EXTENSIVE HUFFMAN-TREE-BASED NEURAL NETWORK FOR THE IMBALANCED DATASET AND ITS APPLICATION IN ACCENT RECOGNITION

By

Jeremy Gordon Merrill

Dr. Yu Liang
Professor of Computer Science
Committee Chair

Dr. Dalei Wu
Associate Professor of Computer Science
Committee Member

Dr. Yingfeng Wang
Associate Professor of Computer Science
Committee Member

EXTENSIVE HUFFMAN-TREE-BASED NEURAL NETWORK FOR THE IMBALANCED DATASET AND ITS APPLICATION IN ACCENT RECOGNITION

By

Jeremy Gordon Merrill

A Thesis Submitted to the Faculty of the University of
Tennessee at Chattanooga in Partial Fulfillment
of the Requirements of the Degree of
Master of Science: Computer Science

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

May 2021

ABSTRACT

To classify the data-set featured with a large number of heavily imbalanced classes, this thesis proposed an Extensive Huffman-Tree Neural Network (EHTNN), which fabricates multiple component neural network-enabled classifiers (e.g., CNN or SVM) using an extensive Huffman tree. Any given node in EHTNN can have an arbitrary number of children. Compared with the Binary Huffman-Tree Neural Network (BHTNN), an EHTNN may have a smaller tree height, involve fewer component neural networks, and demonstrate more flexibility on handling data imbalance. Using a 16-class exponentially imbalanced audio data-set as the benchmark, the proposed EHTNN was strictly assessed based on the comparisons with alternative methods such as BHTNN and single-layer (canonical) CNN. The experimental results demonstrated promises results about EHTNN in terms of Gini index, Entropy value, and the accuracy derived from hierarchical multiclass confusion matrices.

ACKNOWLEDGEMENTS

I would like to begin by thanking my family, friends, employer, and classmates. Your support and motivation through the years to keep me on track and pursue my goals has been unmeasurable and words will never show my appreciation. Additionally, I would like to thank my professors, advisors, committee members as well as Dr. Yu Liang for giving me the support and learning experience to extend my education and for keeping me on track. Furthermore, I’d like to thank the team members of the Networked Intelligence lab for their contributions, constructive criticism and teaching me new concepts and techniques to further my understanding in this field of study. Lastly, I’d like to show my gratitude to the National Science Foundation grants for sponsoring this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

HTNN, Huffman Tree Neural Network

EHTNN, Extensive Huffman Tree Neural Network

AI, Artificial Intelligence

CNN, Convolution Neural Network

2D, 2 Dimensional

SVM, Support Vector Machines

ReLU, Rectified Linear Unit

BHTNN, Binary Huffman Tree Neural Network

API, Application Programming Interface

HTA, Huffman Tree Architecture

LIST OF SYMBOLS

k, number of classes to move to canonical CNN within the EHTNN

α, fractional value indicating how much of the frequency value of the lowest occurring class should be less than the standard deviation of all classes within the canonical CNN

p_j, probability value of a given input belonging to a particular class

H, entropy of the occurrence of a particular class to the other classes in the dataset

WH, weighted entropy

G, impurity (gini) index of a particular class to the other classes in the dataset

WG, weighted impurity (gini) index

W, weighted average (mean)

w_i, number of records in the i-th class

X_i, weight (or percentage) of the i-th class in the dataset

CHAPTER 1

INTRODUCTION

A long-standing issue for audio classification is low accuracy due to an imbalanced data set. This thesis is the result of efforts to discover a better solution to audio classification of voice accent on a skewed data set. A skew data set is a data set in which the training or validation data is skewed towards a particular class or set of classes, sometimes on an exponential level [1]. Skewed data sets can produce too many negative examples vs positive examples, causing the classifier to incorrectly predict a class for a given piece of data. Skewed data sets do not perform as well as a balanced data set. Likewise, a multi-class classification problem makes the prediction accuracy even worse. Skewed data can even alter the ability of the classifier's training by insufficiently providing enough data during training to accurately fit the model. A few concepts related to over-sampling or under-sampling have been utilized to help circumvent this issue [2]. Random re-sampling or random selective sampling also has drawbacks. This technique could also affect the classifier's ability to train and fit the model appropriately as over-sampling less frequent classes doesn't provide enough data variation to build a model that is trained diverse enough to recognize slight variations in data. Moreover, random under-sampling can limit the capacity of the model to fit on a diverse dataset in the same sense of under-sampling. Lastly, sampling does not account for the likelihood of an input data belonging to a skewed class. Sampling indicates the classes are equally weighted to the classifier [2].

This thesis explores a Huffman-tree-based approach to classification by breaking down a large multi-class problem into several smaller problems. The remainder of this thesis is divided into chapters. Chapter 2 will discuss related work to machine learning for audio classification. Chapter 3 will describe the Binary Huffman-Tree Neural Network (BHTNN). Chapter 4 will pronounce the contribution toward the Extensive Huffman Tree Neural Network (EHTNN). The results and comparison of each model is given in Chapter 5, and a conclusion with potential future work is denoted in Chapter 6.

