Raw Waveform Modelling for ASR

A Literature Review

Part II

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 which 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
  - Employ all signal information
  - Learning basis functions
  - Mid-term processing rather than short-term processing
  - No need to <u>exact</u> alignment

Acoustic Modelling using Raw Waveform – Challenges

- High dimensional feature
  - Discriminative models, CNN, matrix factorisation

- Discard prior knowledge about auditory system
  - Initialise first layer using perceptual scales

Our Plan ...

- Part **I** → IDIAP + AACHEN
- Part **II** → Baidu + JHU + Cambridge + Google
- Part **III** → Google + Parametric CNNs

Part I – Summary

- 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 freq, wider filters in high frequencies (trend-wise)

Learning Multiscale Features Directly From Waveforms

*Zhenyao Zhu** [1,2], *Jesse H. Engel** [1], *Awni Hannun*[1]

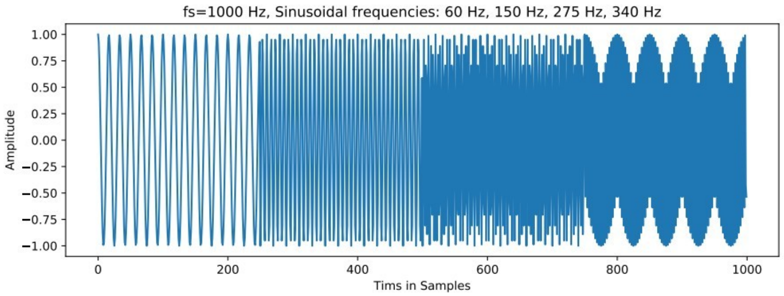
[1]Baidu Silicon Valley AI Lab (SVAIL)
[2]The Chinese University of Hong Kong

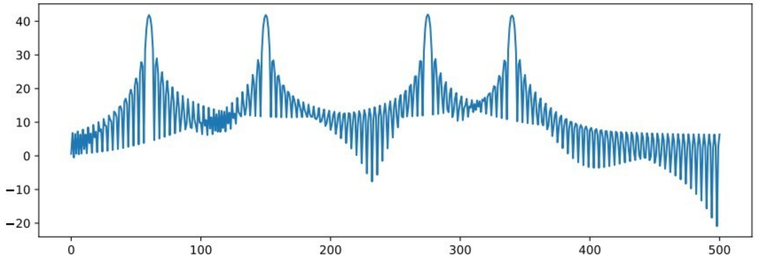
* Authors contributed equally to this work

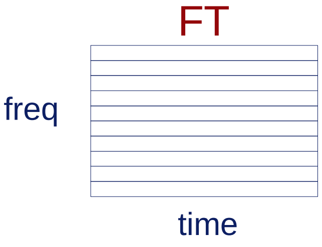
zhuzychn@gmail.com, jengel@baidu.com, awnihannun@baidu.com

