submit your report in PDF online using Learn. Details will be circulated later.

The proportion of marks are explained in the table:

Issue	Percentage
1. Clear description of algorithms used	50%
2. Performance on the dataset	10%
3. Clear MATLAB or Python code	15%
4. Discussion of results	25%

Publication of Solutions

We will not publish a solution set of code. You may make public your solution but only 2 weeks after the submission date. Making the solutions public before then will create suspicions about why you made them public.

Good Scholarly Practice:

Please remember the University requirement as regards all assessed work. Details about this can be found at:

http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct

Furthermore, you are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on a public repository then you must set access permissions appropriately (generally permitting access only to yourself, or your group in the case of group practicals).

Plagiarism Avoidance Advice

You are expected to write the document in your own words. Short quotations (with proper, explicit attribution) are allowed, but the bulk of the submission should be your own work. Use proper citation style for all citations, whether traditional paper resources or web-based materials.

If you use small amounts of code from another student or the web, you must acknowledge the original source and make clear what portions of the code were yours and what was obtained elsewhere.

The school has a robust policy on plagiarism that can be viewed here: http://web.inf.ed.ac.uk/infweb/admin/policies/guidelines-plagiarism .

The school uses various techniques to detect plagiarism, included automated tools and comparison against on-line repositories. Remember: *a weak assignment is not a ruined career (and may not reduce your final average more than 1%), but getting caught at plagiarism could ruin it.*

Late Coursework policy

Please see university-set guidelines regarding late submission and penalties for the same: https://web.inf.ed.ac.uk/infweb/admin/policies/late-submission-coursework