
MULTI-STYLE TRAINING FOR SOUTH AFRICAN CALL CENTRE AUDIO

A PREPRINT

Walter Heymans and Marelle H. Davel
Faculty of Engineering,
North-West University, South Africa
and CAIR, South Africa
walterheyman07@gmail.com

Charl van Heerden
Saigen, South Africa
charl@saigen.co.za

ABSTRACT

Mismatched data is a challenging problem for automatic speech recognition (ASR) systems. One of the most common techniques used to address mismatched data is multi-style training (MTR), a form of data augmentation that attempts to transform the training data to be more representative of the testing data; and to learn robust representations applicable to different conditions. This task can be very challenging if the test conditions are unknown. We explore the impact of different MTR styles on system performance when testing conditions are different from training conditions in the context of deep neural network hidden Markov model (DNN-HMM) ASR systems. A controlled environment is created using the LibriSpeech corpus, where we isolate the effect of different MTR styles on final system performance. We evaluate our findings on a South African call centre dataset that contains noisy, WAV49-encoded audio.

Keywords Automatic speech recognition · Multi-style training · Call centre audio · WAV49 encoding

1 Introduction

Automatic speech recognition (ASR) has been an active field of research since the 1970s and is still being developed and improved today [1, 2, 3]. The word error rate (WER), a typical measure of performance for ASR systems, has been significantly reduced over the last few decades. Main factors that contributed to this improvement were recent developments in deep learning, increased computational power of modern computers, specifically graphical processing units, and large amounts of collected data [4].

ASR systems tend to perform more poorly when there is a large mismatch between training and testing data. Factors that contribute to this mismatch include various forms of background noise, microphone distortion, different recording environments, encoding noise, people that speak in different speaking styles and accents, etc. It is difficult for an ASR system to generalise to new audio with different conditions if no attempt is made during training to handle such variability in the data.

A popular technique to address mismatch in audio for ASR is multi-style training (MTR) [5, 6, 7, 8, 9, 10]. MTR aims to transform the training data to be more representative of the testing data and to learn robust representations of the training data. A new training dataset is created from the existing set by adding a series of MTR styles using data augmentation. These can include: changing the speed and volume [5], speech style [6] or sampling rate [7]; adding time and frequency distortions [9] or background noise; and simulating reverberation [10]. The styles are typically chosen without knowledge of the testing conditions, and must still be able to handle a wide variety of mismatch. In addition, the number of styles that are added must be taken into consideration, because the computational cost of training an ASR system increases significantly with each style that is added.

This is a preprint - the final authenticated publication is available online at:
https://doi.org/10.1007/978-3-030-95070-5_8

In this work, we analyse the effects of MTR in a controlled environment using the LibriSpeech corpus [11]. Speed, volume and noise perturbation are added to clean training data and evaluated on WAV49-encoded development and test sets. We show the performance gain as a result of individual and combined MTR styles. This provides a practical approach to improve ASR systems efficiently on WAV49-encoded audio, often used in South African call centres.

Section 2 gives a brief overview of related work in the field of MTR for deep neural network (DNN) based speech recognition. In Section 3, we introduce the call centre dataset and how we created a controlled environment using the LibriSpeech corpus. Our experimental setup is explained in Section 4 and the results are presented in Section 5. Finally, the key findings are discussed in Section 6.

2 Related Work

We are not aware of any studies that investigate MTR for DNN-based call centre ASR. There are studies that investigate the effects of different perturbation levels, but none of them focus on call centre audio. The perturbation types used in these studies include additive noise and room impulse responses [8], speed [5] and volume [12].

