Problem Description, Work Plan and Project Management

Problem

The need for robust image classification is becoming increasingly prevalent in our daily lives from the mundane, providing offline metadata for smart phone memories to the vital, real-time object recognition in autonomous vehicles and braking systems. The rate of data creation is astounding. PwC predicted in 2018 that the 'universe of data' would grow to 44 zettabytes by 2020. The true figure reaching 59 zettabytes with Statista projecting a further 74 in 2021 alone and 180 in 2025. Much of this data will be comprised of images and videos. To take one company as a study. According to their own 2021 Q1 reporting Facebook has over 2.8 billion active monthly users. Facebook research indicates 4 petabytes of data is created on their platform per day including 350 million photos and 100 million hours of video watch time. The shear amount of data in all areas of image classification be it advertising, vehicles, or medical imaging coupled with sparsity of meaningful information amongst the noise and need for real-time applications make human based classification accurate but completely impractical. It is therefore no surprise that the increased demand for automated image classification has led to the study and development of several image classification techniques.

Background

The aim of this project is to gain an understanding of the fundamental forms of feature extraction and image classification by developing each in turn from scratch. The primary goals are to maximize the precision of these models and compare how different models perform in terms of metrics, feature extraction time and training / testing time. In order to gain an understanding of the techniques involved I will research, evaluate and implement each chosen method of feature extraction and classification in Python without any existing machine learning libraries. The only exceptions to this will be using an existing dataset for training and testing along with a mathematics library (NumPy).

Given that I want to evaluate various models through direct comparisons of their metrics I will choose a well-defined proven dataset namely CIFAR-10. The CIFAR-10 dataset consists of 60000 RGB images each measuring 32x32 pixels. CIFAR-10 as the name suggests contains images for 10 different classes with 6000 images per class. A class is simply defined as a distinct category. The dataset also contains class information for each image (referred to as the image's label). The main benefits of using an existing dataset are the standardised format, the variance of images (both interclass and intraclass) and the elimination of the time-consuming process of gathering, categorizing and standardizing the samples into a format that lends itself to efficient processing.

It should be noted that the CIFAR-10 set is known to have 3.3% duplicate or near-duplicate images between training and testing data. [1] This paper suggests that the duplication could lead to a sense of memorisation in developed models rather than properly generalising to the problem. It proposes creating a ciFAIR set replacing duplicates with new images in the same class. Thereafter, a 9% drop in accuracy was observed (based on a CNN (convolution neural network)) suggesting there was indeed a noticeable amount of overfitting and memorization caused by the training, testing data overlap. This could be something to explore (given ciFAIR is widely distributed) provided the project has reached an advanced stage of development.

CIFAR-10 is also widely understood in the field of machine learning. With many papers containing experimental results for a variety of different models [2], these will provide useful reference points for model evaluation during development.

Requirements

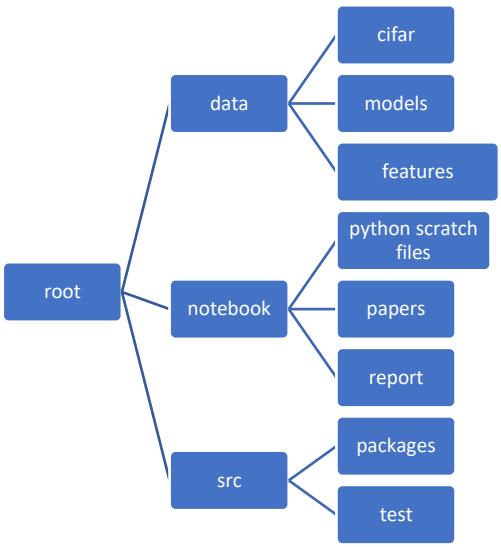
Practices to Follow

The first stage of the project consists of selecting areas for exploration and defining a set of realistic requirements. I have broken these down into sections. The project will ultimately be evaluated using conventional metrics applicable to machine learning with different feature extraction techniques contributing to those metrics. This may increase the difficultly of defining a rigid set of requirements given the projects research-based nature rather than stakeholder requirements. However, it must adhere to standard software engineering practices discussed below.
Firstly, a set of principles to adopt during development

- Single responsibility: Each component of the project should only be responsible for one specific task with similar features grouped using python modules.
- System extensibility: Latter functionality must have the ability to integrate with the existing system without cumbersome refactoring, isolation and encapsulation of features.
- A standardized API: By API I mean similar features will have the same class structure. For instance, all feature extraction classes should have an 'extract' method allowing the switching out of different features during development.
- Avoid duplication.
- Robust Unit testing: Every component should have associated unit tests if required. Parameterized testing should be used for conditional function testing to test a range of inputs.
- KISS: Given many of the feature extraction techniques are purely algorithmic I will focus on clear implementations over optimal solutions. The 'keep it simple stupid' methodology will aid debugging and readability.
- Adopt version control for all aspects of the project, code, assets and writeup. Use of branches and pull requests for larger features.
- Consistent naming convention: I with be using Python and will adopt the PEP8 standard.