E. Loweimi

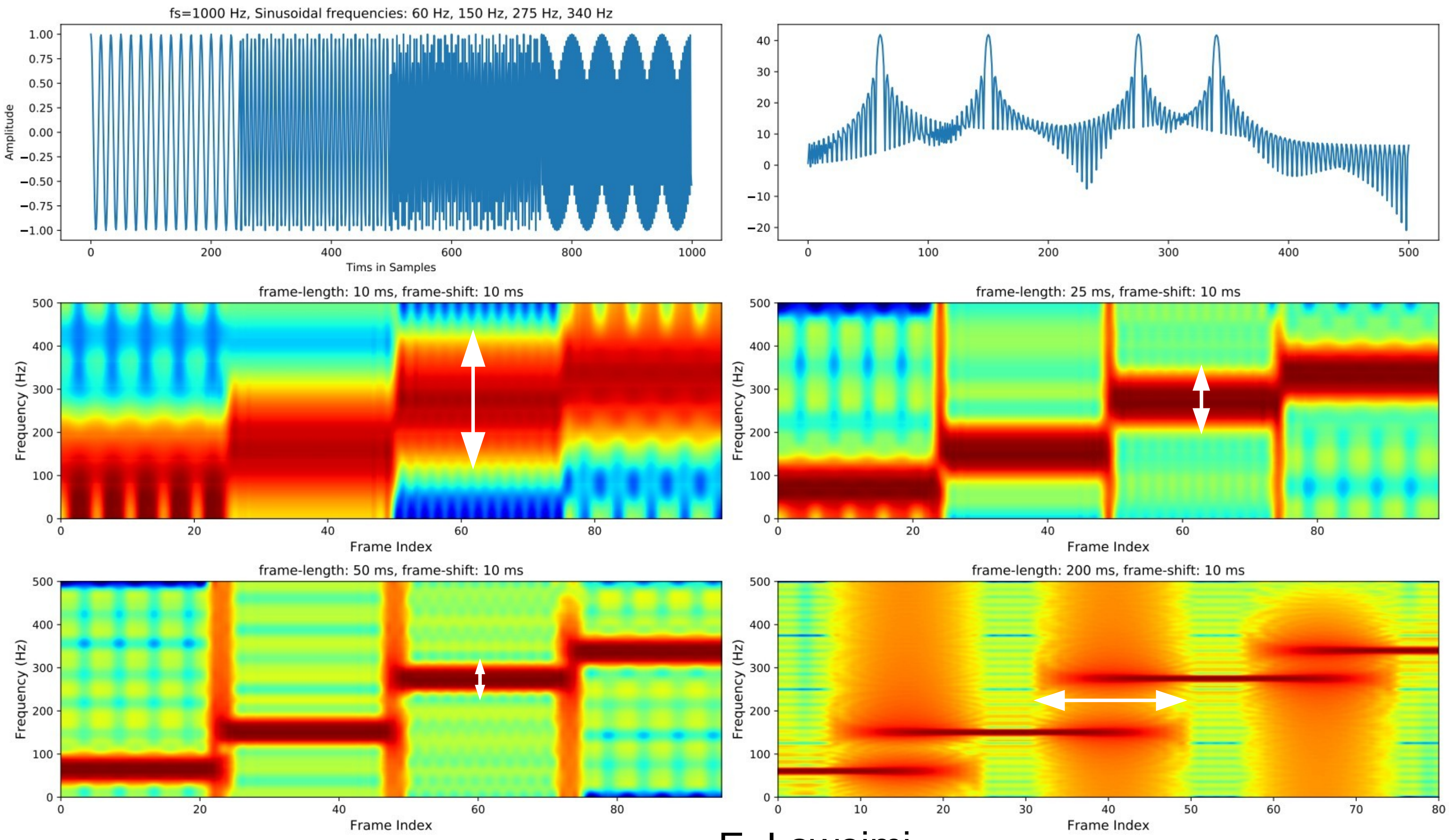
Time-Frequency Analysis

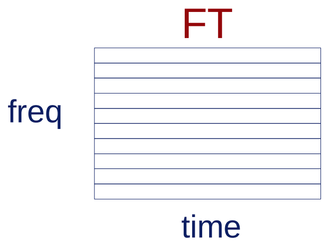


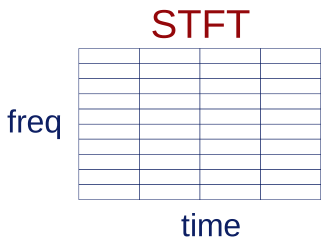




Time-Frequency Analysis

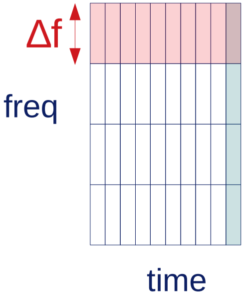


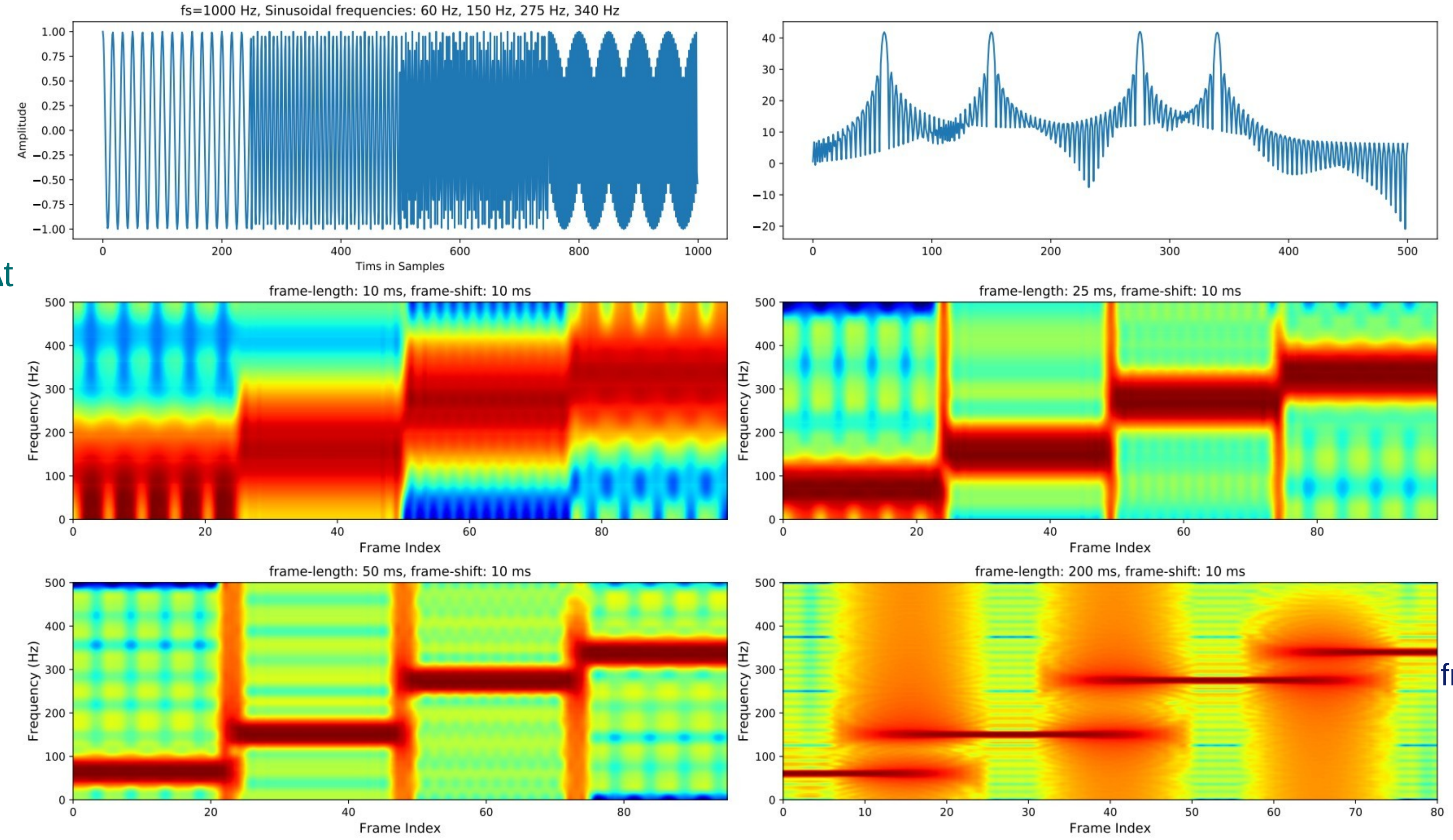




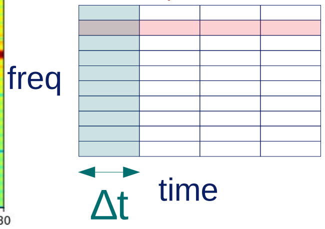
