Assignment 1

CE/CZ4042: Neural Networks and Deep Learning

Deadline: 10th October 2022

- This assignment is to be done **individually**. You can discuss it with your classmates, but **your submission must be your own unique work**.
- Data files and other supporting scripts 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 (i) report, (ii) correctness of the codes.
- The maximum score for this assignment is 100 marks.
  - 90 marks for answering the questions: 45 marks each for parts A and B.
  - 10 marks for quality of presentation in both the report and source codes.
    - Clarity (2m): plots are well-annotated with appropriate title, axes and legend, codes are well-organised and annotated with comments / docstrings in functions wherever necessary.
    - Conciseness (2m): 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 (6m): responses are graded as below expectation/satisfactory/above expectations for each part.
- All submissions should be placed in a single zip file in the following format:
  - **lastname_firstname_A1.zip**
  - The zip file must contain the report in a form of a notebook (.ipynb).
    - **In the report, label each section neatly in Markdown**.
    - You can submit multiple notebooks. You are advised to submit 1 notebook per question. E.g. for Part A, you can name your notebooks as follows: PartA_Q1.ipynb, PartA_Q2.ipynb, PartA_Q3.ipynb, PartA_Q4.ipynb ; same goes for Part B.
  - The zip file can also include other source codes, if necessary.
  - Do not include any data or model checkpoints.
- 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 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 perform polarity detection from voice recordings, based on data in the National Speech Corpus, which is obtained from (https://www.imda.gov.sg/programme-listing/digital-services-lab/national-speech-corpus).

The National Speech Corpus is an initiative by the Info-Communications and Media Development Authority, and it is the first large scale Singapore English corpus. Within the dataset, there are 6 parts. In the fifth segment, speakers are made to communicate in several different styles, including Positive Emotions and Negative Emotions. The original recordings are approximately 20 minutes long. Using the librosa library, the recordings are split into shorter segments and preprocessed to features such as chromagrams, Mel spectrograms, MFCCs and various other features. The preprocessed csv file is provided in this assignment.

We will be using the CSV file named full.csv, which is both provided to you. The features from the dataset are engineered. It contains information from the National Speech Corpus accessed on 1 September 2022 from Info-Communications and Media Development Authority, which is made available under the terms of the Singapore Open Data Licence version 1.0 (https://data.gov.sg/open-data-licence).

The aim is to determine the speech polarity of the engineered feature dataset. The csv file is called full.csv with a row of 77 features that you can use, together with the filename. The “filename” column has the labels associated to them.

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, three hidden layers of 128 neurons each with ReLU activation function, and an output layer with sigmoid activation function. Use a mini-batch gradient descent with 'adam' optimizer with learning rate of 0.001, and batch size = 256. Apply dropout of probability 0.2 to each of the hidden layers.

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. (Note that we make the simplifying assumption here that each data sample is independent, hence a random split is performed.)

Parts	Marks
a) Use the training dataset to train the model for **100 epochs**. Implement early stopping with patience of 3.	6
b) Plot train and test accuracies and losses on training and test data against training epochs and comment on the line plots. Explain the use of early stopping in this question.	4

Total: 10

Question 2

In this question, we will determine 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 5-fold cross-validation on training partition to perform hyperparameter selection.**

You will have to reconsider the scaling of the dataset during the 5-fold cross validation.

Parts	Marks
a) Plot mean cross-validation accuracies on the final epoch for different batch sizes as a scatter plot. Limit search space to batch sizes {128, 256, 512, 1024}. This might take a while to run.	5
b) Create a table of time taken to train the network on the last epoch against different batch sizes. (Hint: Introduce a callback)	3
c) Select the optimal batch size and state a reason for your selection.	2
d) What happens when batch size increases, and why does it happen?	2
e) Plot the train and test accuracies against epochs for the optimal batch size in a line plot.	2

Note: **use this optimal batch size for the rest of the experiments**. Total: 14

Question 3

Find the optimal number of hidden neurons for **first hidden layer** of the **5-layer network (input layer, 3 hidden layers, output layer)** designed in **Question 1 and 2.**

Parts	**Marks**
a) Plot the mean cross-validation accuracies on the final epoch for different numbers of hidden-layer neurons using a scatter plot. Limit the search space of the number of neurons to {64, 128, 256}. Continue using 5-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 using a line plot.	2
d) How does dropouts work, and what is the purpose of dropouts?	2
e) Besides early stopping and dropout, what is another approach that you could take to address overfitting in the model, and how does it work? Implement the approach.	3
Note: **use this optimal number of neurons for the rest of the experiments**.	Total: 12

Question 4

In this section, we will understand the utility of such a neural network in real world scenarios.

