184.702 Machine Learning – Exercise 3.2 Deep Learning for Image or Text Classification

Note: please first read the generic introduction document for the 3rd exercise!

Topic 3.2: Deep Learning

N.b.: if you want to work with Deep Learning, but on more advanced topics, also take a look at the task on security (adversarial machine learning: inversion, evasion, poisoning attacks) and on synthetic image generation (using generative adversarial networks) in exercise Topic 3.1.

3.2.1: Image / Text classification - Feature Extraction & Shallow vs. Deep Learning

The main goal of this exercise is to get a feeling and understanding on the importance of representation and extraction of information from complex media content, in this case images or text. You will thus get some datasets that have an image classification target.

(1) In the first step, you shall try to find a good classifier with „traditional“ feature extraction methods. Thus, pick

For Images

- One simple feature representation (such as a colour histogram, see also the examples provided in TUWEL) and
- A feature extractor based on SIFT (https://en.wikipedia.org/wiki/Scale-invariant_feature_transform) and subsequent visual bag of words (e.g. https://kushalvyas.github.io/BOV.html for python), or a similar powerful approach (such as SURF, ..).

For Text

- One feature extractor based on e.g. Bag Of Words, or n-grams, or similar

You shall evaluate them on a few algorithms (for image, also specifically including an MLP), and parameter settings to see what performance you can achieve, to have a baseline for the subsequent steps.

(2) Then, try to use a deep learning approaches, such as **convolutional neural networks (for images) or recurrent neural networks (for text)**, or other approaches, and see how your performance differs. Try at least two different architectures, they can be (reusing or be based on)existing, well-known ones.

Compare not just the overall measures, but perform a detailed comparison and analysis per class (confusion matrix), to identify if the two approaches lead to different types of errors in the different classes, and also try to identify other patterns.

For images, you can base your DL implementation on the tutorial provided by colleagues at the institute, available at https://github.com/tuwien-musicir/DL_Tutorial/blob/master/Car_recognition.ipynb (you can also check the rest of the repository for interesting code; credit to Thomas Lidy (http://www.ifs.tuwien.ac.at/~lidy/)). Mind also that you will find plenty of examples on how to create and train CNNs / RNNs in various frameworks - tensorflow, keras, pytorch, ...

Also perform a detailed comparison of runtime, considering both time for training and testing, including also the feature extraction components.

For the **datasets** you shall work with, pick two text/image datasets, from the list of suggestions below. If you have proposals for other datasets, please inform me (mayer@ifs.tuwien.ac.at), and we can see if the dataset is fit.

- For Images :
  - The German Traffic Sign Recognition Benchmark (GTSRB), http://benchmark.ini.rub.de/
  - PubFig83: http://vision.seas.harvard.edu/pubfig83/
  - CIFAR-10: https://www.cs.toronto.edu/~kriz/cifar.html
  - Tiny ImageNet: https://tiny-imagenet.herokuapp.com/
  - FashionMNIST: https://github.com/zalandoresearch/fashion-mnist
  - Labeled Faces in the Wild, http://vis-www.cs.umass.edu/lfw/, for creating a classifier recognising different people (1 person == 1 class). See also https://scikit-learn.org/0.19/datasets/#the-labeled-faces-in-the-wild-face-recognition-dataset. For this dataset, it is best to use a subset only, of people that have a reasonable number of images (e.g. >=20).
  - CelebA: https://plg.uwaterloo.ca/cgi-bin/cgiwrap/gvcormac/foo07. Use this dataset, as above the Labeled Faces in the Wild, for a face recognition dataset; take a similar subset.
- For Text Data:
  - 20 newsgroups: http://qwone.com/~jason/20Newsgroups/
  - Reuters: http://www.daviddlewis.com/resources/testcollections/reuters21578/
  - Fake News Dataset: https://github.com/GeorgeMcIntire/fake_real_news_dataset
  - Anything else

Recommendations for CNNs specifically, but also for RNNs:

- Use architectures of your choice – you can work with something simple like a LeNet, optimised for smaller network structures like SqueezeNet, or more advanced architectures (where maybe transfer learning is required for efficiency reasons, see below).
- Use as well data augmentation (you can reuse the code from the tutorial), and compare it to the non-augmented results
- Also consider using transfer learning of pre-trained models
- As stated above, mind that there is plenty of example code available that you can reuse (in your report, state what sources you have utilised).
- The main focus is on analysing the differences in traditional vs. deep learning.

3.2.2: Image-to-image processing via Deep Learning (e.g. CNN-based Image upscaling)

This task aims to make you familiar with some image-to-image deep learning tasks. More precisely, you can work on (a) image upscaling, (b) image deblurring, or (c) image colorization (pick only one!). You shall on purpose distort your images, and then restore them as close as possible to the original state.

We expect you to perform the following steps:

1. Pick a CNN that is applicable for your image-to-image task. For instance, you can use the following approaches. *Please note that we did not test the mentioned implementations/github repositories, so issues are not excluded. Feel free to use any other relevant solution you find.*

a) Super-Resolution Convolutional Neural Network (SRCNN)
- Original paper: https://arxiv.org/abs/1501.00092
- (Unofficial) Pytorch implementation: https://github.com/yjn870/SRCNN-pytorch

b) DeepDeblur
- Original paper: https://openaccess.thecvf.com/content_cvpr_2017/papers/Nah_Deep_Multi-Scale_Convolutional_CVPR_2017_paper.pdf
- Pytorch implementation: https://github.com/SeungjunNah/DeepDeblur-PyTorch

c) Colorful Image Colorization
- Original paper: https://arxiv.org/abs/1603.08511
- Official Pytorch implementation (testing only): https://github.com/richzhang/colorization
- (Unofficial) Pytorch implementation: https://github.com/Time0o/colorful-colorization
- (Unofficial) Tensorflow (Keras) implementation: https://github.com/RoddeTheR/deep-learning-image-colorization

2. Pick a dataset (different from the implementations above) for your task. Keep in mind that you can use any high-resolution dataset (for instance, COCO https://cocodataset.org/#home; subsets of ImageNet, …) and transform it for your task (downscaling, adding blur, converting to grayscale).

3. Train a network (or fine-tune a pre-trained one) on your dataset.

4. Evaluate the performance of the model

- Qualitatively by visualizing and analysing some results (e.g. compare original versus restored (upscaled, …) image)
- Quantitatively, e.g. by computing some image comparison metric (e.g. the PSNR score if it is applicable for your task, or L1 / L2 distances, …), or by comparing the performance of your original vs. the restored (upscaled, …) images when used as a training set e.g. on a classification task.