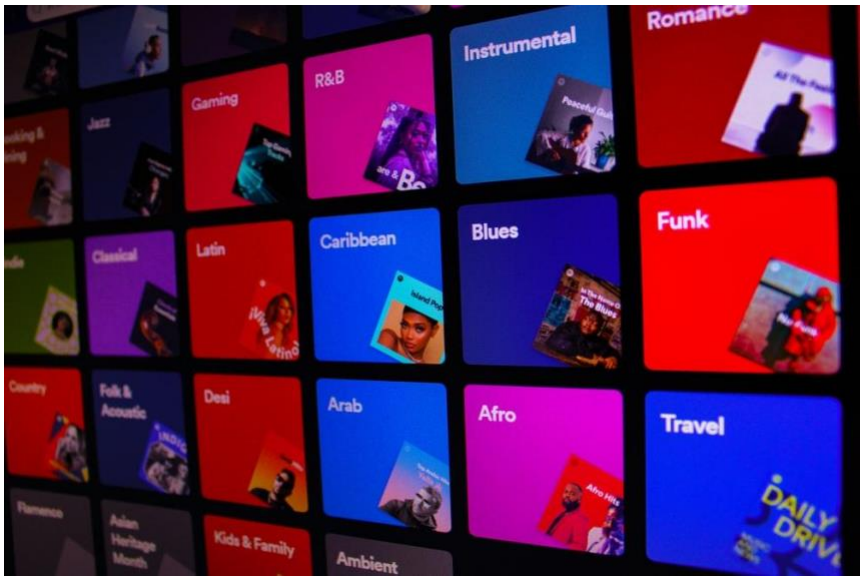
Assignment 1

CE/CZ4042: Neural Networks and Deep Learning

Deadline: 11th October 2021

- This assignment is to be done **individually**.
- Data files for both parts are found in the folder 'Assignment 1' under 'Assignments' on NTULearn. You can use **starter codes start_1a.ipynb and start_1b.ipynb** to begin the assignment.
- Complete both parts A and B of the assignment and submit a report and source codes online via NTULearn before the above-mentioned deadline.
    - The report should contain **all experiment results** (answers to questions) as well as **a conclusion to summarise your findings**.
    - The assessment will be based on both the project report and the correctness of the codes.
- Maximum score for this assignment is 100 marks.
    - 90 marks are allocated for answering the questions: 45 marks each for parts A and B.
    - 10 marks are allocated for quality of presentation in both the report and source codes. This includes:
        - Clarity (2 marks): plots are well-annotated with appropriate title, axes and legend, codes are well-organised and annotated with comments wherever necessary / docstrings in functions
        - Conciseness (2 marks): answers to open-ended questions are on point, only code that are used to generate relevant results are retained in the submission (i.e., remove excessive code)
        - Depth of discussion in conclusion (6 marks, see pointers below)
- The report and source codes should be submitted in the following format:
    - lastname_firstname_A1_report.pdf (report in pdf format); and
    - lastname_firstname_A1_codes.zip (containing all source codes).
- Late submissions will be penalized: **5% for each day up to three days**.
- TAs Mr. Chan Yi Hao and Ms. Charlene Ong are in charge of this assignment. Please email neuralnetworks4042@gmail.com for any queries or issues regarding the assignment. You can also arrange for consultation via Calendly https://calendly.com/neuralnetworks4042.

Part A: Classification Problem (45 marks)



Part A of this assignment aims at building neural networks to classify the GTZAN dataset, which is obtained from (http://marsyas.info/downloads/datasets.html). The GTZAN dataset is a widely used dataset collected from 2000 to 2001 from multiple sources [1]. The original dataset consisted of 1000 audio tracks, spanning 30 seconds each. There are 10 different genres in total. For the purpose of this assignment, we will be using a pre-processed dataset where these audio tracks have been processed into features.

The pre-processed CSV file containing the features is obtained from:
https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification.
Optionally, you can download the audio files from the website to listen to the audio tracks.
The audio files will not be used in this assignment.
You can also visit the Kaggle dataset provider's work on this dataset here:
https://www.kaggle.com/andradaolteanu/work-w-audio-data-visualise-classify-recommend

We will be using the CSV file named **features_30_sec.csv,** which is both provided to you and can also be found on Kaggle. The 30 seconds long audio files consist of 57 features engineered by the dataset owner. For more details about the audio features, refer to Andersson [3]. Explanation of some of the features is provided in the table on the next page.

The aim is to predict the genre of the corresponding audio files in the test dataset after training the neural network on the training dataset. The genres are blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock.