A Clear Project Structure

The project's structure must be logically defined. Having researched various ML libraries and projects. The project structure should contain both the source and writeup and should make use of version control as one of the means of backup. [4]



Folder structure

- Root (FYP-Cifar-Image-Classifier): Contains all directories of the project and root of the Git Repository
- Data: Contains the CIFAR-10 dataset (potentially the ciFAIR set)
- Features: Offline extracted features such as edge data, HOG vectors etc.
- Notebook: Containing write up material, interim reports and resources such as academic papers and machine learning articles.
- Src: Housing all Python source code and tests in a clear package hierarchy.

Features

As set out in the original work plan though are now either implemented or partially implemented.
Feature extraction methods.

- HOG: Currently implemented at block level, HOG vectors can also be visualized.
- Canny Edge Extraction vs Sobel Extraction: Both working.
- SIFT: Part way implemented following [3] key point extraction and refinement is working, though matching still has issues.
- Colour histogram: Implemented.
- PCA: To be implemented in order to reduce vector sizes and thereby model training times.

Image processing features.

- Weighted grayscale conversion implemented
- Image convolution implemented along with padding calculation and stride parameters.

Classification Models.

- KNN: Currently implemented vectorized KNN peak accuracy with HOG at 34%
- Logistic Regression: Currently implemented One vs All which is performing poorly. I believe there is an issue with my method of training. Also implemented in the context of organic vs inorganic reaching 85% accuracy with HOG.
- CNN: Currently in very early stages, recent updates to convolution method adding stride parameter. Layer framework has also been implemented each having a pointer to an activation function.
- Linear regression models can also be persisted to file eliminating the training stages between each test run.

Metrics and Scoring.

- ROC Curves have been implemented along with the ability to calculate the optimal threshold for logistic regression models.
- Confusion matrix and multiclass confusion matrix implemented.
- Metrics calculation: TP TN FP FN metrics calculations added precision, recall, accuracy F1.
-

UI.

- UI also added via PyQt5 and working, though minimal at present
- Chart visualization: ROC plots and comparisons
- Model settings: tweaking of choose features classifier parameters and CNN hyper parameters
- User acceptance testing for all UI elements

List of Hardware and Software

- Multicore CPU for parallel image processing leverage GPU hardware if possible
- Python 3
- NumPy – High performance math library taking advantage SIMD
- PyQt5 for application UI
- VS Code
- Gitlab
- Office
- LaTeX

Work Plan

I will take an agile approach to development. I will strive to quickly synthesise each task and develop a minimal but working implementation. Throughout development I will need to reassess the project as a whole and ensure duplication is minimal, tests have been covered and any tech debt has been planned into latter development. Despite the projects iterative nature certain milestones can be drawn from the requirements defined above. These will be shown as tasks in the Gantt Chart. The chart will also contain review stages to ensure the code base remains clear and previous features are continually improved upon (if they still make sense, are in active use) and organised in appropriate packages. The convolutional neural network will take the most amount of time to research and develop. I am therefore aiming to begin the research early in conjunction with more basic models allocating more development time to the CNN in early 2022. I am also conscious that work on the dissertation must remain as active as possible during development. I will therefore compile all reading material throughout development and store this in the data folder of the project structure. I will also aim to start the dissertation in January of 2022. The video demo will also be started during early to aid development, hopefully presenting the system will help the planning and development of the user interface as in effect I will be test driving the development. It will also be critical to allocate sufficient development time for issues that may arise when evaluating the system. Result comparison may uncover problems with a component otherwise thought complete. I want to plan as much time in as possible. Provided the model metrics are as expected I would also like to compare the results with the ciFAIR-10 dataset and evaluate the impact duplication between training and test has on the metrics.

