Raw Multichannel Processing using Deep Neural Networks

Tara N. Sainath, Ron J. Weiss, Kevin W. Wilson, Arun Narayanan, Michiel Bacchiani, Bo Li, Ehsan Variani, Izhak Shafran, Andrew Senior, Kean Chin, Ananya Misra and Chanwoo Kim

Abstract Multichannel ASR systems commonly separate speech enhancement, including localization, beamforming and postfiltering, from acoustic modeling. In this chapter, we perform multichannel enhancement jointly with acoustic modeling in a deep neural network framework. Inspired by beamforming, which leverages differences in the fine time structure of the signal at different microphones to filter energy arriving from different directions, we explore modeling the raw time-domain waveform directly. We introduce a neural network architecture which performs multichannel filtering in the first layer of the network and show that this network learns to be robust to varying target speaker direction of arrival, performing as well as a model that is given oracle knowledge of the true target speaker direction. Next, we show how performance can be improved by *factoring* the first layer to separate the multichannel spatial filtering operation from a single channel filterbank which computes a frequency decomposition. We also introduce an adaptive variant, which updates the spatial filter coefficients at each time frame based on the previous inputs. Finally we demonstrate that these approaches can be implemented more efficiently in the frequency domain. Overall, we find that such multichannel neural networks give a relative word error rate improvement of more than 5% compared to a traditional beamforming-based multichannel ASR system and more than 10% compared to a single channel waveform model.

1 Introduction

While state-of-the-art automatic speech recognition (ASR) systems perform reasonably well in close-talking microphone conditions, performance degrades in conditions when the microphone is far from the user. In such farfield cases, the speech signal is degraded by reverberation and additive noise. To improve recognition in such cases, ASR systems often use signals from multiple microphones to enhance the speech signal and reduce the impact of reverberation and noise [2, 6, 10].

Multichannel ASR systems often use separate modules to perform recognition. First, microphone array speech enhancement is applied, typically broken into localization, beamforming and postfiltering stages. The resulting single channel enhanced signal is passed to an conventional acoustic model [15, 35]. A commonly used enhancement technique is filter-and-sum beamforming [2], which begins by aligning signals from different microphones in time (via localization) to adjust for the propagation delay from the target speaker to each microphone. The time-aligned signals are then passed through a filter for each microphone and summed to enhance the signal from the target direction and to attenuate noise coming from other directions. Commonly used filter design criteria are based on Minimum Variance Distortionless Response (MVDR) [10, 39] or multichannel Wiener filtering (MWF) [6].

When the end goal is to improve ASR performance, tuning the enhancement model independently from the acoustic model might not be optimal. To address this issue [34] proposed likelihood-maximizing beamforming (LIMABEAM) which optimizes beamformer parameters jointly with those of the acoustic model. This technique was shown to outperform conventional techniques such as delay-and-sum beamforming (i.e. filter-and-sum where the filters consist of impulses). Like most enhancement techniques, LIMABEAM is a model-based scheme and requires an iterative algorithm that alternates between acoustic model inference and enhancement model parameter optimization. Contemporary acoustic models are generally based on neural networks, optimized using a gradient learning algorithm. Combining model-based enhancement with an acoustic model that uses gradient learning can lead to considerable complexity, e.g. [17].

In this chapter we extend the idea of performing beamforming jointly with acoustic modeling from [34], but do this within the context of a deep neural network (DNN) framework by training an acoustic model directly on the raw signal. DNNs are attractive because they have been shown to be able to perform feature extraction jointly with classification [23]. Previous work has demonstrated the possibility of training deep networks directly on raw, single channel, time domain waveform samples [11, 18, 19, 24, 32, 37]. The goal of this chapter is to explore a variety of different joint enhancement/acoustic modeling DNN architectures that operate on multichannel signals. We will show that jointly optimizing both stages is more effective than techniques which cascade independently tuned enhancement algorithms with acoustic models.

Since beamforming takes advantage of the fine time structure of the signal at different microphones, we begin by modeling the raw time-domain waveform directly. In this model, introduced in [18, 31], the first layer consists of multiple time convolution filters, which map the multiple microphone signals down to a single time-frequency representation. As we will show, this layer learns bandpass filters which are spatially selective, often learning several filters with nearly identical frequency response, but with nulls steered toward different directions of arrival. The output of this spectral filtering layer is passed to an acoustic model, such as a convolutional long short-term memory, deep neural network (CLDNN) acoustic model [29]. All layers of the network are trained jointly.

As described above, it is common for multichannel speech recognition systems to perform spatial filtering independently from single channel feature extraction. With this in mind, we next investigate explicitly factorizing these two operations to be separate neural network layers. The first layer in this "factored" raw waveform model consists of short-duration multichannel time convolution filters which map multichannel inputs down to a single channel, with the idea that the network might learn to perform broadband spatial filtering in this layer. By learning several filters in this "spatial filtering layer", we hypothesize that the network can learn filters for multiple different spatial look directions. The single channel waveform output of each spatial filter is passed to a longer-duration time convolution "spectral filtering layer" intended to perform finer frequency resolution spectral decomposition analogous to a time-domain auditory filterbank as in [32]. The output of this spectral filtering layer is also passed to an acoustic model.

One issue with the two architectures above is that once weights are learned during training, they remain fixed for each test utterance. In contrast, some beamforming techniques, such as the generalized sidelobe canceller [14], update weights adaptively within each utterance. We explore an adaptive neural net architecture, where an LSTM is used to predict spatial filter coefficients that are updated at each frame. These filters are used to filter and sum the multichannel input, replacing the "spatial filtering layer" of the factored model described above, before passing the enhanced single channel output to a waveform acoustic model.

Finally, since convolution between two time domain signals is equivalent to the element-wise product of their frequency domain counterparts, we investigate speeding up the raw waveform neural network architectures described above by consuming the complex-valued fast Fourier transform of the raw input and implementing filters in the frequency domain.

2 Experimental Details

2.1 Data

We conduct experiments on a dataset comprised of about 2,000 hours of noisy training data consisting of 3 million English utterances. This data set is created by artificially corrupting clean utterances using a room simulator to add varying degrees of noise and reverberation. The clean utterances are anonymized and hand-transcribed voice search queries, and are representative of Google's voice search traffic. Noise signals, which include music and ambient noise sampled from YouTube and recordings of "daily life" environments, are added to the clean utterances at SNRs ranging from 0 to 20 dB, with an average of about 12 dB. Reverberation is simulated using the image method [1] – room dimensions and microphone array positions are randomly sampled from 100 possible room configurations with T_{60}s ranging from 400 to 900 ms, with an average of about 600 ms. The simulation uses an 8-channel uni-

form linear microphone array, with inter-microphone spacing of 2 cm. Both noise source location and target speaker locations change between utterances; the distance between the sound source and the microphone array varies between 1 to 4 meters.

The primary evaluation set consists of a separate set of about 30,000 utterances (over 20 hours), and is created by simulating similar SNR and reverberation settings to the training set. Care was taken to ensure that the room configurations, SNR values, T_{60} times, and target speaker and noise positions in the evaluation set differ from those in the training set. The microphone array geometry between the training and simulated test sets is identical.

We obtained a second "rerecorded" test set by playing the evaluation set and the noises separately using a mouth simulator and a speaker, respectively, in a living room setting. The signals are recorded using a 7-channel circular microphony array with a radius of 3.75 cm. Assuming an x-axis that passes through two diagonally opposite mics along the circumference of the array, the angle of arrival of the target speaker ranges from 0 to 180 degrees. Noise originates from locations different from the target speaker. The distance of the sources to the target ranges from 1 to 6 meters. To create noisy rerecorded eval sets, the rerecorded speech and noise signals are mixed artificially after scaling noise to obtain SNRs ranging from 0 to 20 dB. The distribution of the SNR matches the distribution used to generate the simulated evaluation set. We create 4 versions of the rerecorded sets to measure generalization performance of our models. The first two have rerecorded speech w/o any added noise. The mic array is placed at the center of the room and closer to the wall, respectively, to capture reasonably different reverberation characteristics. The remaning two subsets correspond to the noisy versions of these sets.

