Tone Detection in Vietnamese Speech with Convolutional Neural Networks

Ramon Casas

Deep Learning Final Project

Universitat Pompeu Fabra – 2025

Contents

1. Introduction 2
2. State of the Art 2
3. Methodology 3
    3.1. Data Analysis 3
        3.1.1. Data Collection: Building the Datasets 3
        3.1.2. Data Preprocessing: Preparing for Training 4
    3.2. Models and Optimization 4
        3.2.1. Custom CNN 4
        3.2.2. Transfer Learning 4
4. Experiments 5
    4.1. Custom CNN model 5
    4.2. Wav2Vec2 5
    4.3. ToneNet 5
5. Results 6
    5.1. Custom CNN model 6
    5.2. Wav2Vec2 Transfer Learning Results 8
    5.3. ToneNet Transfer Learning Results 8
6. Discussion 9
7. Conclusions 10
8. References 10

1. Introduction

In tone (or tonal) languages, "word meanings or grammatical categories (such as tense) are dependent on pitch levels" (Crystal, 2008: 486). Vietnamese is a tone language, which means that each of the syllables that form its vocabulary carry a tone contour that results in a difference in meaning. It has six tones (ngang, huyền, sắc, hỏi, ngã, nặng), which are orthographically marked by a diacritic mark and which depending on the region are fully pronounced or not. In the Hanoi dialect, the one widely used in official settings in Vietnam, these six tones are fully reflected. These six tones are marked as follows (Binh Ngo, 2021: 277):

ta “we” (mid-level)

tà “evil” (low-falling)

tá “dozen” (high-rising)

tả “describe” (low-falling-rising)

tã “diaper” (high-rising broken)

tạ “weights” (low-falling broken)

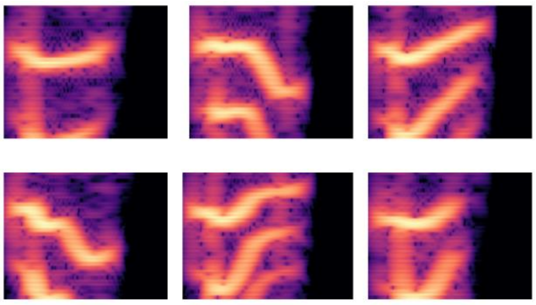


Figure 0. MEL spectrograms of the different tones for the syllable ba *extracted from our drill dataset (see section 3.1)*

Adult learners of tonal languages who have had no prior exposure to this kind of phonetical distinction may struggle when attempting to learning them, specially (but not only) in their early stages of the learning process. This is where a feedback tool that is able to assess learners whether their pronunciation is accurate or not can be extremely useful.

In this project, we present a deep-learning-based Vietnamese tone classification system. To do so, we train convolutional neural networks, and we explore the possibility of training the model using synthetic data generated by state-of-the-art text-to-speech models, together with a small amount of real speakers' data in order to see the performance in the classification task on real world data. After some experiments with custom CNNs and transfer-learning from previous works on Mandarin (Gao et al., 2019), we show how our model is able to learn and classify tones when spoken by the voices it has been trained with (achieving ~97% accuracy), but still struggles with the unseen voices (with performances ranging from ~29% to ~69% of accuracy). Our work shows a solid first step approach to the Vietnamese tone classification task and demonstrates the feasibility of the task in realistic conditions. It paves the way towards the deployment of pronunciation-feedback applications that help learners in this difficult task.

2. State of the Art

Automatic tone classification has been studied mainly applied to the case of Mandarin Chinese. Gao et al. (2019) introduced ToneNet, a CNN taking log-mel spectrograms as input an achieving over 99% accuracy on their test sets. Their work shows that CNN architectures allow to capture pitch contours and distinguish between Mandarin tones. Also in the Mandarin scope, Tang and Li (2021) proposed and “end-to-end” architecture to explore the short-term contextual analysis in order to distinguish in-sentence tones, i.e. non isolated syllables. Finaly, Huang, Hu, and Xu (2017) explored RNN-based tone modelling, feeding sequences of spectral features into bidirectional LSTMs.

For Vietnamese, prior work by Bui Thanh Hung (Bui, 2020) applies generic CNNs for a Vietnamese voice classification task, but does not isolate tone in his approach. In fact, as Vietnamese pronunciation of tones varies throughout the country, we reckon that introducing tone detection in the model could improve the voice classification task. To our knowledge, no prior work has targeted Vietnamese tone recognition in isolation using deep learning; our project thus bridges that gap by adapting Mandarin-focused architectures and by synthesizing balanced Vietnamese syllable corpora for training.