CHAPTER 2

MACHINE LEARNING ENABLED AUDIO CLASSIFICATION

The topic of audio classification using machine learning is not a new concept. Plenty of research within the artificial intelligence (AI) field of computer science has been performed. Many researchers have discovered multiple neural network techniques that can be used in order to train an AI machine to accurately classify snippets of audio based on music genre, what instrument is playing, or what words an individual is speaking. Most of these audio classification techniques use a specific type of neural network called a Convolution Neural Network (CNN). This machine learning method is mostly used to identify features of a two-dimensional image. Audio can be used as the input to a CNN whenever it is converted into a spectrogram using a mathematical signal processing technique called Fourier Transform. This converts the audio signal into a frequency vs time graph that highlights the dominant frequency at any given time within an audio snippet. This process is shown in Figure 2.1.

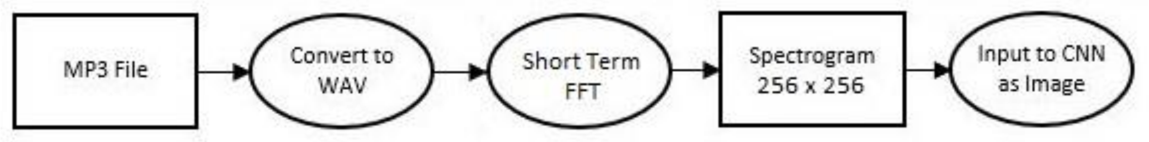


Figure 2.1

Diagram illustrating audio files as CNN input (1s sample of WAV)

This thesis will cover the practical use of using CNNs to correctly classify an audio recording of a human speaking words into a microphone by nationality through identifying frequency features of the person’s accent. While this technique isn’t much different than music genre classification, this thesis will extend on the concept by exploring how to increase the classification accuracy of the network where there are a higher than usual number of categorical classes. Having a high number of classes reduces the ability for a single CNN to accurately classify an input. Additionally, the data set being used is unbalanced on an exponential scale. This thesis will explore a technique for increasing accuracy of the model, reducing the time complexity, and minimizing the entropy and impurity (Gini) index.

Using machine learning to analyze audio is not a new concept. Several different types of neural networks exist that are used to process audio depending on the goal of the model. In practice, many audio classifications are transformed into images during the data pre-processing stage of the neural network [3]. The 2D visual representation of the audio is produced by a Fourier transform which generates a frequency vs time plot. This image is useful for classification and sequential analysis because a Fourier transform highlights dominant frequencies at any given moment throughout the duration of the audio sample [3]. A Fourier transform (often referred to as a spectrogram) is then delivered as input into an image based neural network model.

Data Preparation for Accent Recognition

Many datasets are available for this type of work. For this thesis, the “common-voice” dataset was retrieved from Kaggle. This dataset comprises of hundreds of thousands of mp3 files containing words spoken by many different people. On average, each mp3 file is only a few seconds long. A quick sample of a few of the files reveals the person is either speaking a sentence or multiple random words, some with a brief pause in between. The dataset is accompanied by a meta data file that contains a record for each mp3 file – indicating the corresponding filename, words spoken, and some demographics about the person speaking (gender, age, and accent) [4]. To simplify the training process, a subset of the data was used to speed up the training process and to prevent the learning engine from exceeding system resources. The first twenty-five thousand records in the training dataset that contained a value for the accent field were used.

To further prepare the data, each audio file was sampled for 1 second to create the same spatial resolution for the spectrogram. The audio recordings provided in the data set are not of equal length and the resolution mismatch would cause false positives in the neural network's convolution and max-pooling layers.

Each spectrogram image for the model input was prepared using the ImageDataGenerator class within the Keras API. This class simplifies the import of training and validation data into a tensor data type as well as loading the correct classification labels for each piece of data. Additionally, the spectrograms were resized to a square image (which is ideal for CNNs) and converted to grayscale to eliminate color differences on the dataset.

Support Vector Machine Enabled Audio Classification

Support Vector Machines (SVM) are occasionally used to classify images in a machine learning model where a finite feature can be chosen [3,5]. The SVM can be used to classify images using more than one feature; however, additional features are computationally more expensive quadratically [5]. This thesis uses CNN over SVM as the model for these reasons.

However, SVMs are most applicable to binary classification models where the neural network must classify an image input between one class or another [5]. SVMs can be used to classify images into multiple classes; however, they must be implemented in a chained manner [5].

Convolutional Neural Network Enabled Audio Classification

Convolutional Neural Networks (CNN) are also used for classification by using filters to identify multiple features within the image that are similar to other images within the same class [6]. Each image from the input is trained by convoluting a subset of pixels from the image with a kernel matrix, resulting in a convoluted matrix. Weights are then applied to the parameters of the network during training as the network is identifying important features and enhancing the prediction accuracy of the network [6]. The convolutional layer is followed by a pooling layer, extracting the most important features from the convolution and reduces the feature dimensions [6]. Finally, an activation function is used to determine which neurons of the neural network are important to the classification. Recent studies have concluded the Rectified Linear Unit (ReLU) function is best for CNNs because it does not activate all neurons at once [6]. ReLU has also been shown to have better performance than other functions, such as sigmoid [6].

CHAPTER 3

BINARY HUFFMAN TREE NEURAL NETWORK

As a remedy to the skew data set, a SVM-based Binary Huffman Tree Neural Network was proposed in [7] as a refined method to increase classification and prediction accuracy. However, due to the computational limitations of SVM-based neural networks for data with multiple features, we are proposing a CNN-based Binary Huffman Tree Neural Network.