2.2 Baseline Acoustic Model

We compare the models proposed in this chapter to a baseline CLDNN acoustic model trained using log-mel features [29] computed with a 25ms window and a 10ms hop. Single channel models are trained using signals from channel 1, $C = 2$ channel models use channels 1 and 8 (14 cm spacing), $C = 4$ channel models use channels 1, 3, 6, and 8 (14 cm array span, with a microphone spacing of 4cm-6cm-4cm).

The baseline CLDNN architecture is shown in the `CLDNN` module of Figure 1. First, the `fConv` layer performs convolution across the frequency dimension of the input log-mel time-frequency feature to gain some invariance to pitch and vocal tract length. The architecture used for this convolutional layer is similar to that proposed in [25]. Specifically, a single convolutional layer with 256 filters of size 1×8 in time-frequency is used. Our pooling strategy is to use non-overlapping max pooling along the frequency axis, with a pooling size of 3. The pooled output is given to a 256-dimensional linear low-rank layer.

The output of frequency convolution is passed to a stack of `LSTM` layers, which model the signal across long time scales. We use 3 LSTM layers, each comprised

of 832 cells, and a 512 unit projection layer for dimensionality reduction following [33]. Finally, we pass the final LSTM output to a single fully connected DNN layer comprised of 1,024 hidden units. Due to the high dimensionality of the 13,522 context-dependent state output targets used by the language model, a 512-dimensional linear output low rank projection layer is used prior to the softmax layer to reduce the number of parameters in the overall model [26]. Some experiments in the chapter do not use the frequency convolution layer, and we will refer to such acoustic models as LDNNs.

During training, the CLDNN is unrolled for 20 time steps and trained using truncated backpropagation through time (BPTT). In addition, the output state label is delayed by 5 frames, as we have observed that information about future frames helps to better predict the current frame [29].

Unless otherwise indicated, all neural networks are trained using asynchronous stochastic gradient descent (ASGD) optimization [9] to optimize a cross-entropy (CE) criterion. Additional sequence training experiments also use distributed ASGD [16]. All networks have 13,522 context-dependent (CD) output targets. The weights for all CNN and DNN layers are initialized using the Glorot-Bengio strategy [13], while those of all LSTM layers are randomly initialized using a uniform distribution between -0.02 and 0.02. We use an exponentially decaying learning rate, initialized to 0.004 and decaying by 0.1 over 15 billion frames.

3 Multichannel Raw Waveform Neural Network

3.1 Motivation

The proposed multichannel raw waveform CLDNN is related to filter-and-sum beamforming, a generalization of delay-and-sum beamforming which filters the signal from each microphone using a finite impulse response (FIR) filter before summing them. Using similar notation to [34], filter-and-sum enhancement can be written as follows:

$$y[t] = \sum_{c=0}^{C-1} \sum_{n=0}^{N-1} h_c[n] x_c[t-n-\tau_c] \tag{1}$$

where $h_c[n]$ is the nth tap of the filter associated with microphone c, $x_c[t]$, is the signal received by microphone c at time t, τ_c is the steering time difference of arrival induced in the signal received by microphone c used to align it to the other array channels, and $y[t]$ is the output signal. C is the number of microphones in the array and N is the number of FIR filter taps.

3.2 Multichannel filtering in the time domain

Enhancement algorithms implementing Equation 1 generally depend on an estimate of the steering delay τ_c obtained using a separate localization model, and they compute filter parameters $h_c[n]$ by optimizing an objective such as MVDR. In contrast, our aim is to allow the network to jointly estimate steering delays and filter parameters by optimizing an acoustic modeling classification objective. The model captures different steering delays using a bank of P multichannel filters. The output of filter $p \in \{0,\ldots,P-1\}$ can be written as follows:

$$y^p[t] = \sum_{c=0}^{C-1}\sum_{n=0}^{N-1} h_c^p[n]x_c[t-n] = \sum_{c=0}^{C-1} x_c[t] * h_c^p \tag{2}$$

where the steering delays are implicitly absorbed into the filter parameters $h_c^p[n]$. In this equation, '*' denotes the convolution operation

The first layer in our raw waveform architecture implements Equation 2 as a multichannel convolution (in time) with a FIR spatial filterbank $h_c = \{h_c^1, h_c^2, \ldots h_c^P\}$ where $h_c \in \Re^{N\times P}$ for $c \in \{0,\cdots,C-1\}$. Each filter h_c^p is convolved with the corresponding input channel x_c, and the overall output for filter p is computed by summing the result of this convolution across all channels $c \in \{0,\cdots,C-1\}$. The operation within each filter is equivalent to an FIR filter-and-sum beamformer, except that it does not explicitly shift the signal in each channel by an estimated time difference of arrival. As we will show, the network learns the steering delay and filter parameters implicitly.

The output signal remains at the same sampling rate as the input signal, which contains more information than is typically relevant for acoustic modeling. In order to produce an output that is invariant to perceptually and semantically identical sounds appearing at different time shifts we pool the outputs in time after filtering [18, 32], in an operation that has an effect similar to discarding the phase in the short-time Fourier transform. Specifically, the output of the filterbank is max-pooled across time to give a degree of short term shift invariance, and then passed through a compressive non-linearity.

As shown in [18, 32], single channel time convolution layers similar to the one described above implement a conventional time-domain filterbank. Such layers are capable of implementing, for example, a standard gammatone filterbank, which consists of a bank of time-domain filters followed by rectification and averaging over a small window. Given sufficiently large P, the corresponding multichannel layer can (and as we will show, does in fact) similarly implement a frequency decomposition in addition to spatial filtering. We will therefore subsequently refer to the output of this layer as a "time-frequency" feature representation.

A schematic of the multichannel time convolution layer is shown in the `tConv` block of Figure 1. First, we take a small window of the raw waveform of length M samples for each channel C, denoted as $\{x_0[t], x_1[t], \ldots, x_{C-1}[t]\}$ for $t \in 1,\cdots,M$. The signal from each channel x_c is convolved with a bank of P filters with N taps $h_c = \{h_c^1, h_c^2, \ldots, h_c^P\}$. When the convolution is strided by 1 in time across M sam-

ples, the output from the convolution in each channel is $y_c[t] \in \Re^{(M-N+1)\times P}$. After summing $y_c[t]$ across channels c, we max pool the filterbank output in time (thereby discarding short term phase information), over the entire time length of the output signal $M-N+1$, to produce $y[t] \in \Re^{1\times P}$. Finally, we apply a rectified non-linearity, followed by a stabilized logarithm compression[1], to produce $z[l]$, a P dimensional frame-level feature vector at frame l. We then shift the window around the waveform by 10ms and repeat this time convolution, producing a sequence of feature frames at 10ms intervals.

To match the time-scale of the log-mel features, the raw waveform features are computed with an identical filter size of 25ms, or $N = 400$ at a sampling rate of 16kHz. The input window size is 35ms ($M = 560$) giving a 10ms fully overlapping pooling window. Our experiments explore varying the number of time-convolutional filters P.