Parts	**Marks**
a) Record yourself with a wav file using (https://voice-recorder-online.com/) for 5 seconds, either in a positive or a negative manner. Preprocess the data using the provided preprocessing script (data_preprocess.ipynb) and prepare the dataset.	2
b) Do a model prediction on your sample test dataset and obtain the predicted label using a threshold of 0.5. The model used is the optimized pretrained model using the selected optimal batch size and optimal number of neurons.	3
c) Find the most important features on the model prediction for your test sample using *SHAP.* Plot the local feature importance with a force plot and explain your observations. (Refer to the documentation and these three useful references: https://christophm.github.io/interpretable-ml-book/shap.html#examples-5, https://towardsdatascience.com/deep-learning-model-interpretation-using-shap-a21786e91d16, https://medium.com/mlearning-ai/shap-force-plots-for-classification-d30be430e195)	4
	Total: 9

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 speech polarity. 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)

On 16 December 2021, the Singapore government introduced a flurry of measures to cool down the property market. This includes measures targeted at the public property market, i.e. Housing Development Board (HDB) flats: HDB loan-to-value (LTV) ratio were lowered from 90% to 85%, which makes it more difficult for buyers to finance large purchases. Consequentially, demand should fall and thus housing prices should fall too. Since then, HDB resale prices have still been on a sustained rise, but the number of transactions has reduced.

A team of data scientists have previously developed a model to predict HDB prices and found a subset of factors that best explains them. In this assignment, we will investigate whether the change in LTV ratio has made any impact on model performance and their analysis.

This assignment uses publicly available data on HDB flat prices in Singapore, obtained from data.gov.sg on 5th August 2021 and 19th August 2022. The original datasets are 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. Use the dataset provided to you.
- 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.
- **Sample code is given in file 'start_1b.ipynb'** to help you get started with this problem.

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 have a mix of numeric and categorical features – this dataset is one such example. To build models on such data, categorical features have to be **encoded**. Also, before applying neural networks, it is a good practice to try simpler machine learning algorithms first.

For all models in Part B of the assignment, the following features should be used:
- **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

One-hot encoding should be applied on categorical features.
Standardisation should be performed on numeric features.

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 and 2022 used as test data). 2

Why is this done instead of using random train/test splits?

b) A team of data scientists has implemented a linear regression model via Scikit-learn. They obtained a test **R^2 value of 0.627** and happily shared with you that their model only took a few seconds to train. They suggest you to try out an equivalent deep learning model to see if you get a similar result. Recall that a linear regression model is equivalent to a neural network with only 1 Dense layer (i.e. no hidden layer) with linear activation and 1 output node. 5

However, modelling such a mix of feature types with neural networks requires some changes to the input layer. Implement this neural network by following this tutorial from the Keras documentation which guides you through the process of using the Functional API to do so. After encoding / standardisation, the features should be **concatenated**. Your architecture should resemble the figure shown in Appendix A.

c) The team suggests you to train the model for 50 epochs using mini-batch gradient descent with batch size = 256, **Adam** optimiser (with a default learning rate of $\alpha = 0.001$) and mean square error as cost function. However, you find that your results are far off from their model. Change the optimiser to **SGD** (with default learning rate of $\alpha = 0.01$) and observe how the problem gets fixed. **Report the test R^2 value** and **explain why** the change to SGD fixes the problem faced when using Adam optimiser. (Hint: Look carefully at how Adam is implemented and see how SGD is different.) 3

d) Add 1 hidden layer (10 units) to the architecture in Q1c and train it with the same configuration as in Q1c (i.e. with Adam) except that the **learning rate is increased to 0.08**. Report the **test R^2 value**. 2

e) Compare the performance of the linear regression model to the Dense layer (Q1c) and the NN architecture (Q1d) and suggest reasons for the observations you made. 3

Total: 15

Question 2

Neural networks offer much more than fundamental machine learning algorithms. In this part of the assignment, we will investigate one of its advantages: the use of trainable embeddings. Also, we will learn how to set up a quick and convenient way of tuning your neural network models.

Instead of using one-hot encoding, an alternative approach is to use embeddings to encode categorical variables. Such an approach utilises the ability of neural networks to learn richer representations[1] of the data – an edge it has over traditional ML models.