Doulaty et al. investigated a method to automatically identify noise perturbation levels in a target set of utterances [8]. They used a ‘voice-search’ dataset for their experiments. A noisy test set was created by perturbing a clean set with different perturbation styles including additive background noise and room impulse responses using different signal-to-noise ratios (SNRs). MTR was used to train a number of multi-layer perceptron (MLP) models, each with different perturbation levels. The MTR model that was trained on data with the closest matched conditions was then used to evaluate the target utterance. Their study revealed that accurately matched noise perturbation levels result in better ASR performance. This showed the importance of selecting conditions for MTR that are matched to that of the test set.

Speed perturbation is a common technique that is applied widely in MTR setups. Ko et al. [5] investigated making two copies of the original dataset, one slowed down by 10% and another made 10% faster. An average relative improvement of 4.3% was observed across 4 different tasks, with a relative improvement of 6.7% on Switchboard [13], a conversational telephone corpus. The improvement on the full LibriSpeech corpus [11] was 3.2% and only 0.32% relatively on the ASpiRE corpus [14]. The authors attribute this small improvement on the ASpiRE corpus to simulated reverberation that was already applied to the training data.

Gokay and Yalcin investigated the effects of speed and volume perturbation in a low-resource 10 hour Turkish dataset of natural speech from a professional speaker [12]. They used an end-to-end ASR system based on Deep Speech 2 [15]. Adding speed and volume perturbation, individually and together improved their WER by between 8.2% and 12.9% relatively. They also added 10 hours of new training data without MTR and the improvement was 26.3% relatively, much better than any MTR technique they used. The effect of MTR was amplified because their original training dataset was very small. As more matched data is added, the improvements of MTR (speed and volume perturbation in this case) should become less apparent, because the training data itself can include more conditions.

Our work focuses on having a single model instead of multiple MTR models each trained with different conditions. We apply different perturbation styles (additive noise, speed and volume) to clean training data in a controlled environment to analyse the effect of each method on DNN-based ASR using the LibriSpeech corpus (encoded using WAV49 encoding). The findings are applied to a proprietary South African call centre dataset that is WAV49 encoded.

3 Data

We use two datasets: our final aim is to use MTR to determine how much we can improve the performance of an ASR system on mismatched call centre data, but first experiment with individual styles in a controlled environment. An overview of the call centre dataset is given in Section 3.1; and in Section 3.2 we explain how the controlled environment is set up using the LibriSpeech corpus.

3.1 South African Call Centre Dataset

Call centres handle very large amounts of data on a daily basis. Typically, all calls are recorded and stored for future reference, legal purposes and call centre speech analytics. Due to the large number of calls, the recordings are often compressed for longer term storage. This can decrease the required storage space by up to twenty times. Although compression is beneficial for storage requirements, it is a challenging problem for ASR systems. We use a proprietary South African call centre dataset, referred to from here as the SACC corpus. All data in the SACC corpus consists of narrow-band single channel recordings. The corpus is mostly South African English, but there are occasional non-

English words from other official South African languages. Utterances with mostly non-English speech have been removed from the corpus. Table 1 shows the datasets in the SACC corpus after mostly non-English utterances were removed. There are 48.8 hours of training data that were originally not encoded, with an encoded version created using Sox¹. The training, development and test sets were created from the same set by dividing the corpus into three parts. A 1.2 hour held-out test set that was recorded and processed by the call centre at a later stage is also available for final testing. Calls are compressed in three steps, namely, lowering the sampling rate, combining dual channel audio to a single channel, and encoding the audio with WAV49 encoding.

3.1.1 Sampling Rate:

Most high quality ASR systems work with wide-band audio that is sampled at 16 kHz. All data in the SACC corpus is narrow-band (8 kHz). Narrow-band audio has less frequency information available than wide-band, which tends to hurt ASR systems slightly.

3.1.2 Channel Combination:

Telephone calls usually have two channels, one for the call center agent and one for the client (person who called or is being called). Two audio channels use twice as much storage space as a single channel, which is why they are combined to form only one channel. This creates three problems for an ASR system: (1) noise from both channels are present in the new signal, (2) overlapping speech and (3) speaker confusion. All of these factors contribute to decreased speech recognition performance.

