ITI106 Foundation of Deep Learning

Practical Assignments

Pamela Sin - M374907

1. Introduction and background - GTZAN dataset

This assignment aims at building neural networks to classify the GTZAN dataset. The GTZAN dataset is a widely used dataset collected from the year 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 obtained from Kaggle (https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification).

We will be using the CSV file named features_30_sec.csv. The 30-seconds long audio files consist of 57 features engineered by the dataset owner. 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 10 genres are blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock.

We first read the data from the file features_30_sec.csv. There are 1000 data samples in total, which is 100 data samples for each genre. Each is a row of 60 columns, which consist of the filename (where the original audio file can be found), length of the audio file, label, and the 57 features which will be used.

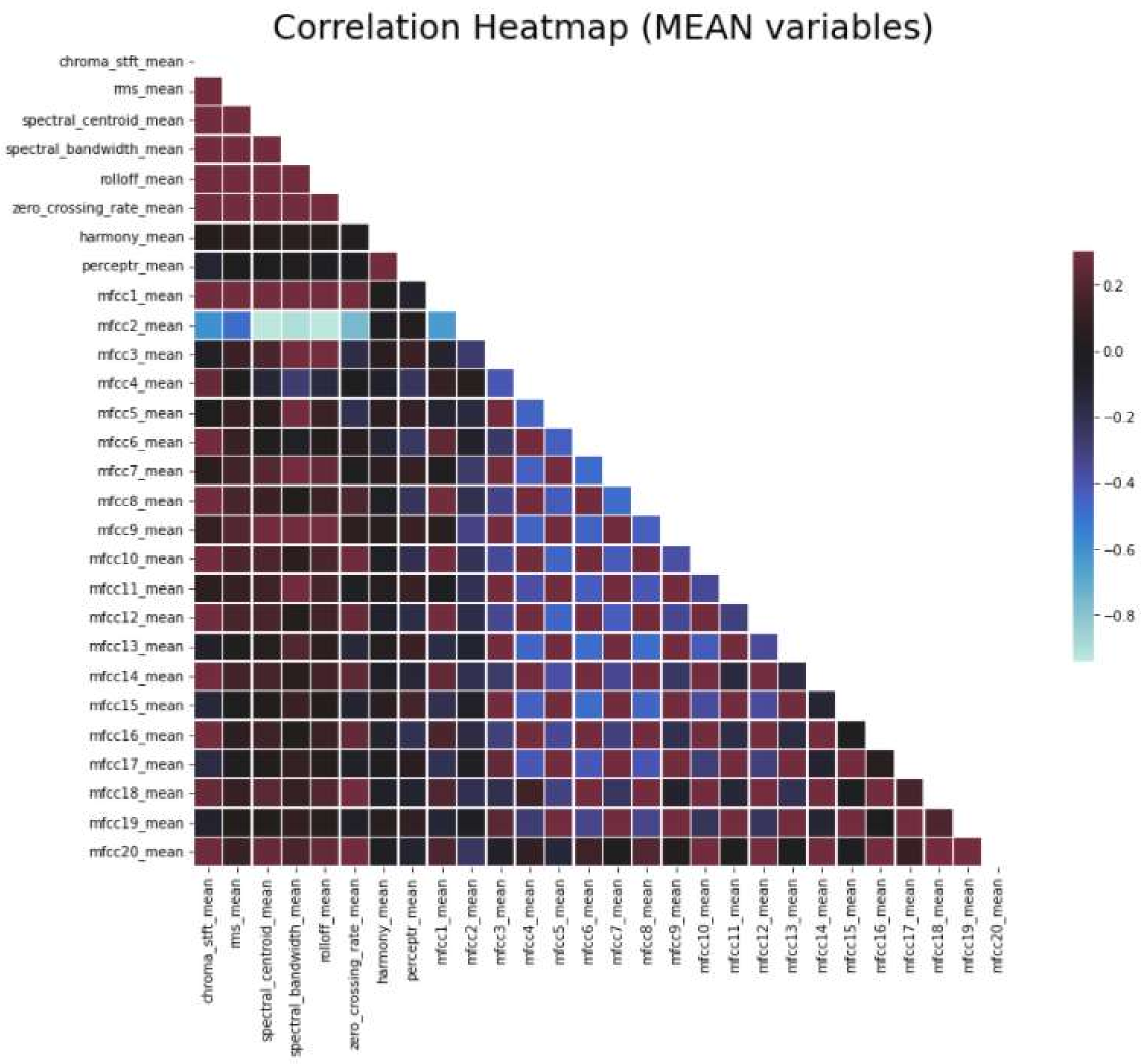
The explanation of some of the features is provided in the table below:

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 zerocrossings 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-areharmonics-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

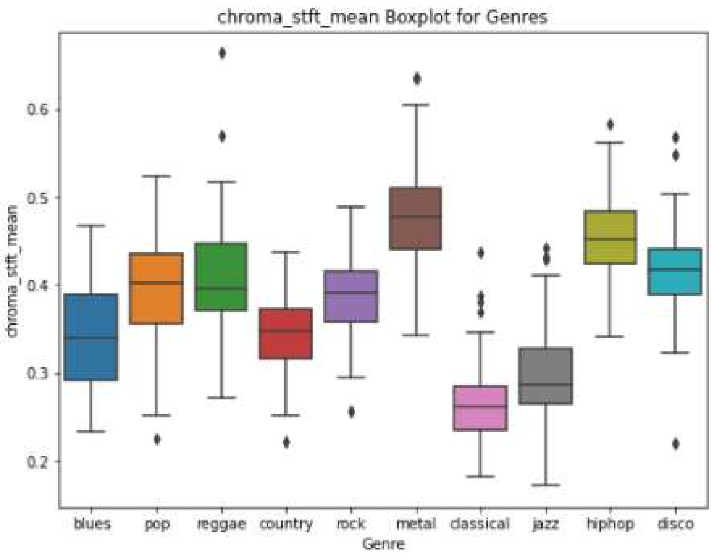
2. Exploratory data analysis and data pre-processing

A correlation heatmap of the features which are mean variables is plotted, showing in a glance which variables are correlated, to what degree and in which direction. For example, we find that mfcc2_mean and spectral_centroid_mean are highly correlated to each other, reaching a correlation of close to 1.00. The feature mfcc2_mean appears to be correlated to quite a few other features as indicated by the lighter colored cells in the heatmap, indicating a potential for dimensionality reduction to be useful.



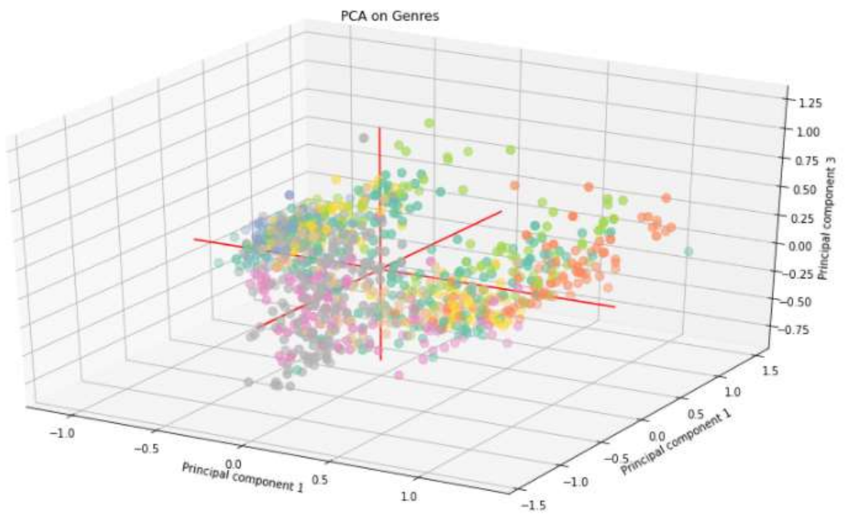
A boxplot for each of the 57 features is being plotted, providing visual representation for the distribution of each genre with respect to a given feature.

