Raw Waveform Modelling for ASR

A Literature Review

Part I

Erfan Loweimi

Centre for Speech Technology Research (CSTR)
The University of Edinburgh

Listen! 12.2.2020

ASR via Divide-and-Conquer Paradigm

- Divide into several simpler & directly solvable sub-tasks:
  - Feature extraction, Language Modelling, Acoustic Modelling
  - Solved/Optimised independently
- Feature Extraction → human speech perception & production
- Acoustic Modelling → Sequence & Statistical Modelling

ASR via Divide-and-Conquer Paradigm

- Divide into several simpler & directly solvable sub-tasks:
  - Feature extraction, Language Modelling, Acoustic Modelling
  - Solved/Optimised independently
- Feature Extraction → human speech perception & production
- Acoustic Modelling → Sequence & Statistical Modelling
- Raw waveform modelling premise
  - DNNs are powerful enough to solve FE and AM simultaneously

Acoustic Modelling using Raw Waveform – Advantages

- Learned vs handcrafted pipeline
  - Task-oriented
    - Optimal for the task (Jointly learned with the AM)
  - Employ all signal information
    - No info loss; Includes phase (all-pass) spectrum
  - Learning basis functions
    - Instead of Fourier transform's $exp(j\omega_k n)$
  - Mid-term processing rather than short-term processing
    - No need to accurate frame boundaries, learning masking, ...

Acoustic Modelling using Raw Waveform – Challenges

- High dimensional feature
  - DNNs as discriminative models are less sensitive to dim
  - Possible solution(s): CNN or matrix factorisation for MLP
- Using no prior knowledge about auditory system
  - Possible solution: first layer init. using prior knowledge
    - Nonparametric CNNs: Gammatone filters
    - Parametric CNNs: perceptual scales

Our Plan ...

- **Part I** → IDIAP + AACHEN
- **Part II** → Google + Multi-Resolution
- **Part III** → Parametric CNNs

Estimating Phoneme Class Conditional Probabilities from Raw Speech Signal using Convolutional Neural Networks

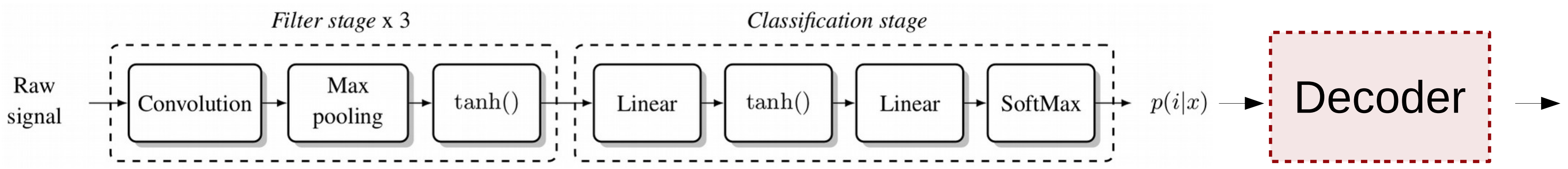
Dimitri Palaz[1,2], *Ronan Collobert*[1], *Mathew Magimai.-Doss*[1]

[1]Idiap Research Institute, Martigny, Switzerland
[2]Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
{dimitri.palaz, ronan.collobert, mathew}@idiap.ch

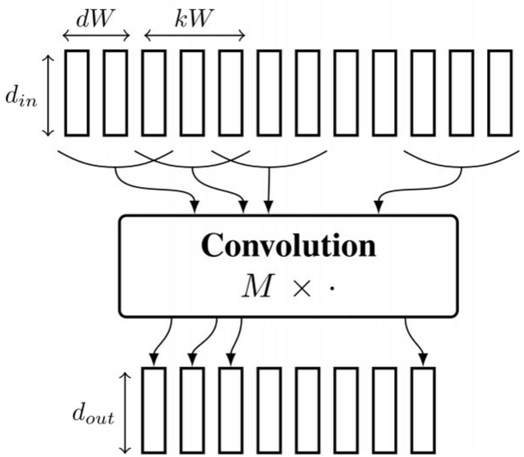
Palaz, et al, 2013 -- IDIAP

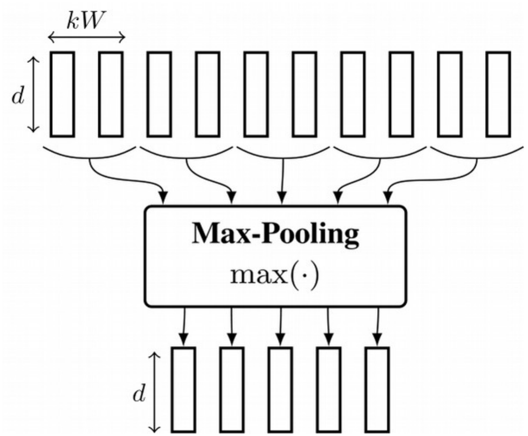
- **Goal**: Phoneme class conditional probability estimation from raw waveform
- **System** → CNN+MLP
- **Baseline**: MFCC + ANN
- **Task**: TIMIT

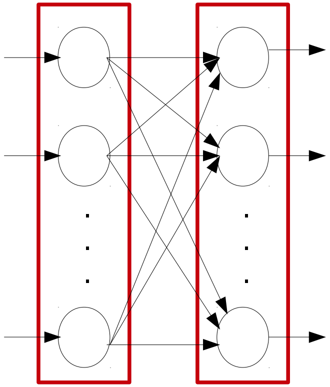
Architecture



CRF or HMM





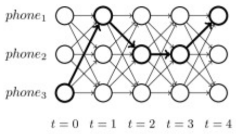


Decoder: CRF

- CRF: Discriminative model for structured prediction
  - Discourage/encourage unlikely/likely sequences
  - Learns transition between classes and duration modelling

$$\overset{\text{Path score}}{\boxed{s}}([x]_1^T ; [i]_1^T) = \sum_{t=1}^{T} \Big(\overset{\text{NN score for } x_t}{\boxed{f_{i_t}(x_t)}} + \overset{\text{Transition between labels}}{\boxed{A_{i_t, i_{t-1}}}} \Big)$$

- At inference time $\rightarrow$ find optimal path (Viterbi)



$$\underset{[j]_1^T}{\operatorname{argmax}} \; s([x]_1^T, [j]_1^T, \theta)$$

Experimental Setup

- Toolkit: Torch7 → Collobert et al., 2011
- Raw wave features Normalisation → MVN per-frame
- Hyperparameters tuning
  - Grid search on pre-specified ranges
  - Early-stopping on Dev set

Parameter	Range
Input window size (ms)	100-700
Kernel width (kW)	1-9
Number of filters per kernel (d_{out})	10-90
Number of hidden units in the class. stage	100-1500

- Decoding
  - CRF → without duration constrain
  - HMM → 3-state duration constraint + all phonemes equally probable

DNN Hyperparameter Tuning

- Optimal values for MFCC:
  - Context len: 30 frames (290ms)
  - kW: L_1: 39, L_2: 5, L_3: 7
  - dW: L_1: 10, L_2: 1, L_3: 1
  - #Filters: 80
  - Pooling width: 0
  - #nodes@hiddenUnits: 500

- Optimal values for Raw:
  - Frame length: 270 ms
  - kW: L_1: 10, L_2: 5, L_3: 9
  - dW: L_1: 10, L_2: 1, L_3: 1
  - #Filters: 90
  - Pooling width: 3
  - #nodes@hiddenUnits: 500

kW: kernel width
dW: stride

Hyperparameters; MFCC vs Raw

- Optimal frame length is Similar (270 vs 290 ms)
- kW-L_1: 10 vs 39 (samples) → longer filters for Raw
- dW-L_1: both 10 (samples)
- #filters: 80 vs 90
- Pooling width: 0 vs 3 → redundancy in Raw → subsampling
- #Nodes in MLP → both 500 → same requirement at high level

Phoneme Recognition on TIMIT

- MFCC is better that Raw
- CNN is better than MLP
  - Feature: MLP → CNN; Gain
  - MFCC (HMM): 66.65 → 70.52; **+3.87**
  - Raw (HMM): 38.91 → 67.88; **+28.97**
- CRF is better than HMM
  - Learns bigram LM over phonemes (?)
  - Gain → Raw: 1.59%; MFCC: 1.28% abs

Features	Arch.	Decoding	Num. param.	Test acc.
MFCC	MLP	HMM	196’040	66.65
Raw	MLP	HMM	740’540	38.91
Raw	CNN	HMM	720’110	67.88
Raw	CNN	CRF		69.47
MFCC	CNN	HMM	860’700	70.52
MFCC	CNN	CRF		71.80

Phoneme Recognition on TIMIT

- Systems:
  - Baseline: MFCC → MLP[2L] → HMM
  - CNN+CRF: Raw → CNN+MLP[2L] → CRF
  - Are they comparable?!
- Baseline is better than Raw
- High resolution class definition (for TIMIT) is better (?!)
  - Data size, CRF effect, ...

