AAI3001 - the small project

Alex

September 22, 2023

Submission due Week 8 , Tuesday 11:59pm

1 The small project 1

Learning goals:

- running a small project from the start, including
  - Data splitting
  - writing a custom data set
  - training using Finetuning
  - evaluation of the results
- the predicted scores contain information about the likelihood of errors.

The dataset `https://zenodo.org/record/7711810#.ZAm3k-zMKEA` is a multi-class landuse dataset. Problem is to predict the dominant land-use from a satellite image.

It comes from this paper: `https://ieeexplore.ieee.org/abstract/document/8519248` .

img credit: Dataset

1.1 What to consider before starting to train

- to split the data into training, validation and testing
- to verify that your splits are disjoint after creating them
- to set up the code so that it can run when the split is re-generated

1.2 an overview over tasks

- split your data into train/val/test datasets using a random generator seed
- write a dataset class for the train/val/test datasets. In the sci-kit learn library is a useful function that can help you to binarise strings. You are not allowed to use to dataset classes from tensorflow or pytorch for that set. Why ? Writing Dataset readers might be the start in a new company.

Task1 write code to train, validate the trained model and for final testing on the test split.

  - Report the average precision measure (google for it, sci-kit learn has it too in a metrics subpackage, you can use that one) on the val and test sets – for every class of the 10, and the mean average precision over all 0 classes. The average precision is a measure for the quality of a ranking of predictions.
    * note in particular: the average precision for one class is computed over the set of all predicted scores for this class. All predicted scores refers to scores for this one class obtained for all samples in the validation set and their labels for this particular class (which is binary, presence or absence). You need to compute average precision using an ordered set of prediction scores (for the whole validation set) and the corresponding binary labels (for the whole validation set). It is not computed in the same way as accuracy (one number for each sample, then averaged over all samples) would be. Therefore: to compute mean average precision, you must collect the set of predictions for all samples, and this for all 17 classes.
  - Report the accuracy per class, and the average of it over all classes on the val and test sets
  - Report curves for the train and Val loss per epoch for the finally selected model.
  - do not optimize on the test set, that is, report test performance only once, after all hyperparameters are selected on the val set.
  - try out at least 3 different schemes of Data Augmentation on the validation set. Different can mean different sets of augmentations or different sets of their limiting parameters

Task 2 Do now the same but with a multi-label prediction, a multilabel loss, and modified labels, for which

Labelling Changes

- you set the label Annualcrop to 1 whenever PermanentCrop is 1 and the other way round
- you set HerbaceousVegetation to 1 if Forest is 1 (but NOT the other way round)
- you do these changes during training, validation and testing.
- note that the prediction now changes. You do not use softmax and predict strictly a single class anymore.

  Note that your classification accuracies can get worse with the new way of predicting the presence of a class. This is okay here.

Additional Info:

- you finetune a new model with this schema
- for this just try out 1 Data Augmentation Schema
- report the same curves and measures as in Task 1.
- put the training/eval code for this in a separate python file

Why do I ask you to do that ? The given original labels are mutually exclusive multiclass labels, however note that given ground truth is often an approximation to the content and not the type of truth encountered in religion :) .

- the highway class usually also contains other structures (residential or pasture or crops or vegetation). There is no patch which contains only highway.
- Pasture is often neighboring to crops. Crops are often neighboring to forest.
- Forest is a type of herbaceous vegetation, unless you start a novel biological classification with plants, animals, viruses and ... trees.

(Bonus) You can add predictions from the hyperspectral part of the dataset and see if it helps. However this requires you to write your own custom network class

- it should be derived from nn.Module, so that a call to its `.named_parameters()` returns all trainable parameters
- which encapsules at least two pretrained models as members of itself - one for RGB and at least one for all hyperspectral channels
- which overwrites the forward pass function (`functools.partial` can help) of these two encapsuled models to stop at the last feature layer before the classification layer and to return feature maps from these second-last layers

  - which concatenates those features and passes them into a `nn.Linear` layer for the prediction on features from RGB and hyperspectral processing.
  - you can pass all hyperspectral channels through the same model, by that you train a joint model for all hyperspectral channels.

- write a report of 2 to 5 pages on what you did. Aim is so that others would be able to reproduce your results - it should contain what is needed to use your code and to reproduce your results (including seeds).

  Note down loss function, learning rate schemes, training procedures, **all relevant hyperparameters used in evaluation and the finally chosen hyperparameters** and so on.

  It does not need to be lengthy, just be self-containing. Short sentences are okay, prettyness of language does not matter. It is not an English language essay contest.

1.3 Deliverables

- training phase: code for training on the dataset with finetuning
- a trained model
- validation phase: code which uses the pretrained model to predict on the test set images and saves those predictions, and which computes the mean average precision over all 10 classes.
- **a reproduction routine: the scores from the pretrained model computed on the fly when we run your code should be compared against the scores which you saved when you ran your code**
- a brief pdf-report containing the following:
  - your name and your matriculation number
  - setting you used to define the last layer
  - describes the experimental parameters of the training (learning rate batch size, seed values and anything else necessary to reproduce the training)
  - shows train/val curves of the loss for the setting which you used to save the model,
  - novels longer than 10 pages will not be entertained.
- put everything: codes, saved model, saved predictions, the pdf and everything else you want to add into one single zip file.

1.4 Coding Guidelines

- path portability: all paths (dataset, pretrained model, saved predictions) must be relative to the root path of the main .py-file or relative to some path variable in the code
- no absolute paths
- reproducibility: set all involved seeds to fixed values (python, numpy, torch)
- a python file for the code or several files
- it should run using the following steps:
  - unpacking the zip files
  - set **one single path** for the root of the dataset. This must be documented. Nothing else should be needed to set it up
- code should run without typing tons/dozens/piles of parameters on the command line!! `python blafile.py`. Configuration parameters should be inside the code in one place
- python scripts. No jupyter notebooks.

Do you think that whoever will check your code, would like to follow up errors in a notebook ??

+ Repeat this then for 50 students?
Do you think you can write prototype code in jupyter notebooks for any projects which are beyond the very smallest ones?

1.5 Additional caveats:

- We do one thing wrong here: you try it out with only one train/val/test split