3.1.3 WAV49 Encoding:

Compressing audio with a codec can reduce the storage space significantly, but also keep most of the original audio quality. There are many different compression methods, such as: Free Lossless Audio Codec (FLAC), MPEG Audio Layer III (MP3), Advanced Audio Coding (AAC), Ogg Vorbis, Speex and Opus [16]. The SACC corpus is WAV49-encoded: a full-rate GSM 06.10 codec with a compression ratio of 10:1 is applied to the audio file [17]. It is then saved in a WAV file format resulting in a WAV49-encoded file [18].

Table 1: SACC corpus subsets with sampling rate, encoding and total duration.

Dataset	Sampling rate	Encoding	Hours
train	8 kHz	-	48.8
train-e	8 kHz	WAV49	48.8
dev	8 kHz	-	7.1
dev-e	8 kHz	WAV49	7.1
test	8 kHz	-	6.2
test-e	8 kHz	WAV49	6.2
held-out test	8 kHz	WAV49	1.2

3.2 LibriSpeech Corpus with WAV49 Encoding

The LibriSpeech corpus contains 1 000 hours of English audiobook recordings sampled at 16 kHz [11]. It is a freely available public dataset that is used as a benchmark for many state-of-the-art ASR models [9, 19]. The dataset contains about 460 hours of clean training data, 500 hours of noisy training data, two development and two test sets (one clean and one noisy each). There are also four different language models included ranging from a small tri-gram to a large unpruned 4-gram language model.

We use the LibriSpeech corpus to create a controlled environment for MTR experiments. The corpus is well suited for this, since a large portion of the corpus has been labeled as “clean”, meaning that the recordings do not have much noise. Using these clean audio recordings, noise and encoding can easily be added to simulate call centre audio conditions.

We use the 100 hour subset of the LibriSpeech corpus for training data and the small tri-gram (tg-small) language model for faster decoding and making the comparison of different acoustic models more efficient. To simulate call centre conditions, we add background noise to the clean development (*dev-clean*) and test (*test-clean*) sets using the QUT-NOISE corpus [20] with a signal-to-noise ratio (SNR) of 5 dB. For our training data, we add noise using the

¹<http://sox.sourceforge.net>

Musan noise corpus [21] to create an artificial mismatch in noise conditions. An artificial mismatch is created between training and test data, because the noise corpus used to create the test set has different types of noise than the corpus used to perturb the training data. For both noise corpora, we randomly add a noise file to each utterance for the total duration of the utterance. We also encode these sets with WAV49 encoding and reduce their sampling rate to 8 kHz using Sox.

Table 2 shows the training datasets we use for MTR. Different combinations are used to see the combined performance impact on the development set. Speed and volume perturbation was applied to the training data with a change of 10% and 20% respectively. The speed was either increased or decreased with equal probability - approximately half of the utterances have a slower speed compared to the original set and the other half have a faster speed. Volume is handled in a similar way. Table 3 shows the development and test sets that are created from the *dev-clean* and *test-clean* subsets, in a similar manner as the training set but with fewer conditions.

Table 2: Multi-style training datasets created using the 100 hour clean LibriSpeech subset (*train-clean-100*).

Dataset name	Encoding	Noise corpus	SNR	Speed	Volume
train-clean	-	-	-	-	-
train-clean-8k	-	-	-	-	-
train-clean-e	WAV49	-	-	-	-
train-noisy-e-5	WAV49	QUT	5	-	-
train-clean-e-s	WAV49	-	-	10%	-
train-clean-e-v	WAV49	-	-	-	20%
train-clean-e-sv	WAV49	-	-	10%	20%
train-musan-e-5	WAV49	Musan	5	-	-
train-musan-e-10	WAV49	Musan	10	-	-
train-musan-e-15	WAV49	Musan	15	-	-
train-musan-e-20	WAV49	Musan	20	-	-
train-musan-e-15-s	WAV49	Musan	15	10%	-
train-musan-e-15-v	WAV49	Musan	15	-	20%
train-musan-e-15-sv	WAV49	Musan	15	10%	20%

Table 3: Development and test datasets created using the LibriSpeech *dev-clean* and *test-clean* sets.