Parts **Marks**

a) Further split the data from year 2020 and before (i.e. those not in test set) by using data from year 2020 as validation set and the rest as the training set. 1

b) For **each categorical** variable, replace the one-hot encoding with the layer tf.keras.layers.Embedding(). Set output_dim = floor(num_categories//divisor). 'num_categories' refers to the number of categories in the categorical variable. 'divisor' is a parameter which we will tune later (Hint: You will still need the lookup classes from Q1b. Read the documentation to find out what to change.) 3

The Embedding layer produces a 2D output (3D, including batch), which cannot be concatenated with the other features. Add a **Flatten** layer to resolve this.

c) Via a callback, introduce early stopping (based on **val_loss**, with patience of 10 epochs) to the model. 5

Using this as a reference, use KerasTuner (with the **RandomSearch** algorithm) to tune the model on the validation set, according to the following ranges:

- Number of neurons: min=4, max=32, step=4
- Learning rate: min=1e-4, max=2e-1, sampling='log'
- Divisor: min=1, max=2, step=1

Run 10 iterations of parameter search (i.e. max_trials=10), each for 50 epochs and report the best set of hyperparameters (based on validation accuracy).

d) Using the best model configuration, train a model on the non-test split (i.e. year 2020 and before) for 50 epochs. Generate a plot to show how the train and test **root** mean square errors (RMSE) changes across epochs.
(Tip: You can skip the first few epochs if the plot gets dominated by them) 2

e) Using the model from the best epoch, report the test R^2 value and show the top 30 test samples with the largest errors. List down any trends you find in these samples and suggest ways to reduce these errors.
(Tip: Add the prediction error as a column in the DataFrame and sort by it.) 4

Total: 15

[1] Instead of just cramming all the information about a category into a single number, embeddings 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.

Question 3

Model degradation is a common issue faced when deploying neural network models in the real world. In typical coursework settings, you learn the ropes by experimenting on toy datasets, which only offers a static snapshot of the situation. Real life problems, such as the analysis of factors influencing HDB prices, have new data points coming in daily that might exhibit a different pattern from older data points due to factors such as changes in government policy or market sentiments. In such situations, models trained on older data points that differ greatly from the new data could perform poorly. In the last part of this assignment, we will investigate whether this has happened.

There are 2 datasets to work with: 'HDB_price_prediction.csv' and 'HDB_price_prediction_old.csv'. The latter is a subset of the former: both start from the same date but the latter ends on August 2021 while the former has data until August 2022. Both have the same set of training data (2020 and before) but the test data for the latter (i.e. 'old test set') is up till August 2021, while the test set from the former has complete data from 2021, along with data till August 2022 ('new test set').

Parts	**Marks**
a) Apply your model from Q2d on the 'old test set'. On the 'new test set', split it into 2021 and 2022. For all 3 test sets, report the test R^2 value you obtained.	4
b) The team that produced the linear regression model shared with you their results (test R^2 values): 0.76 on the old test set, 0.715 when only using 2021 data as test set and 0.464 when only using 2022 data as test set. In light of this (along with their result in Q1b and your results from Q3a), compare the extent to which model degradation has impacted your model to that of the team's linear regression model and explain why this has occurred.	2
c) Model degradation could be caused by various data distribution shifts[2]: covariate shift (features), label shift and/or concept drift (altered relationship between features and labels). Recall that machine learning models generally need the test data distribution to be similar to the training data distribution. With appropriate plots, visualise the distributions of all the features and labels used by the model. Which variable(s) showed the **largest covariate/label shift** that might have led to the drop in model performance as seen in Q3b? With these insights, suggest a way to address the problem of model degradation. (Note: Only include plots relevant to your answer. **Do not include all plots**.)	4
d) The team passed you a script ('**RFE.py**') that recursively removes features from a neural network, so as to find the best feature subset. Run this piece of code with your model from Q2d and report the best feature subset obtained.	2
e) RFE on the 'old test set' eliminated features *degree_centrality* and *month*. It also showed that *dist_to_dhoby* and *dist_to_nearest_stn* are crucial (removing them leads to higher test loss)**.** Compare these features to those in Q3d and discuss whether concept drift has occurred.	3