Time-Frequency Resolution **Trade-off**

STFT, small Δt





STFT, small Δf



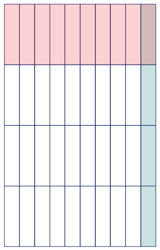
E. Loweimi

Time-Frequency Resolution **Trade-off**

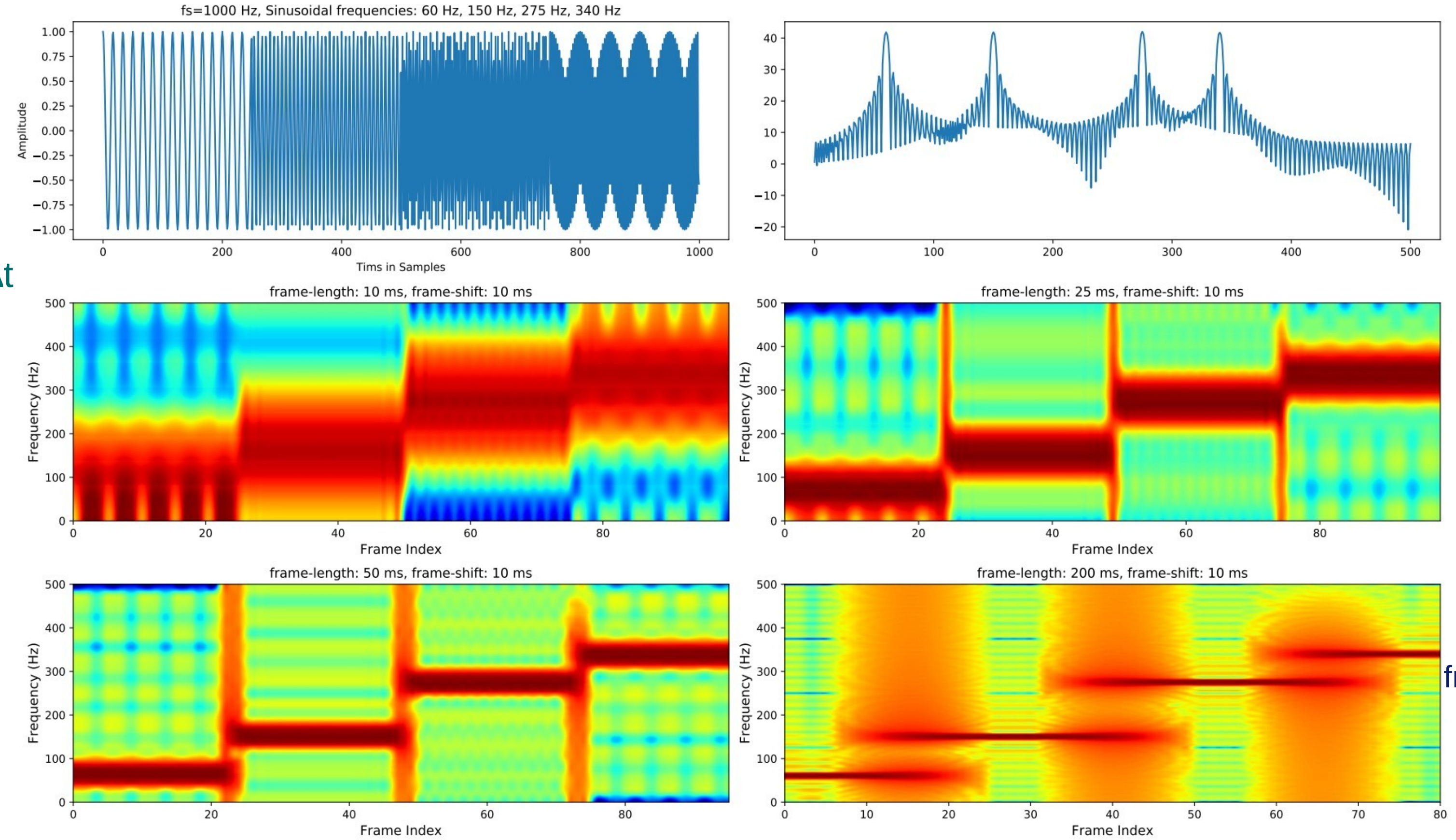
STFT, small Δt

Δf

freq



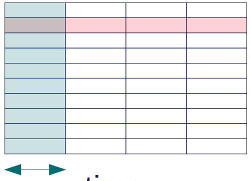
time



Change inside cell is not detectable.