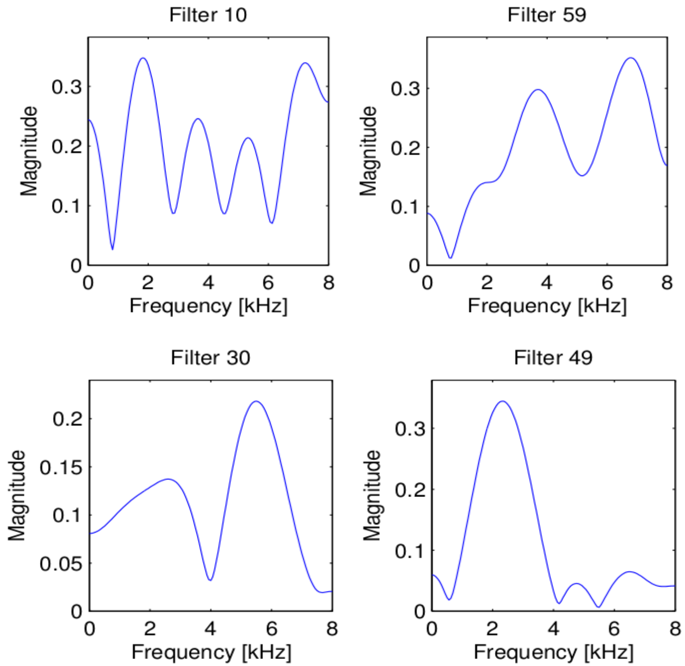
System	#Classes	#Param.	Test acc.
Baseline	39	196'040	66.65 %
CNN+CRF	39	873'340	65.81 %
CNN+CRF	117	986'680	67.84 %
CNN+CRF	183	803'363	70.08 %

– 183 = 3* x 61 (TIMIT orignal transcription)
– 117 = 3 x 39 (61** → collapsed to 39 [or 48])

* #states per phoneme
** Too narrow description for practical uses & easy to perplex

Frequency Response of Learned Filters

- Similarity to Auditory filtering is not investigated
  - Filters are multimodal
    - Centre frequency & BW (?)
    - Difficult to compare with auditory filters



- They are "… *Matching filters* ..."
  - "*Matched filter*" is the standard term!

CONVOLUTIONAL NEURAL NETWORKS-BASED CONTINUOUS SPEECH RECOGNITION USING RAW SPEECH SIGNAL

Dimitri Palaz[⋆†] *Mathew Magimai.-Doss*[⋆] *Ronan Collobert*[⋆]

[⋆] Idiap Research Institute, Martigny, Switzerland
[†] Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
{dimitri.palaz, mathew}@idiap.ch ronan@collobert.com

ICASSP 2015 Brisbane Australia

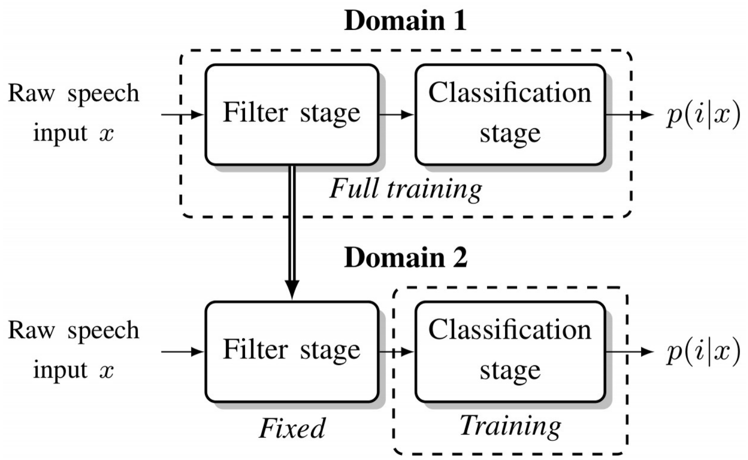
E. Loweimi

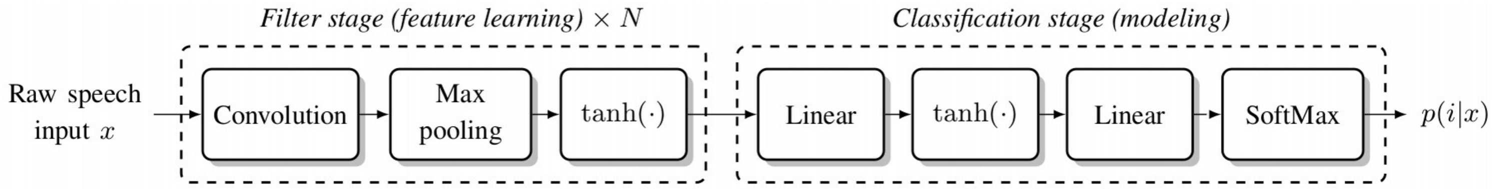
Contributions

- **Scalability**
  - LVCSR (WSJ SI-284)
- **Feature Invariance across Domains**
  - WSJ ↔ TIMIT
- **Filter Interpretation**

Feature Invariance across Domains

- **Stage 1**
  - Full training on Domain 1
- **Stage 2**
  - **Fix** the front-end (CNN)
  - **Train** the Fully-connected (FC)





Set up – Hyperparameters

- **Parameters** → w_{in}, kW_n, dW_n, d_n, kW_{mp}, *#hidden units*
- **Tuning** → Early stopping on dev + Grid search
- **Systems**
  - CNN-1L: 3xCNN + 1 HiddenLay MLP
  - CNN-3L: 3xCNN + 3 HiddenLay MLP

Parameters	Units	Range
Input window size (w_{in})	ms	100-700
Kernel width of the first conv. (kW_1)	samples	10-90
Kernel width of the n^{th} conv. (kW_n)	frames	1-11
Number of filters per kernel (d_{out})	filters	20-100
Max-pooling kernel width (kW_{mp})	frames	2-6
Number of hidden units in the classifier	units	200-1500

- **Optimal Setup for WSJ SI-284**
  - *W_{in}: 310ms, kW_n: first layer → 50 samples (3ms); other layers: 5 frames (0.6ms), dW_n: 10 samples, d_n: 80-60-60 filters, kW_{mp}: 2 pooling width, 500 / 1000 hidden units*

Experimental Results – LVCSR

- Raw is better than MFCC
- Deeper model is better, esp. for Raw
  - RWERR* in 1L to 3L
    - MFCC-ANN → 8.6%
    - Raw-CNN → 16.4%
- ANN vs CNN with identical #Params
  - Does it make the comparison fair?

Features	System	#Params.	WER
MFCC	*ANN-1L*	3.1M	7.0 %
MFCC	*ANN-3L*	5.6M	6.4 %
RAW	*CNN-1L*	3.1M	6.7 %
RAW	*CNN-3L*	5.6M	5.6 %

NN-xL → x: #hidden layers, 1 or 3

*Relative WER reduction

Transfer Learning

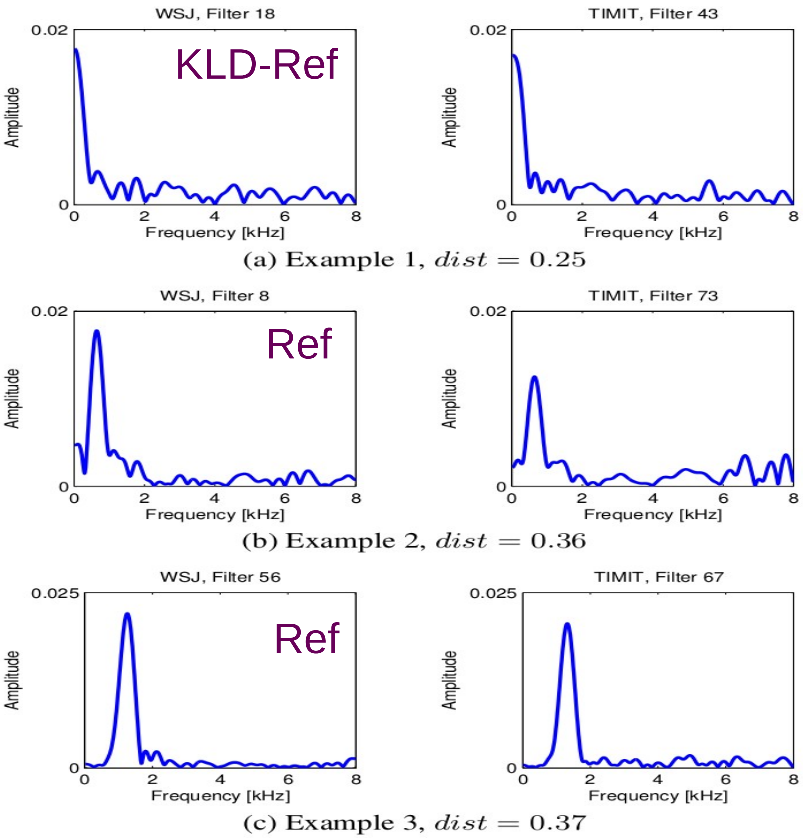
- Feature Invariance (Train domain → Test domain)
  - Let WT = "WSJ → TIMIT" & TW: "TIMIT → WSJ"
- Performance loss (mismatch - match)
  - WT - TT → 0.1%
  - TW - WW → 3.4%

Test domain	Features	Error Rate
TIMIT	Learned on TIMIT	32.3 %
	Learned in WSJ	32.4 %
WSJ	Learned on WSJ	6.7 %
	Learned on TIMIT	10.1 %

Features trained on WSJ work well for TIMIT, not vice versa!

WSJ and TIMIT Filters Comparison

- Comparison of learned filters
  - Similarity measure: KLD
- Some domain invariance ...
  - Is this comparison meaningful?
    - Recall WT and TW
- No significant activity at f > 2kHz



(a) Example 1, $dist = 0.25$

(b) Example 2, $dist = 0.36$

(c) Example 3, $dist = 0.37$

Analysis of CNN-based Speech Recognition System using Raw Speech as Input

Dimitri Palaz[1,2], *Mathew Magimai.-Doss*[1], *Ronan Collobert*[3,1]

[1]Idiap Research Institute, Martigny, Switzerland
[2]Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
[3]Facebook AI Research, Menlo Park, CA, USA
{dimitri.palaz, mathew}@idiap.ch, ronan@collobert.com

