Manchester Metropolitan University

ASSIGNMENT COVER SHEET

Unit title:	6G6Z1105: Artificial Intelligence
Assignment set by:	John Darby and Nicholas Costen
Assignment ID:	1CWK50
Assignment title:	Coursework
Assessment weighting:	50%
Type: (Group/Individual)	Individual
Hand-in deadline:	See Moodle
Hand-in format and mechanism:	Moodle

Learning outcomes being assessed:

LO1: Analyse a real-world problem and select an appropriate combination of algorithms, building blocks and techniques from AI to compose a solution.

LO2: Demonstrate the capability of capturing knowledge and preparing it in a suitably revised form for creating an AI classifier.

LO3: Appraise and evaluate theoretical and practical issues underpinning AI and justify design choices for AI problem solving strategies.

LO4: Design, execute and evaluate an experimental plan to create and optimise a small real-world system incorporating AI techniques.

Note: it is your responsibility to make sure that your work is complete and available for marking by the deadline. Make sure that you have followed the submission instructions carefully, and your work is submitted in the correct format, using the correct hand-in mechanism (e.g. Moodle upload). If submitting via Moodle, you are advised to check your work after upload, to make sure it has uploaded properly. Do not alter your work after the deadline. You should make at least one full backup copy of your work.

Penalties for late hand-in: see Regulations for Undergraduate Programmes of Study (https://www.mmu.ac.uk/academic/casqe/regulations/assessment-regulations.php). The timeliness of submissions is strictly monitored and enforced.

All coursework has a late submission window of 5 working days, but any work submitted within the late window will be capped at 40%, unless you have an agreed extension. Work submitted after the 5-day window will be capped at zero, unless you have an agreed extension.

Please note that individual tutors are unable to grant extensions to coursework.

Exceptional Factors affecting your performance: see Regulations for Undergraduate Programmes of Study (https://www.mmu.ac.uk/academic/casqe/regulations/assessment/docs/ug-regs.pdf). For advice relating to exceptional factors, please see the following website: https://www2.mmu.ac.uk/student-case-management/guidance-for-students/exceptional-factors/ or visit a Student Hub for more information.

Plagiarism: Plagiarism is the unacknowledged representation of another person's work, or use of their ideas, as one's own. Manchester Metropolitan University takes care to detect plagiarism, employs plagiarism detection software, and imposes severe penalties, as outlined in the Student Handbook (http://www.mmu.ac.uk/academic/casqe/regulations/docs/policies_regulations.pdf and Regulations for Undergraduate Programmes (https://www.mmu.ac.uk/academic/casqe/regulations/assessment-regulations.php). Bad referencing or submitting the wrong assignment may still be treated as plagiarism. If in doubt, seek advice from your tutor.

As part of a plagiarism check, you may be asked to attend a meeting with the Unit Leader, or another member of the unit delivery team, where you will be asked to explain your work (e.g. explain the code in a programming assignment). If you are called to one of these meetings, it is very important that you attend.

Assessment Criteria:	Indicated in the attached assignment specification.
Formative Feedback:	Opportunities to discuss progress on the assessment will be provided during labs, drop-in sessions and tutor's office hours during the Artificial Intelligence teaching block.
Summative Feedback format:	A final summative mark and a completed marksheet containing highlighted criteria and feedback for each section will be made available via Moodle. See the attached specification for more details.