Image Classification From Stratch

FYP - Cifar Image Classifier

20-Sep-21 START DATE

09-Jan-22 END DATE

Task ID	Task Name	Start Date	End Date	Duration (Weeks)
T01	Research appropriate features & models that can realistically be delivered within the time and focus on a iterative approach to their implementations	20/9/21	3/10/21	2
T02	Research the CIFAR-10 dataset necessary tools for development	20/9/21	3/10/21	2
T03	Initial project setup, python venv dependencies and source control. Implementation methods for reading CIFAR dataset. First iteration of core algorithms i.e. convolution.	20/9/21	3/10/21	2
T04	Simple feature extraction i.e. greyscale and colour histogram	4/10/21	10/10/21	1
T05	Feature extraction techniques sobel edges and colour histogram	4/10/21	10/10/21	1
T06	Simple model (KNN) development (concurrent along side T04)	4/10/21	17/10/21	2
T07	Evaluate techniques developed using naive metrics such as accuracy to gather baseline results with different features	11/10/21	17/10/21	1
T08	Advanced feature extraction techniques HOG RnD. Implement HOG and test with KNN agasinst other features. Also work on HOG visualization to prove the algoritm works as expected	18/10/21	31/10/21	2
T09	Review current state of the program, refactor to ensure consistency of code, ensure adequate test coverage. Normalise CIFAR-10 on load.	1/11/21	7/11/21	1
T10	Research Logistic Regression implementation approach. Look at common LR python class structures such as Keras	1/11/21	14/11/21	2
T11	Design class for LR ensuring the class shares simlarities (black box) with KNN classifier. Each model should be interchange to aid testing and evaluation.	8/11/21	28/11/21	3
T12	Logistic regression models require a lot of time to train. Work on persisting the models to file	22/11/21	28/11/21	1
T13	Project cleanup fix issues with HOG visual at cell level. Cleanup comments in code. Work on UI implementation add PyQt5 support to the project and confirm matplotlib support in PyQt5	29/11/21	5/12/21	1
T14	Metrics. Work on the initial implementation and tests for metrics for both binar and multiclass. Work on Confusion matrix generation ROC curve, thresholds and run these features against logistic regression model.	29/11/21	12/12/21	2
T15	Exams. Continue project reasearch focusing particularly on CNN implementation for the new year. Along with SIFT implementation (after researching) SIFT appears to be better for matching images rather than determining images though I would still like to implement it given it is more challenging.	13/12/21	26/12/21	2
T16	Implentation of SIFT following David Lowe's original paper. Also implement Canny Edge extraction algorithm for comparison with sobel	27/12/21	9/1/22	2
T17	Design API for CNN and update convolution algorithm, if possible optimize and also add stride paramter.	3/1/22	9/1/22	1
T18	Prepare Interim demo of FYP Demo confirmed for 12/1/22	3/1/22	9/1/22	1