E. Loweimi

Contributions

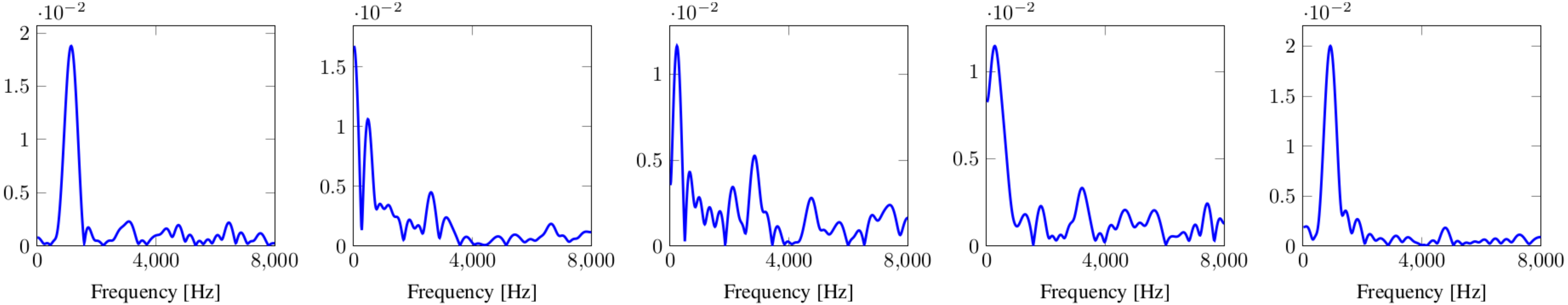
- Investigate performance in match and mismatch conditions
  - TIMIT+Noise & Aurora-2
- Further interpretation
  - Average Frequency Response for Vowels
  - Filter Analysis

Filter Analysis

- Compute **Vowel** Average Frequency Response

1) *n* = zeros(*num-vowels, num-filters*)
2) For *i*, *vowel* in enumerate(*Vowels*):
    1) Propagate centre frame of *vowel* into the NN
    2) j = argmax to find most firing filter # j is index of most firing filter
    3) *n[i,j]* += 1 *# number of times filter j is triggered when vowel i is propagated*
3) Keep the five most firing filters for each vowel and normalise the counts $\lambda_i = n_i / \Sigma_j n_j$
4) Compute weighted mean of Filters using normalised counts (λ_i)

Filter Analysis for vowel /*iy*/



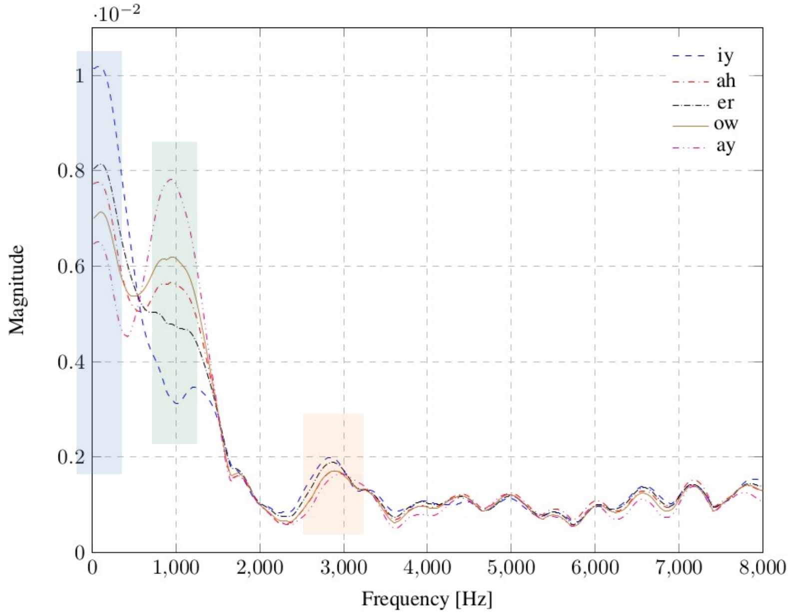
(a) $\lambda_1 = 0.28$ (b) $\lambda_2 = 0.23$ (c) $\lambda_3 = 0.22$ (d) $\lambda_4 = 0.17$ (e) $\lambda_5 = 0.08$

Five most firing filters, with their *proportion factor* (λ) for /*iy*/

bean

Average Frequency Response

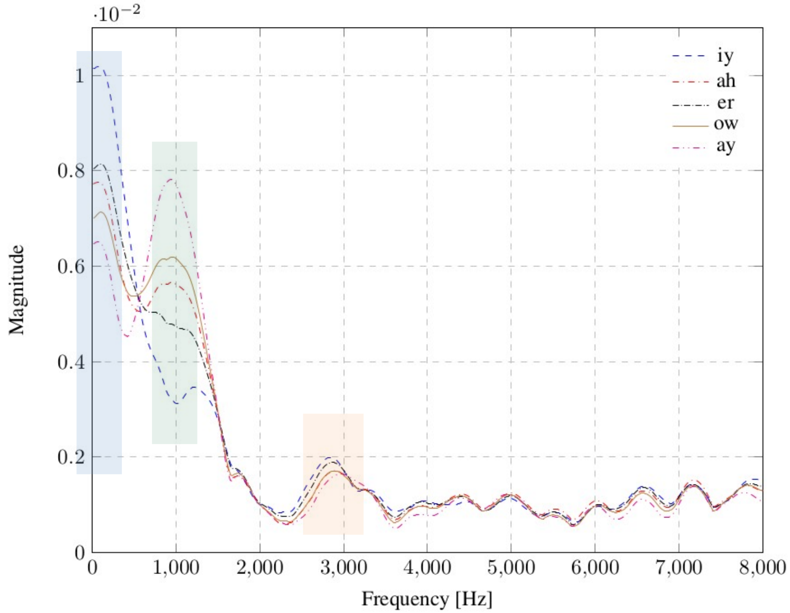
- Task → TIMIT phone recognition
- Data → TIMIT DevSet
- Vowels → */iy/*, */ah/*, /er/, /ow/ and */ay/*



Average Frequency Response

- Task → TIMIT phone recognition
- Data → TIMIT DevSet
- Vowels → /iy/, /ah/, /er/, /ow/ and /ay/

- How similar the spectrum is to vowels power spectrum?
- Formant structure ...
  - Many vowels have the same **F1**, **F2** and **F3**!!!



Mismatch Scenario – TIMIT

- TIMIT Multi-style training (NOT NTIMIT!)
  - Noise added via FaNT
  - Noise signals → NoiseX-92 database
  - Train Noise: Car, Operation, Lynx, Minigun; SNR: 5-20 dB + clean
  - Test Noise: F-16 and Factory; SNR: 0 to 30 dB

- Baseline → MFCC (w/o normalisation!!!) + ANN-1L [500 nodes]
- Raw: CNN-L3 + ANN-L2 [500 nodes]

#Hidden_ layers

Mismatch Scenario – TIMIT

- Mismatch (Clean)
  - Raw outperforms MFCC ≥ 10 dB
- Match (Multi)
  - Raw outperforms MFCC at all SNRs

SNR [dB]	ANN		CNN	
Training	clean	multi	clean	multi
30dB	52.5	54.3	65.5	66.8
25dB	46.7	50.8	59.7	64.8
20dB	40.3	46.6	50.5	60.8
15dB	32.7	41.1	39.1	53.5
10dB	26.1	34.2	27.8	42.8
5dB	21.2	26.4	18.3	30.8
0dB	17.4	20.2	9.9	21.4

Phone Recognition Rate (PRR)
ANN ↔ MFCC
CNN ↔ Raw

Mismatch Scenario – Aurora-2

ANN ↔ MFCC , CNN ↔ Raw

	Test A							Test B						
SNR [dB]	clean	20	15	10	5	0	-5	clean	20	15	10	5	0	-5
	Clean Training													
ANN	96.9	86.1	74.1	51.4	25.5	13.9	10.1	97.4	87.3	77.8	59.8	33.5	15.0	8.9
CNN	97.3	88.3	76.1	53.0	24.7	11.2	8.0	97.2	90.4	83.2	64.9	38.7	19.1	10.1
	Multi-conditional Training													
ANN	92.1	91.6	89.0	83.4	70.0	38.2	14.5	92.1	85.1	80.9	73.8	59.7	34.1	14.5
CNN	97.6	97.4	96.6	93.9	84.8	55.1	19.5	97.6	94.8	93.4	89.0	77.4	48.0	18.7

- Clean Training (Mismatch)
  - TestSet A: Raw outperform MFCC → SNR ≥ 10 dB
  - TestSet B: Raw outperforms MFCC, at all SNRs
- Multi-condition Training
  - Raw (CNN) outperforms MFCC (ANN), at all SNRs

Conclusion for TIMIT and Aurora-2

- In clean training (mismatch) condition
  - Raw outperforms MFCC ≥ 10 dB
- In multi-condition (match) scenario
  - Raw outperforms MFCC at ALL SNRs
- Comparison is not fair!!!
  - Shallow ANN vs Deep CNN
  - MFCCs are not normalised

Acoustic Modeling with Deep Neural Networks Using Raw Time Signal for LVCSR

Zoltán Tüske[1], *Pavel Golik*[1], *Ralf Schlüter*[1], *Hermann Ney*[1,2]

[1]Human Language Technology and Pattern Recognition, Computer Science Department,
RWTH Aachen University, 52056 Aachen, Germany
[2]Spoken Language Processing Group, LIMSI CNRS, Paris, France
{tuske,golik,schlueter,ney}@cs.rwth-aachen.de

E. Loweimi

Contributions

- Raw waveform modelling on LVCSR tasks
- Comparison with many conventional featurs
  - MFCC, FBANK, PLP, GT and |FFT|
- MLP for acoustic modelling
- Investigation of data amount and activation function roles
- Interpretation of learned filters

Experimental Setup

- Training data from *Quaero* project
  - Training set: 50 and 250h, DevSet and Testset 3.5h, each
- Language model: 4-gram
- DNN: MLP with 6 hidden layers (2k units) → 30-35M parameters
- DNN initialisation: Discriminative pre-training (DPT)
- DNN input feature: 17 stacked frames ~ 185 ms
- Toolkit: ***RASR***
- Baseline: GMM-HMM → 30M trainable-parameters
- MFCC per frame: LDA on 9 consecutive frames → 45 features per frame
  - **MVN** norm

Experimental Results – 50h

- DNN is better than GMM
  - ~ 20 %, relative
- Raw → significantly worse
  - ~ 16-20% relative to GMM
  - ~ 45-50% relative to DNN

50h training data

Features	model	dev	eval
MFCC	GMM	24.4	31.6
MFCC	DNN	19.4	25.3
time signal	DNN	29.4	36.8

Effect of Each Stage of MFCC (Relative)