Dataset name	Source dataset	Encoding	Noise corpus	SNR	Hours
dev-clean-e	dev-clean	WAV49	-	-	5.4
dev-noisy-e-5	dev-clean	WAV49	QUT	5	5.4
test-noisy-e-5	test-clean	WAV49	QUT	5	5.4

4 Experimental Setup

We use the Pytorch-Kaldi² ASR toolkit to train a context-dependant deep neural network hidden Markov model (CD-DNN-HMM) ASR system [22]. We use the toolkit’s default training setup with the standard scoring scripts for the LibriSpeech corpus. Everything in our setup is the same as theirs, except that we additionally optimise four selected hyperparameters (batch size, learning rate, language model weight and word insertion penalty).

We use the default MLP acoustic model for the LibriSpeech corpus. The acoustic model is a 5 hidden-layer network with 1 024 hidden units per layer. All hidden layers use rectified linear unit (ReLU) activation functions with batch normalisation and dropout with probability of 0.15. The output layer does not use batch normalisation or dropout and has a softmax activation function. We use feature-space maximum likelihood linear regression (fMLLR) input features with a temporal context window of 11 frames [23]. Our model trained on the *train-clean-100* subset of the LibriSpeech corpus achieved similar results to public baselines [22].

All networks are trained with the stochastic gradient descent (SGD) optimiser and a learning rate scheduler that halves the learning rate when the relative improvement³ on the development set is less than 0.001. The acoustic model is trained with the negative log-likelihood loss function to predict HMM state probabilities. The batch size and learning

²Available at: <https://github.com/mravanelli/pytorch-kaldi>

³Senone error rate is used to measure performance after each training epoch.

Table 4: WER results of models with different sampling rates on *dev-clean* and *dev-clean-e*. Average WER and standard error is shown over 3 seeds.

Train set	Sampling rate	dev-clean	dev-clean-e
Wide-band			
train-clean	16 kHz	8.88 ± 0.10	19.32 ± 0.11
train-clean-e	16 kHz ⁴	10.71 ± 0.05	11.02 ± 0.03
Narrow-band			
train-clean	8 kHz	10.29 ± 0.03	11.44 ± 0.04
train-clean-e	8 kHz	10.76 ± 0.02	11.21 ± 0.03

rate are optimised on the development set using a grid search. We found that these two hyperparameters have the largest effect on WER. The language model weight and word insertion penalty that gave the lowest WER on the development set are used for the final systems. All other hyperparameters are kept fixed. All networks are trained for 24 epochs; at this point all networks have converged. Each network is trained with three different random initialisation seeds. We report on the average WER and WER standard error across seeds.

5 Analysis

In this section, we first investigate the effect of sampling rate differences between training and testing data on the LibriSpeech corpus (Section 5.1). This is done to see how much of the performance is lost due to WAV49 encoding and how much is due to the difference in sampling rate. Then we create a controlled environment using the LibriSpeech corpus to isolate the effects that different styles in an MTR setup has on system performance (Section 5.2). Noise, speed and volume perturbation are applied to a clean training set to measure how much each of these techniques can improve the WER on the noise-perturbed test set. We also look at how larger networks can benefit MTR setups (Section 5.3). Finally, the findings in the controlled experiment are applied to the SACC corpus (Section 5.4).

5.1 Sampling Rate Differences on the LibriSpeech Corpus

Two sets of networks are trained on the *train-clean* and *train-clean-e* datasets using 16 kHz and 8 kHz audio, respectively. Encoded data, which was previously downsampled during the encoding stage, was now upsampled to 16 kHz using Sox to train a 16 kHz model. We also downsampled the clean training and development sets to 8 kHz to measure the performance difference caused by WAV49 encoding when unencoded narrow-band audio is used for training.