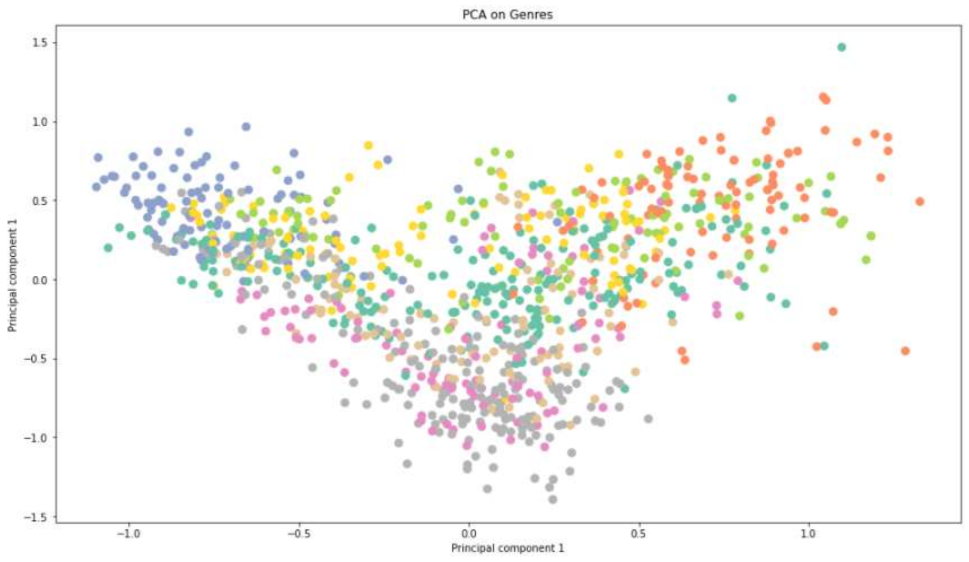
For example, in the boxplot for chroma_stft_mean for each genre, the metal genre has a relatively higher chroma_stft_mean and the classical genre has a relatively lower chroma_stft_mean as compared to the other genres In the neural network, this feature may be especially indicative in distinguishing metal and classical genres.



Principal component analysis (PCA) specifying 3 components was done using Singular Value Decomposition of the data to project it to a lower dimensional space.

The explained variance ratio, which is the percentage of variance explained by each of the selected components, is 0.24644968, 0.22028192 and 0.09803759 respectively. PCA on three-dimensions and two-dimensions are plotted to obtain a visual representation on clusters of datapoints for each genre.





Label encoder was used to encode the target labels from 0 to 9 (for the 10 classes).

The dataset was divided into a 70:30 ratio for training and testing. A standard scaler fit transform was done on the input features of the train set, and a standard scaler transform was done on the input features test set.

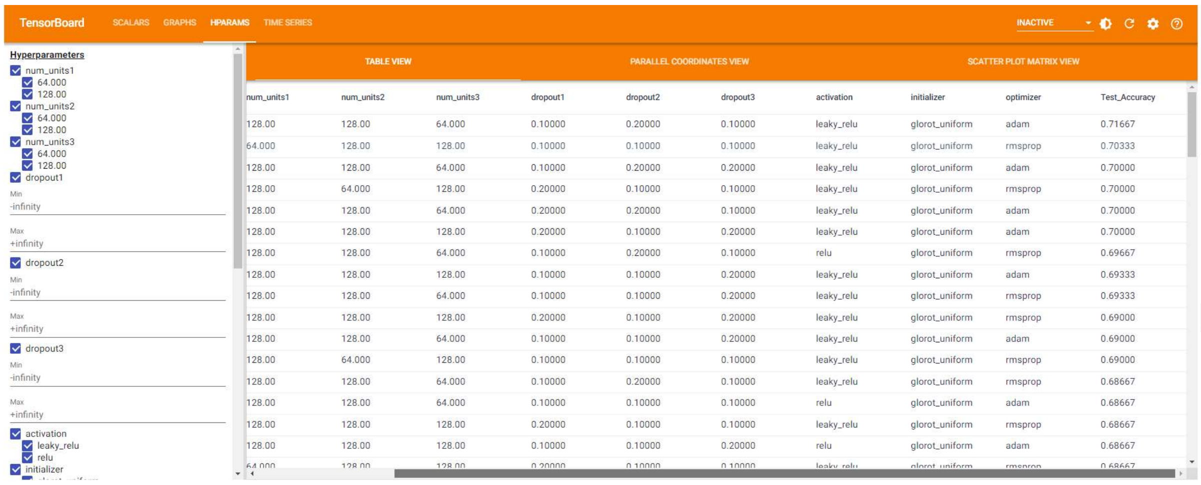
3. Feedforward deep neural network hyperparameter-tuning

A feedforward deep neural network which consists of an input layer, three hidden layers and an output softmax layer was constructed. 768 (2^8 * 3) experiments were performed, where the validation accuracy of all combinations of the below hyperparameters were recorded. Each experiment was run for 10 epochs.

