Machine Learning Applications for Predicting Molecular Structures

Final Project Report – COSC320

Project Group: B

Authors: Matt Ellis, Joey Huang, Andrew McKenzie, Matt Smith, Lyam Talbot

Submitted for: COSC320/591 Information Technology Project

Due Date: 30th May 2025

Unit Coordinator: Professor Raymond Chiong

Word Count: 2347 words

Table of Contents

PROJECT TITLE 3
GROUP MEMBERS 3
CLIENT NAME AND CONTACT DETAILS 3
PROJECT OUTCOME SUMMARY 4
DESCRIPTION OF PROJECT 5
Project Usage Flowchart 6
PROJECT DIARY 7
ACTUAL VS PLANNED GANTT CHANT COMPARISON 7
GROUP MEMBER CONTRIBUTIONS 8
REQUIREMENTS AND DELIVERABLES 9
Completed: 9
Remaining: 9
Additional: 10
BUG REPORT 11
QUALITY TESTING REPORT 12
Testing Structure 12
Model Performance Overview 12
Detailed Metrics 13
Training Performance 15
Key Considerations 17
Lessons Learned 18

Table of Tables

Table 1 - Group Members 3
Table 2- Client Details 3
Table 3 - Key Events 7
Table 4 - Group Member Contributions 8

Table of Figures

Figure 1 - Project Usage Flowchart 6
Figure 2 - Project Gantt Chart Comparison 7
Figure 3 - Model Predictions Across Both Test Datasets 12
Figure 4 - Model Accuracy Across Both Test Datasets 12
Figure 5 - Confusion Matrix (0.3 Angstroms) 14
Figure 6 - Confusion Matrix (0.5 Angstroms) 14
Figure 7 - Confusion Matrix (Baseline) 15
Figure 8 - Training Loss vs Validation Loss Plot 16

Project Title

Machine Learning Applications for Predicting Molecular Structures

Group Members

Member Name	Email
Matt Ellis	mellis35@myune.edu.au
Joey Huang	chuang26@myune.edu.au
Andrew McKenzie	amcken33@myune.edu.au
Matt Smith	msmit328@myune.edu.au
Lyam Talbot	ltalbot6@myune.edu.au

Table 1 - Group Members

Client Name and Contact Details

Name:	Professor Amir Karton
Role:	Professor of Chemistry
Email:	Amir.Karton@une.edu.au
Phone:	+61 02 6773 5528

Table 2- Client Details

Project Outcome Summary

We were able to achieve reliable performance of our model for different types of organic molecules with real or ‘synthesised’ spatial coordinates of ‘scrambled’ molecules, adapted from the QM9 dataset. Our model initially performed remarkably well on our initial dataset; however, we suspected that the scrambled molecules were too ‘obvious’ for the model to predict. We then curated a new dataset of scrambled molecules whose spatial coordinates differed more subtly from those of the original QM9 molecules. This saw an initial dip in model performance, which we attempted to rectify with the addition of HOMO-LUMO energy gaps as another input feature for the model to learn from. Although we have not been able to re-achieve accuracy rates similar to what we initially achieved on our first dataset, which was approximately 90%, we were still able to achieve a considerable test accuracy of 81%.

As detailed in the Statement of Project Scope in Report 1, this project was focused on investigating and prototyping a solution to a previously unexplored problem rather than delivering a specific software program. However, one potential use is assisting in exploring the chemical space. For example, a large collection of molecular structures could be randomly generated, then run through our model to identify which are in equilibrium. The balance between precision and recall we’ve achieved (covered in detail in the Quality Testing Report section) may be useful in this context if the model is treated as a filtering tool. That is, it could significantly narrow down candidate structures, filtering out roughly 65% of non-equilibrium structures and leaving researchers with a much smaller set which should still contain the vast majority of in-equilibrium structures from the initial randomly generated set. This smaller, more manageable set could be examined more carefully to weed out the 26% or so that isn’t truly in equilibrium.

Description of Project

Our project explored the use of machine learning (ML) models to validate the equilibrium state of molecules based on their atoms' relative spatial coordinates. We then developed a Graph Neural Network (GNN) and used the QM9 dataset for its vast quantity of molecules with computationally and experimentally verified properties to train the model to take in the relative spatial coordinates and classify the molecule as either being in equilibrium or not in equilibrium.