STFT, small Δf

freq



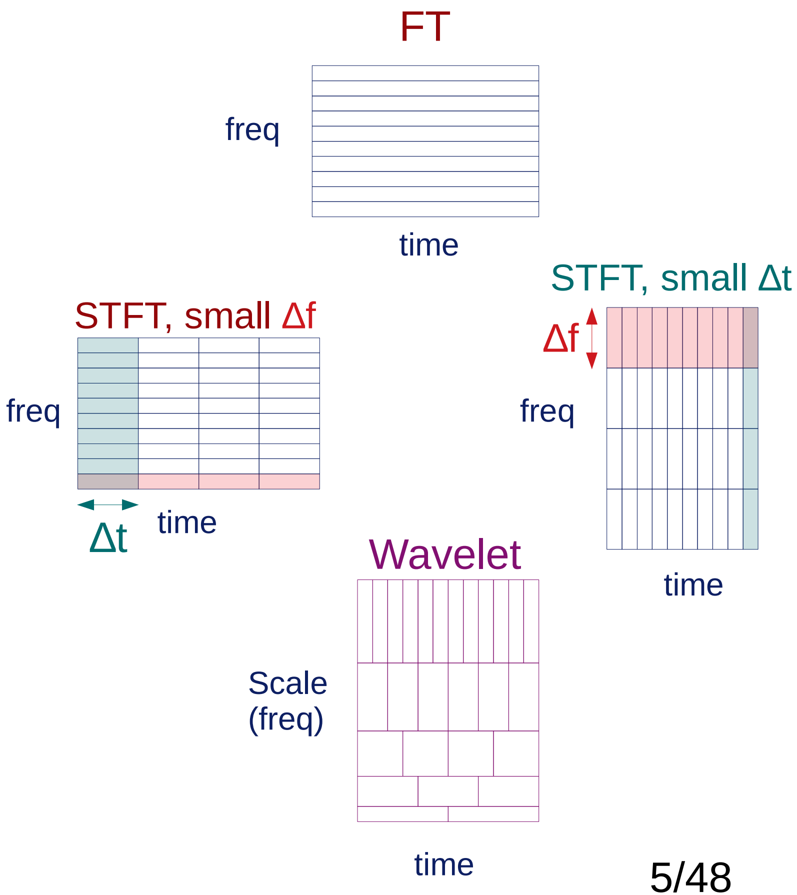
Δt

time

E. Loweimi

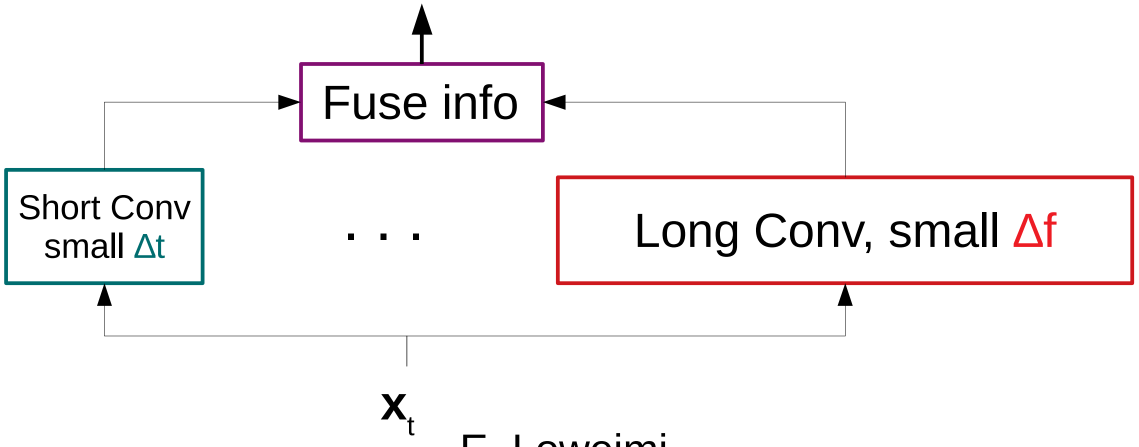
Time-Frequency Resolution Trade-off

- Gabor uncertainty principle: $\Delta t \Delta f \geq 1/4\pi$
  - Δt/Δf: uncertainty in temporal/spectral localisation
  - Trade-off → ↓Δf *necessarily* means ↑Δt & vv
  - Lower uncertainty ≡ higher resolution
  - X-resolution: localisation accuracy@x-domain
- Longer filter/window in time domain
  - Larger Δt and necessarily smaller Δf
- STFT → uniform resolution allocation
- Wavelet → non-uniform res. allocation
  - Smaller Δt for higher frequencies & vv



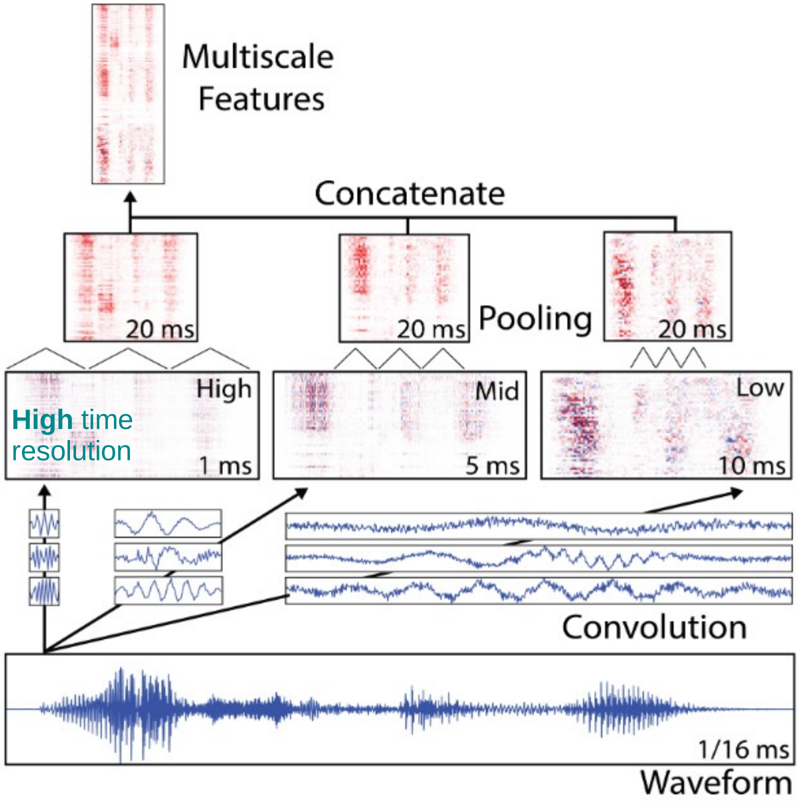
Can we improve BOTH Δf & Δt?

- IMPOSSIBLE in a single Conv Layer … BUT ...
- … What about parallel CNNs with different filter lengths?
  - Fuse info from representations with small Δt & Δf
  - Cost: more memory and computation