The WER results are shown in Table 4. All data in the top section is used at 16 kHz (training and development sets); while the data in the bottom section is used at 8 kHz. The *train-clean* model performed the best on the *dev-clean* set for both frequencies. Evaluating the clean 16 kHz model (*train-clean*, 16 kHz) on the encoded development set drastically increased the WER to 19.32%. By downsampling the clean training data (*train-clean*, 8 kHz), the result improved by 40.1% relative WER to 11.44%. This network, trained only using unencoded data, performed only slightly worse than the *train-clean-e* model (11.44% vs 11.21% WER). The large difference in WER can be reduced significantly by downsampling unencoded training data. Encoding the training data only gave a slight improvement over the 8 kHz *train-clean* model of 2.0% relative WER.

We observe that most of the mismatch is a result of the difference in sampling rate and not due to encoding. The improvement achieved when downsampling unencoded training data was 40.1% relative WER and only 2.0% when encoding the training set.

5.2 Multi-style Training on the LibriSpeech Corpus

In this experiment, we analyse the effect of different MTR styles on a set with mismatched noise conditions. Different combinations of noise, speed and volume perturbation are used for training data. Table 5 shows the WER results on the development set (*dev-noisy-e-5*). Information about the training datasets used by each model is shown in Table 2. All data, except for the 8 kHz *train-clean-8k* set is upsampled to 16 kHz using Sox. The upsampling process used does not attempt to interpolate values in order to add high frequency information that was lost during downsampling. Only the existing lower frequency components are retained.

⁴Up-sampled from 8 kHz.

⁵Training and test data is used at 8 kHz.

Table 5: WER on development set (*dev-noisy-e-5*) using training datasets with different styles. Average WER and standard error is shown over 3 seeds.

Model	Datasets	Size	Dev WER
Variations of clean set			
train-clean	train-clean	1	36.46 \pm 0.14
train-clean-8k ⁵	train-clean @ 8 kHz	1	33.23 \pm 0.21
train-clean-e	train-clean-e	1	28.06 \pm 0.03
Speed and volume			
train-clean-e-s	train-clean-e + s	2	27.83 \pm 0.02
train-clean-e-v	train-clean-e + v	2	28.21 \pm 0.07
train-clean-e-sv	train-clean-e + sv	2	28.25 \pm 0.10
train-clean-e-s-v	train-clean-e + s + v	3	28.04 \pm 0.13
Noise			
train-musan-e-5	train-musan-e-5	1	29.29 \pm 0.34
train-musan-e-10	train-musan-e-10	1	27.29 \pm 0.12
train-musan-e-15	train-musan-e-15	1	23.30 \pm 0.06
train-musan-e-20	train-musan-e-20	1	26.64 \pm 0.09
Speed, volume and noise			
train-musan-e-15-s	train-musan-e-15 + s	2	24.09 \pm 0.05
train-musan-e-15-v	train-musan-e-15 + v	2	23.97 \pm 0.11
train-musan-e-15-sv	train-musan-e-15 + sv	2	24.34 \pm 0.02
train-musan-e-15-s-v	train-musan-e-15 + s + v	3	23.80 \pm 0.09
train-musan-e-15-s-v-sv	train-musan-e-15 + s + v + sv	4	23.89 \pm 0.08
Matched noise			
train-noisy-e-5	train-noisy-e-5	1	19.75 \pm 0.04

Similar to the results in Section 5.1, downsampling the clean training data improved the WER, although the improvement is much less than before. Adding WAV49 encoding to the training data improved the relative WER of the *train-clean-8k* model by 15.6%, much more than observed in Section 5.1.