Huffman Tree Architecture and Numerical Analysis

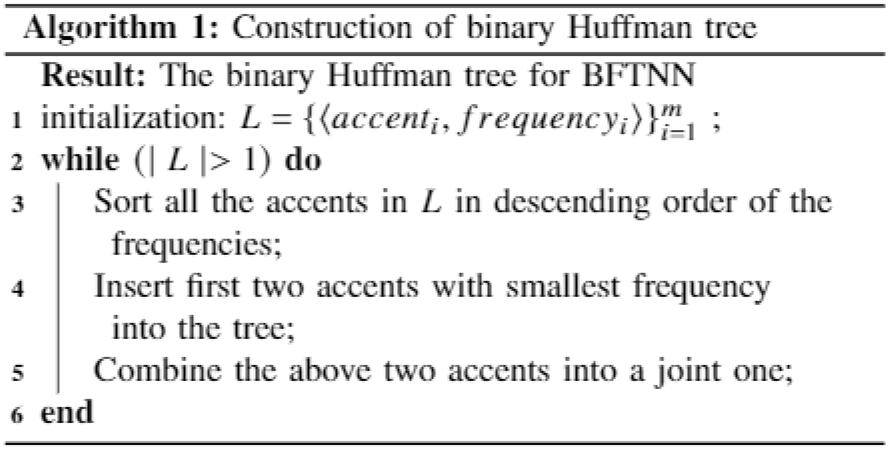
The Huffman tree increases accuracy by predicting if inputs are classified as the highest frequently occurring class first, followed by another classification of the next higher occurring class, and so on [7-9]. As a result, this technique requires a numerical analysis to be performed on the data set. The analysis will determine how to initially configure the Huffman tree. Each class is listed in descending order with the frequency of occurrence within the dataset. Such an analysis is given in Table 3.1.

Table 3.1

Frequency of Common Voice Accents for Sub-Sample of 25,000

Code	**Accent**	**Frequency**
a	US	11974
b	England	5751
c	India	1806
d	Australia	1677
e	Canada	1486
f	Scotland	565
g	Africa	445
h	New Zealand	422
i	Ireland	364
j	Philippines	125
k	Wales	104
l	Bermuda	80
m	Malaysia	75
n	Singapore	50
o	Hong Kong	42
p	South Atlantic	34

Huffman coding defines the algorithm and procedure in order to build a visual binary tree that represents the frequency table. Starting with the classes of the lowest frequency, create two tree nodes. Create a parent node with the sum of the frequencies of the child nodes. As you move up the table, if the next class's frequency exceeds the total of the recently created parent node, a new tree should be created and then combined with the current tree with a new parent node [7-9]. Figure 3.1 shows the resulting binary Huffman Tree derived from the Common Voice Accent data set [4]. The parent tree nodes represent binary neural network classifiers.



Algorithm 1: Construction of binary Huffman tree

```
Result: The binary Huffman tree for BFTNN
initialization: $L = \{\langle accent_i, frequency_i \rangle\}_{i=1}^{m}$ ;
while (| $L$ |> 1) do
    Sort all the accents in $L$ in descending order of the
      frequencies;
    Insert first two accents with smallest frequency
      into the tree;
    Combine the above two accents into a joint one;
end
```