- Best feature: MFCC; Worst: Raw
- MFCC utterance norm → RWERR*<2.2%
  - VTLN → < 1%
- MFCC-dim.: 16 → 20 => RWERR < 1.5%
- CRBE slightly worse than MFCC
  - 20 filters is better than 40!
- |FFT| → worse than MFCC and CRBE
  - $|.|^{0.1}$ slightly helps despite huge statistical effect!
- Utter-norm is better than global norm

Features	dim.	dev	eval
MFCC	16		
+ global norm.		19.8	26.1
+ utterance norm.		19.7	25.5
+ VTLN		19.4	25.3
MFCC	20		
+ VTLN + utterance norm.		19.1	25.2
CRBE			
+ VTLN + utterance norm.	20	19.5	25.7
	40	19.7	26.2
|FFT|	257		
+ global norm.		21.3	27.8
+ 10th root		21.0	27.5
+ utterance norm.			
+ 10th root		20.6	26.8
time signal	160		
+ global norm.		29.4	36.8
+ utterance norm.		28.9	35.0

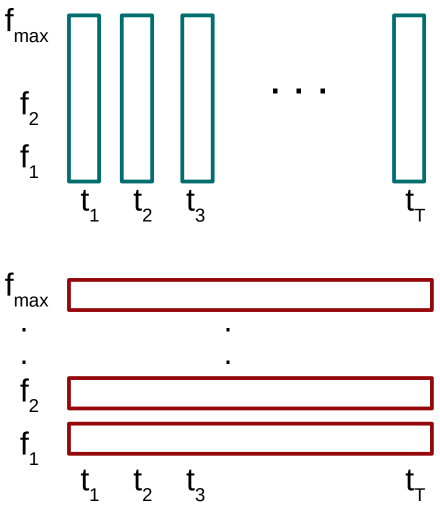
* RWERR: Relative WER Reduction

MFCC vs PLP and GT

- Performance on Dev
  - MFCC, PLP = GT*
- Performance on Eval
  - PLP, MFCC, GT
- Feature combination is helpful
  - Info redundancy or complementary
    - MFCC/PLP are FT-based, GT is not

DNN + VTLN + Utter-level norm

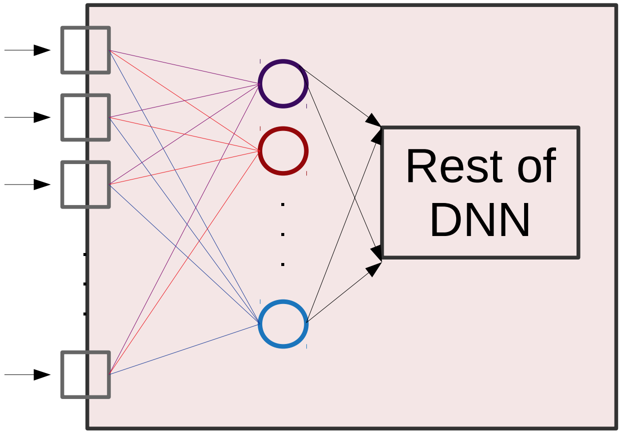
Features	dev	eval
MFCC	19.1	25.2
PLP	19.2	24.8
GT	19.2	25.5
MFCC + PLP + GT	18.4	24.2

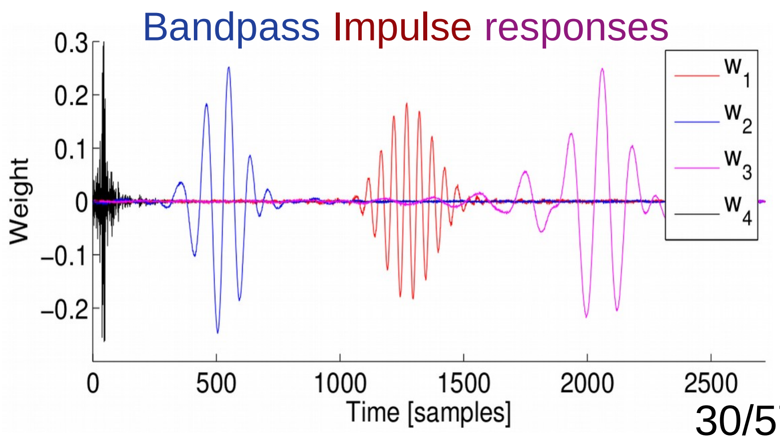


GT*: GammaTone Feature, Schulter et al, 2007

Interpretation of First layer’s Weights

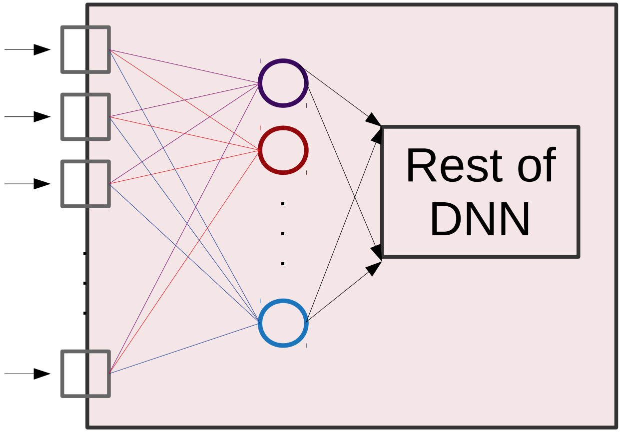
- First layer → filterbank → time-frequency analysis
- Filters aren’t symmetric & centred
- Some filters are just **shifted** replica

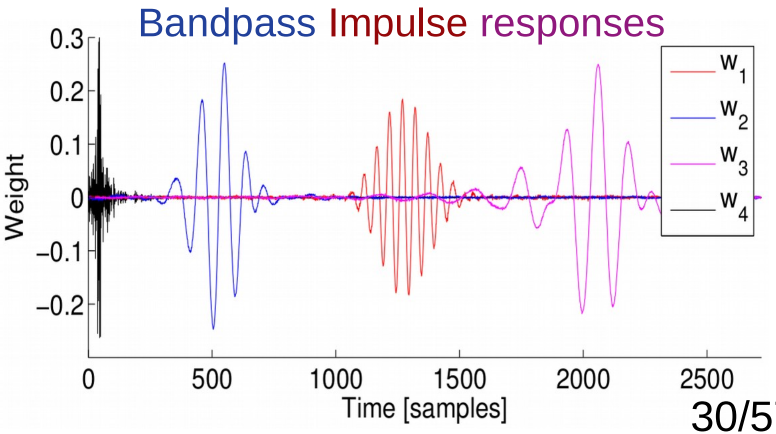




Interpretation of First layer's Weights

- First layer  ank → time-frequency a
- Filters aren' etric & centred
- Some filters are just **shifted** replica
- **Interpretation**
  - Similar magnitude, different phase
  - Emulate shifting in CNNs!
    - CNNs do not shifted replica! ===>>> fewer parameters





Interpretation of First layer's Weights

- Sort the filters based on f_c
  - $f_c \approx$ argmax of freq response (W_i)
    - g → Gaussian kernel → low-pass filter
- Compute the bandwidth (f_b)
  - Using Noise Equivalent Bandwidth

$$y_i = \boldsymbol{w}_i^T \ \boldsymbol{x}$$

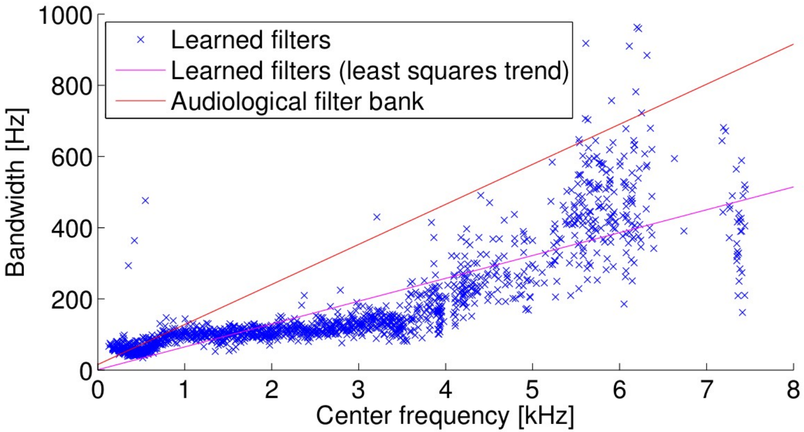
$$\boldsymbol{W}_i = |FFT\{w_{i,:}\}|$$

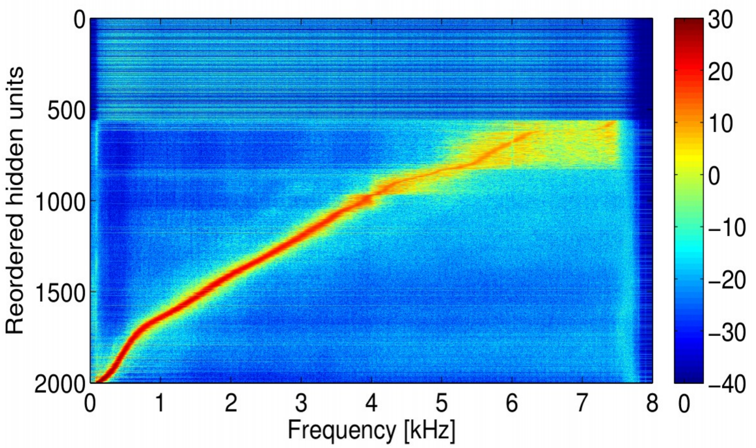
$$\hat{\boldsymbol{W}}_i = \boldsymbol{W}_i * g$$

$$f_c^i = \underset{j}{\operatorname{argmax}} \hat{\boldsymbol{W}}_{i,j}$$

$$f_b^i = \frac{\sum_j W_{i,j}^2}{\max_j W_{i,j}^2}$$

Interpretation of First layer's Weights





* Filters are unimodal, bandpass and narrow

* Filters' Bandwidth ...