- Number of neurons in first hidden layer: 64, 128
- Dropout probability in first hidden layer: 0.1, 0.2
- Number of neurons in second hidden layer: 64, 128
- Dropout probability in second hidden layer: 0.1, 0.2
- Number of neurons in third hidden layer: 64, 128
- Dropout probability in third hidden layer: 0.1, 0.2
- Activation function for hidden layer neurons: ReLu, LeakyReLu
- Weight initialiser: glorot_uniform, random_uniform
- Optimiser: Adam, SGD , Rmsprop

4. Discussion of results and models development

TensorBoard was used to provide visualization of the experiment results.



Hyperparameters

- ☑ num_units1
  - ☑ 64.000
  - ☑ 128.00
- ☑ num_units2
  - ☑ 64.000
  - ☑ 128.00
- ☑ num_units3
  - ☑ 64.000
  - ☑ 128.00
- ☑ dropout1
  - Min: -infinity
  - Max: +infinity
- ☑ dropout2
  - Min: -infinity
  - Max: +infinity
- ☑ dropout3
  - Min: -infinity
  - Max: +infinity
- ☑ activation
  - ☑ leaky_relu
  - ☑ relu
- ☑ initializer

num_units1	num_units2	num_units3	dropout1	dropout2	dropout3	activation	initializer	optimizer	Test_Accuracy
128.00	128.00	64.000	0.10000	0.20000	0.10000	leaky_relu	glorot_uniform	adam	0.71667
64.000	128.00	128.00	0.10000	0.10000	0.10000	leaky_relu	glorot_uniform	rmsprop	0.70333
128.00	128.00	64.000	0.10000	0.20000	0.20000	leaky_relu	glorot_uniform	adam	0.70000
128.00	64.000	128.00	0.20000	0.10000	0.10000	leaky_relu	glorot_uniform	rmsprop	0.70000
128.00	128.00	64.000	0.20000	0.20000	0.10000	leaky_relu	glorot_uniform	adam	0.70000
128.00	128.00	128.00	0.20000	0.10000	0.20000	leaky_relu	glorot_uniform	adam	0.70000
128.00	128.00	64.000	0.10000	0.20000	0.10000	relu	glorot_uniform	rmsprop	0.69667
128.00	128.00	128.00	0.10000	0.10000	0.20000	leaky_relu	glorot_uniform	adam	0.69333
128.00	128.00	64.000	0.10000	0.10000	0.20000	leaky_relu	glorot_uniform	rmsprop	0.69333
128.00	128.00	128.00	0.20000	0.10000	0.20000	leaky_relu	glorot_uniform	rmsprop	0.69000
128.00	128.00	64.000	0.10000	0.10000	0.20000	leaky_relu	glorot_uniform	adam	0.69000
128.00	64.000	128.00	0.10000	0.10000	0.10000	leaky_relu	glorot_uniform	rmsprop	0.69000
128.00	128.00	64.000	0.10000	0.20000	0.10000	leaky_relu	glorot_uniform	rmsprop	0.68667
128.00	128.00	64.000	0.10000	0.10000	0.10000	relu	glorot_uniform	adam	0.68667
128.00	128.00	128.00	0.20000	0.10000	0.10000	leaky_relu	glorot_uniform	rmsprop	0.68667
128.00	128.00	128.00	0.10000	0.10000	0.20000	relu	glorot_uniform	adam	0.68667
64.000	128.00	128.00	0.20000	0.10000	0.10000	leaky_relu	glorot_uniform	rmsprop	0.68667

The above results are sorted according to decreasing order of accuracy.