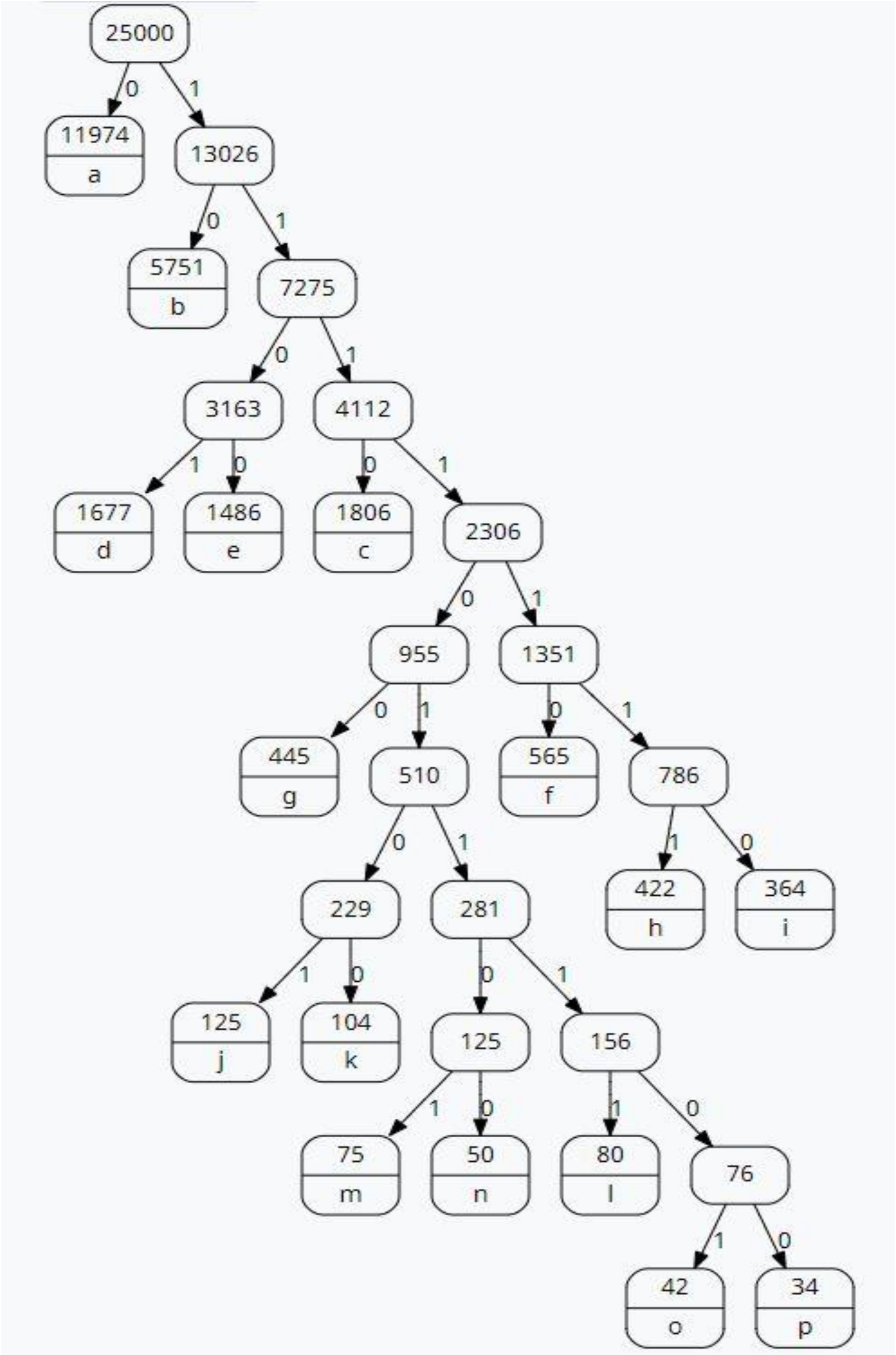


Figure 3.1

Binary Huffman Tree Neural Network (BHTNN) with 15 component classifiers

BHTNN Based on Support Vector Machines

Since SVMs are binary classification networks; they must be chained together when implementing a multi-class output [5]. Such a method can be achieved by implementing a Binary Huffman Tree Neural Network (BHTNN) with multiple individually trained SVM networks. Using a binary tree allows the machine learning engineer to create multiple SVM networks in a one-against-all or one-against-one dataset. Eventually, each class will be represented within the decision tree [5]. Using a BHTNN tree will increase the classification accuracy because the most commonly occurring class will be the first neural network in the model in a one-against-all implementation [7]. Using the Huffman encoding algorithm, a binary tree is developed with all childless nodes representing a class and all parent nodes referring to an individually trained neural network [7].

BHTNN Based on CNN

Each decision point within the BHTNN is a separately trained CNN. Each network is configured for binary classification as a one-against-all, some-against-all, or a one-against-one data-set. At the top of the tree, the first classification is a one-against-all. Mid-way through the tree, there are a few some-against-some networks. It should be noted that these networks are no more than three classes-against-all. In this case, the network is predicting if an input is any of the two or three classes vs everything else. Accuracy suffers slightly in the some-against-all and the some-against-some networks. The one-against-one networks are simpler than the others as they are only predicting between two classes.

The model is created on a Python Anaconda interpreter using TensorFlow’s Keras API library for machine learning. The CNN contains five convolution layers each followed by a max-pooling layer. These convolution and max-pooling layers highlight and pinpoint the identifiable features of each input. The model is then followed up by a dropout layer that randomly eliminates some of the input units to prevent over-fitting. The dropout factor is set to 0.5 due to the vast nature of multiple features within the data on spectrograms. The model then passes through a flatten layer and two dense layers. The final dense layer's output size is one since the loss objective function is binary-crossentropy (indicating a yes or no which is illustrated by the 1 or 0 on the tree path in Figure 3.1). Up to forty training epochs were performed for each network. For the first two CNNs, a steps-per-epoch value was specified to prevent over-fitting due to the high number of training data for the two most frequent classes. The dropout factor was reduced to 0.2 to compensate for the reduced training steps. The CNN at each decision point in the BHTNN is illustrated in Figure 3.2.