– Varies between 100-1000 Hz

– Trend-wise increases by centre freq

Using ReLU instead of Sigmoid

- Relative gain on Eval Set
  - MFCC → 5.5%
  - Combination → 10%
  - |FFT| → 7.8%
  - **Raw → 18.6%**

Features	dev		eval	
	sigmoid	ReLU	sigmoid	ReLU
MFCC	19.1	18.0	25.2	23.8
MFCC + PLP + GT	18.4	16.6	24.2	21.7
|FFT|	20.6	18.4	26.8	24.7
time signal	28.9	22.6	35.0	28.5

Relative Gain on Eval
- MFCC → 5.5%;
- Combination → 10%
- |FFT| → 7.8%
- Raw → 18.6%

- Highest relative gain for Raw, Why?
  - ReLU’s sparsity is good for high dim(?)
  - |FFT| is high-dim, but MFCC level gain!

Relative gain of ReLU for 50 & 250h

- ReLU gain* for 50 & 250
  - MFCC → 5.5 vs -3.4%
  - Combination → 10.3 vs 4.5%
  - |FFT| → 7.8 vs +0%
  - Raw → 18.6 vs 8.2%

- ReLU is less useful when more data is available

50h

Features	dev		eval	
	sigmoid	ReLU	sigmoid	ReLU
MFCC	19.1	18.0	25.2	23.8
MFCC + PLP + GT	18.4	16.6	24.2	21.7
|FFT|	20.6	18.4	26.8	24.7
time signal	28.9	22.6	35.0	28.5

250h

Features	dev		eval	
	sigmoid	ReLU	sigmoid	ReLU
MFCC	15.2	15.9	20.4	21.1
MFCC + PLP + GT	14.8	14.0	19.8	18.9
|FFT|	16.1	15.8	21.6	21.5
time signal	19.2	17.6	25.6	23.5

* relative to sigmoid

Relative gain of 250h for Sigm & ReLU

- 50 → 250 relative gain; Sigm vs ReLU
  - MFCC → 19.0, 11.3
  - Combination → 18.2, 12.9
  - |FFT| → 19.4, 13.0
  - Raw → 26.9, 17.5
- Sigmoid further benefits from data
- Data amount is more important than activation function

50h

Features	dev		eval	
	sigmoid	ReLU	sigmoid	ReLU
MFCC	19.1	18.0	25.2	23.8
MFCC + PLP + GT	18.4	16.6	24.2	21.7
|FFT|	20.6	18.4	26.8	24.7
time signal	28.9	22.6	35.0	28.5

250h

Features	dev		eval	
	sigmoid	ReLU	sigmoid	ReLU
MFCC	15.2	15.9	20.4	21.1
MFCC + PLP + GT	14.8	14.0	19.8	18.9
|FFT|	16.1	15.8	21.6	21.5
time signal	19.2	17.6	25.6	23.5

Performance Gap: Raw vs MFCC

- Performance gap = $\text{WER}_{\text{RAW}} - \text{WER}_{\text{MFCC}}$
  - 50h, Sigmoid → 9.8%
  - 50h, ReLU → 4.7%
  - 250h, Sigmoid → 5.2%
  - 250h, ReLU → 2.4%

- Applying ReLU halves the gap (relative to sigmoid)
- Using 5x more data [50→250h] halves the gap

Effect of First layer Initialisation

- Initialisation with GT filters
  - 32 filters (i) + shifted copies
  - No update → fixed first layer
- Init. with GT has almost no effect
  - Note: Filters are not learned as parametric (f_c, BW) models!
- Fixing first layer worsen the results
  - Eval → 2.4% abs, 8.4% rel

Weight initialization	update allowed	dev	eval
random	yes	22.6	28.5
GT	yes	22.4	28.7
	no	24.9	31.1

$$f_c^i = 228.85(\exp(i/9.265) - 1)$$

$$f_b^i = 24.7 + \frac{f_c^i}{9.265}$$

$$i = 1, 2, ..., 32$$

$$\text{order} = 4$$

$$h^i(t) = t^{n-1}\exp(-2\pi f_b^i t)\cos(2\pi f_c^i t + \phi_i)$$

Convolutional Neural Networks for Acoustic Modeling of Raw Time Signal in LVCSR

Pavel Golik[1], *Zoltán Tüske*[1], *Ralf Schlüter*[1], *Hermann Ney*[1,2]

[1]Human Language Technology and Pattern Recognition, Computer Science Department, RWTH Aachen University, 52056 Aachen, Germany
[2]Spoken Language Processing Group, LIMSI CNRS, Paris, France

{golik,tuske,schlueter,ney}@cs.rwth-aachen.de

Lehrstuhl Informatik 6
Human Language Technology and Pattern Recognition

INTERSPEECH 2015
September 6 - 10
Dresden, Germany

Contributions

- First layer → bank of bandpass filters → time-freq analysis
- Performance gap between raw and MFCC reduced by using ReLU and more training data

- This paper
  - Replacing MLP with CNN
  - Further interpretation of learned filters

Experimental Setup

- Training data: Quaero, English, train11
- Dev and Eval sets: 3.5h
- LM: 4-gram
- Random initialisation + layer-wise discriminative pre-training
- MFCC → 45 dim, LDA on 9 consecutive frames
- Toolkit: RASR
- HMM-GMM parameters: 30M
- DNN input, 2000 ReLU unites per layer
  - Baseline: MFCC of 17 stacked frames
  - Raw: 10ms x 17 => 170 ms, frame shift = 10
- Conv layer: kernel len: 256 samples (16 ms), stride: 31 samples (2 ms), #channels = 128

Experimental Results

- Baseline
  - MFCC: GMM → DNN [30%]
  - Raw: 9 → 12 layers
    - RWERR: Dev 1.9%, Eval 3%

Features	model	# hidden layers	dev	eval
MFCC	GMM	-	24.4	31.6
	DNN	9	16.9	22.1
time signal	DNN	9	20.7	26.3
	DNN	12	20.3	25.5

- 1Conv layer + MLP
  - **1Conv+MLP-5L ≈ MLP-12L**
    - [WER] A conv layer ≡ 7 MLP layers
  - 1Conv+MLP-12L → rel. gain 5.9%

1Conv+ MLP-xL

	Fully connected layers						
	5	7	8	9	10	11	12
dev	20.3	19.5	19.1	18.9	18.6	18.7	18.5
eval	25.6	25.1	24.3	24.3	24.1	23.9	24.0

Conv Layer Hyperparameters Effect

- Conv. kernel length (k) is not a critical choice
  - 8-64ms filters → similar WER
    - No need to multi-resolution!

1CNN + MLP-10L

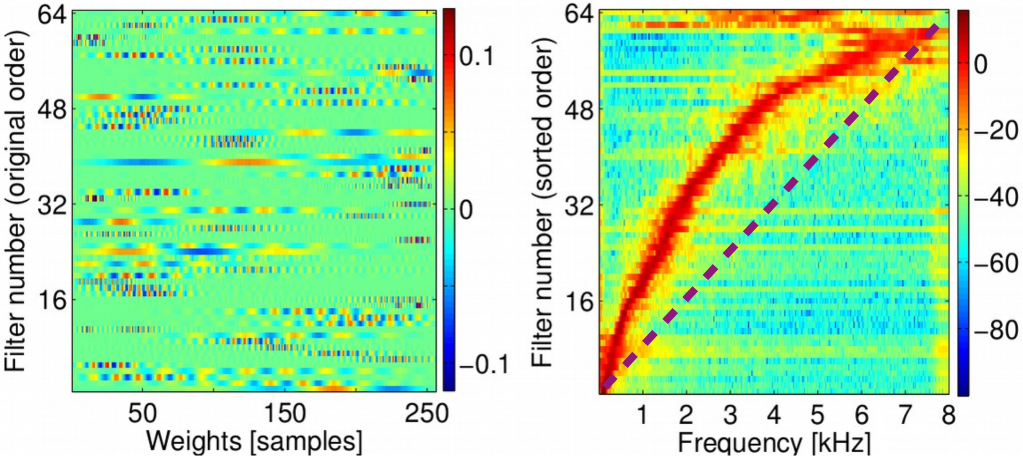
	Filter length k in samples			
	128	256	512	1024
dev	18.8	18.6	18.8	18.9
eval	24.2	24.1	24.1	24.3

- Adding second conv layer slightly improves the results; ~ 2% relative

number of hidden layers		WER [%]	
convolutional	fully connected	dev	eval
1	10	18.6	24.1
	11	18.7	23.9
	12	18.5	24.0
2	10	18.3	23.6
	11	18.2	23.4

Learned Filters of Conv-L1

- A bank of bandpass(?) filters
  - Time-frequency analysis
- Some similarity to auditory filters
  - More filters in low frequencies
    - f_c is a sub-linear function of (sorted) filter index
    - 78% of filters are below 4 kHz
  - BW increases with f_c, trend-wise