3. Methodology

3.1. Data Analysis

3.1.1. Data Collection: Building the Datasets

When first facing the dataset collection task, we searched for some Vietnamese corpuses available online. One of the most extensive ones is the VIVOS dataset[1], which consists of 15 hours of recording speech with text files containing the audio transcriptions. The first intended approach was to build a script in order to trim individual syllables. However, this was a difficult task and automatic trimming did not give out good results. Moreover, tones in contact tend to behave differently and can be influenced by surrounding tones, a reason why we decided to change the approach and look for datasets with individually recorded syllables.

Due to the time constraints of the project, gathering the necessary amount of real voice data was a really difficult task. Therefore, we decided to find an alternative: using text-to-speech (TTS) tools in order to obtain data. What we did was take the unique syllables[2] from the VIVOS dataset (a total of 4,850) and used the Google Text To Speech API in order to generate[3] the syllables with 6 different voices corresponding to 2 of their best performing models: vi-VN-Wavenet-A, vi-VN-Wavenet-B, vi-VN-Wavenet-C, vi-VN-Wavenet-D, vi-VN-Neural2-A, vi-VN-Neural2-D. Once the *.wav* files for each syllable were obtained, we built a balanced dataset in which every tone had the same number of syllables, i.e. we took the number of syllables corresponding to the least represented tone. The final dataset of synthetic data consists of 11,217 syllables.

Apart from synthetic data, we still wanted some representation of real voices for training our model. Different approaches were considered, and finally we decided to obtain a set of syllables from pronunciation drill exercises from a Vietnamese as a foreign language textbook (Ngo, 2015). From the audios of the book, we manually erased the English instructions using the audio tool Audacity, and then we did an automatic syllable trimming with a python script. The resulting set consists of 1,557 syllables corresponding to two native Vietnamese speakers. All scripts used for this dataset building are available in the *scripts_utilities* directory of our GitHubt repository.

Finally, we collected a small set of real voice syllables from a wider variety of speakers from Forvo[4], which is a pronunciation website in which (native) speakers around the world upload audio recordings of specific words. From this site, we downloaded a total of 52 syllables, which have been used exclusively to test the performance of our models.

Lastly, for training purposes we tried different combinations of the datasets: we trained with only synthetic voices, with real voices, and with a combination of both. For this, we built what we called a small dataset with 1,800 synthetic samples plus the 1,557, and a large one consisting of the whole set of both synthetic and real voice data. In all the training steps we performed a split of 80% training, 10% validation, and 10% testing. Our final datasets are described in the following table:

Dataset[5]	**Number of Samples**
Synthetic Dataset Large	11,217
Synthetic Dataset Small	1,800
Real Voice Dataset - Pronunciation Drills	1,557
Combined Dataset Small	3,357
Combined Dataset Large	12,774
Real Voice Dataset – Forvo Syllables – Unseen Voices	52 [ONLY FOR TESTING]

Figure 1. Dataset descriptions with number of samples per set

[1] Available at https://www.kaggle.com/datasets/kynthesis/vivos-vietnamese-speech-corpus-for-asr
[2] We used the *unique_syllable_extraction_vivos.py* script available in the GitHub repository
[3] Using the *synth_syllables_Cloud_FullSet_Vivos.py* script available in the GitHub repository
[4] https://forvo.com/
[5] Throughout the report we refer to these datasets in different ways, which are listed in this table.