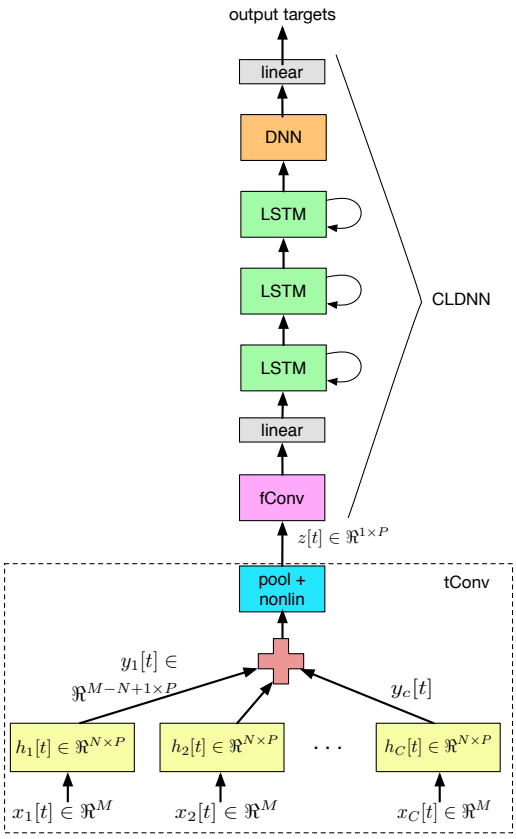


Fig. 1: Multichannel raw waveform CLDNN architecture.

[1] We use a small additive offset to truncate the output range and avoid numerical instability with very small inputs: $\log(\cdot + 0.01)$.

As shown in the CLDNN block of Figure 1, the output of the time convolutional layer (`tConv`) produces a frame-level feature, denoted as $z[l] \in \Re^{1\times P}$. This feature is then passed to a CLDNN model [29] described in Section 2, which predicts context dependent state output targets.

3.3 Filterbank spatial diversity

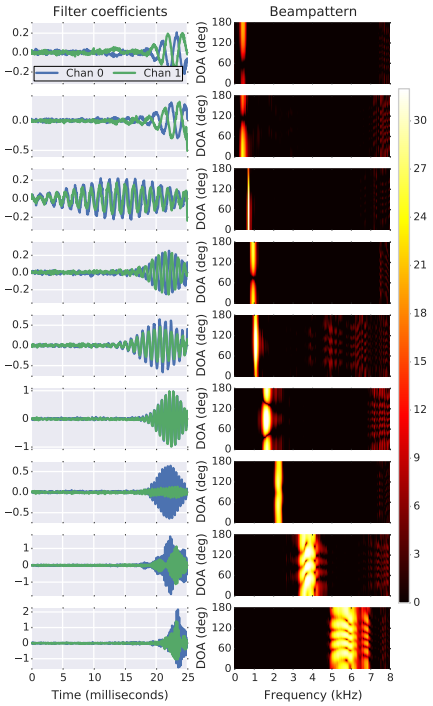


Fig. 2: Example filter coefficients and corresponding spatial response beampatterns learned in a network with 128 `tConv` filters trained on 2 channel inputs. Some filters learned by this network have nearly-identical center frequencies but different spatial responses. For example, the top two example filters both have center frequencies of about 440Hz, but the first filter has a null at a direction of arrival of about 60 degrees, while the second has a null at about 120 degrees. The corresponding phase difference between the two channels of each filter is visible in the time domain filter coefficients plotted on the left.

Figure 2 plots example multichannel filter coefficients and their corresponding spatial responses, or beampatterns, after training for `tConv`. The beampatterns show the magnitude response in dB as a function of frequency and direction of

arrival, i.e. each horizontal slice of the beampattern corresponds to the filter's magnitude response for a signal coming from a particular direction, and each vertical slice corresponds to the filter's response across all spatial directions in a particular frequency band. Lighter shades indicate regions of the frequency-directions space which are passed through the filter, while darker shades indicate regions which are filtered out. Within a given beampattern, we refer to the frequency band containing the maximum overall response as the filter's *center frequency* (since the filters are primarily bandpass in frequency), and the direction corresponding to the minimum response in that frequency as the filter's *null direction*.

The network tends to learn filter coefficients with very similar shapes in each channel except they are slightly shifted relative to each other, consistent with the notion of a steering delay τ_c described in Section 3. Most filters have a bandpass response in frequency, with bandwidths that increase with center frequency, and many are steered to have stronger response for signals arriving from a particular direction. Approximately two-thirds of the filters in the model shown in Figure 2 demonstrate a significant spatial response, i.e. show a difference of at least 6dB between the direction with the minimum and maximum response at the filter center frequency. Such strong spatial responses are clearly visible in the null near 120 degrees in the second filter, and a similar null near 60 degrees in the fourth filter shown in Figure 2.

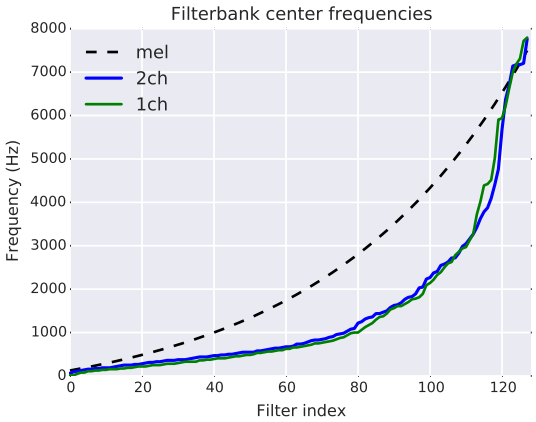


Fig. 3: Comparison of the peak response frequencies of waveform CLDNN filterbanks trained on one- and two-channel inputs to the standard mel frequency scale.

Figure 3 plots the peak response frequencies of filterbanks from networks trained on a one- and two-channel networks of the form shown in 2. The two networks converge to similar frequency scales, both consistently allocating many more filters to low frequencies compared to the mel scale. For example, the learned filterbanks have roughly 80 filters with peak responses below 1000Hz, while a 128-band mel scale has only 40 bands with center frequencies below 1000Hz. The network also

learns subsets of filters with the same overall shape and frequency response but tuned to have nulls in different directions, as illustrated by the top two example filters in Figure 2. Such diversity in spatial response gives upstream layers information that can be used to discriminate between signals arriving from different directions.

Because each filter has a fixed directional response, the ability of the network to exploit directional cues is constrained by the number of filters it uses. By increasing the number of filters, we can potentially improve the spatial diversity of the learned filters and therefore allow the network to better exploit directional cues. Table 1 demonstrates the effect of increasing the number of filters on overall word error rate (WER). Improvements saturate at 128 filters for networks trained on 2 channel inputs, while 4 and 8 channels networks continue to improve with 256 filters. With additional input channels the `tConv` filters are able to learn more complex spatial responses (even though the total array span is unchanged), enabling the network to make use of additional filterbank capacity to improve performance.

Filters	2ch (14cm)	4ch (4-6-4cm)	8ch (2cm)
128	21.8	21.3	21.1
256	21.7	20.8	20.6
512	-	20.8	20.6

Table 1: WER for raw waveform multichannel CLDNNs with different number of input channels. The inter-microphone spacing is given in parentheses.

3.4 Comparison to log-mel

We train baseline multichannel log-mel CLDNNs by computing log-mel features for each channel, and treating these as separate feature maps into the CLDNN. Since the raw waveform model improves as we increase the number of filters, we perform the same experiment for log-mel. It should be noted that concatenating magnitude-based features (i.e., log-mel) from different channels into a neural network has been shown to give improvements over single channel [36, 22].

Table 2 shows that for log-mel, neither increasing the number of filters (frequency bands) nor increasing the number of microphone channels has a strong effect on word error rate. Since log-mel features are computed from the FFT magnitude, the fine time structure (stored in the phase), and therefore information about inter-microphone delays, is discarded. Log-mel models can therefore only make use of the weaker inter-microphone level difference cues. However, the multichannel time-domain filterbanks in the raw waveform models utilize the fine time structure and show larger improvements as the number of filters increase.

Comparing Tables 1 and 2 we can see that raw waveform models consistently outperform log-mel, particularly for larger number of channels where more spatial diversity is possible.