Although the implementation of the GNN was a very simplified version of a Graph Convolutional Neural Network (GCNN) that relies heavily on some well-thought-out data processing techniques. The performance was promising.

The resulting outputs have been the model neatly packaged into a Python script, which is runnable from the Command Line Interface, to classify a molecule comprising its atom types and relative coordinates in csv format as either in equilibrium or out-of-equilibrium. Additionally, it has also resulted in a well-documented workflow for further training, investigation, or development of the model. Figure 1 below shows the two use cases in a flow chart format. For the client to run predictions, they simply need to structure the atoms and coordinates in a csv file as shown in the Client Instructions PDF and run it through the program. For further development, we have thoroughly documented a range of scripts we used to manipulate data, the Jupyter Notebooks for dataset creation and model training, and included a readme file of the steps to follow. Additionally, as shown in the lower left-hand corner of the flow chart, we have provided a script to take the csv files of molecules and convert them to space-separate plain text files for visualisation in a molecular viewer.

Project Usage Flowchart

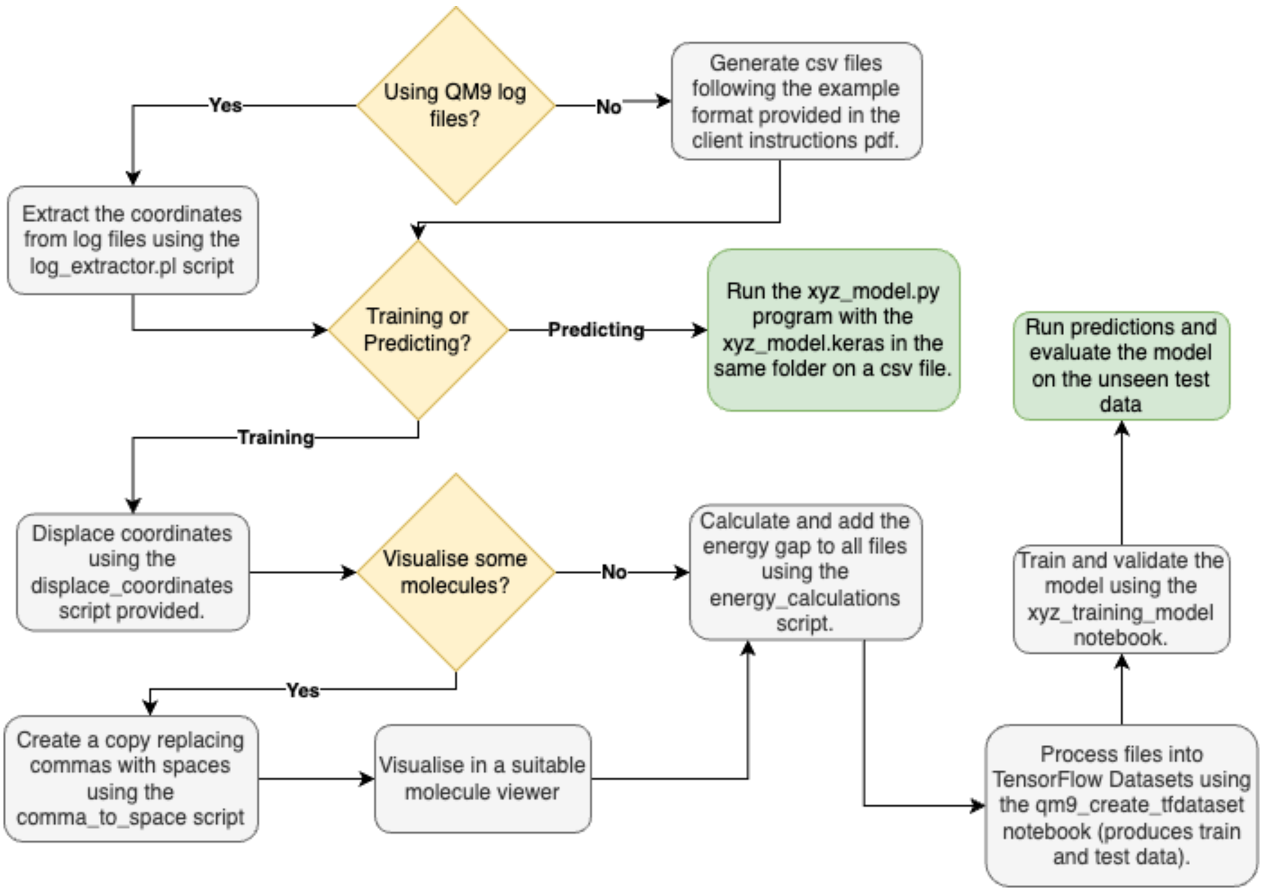

Using QM9 log files?
Yes
No
Generate csv files following the example format provided in the client instructions pdf.
Extract the coordinates from log files using the log_extractor.pl script
Training or Predicting?
Predicting
Run the xyz_model.py program with the xyz_model.keras in the same folder on a csv file.
Run predictions and evaluate the model on the unseen test data
Training
Displace coordinates using the displace_coordinates script provided.
Visualise some molecules?
No
Calculate and add the energy gap to all files using the energy_calculations script.
Train and validate the model using the xyz_training_model notebook.
Yes
Create a copy replacing commas with spaces using the comma_to_space script
Visualise in a suitable molecule viewer
Process files into TensorFlow Datasets using the qm9_create_tfdataset notebook (produces train and test data).

Figure 1 - Project Usage Flowchart

Project Diary

Date/s	Event
12th March	First meeting with Amir and Raymond.
22nd March – 9th April	Preliminary research, design concepts.
22nd March – 11th April	Preprocessing of the input data.
30th March	Submission of first report.
9th April – 16th April	Designing the model.
30th April	Submission of second report.
17th April – 8th May	Developing the model.
8th May – 13th May	Fine tuning and testing the model.
8th May – 23rd May	Drafting and finalising project report.
11th May – 23rd May	Producing documentation and client instructions.
23rd May	Project delivery.