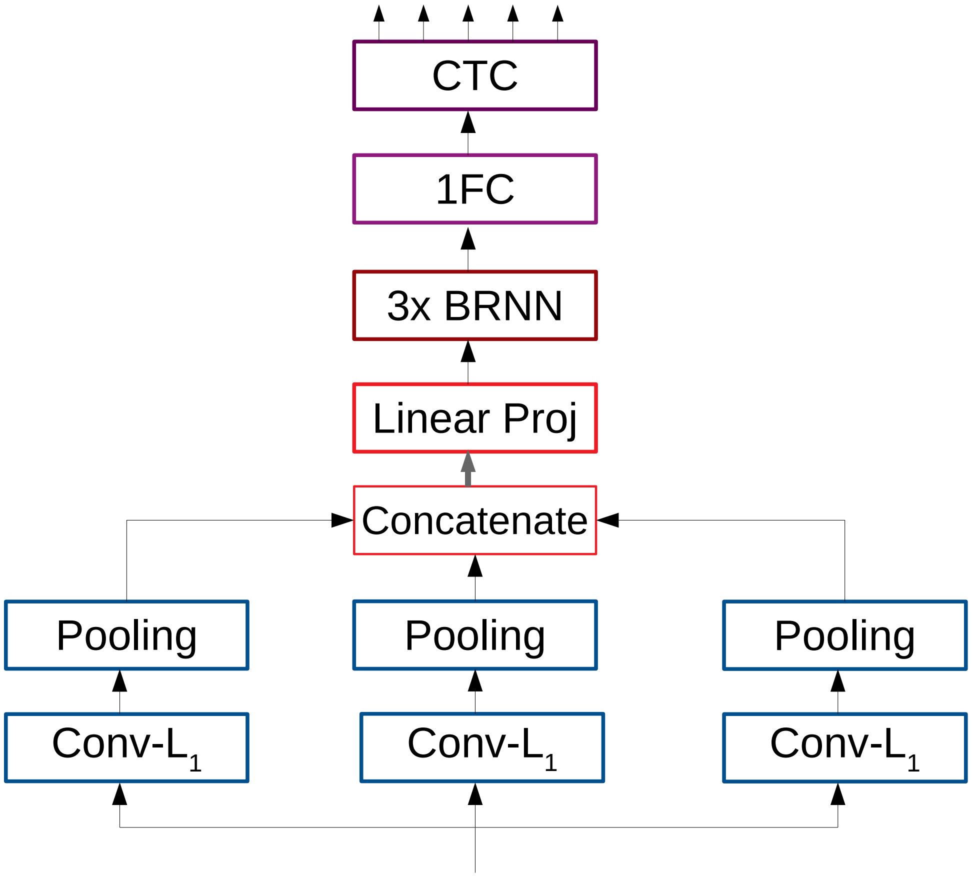
Multi-scale Analysis

- **Idea:** Ensemble of transformations with different resolutions
  - Resolution ≡ Scale
- **Implementation:** Three parallel Conv layers with different filter len
  - 1ms → small Δt; 10ms → small Δf
- **Info Fusion**: Concatenate & linear combination of feature maps



Architecture

- 3 Parallel Conv layers
  - Multi-resolutions
- MaxPooling
  - Consistent sampling rate
- Concat. + Lin projection
  - Info fusion + Dim-Red
- 3x BRNN → 1FC → CTC



Experimental Setup

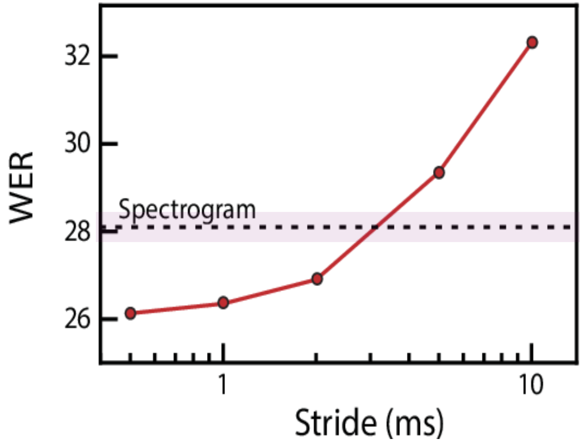
- Data: 2400h, 16 kHz → diverse genre
  - Read, conversational, accented and noisy
- Training
  - SGD, Nesterov momentum, batch-norm per layer
- CTC supplemented with Kneser-Ney 5-gram LM
- Baseline feature: |FFT| (20ms, 10ms)

Single-scale CNN; Stride matters ...

- Smaller stride → Better WER
  - Denser sampling; more info
  - Stride is NOT related to resolution!

- Raw outperforms **baseline** when stride is less than 2ms (fair?)

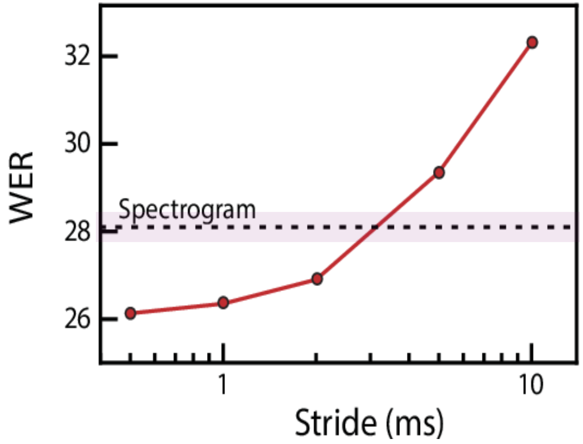
Type	Spectrogram / Convolution			Pooling	WER(%)
	# Features	Window	Stride	Stride	
FFT	161	20ms	10ms	2	**28.10**
wav	161	20ms	10ms	2	32.31
wav	161	20ms	5ms	4	29.35
wav	161	20ms	2ms	10	26.90
wav	161	20ms	1ms	20	26.35
wav	161	20ms	0.5ms	40	**26.13**



Single-scale CNN; Stride matters ...

- Smaller stride → Better WER
  - Denser sampling; more info
  - Stride is NOT related to resolution!