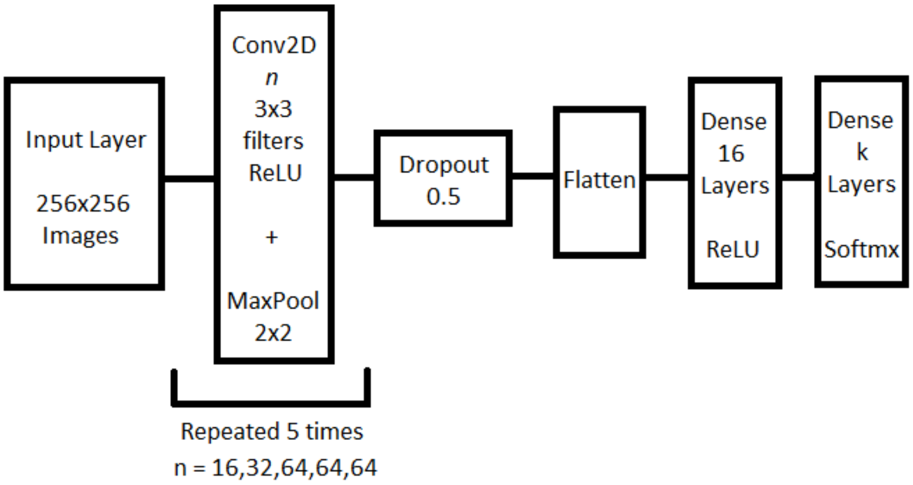


Figure 3.2

Illustration of the Convolution Neural Network (CNN) Model

A few other model callbacks were used to further prevent over-fitting. An early stopping callback was established to stop the training if the validation accuracy had not increased after fifteen epochs. Also, the ReduceLRonPlateau function was utilized – which reduces the learning rate of the optimizer function if the validation accuracy does not increase after five epochs. Finally, a model checkpoint was used to save the parameter weights of the epoch that yields the highest validation accuracy. These peak accuracy values are given in the results section.

Binary Huffman Tree Neural Network

Huffman encoding is typically used as a lossless compression algorithm by indicating frequently used values first with less frequently used value later. The bit-storage technique is optimal by using less bits for commonly occurring values and more bits for less commonly occurring values. Visually, this would represent a higher position in a binary tree, and a lower position in a binary tree respectively. The bit length indicates how many branches to travel in the tree in order to find a particular value. More bits will result in a higher time complexity for the worst-case scenario [7].

This thesis explores a machine learning model that will increase the accuracy of a canonical CNN in order to achieve a better performing model. These models are typically ideal for image classification; however, when CNNs have a higher number of output classes, the prediction accuracy of the model decreases [6]. This technique will extend the CNN by encompassing multiple CNNs in a Huffman Tree Architecture (HTA). The goal of this method is to improve accuracy of a multi-class CNN by implementing a HTA with binary CNNs. This will improve the classification accuracy of an exponentially unbalanced dataset (as compared to a flat multi-class categorical cross-entropy CNN). HTAs within neural networks have been used before, but with SVMs [8-9]. Since SVMs are not computationally efficient for multiple features within an input image, this thesis will explore a CNN-based HTA, an extensive CNN-based HTA with a reduced height, and an improved CNN-based HTA, utilizing entropy and impurity (Gini) index calculations to re-arrange nodes for better accuracy.

CHAPTER 4

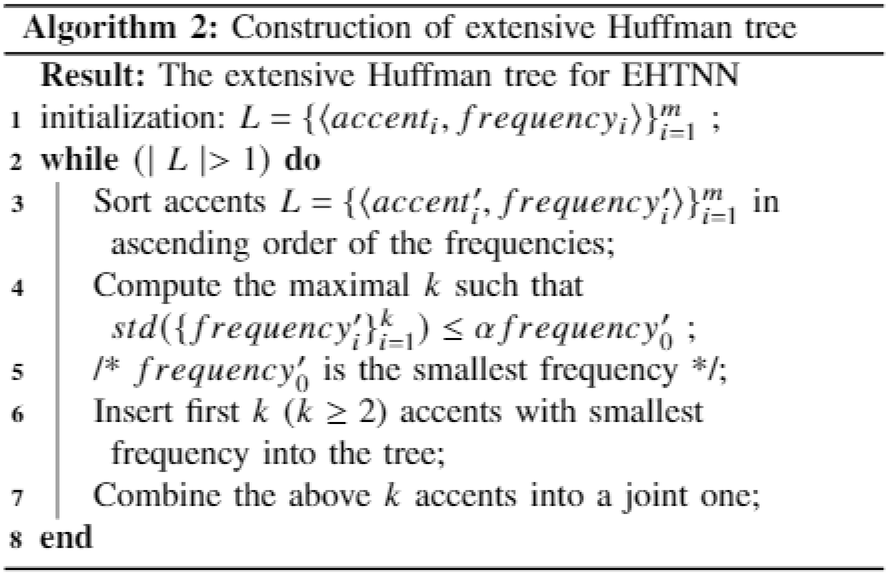
EXTENSIVE HUFFMAN TREE NEURAL NETWORK

For a given n-class classification problem, Binary Huffman Tree Neural Networks (BHTNN) always consist of (at most) (n-1) component binary classification neural networks; therefore, BHTNN lacks flexibility in architecture configuration. In the worst case (the height of the tree is n-1), a decision may be made by passing through (n-1) neural networks.

As a remedy, this thesis proposes an Extensive Huffman-Tree Neural Network (EHTNN):

- Each non-leaf EHTNN node has an arbitrary number of (two or more than two) sub-trees. As a result, a lesser number of component neural networks (binary or multi-class) are needed. The height of the resulting EHTNN can be reduced as well.
- The EHTNN is improved according to specific measurement metrics such as entropy or Gini index.