Table 3 - Key Events

Actual vs Planned Gantt Chant Comparison

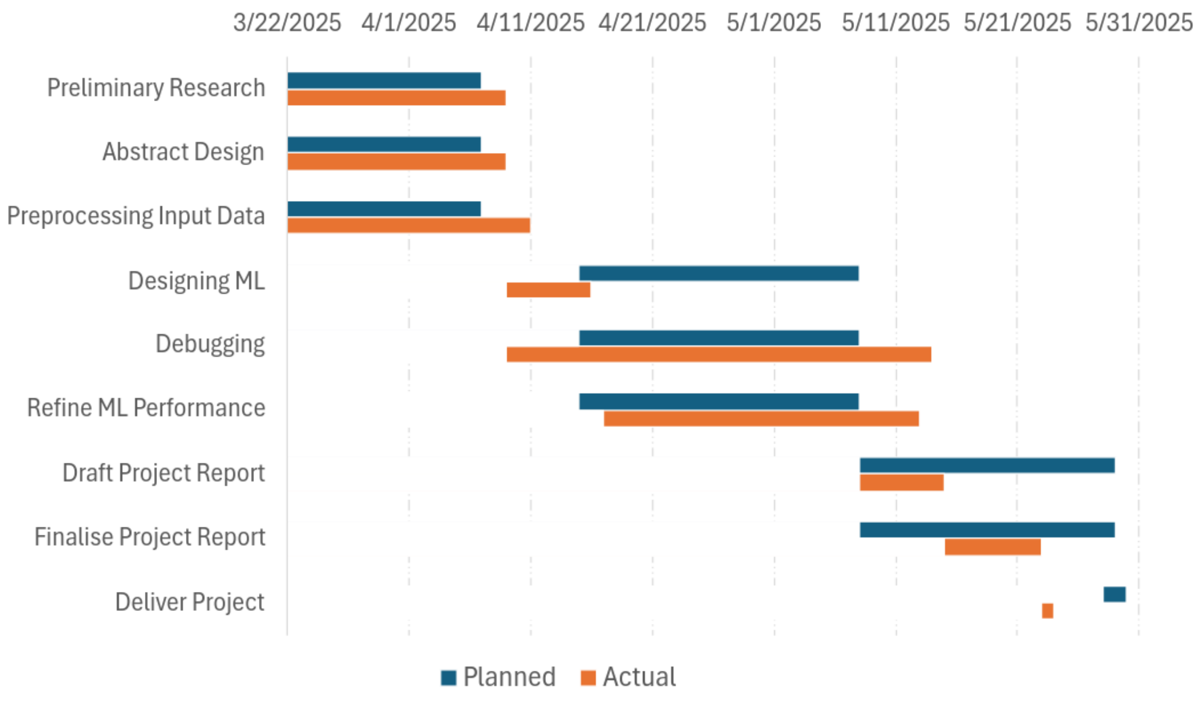


Figure 2 - Project Gantt Chart Comparison

Group Member Contributions

Name	**Contributions**
All Members	- Initial research. - Report 1. - Numerous scripts for manipulating data. - Final Report.
Matt Ellis	- Majority of report 2. - Development of deployment script. - Comparison Gantt Charts. - Model performance analysis.
Joey Huang	- HOMO-LUMO function development & implementation. - Validation dataset implementation. - Client instruction manual.
Andrew McKenzie	- Developing the initial test model. - Part of report 2. - Initial test dataset. - Report/presentation templates. - Documentation of Notebooks. - Development of deployment script.
Matt Smith	- Organising team meetings. - Editing of the final presentation video.
Lyam Talbot	- Initial Gantt chart spreadsheet. - Adjacency function development. - Model hyper-parameter refinement. - Dataset curation for HOMO-LUMO calculations of all molecules. - TensorFlow dataset curation of all molecules. - Model fine-tuning and export.

Table 4 - Group Member Contributions

Requirements and Deliverables

Completed:

- Dataset curation of QM9 molecules used for training and testing in our Tensorflow Neural Network.
- Input features and architecture of our Tensorflow Neural Network Model predicting equilibrium state, kept in xyz_model.py.
    - Develop the following input features for the neural network to predict the equilibrium state of a particular molecule:
        - Dense adjacency matrix showing the remaining valencies in the diagonals and bonds between each atom in the off-diagonal.
        - Another adjacency matrix determines the Euclidean distances between the atoms.
        - HOMO-LUMO energy gap of the molecule.
        - All have been and can be calculated simply using the spatial coordinates of the atoms within a molecule.
    - Designing and compiling the structure and layers constituting the neural network:
        - Graph convolutional network layers.
        - Dense layers.
        - Mask pooling.
        - Dropout layers.
- Perl script for extracting the atom types and spatial coordinates from an entire directory of log files and outputting them in csv format ready for input to the model.
- Handover document/s for future teams to pick up (the README.md file)
- Final testing of the saved model for data evaluation.
- Client instructions document.

Remaining:

- No requirements or deliverables outstanding.

Additional:

(Completed but not a requirement or deliverable)

- Script for generating the out-of-equilibrium molecules for training.
- Script for taking the csv files of coordinates and replacing the commas with whitespace for molecular viewing programs.

Bug Report

1. **Integrating HOMO-LUMO energy input into the neural network:**

Problem:

- Original input features simply consisted of adjacency matrices which were inputted into the network as Graph Convolution layers; no other input feature layers have used to build the neural network prior.
- Needed to figure out how to input energy, which was a scalar feature, into the network, alongside the other GCN layers.
- Tried concatenating energy values with the GCN layers but was continually rejected as it was incompatible with the existing architecture of the network, continually giving the 'incompatible shapes' error:

```
ValueError: A `Concatenate` layer requires inputs with matching shapes except for the concatenation axis. Received: input_shape=[(None, None, 32), (None, 1)]
```

Solution:

- Could not simply concatenate i.e. 'attach' the scalar energy value to the rest of the GCN layers seen in the error log given above (input_shape = [(None, None, 32), (None, 1)].
- Needed to be integrated with a separate layer on its own, separate to the GCN layers.
- Figured out to incorporate the energy values through a dense layer, which was added after the GCN layers.

```
energy_dense = Dense(8, activation='relu', name="energy_transform")(input_energy_gap)
```