Filters	2ch (14cm)	4ch (4-6-4cm)	8ch (2cm)
128	22.0	21.7	22.0
256	21.8	21.6	21.7

Table 2: WER for log-mel multichannel CLDNNs.

3.5 Comparison to oracle knowledge of speech TDOA

Note that the models presented in the previous subsection do not explicitly estimate the time difference of arrival of the target source arriving at different microphones, which is commonly done in beamforming [2]. Time difference of arrival (TDOA) estimation is useful because time aligning and combining signals steers the array such that the target speech signal is enhanced relative to noise sources coming from other directions.

In this section, we analyze the behavior of raw waveform CLDNNs when the signals are time aligned using the true TDOA calculated using the room geometry. For the delay-and-sum (D+S) approach, we shift the signal in each channel by the corresponding TDOA, average them together, and pass the result into a 1-channel raw waveform CLDNN. For the time-aligned multichannel (TAM) approach, we align the signals in time and pass them as separate channel inputs to a multichannel raw waveform CLDNN. Thus the difference between the multichannel raw waveform CLDNNs described in Section 2 and TAM is solely in how the data is presented to the network (whether or not they are first explicitly aligned to "steer" toward the target speaker direction); the network architectures are identical.

Feature	1ch	2ch (14cm)	4ch (4-6-4cm)	8ch (2cm)
oracle D+S	23.5	22.8	22.5	22.4
oracle TAM	23.5	21.7	21.3	21.3
raw, no tdoa	23.5	21.8	21.3	21.1

Table 3: WER with oracle knowledge of the true target TDOA. All models use 128 filters.

Table 3 compares the WER of D+S, TAM, and raw waveform models when we do not shift the signals by the TDOA. First, notice that as we increase the number of channels, D+S continues to improve, since finer spatial sampling reduces the sidelobes of the spatial response, leading to increased suppression of noise and reverberation energy arriving from other directions. Second, notice that TAM always has better performance than D+S, as TAM is more general than D+S because it allows individual channels to be filtered before being combined. But notice that the raw waveform CLDNN, without any explicit time alignment or localization (TDOA estimation), performs as well as TAM with the time alignment. This shows us that the trained un-aligned network is implicitly robust to varying TDOA.

3.6 Summary

Model	WER - CE	WER - Seq
raw, 1ch	23.5	19.3
D+S, 8ch, oracle	22.4	18.8
MVDR, 8ch, oracle	22.5	18.7
raw, unfactored, 2ch	21.8	18.2
raw, unfactored, 4ch	20.8	17.2
raw, unfactored, 8ch	20.6	17.2

Table 4: Raw waveform model WER after sequence training. All models use 128 filters.

To conclude this section, we show the results after sequence training in Table 4. We also include results for 8 channel oracle D+S, where the true target speech TDOA is known, as well as oracle MVDR [39] where the true speech and noise estimates are known in addition to the target TDOA. Table 4 shows that the raw unfactored model, even using only 2 channel inputs and no oracle information, outperforms the single channel and oracle signal processing methods. Using 4 channel inputs, the raw-waveform unfactored model achieves between an 8-10% relative improvement over single channel, D+S and MVDR.

4 Factoring Spatial and Spectral Selectivity

4.1 Architecture

In multichannel speech recognition systems, multichannel spatial filtering is often performed separately from single channel feature extraction. However, in the unfactored raw-waveform model, spatial and spectral filtering are done in one layer of the network. In this section, we factor out spatial and spectral filtering into separate layers, as shown in Figure 4.

The motivation for this architecture is to design the first layer to be spatially selective, while implementing a frequency decomposition shared across all spatial filters in the second layer. Thus the combined output of the second layer will be the Cartesian product of all spatial and spectral filters.

The first layer, denoted by `tConv1` in the figure, again models Equation 2 and performs a multichannel time-convolution with a FIR spatial filterbank. The operation of each filter $p \in \{0,\dots,P-1\}$, which we will refer to as a spatial look direction in the factored model, can again be interpreted as a filter-and-sum beamformer, except that any overall time shift is implicit in the filter coefficients rather than being explicitly represented as in Equation 1. The main differences between the unfactored and factored approaches are as follows. First, both the filter size N

and number of filters P are much smaller in order to encourage the network to learn filters with a broadband response in frequency that span a small number of spatial look directions needed to cover all possible target speaker locations. The shorter filters in this layer will have worse frequency resolution than those in the unfactored model, but that will be dealt with in the next layer. We hope that this poor frequency resolution will encourage the network to use this first layer to focus on spatial filtering, with a limited spectral response. To make the combination of the first two layers of the factored model conceptually similar to the first layer of the unfactored model (i.e., a bank of bandpassed beamformers), the multi-channel (first) filter layer is not followed by any non-linear compression (i.e. ReLU, log), and we do not perform any pooling between the first and second layers.

The second time-convolution layer, denoted by `tConv2` in the figure, consists of longer-duration single-channel filters. It therefore can learn a decomposition with better frequency resolution than the first layer but is incapable of doing any spatial filtering. Given the P feature maps from the first layer, we perform a time convolution on each of these signals, very similar to the single-channel time-convolution layer described in [32], except that the time convolution is shared across all P feature maps or "look directions". We denote this layer's filters as $g \in \Re^{L \times F \times 1}$, where 1 indicates sharing across the P input feature maps. The "valid" convolution produces an output $w[t] \in \Re^{(M-L+1) \times F \times P}$. The output of the spectral convolution layer, for each look direction p and each filter f, is given by Equation 3.

$$w_f^p[t] = y^p[t] * g_f \tag{3}$$

Next, we pool the filterbank output in time thereby discarding short-time (i.e. phase) information, over the entire time length of the output signal, to produce an output of dimension $1 \times F \times P$. Finally, we apply a rectified non-linearity, followed by a stabilized logarithm compression, to produce a frame-level feature vector at frame l, i.e., $z_l \in \Re^{1 \times F \times P}$, which is then passed to a CLDNN model. We then shift the window of the raw waveform by a small (10ms) hop and repeat this time convolution to produce a set of time-frequency-direction frames at 10ms intervals.

4.2 Number of Spatial Filters

We first explore the behavior of the proposed factored multichannel architecture as the number of spatial filters P varies. Table 5 shows that we get good improvements up to 10 spatial filters. We did not explore above 10 filters due to the computational complexities of passing 10 feature maps to the `tConv2` layer. The factored network, with 10 spatial filters, achieves a WER of 20.4%, a 6% relative improvement over the 2 channel unfactored multichannel raw-waveform CLDNN. It is important to note that since the `tConv2` layer is shared across all look directions P, the total number of parameters is actually less than the unfactored model.

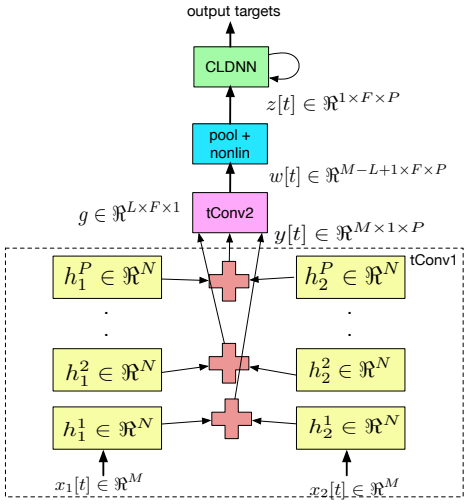


Fig. 4: Factored multichannel raw waveform CLDNN architecture for *P* look directions. The figure shows two channels for simplicity.

# Spatial Filters *P*	WER
baseline 2 ch, raw [31]	21.8
1	23.6
3	21.6
5	20.7
10	20.4

Table 5: WER when varying the size of the spatial filters in `tConv1`. All models use 128 filters for `tConv2` and results are presented for 2 channels.

4.3 Filter Analysis

To better understand what the `tConv1` layer learns, Figure 5 plots two-channel filter coefficients and the corresponding spatial responses, or beampatterns, after training.