3.1.2. Data Preprocessing: Preparing for Training

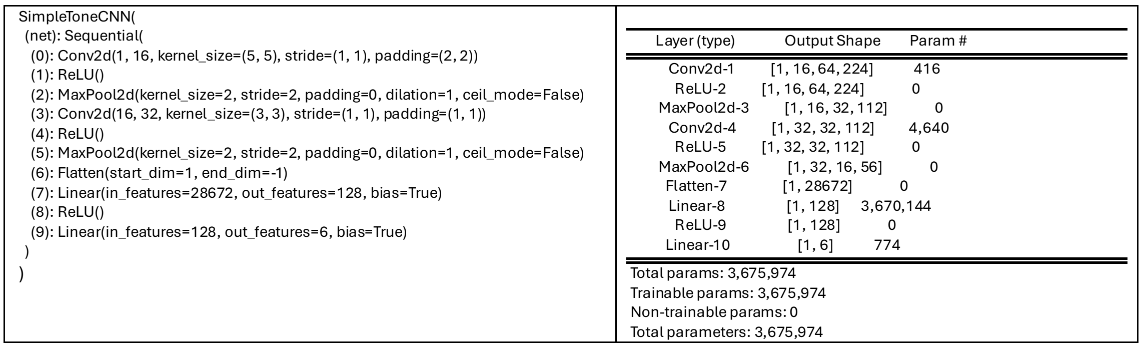
Until this point, the dataset consists of *.wav* audio files of each syllable. In order to use these files for training and building our model, we had to preprocess the data. To do so, we got inspiration from the ToneNet approach (Gao et al., 2019), where they extracted the MEL spectrograms –see Roberts (2020) for a introduction to the concept– and turned the audio files into images to feed the CNN. In our case, although we also generated *.png* for visual inspection, we fed *.npy* arrays to the model. This means that while the ToneNet CNN received inputs with 3 channels, our CNN received 1-channel inputs. To do the MEL extraction, we used the librosa library[6] to turn each 16 kHz waveform into the fixed-size mel-spectrograms required by our CNNs. Our script walks through every *.wav* under a root folder, computes a 64 × 225 mel-spectrogram (50–350 Hz band, as suggested in ToneNet as well), and writes both a *.npy* tensor and a *.png* for visualization. We also built a *metadata.csv* file linking each example to its syllable, tone, mode, and file paths.

In order to increase performance in our models on unseen voices, we thought of applying data augmentation. We took the idea of SpecAugment (Park et al., 2019)[7] to do so: a technique that randomly masks out contiguous bands along the time and/or frequency axes of the log-mel spectrogram, forcing the model to learn more robust, context-aware features. In our spec_augment helper, we zero out up to 30 % of the mel-bins (frequency mask) and up to 30 % of the time-frames (time mask), replacing those regions with −80 dB (the lower clipping bound). We then wrap our (X, y) tensors in an AugmentedMelDataset that, with 50 % probability on each sample during training, applies these masks before re-casting the result as a PyTorch tensor.

3.2. Models and Optimization

3.2.1. Custom CNN

In order to train our model, we first implemented a lightweight CNN on single-channel 64x225 inputs with PyTorch. After trying different configurations, we decided to proceed with the following one:



```
SimpleToneCNN(
 (net): Sequential(
  (0): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (1): ReLU()
  (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (4): ReLU()
  (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (6): Flatten(start_dim=1, end_dim=-1)
  (7): Linear(in_features=28672, out_features=128, bias=True)
  (8): ReLU()
  (9): Linear(in_features=128, out_features=6, bias=True)
 )
)
```

```
   Layer (type)        Output Shape        Param #
==================================================
     Conv2d-1       [1, 16, 64, 224]         416
       ReLU-2       [1, 16, 64, 224]           0
  MaxPool2d-3       [1, 16, 32, 112]           0
     Conv2d-4       [1, 32, 32, 112]       4,640
       ReLU-5       [1, 32, 32, 112]           0
  MaxPool2d-6        [1, 32, 16, 56]           0
    Flatten-7             [1, 28672]           0
     Linear-8               [1, 128]   3,670,144
       ReLU-9               [1, 128]           0
    Linear-10                 [1, 6]         774
==================================================
Total params: 3,675,974
Trainable params: 3,675,974
Non-trainable params: 0
Total parameters: 3,675,974
```

Figure 2. Custom CNN model description

We trained this network on the mel-spectrograms (1x64x225), using the Adam optimizer (initial learning rate 1x10^{-3}) and cross-entropy loss. To monitor learning, we recorded both loss and classification accuracy on training and validation splits at every epoch, and we evaluated final performance on held-out test folds as well as on our external Forvo set of unseen voices.

3.2.2. Transfer Learning

Rather than learning all filters from scratch, we also explored two transfer-learning strategies that leverage pre-trained speech representations:

Wav2Vec2-based Classifier

[6] https://librosa.org/doc/latest/generated/librosa.feature.melspectrogram.html
[7] We also used a repository that references this paper: https://github.com/pyyush/SpecAugment

We took Facebook's "wav2vec2-base" model[8] as a frozen feature extractor and appended a lightweight classification head: a linear projection to 256 dimensions with ReLU and dropout, followed by a final linear layer over the six tones. Inputs were our wav files from the drill dataset processed by the official Wav2Vec2Processor into model-compatible tensors. During fine-tuning, only the head's parameters were updated (learning rate 1×10^{-4}), while the base model remained fixed. We trained for 50 epochs, splitting our pronunciation-drill recordings into 80 % train, 10 % validation, and 10 % test, and tracked train/validation accuracy curves as well.