- After the energy values were added as a separate layer, we could then concatenate it back and combine it with the rest of the GCN and graph layers in the network.

```
combined = Concatenate(name="combined_features")([graph_embedding, energy_dense])
```

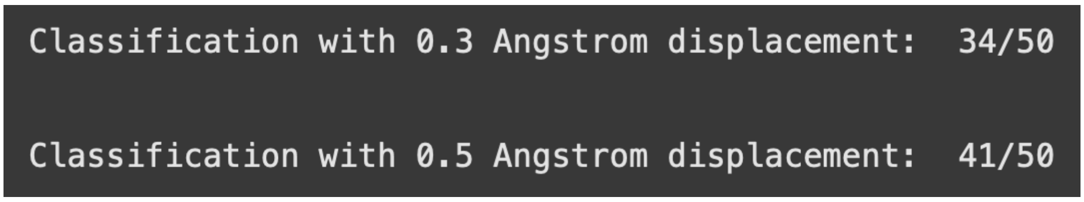
Quality Testing Report

Testing Structure

The model was trained on a dataset of 16000 molecules from the QM9 dataset in addition to another equally sized group for which random atoms' coordinates were displaced by a total of 0.3 angstroms per molecule. It was then tested on two sets of molecules, each run through the same script to be displaced by either 0.3 or 0.5 angstroms (4000 and 3974 molecules, respectively), to observe performance on different levels of deviation from equilibrium structures.

Model Performance Overview

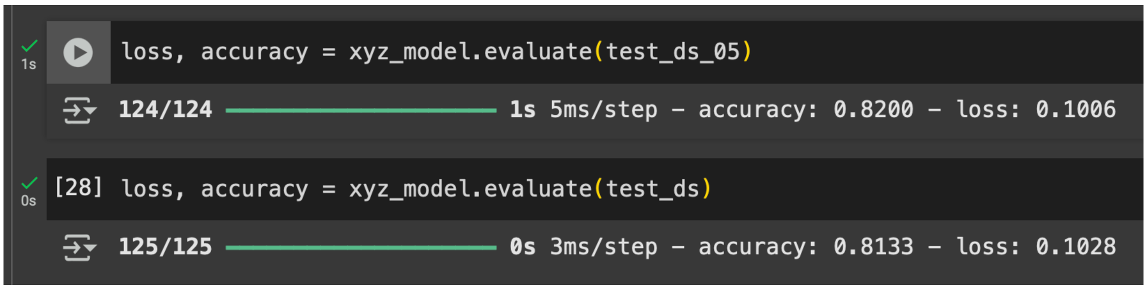
Overall, the model has shown promising results with approximately an 80% accuracy rate on the unseen test data. Comparing the two test sets, the model performs slightly better on the 0.5 displacement set (refer to figure 3 below) when using a small number of samples. However, as the number of samples grows it converges to a relatively equal performance on both sets (refer to figure 4 below).



```
Classification with 0.3 Angstrom displacement:  34/50

Classification with 0.5 Angstrom displacement:  41/50
```

Figure 3 - Model Predictions Across Both Test Datasets



```
1s      loss, accuracy = xyz_model.evaluate(test_ds_05)
        124/124 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8200 - loss: 0.1006

0s [28] loss, accuracy = xyz_model.evaluate(test_ds)
        125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8133 - loss: 0.1028
```

Figure 4 - Model Accuracy Across Both Test Datasets

Detailed Metrics

The test set is a mix of equilibrium and out-of-equilibrium molecules, and while the model's overall accuracy is around 80%, the individual correct classification rate for each of these two types of molecules varies.

For both test sets, around 74% of the molecules classified as in-equilibrium actually are, but 26% are false positives. On the other hand, for both sets about 99.5% of molecules the model decides are out-of-equilibrium are so. In other words, we can be very confident when the model identifies a molecule as out-of-equilibrium, and the group identified as in-equilibrium will include practically every truly in-equilibrium molecule (over 99.6%). However, this comes at the cost of being a bit lenient with what the model considers in-equilibrium. To an extent, this lenient behaviour is somewhat desirable in this context, as molecules whose properties only deviate slightly from equilibrium structures and values are not necessarily out-of-equilibrium.