Despite the intuition described in Section 4, the first layer filters appear to perform both spatial and spectral filtering. However, the beampatterns can nevertheless be categorized into a few broad classes. For example, filters 2, 3, 5, 7, and 9 in Figure 5 only pass through some low frequency subbands below about 1.5 kHz, where most vowel energy occurs, but steered to have nulls in different directions. Very little spatial filtering is done in high-frequency regions, where many fricatives and stops occur. The low frequencies are most useful for localization because they are not subject to spatial aliasing and because they contain much of the energy in the speech signal; perhaps that is why the network exhibits this structure.

To further understand the benefit of the spatial and spectral filtering in `tConv1`, we enforce this layer to only perform spatial filtering by initializing the filters to be an impulse centered at a delay of zero for channel 0, and offset from zero in

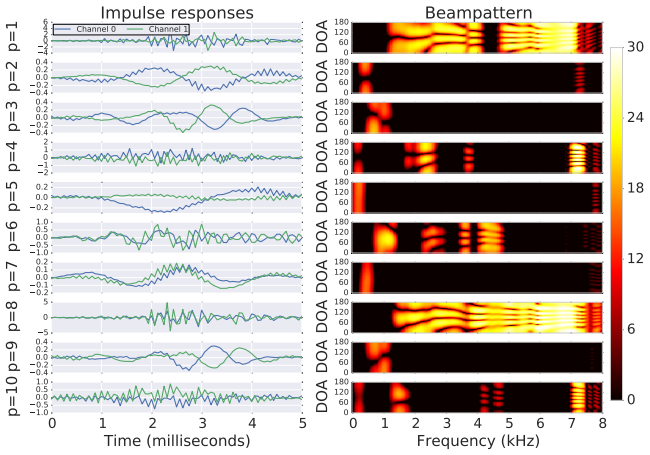


Fig. 5: Trained filters and spatial responses for 10 spatial directions.

channel 1 by different delays for each filter. By not training this layer, this amounts to performing delay-and-sum filtering across a set of fixed look directions. Table 6 compares performance when fixing vs. training the `tConv1` layer. The results demonstrate that learning the filter parameters, and therefore performing some spectral decomposition, improves performance over keeping this layer fixed.

# Spatial Filters P	`tConv1` Layer	WER
5	fixed	21.9
5	trained	20.9

Table 6: WER for training vs. fixing the `tConv1` layer, 2 channel.

4.4 Results Summary

To conclude this section, we show the results after sequence training, comparing the factored and unfactored models. Notice that the 2 channel factored model provides 6% relative improvement over the unfactored model, while the 4 channel model provides 5% relative improvement. We do not go above 4 channels, as results from Table 4 in Section 3.6 show that there is no difference between 4 and 8 channels.

Method	WER - CE	WER - Seq
raw, unfactored, 2ch	21.8	18.2
raw, factored, 2ch	20.4	17.2
raw, unfactored, 4ch	20.8	17.2
raw, factored, 4ch	19.6	16.3

Table 7: Factored Model WER after sequence training, simulated

5 Adaptive Beamforming

While the unfactored model improves over the factored model, the model also suffers from a few drawbacks. First, the learned filters in this model are fixed during decoding, which potentially limits the ability to adapt to previously unseen or changing conditions. In addition, since the factored model must perform spectral filtering for every look direction, this comes with a large computational complexity.

5.1 NAB Model

To address the limited adaptability and reduce the computational complexity of the models from [31, 30], we propose a neural network adaptive beamforming (NAB) model [21] which re-estimates a set of spatial filter coefficients at each input frame. The NAB model is depicted in Figure 6. At each time frame l, it takes in a small window of M waveform samples for each channel c from the C channel inputs, denoted as $x_0(l)[t], x_1(l)[t], \cdots, x_{C-1}(l)[t]$ for $t \in \{0, \cdots, M-1\}$. Additional to previous notations, the frame index l is explicitly used in this section to emphasize the frame dependent filtering coefficients. For simplicity, the figure shows an NAB model with $C = 2$ channels. We will describe the different NAB modules in subsequent subsections.

5.1.1 Adaptive Filters

The adaptive filtering layer is given by Equation 4, where $h_c(l)[n]$ is the estimated filter for channel c at time frame l. This model is very similar to the FS model from Equation 1, except now the steering delay τ_c is implicitly absorbed into the estimated filter parameters.

$$y(l)[t] = \sum_{c=0}^{C-1} \sum_{n=0}^{N-1} h_c(l)[n] x_c(l)[t-n] \tag{4}$$

In order to estimate $h_c(l)[t]$, we train a filter prediction (FP) module with one shared LSTM layer, one layer of channel-dependent LSTMs and linear output projection layers to predict N filter coefficients for each channel. The input to the *FP*

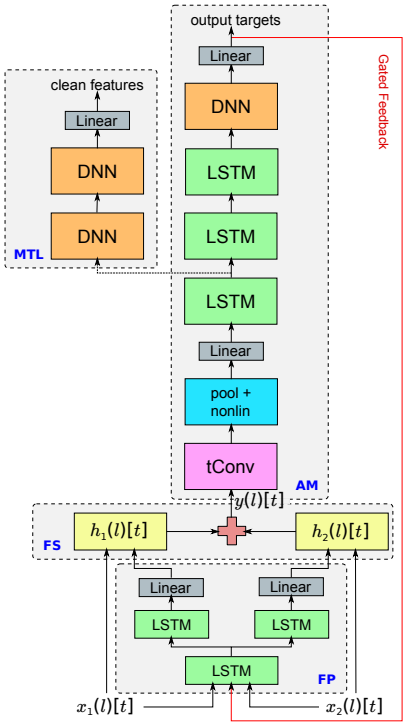


Fig. 6: *Neural network adaptive beamforming (NAB) model architecture. It consists of filter prediction (FP), filter-and-sum (FS) beamforming, acoustic modeling (AM) and multi-task learning (MTL). The figure shows two channels for simplicity.*

module is a concatenation of frames of raw input samples $x_c(l)[t]$ from all the channels, and can also include features typically computed for localization such as cross correlation features [20, 40, 41]. The estimation of *FP module* parameters are jointly done with AM parameters by directly minimizing a cross-entropy or sequence loss function. Following Equation 4 the estimated filter coefficients $h_c(l)[t]$ are convolved with input samples $x_c(l)[t]$ for each channel. The outputs of the convolution are summed across channels to produce a single channel signal $y(l)[t]$.

After adaptive FS, the single channel enhanced signal $y(l)[t]$ is passed to an *AM module* (Figure 6). We adopt the single channel raw waveform CLDNN model [32] for acoustic modeling, except that we now skip the frequency convolution layer as it has recently been shown in [27] to not help for noisier tasks. During training, the AM and FP (Figure 6) are trained jointly.

5.1.2 Gated Feedback

Augmenting the network input at each frame with the prediction from the previous frame has been shown to improve performance [3]. To investigate the benefit of

feedback in the NAB model, we pass the AM prediction at frame $l-1$ back to the FP model at time frame l (red line in Figure 6). Since the softmax prediction is very high dimensional, we feed back the low-rank activations preceding the softmax to the *FP module* to limit the increase of model parameters [42].

