Dr. Muhammad Rana, Dr. Anjan Dutta
TAs: Ammarah, Sergio, Haohe, Umar, Anindya, Jiantao
Institute for People-Centred AI
University of Surrey

Spring Term 2023

EEEM068: Applied Machine Learning
Project: Vision Transformers for Knee Abnormality classification
Project TA Lead: Umar Marikkar (u.marikkar@surrey.ac.uk)

- Submission deadline: **TBC**

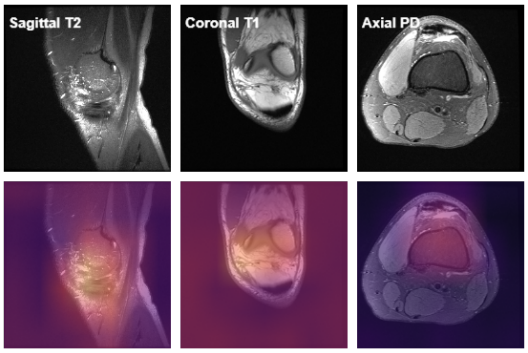

Figure 1: Instances of knee MRI images from the MRNet dataset

Project specification

Overview

This project will require you to apply the learnt knowledge that you are supposed to acquire in the lectures, labs and tutorials of the EEEM068: Applied Machine Learning modules. This project is one of the options which you can opt in for undertaking summative assessment, carrying 25% of the total module marks. In this project, you will be training a Small-sized Vision Transformer (ViT) and fine-tuning for a Medical Imaging task such as knee abnormality classification.

Submission

TBC

You should submit a zip file containing the following:

- Your code;
- A 5-pages report (IEEE double column format) explaining your code, visualising your results you obtained and discussing your observations. If you want you can include additional pages only for visualisations and references (this is optional and you won't lose any marks if you do not include additional pages). However, the main text of your report should fit in the first 5 pages and the additional pages (if any) should only include visualisations with short captions and references.

Please note that for all the visualisations and tables that you include in your report, it is important to include a reference in the main text (typically using a Figure or Table number).

Background

For theory of ViTs, please see the associated course material and the code implementation can be found in the 'pytorch image models - timm' library. The idea of this project will be to train ViTs to classify 3 categories of knee injuries that commonly occur during sports, thereby providing quick diagnosis which can potentially help in faster treatment.

You will have to implement a similar procedure as of Workshop 4, where you implemented CNN for image classification and for transfer learning. You can use the examples in Workshop 4 as your starting point. However, please note that you would need to modify and expand it significantly, as you will be using a ViT instead of a CNN.

Knee MRI dataset and ViT-small network

You will download the knee MRI dataset ("`MRNet`") which includes 1370 knee MRI exams performed at the Stanford Univerity Medical Center. The dataset contains 1,104 (80.6%) abnormal exams, with 319 (23.3%) ACL tears and 508 (37.1%) meniscal tears. This dataset can be downloaded through kaggle here [1].

In this dataset, each subfolder within the train and validation directories contain MRI's taken from the axial, coronal and sagittal planes, respectively. You may use any or all planes to build yor dataset. For starters, using only the images taken on the sagittal plane will be sufficient. You may formulate this problem as a multi-class classification, where you will be required to predict one of the three injuries for each image.

Note that MRI scans are in 3D volumetric format, and hence you may use a couple of slices per 3D scan to create two 2D images per MRI scan. However this is completely upto you, you may also use 3D volumes as inputs if need be.

As in all cases, please report the accuracies on the validation set.

You may use SiT-small model available within the `pytoch image models - timm` package. You may load an ImageNet pre-trained checkpoint (SiT-S) from the SiT repository (`https://github.com/Sara-Ahmed/SiT`). SiT should give a better starting point for pretrained models compared to ViT-S, DeiT-S or others supervised models. Feel free to use any other model which you think is more suitable SiT-S is there just as an example.

Train the ViT

- Load the "`MRNet`" dataset by implementing an appropriate dataset and dataloader. Each datapoint is a numpy (.npy) file.
- Since most of the standard ViT models (including SiT-S) are trained on images with size 224×224 it is necessary to resize the images to 224×224. Incorporate the necessary transformation functions which can resize the images as above.
- Generally when the train and validation sets have been given, you need not split the dataset. However, if you are using the validation set (as has been mentioned previously) to report the test accuracies, you may further split the train set into a train and validation component for hyperparameter tuning.
- Use data augmentation of the training set, by including random X and Y translations, random rotations (for a range around -20 to 20 degrees, since very large rotations would not make sense), as well as random scalings.
- Create a ViT model by removing the last layers (classification heads) of ViT and adding new layers that are consistent with the classification problem that you need to solve. To learn faster in the new layers than the transferred layers from the ViT, try different and increased learning rate. Have a look on this discussion thread to know how to set different learning rate for a specific layer.
- Train the ViT model. Try to tune the hyperparameters, such as batch size, number of epochs, learning rate etc. Try to plot the training progress using the `tensorboard`.

Test the ViT

- Calculate the top-1 classification accuracies and the confusion matrix for the validation set. Plot the latter in such a way that the confusing cases could be understood effectively.
- After training and validation on the MRNet dataset, use a few images of knee MRI scans publicly available on the internet (for example, search ACL tear sagittal plane MRI) and pass them through your model. Report the difference in performance when using data from the MRNet dataset as your query images, as opposed to MRI images fetched online. It is important to understand why the predictions may be as they are, hence please provide insights on the predictions you obtain.
- Try different values of hyperparameters to improve the training behaviour and the accuracy measures. Observe how the choice of hyperparameters affects the results.

Discuss your Observations

- In your report, you should discuss your observations with reference to the fine-tuning of hyper-parameters, training behaviour, accuracy measures, confusion matrices and the visualisations of the ViT that you have produced. Whenever appropriate, please refer to the corresponding figures / tables to make your observations more concrete.

[1] `https://www.kaggle.com/datasets/cjinny/mrnet-v1?resource=download`

Extra credit

Extra credit will be awarded if one could potentially perform additional tasks related to the main recognition task. These additional tasks might include but not limited to visualization, interface design etc. For example, you may perform weakly supervised localization by plotting the class attention maps (how the class token attends to each image token in the ViT) and observe if the class token is able to localize the relevant injury in the MRI.

MARKING CRITERIA

The project will be assessed giving the 50% weight (12.5 marks) to technical report and 30% weight (7.5 marks) to functionality and 20% weight (5 marks) to code quality according to the following criteria:

REPORT QUALITY [50 marks]

Whether the results are well presented and discussed.
In particular:

- Is the report well written and clear?
- Is the report well structured?
- Are the figures clear and well explained?
- Does the report provide a clear explanation of what has been done to solve the problem?
- Is there a sufficient discussion regarding observations on the produced results?

The distribution of the marks within the report are as follows:

- Abstract: 10 marks,
- Introduction: 10 marks,
- Literature (minimum 5 papers): 15 marks,
- Methodology: 25 marks,
- Experiments: 30 marks,
- Conclusion and future work: 10 marks

FUNCTIONALITY [30 marks]

Whether the submitted program performs as specified.
In particular did the code implement all the steps specified in the previous sections?

CODE QUALITY [20 marks]

Quality and efficiency of the coding, including appropriate use of documentation.
In particular:

- Is the code efficient?
- Is the code extensible and maintainable?
- Is the code clear and well documented?