Scenario

You're working for a company who want to include a new automatic image classification feature in their software application. Your line manager has asked you to undertake an initial investigation based on a particular classifier and shortlist of image features you're already familiar with, plus an additional investigation into a new kind of image feature you haven't met before. The relevant part of her initial email is reproduced below:

> "[…] I'm particularly interested in what kind of features we should use for classification. We need respectable performance, but it doesn't have to be perfect. I'm wondering what the simplest kind of image feature we could get away with is. I probably need to see some performance metrics, along with some idea of how tricky each feature is to extract. I've drawn up an initial short-list of features I'm interested in, in order of increasing complexity:
>
> 1. Filesize-based features
> 2. Brightness-based features
> 3. Edge-based features
> 4. HOG[1]-based features
> 5. BoVW[2]-based features
> 6. CNN[3]-based features
>
> Please will you produce a Matlab live script that uses the MerchData dataset to investigate the following two issues for each of the features on my short-list:
>
> A. What kind of classification performance is it possible to get using Matlab's built-in functionality? Sticking with a single, simple classifier (please use k-NN, with k=3) and calling on any bit of built-in Matlab functionality that you need, what kind of performance is it possible to get with the MerchData dataset? Feature 1 is possibly a bit too simple, but I've thrown together a quick .csv file you can use to investigate, so hopefully it'll be quick for you to get some results either way.
> B. How difficult each feature extraction process will be for us to implement for ourselves in the main codebase? How much of the built-in Matlab functionality you use in A are you able to understand and re-implement for yourself? You should demonstrate identical results with the built-in functionality wherever possible. I'm happy for you to work in Matlab, producing your re-implementations using standalone functions (.m files), but if you would prefer, I'm also happy for you to work in any other language you like, as long your work is still called upon directly from your Matlab live script.[4] Don't worry about coding up extraction of feature 1 – I was able to do that very easily – but if you could code up the k-NN classifier instead, that would be great.
>
> I know that you haven't worked with feature 6 before, but I've done some initial investigations myself and will write to you with some further guidance[5].
>
> I would like to share your investigation with the rest of the team. They are all strong coders, but not necessarily strong on the Machine Learning side, so please ensure that your code is appropriately commented so that they can follow what you are doing, and why, at each stage of the investigation. […]"

[1] HOG = Histogram of Oriented Gradients
[2] BoVW = Bag of Visual Words
[3] CNN = Convolutional Neural Network
[4] Guidance on calling work written in other languages *from* Matlab is available here
[5] Available in Appendix A

Marking Criteria

To generate your final mark, the unit tutors will highlight the relevant band(s) and indicative language within the criteria set out below. We will also provide you with written feedback (things you may find useful for future work) and a brief overall summary of our assessment of the work. Finally, we will select an overall final mark from the University's stepped marking scheme.

Issue A (total weight 70%):

86%-100%	**All** features have been attempted. Investigations attempted are insightful and illuminating. Indicative language: Creative, insightful, illuminating, inspiring, exciting, authoritative
70%-85%	Feature 1 (filesize) **plus at least four** other features have been attempted. Investigations attempted are persuasive and convincing. Indicative language: Persuasive, sophisticated, original, reflective, ambitious, meticulous, critical, convincing, unexpected
60%-69%	Feature 1 (filesize) **plus at least three** other features have been attempted. Investigations attempted are thorough. Indicative language: Fluent, thorough, analytical, precise, rigorous
50%-59%	Feature 1 (filesize) **plus at least two** other features have been attempted. Investigations attempted are clear and careful. Indicative language: Clear, confident, consistent, thoughtful, accurate, careful, congruent, coherent
40%-49%	Feature 1 (filesize) **plus at least one** other feature has been attempted. Investigations attempted are sufficient. Indicative language: Satisfactory, sufficient, adequate, descriptive
35%-39%	Feature 1 (filesize) **or** any one other feature has been attempted. Investigations attempted are incomplete or inadequate. Indicative language: Incomplete, inadequate, inconsistent, derivative, contradictory, superficial, irrelevant, limited
20%-34%	Feature 1 (filesize) **or** any one other feature has been attempted. Investigations attempted are insufficient or extremely limited. Indicative language: Erroneous/wrong, missing, extremely limited, inappropriate, insufficient, incoherent, unstructured
0%-19%	No features have been attempted. Investigations are absent or formless. Indicative language: Absent/none, lacking, formless, detrimental
Written feedback from unit tutors:	