Speed and volume perturbation were applied to the clean encoded training data. Different combinations of datasets were evaluated. A small improvement in WER (0.8% relative) was observed when using only speed perturbation. None of the other combinations resulted in a notable improvement; the *train-clean-e-v* and *train-clean-e-sv* networks performed worse than without perturbation. This may be because the training data already captures a large range of speed and volumes, or that the development set does not vary much in terms of speed and volume.

Noise perturbation was applied to clean training data using four different SNR values and the sets were encoded afterwards. The performance with the 15 dB network was much better than the rest (12.5% to 20.4% relative WER), despite the fact that the development set used an SNR of 5 dB. The different noise corpora, QUT-NOISE vs Musan, can explain the difference. Energy in the noise files are distributed differently, so the SNR values are not directly comparable. The *train-musan-e-15* model performed 17.0% relatively better than the *train-clean-e* model. This emphasises how important matched training and test conditions are. It is very important to use the correct SNR for noise perturbation, because it has a large influence on system performance.

We used the best noise-perturbed training dataset and added speed and volume perturbation. The WER when using speed and volume perturbation in any combination did not improve the result compared to using only additive noise. A similar phenomenon was observed in [5] on the ASPIRE corpus, where they only observed an absolute WER improvement of 0.1%. We further investigate this in Section 5.3.

Finally, we trained a network using a noise-matched training dataset, *train-noisy-e-5* that also uses the QUT-NOISE corpus. The WER for this model is 17.0% relatively better than the best MTR model. This shows that MTR has many shortcomings when test conditions are significantly different from the training set. When encountering unseen environments on a new test set, most systems will probably struggle to do well.

Up to this point, all results were reported on the development set. Table 6 shows the WERs on the test set (*test-noisy-e-5*) using six selected models that performed the best in each category on the development set. The results on the test set are very similar to those on the development set. The best MTR model of all the combinations tested, is the one that used only noise perturbation using the SNR value that performed the best on the development set. The difference in WER between the best MTR model and the *train-clean* model is 33.5% relative. MTR can clearly reduce the WER

Table 6: WER on test set (*test-noisy-e-5*) using training datasets with different styles. Average WER and standard error is shown over 3 seeds.

Model	Datasets	Size	Test WER
Variations of clean set			
train-clean	train-clean	1	36.92 \pm 0.18
train-clean-e	train-clean-e	1	29.16 \pm 0.12
Speed and volume			
train-clean-e-s	train-clean-e + s	3	28.61 \pm 0.11
Noise			
train-musan-e-15	train-musan-e-15	1	24.54 \pm 0.22
Speed, volume and noise			
train-musan-e-15-s-v	train-musan-e-15 + s + v	3	24.87 \pm 0.08
Matched noise			
train-noisy-e-5	train-noisy-e-5	1	20.48 \pm 0.03

Table 7: WER on development (*dev-noisy-e-5*) and test set (*test-noise-e-5*) using MLP acoustic models with 2 048 hidden units per layer. Average WER and standard error is shown over 3 seeds.

Model	Size	Dev WER	Test WER
Encoded			
train-clean-e (1 024x5)	1	28.06 \pm 0.03	29.16 \pm 0.12
train-clean-e (2 048x5)	1	26.82 \pm 0.08	27.74 \pm 0.14
Noise			
train-musan-e-15 (1 024x5)	1	23.30 \pm 0.06	24.54 \pm 0.22
train-musan-e-15 (2 048x5)	1	23.15 \pm 0.10	24.55 \pm 0.04
Speed, volume and noise			
train-musan-e-15-s-v (1 024x5)	3	23.80 \pm 0.09	24.87 \pm 0.08
train-musan-e-15-s-v (2 048x5)	3	22.81 \pm 0.06	24.34 \pm 0.08

significantly if the conditions are properly chosen, but MTR still performed 16.5% worse than the *train-noisy-e-5* model with matched conditions.