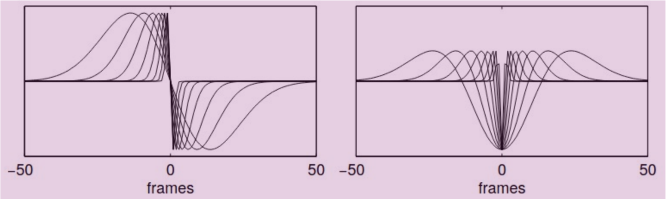


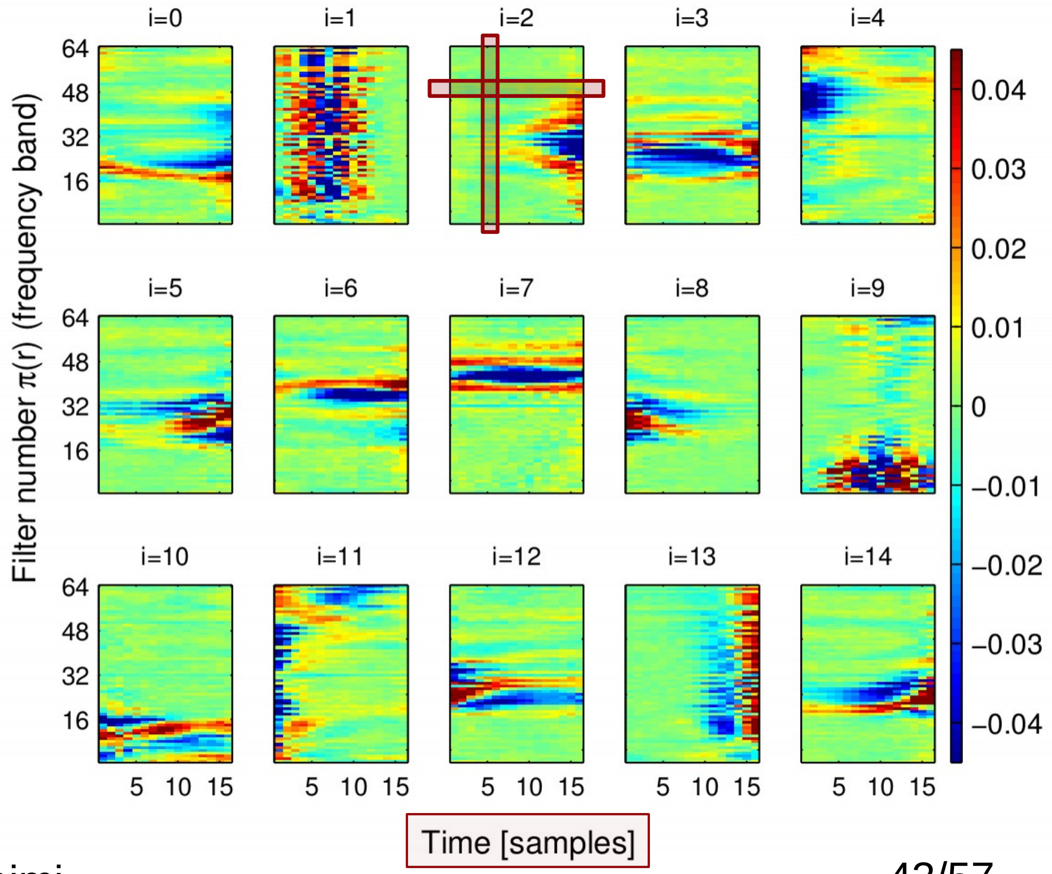
Time

Estimated centre freq and bw in frequency domain

Learned Filters of Conv-L2

- Recognisable Patterns
  - *Non-stationary* (?) in both directions → 0,8,10,14
  - Time-invariant → 7
  - Matched filter
  - MRASTA and Gaussian-like → 1,9,13





ACOUSTIC MODELING OF SPEECH WAVEFORM BASED ON MULTI-RESOLUTION, NEURAL NETWORK SIGNAL PROCESSING

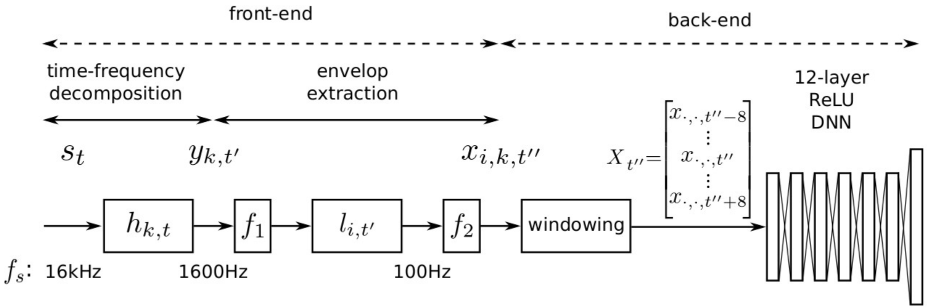
Zoltán Tüske, Ralf Schlüter, Hermann Ney

Human Language Technology and Pattern Recognition, Computer Science Department,
RWTH Aachen University, 52056 Aachen, Germany

{tuske, schlueter, ney}@cs.rwth-aachen.de

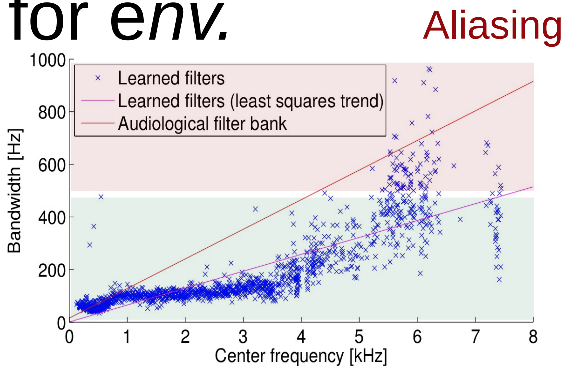
Contributions

- Generalising the downsampling and env-extractor block (Max-Pooling) and make it trainable
  - EnvExtractor: Rectifier + Low-pass filter
- Learning multi-resolution spectral representation
  - Time-freq analysis with multiple spectro-temporal resolutions



Max-Pooling

- Performs *subsampling* and Lowpass filter for *env. extraction*

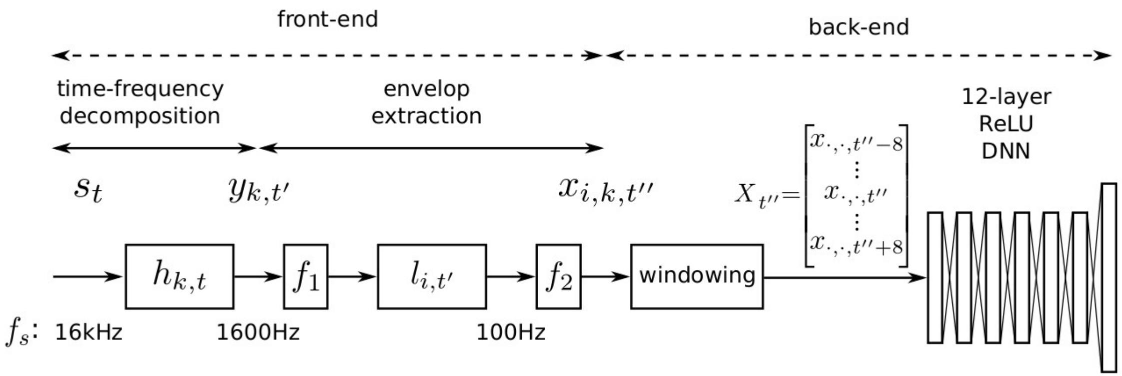


- Subsampling could lead to aliasing
  - Baseband vs bandpass sampling
    - Sampling under Nyquist rate is possible for bandpass signals
  - E.g. for 1ms stride → sampling rate [*Approximately*] is 1 kHz
    - Undersampling for BW > 500 Hz
    - Oversampling for BW < 500 Hz

Multi-resolution Signal Process. via NNs

1) $h_{k,t}$: Impulse response of k^{th} FIR filter with N_{TF} taps (1D-ConvLayer1) → Time-frequency (TF) analysis
   - Shared over time, similar to TDNN
2) Stride by 10 samples
   – Subsampling by factor 10 ($t=10t'$)
3) f_1: half (ReLU) or full (Abs)
4) $l_{i,t'}$: Impulse response of i^{th} FIR filters with N_{ENV} taps (1D-ConvLayer2)
   – Trainable env-extractor + Multi-Resolution Proc.
   – Shared over time and TF filters
5) Stride by 16 samples
   – Subsampling by factor 16 ($t=160\,t''$)
6) f_2: Rectification + non-linearity (log/root)
   – Its output, $x_{k,i,t''}$ interpretable as CRBE
7) Windowing → Feature-dim: K x L x (2x8+1)

Context length

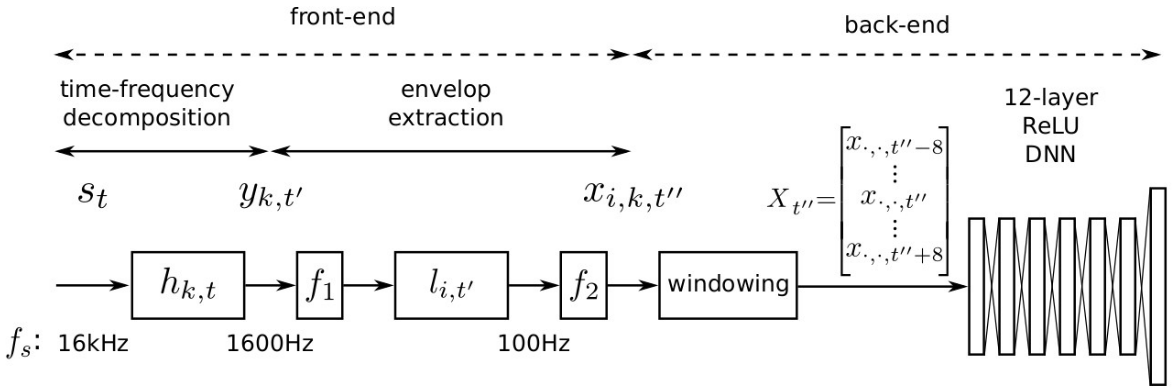


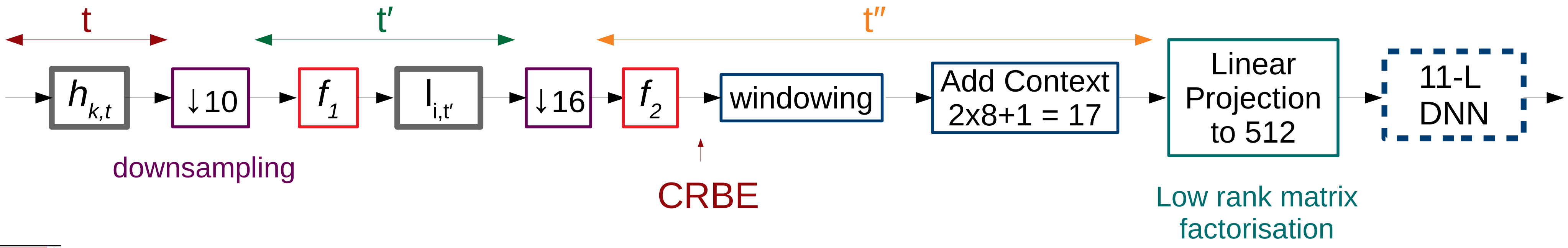
$$y_{k,t'} = s_t * h'_{k,t} \overset{\text{FIR}}{=} \sum_{\tau=0}^{N_{TF}-1} s_{t+\tau-N_{TF}+1} \cdot h_{k,\tau}$$

$$x_{i,k,t''} \overset{\text{FIR}}{=} f_2\left(\sum_{\tau=0}^{N_{ENV}-1} f_1(y_{k,t'+\tau-N_{ENV}+1}) \cdot l_{i,t}\right)$$