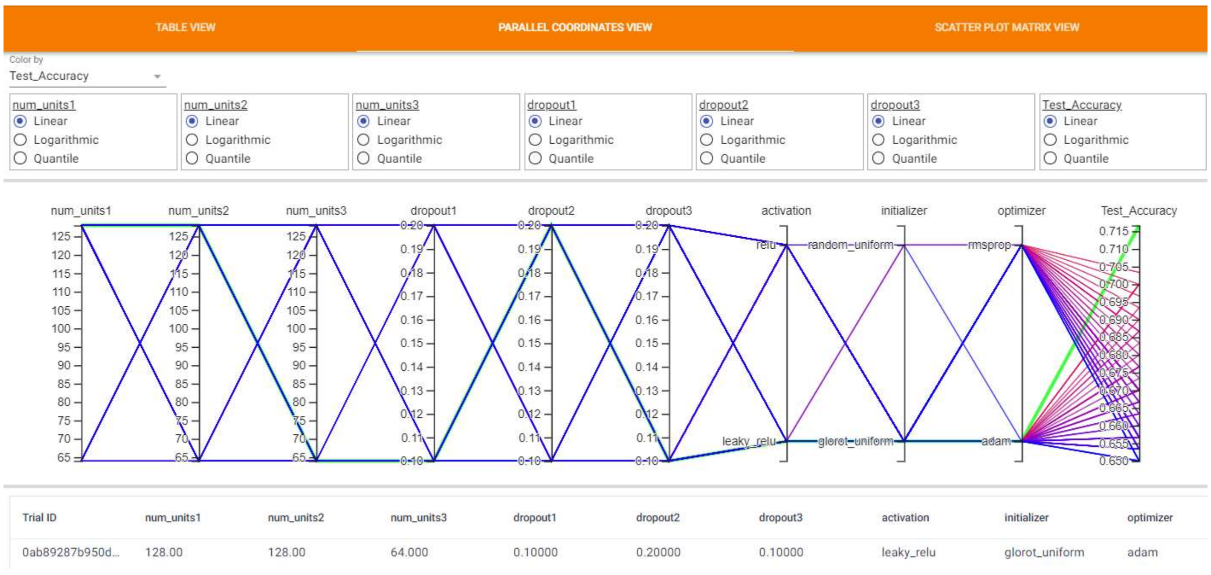
The top 99 results have "glorut_uniform" as the weight initialiser. The Glorot uniform initializer is also called Xavier uniform initializer. The aim of weight initialization is to prevent layer activation outputs from exploding or vanishing during a forward pass through a deep neural network. If either occurs, loss gradients will either be too large or too small to flow backwards beneficially, and the network will take longer to converge. In literature, there is evidence to show that Xavier Initialization keeps the variance the same across every layer, and hence it has an edge over random uniform initialiser which simply generates tensors with a uniform distribution.

At least the top 100 results have the optimiser as either Adam or RMSProp (Root Mean Squared Propagation). RMSProp is essentially gradient descent being updated to use an automatically adaptive step size for each input variable using a decaying moving average of partial derivatives. Adam is an algorithm for gradient-based optimization of stochastic objective functions, combining the advantages of two RMSProp and Adaptive Gradient Algorithm (AdaGrad) and computes individual adaptive learning rates for different parameters. The Adam and RMSProp optimization algorithms have specific implementations that are an edge to SGD, explaining our empirical results.

For the ReLU activation function, the gradient is 0 for all the values of inputs that are less than zero, which may deactivate neurons in a certain region and may cause dying ReLU problem. With Leaky ReLU, there is a small gradient allowed when the unit is not active by defining an extremely small linear component of the input to the activation function instead of 0. This could be a possible explanation for a larger percentage of the top 100 experiments (with the best validation accuracy) having the Leaky ReLU activation function for the hidden neurons instead of ReLU.

In observing the trend of results, for the other hyperparameters, there does not appear to be a clear consistency in the choice for that hyperparameter across the experiments which would definitely lead to a better accuracy.