5.3 Multi-style Training Using Larger Networks on the LibriSpeech Corpus

By adding speed and volume perturbation, you also add more training data. It is possible that the network with only 1 024 hidden units is too small to capture the larger data distribution. Increasing the network capacity should help the models using more training datasets generalise better and possibly give an advantage to the speed and volume perturbation networks.

Using the same training and optimisation protocol described in Section 4, we train three networks with 2 048 hidden units per layer instead of 1 024. The average WERs are shown in Table 7 over three seeds for both network sizes; the dimensions of the hidden layers are shown in brackets. The performance of all models improved, but the model using speed and volume perturbation improved more than the model only using noise perturbation. The larger capacity benefits the model with more training data, but also the model using only encoded training data. The difference between the small and large network for the *train-musan-e-15* model is almost negligible.

This experiment confirmed the hypothesis that the MTR models required more capacity to outperform the noise-perturbed network. There is however an increased computational cost when doubling the number of hidden units on top of the three times more training data. This becomes an important trade-off to consider if computational resources are limited.

5.4 Multi-style Training on the SACC Corpus

We now evaluate our findings on the SACC corpus described in Section 3.1. A baseline DNN acoustic model is trained using fMLLR features and 8 kHz unencoded training data using the protocol described in Section 4. Another network is trained using only a single set of WAV49-encoded training data. For MTR, we use normal and encoded training data, and apply speed perturbation to the encoded training data only. We did not include volume perturbation, because the experiment on the LibriSpeech corpus did not show a consistent improvement when using it. The unencoded training data is included, because we want to jointly perform well on both encoded and unencoded testing data. We did not

Table 8: WER results on dev/test sets for the SACC corpus. Average WER is shown over 3 seeds.

Model	dev	dev-e	test	test-e	held-out test
train	28.41	28.91	33.14	33.43	41.90
train-e	28.40	28.63	33.36	33.04	41.80
MTR	27.98	28.19	32.77	32.46	41.42

add any noise, because the training data was already noisy and came from the same call centre as our development and test sets. The results on the development and test sets are shown in Table 8.

The model that used only the encoded training set performed better on encoded test sets, but similar to the baseline model on the *dev* set and worse on the *test* set. The MTR model performed the best across all five datasets with relative WER improvements of between 1.1% and 2.9%.

Since the training data is well matched with the testing data, MTR does not provide large improvements. It does not hurt performance, and small consistent improvements are possible, but the real advantage of MTR is only observed if there is a significant mismatch.

6 Conclusion

MTR can have a very positive effect on ASR performance, given that the styles are chosen appropriately. Speed and volume perturbation can slightly reduce WER in some scenarios, but are computationally much more expensive. When using a system on narrow-band test sets, training with narrow-band audio is absolutely necessary.

The two styles that gave the best improvement on the WAV49 encoded LibriSpeech corpus was: (1) encoding training data (2) and noise perturbation if there is no noise in the training set. However, if the SNR of the added noise is completely different to the test conditions, it can hurt the system. The best MTR setup outperformed the clean baseline on the test set by 33.5% relative WER, but still performed worse than the noise-matched model by 16.5%.

Only two MTR styles were used on the SACC corpus (encoding and speed perturbation), since the recordings already contained noise and were narrow-band. The relative WER improvement on the test sets were limited, ranging from 1.1% to 2.9% when using MTR. The improvements of MTR are small on this corpus, which we attribute to the training and test data being well matched.

With proper network capacity, MTR does not hurt system performance, even when the data is very well matched. Consistent small improvements are observed in matched datasets, with very large improvements achieved on mismatched datasets.

Acknowledgement

The authors acknowledge the Centre for High Performance Computing (CHPC), South Africa, for providing computational resources used in this research.