This feedback connection gives the *FP module* high level information about the phonemic content of the signal to aid in estimating beamforming filter coefficients. This feedback is comprised of model *predictions* which may contain errors, particularly early in training, and therefore might lead to poor model training [3]. A gating mechanism [8] is hence introduced to the connection to modulate the degree of feedback. Unlike conventional LSTM gates, which control each dimension independently, we use a global scalar gate to moderate the feedback. The gate $g^{\mathtt{fb}}(l)$ at time frame l, is computed from the input waveform samples $\boldsymbol{x}(l)$, the state of the first FP LSTM layer $\boldsymbol{s}(l-1)$, and the feedback vector $\boldsymbol{v}(l-1)$, as follows:

$$g^{\mathtt{fb}}(l) = \sigma(\boldsymbol{w}_x^T \cdot \boldsymbol{x}(l) + \boldsymbol{w}_s^T \cdot \boldsymbol{s}(l-1) + \boldsymbol{w}_v^T \cdot \boldsymbol{v}(l-1)) \tag{5}$$

where $\boldsymbol{w}_x$, $\boldsymbol{w}_s$ and $\boldsymbol{w}_v$ are the corresponding weight vectors and σ is an elementwise non-linearity. We use a logistic function for σ which outputs values in the range $[0,1]$, where 0 cuts off the feedback connection and 1 directly passes the feedback through. The effective FP input is hence $\Big[\boldsymbol{x}(l), \quad g^{\mathtt{fb}}(l)\boldsymbol{v}(l-1)\Big]$.

5.1.3 Regularization with MTL

Multi-task learning has been shown to yield improved robustness [30, 12, 7]. We adopt an *MTL module* similar to [30] during training by configuring the network to have two outputs, one recognition output which predicts CD states and a second denoising output which reconstructs 128 log-mel features derived from the underlying clean signal. The denoising output is only used in training to regularize the model parameters; the associated layers are discarded during inference. In the NAB model the *MTL module* branches off of the first LSTM layer of the *AM module*, as shown in Figure 6. The *MTL module* is composed of two fully connected DNN layers followed by a linear output layer which predicts clean features. During training the gradients back propagated from the two outputs are weighted by α and $1-\alpha$ for the recognition and denoising outputs respectively.

5.2 NAB Filter Analysis

The best NAB model found in [21] has the following configurations:

1. the *FP module* has one shared 512-cell LSTM layer across channels, one layer of channel-dependent 256-cell LSTMs and one layer of channel-dependent 25-dimensional linear projection layer;

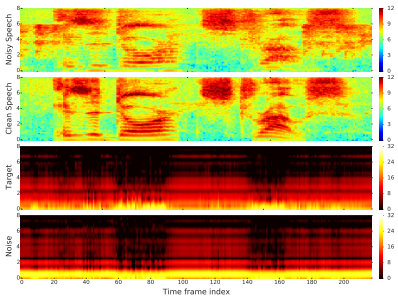


Fig. 7: *Visualizations of the predicted beamformer responses at different frequency (Y-axis) across time (X-axis) at the target speech direction (3rd) and interfering noise direction (4th) with the noisy (1st) and clean (2nd) speech spectrograms.*

2. the *FP module* takes in the concatenation of raw waveform samples from each channel;
3. the *FP module* outputs a 1.5ms filter (25-dimensional vector) for each channel;
4. the *AM module* is a single channel raw waveform LDNN model [32] with 256 `tConv` filters and without the frequency convolution layer [27], which is also similar to other multichannel models discussed in this chapter;
5. 128-dimensional clean log-mel features are used as the secondary reconstruction objectives with a weight of 0.1 for MTL;
6. per-frame gated feedback connection from the bottleneck layer right before the *AM module*'s softmax layer is appended to the *FP module*'s input.

Figure 7 illustrates the frequency responses of the predicted beamforming filters at the target speech and interfering noise directions. The SNR for this utterance is 12dB. The responses in the target speech direction have relatively more speech-dependent variations than those in the noise direction. This may indicate that the predicted filters are attending to the speech signal. Besides, the responses at high speech-energy regions are generally lower than others, which suggests the automatic gain control effect of the predicted filters.

5.3 Result Summary

Finally, to conclude this section, we show the results after sequence training compared to the factored model. Since the NAB model is trained without frequency convolution (i.e., LDNN), we do the same for the factored model. Table 8 shows that while the factored model can potentially handle different directions by enumerating many look directions in the spatial filtering layer, the adaptive model can achieve similar performance with much less computational complexity, as measured by both the parameters and multiplies and additions (M+A) of the model, as shown in the Table.

Model	WER (%) CE	Seq.	Param (M)	MultAdd (M)
factored	20.4	17.1	18.9	35.1
NAB	20.5	17.2	24.0	28.8

Table 8: Comparison between 2-channel factored and adaptive models.

6 Filtering in the Frequency Domain

Until now, we have presented three multichannel models in the time domain. However, it is well know that convolution between two time domain signals is equivalent to the element-wise product of their frequency domain counterparts [4, 5]. A benefit of operating in the complex FFT space is that element-wise products are much faster to compute compared to convolutions, particularly when the convolution filters and input size is large as in our multichannel raw waveform models. In this section, we describe how we can implement both the factored Model from Section 4 and the NAB Model from Section 5, in the frequency domain.

6.1 Factored Model

In this section, we describe the factored model in the frequency domain.

6.1.1 Spatial Filtering

For frame index l and channel c, we denote $X_c[l] \in \mathbb{C}^K$ as the result of an M-point Fast Fourier Transform (FFT) of $x_c[t]$ and $H_c^p \in \mathbb{C}^K$ as the FFT of h_c^p. Note that we ignore negative frequencies because the time domain inputs are real, and thus our frequency domain representation of an M-point FFT contains only $K = M/2+1$ unique complex-valued frequency bands. The spatial convolution layer in Equation 2 can be represented by Equation 6 in the frequency domain, where $\cdot$ denotes element-wise product. We denote the output of this layer as $Y^p[l] \in \mathbb{C}^K$ for each look direction p:

$$Y^p[l] = \sum_{c=0}^{C} X_c[l] \cdot H_{c-1}^p \tag{6}$$

There are many different algorithms for implementing the "spectral filtering" layer in the frequency domain, some of which are presented here [28]. Just to give readers a high level overview of "spectral filtering", in this chapter we choose to describe only the Complex Linear Projection [38] method.

6.1.2 Spectral Filtering: Complex Linear Projection

It is straightforward to rewrite the convolution in Equation 3 as an element-wise product in frequency, for each filter f and look direction p:

$$W_f^p[l] = Y^p[l] \cdot G_f \tag{7}$$

In the above equation, $W_f^p[l] \in \mathbb{C}^K$ and $G_f \in \mathbb{C}^K$ is the FFT of the time domain filter g_f in Equation 3. There is no frequency domain equivalent to the max-pooling operation in the time domain. Therefore to mimic max-pooling exactly requires taking the inverse FFT of $W_f^p[l]$ and performing the pooling operation in the time domain, which is computationally expensive to do for each look direction p and filter output f.

As an alternative [38] recently proposed the Complex Linear Projection (CLP) model which performs average pooling in the frequency domain and results in similar performance to a single channel raw waveform model. Similar to the waveform model the pooling operation is followed by a point-wise absolute-value nonlinearity and log compression. The 1-dimensional output for look direction p and filter f is given by:

$$Z_f^p[l] = \log \left| \sum_{k=1}^{N} W_f^p[l,k] \right| \tag{8}$$

6.2 NAB Model

In the frequency-domain NAB setup, we have an LSTM which predicts complex FFT (CFFT) inputs for both channels. Given a 512-point FFT input, this amounts to predicting 4×257 frequency points for real and imaginary components for 2 channels, which is much more than the predicted filter size in the time domain (i.e., $1.5ms = 25$ taps). After the complex filters are predicted for each channel, element-wise product is done with the FFT of the input for each channel, mimicking the convolution in Equation 4 in the frequency domain. The output of this is given to a single channel LDNN in the frequency domain, which does spectral decomposition using the CLP method, and then acoustic modeling.