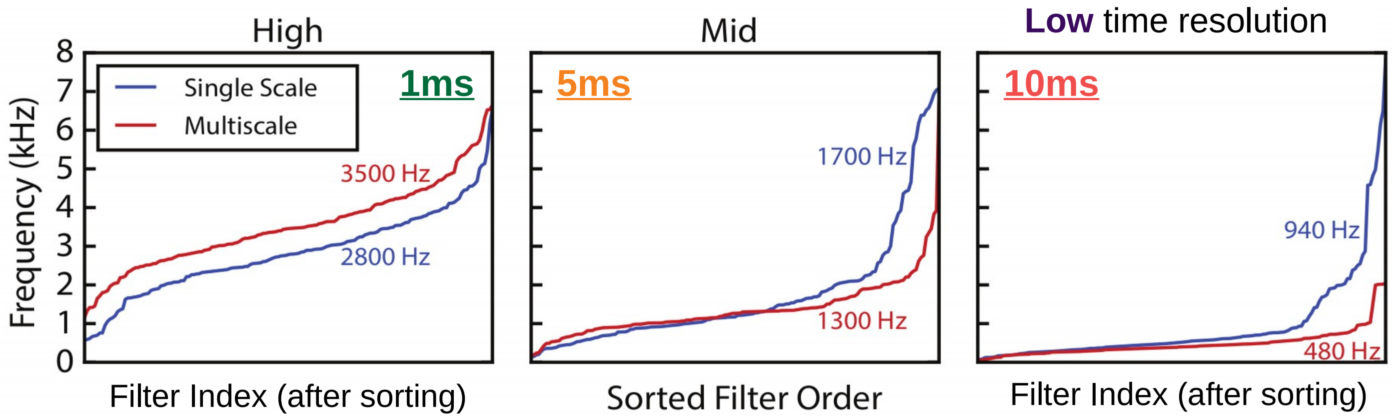
Type	Spectrogram / Convolution			Pooling	WER(%)
	# Features	Window	Stride	Stride	
FFT	161	20ms	10ms	2	**28.10**
wav	161	20ms	10ms	2	32.31
wav	161	20ms	5ms	4	29.35
wav	161	20ms	2ms	10	26.90
wav	161	20ms	1ms	20	26.35
wav	161	20ms	0.5ms	40	**26.13**



- TotalStride (TS) is fixed (in 20ms) to keep sampling rate consistent
  - TS = conv-stride x pooling stride
  - TS ≡ downsampling factor

$$M = \left\lfloor \frac{T-L}{\boxed{S}\,\boxed{P}} \right\rfloor + 1$$

Multi-scale CNNs Spectral Centroid

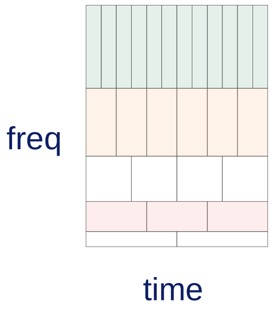


– Printed values → Average f_cs for ConvL

  * f_c = filter spectral centroid

– Multi-scale learning allows each scale to focus on frequencies it mostly efficiently represents

  * Short filters move toward high frequencies [2800 → 3500 Hz]
  * Long-filters move toward low frequencies [940 → 480 Hz]



Experimental Results

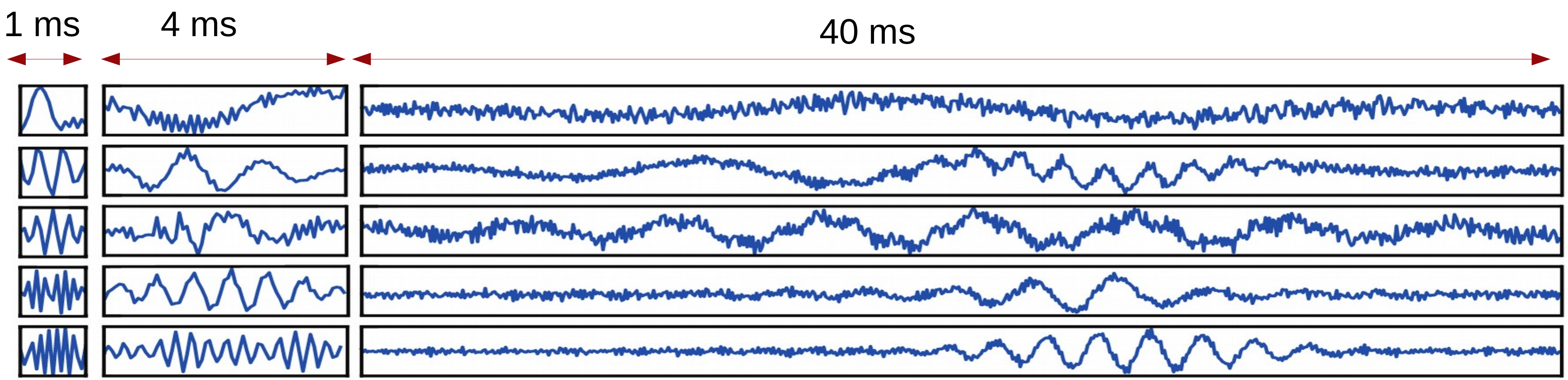
- Filter Len (scales): 1, 4, 40ms
  - **40ms** is optimal
  - longer filters r more flexible!
- Multi-scale outperforms single even with identical #filters (161)
- More filters improves the WER
- Widening BN layer slightly helps

# Features			WER(%)
High (1ms)	Mid (4ms)	Low (40ms)	
161	0	0	32.84
0	161	0	27.69
0	0	161	26.54
61	50	50	**25.67**

Convolution stride = ¼ filter length scales
Bottleneck size: 161
Bottleneck size for *: 800

# Features			WER(%)
High (1ms)	Mid (4ms)	Low (40ms)	
61	50	50	25.67
161	161	161	23.78
160	320	640	23.52
160	320	640	**23.28***

Typical Learned Filters – Impulse Responses



– Short filters focus on high freq; long filters on low frequencies
– Some filters localized in frequency (similar to sinusoid)
– Phase shifted filter pairs are also found → phase info importance

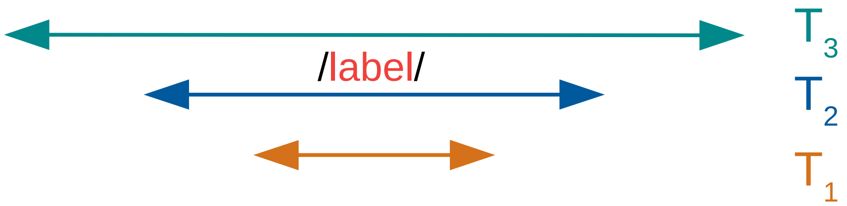
Multi-Span Acoustic Modelling using Raw Waveform Signals

P. von Platen[1,2], *C. Zhang*[1], *P. C. Woodland*[1]

[1] Cambridge University Engineering Dept., Trumpington St., Cambridge, CB2 1PZ U.K.
[2] Institute of Communication Systems (IKS), RWTH Aachen University, Germany
`{pwv20,cz277,pcw}@eng.cam.ac.uk`