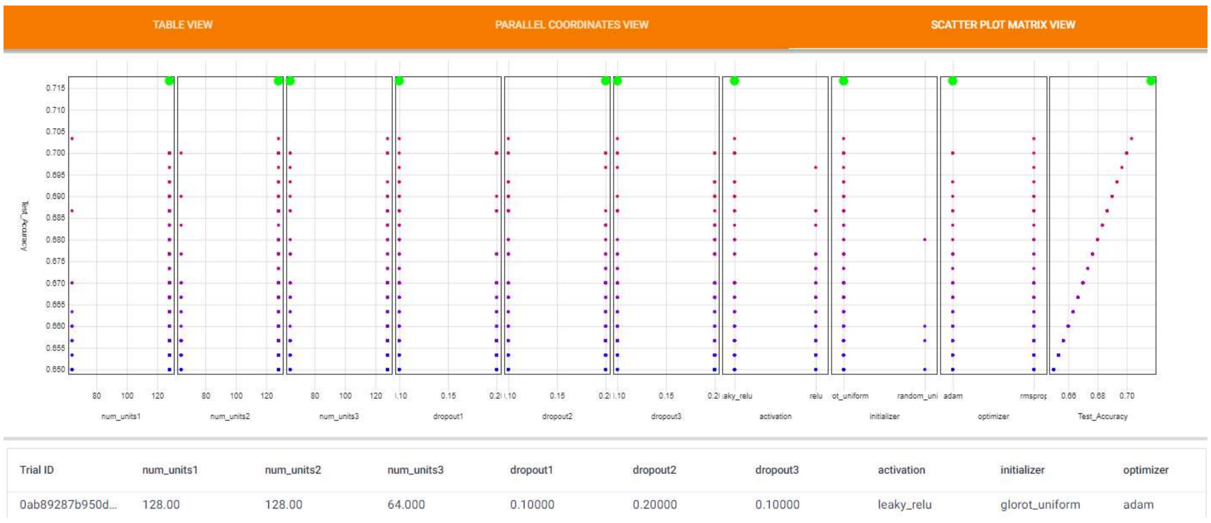
In explaining the effect of dropouts, adding dropout layers probabilistically drops out nodes in the network during training, where some number of layer outputs are randomly dropped out and temporarily removed from the network along with its incoming and outgoing connections. This is a regularization method to reduce overfitting and improve generalization errors. Dropout has an effect of making the training process more noisy and forces nodes within a layer to probabilistically take on more or less responsibility for the inputs. Thus with dropouts, the model is less prone to overfitting and increased generalization errors.



Color by
Test_Accuracy

num_units1	num_units2	num_units3	dropout1	dropout2	dropout3	Test_Accuracy
[✓ Linear]	[✓ Linear]	[✓ Linear]	[✓ Linear]	[✓ Linear]	[✓ Linear]	[✓ Linear]
Logarithmic	Logarithmic	Logarithmic	Logarithmic	Logarithmic	Logarithmic	Logarithmic
Quantile	Quantile	Quantile	Quantile	Quantile	Quantile	Quantile

Trial ID	num_units1	num_units2	num_units3	dropout1	dropout2	dropout3	activation	initializer	optimizer
0ab89287b950d...	128.00	128.00	64.000	0.10000	0.20000	0.10000	leaky_relu	glorot_uniform	adam



Trial ID	num_units1	num_units2	num_units3	dropout1	dropout2	dropout3	activation	initializer	optimizer
0ab89287b950d...	128.00	128.00	64.000	0.10000	0.20000	0.10000	leaky_relu	glorot_uniform	adam

From empirical results specific to the GTZAN dataset, the model that incorporates 128 neurons in the first hidden layer with a dropout probability of 0.1, 128 neurons in the second hidden layer with a dropout probability of 0.2, 64 neurons in the third hidden layer with a dropout probability of 0.1, glorut_uniform as the weight initialiser and Adam optimiser produced the best validation accuracy of 0.71667.

Based on the best set of hyperparameters obtained, the model was run for a fresh 50 epochs based on the same 70% train and 30% test split.

For the graph of training accuracy against training epochs, the accuracy starts at approximately 0.4386 after the first epoch, and generally increases at a decreasing rate, reaching the highest training accuracy of 0.9200 after 50 epochs. For the graph of testing accuracy against training epochs, the accuracy starts at approximately 0.4633 after the first epoch, and generally increases at a decreasing rate, plateauing around 30 epochs, reaching an accuracy of about 0.7367 after 38 epochs. Generally, at each training epoch, the training accuracy is higher than the testing accuracy. The approximate number of epochs where the test error begins to converge is also around 30, where the loss starts to plateau at around a loss of 1.75.