6.3 Results: Factored Model

6.3.1 Performance

First, we explore the performance of the frequency domain factored model. Note this model does not have any frequency convolution layer. We explore this for a similar setting to most efficient raw-waveform factored setup [28], namely $P = 5$

look directions in the spatial layer and $F = 128$ filters in the spectral layer. The input is 32ms instead of 35ms like raw-waveform, as this allows us to take a $D = 512$-point FFT without zero-padding at a sampling rate of 16kHz. A 35-ms input would have required us to take a 1024-point FFT, and we have not found any big difference in performance between 32 and 35ms inputs for raw-waveform.

Table 9 shows the performance of the time and frequency domain factored models, as well as the total number of multiplication and addition operations (M+A) for different layers of the model. The table shows that the CLP factored model reduces the number of operations by a factor of 1.9x over the best waveform model, with a small degradation in WER.

Model	Spatial M+A	Spectral M+A	Total M+A	WER CE	WER ST
time	525.6K	15.71M	35.1M	20.4	17.1
CLP	10.3K	655.4K	19.6M	20.5	17.2

Table 9: Frequency Domain Factored Model Performance

However, given that the frequency models are more computationally efficient, we explore improving WER by increasing the window size (and therefore computational complexity) of the factored models. Specifically, since longer windows typically help with localization [39], we explore using 64ms input windows for both models. With a 64ms input, the frequency models require a 1024-point FFT. Table 10 shows that the frequency models improve the WER over using a smaller 32ms input, and still perform roughly the same. However, the frequency model now has an even larger computational complexity savings of 2.7x savings compared to the time domain model.

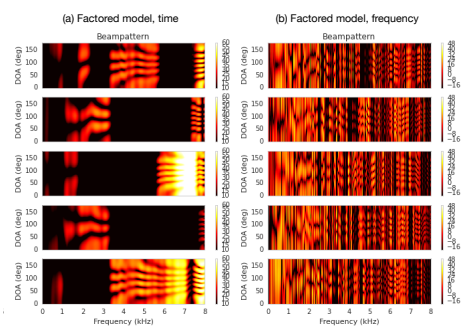
Feat	Spatial M+A	Spectral M+A	Total M+A	WER ST
time	906.1K	33.81M	53.6M	17.1
CLP	20.5K	1.3M	20.2M	17.1

Table 10: Results with a 64ms Window Size

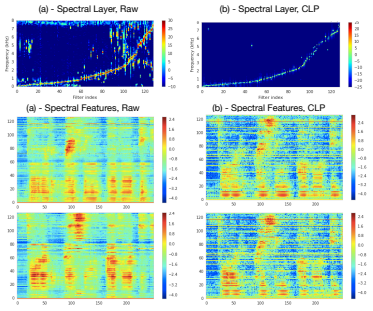
6.3.2 Comparison between learning in time vs. frequency

Figure 8a shows the spatial responses (i.e., beampatterns) for both the time and frequency domain spatial layers. The beampatterns show the magnitude response in dB as a function of frequency and direction of arrival, i.e. each horizontal slice of the beampattern corresponds to the filter's magnitude response for a signal coming from a particular direction. In each frequency band (vertical slice), lighter shades indicate that sounds from those directions are passed through, while darker shades indicate directions whose energy is attenuated. The figures show that the spatial filters

learned in the time domain are band-limited, unlike those learned in the frequency domain. Furthermore, the peaks and nulls are aligned well across frequencies for the time domain filters.



(a) Beampatterns of Time and Frequency Models



(b) Time and Frequency Domain Spatial Responses

The differences between these models can further be seen in the magnitude responses of the spectral layer filters, as well as in the outputs of the spectral layers from different look directions plotted for an example signal. Figure 8b illustrates that the magnitude responses in both time and CLP models look qualitatively similar, and learn bandpass filters with increasing center frequency. However, because the spatial layers in time and frequency are quite different, we see that the spectral layer outputs in time are much more diverse in different spatial directions compared to the CLP model.

At some level, time-domain and frequency-domain representations are interchangeable, but they result in networks that are parameterized very differently. Even though the time and frequency models all learn different spatial filters, they all seem to have similar WERs. There are roughly 18M parameters in the LDNN model that sits above the spatial/spectral layers, which accounts for over 90% of the parameters in the model. Any differences between the spatial layers in time and frequency are likely accounted for in the LDNN part of the network.

6.4 Results: Adaptive Model

Next, we explore the performance of the frequency domain NAB model. Table 11 shows the WER and computational complexity of the raw-waveform and CLP NAB models. While using CLP features greatly reduces computational complexity, the performance is worse than the raw-waveform model. One hypothesis we have is that frequency domain processing requires predicting a higher dimensional filter, which we can see from the table leads to a degradation in performance.

Model	WER (%)	Param (M)	MultAdd (M)
raw	20.5	24.6	35.3
CLP	21.0	24.7	25.1

Table 11: Comparison between time and frequency NAB models.

7 Final Comparison, Re-recorded Data

Finally, we also evaluated the performance of different multichannel models presented in this chapter on a real "Rerecorded" test set. Reverberation-I is when the microphone is placed on a coffee table, whereas Reverberation-II is when the mic is placed on a TV stand. Since this set contains a circular microphone geometry but our models are trained on a linear microphone geometry, we only report results with 2 channels to form a linear array with a 7.5cm spacing. The models however are trained with a 14cm spacing.

Table 12 shows the results with different multichannel models. All raw-waveform models are trained with 35-ms inputs and 128 spectral decomposition filters. The factored model has 5 look directions. The CLP factored model is trained with a 64-ms input, 5 look directions, and 256 spectral decomposition filters. All frontends use an LDNN architecture in the upper layers of the network.

Notice that the 2 channel raw factored model gives a 13% relative improvement over single channel, with larger improvements in noisier test sets, which is to be expected. In addition, the CLP factored model performs slightly worse than the raw factored model on this test set. One hypothesis is that the CLP factored model captures much less spatial diversity than the raw waveform model, as shown in Figures 8a and 8b. Finally, the NAB model performs much worse than the factored model. Perhaps because the NAB model learns a set of adaptive filters, it is more sensitive to mismatches between training and test conditions compared to the other models.

Model	Rev.-I	Rev.-II	Rev.-I Noisy	Rev.-II Noisy	Ave
1 channel raw	18.6	18.5	27.8	26.7	22.9
2 channel raw, unfactored	17.9	17.6	25.9	24.7	21.5
2 channel raw, factored	17.1	16.9	24.6	24.2	20.7
2 channel CLP, factored	17.4	17.1	25.7	24.4	21.2
2 channel raw, NAB	17.8	18.1	27.1	26.1	22.3

Table 12: WER on "Rerecorded" set

8 Conclusions and Future Work

In this chapter, we introduced a methodology to do multichannel enhancement and acoustic modeling jointly within a neural network framework. First, we developed a unfactored raw-waveform multichannel model, and showed that this model performed as well as a model given oracle knowledge of the true location. Next, we introduced a factored multichannel model to separate out spatial and spectral filtering operations, and found that this offered an improvement over the unfactored model. Next, we introduced an adaptive beamforming method, which we found to match the performance of the multichannel model with far fewer computations. Finally, we showed that we can match the performance of the raw-waveform factored model, with far fewer computations, with a frequency-domain factored model. Overall, the factored model provides between a 5-13% relative improvement over single channel and traditional signal processing techniques, on both simulated and rerecorded sets.