Multi-span Acoustic Modelling; Idea

- Combine multiple input streams with different lengths
  - Multi-span ≡ multi-stream
- All streams share the centre and label
- i^{th} span len (T_i) is a function of CNN parameters

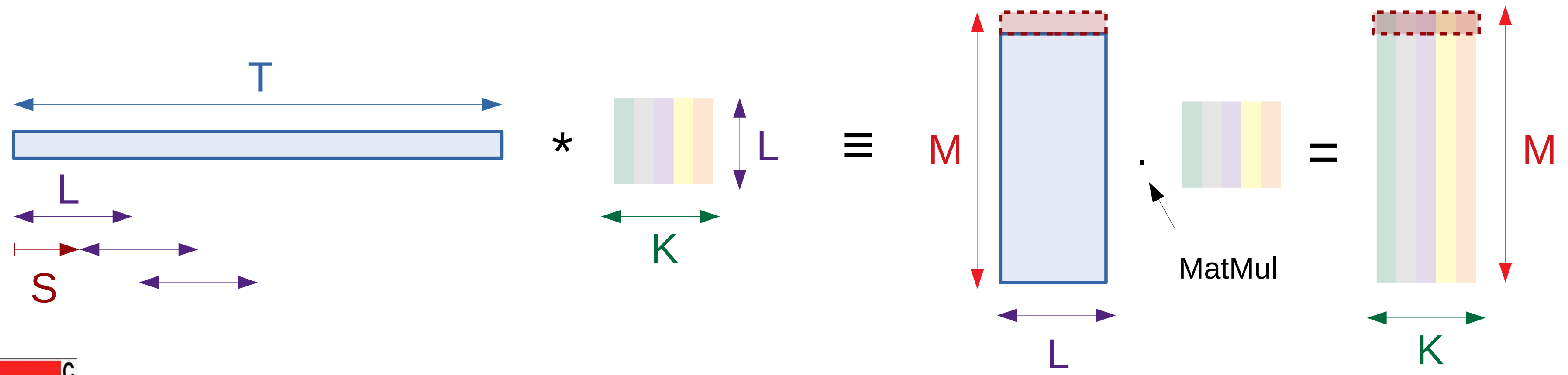


1D-Conv Review

$$\underbrace{\tilde{\mathbf{y}}_k}_{} = \overset{k^{th}\text{ channel}}{\boxed{\mathbf{w}_k}} \; \overset{\text{Conv with Stride S}}{\boxed{*^S}} \; \overset{\text{Input x; length T}}{\boxed{\mathbf{x}_1^T}}$$

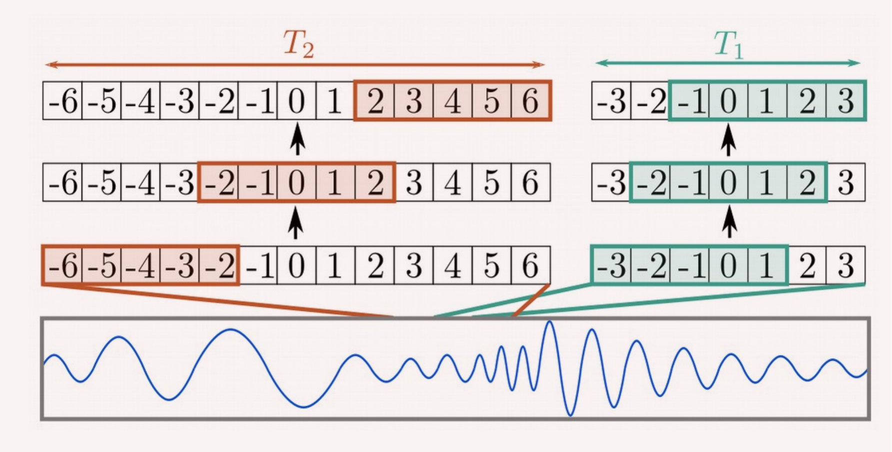
$$M = \left\lfloor \frac{T - L}{S} \right\rfloor + 1$$

– T: input length in samples
– L: filter length in samples
– K: number of filters (5 here)
– S: stride in samples
– M: Conv output length in samples (per channel)



Multi-span CNN

- T: span/stream length
  - T = (M-1)S + L
- For i^{th} stream ...
  - Fix M_i in M
  - Set L_i & S_i; Now find T_i
- Goal: learn more diverse feature representation
  - Contextual info



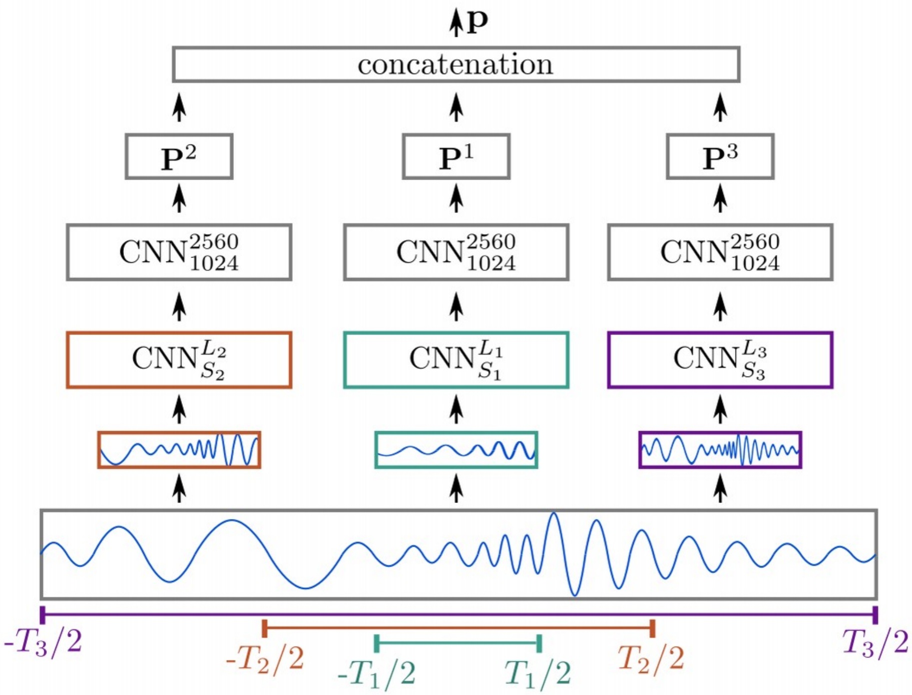
L_2:5, S_2:4, T_2:13 M=3 L_1:5, S_1:1, T_1:7

$$M = \left\lfloor \frac{T-L}{S} \right\rfloor + 1$$

$$T = (M-1)S + L$$

Multi-Span CNN Architecture

- Each stream processed by ...
  - A stack of two CNNs
  - Linear projection (P_i)
    - Dim reduction $R^{MxK} \rightarrow R^{150}$
- Concatenated [P_1, P_2 , P_3]
- MLP with 4 hidden layers
  - 512 ReLU unite per layer