(Mirrored and) Shifted

Pipeline


front-end
back-end
time-frequency decomposition
envelop extraction
12-layer ReLU DNN
s_t
$y_{k,t'}$
$x_{i,k,t''}$
$X_{t''} = \begin{bmatrix} x_{\cdot,\cdot,t''-8} \\ \vdots \\ x_{\cdot,\cdot,t''} \\ \vdots \\ x_{\cdot,\cdot,t''+8} \end{bmatrix}$
$h_{k,t}$
f_1
$l_{i,t'}$
f_2
windowing
f_s: 16kHz
1600Hz
100Hz


t
t′
t″
$h_{k,t}$
↓10
f_1
$l_{i,t'}$
↓16
f_2
windowing
Add Context 2x8+1 = 17
Linear Projection to 512
11-L DNN
downsampling
CRBE
Low rank matrix factorisation

Experimental Setup

- Training: CE, SGD+Momentum, l_2 reg., discriminative pre-training
- TF filters: 150 filters with N_{TF}=512 samples (32 ms)
- $l_{i,t'}$ $\rightarrow$ 16 < N_{ENV} < 40 samples, 5 < #filters [denoted by *max(i)*!!!] < 20

- Training data: 250h, Dev and Eval: 3h each
- Toolkit: RASR
- Architecture: CNN front-end + 12-layer MLP with 2000 ReLU units
- Low-rank linear factorisation at the first layer to 512
  - Feature-dim = K x L x M, e.g. 150 x 20 x 17 = 51000
  - K: #filters@L1, L: #filters@L2, M: Context_Len

Experimental Setup – Single EnvExt

- GT is better than Raw
- Second Conv-L is useful
- Trainable env-extractor is as effective as Max-pool
- N_{ENV} is not a critical param.
  - *Overlapping max-pooling* for N_{ENV} > 16 (Stride@L2 is 16)
    - No significant effect

$l_{i,t}$ type	N_{ENV}	WER dev	WER eval
max	16	14.4	19.9
	25	14.3	19.8
	40	14.4	19.7
FIR	40	14.1	19.8
Gammatone		13.5	18.4
time-signal DNN		15.1	20.5

– #TF=50, f_1=| . |, f_2=| . $|^{0.4}$
– Single envelope detector
– time-signal DNN → using only one conv layer

Experimental Setup – Multiple EnvExt

- *#TF: 50→150, WER: 19.8 → 19.3*
- *Abs* is slightly better than *ReLU*
- N_{ENV} is not a critical param., but should not be too large (e.g. 160)
- Root comp. helpful when $N_{ENV} \leq 40$
- Optimal setup
  - $\max(i)=5$, $N_{ENV}=40$, $f_1=|.|$, $f_2=|.|^{0.4}$
    - For GT features is 0.1

$\max(i)$	N_{ENV}	f_1	f_2	WER	
				dev	eval
5	40	ReLU	-	14.2	19.6
		ReLU	ReLU	14.2	19.5
		ReLU	ReLU+root	14.0	19.2
		Abs	-	14.2	19.6
		Abs	Abs	14.2	19.3
		Abs	Abs+root	**13.7**	**18.7**
		Abs+root	Abs	13.8	**18.7**
10	80	Abs	Abs	13.9	19.0
		Abs+root	Abs	13.8	19.0
20	160	Abs	Abs	14.3	19.3
		Abs	Abs+root	14.4	19.6

– #TF: 150
– max(i): #filters of Conv-L2

Transfer Learning + MVN

- Front-end learned for MLP and fixed
- MVN on segment level
  - Fix learned front-end → Dump & normalise features → Learn back-end again (MLP & LSTM)
- NN_1: features + context (17) → low-rank factorization layer
- NN_2: ONLY $x_{i,k,t''}$ w/o context [LSTM]

front-end type	front-end dim.	back-end	normalization mean	normalization variance	WER [%] dev	WER [%] eval
NN_1	512	MLP			13.7	18.7
			×	×	13.5	18.5
GT	70		×		**13.1**	**17.8**
NN_1	512	LSTM			14.5	18.7
			×		14.5	19.1
			×	×	13.0	16.8
NN_2	750		×		13.0	17.1
			×	×	13.9	18.1
GT	70		×	×	11.3	**14.5**
			×		11.6	14.9
					11.2	14.6

Baseline, first raw, best system from previous table max(i)=5, N_{ENV}=40, f_1=|.|, f_2=$|.|^{0.4}$

NN2 → 750 = 150 (k) x 5 (L)

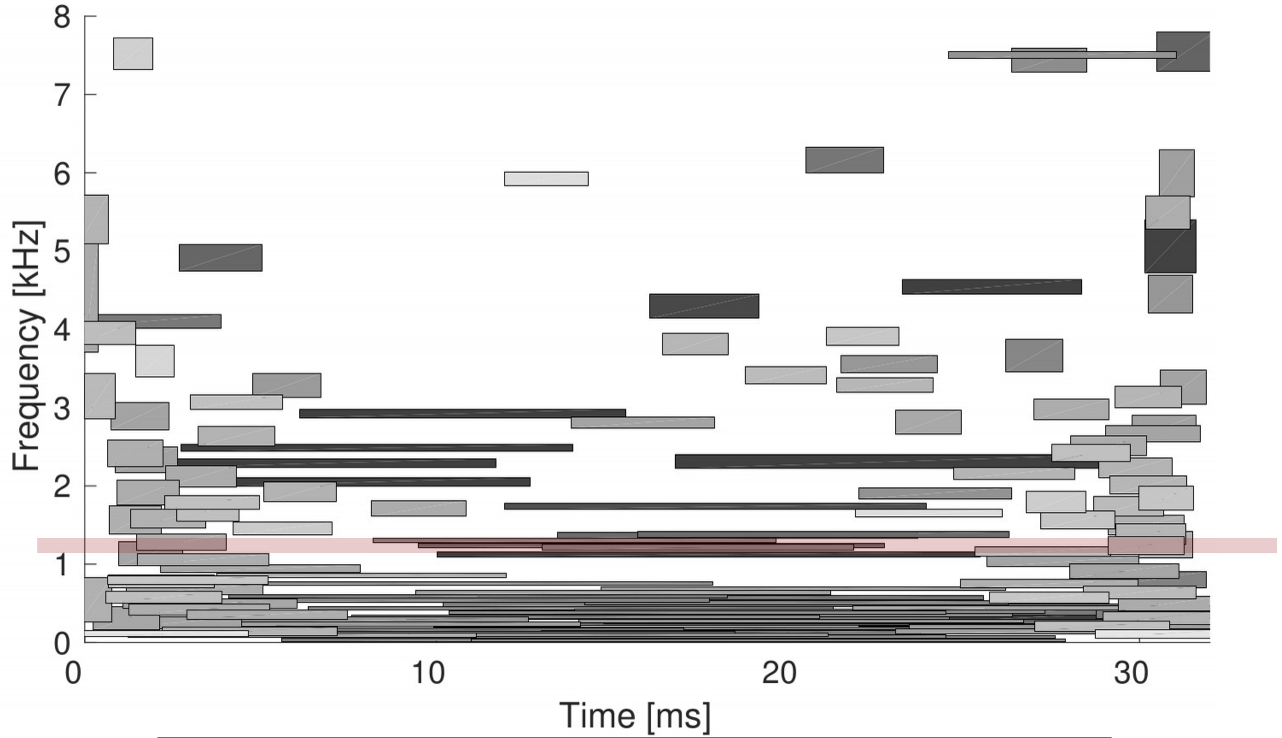
Transfer Learning + MVN

- GT is better than raw
- LSTM for NN_1
  - MN worsen, MVN improves
  - Front-end is learned based on MLP and is fixed (Mismatch!)
- Using LSTM is useful for NN_2 & GT
  - Both are w/o context
    - LSTM takes care of context!

front-end		back-end	normalization		WER [%]	
type	dim.		mean	variance	dev	eval
NN_1	512	MLP			13.7	18.7
			×	×	13.5	18.5
GT	70		×		**13.1**	**17.8**
NN_1	512	LSTM			14.5	18.7
			×		14.5	19.1
			×	×	13.0	16.8
NN_2	750		×		13.0	17.1
			×	×	13.9	18.1
GT	70		×	×	11.3	**14.5**
			×		11.6	14.9
					11.2	14.6

Multi-Resolution Processing in TF Stage

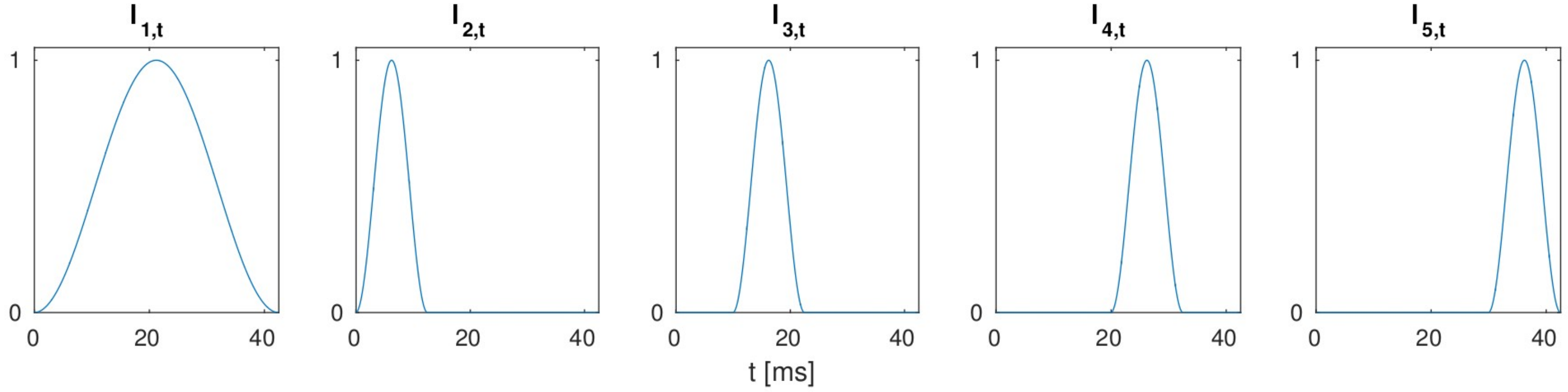
Fc, BW (in freq)
and
Pulse centre and
duration (in time)
are estimated