Issue B (total weight 30%):

86%-100%	The k-NN classifier **plus all** of the features 2-6 have been attempted. Investigations attempted are insightful and illuminating. Indicative language: Creative, insightful, illuminating, inspiring, exciting, authoritative
70%-85%	The k-NN classifier **plus at least four** of the features 2-6 have been attempted. Investigations attempted are persuasive and convincing. Indicative language: Persuasive, sophisticated, original, reflective, ambitious, meticulous, critical, convincing, unexpected
60%-69%	The k-NN classifier **plus at least three** of the features 2-6 have been attempted. Investigations attempted are thorough. Indicative language: Fluent, thorough, analytical, precise, rigorous
50%-59%	The k-NN classifier **plus at least two** of the features 2-6 have been attempted. Investigations attempted are clear and careful. Indicative language: Clear, confident, consistent, thoughtful, accurate, careful, congruent, coherent
40%-49%	The k-NN classifier **plus at least one** of the features 2-6 have been attempted. Investigations attempted are sufficient. Indicative language: Satisfactory, sufficient, adequate, descriptive
35%-39%	The k-NN classifier **or** any one other feature has been attempted. Investigations are incomplete or inadequate. Indicative language: Incomplete, inadequate, inconsistent, derivative, contradictory, superficial, irrelevant, limited
20%-34%	The k-NN classifier **or** any one other feature has been attempted. Investigations are insufficient or extremely limited. Indicative language: Erroneous/wrong, missing, extremely limited, inappropriate, insufficient, incoherent, unstructured
0%-19%	No k-NN classifier or any other feature has been attempted. Investigations are absent or formless. Indicative language: Absent/none, lacking, formless, detrimental
Written feedback from unit tutors:	

Overall final mark and feedback:

Overall feedback from unit tutors:
Final overall stepped mark:

Submission

You should submit a single .zip file (please don't use any other compression formats), containing a Matlab live script (.mlx) that brings together all your work, along with any standalone Matlab functions (.m) upon which your live script depends.

Work on issue B typically involves re-implementing a key built-in Matlab function for yourself (e.g., replacing imgradientxy() with my_imgradientxy()) and demonstrating equivalent results wherever possible. Wherever you attempt this, it is sufficient simply to comment out a call to the original built-in Matlab function and substitute the new call to your own re-implementation on the line below. The marker will uncomment your original calls as part of their investigation into the equivalence of your re-implementation. You don't need to duplicate the entire training/classification stage in order to demonstrate this equivalence explicitly.

If you complete re-implementation work in a language other than Matlab, remember that work must still be called from your main Matlab live script (guidance on calling work coded in other languages is available here). Remember that working in languages other than Matlab is not a requirement, it is just there as an *option* for anyone who would like to.

If you complete work on a Colab notebook as part of your investigation of feature 6, issue B (see also Appendix A), then your .zip file should also include your final IPython notebook file (.ipynb) with output.

Please take time to put your submission together carefully and make sure you test it thoroughly before finalising it: We recommend unzipping it into a brand new empty folder, navigating to that folder using Matlab, and checking that the live script(s) (.mlx) run(s) as expected. This is what the unit tutors will do when marking your work. It's very easy to miss including a supporting standalone function if you rush the submission process, and **no additional files can be accepted after the deadline has passed**.

Feedback

Opportunities for discussion and feedback will be made available through the various tutor-led activities during the teaching block. This will include opportunities for Q&A in the timetabled on-campus activities, and during drop-in sessions and one-to-one tutor appointments.

Your final summative feedback sheet for this assignment will consist of a copy of the marking criteria set out in this specification document, with highlighting added to indicate your level of performance on both issues A and B, along with written feedback from the unit tutors, and a final stepped mark for the overall assignment out of 100. The feedback sheet will be returned to you via Moodle.