Figures 5 through 7 compare the number of false positive, true positive, false negative and true negative predictions across the two test sets and a baseline where predictions are made on the 0.3 displacement by a model which always classifies molecules as in-equilibrium (the label '1' corresponds to 'in-equilibrium'). Overall, our model performs substantially better than the baseline. This can be quantified using an F1-score, which is roughly 0.662, 0.846 and 0.849 for the baseline, 0.3 set, and 0.5 set respectively.

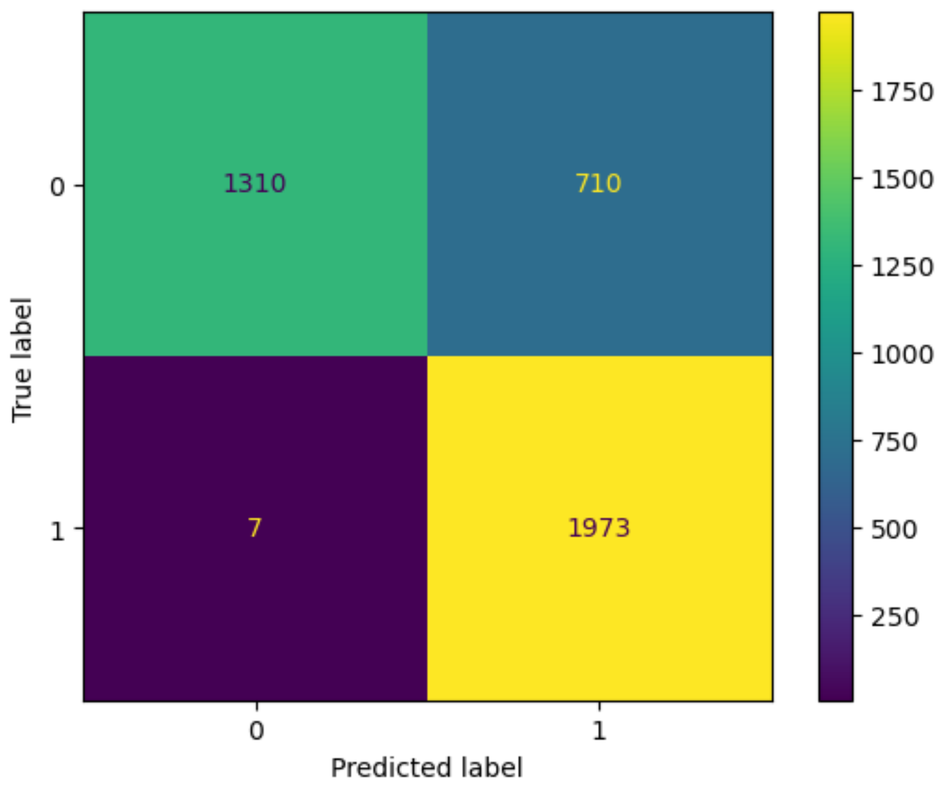


Figure 5 - Confusion Matrix (0.3 Angstroms)