In order to select the appropriate numbers of classes to move to the canonical CNN portion of the EHTNN, Algorithm 2 is considered. The goal of the algorithm is to find the ideal number of classes, *k*, that will result in the standard deviation of *k* classes being lower than some arbitrary fraction, α of the frequency of the smallest class. Different from BHTNN (constructed using Algorithm 1), EHTNN fabricates multiple neural networks (or nodes) together using an extensive Huffman tree, whose construction is described by Algorithm 2.



```
Algorithm 2: Construction of extensive Huffman tree
Result: The extensive Huffman tree for EHTNN
initialization: $L = \{\langle accent_i, frequency_i \rangle\}_{i=1}^{m}$ ;
while (| L |> 1) do
    Sort accents $L = \{\langle accent'_i, frequency'_i \rangle\}_{i=1}^{m}$ in
      ascending order of the frequencies;
    Compute the maximal $k$ such that
      $std(\{frequency'_i\}_{i=1}^{k}) \leq \alpha frequency'_0$ ;
    /* $frequency'_0$ is the smallest frequency */;
    Insert first $k$ ($k \geq 2$) accents with smallest
      frequency into the tree;
    Combine the above $k$ accents into a joint one;
end
```

Accuracy of each decision point in the tree is one metric to consider when determining the overall effectiveness of a BHTNN with an unbalanced data-set. However, there is a concern for point-of-no-return if an inaccurate decision is made at a higher point in the tree for an input that belongs to another class. Further learning on an input data that does not belong to either class in a BHTNN can negatively affect the overall performance of each neural network in the tree. The EHTNN is a concept to combine a multi-class CNN with a Huffman tree to converge the benefits of both models. Referring back to the frequencies given from the numerical analysis in Table 3.1, the unbalance of the data becomes less apparent as we move down the table. This is the point in which we stop the Huffman tree and convert to a multi-class CNN to handle the remaining classes that are more balanced to each other as compared to the more frequently occurring classes at the top of the table. For our analysis, the splitting point will be after the fifth class in the table, resulting in a tree as illustrated in Figure 4.1. Additionally, reducing the height of the tree also decreases the time complexity and training efforts for the entire EHTNN.

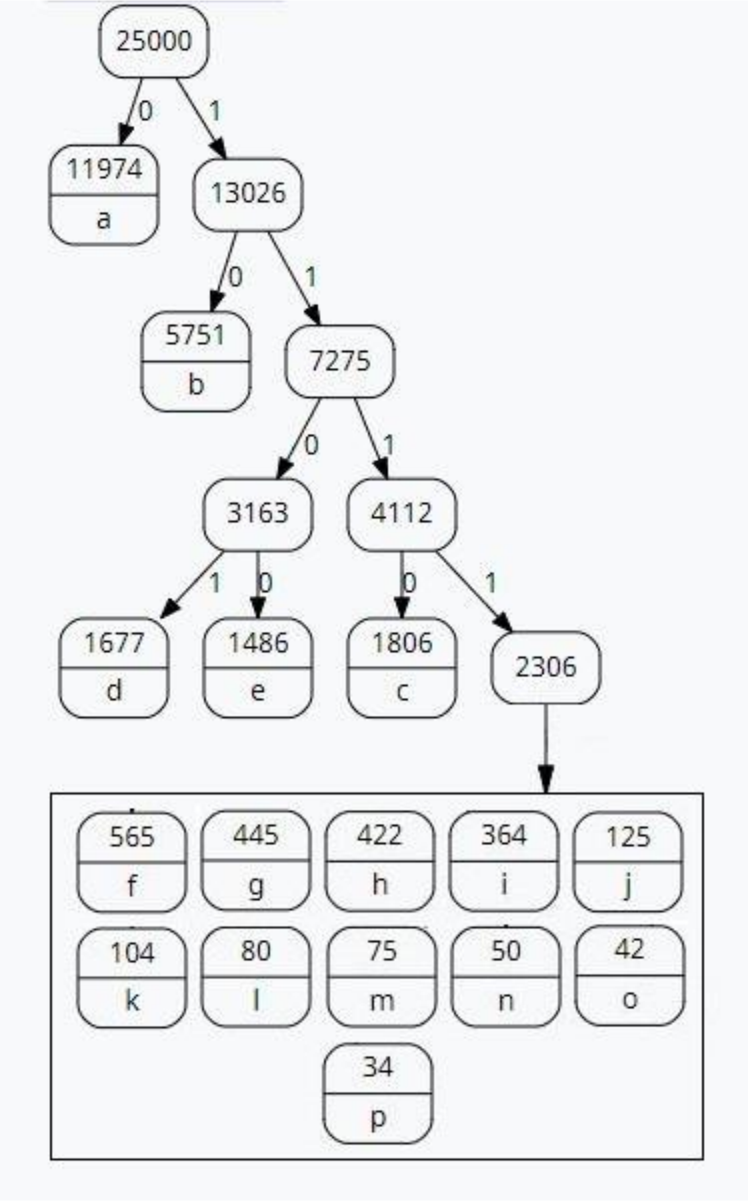


Figure 4.1

Extensive Huffman Tree Neural Network (EHTNN) with 6 Component Classifiers

Measurement Metrics of the EHTNN

Decision-tree-based neural networks can be measured for efficiency using multiple parameters. Two major factors are the entropy value and the impurity (Gini) index. By applying these concepts to the EHTNN, we can improve the configuration of tree through minimizing the Gini index and the entropy. Another measurement factor is the confusion matrix which visualizes the accuracy of each component neural network with a table of correct and incorrect guesses of each class.

