Course: Big Data
Topic: Team assignments
Subject: Description and practical details

Introduction

100% of your grade in the Big Data course is based on a team assignment, combined with conceptual questions. The conceptual questions are based on the discussion topics from each lesson.
This document will focus on the team assignment, and explains in more detail what is expected of you.

The assignment's main goal (MVP) is an image classification app in Streamlit, but below many possible extensions, or extras, are provided.
At least one of those extras should be included as a mandatory part of the assignment. The team can choose which one. Other additional extras can earn the team some extra credits.

Assignment – Image Classification

A. Collecting / Selecting dataset

- Collect or select a dataset of images that can be categorized in at least 5 different classes. Be creative.
- Make sure the images you want to classify are challenging enough (so no 'car versus truck versus plane'-type of classifications).
- The level of challenge and creativity, concerning the dataset, will be a factor in grading.
- Examples:
    - Images of different types of flowers.
    - Images of different car brands.
    - Images of different breeds of horses.
    - Images of different types of fighter jets.
- Assess the quality of your dataset, including an EDA. If it's not up to par, take steps to remedy. Include this analysis in your final report.

Possible Extra:
    - Include images with more than one label, and extend your model, so you can do multi-label predictions

B. Modeling using fastai

- Use fastai's / Pytorch modelling techniques, including transfer learning, and some of the advanced options we've discussed in class, to build a state of the art deep learning model that is able to classify the images.
    - To be clear: fastai/Pytorch has to be used. See Extra for Keras / Tensorflow models
- Assess the quality of the learned model. Include this analysis in your final report.

Possible Extra:

- Keras/TensorFlow models can only be used as a possible extra, to be able to compare the ease of use/training versus the fastai/Pytorch models. Make sure the Keras models also apply advanced learning options, and employ transfer learning.

Possible Extra:

- Research and apply ROC and AUC metrics to assess the quality of your classification model

Possible Extra:

- Benchmark your model's performance to the performance of Google's TeachableMachine. And/Or compare your results with plug-and-play solutions like Google's AutoML Vision or Google's Vision API (use your GCP student credits). Or, you can even give Google's brand-new Vertex AI platform a spin

C. Deploy as WebApp, via Streamlit

- Make an interactive web application using Streamlit, where:
  - You can explore / evaluate the dataset (basics of part A of this assignment)
  - You can train a new image classifier model, by selecting advanced fastai options, and save the trained model when done (basics of part B)
  - You can test a pre-saved model, by uploading a new image and have the application tell you the class the image belongs to.

Possible Extra:

- Create an API endpoint around your model, by using FastAPI (don't confuse it with fastai), or by using Flask.
  - Make sure you can test your API (curl, Postman,…)

D. Possible Extra: NLP

- We've covered NLP during the last lessons, so it would be a shame not to include it in the assignment. But since NLP is a completely different task then image classification, you must first create a multipage Streamlit app, so you have a dedicated page for NLP

  - Possible Extra:
    - Create a Streamlit page where we can train and evaluate the ULMfit algorithm, on a new text dataset of your own choice, for instance the toxic comment dataset from Kaggle (https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data)
  - Possible Extra:
    - Create a Streamlit page where we can try out some of the pretrained (Transformer-based) models from HuggingFace, like the Question-Answer task, or the Sentiment-Analysis, or the Text Summarization task.

        - Provide an input option for the user, run it through the model, and present the output

Deliverables

- A report that explains your approach and implementation for the different aspects of the assignment.
- The Python code used to complete the assignment, with instructions on how to install required dependencies and run all programs.
- A presentation, given during week 6 of the course.

How you and your team will be graded

- *Approach*
    - Techniques used to collect your data
    - Techniques used to train deep learning model.
    - Techniques used to calculate model quality, and interpret the results.
    - Techniques used to deploy your model.
- *Implementation*:
    - Reproducibility and portability of the submitted programs.
    - Does the code run without errors?
    - Cleanliness and readability of the code.
    - The mandatory extra
- *Communication*:
    - Quality and style of the report, and the presentation.
    - Answers to follow-up questions during presentation.
    - Answers to conceptual questions during presentation.
- *Additional weight factors*:
    - Informal peer review (performed by instructors over the weeks)
    - Going the extra mile (more than one extra)

Practical details

- All assignments and presentations are done in the same team of 3 people.
- The online block is available to work on this assignment with your team. Your instructor will check in with you, via Teams, on a regular basis to see how things are going and gauge team dynamics.
- The **Python code and written report pdf** has to be handed in as a zip file before **Wednesday 14 December at 23:59** through Canvas.
    - You don't have to hand in your presentation.
- Your team will **present/defend** on Monday 19/12 or Tuesday 20/12.
    - Your team presentation should be about 15-20 minutes (including a live demo of the application).
    - After your presentation, around 10 minutes are reserved per student for follow-up and conceptual questions.