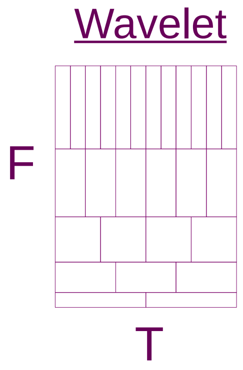


– 150 filters ($h_{k,t}$)
– f_c and BW est.
– grayscale prop with amp

Each subband is covered by band-pass filters with different BWs

Multi-Resolution Processing in Conv-L2



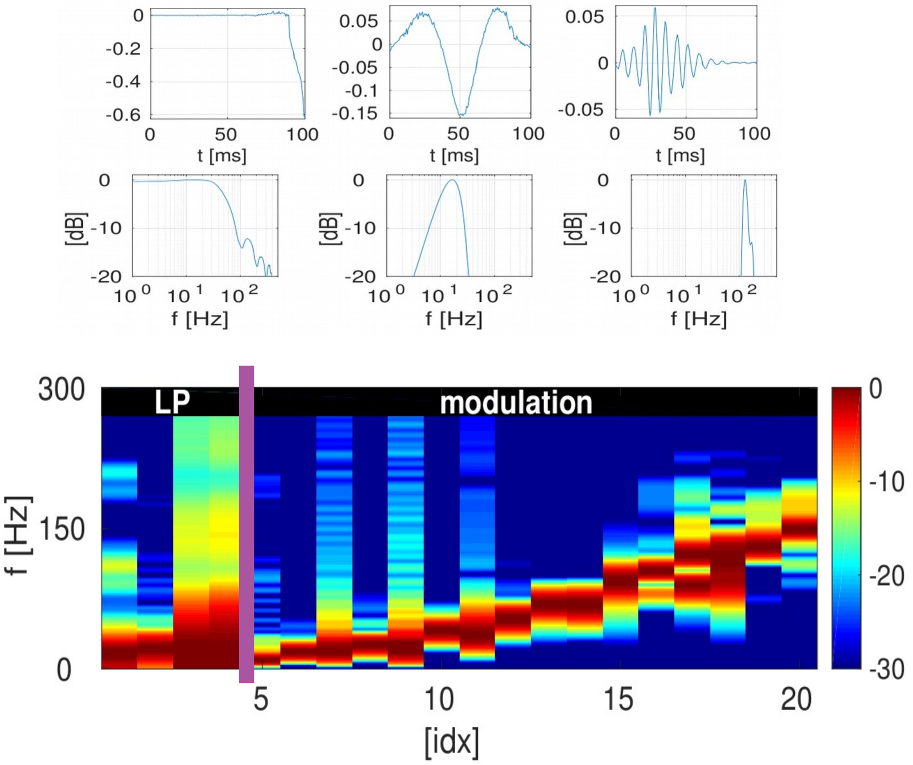


- These are hypothetical, not learned, filters impulse responses. Wavelet-like processing
  - l_1: deals with slowly varying components (low freq)
  - $l_{2:5}$: deals with faster varying components + localisation

Envelope Detection Filters

- Learned $l_{i,t}$s are Lowpass (LP) and bandpass (Modulation) filters!
  - Separated based on energy@0Hz
- Filters sorted based on the highest 3 dB cut-off frequency (not argmax)
- Here, Modulation frequency range is 0-200 Hz
  - 1-50Hz covers the modulation content of speech signal [33]

max(i) = 20



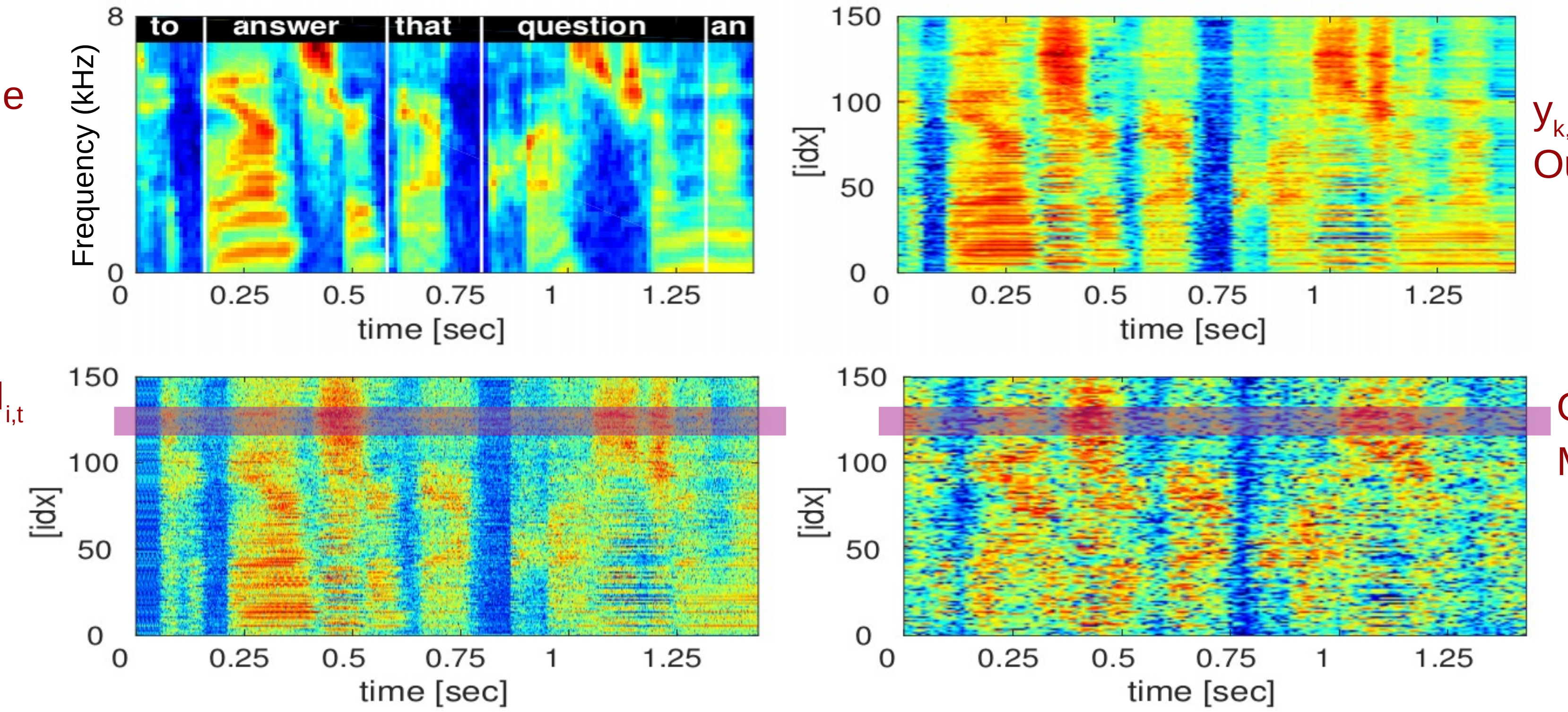
CRBEs

Gammatone

$y_{k,t'}$
Output of $h_{k,t}$

Output of $l_{i,t}$
LP part

Output of $l_{i,t}$
Mod, 40Hz



Static over time axis

Dynamic over time axis

Experimental Results – CHiME4

Features	Fine-tuned	Beamformed	Dev Bus	Dev Caf	Dev Ped	Dev Str	Dev **Avg.**	Eval Bus	Eval Caf	Eval Ped	Eval Str	Eval **Avg.**
MFCC	-	-	19.7	14.2	9.7	13.7	**14.3**	30.4	25.2	20.2	16.6	**23.1**
		+	12.2	10.9	8.6	10.6	**10.6**	19.0	16.6	14.7	13.8	**16.0**
	+	-	19.4	13.6	9.1	14.4	**14.1**	32.2	25.7	21.1	17.9	**24.2**
		+	10.4	9.1	6.6	9.2	**8.8**	17.0	14.2	12.4	12.2	**13.9**
raw TS	-	-	30.7	19.9	17.5	21.3	**22.4**	63.6	38.6	34.0	25.0	**40.3**
		+	30.1	15.6	16.1	17.9	**19.9**	49.9	27.4	24.3	23.6	**31.3**
	+	-	27.8	27.3	19.0	24.9	**24.7**	51.9	48.9	39.8	26.1	**41.7**
		+	14.8	11.8	8.8	11.7	**11.8**	29.5	21.5	18.6	16.3	**21.5**

- MFCC outperforms with a significant margin on this task
  - 40% Relative lower WER

T. Menne, Z. Tüske, R. Schlüter, and H. Ney. *Learning Acoustic Features from the Raw Waveform for Automatic Speech Recognition, 2018*

Part I – Conclusion

- Conventional features are still better
- Architecture is important (CNN rather than MLP)
- Data amount and activation function can narrow the gap
- Interpretability
  - First layer → time-frequency analysis
  - Second layer → modulation spectrum processing
  - Filters resemble auditory filters
    - More filters in low frequency, wider filters in high frequencies

End of Part I

- Thanks for Your Attention!
- Q/A

- Part II (Next week): Google + Multi-Resolution
- Part III: Parametric CNNs