References

- [1] James Baker. The DRAGON system - An overview. *1975 IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29, 1975.
- [2] Frederick Jelinek, Lalit Bahl, and Robert Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. *1975 IEEE Transactions on Information Theory*, 21(3):250–256, 1975.
- [3] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE access*, 7:19143–19165, 2019.
- [4] Xugang Lu, Sheng Li, and Masakiyo Fujimoto. Automatic speech recognition. In *Speech-to-Speech Translation*, pages 21–38. Springer, 2020.
- [5] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *Proceedings of INTERSPEECH*, pages 3586–3589. International Speech Communication Association, 2015.

- [6] Richard Lippmann, Edward Martin, and D Paul. Multi-style training for robust isolated-word speech recognition. In *1987 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 12, pages 705–708. IEEE, 1987.
- [7] Jinyu Li, Dong Yu, Jui-Ting Huang, and Yifan Gong. Improving wideband speech recognition using mixed-bandwidth training data in CD-DNN-HMM. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 131–136. IEEE, 2012.
- [8] Mortaza Doulaty, Richard Rose, and Olivier Siohan. Automatic optimization of data perturbation distributions for multi-style training in speech recognition. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 21–27. IEEE, 2016.
- [9] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In *Proceedings of INTERSPEECH*, pages 2613–2617. International Speech Communication Association, 2019.
- [10] Igor Szöke, Miroslav Skácel, Ladislav Mošner, Jakub Paliesek, and Jan Honza Černocký. Building and evaluation of a real room impulse response dataset. *2019 IEEE Journal of Selected Topics in Signal Processing*, 13(4):863–876, 2019.
- [11] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5206–5210. IEEE, 2015.
- [12] Ramazan Gokay and Hulya Yalcin. Improving low resource Turkish speech recognition with data augmentation and TTS. In *International Multi-Conference on Systems, Signals & Devices (SSD)*, pages 357–360. IEEE, 2019.
- [13] John J Godfrey, Edward C Holliman, and Jane McDaniel. SWITCHBOARD: Telephone speech corpus for research and development. In *1992 IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 517–520. IEEE, 1992.
- [14] Mary Harper. The automatic speech recognition in reverberant environments (aspire) challenge. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 547–554. IEEE, 2015.
- [15] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in English and Mandarin. In *International Conference on Machine Learning*, pages 173–182. PMLR, 2016.
- [16] Ingo Siegert, Alicia Flores Lotz, Linh Linda Duong, and Andreas Wendemuth. Measuring the impact of audio compression on the spectral quality of speech data. *Studientexte zur Sprachkommunikation: Elektronische Sprachsignalverarbeitung 2016*, pages 229–236, 2016.
- [17] EN ETSI. 300 961 v7. 0.2 (1999). *Digital cellular telecommunications system (Phase 2+); Full rate speech; Transcoding (GSM 06.10 version 7.0. 2 Release 1998)*, 1999.
- [18] Jim Van Meggelen, Russell Bryant, and Leif Madsen. *Asterisk: The Definitive Guide: Open Source Telephony for the Enterprise*. O’Reilly Media, 2019.
- [19] Qiantong Xu, Alexei Baevski, Tatiana Likhomanenko, Paden Tomasello, Alexis Conneau, Ronan Collobert, Gabriel Synnaeve, and Michael Auli. Self-training and pre-training are complementary for speech recognition. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3030–3034. IEEE, 2021.
- [20] David Dean, Sridha Sridharan, Robert Vogt, and Michael Mason. The QUT-NOISE-TIMIT corpus for evaluation of voice activity detection algorithms. In *Proceedings of INTERSPEECH*, pages 3110–3113. International Speech Communication Association, 2010.
- [21] David Snyder, Guoguo Chen, and Daniel Povey. MUSAN: A Music, Speech, and Noise Corpus, 2015. arXiv:1510.08484v1.
- [22] Mirco Ravanelli, Titouan Parcollet, and Yoshua Bengio. The Pytorch-Kaldi speech recognition toolkit. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6465–6469. IEEE, 2019.
- [23] Mark JF Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998.