Getting help

Help is always available to you in your weekly timetabled sessions. In addition, you can contact either member of unit staff as follows:

- Dr John Darby, j.darby@mmu.ac.uk, office E152, [Find me on MS Teams]
- Dr Nicholas Costen, n.costen@mmu.ac.uk, office E149, [Find me on MS Teams]

Appendix A

[*Note: working on this final feature will require you to conduct some independent reading and investigation (see also the links below, which you will need to follow up). However, existing knowledge of the underlying theory behind CNNs is not required for working on this final approach and you are in a position to complete it immediately, and without waiting for any later lectures/labs.*]

CNN-based features are new to both you and your line manager, but she has been doing some initial playing around in Matlab and believes that these features could potentially give the best image classification performance of all. In addition, the models used to extract the features could even be repurposed for the classification task itself, removing the need to use k-NN altogether. The relevant part of her follow-up email regarding CNNs is reproduced below:

"[…] We might not fully understand the theory behind CNNs yet, but we do both understand image convolutions, right? Well you can think of a CNN as a 'black box' for generating useful filters which we can convolve with our images to find features which are valuable for doing classification… These filters can operate at different scales within the images, e.g., some looking for smaller and more general 'low-level' features, or others looking for larger and more specific 'high-level' features…

CNNs have to learn these filters from lots of data, and they can take a long time to train :-/ But… Look how easy it is to classify a new image with a CNN someone else has trained previously… Link 1

The problem for us is that this particular CNN has been trained on a dataset that doesn't contain the objects in the MerchData dataset (or the ones that we're likely to see in our actual application domain), so it would inevitably give incorrect predictions… The same is true for most widely available pre-trained CNNs…

But… You can still use a pre-trained CNN to extract features from your images (low-level or high-level features…), even if it hasn't 'seen' the objects the images contain before, and then you can pass those features over to another classifier of your choice to do the learning: Link 2

If that approach doesn't give good enough classification accuracy, then you can even retrain a small portion of the parameters in an existing CNN based on your new dataset… So not starting again and learning all the low-level features, but perhaps looking at how those low-level features can be combined together into the high-level features of the objects in your own particular dataset. You end up with a model capable of performing the classification step as well as the feature extraction step, and because you're only estimating a small fraction of the total number of original model parameters, (re-)training times are fast (at least versus training a CNN from scratch): Link 3

Please follow up the links I've shared above to add an issue A investigation based on the resnet50() model to your Matlab live script. Your investigation should cover:

i. Using the model straight 'off-the-shelf' to try classifying one of your testing images; (Note: you will need to resize your image)
ii. Using the model to extract CNN-based features from both your training and testing images (please compare using both low-level and high-level features), and using them to evaluate classification accuracy using k-NN with k=3; (Note: calling activations() with 'OutputAs', 'rows' will avoid you having to reorientate your training data before passing it to fitcknn())
iii. Retraining the last three layers of the model so that the architecture itself classifies between the 5 different objects in the MerchData (without any need for using k-NN). (Note: you don't need to worry about freezing any other layers, or applying any extra transformations to your training images)

For issue B, I'm not convinced that re-implementing Matlab's built-in CNN functionality is viable. It would be a big task for you, and also for the engineers who port your work into the other languages we use. Instead, I'm interested in whether we could use an existing library that would be capable of supporting all of our needs on each of the different platforms we develop for (e.g., browser, Android, iOS, …). I'm particularly interested in TensorFlow. Please will you see which aspects of your issue A investigation you can reproduce using Python and Tensorflow in a simple Colaboratory (Colab) notebook (all you need is a browser and an Internet connection). You can work with any existing pre-trained model that you think is appropriate for the task; it doesn't have to be resnet50.

I'll point you to the landing page of the TensorFlow tutorials as a starting point: here; but of course feel free to approach learning more about Tensorflow in whatever way you wish.

As before, please comment your work in both Matlab and Python appropriately so that myself and the other members of the team can follow what you are doing, and why, at each stage of your investigation. […]"