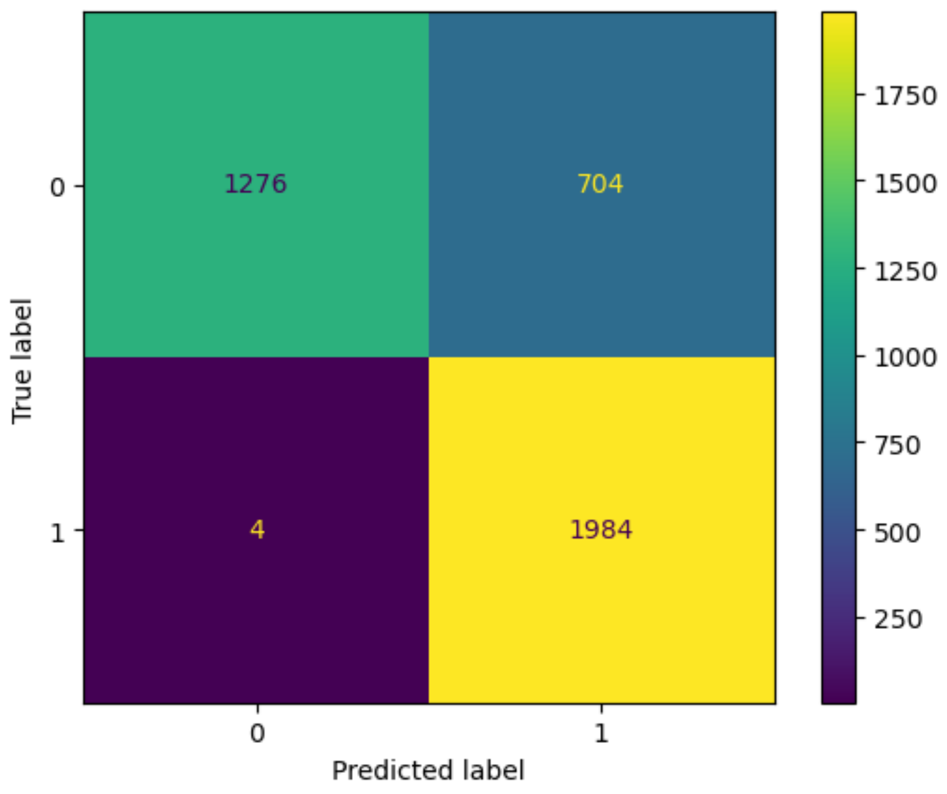


Figure 6 - Confusion Matrix (0.5 Angstroms)

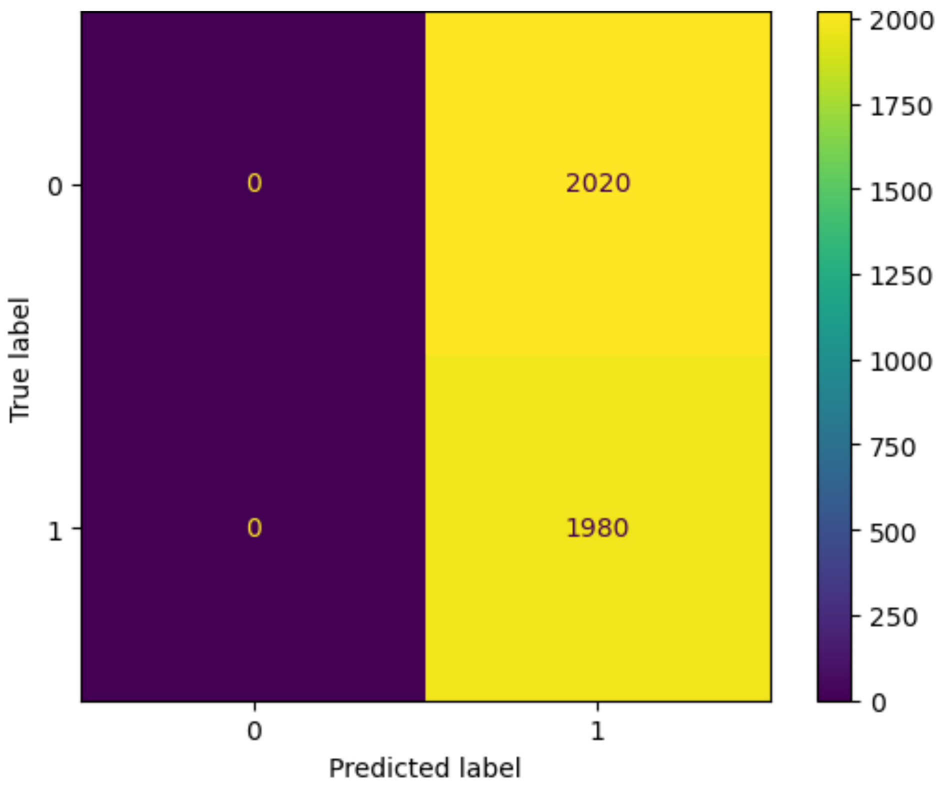


Figure 7 - Confusion Matrix (Baseline)

Training Performance

Figure 8 below shows the training loss vs validation loss. The loss is a measure of the error rate of the model, where our goal is to minimise the loss value as much as possible.