Read the data from the file: **features_30_sec.csv.** Each data sample is a row of 60 columns, which consist of filename (where you can find the original audio file if you like), length of audio, label, and the 57 features which you will use.

Tip: You can use the sample code given in file **start_1a.ipynb** to do pre-processing of data.

Type of features	**Explanation**
Chroma (e.g. chroma_stft_mean)	Describes the tonal content of a musical audio signal in a condensed form (Stein et al, 2009) [2]
Rms (e.g. rms_mean)	Square root of average of a squared signal (Andersson) [3]
Spectral (e.g. spectral_centroid_mean)	Spectral Centroid is a metric of the centre of gravity of the frequency power spectrum (Andersson) [3]
Rolloff (e.g. rolloff_mean)	Spectral rolloff is a metric of how high in the frequency spectrum a certain part of energy lies (Andersson) [3]
Zero crossing (e.g. zero_crossing_mean)	Zero-crossing rate is the number of time domain zero-crossings within a processing window (Andersson) [3]
Harmonics (e.g. harmony_mean)	Sound wave that has a frequency that is a n integer multiple of a fundamental tone *Refer to link: https://professionalcomposers.com/what-are-harmonics-in-music/*
Tempo	Periodicity of note onset pulses (Alonso et al, 2004)
MFCC (Mel Frequency Cepstral Coefficient)	Small set of features (usually about 10-20) which concisely describe the overall shape of a spectral envelope *Refer to link:* *https://musicinformationretrieval.com/mfcc.html*

Question 1

Design a feedforward deep neural network (DNN) which consists of an input layer, one hidden layer of 16 neurons with ReLU activation function, and an output softmax layer. Use an stochastic gradient descent with 'adam' optimizer with default parameters, and batch size = 1. Apply dropout of probability 0.3 to the hidden layer.

Divide the dataset into a 70:30 ratio for training and testing. Use appropriate scaling of input features. We solely assume that there are only two datasets here: training & test. We would look into validation in Question 2 onwards.

Parts	**Marks**
a) Use the training dataset to train the model for **50 epochs. Note: Use 50 epochs for subsequent experiments.**	6
b) Plot accuracies on training and test data against **training epochs** and comment on the plots.	2
c) Plot the losses on training and test data against **training epochs.** State the approximate number of epochs where the test error begins to converge.	2
	Total: 10

Question 2

In this question, we will compare the performance of the model **using stochastic gradient descent** and **mini-batch gradient descent**, as well as determining the **optimal batch size** for mini-batch gradient descent. Find the optimal batch size for mini-batch gradient descent by training the neural network and evaluating the performances for different batch sizes. **Note: Use 3-fold cross-validation on training partition to perform parameter selection.**

Parts	**Marks**
a) Plot mean cross-validation accuracies over the training epochs for different batch sizes. Limit search space to batch sizes {1,4,8,16,32, 64}.	3
b) Create a table of median time taken to train the network for one epoch against different batch sizes. (Hint: Introduce a callback)	3
c) Select the optimal batch size and state reasons for your selection.	3
d) What is the difference between mini-batch gradient descent and stochastic gradient descent and what does this mean for model training?	2
e) Plot the train and test accuracies against epochs for the optimal batch size.	2
Note: **use this optimal batch size for the rest of the experiments**.	Total: 13

Question 3

Find the optimal number of hidden neurons for the **2-layer network (i.e., one hidden layer)** designed in **Question 1 and 2.**

Parts	**Marks**
a) Plot the cross-validation accuracies against training epochs for different numbers of hidden-layer neurons. Limit the search space of the number of neurons to {8, 16, 32, 64}. Continue using 3-fold cross validation on training dataset.	3
b) Select the optimal number of neurons for the hidden layer. State the rationale for your selection.	2
c) Plot the train and test accuracies against training epochs with the optimal number of neurons.	2
d) What other parameters could possibly be tuned?	2
Note: **use this optimal number of neurons for the rest of the experiments**.	Total: 9

Question 4

After you are done with the 2-layer network, design a **3-layer network with two hidden-layers** with ReLU activation, each consisting of the optimal number of neurons you obtained in Question 3, (apply a dropout with a probability of 0.3 for each hidden layer), and train it with a batch size of 1.

Parts	**Marks**
a) Plot the train and test accuracy of the 3-layer network against training epochs.	6
b) Compare and comment on the performances of the optimal 2-layer network from your hyperparameter tuning in Question 2 and 3 and the 3-layer network.	2
	Total: 8

Question 5 (let's dig deeper!)

We are going to dissect the purpose of dropout in the model.