5. Comparison between deep learning and classical machine algorithms

One difference between deep learning and classical machine learning algorithms is that deep learning requires minimal feature engineering while machine learning requires more human intervention such as complex feature engineering to obtain results. In machine learning, usually a deep dive exploratory data analysis is first performed, then dimensionality reduction for easier processing, then the best features are hand-selected to pass over to the machine learning algorithm. However, using deep neural networks, the data can be passed directly into the network and this can eliminate the feature engineering stage. For example, with respect to the GTZAN dataset, features such as harmonics, tempo and roll-off are engineered features that industry professionals are aware of and use. Using classical machine learning algorithms, these hand-engineered features can be fed into the model. In contrast, deep learning networks have their own capacity to learn their own features from raw inputs, and earlier layers in the network can learn basic features while later layers can learn more complex features. Using the current approach of using feed-forward neural networks to model such engineered features have the limitation of the network only "seeing" the features humans have hand engineered, whereas feeding the network the raw audio inputs (or a representation of it) the network might be able to extract features on its own that might be more useful even if it is not easily human-interpretable. Thus, an alternative to approaching the GTZAN classification problem using deep neural networks is to firstly convert raw audio into an image-like representation, which encodes the frequency and amplitude information overtime of the raw audio signal (create a spectrogram, which is a visual way of representing the signal strength of a signal over time at various frequencies present in the waveform). For the model, a simple convolutional neural network is used to recognise the features in the spectrograms.

Secondly, deep learning techniques can be adapted to different applications and domains more easily than classical machine learning algorithms. For example, in transfer learning, pre-trained convolutional neural networks can be used as feature extractors for a different dataset in objection detection and segmentation problems in computer vision. The use of pre-trained networks as front-end feature extractors helps achieve a higher performance in a shorter period of time. In contrast, with classical machine learning, domain-specific and application-specific techniques are required to build high-performing models, with specialised study within a specific area.

Thirdly, deep learning requires large datasets to achieve high performance while classical machine learning can outperform deep networks for smaller datasets. For example, pre-trained networks like Inception and VGG were trained on millions of images in order to obtain the set of weights for the network. Deep learning networks scale effectively with data. This in turn is tied to the fact that deep neural networks is computationally expensive with the use of high-end GPUs, while classical machine learning algorithms can be trained with CPUs. More iterations of different machine learning algorithms can be done given the shorter computational time as well.

Fourthly, deep learning networks are less human-interpretable being a "black box", while classical machine learning is more interpretable and easier to explain. Machine learning algorithms involve direct feature engineering, and hence with more understanding of the data and the algorithm, the model design is more straightforward. For example, decision trees provide a graphical and intuitive way to understand what the algorithm does.

6. Conclusion

In conclusion, neural networks are increasingly being used in a wide variety of applications in modern-day usage, from computer vision to natural language processing. Top-performing models now come from this area, especially in this data-driven world where big data is out there for people to harness and understand. Neural networks have significant advantages over classical machine learning algorithms, and this assignment explores its use in a multi-classification problem.

Specific to the GTZAN dataset, we find that overfitting can pose a problem for smaller datasets. As a further extension of the project, the following approaches to address overfitting in the model could be tested by performing a greater number of experiments. Firstly, the neural network can be trained on a larger dataset. In this GTZAN dataset, each original data sample is a 30-second audio file. In place of this, 3-second chunks of each audio file can be taken instead, which would increase the number of datapoints by 10 times. Secondly, putting constraints on the weights of the model such as constraining them to be within a range. Thirdly, early stopping by monitoring model performance on the validation set and stopping when performance degrades. Fourthly, data augmentation can be considered, adding noise to the data samples so that the final model is more robust.

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.

[5] https://www.deeplearning.ai/ai-notes/initialization/

[6] https://towardsdatascience.com/weight-initialization-in-neural-networks-a-journey-from-the-basics-to-kaiming-954fb9b47c79

[7] https://machinelearningmastery.com/gradient-descent-with-rmsprop-from-scratch/#:~:text=Root%20Mean%20Squared%20Propagation%2C%20or,step%20size%20for%20each%20parameter.

[8] https://medium.com/syncedreview/iclr-2019-fast-as-adam-good-as-sgd-new-optimizer-has-both-78e37e8f9a34

[9] https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

[10] https://www.analyticsvidhya.com/blog/2021/06/introduction-to-audio-classification/

[11] https://towardsdatascience.com/deep-learning-vs-classical-machine-learning-9a42c6d48aa