ToneNet-based Classifier

ToneNet (Gao et al., 2019) is a CNN originally trained on Mandarin mel-spectrograms (64×225, 50–350 Hz bandpass). We converted their provided Keras model[9] to ONNX, imported it into PyTorch with onnx2pytorch, and swapped out the final output layer. Mandarin has 4 tones (or 5 if we take into account the neutral tone, but they did not consider it in ToneNet), so we had to change their 4-way output layer for a new 6-way head. We then experimented with four "freeze modes", which we detail in the next section.

4. Experiments

In order to assess performance of our different approaches described above, we conducted several experiments. All of our experiments follow a similar evaluation protocol: first we construct the train/validation/test splits (80 %/10 %/10 %) on the synthetic, drill, or combined datasets; then we train each model variant and report its accuracy on both the held-out test fold and a separate Forvo set of 52 real-speaker syllables. In every case we track classification loss and accuracy on the training and validation sets at each epoch, to inspect convergence and overfitting.

4.1. Custom CNN model

We first established performance of the SimpleToneCNN when trained solely on synthetic TTS data, solely on real pronunciation-drill recordings, and on their combination—both in "small" (1 800 synth + 1 557 drill) and "large" (11 217 synth + 1 557 drill) variants. For each dataset, we trained for 25 epochs. This allowed us to measure whether combining synthetic and real data yields better generalization. Additionally, we used our combined datasets and experimented with using or not using data augmentation.

4.2. Wav2Vec2

We fine-tuned the wav2vec2-base model with our pronunciation-drill dataset. We froze all convolutional feature-extractor parameters and trained only the appended head for up to 50 epochs. We evaluated on both the drill test fold and the Forvo set to assess generalization to unseen speakers.

4.3. ToneNet

We converted ToneNet's pre-trained Mandarin model to PyTorch, replaced its final layer with a 6-way output head, and experimented with four freeze modes: Mode 0 ("head-only") froze all layers except the new head; Mode 1 unfroze the head plus the two 1 024-unit dense layers; Mode 2 additionally unfroze the last convolutional block; Mode 3 performed full fine-tuning of every layer.

For each mode, we ran a grid over learning rates ($1\text{x}10^{-3}$, $1\text{x}10^{-4}$, $1\text{x}10^{-5}$), batch sizes (16, 32), and augmentation on/off. We trained each configuration for 8 epochs using discriminative learning rates —lowest for earlier convolutional layers, medium for dense layers, and highest for the new head— and we logged per-epoch loss and accuracy on both training and validation sets. With the best 10 performing models during 8 epochs, we did a second round of training. With these trained models, we computed final accuracies on the test splits and on the Forvo

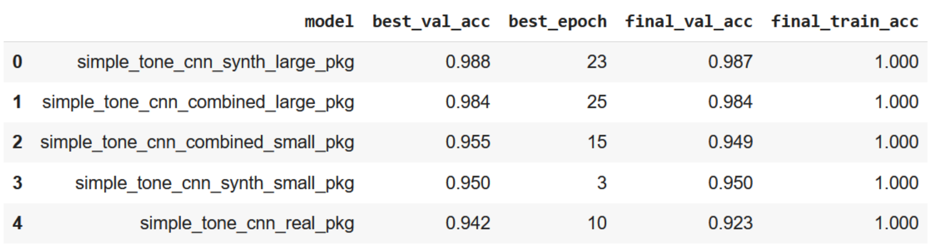
[8] Available at https://huggingface.co/docs/transformers/model_doc/wav2vec2
[9] Available at their GitHub repository: https://github.com/saber5433/ToneNet/tree/master

pronunciation set to compare the impact of layer freezing, learning rate scheduling, and SpecAugment when adapting ToneNet to Vietnamese.

5. Results

5.1. Custom CNN model

First, we trained our custom CNN using different sets of data (which were described in section 3). Here we have the results of the trainings:



	model	best_val_acc	best_epoch	final_val_acc	final_train_acc
0	simple_tone_cnn_synth_large_pkg	0.988	23	0.987	1.000
1	simple_tone_cnn_combined_large_pkg	0.984	25	0.984	1.000
2	simple_tone_cnn_combined_small_pkg	0.955	15	0.949	1.000
3	simple_tone_cnn_synth_small_pkg	0.950	3	0.950	1.000
4	simple_tone_cnn_real_pkg	0.942	10	0.923	1.000

Figure 3. Performance of the different models using our SimpleToneCNN model architecture

And here the training and validation loss and accuracy curves throughout epochs:

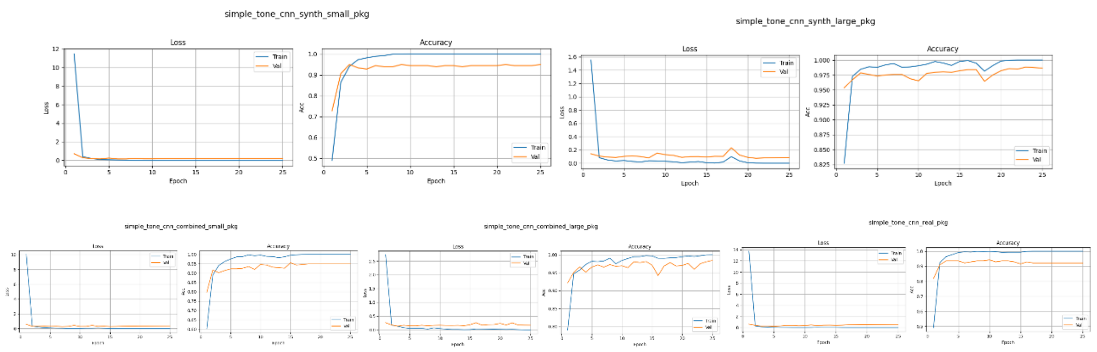


Figure 4. Loss and training curves of the different training process of our SimpleToneCNN

It is interesting also to assess performance on the other testing sets when training. For example, checking performance on real voices when training with synthetic ones or vice versa.

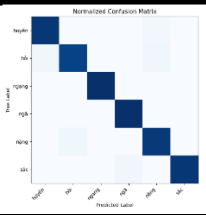
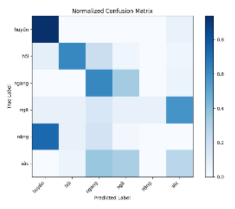
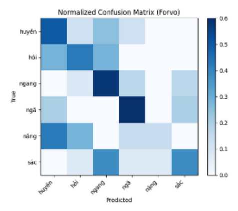
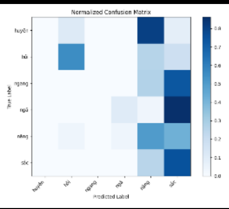
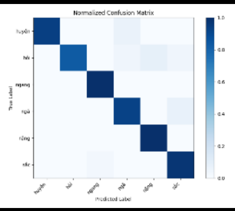
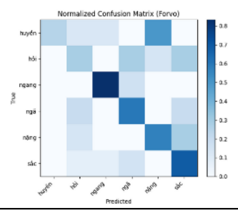
	Synthetic Test	Drill Test	Forvo Test
Synthetic Data			
Real Voices – Drill Dataset			

Figure 5. Confusion matrices of the models comparing their performance on the different test sets.

Training combined datasets and evaluating in separate sets

Next, we present the results of training using the combined datasets (i.e. mixing the synthetic data and the real pronunciation drill data together) and testing their performance on different test splits:

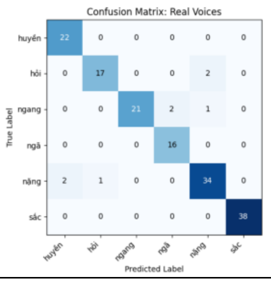
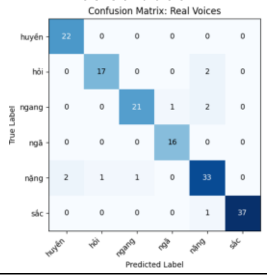
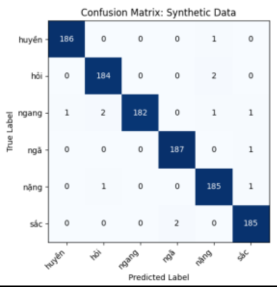
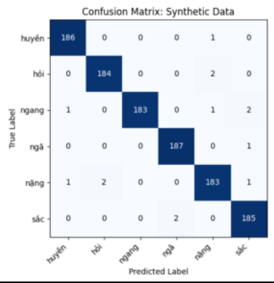
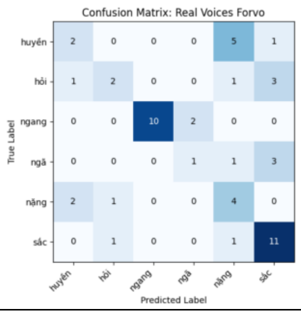
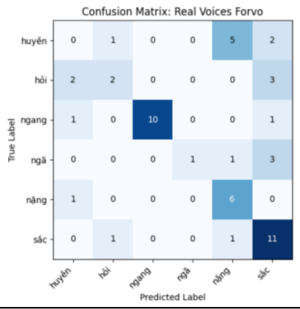
	Combined Large Training	Combined Small Training
Drill Dataset Test	95% acc. 	93.59% acc. 
Synthetic Dataset Test	98.84% acc. 	98.75% acc. 
Forvo Dataset Test	57.69% acc. 	57.69% acc. 