Parts	**Marks**
a) Why do we add dropouts? Investigate the purpose of dropouts by removing dropouts from your **original** 2-layer network (before changing the batch size and number of neurons). Plot accuracies on training and test data with neural network without dropout. Plot as well the losses on training and test data with neural network without dropout.	3
b) Explain the effect of removing dropouts.	1
c) What is another approach that you could take to address overfitting in the model?	1
	Total: 5

Possible discussion pointers for conclusion

Besides summarising the key findings from each question, take a step back to analyse the entire modelling pipeline and think about ways to improve it. Here are some aspects of the pipeline that you can consider:

- We now have a classifier that predicts the genre of audio files based on features obtained from processing these audio tracks. What are some limitations of the current approach (using FFNs to model such engineered features)?
- Out of the parameters that were tuned, which was most impactful in terms of improving the model performance and what could be some reasons for that?
- Considering that audio tracks are originally waveforms, what are some alternative approaches to achieve the goal of genre classification? What kind of neural network architectures will be used instead?
- What other datasets and tasks can this approach of modelling waveform data be used for? What changes to the pipeline, if any, will you have to make when approaching these problems?
- **You are encouraged to include your own pointers!**

Part B: Regression Problem (45 marks)

In Singapore, resale prices of Housing Development Board (HDB) flats[1] have been on the rise over the past year. The HDB Resale Price Index is inching towards the all-time high previously made in April 2013. It was claimed that the price increase has been a broad-based one but we want to analyse the data more deeply to see if there are other factors behind this increase.

Thus, we have two goals in part B: (i) perform retrospective[2] prediction of HDB housing prices, (ii) identify the most important features that contributed to the prediction.

This assignment uses publicly available data on HDB flat prices in Singapore, obtained from data.gov.sg on 5th August 2021. The original set of features have been modified and combined with other datasets to include more informative features, as explained on the next page.

Important notes:

- **Do not** download the latest data from the website. Please use the dataset provided to you via NTULearn as it contains additional features derived from other datasets.
- Data cleaning has already been performed. You are **not expected to include any more data cleaning steps**. Modelling (and analysis of results) is the focus of this assignment.
- In the sample code given to you, the seed has been set. **Do not remove the seed**. If you choose not to use the sample code, make sure you set the seed to 42. Refer to the sample code to see how the seed should be set at the start of the script.
- The neural network used in this part is small and does not require GPU. You should be able to run the analysis on your own machines without GPU, or on Google Colab. Thus, **do not use any GPUs** to run your analysis (because CUDA has non-deterministic operations that cannot be turned off, preventing your work from being reproducible).
- **Sample code is given in file 'start_1b.py'** to help you get started with this problem.

[1] HDB flats refer to public housing in Singapore, where a large majority of the population lives in.

[2] Note that this exercise does not make use of temporal information to predict future prices, since we have not covered models that can model sequential data (i.e. time series analysis via Recurrent Neural Networks).