Entropy

The information entropy value is a statistical measurement of the uncertainty in an outcome. Values are between zero and one. The goal of the decision tree is to arrange the decisions in the tree to minimize the entropy value. Our EHTNN is rearranged by changing the value of α to increase the number of classes move to the canonical portion of the tree. The entropy value is then re-computed to determine the improvement of the decision tree arrangement. The entropy formula is given by Equation 4.3 where p_j represents the probability of the class within its respective network. Each class within the tree has a computed entropy and it is weighted by multiplying by the probability of each class within the dataset, given by Equation 4.1. The weighted entropy values are summed to determine the final weighted entropy of the tree [8].

$$WH = \sum_{m} w_m * \ H_m$$

Equation 4.1

Weighted Entropy of Class Probability

The weight, w_m of the m-th component classifier is defined in Equation 4.2 where f is the frequency of the respective class, and the entropy of the m-th component classifier, H_m is defined in Equation 4.3.

$$w_m = \frac{f_m}{\sum_i f_i}$$

Equation 4.2

Computed Weight of m-th Class

$$H_m = -\sum_j p_j * \log_2 p_j$$

Equation 4.3

Entropy Value of the m-th Class

Gini Index

The Gini index is a measurement that indicates a level of dataset contamination or impurity. Similar to information entropy, the Gini index is a value between zero and one and a higher value indicates more data contamination. As a result, the goal of a decision tree is to reduce the value of the Gini index. A lower measurement value would signify the purity of the dataset. As a continuation of the information entropy arrangement, this thesis will apply a weighted Gini index given by Equation 4.4 where the weight, w_m of the m-th component classifier is defined in Equation 4.2 and the corresponding Gini index, G_m is given in Equation 4.5 [9].

$$WG = \sum_{m} w_m * G_m$$

Equation 4.4

Weighted Impurity (Gini) Index of Class Probability

$$G_m = 1 - \sum_{j} p_j^2$$

Equation 4.5

Impurity (Gini) Index of Class Probability

Hierarchical MultiClass Confusion Matrix

As addressed above, EHTNN consists of multiple neural network classifiers organized in a hierarchical architecture. The accuracy of m-th component classifier can be measured using micro-averaged F1-score, known as Precision and Recall and defined in Equation 4.6, where M indicates the multiclass confusion matrix for component classifier, m. This measurement illustrates that when we are calculating the metrics globally, all of the measures (accuracy, precision, recall, and micro-averaged F1-score) become equal.

$$Accuracy_m = Precision_m = Recall_m = \frac{\sum_i M_{ii}}{\sum_{i \neq j} M_{ij}}$$

Equation 4.6

F1 Score – Precision and Recall are Equivalent to Accuracy

The confusion matrices for the first two component classifiers of EHTNN are given in Table 4.1 and Table 4.2.

Table 4.1

Confusion Matrix – 1st Component Classifier (US vs NotUS – 67% Accuracy)

		Actual Class	
		US	NotUS
Predicted Class	US	**8022**	3952
	NotUS	4299	**8727**

Table 4.2

Confusion Matrix – 2nd Component Classifier (England vs NotEngland – 72% Accuracy)

		Actual Class	
		England	NotEngland
Predicted Class	England	**4130**	1621
	NotEngland	2051	**5224**

Following Equation 4.7, we have a global accuracy value of EHTNN that can be formulated as the weighted sum of the component classifiers.

$$WA = \sum_{m} w_m * \ Accuracy_m$$

Equation 4.7

Weighted Accuracy

Matthews Correlation Coefficient (MCC) or phi coefficient is a metric that is used to determine the quality of binary classifiers based on the confusion matrix. Values range from -1 to 0 to +1, incorrect prediction to random prediction to perfect prediction respectively. MCC is not considered for a metric in this thesis because it is only applicable for a completely binary classification Huffman tree. Therefore, it is not applicable for EHTNN. While many of the component classifiers in the EHTNN are binary, the canonical classifier is categorical. As a result, MCC cannot be applied to the entire tree.

CHAPTER 5

EXPERIMENTAL RESULTS

In order to establish a baseline, the same CNN model was trained in categorical mode; however, the loss function was changed to categorical-crossentropy with the number of output layers on the final dense layer set to sixteen (the total number of unique classes in [4]). This is the control model to make an improvement comparison to the BHTNN and EHTNN models. After several epochs, the learning converged on 47% accuracy. In comparison, the BHTNN and EHTNN both yielded (on average) a 66% accuracy. By the accuracy values alone, it can be observed that the Huffman-tree based models provide a better classification prediction for spectrograms. The validation accuracy of each CNN is given in Table 5.1.

Table 5.1

Accuracy Results for Each CNN in the BHTNN

Code	CNN	Accuracy
a	US/notUS	0.6718
b	England/NotEngland	0.7180
	Australia or Canada/neither	0.6562
de	Australia/Canada	0.5282
c	India/notIndia	0.6250
	Scotland, New Zealand, or Ireland/neither	0.6750
f	Scotland/notScotland	0.5797
hi	New Zealand/Ireland	0.6842
g	Africa/notAfrica	0.6078
	Philippines or Wales/neither	0.5926
jk	Philippines/Wales	0.7143
	Malaysia or Singapore/neither	0.6923
mn	Malaysia/Singapore	0.6250
l	Bermuda/notBermuda	0.6670
op	Hong Kong/South Atlantic	0.6670

As indicated, each CNN and class are not equally weighted to others, so the overall accuracy of the BHTNN and EHTNN cannot be computed by a mean. We can compute the total accuracy of each model by using a weighted mean. The formula for a weighted arithmetic mean is given in Equation 4.7. By applying Equations 4.1, 4.4, and 4.7, we obtain the results for each method as indicated in Table 5.2.