Figure 6. Confusion matrices of the different models with the combined large and small datasets

Data Augmentation

The last training results of our custom CNN model shows the performance of the model when trained using the combined datasets and tested with the Forvo syllable dataset.

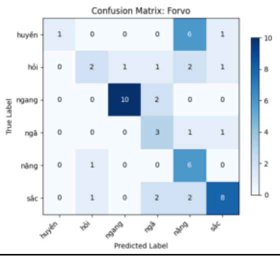
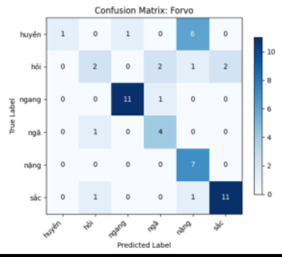
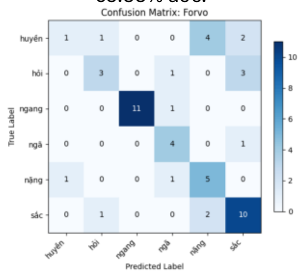
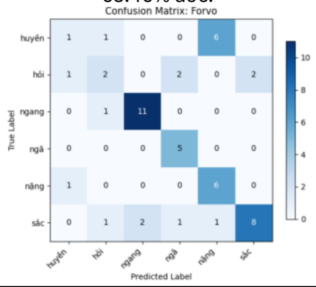
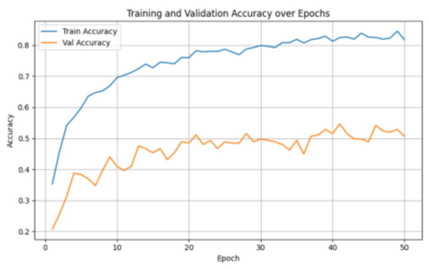
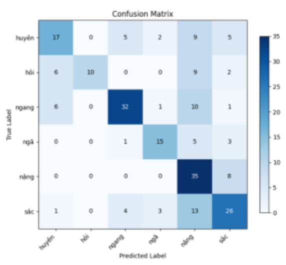
	Without Data Augmentation	With Data Augmentation
Combined Small Dataset	58% acc. 	69% acc. 
Combined Large Dataset	65.38% acc. 	63.46% acc. 

Figure 7. Confusion matrices of the CNN model trained with the different combined datasets on the Forvo syllables

5.2. Wav2Vec2 Transfer Learning Results

The training and validation accuracy curves over epochs of the training with the Wav2Vec2 fine-tuned model (with the combined small dataset), as well as the confusion matrix on the test set are shown below:





Figures 8 and 9. Training and validations curves of the Wav2Vec2 model (left) and its confusion matrix on the test set (right)

5.3. ToneNet Transfer Learning Results

Using the same methodology we described in section 5.1. for the custom CNN, i.e. training with the combined dataset and testing separately, we obtained the following results:

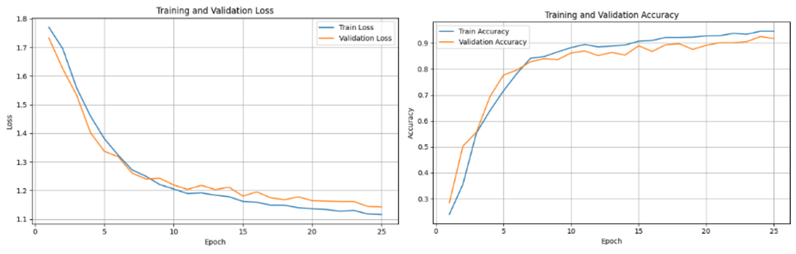


Figure 10. Loss and accuracy curves of the ToneNet fine-tuned model with only the last layer fine-tuned

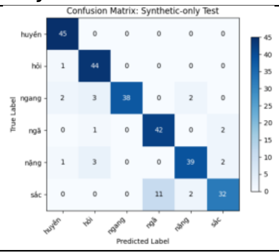
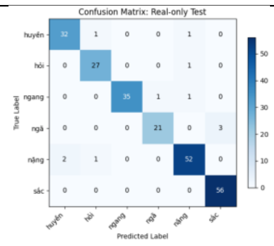
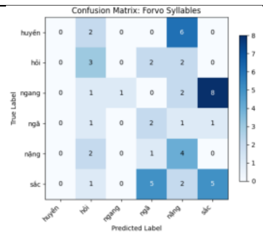
Synthetic – 89% acc.	Drills – 95% acc.	Unseen Voices – 29% acc.
		