References

[1] Allen, J.B., Berkley, D.A.: Image Method for Efficiently Simulation Room-Small Acoustics. Journal of the Acoustical Society of America **65**(4), 943 – 950 (1979)
[2] Benesty, J., Chen, J., Huang, Y.: Microphone Array Signal Processing. Springer (2009)
[3] Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. In: Advances in Neural Information Processing Systems, pp. 1171–1179 (2015)
[4] Bengio, Y., Lecun, Y.: Scaling Learning Algorithms Towards AI. Large Scale Kernel Machines (2007)
[5] Bracewell, R.: The Fourier Transform and Its Applications, 3 edn. McGraw-Hill (1999)
[6] Brandstein, M., Ward, D.: Microphone Arrays: Signal Processing Techniques and Applications. Springer (2001)
[7] Chen, Z., Watanabe, S., Erdoğan, H., Hershey, J.R.: Speech enhancement and recognition using multi-task learning of long short-term memory recurrent neural networks. In: Proc. Interspeech, pp. 3274–3278. ISCA (2015)
[8] Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Gated feedback recurrent neural networks. arXiv preprint arXiv:1502.02367 (2015)
[9] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Le, Q., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., Ng, A.: Large Scale Distributed Deep Networks. In: Proc. NIPS (2012)
[10] Delcroix, M., Yoshioka, T., Ogawa, A., Kubo, Y., Fujimoto, M., Ito, N., Kinoshita, K., Espi, M., Hori, T., Nakatani, T., Nakamura, A.: Linear Prediction-

based Dereverberation with Advanced Speech Enhancement and Recognition Technologies for the REVERB Challenge. In: REVERB Workshop (2014)
[11] Dieleman, S., Schrauwen, B.: End-to-end learning for music audio. In: Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, pp. 6964–6968. IEEE (2014)
[12] Giri, R., Seltzer, M.L., Droppo, J., Yu, D.: Improving speech recognition in reverberation using a room-aware deep neural network and multi-task learning. In: Proc. ICASSP, pp. 5014–5018. IEEE (2015)
[13] Glorot, X., Bengio, Y.: Understanding the Difficulty of Training Deep Feedforward Neural Networks. In: Proc. AISTATS (2014)
[14] Griffiths, L.J., Jim, C.W.: An alternative approach to linearly constrained adaptive beamforming. IEEE Transactions on Antennas and Propagation **30**(1), 27–34 (1982)
[15] Hain, T., Burget, L., Dines, J., Garner, P., Grezl, F., Hannani, A., Huijbregts, M., Karafiat, M., Lincoln, M., Wan, V.: Transcribing Meetings with the AMIDA Systems. IEEE Transactions on Audio, Speech, and Language Processing **20**(2), 486–498 (2012)
[16] Heigold, G., McDermott, E., Vanhoucke, V., Senior, A., Bacchiani, M.: Asynchronous Stochastic Optimization for Sequence Training of Deep Neural Networks. In: Proc. ICASSP (2014)
[17] Hershey, J.R., Roux, J.L., Weninger, F.: Deep Unfolding: Model-Based Inspiration of Novel Deep Architectures. CoRR **abs/1409.2574** (2014)
[18] Hoshen, Y., Weiss, R.J., Wilson, K.W.: Speech Acoustic Modeling from Raw Multichannel Waveforms. In: Proc. ICASSP (2015)
[19] Jaitly, N., Hinton, G.: Learning a Better Representation of Speech Soundwaves using Restricted Boltzmann Machines. In: Proc. ICASSP (2011)
[20] Knapp, C.H., Carter, G.C.: The generalized correlation method for estimation of time delay. Acoustics, Speech and Signal Processing, IEEE Transactions on **24**(4), 320–327 (1976)
[21] Li, B., Sainath, T.N., Weiss, R.J., Wilson, K.W., Bacchiani, M.: Neural Network Adaptive Beamforming for Robust Multichannel Speech Recognition. In: Proc. Interspeech (2016)
[22] Liu, Y., Zhang, P., Hain, T.: Using Neural Network Front-ends on Far-field Multiple Microphones based Speech Recognition. In: Proc. ICASSP (2014)
[23] Mohamed, A., Hinton, G., Penn, G.: Understanding how Deep Belief Networks Perform Acoustic Modelling. In: ICASSP (2012)
[24] Palaz, D., Collobert, R., Doss, M.: Estimating Phoneme Class Conditional Probabilities From Raw Speech Signal using Convolutional Neural Networks. In: Proc. Interspeech (2014)
[25] Sainath, T.N., Kingsbury, B., Mohamed, A., Dahl, G., Saon, G., Soltau, H., Beran, T., Aravkin, A., Ramabhadran, B.: Improvements to Deep Convolutional Neural Networks for LVCSR. In: Proc. ASRU (2013)
[26] Sainath, T.N., Kingsbury, B., Sindhwani, V., Arisoy, E., Ramabhadran, B.: "Low-Rank Matrix Factorization for Deep Neural Network Training with High-Dimensional Output Targets. In: Proc. ICASSP (2013)

[27] Sainath, T.N., Li, B.: Modeling Time-Frequency Patterns with LSTM vs. Convolutional Architectures for LVCSR Tasks. In: Proc. Interspeech (2016)
[28] Sainath, T.N., Narayanan, A., Weiss, R.J., Wilson, K.W., Bacchiani, M., Shafran, I.: Reducing the Computational Complexity of Multimicrophone Acoustic Models with Integrated Feature Extraction. In: Proc. Interspeech (2016)
[29] Sainath, T.N., Vinyals, O., Senior, A., Sak, H.: Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks. In: Proc. ICASSP (2015)
[30] Sainath, T.N., Weiss, R.J., Wilson, K.W., Narayanan, A., Bacchiani, M.: Factored Spatial and Spectral Multichannel Raw Waveform CLDNNs. In: Proc. ICASSP (2016)
[31] Sainath, T.N., Weiss, R.J., Wilson, K.W., Narayanan, A., Bacchiani, M., Senior, A.: Speaker Localization and Microphone Spacing Invariant Acoustic Modeling from Raw Multichannel Waveforms. In: Proc. ASRU (2015)
[32] Sainath, T.N., Weiss, R.J., Wilson, K.W., Senior, A., Vinyals, O.: Learning the Speech Front-end with Raw Waveform CLDNNs. In: Proc. Interspeech (2015)
[33] Sak, H., Senior, A., Beaufays, F.: Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. In: Proc. Interspeech (2014)
[34] Seltzer, M., Raj, B., Stern, R.M.: Likelihood-maximizing Beamforming for Robust Handsfree Speech Recognition. IEEE Trascations on Audio, Speech and Language Processing **12**(5), 489–498 (2004)
[35] Stolcke, A., Anguera, X., Boakye, K., Çetin, O., Janin, A., Magimai-Doss, M., Wooters, C., Zheng, J.: The SRI-ICSI Spring 2007 Meeting and Lecture Recognition System. Multimodal Technologies for Perception of Humans **Lecture Notes in Computer Science**(2), 450–463 (2008)
[36] Swietojanski, P., Ghoshal, A., Renals, S.: Hybrid Acoustic Models for Distant and Multichannel Large Vocabulary Speech Recognition. In: Proc. ASRU (2013)
[37] Tüske, Z., Golik, P., Schlüter, R., Ney, H.: Acoustic Modeling with Deep Neural Networks using Raw Time Signal for LVCSR. In: Proc. Interspeech (2014)
[38] Variani, E., Sainath, T.N., Shafran, I.: Complex Linear Projection (CLP): A Discriminative Approach to Joint Feature Extraction and Acoustic Modeling. In: Proc. Interspeech (2016)
[39] Veen, B.D., Buckley, K.M.: Beamforming: A Versatile Approach to Spatial Filtering. IEEE ASSP Magazine **5**(2), 4–24 (1988)
[40] Xiao, X., Watanabe, S., Erdogan, H., Lu, L., Hershey, J., Seltzer, M.L., Chen, G., Zhang, Y., Mandel, M., Yu, D.: Deep beamforming networks for multi-channel speech recognition
[41] Xiao, X., Zhao, S., Zhong, X., Jones, D.L., Chng, E.S., Li, H.: A learning-based approach to direction of arrival estimation in noisy and reverberant environments. In: Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on, pp. 2814–2818. IEEE (2015)

[42] Zhang, Y., Chuangsuwanich, E., Glass, J.R.: Extracting deep neural network bottleneck features using low-rank matrix factorization. In: ICASSP, pp. 185–189 (2014)