The model firstly learns the features that indicate the equilibrium state using the dataset of molecules allocated to our training subset. At the end of each training cycle, its performance is then assessed against a small validation subset of molecules from our dataset that it hasn't seen yet, to determine how accurately the model's predictions are on the equilibrium state of unseen molecules using the features it has learned during training. A sign that the model is not learning important features and patterns of the dataset is when validation accuracy stagnates, even if training accuracy improves.

From Figure 8 below, it can be seen that our model was quick to learn, evident in the steep reduction in training and validation loss until after cycle four, where it then ceased to improve. Our model consistently performed better on the validation set than on our training data, possibly due to two main reasons:

- Our dropout layer removes some weights and biases from our network during training to help prevent our model from overfitting, i.e. 'memorising' the data. However, these weights and biases are reintroduced back into the model during validation, which may have boosted its performance on validation data.
- Another plausible explanation might be due to the significantly smaller size of the validation dataset that may not fully represent of all the molecules that the model needed to learn during training, making predictions easier.

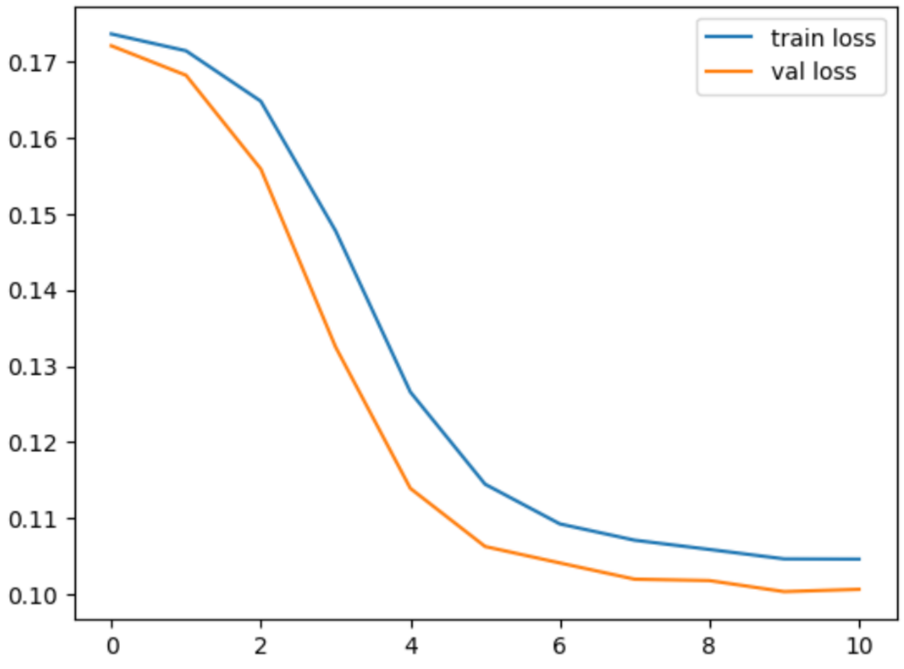


Figure 8 - Training Loss vs Validation Loss Plot

Key Considerations

Based on these results, a key point that needs to be considered is the fact that there is currently very little understanding of what the model has learnt to base its predictions on. Given the type of data, investigating this would be a very time-consuming task, but equally important. Any trends in what the model can and can't predict will shed light on not only whether the model is generalising well, but they will also provide critical insights into what types of molecular structures might be underrepresented in the training data.

It is also worth noting that the model is currently restrained to an input limit of 26 atoms for any molecule, and only five different types of atoms are represented in the QM9 dataset. Although this can be increased reasonably easily, it will also increase the computational cost of training.

Lessons Learned

1. Large datasets are very hard to handle and coordinate using OneDrive; possibly a shared Google Drive account would have helped for Colab, but a shared solution for Turing would have been needed as well.
2. Version control was difficult, as well as knowing which version of the model was the most recent and integrating changes from different members. Using a version control system like Git with GitHub would have made this easier.
3. Researching a new technical technique within your own field while also researching a topic within a new field of science is very difficult to perform simultaneously, ideally, this would be split into first researching the problem and developing a better understanding of the field before considering any technical solutions.
4. Data preparation, curation, and verification are more time-consuming and important than expected. The quality is critical to validate results and needs its own research and preparation time allocated.