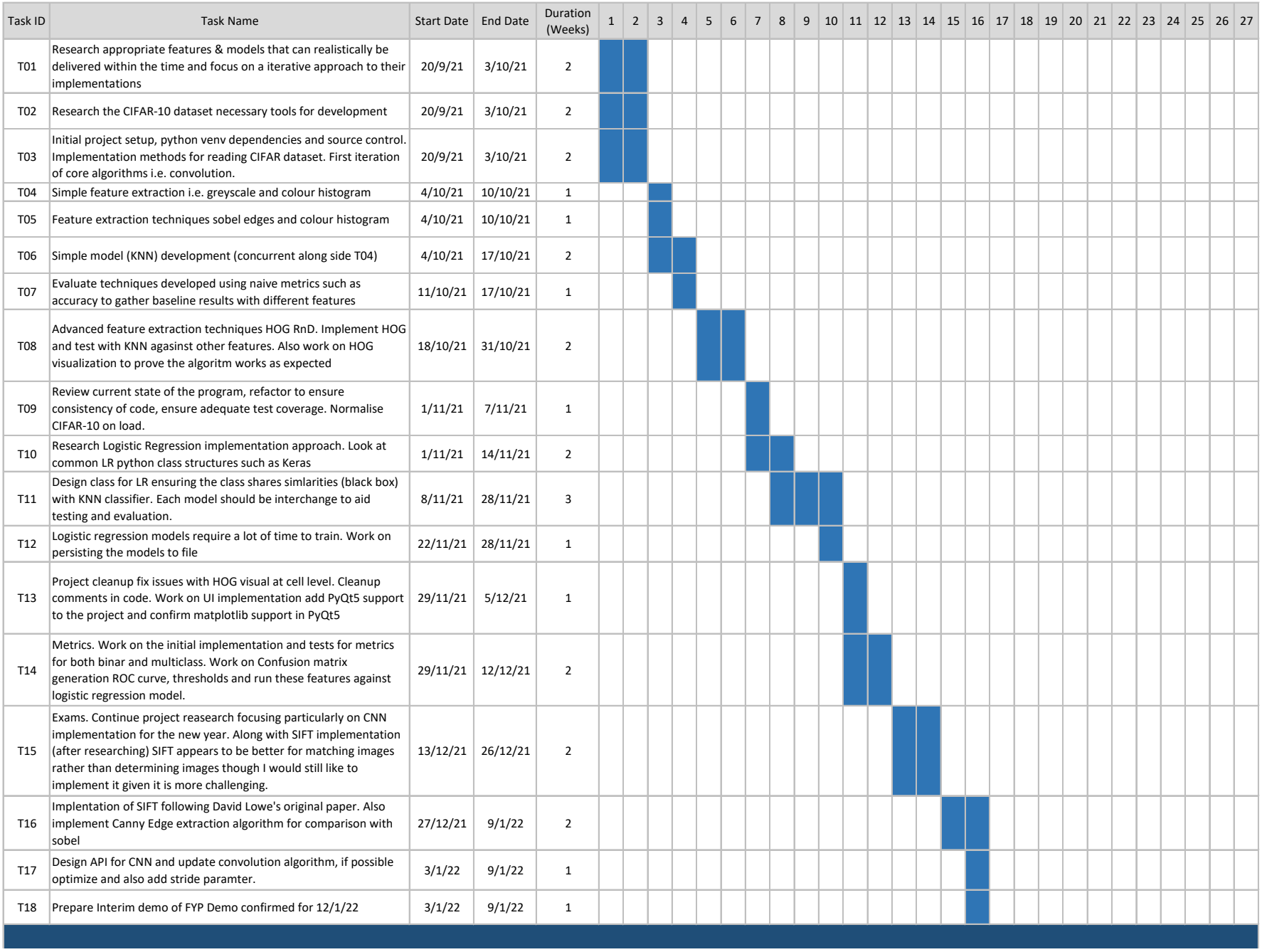


Image Classification From Stratch

FYP - Cifar Image Classifier

10-Jan-22 START DATE

06-Apr-22 END DATE

Task ID	Task Name	Start Date	End Date	Duration (Weeks)	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
T19	Further preparation for FYP interim demostration. Conduct FYP demo	10/1/22	16/1/22	1																											
T20	Review of the codebase. Do all features developed still have a place in the project? Is there common functionality in features that could be extracted into seprate units. Are there any aspects of the project which are not functioning and are contributing to technical debt.	17/1/22	30/1/22	2																											
T21	Futher research into CNN particularly back propagration and whether the existing gradient decsent is applicable to CNN, does this cause a refactor of LR classifier	17/1/22	23/1/22	1																											
T22	Further work on Metrics code is necessary as there seems to be issues with metrics in multiclass runs with LR	24/1/22	30/1/22	1																											
T23	CNN implementation, along with comparisions to LR models and metrics, exploring different features and experimenting with different layers	24/1/22	6/3/22	6																											
T24	Begin dissertation. Research format of dissertation along with information from CSC-3002. Project Introduction and background information on CIFAR	7/2/22	6/3/22	4																											
T25	Review of the current system focusing on UI and UX test coverage	31/1/22	13/2/22	2																											
T26	UI development. By this point I imagine UI will be the most lacking aspect of the system. Adding as a task to allow for further development time if necessary	7/2/22	20/2/22	2																											
T27	Review of feature extraction methods. Are they robust. Can they be visualized in a meaningful way that demostrates that the implementations work. Can FE methods be tweaked via UI parameters, can they be optimised. Review of models can they be evalutated in a way which lends itself to comparison in the dissertation. Are metrics on par with expected results from researched experiments?	21/2/22	6/3/22	2																											
T28	Continue the dissertation to the end of the project. Submission 1st April	7/3/22	27/3/22	3																											
T29	Possible development time, should bugs / issues arise whilst writing the report.	7/3/22	27/3/22	3																											
T30	Prepartion of the demo video. Obtain video for early parts of the system on a continual basis. Submission 6th April	7/2/22	27/3/22	7																											
T31	Final Week and Submission	28/3/22	10/4/22	2																											

References

[1] Barz, B. and Denzler, J., 2020. Do we train on test data? purging cifar of near-duplicates. *Journal of Imaging*, *6*(6), p.41.
[2] Y. Abouelnaga, O. S. Ali, H. Rady and M. Moustafa, "CIFAR-10: KNN-Based Ensemble of Classifiers," 2016 International Conference on Computational Science and Computational Intelligence (CSCI), 2016, pp. 1192-1195, doi: 10.1109/CSCI.2016.0225.
[3] LoweDavid, G. “Distinctive Image Features from Scale-Invariant Keypoints.” International Journal of Computer Vision (2004): n. pag.
[4] Pykes, K., 2020. *Structuring Machine Learning projects*. [online] Medium. Available at: https://towardsdatascience.com/structuring-machine-learning-projects-be473775a1b6