Figure 11. Confusion matrices of the ToneNet fine-tuned model with only the last layer fine-tuned

Then, we conducted the experiments described in section 4.3. As described before, we first performed some experiments with 8 epochs. Here we present the Top 5 Configurations for the test and the unseen voices sets:

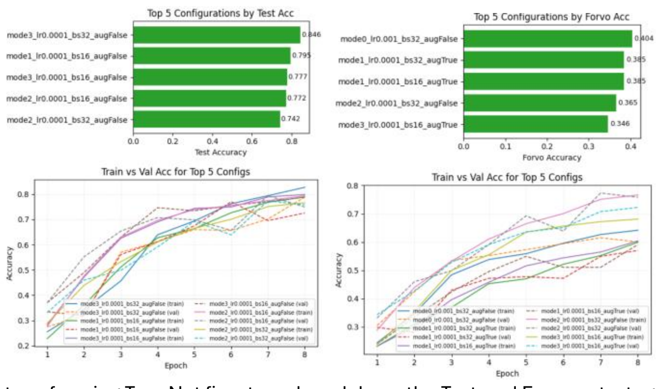
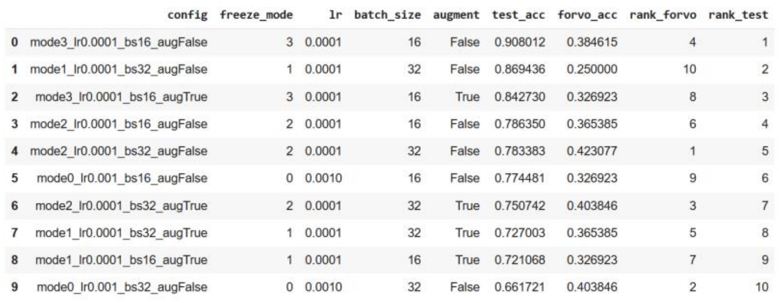


Figure 12. Performance of the best-performing ToneNet fine-tuned models on the Test and Forvo sets, together with their Accuracy curves

Afterwards, the top 10 models were trained again for 25 epochs. These were our best performing models ordered by test accuracy:



	config	freeze_mode	lr	batch_size	augment	test_acc	forvo_acc	rank_forvo	rank_test
0	mode3_lr0.0001_bs16_augFalse	3	0.0001	16	False	0.908012	0.384615	4	1
1	mode1_lr0.0001_bs32_augFalse	1	0.0001	32	False	0.869436	0.250000	10	2
2	mode3_lr0.0001_bs16_augTrue	3	0.0001	16	True	0.842730	0.326923	8	3
3	mode2_lr0.0001_bs16_augFalse	2	0.0001	16	False	0.786350	0.365385	6	4
4	mode2_lr0.0001_bs32_augFalse	2	0.0001	32	False	0.783383	0.423077	1	5
5	mode0_lr0.001_bs16_augFalse	0	0.0010	16	False	0.774481	0.326923	9	6
6	mode2_lr0.0001_bs32_augTrue	2	0.0001	32	True	0.750742	0.403846	3	7
7	mode1_lr0.0001_bs32_augTrue	1	0.0001	32	True	0.727003	0.365385	5	8
8	mode1_lr0.0001_bs16_augTrue	1	0.0001	16	True	0.721068	0.326923	7	9
9	mode0_lr0.001_bs32_augFalse	0	0.0010	32	False	0.661721	0.403846	2	10

Figure 13. Performance of the best-performing models of the fine-tuned ToneNet CNN after hyperparameter experimentation

6. Discussion

Our experiments illuminate several key insights. First, tone classification on isolated syllables is highly effective: both a small custom CNN and transfer-learned ToneNet exceed 90 % accuracy when test speakers match the training distribution (See Figures 3 and 4). This underscores that mel-spectrograms in the 50–350 Hz band, paired with deep CNN filters, are able to capture static tone contours. Figures 5 and 6 show that using the combined datasets, i.e. mixing synthetic and real voices in the same training set, led to good performance on both kinds of testing sets. This was not the case when we trained only on synthetic or only on real voices: performance on the other test sets was bad.