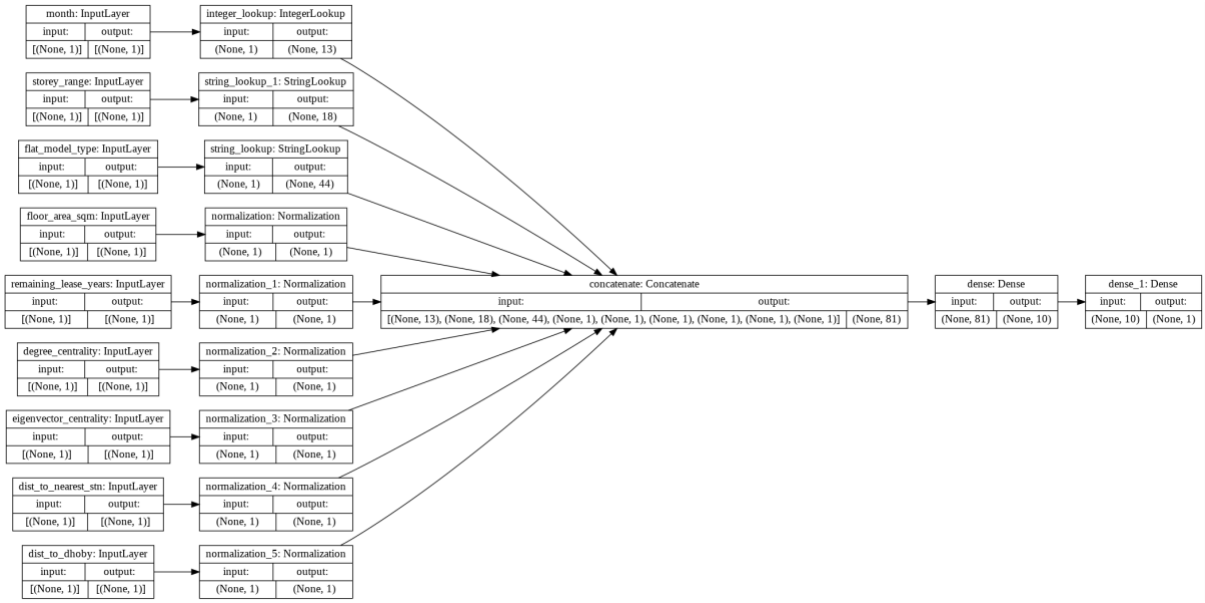
Feature	Type	Explanation
month	Categorical (Integer)	Which month the resale transaction was performed.
year	Categorical (Integer)	Which year the resale transaction was performed. Used to split the dataset into train and test sets. **NOT used to train the model**.
full_address	Categorical (String)	Address of the flat. **Not used** for modelling as other metrics derived from it are used instead (dist_to_nearest_stn, dist_to_dhoby).
nearest_stn	Categorical (String)	Closest MRT station to the flat. **Not used** for modelling as other metrics derived from it are used instead (degree_centrality, eigenvector_centrality).
dist_to_nearest_stn	Numeric	Distance from the flat to the nearest MRT station, in kilometres. Computed via latitude and longitude. Flats near MRT stations tend to fetch higher prices.
dist_to_dhoby	Numeric	Distance from the flat to Dhoby Ghaut MRT station, in kilometres. Computed via latitude and longitude. Dhoby Ghaut is chosen as it is centrally located. Flats in the Central region are typically more costly.
degree_centrality	Numeric	A metric (computed for the MRT station closest to the flat) that represents the degree of the node, i.e., how many edges are connected to the node. (Rationale: flats near 'interchange' stations - stations with more than 1 MRT line - are likely to be more well connected / offer more transport options and thus have higher value. Stations in the central areas tend to have more than 1 MRT line too).
eigenvector_centrality	Numeric	A more global metric than degree_centrality as it captures neighbourhood information. When eigenvector centrality of a node is high, the nodes adjacent to it are likely to have high values too.
flat_model_type	Categorical (String)	Type of flat. See this reference for more details. You're not expected to understand all flat types.
remaining_lease_years	Numeric	HDB flats are originally sold by HDB with a 99-year lease. Generally, with other variables held constant, flats with higher remaining lease will fetch a higher value. The original data was stored in years and months – this was turned into a scalar by converting it into months and dividing that value by 12.
floor_area_sqm	Numeric	Size of the flat in square meters. Generally, larger houses are more expensive.
storey_range	Categorical (String)	Which floor the flat is at. Generally, the higher the flat is, the more expensive it will be.
resale_price	Numeric	Flat prices in Singapore Dollars. **Target to predict.**

Question 1

Real world datasets often contain a mix of numeric and categorical features – this dataset is one such example. Modelling such a mix of feature types with neural networks requires some modifications to the input layer. This tutorial from the Keras documentation guides you through the process of using the Functional API to do so.

Parts	**Marks**
a) Divide the dataset ('HDB_price_prediction.csv') into train and test sets by using entries from year 2020 and before as training data (with the remaining data from year 2021 used as test data). Why is this done instead of random train/test splits?	2
b) Following this tutorial, design a 2-layer feedforward neural network consisting of an input layer, a hidden layer (10 neurons, ReLU as activation function), and a linear output layer. One-hot encoding should be applied to categorical features and numeric features are standardised. After encoding / standardisation, the input features should be **concatenated**. The input layer should use these features: - **Numeric** features: dist_to_nearest_stn, dist_to_dhoby, degree_centrality, eigenvector_centrality, remaining_lease_years, floor_area_sqm - **Categorical** features: month, flat_model_type, storey_range Your architecture should resemble the figure shown on the next page.	5
c) On the training data, train the model for 100 epochs using mini-batch gradient descent with batch size = 128, Use 'adam' optimiser with a learning rate of $\alpha = 0.05$ and mean square error as cost function. (Tip: Use smaller epochs while you're still debugging. On Google Colaboratory, 100 epochs take around 10 minutes even without GPU.)	3
d) Plot the train and test **root** mean square errors (RMSE) against epochs (Tip: **skip the first few epochs**, else the plot gets dominated by them).	2
e) State the epoch with the lowest test error. State the test R^2 value at that epoch. (Hint: Check the output returned by model.fit(). Use a custom metric for computing R^2.)	3
f) Using the model from that best epoch, plot the predicted values and target values for a batch of 128 test samples. (Hint: Use a callback to restore the best model weights. Find out how to retrieve a batch from tf.BatchDataset. A scatter plot will suffice.)	5

Total: 20



Question 2

Instead of using one-hot encoding, an alternative approach entails the use of embeddings to encode categorical variables. Such an approach utilises the ability of neural networks to learn richer representations[3] of the data – an edge it has over traditional ML models.