Table 5.2

Measurement Metrics of Each Method

Method	Accuracy (WA)	Entropy (WH)	Gini (WG)
Flat CNN	0.47	0.65	0.72
Binary HTNN	0.66	0.99	0.49
EHTNN (α = 0.5)	0.66	0.96	0.53
EHTNN (α = 1.5)	0.77	0.89	0.48

As indicated in the results above, the accuracy remains steady at 66% for both the BHTNN and the EHTNN, but rose slightly to 77% for the second EHTNN (α = 1.5). The entropy decreased from 0.96-0.99 to 0.89 on the second EHTNN while the Gini index fell from 0.72 on the flat CNN to 0.48 for the second EHTNN. When the EHTNN is improved for lower entropy and Gini index values, the accuracy of the model increases [10-11].

This work also includes a method for training a canonical SVM to support the decision of choosing the CNN methodology over SVM. The training time is given only for the canonical methodologies to illustrate the difference between SVM and CNN. This comparison is shown in Table 5.3

Table 5.3

Canonical Training Time for SVM and CNN Methods

Method	**Training Time (sec)**	**Epochs**	**Time per Epoch**
Flat SVM	1683	14	120.2
Flat CNN	1451	16	90.7

CHAPTER 6

CONCLUSION AND FUTURE WORK

The work of this thesis yielded a higher accuracy value for the prediction model by using a HTNN. As stated, the Huffman tree is not a new concept for machine learning models. It was previously used with an SVM-based model. Due to the limitation of SVM-based decision trees, this thesis explored increasing model prediction accuracy when a high number of output classes are present and the input must be trained on multiple features. The CNN-based HTNNs yielded an accuracy increase of roughly twenty percent to twenty-five percent (20%-30%). While the entropy value and Gini indices decreased with a HTNN vs canonical, the rearranged EHTNN lowered these values for improved accuracy. Even though a lower entropy and Gini index are ideal in a machine learning model, the higher values are an acceptable trade-off for higher accuracy.

Future work can be explored to further develop and optimize the EHTNN by utilizing more data preparation. Expanding on the audio processing functionality before generating a spectrogram can decrease the over-fitting potential of the model. The spectrograms produced by the Python matplotlib library were adequate for use in the model; however more analysis can be performed to detect a higher audio amplitude so the data sample isn't a brief moment of silence.

Additionally, more work is needed to further arrange a decision tree based on the lowest values for entropy and the Gini index. An ideal implementation would be to compute the entropy and Gini index for each possible arrangement of the decision tree and train the networks as such. Further research and studies can be conducted to re-arrange the decision tree after initial training to keep the accuracy high while keeping the entropy and Gini index low as the tree predicts and obtains more data throughout the lifecycle of the tree.

REFERENCES

[1] J. Cervantes, F. García-Lamont, A. López, L. Rodriguez, J. S. Ruiz Castilla and A. Trueba, "PSO-Based Method for SVM Classification on Skewed Data-Sets," in *Advanced Intelligent Computing Theories and Applications,* Cham, 2015.

[2] M. Koziarski, "Radial-Based Undersampling for imbalanced data classification," *Pattern Recognition,* vol. 102, p. 107262, 2020.

[3] M. Esmaeilpour, P. Cardinal and A. L. Koerich, "A Robust Approach for Securing Audio Classification Against Adversarial Attacks," *IEEE Transactions on Information Forensics and Security,* vol. 15, pp. 2147-2159, 2020.

[4] Mozilla, "Common Voice," 2017.

[5] R. Archibald and G. Fann, "Feature Selection and Classification of Hyperspectral Images With Support Vector Machines," *IEEE Geoscience and Remote Sensing Letters,* vol. 4, pp. 674-677, 2007.

[6] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng and M. Chen, "Medical image classification with convolutional neural network," in *2014 13th International Conference on Control Automation Robotics Vision (ICARCV),* 2014.

[7] A. Moffat, "Huffman Coding," *ACM Comput. Surv.,* vol. 52, 8 2019.

[8] D. Pomorski and C. Desrousseaux, "Improving performance of distributed detection networks: An entropy-based optimization," *Signal Processing,* vol. 81, pp. 2479-2491, 2001.

[9] D. Bertsimas and J. Dunn, "Optimal classification trees," *Mach Learn,* vol. 106, pp. 1039-1082, 2017.

[10] G. Zhang, "Support Vector Machines with Huffman Tree Architecture for Multiclass Classification," in *Progress in Pattern Recognition, Image Analysis and Applications*, Berlin, 2005.

[11] F. Wu, W. Hu and Y. Sun, "A novel multi-class fault diagnosis approach based on support vector machine of particle swarm Optimization and Huffman tree," in *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM),* 2015.

VITA

Jeremy Gordon Merrill, grew up in Soddy Daisy Tennessee where he attended grade school. Upon graduating Soddy Daisy High School in 2008, he pursued an undergraduate degree in Computer Engineering from Tennessee Technological University in Cookeville, Tennessee. After taking a five-year hiatus from college education, he returned to the University of Tennessee at Chattanooga in 2018 as a part-time student while continuing a full-time employment career. Jeremy graduated with a Master of Science degree in Computer Science in May of 2021.