Second, generalization to unseen voices remains a major challenge. All models —custom CNN, wav2vec2 head, and fine-tuned ToneNet— had a relatively bad performance (to 20–69%) on our Forvo set. Data augmentation with SpecAugment mitigated this to some extent, improving Forvo accuracy by up to 12 points until 69% using our custom CNN, but did not reach good enough results. This suggests that variation in speaker pitch range, recording conditions, and other effects introduce distribution shifts that our isolated-syllable models do not fully address. If we take a look at Figure 7, which shows the best performance on the unseen voices dataset with 69% accuracy, we can see that some tones are classified very well, while some others are not. The two low tones (huyền and nặng) seem to be classified similarly, which is what is making the accuracy drop. This makes sense and gives hope because they are not two random different tones: they share the fact that they are low. Apart from these two, the other tone that our model is unable to detect well most of the times is hỏi.

Third, our engineered transfer-learning from large speech models (wav2vec2) yielded no good results for pure tone classification. Figure 8 shows that while the model learns to distinguish the training syllables, the validation set remains at ~55%. This is a clear sign of overfitting and demonstrates that the model is not learning to distinguish syllables based on their tone. Hence, while wav2vec2 encodes phonetic and articulatory patterns, it does not natively emphasize pitch contours and hence tone detection.

Fourth, fine-tuning ToneNet's dense and convolutional blocks similarly boosts in-domain accuracy but fails to generalize out-of-domain, i.e. to perform well on the unseen voices of the Forvo dataset: in a well performing model in the test sets using the same training logic we used for the custom CNN, the performance on the Forvo dataset wast of 29% accuracy. Even with many experiments we were not able to obtain a performance greater than 42% accuracy on the Forvo dataset, indicating that pre-training on Mandarin speech alone is not sufficient to transfer Vietnamese tonal nuance. Our intuition was that having been trained on many different voices of Mandarin speakers would enable us to obtain great results on our Vietnamese unseen voices, but experiments showed either that we were wrong or that we have not been able to perform the optimal fine-tuning.

Finally, our results suggest that data diversity —in speaker profiles, recording conditions, and syllable contexts— may matter more than incremental architecture or hyperparameter tweaks and model engineering. While careful layer freezing and discriminative learning rates optimize in-domain performance, real-world tone-feedback tools will require richer, more varied real-speech corpora and further augmentation strategies (e.g. pitch shifting, reverberation) to achieve robust cross-speaker generalization.

7. Conclusions

In this project we have presented a complete pipeline for Vietnamese tone classification using CNNs. We have (1) generated synthetic data via Google Cloud TTS over six high-quality voices; (2) implemented a mel-spectrogram feature extraction pipeline inspired by ToneNet (Gao et al., 2019); (3) designed a custom CNN architecture that achieved ~96% on synthetic+real data, but which struggled to perform on unseen voices, where we reached a maximum accuracy performance of 69%; (4) performed an extensive hyperparameter experimentation of transfer learning using a preexisting CNN trained on Mandarin (ToneNet, ibid.) that showed that this may not be the best strategy for our task; and (5) showed how ASR models like Wav2Vec2 may not be the best solution for such a specific task as tone recognition.

Future work should focus on expanding the available data of syllables in isolation in order to train our models with a wider variety of data. Moreover, it is also a future task to explore if other data augmentation techniques can overcome the lack of data challenge. Ultimately, these advances will enable real-time pronunciation feedback tools to support Vietnamese learners in mastering its challenging tonal system.

8. References

Bui, T. H. (2020). Vietnamese voice classification based on deep learning approach. *International Journal of Machine Learning and Networked Collaborative Engineering*, 4(4), 171–180.

Crystal, D. (2008). *A Dictionary of Linguistics and Phonetics* (6th ed.). Blackwell Publishing.

Gao, Q., Sun, S., & Yang, Y. (2019). ToneNet: A CNN model of tone classification of Mandarin Chinese. In *Proceedings of Interspeech* 2019 (pp. 3367–3371).

Huang, H., Hu, Y., & Xu, H. (2017). Mandarin tone modeling using recurrent neural networks. *arXiv*. arXiv:1711.01946

Ngo, B. (2015). *Elementary Vietnamese* (3rd ed.). Tuttle Publishing.

Ngo, B. (2021). *Vietnamese: An Essential Grammar*. Routledge.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., & Le, Q. V. (2019). SpecAugment: A simple data augmentation method for automatic speech recognition. In *Proceedings of Interspeech 2019* (pp. 2613–2617).

Roberts, L. (2020, March 6). *Understanding the Mel spectrogram*. Medium. https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53

Tang, J., & Li, M. (2021). End-to-end Mandarin tone classification with short-term context information. *arXiv*. arXiv:2104.05657