Total: 15

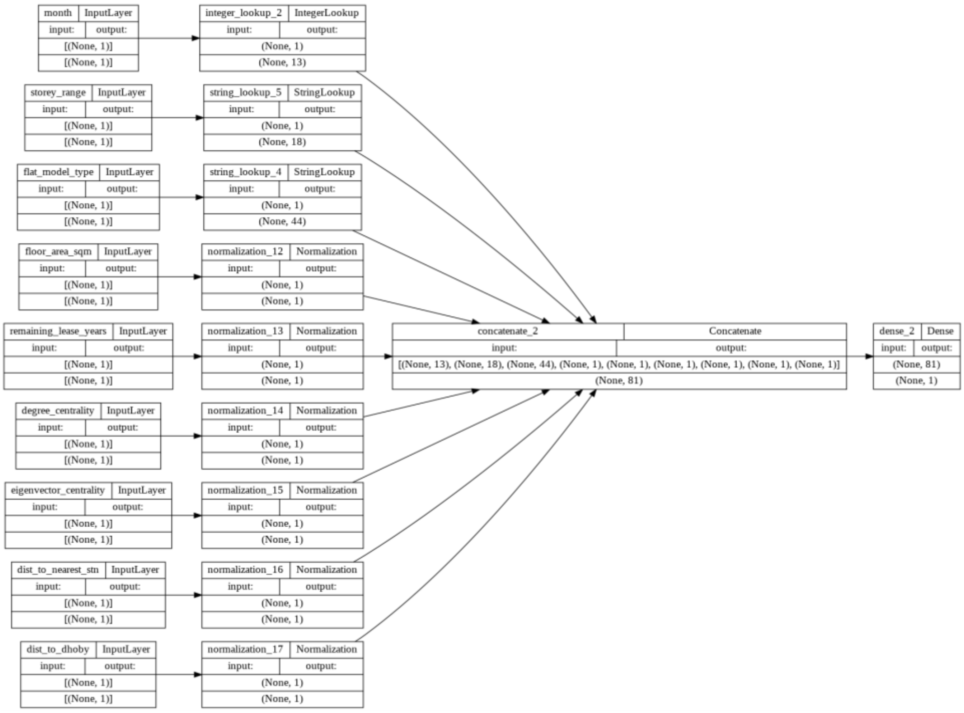
[2] There are various conflicting terminologies in the literature. Let's stick to this reference for this assignment.

Possible discussion pointers for conclusion

Besides the discussion pointers mentioned in Part A,

- In Q1, we compared a linear regression model to an equivalent neural network architecture and also saw how adding a hidden layer changes model performance. In Q2, we saw how adding an Embedding layer introduces more learnable parameters to the neural network. What other benefits do neural networks have over other machine learning approaches? In cases where neural networks perform better, is it possible to modify 'traditional' machine learning algorithms to close up the gap?
- In Q2, we tried out another approach of model tuning. KerasTuner offers many other algorithms – how do Bayesian optimisation or HyperBand work? Are they necessarily better than random search? Also, is random search better than grid search?
- In Q3, we witnessed what happened to machine learning models if they are not updated with the latest datasets and looked at whether covariate shift, label shift or concept drift has occurred. Which of these have led to model degradation? Was the change in LTV ratio the cause of it (if so, how did it affect the model performance)?
- **You do not have to answer all the above discussion pointers.** You can choose to deep dive into one of it and write a paragraph or two to summarise your thoughts + reflect on what you have learnt from this part of the assignment.
- **Feel free to include your own pointers**, as long as they are within the scope of Part B. For instance, we see that the RMSE is still rather high – what else can we do to reduce it? There are many possible extensions to the current model especially because we are limited to using vanilla neural networks for this assignment.

Appendix A



month | InputLayer | input: [(None, 1)] | output: [(None, 1)]
integer_lookup_2 | IntegerLookup | input: (None, 1) | output: (None, 13)
storey_range | InputLayer | input: [(None, 1)] | output: [(None, 1)]
string_lookup_5 | StringLookup | input: (None, 1) | output: (None, 18)
flat_model_type | InputLayer | input: [(None, 1)] | output: [(None, 1)]
string_lookup_4 | StringLookup | input: (None, 1) | output: (None, 44)
floor_area_sqm | InputLayer | input: [(None, 1)] | output: [(None, 1)]
normalization_12 | Normalization | input: (None, 1) | output: (None, 1)
remaining_lease_years | InputLayer | input: [(None, 1)] | output: [(None, 1)]
normalization_13 | Normalization | input: (None, 1) | output: (None, 1)
degree_centrality | InputLayer | input: [(None, 1)] | output: [(None, 1)]
normalization_14 | Normalization | input: (None, 1) | output: (None, 1)
eigenvector_centrality | InputLayer | input: [(None, 1)] | output: [(None, 1)]
normalization_15 | Normalization | input: (None, 1) | output: (None, 1)
dist_to_nearest_stn | InputLayer | input: [(None, 1)] | output: [(None, 1)]
normalization_16 | Normalization | input: (None, 1) | output: (None, 1)
dist_to_dhoby | InputLayer | input: [(None, 1)] | output: [(None, 1)]
normalization_17 | Normalization | input: (None, 1) | output: (None, 1)
concatenate_2 | Concatenate | input: [(None, 13), (None, 18), (None, 44), (None, 1), (None, 1), (None, 1), (None, 1), (None, 1), (None, 1)] | output: (None, 81)
dense_2 | Dense | input: (None, 81) | output: (None, 1)

References

[1] Koh JX, Mislan A, Khoo K, Ang B, Ang W, Ng C, Tan YY. Building the singapore english national speech corpus. Malay. 2019;20(25.0):19-3.

[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.