Parts	**Marks**
a) Add an Embedding layer with output_dim = floor(num_categories/2) **after** the one-hot embeddings for **categorical** variables. (Hint: Use the tf.keras.layers.Embedding() later. Read the documentation carefully to ensure that you define the correct function parameters[4].)	4
b) The Embedding layer produces a 2D output (3D, including batch), which cannot be concatenated with the other features. Look through the Keras layers API to determine which layer to add in, such that all the features can be concatenated. Train the model using the same configuration as Q1. (Tip: A full run takes ~15 mins, so reduce epochs when debugging your code but remember to switch it back to 100.)	3
c) Compare the current model performances in terms of both test RMSE and test R^2 with the model from Q1 (at their own best epochs) and suggest a possible reason for the difference in performance.	3
	Total: 10

[3] Instead of just cramming all the information about a category into a number, a matrix has more capacity to encode more meaningful relationships among the categories, e.g. the embeddings of older flat types could possibly be close together while having a large distance from newer flat types.

[4] In the Keras tutorial, the function lookup_class (which actually is either the StringLookup or the IntegerLookup function) accepts a parameter 'output_mode=binary'. This actually performs one-hot encoding, i.e. gives the same results as 'output_mode=one_hot'. You can verify this by checking the output shapes.

Question 3

Recursive feature elimination (RFE) is a feature selection method that removes unnecessary features from the inputs. It can also shed some insights on how much each feature contributes to the prediction task.

Parts	Marks
a) Continue with the model architecture you have after Q2. Via a callback, introduce early stopping (based on val_loss, with patience of 10 epochs) to the model.	2
b) Start by removing one input feature whose removal leads to the minimum drop (or maximum improvement) in performance[5]. Repeat the procedure recursively on the reduced input set until the optimal number of input features is reached[6]. Remember to remove features one at a time. Record the RMSE of each experiment neatly in a table (i.e., without feature 1, without feature 2, etc.). (Hint: Use a binary vector mask to keep track of the features. When you remove a feature, you do not have to repeatedly remove the initialisation of the input layers for each feature. Just choose which to include when you concatenate the features. Make sure to **clear the session** at every iteration of feature elimination. A full run take ~2hrs.)	8
c) Compare the performances of the model with all 9 input features (from Q2) and the best model arrived at by RFE, in terms of both RMSE and R^2.	2
d) By examining the changes in model performance whenever a feature is removed, evaluate the usefulness of each feature for the task of HDB resale price prediction.	3
	Total: 15

Possible discussion pointers for conclusion

Besides the discussion pointers mentioned in Part A,

- From RFE, we have an idea of which features are (un)important when performing the prediction, but this was done for the entire dataset. How do we make use of this information to find out what could be the factors that lead to the price increase?
- **Feel free to include your own pointers**, but try to focus on modelling-related issues instead of what other data can be added (e.g. mature / non-mature estates, adding the latest MRT lines / bus information / amenities (schools, market, malls, etc).

[5] Given k features, to determine which of the k features will cause the minimum drop / maximum increase when that feature is removed, you will have to perform k experiments. After removing that feature, k-1 features will be left and you will have to perform k-1 experiments to determine the next feature to remove.

[6] The feature removal goes on until either 1 feature is left, or the model performance does not improve from the previous best (e.g. when there are 7 features left, if none of the 7 experiments performed does better than the best performance of the model with 8 features, the RFE algorithm terminates).

References

[1] Tzanetakis G, Cook P. Musical genre classification of audio signals. IEEE Transactions on speech and audio processing. 2002 Nov 7;10(5):293-302.

[2] Stein M, Schubert BM, Gruhne M, Gatzsche G, Mehnert M. Evaluation and comparison of audio chroma feature extraction methods. InAudio Engineering Society Convention 126 2009 May 1. Audio Engineering Society.

[3] Andersson T. Audio classification and content description. 2004.

[4] Miguel Alonso BD, Richard G. Tempo and beat estimation of musical signals. In Proceedings of the International Conference on Music Information Retrieval (ISMIR), Barcelona, Spain 2004.