Multi-Span CNN Architecture

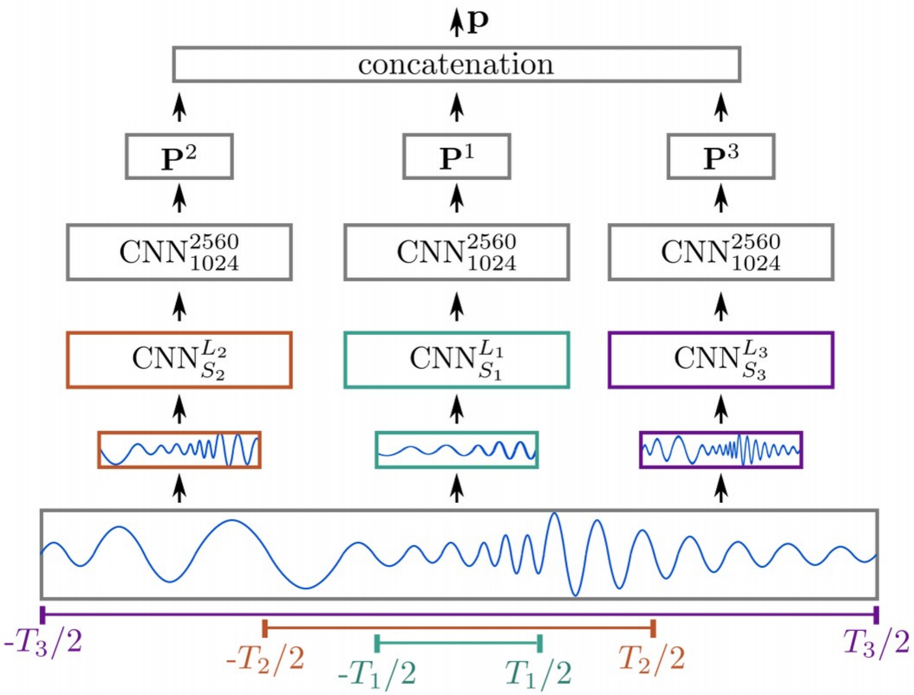
$$\mathbf{y}_k^i = \mathbf{w}_k *^S \mathbf{x}_{-T_i/2}^{T_i/2}$$

$$\mathbf{y}^i = CNN_{S_i}^{L_i}(\mathbf{x}_{-T_i/2}^{T_i/2}, M_i)$$

$$\mathbf{o}^i = CNN_{S_{i_2}}^{L_{i_2}}(\mathbf{y^i}, M_{i_2})$$

$$\mathbf{p}^i = \mathbf{P}^i\ \mathbf{o}^i_{flatten}$$

$$\mathbf{p} = \text{concatenate}(\mathbf{p}^1, \mathbf{p}^2, \mathbf{p}^3)$$



Multi-Span Processing Interpretation

- ... M is fixed, L, S and T (span) vary. This **COULD** mean ...

Multi-Span Processing Interpretation

- ... M is fixed, L, S and T (span) vary. This **COULD** mean ...
  - Multi-resolution processing
    - Filters with different L **and** fixed S
  - Multi-rate sampling
    - Filters with different S **and** fixed L

Multi-Span Processing Interpretation

- ... M is fixed, L, S and T (span) vary. This **COULD** mean ...
  - Multi-resolution processing
    - Filters with different L **and** fixed S
  - Multi-rate sampling
    - Filters with different S **and** fixed L

- Which one is better? Multi-resolution or multi-rate?

Experimental Setup

- Databases: CHiME4 and AMI
- Toolkit: HTK 3.5.1 and PyHTK
- Training: CE, SGD, Momentum, Weight decay, NewBob+ learning rate scheduler, 10% CrossVal
- First ConvLayer
  - M_i=200, K = #kernels = 64, L & S adjusted
- Second ConvLayer setting, for all streams,
  - M_{i2}=11, S_{i2}=1024, L_{i2}=2560, K_2=64 ???
- DNN on top of concatenated features → MLP-4HL-512-ReLU

CHiME4 – Single Span

- WER_{Fbank} < $WER_{Single\text{-}span\ Raw}$
- Fixing S in 10 samples (~0.6ms)
  - Optimal L: 50 samples [~3ms]
- Fixing L in 50 samples
  - Optimal S: 15 samples
  - Too short (4) or too long (20 samples) is not optimal

	samples		ms	WER
ID	S	L	T	dev
F_{160}^{400}	160	400	125	18.1
I_{10}^{400}	10	400	149	20.2
I_{10}^{100}	10	100	131	19.4
I_{10}^{50}	10	50	128	19.3
I_{10}^{25}	10	25	125	20.7
I_{4}^{50}	4	50	53	23.2
I_{9}^{50}	9	50	115	19.7
I_{15}^{50}	15	50	190	18.3
I_{20}^{50}	20	50	252	20.7

F_{160}^{400} : FBank baseline

400: 25ms, 160: 10ms

CHiME4 – Multi-Span

- Multi-span with optimal setting outperforms Fbank & single

	samples		ms	WER
ID	S	L	T	dev
$M^{50,100,400}_{15,15,15}$	15	50,100,400	190-212	18.4
$M^{50,100,400}_{4,9,15}$	4,9,15	50,100,400	53-212	17.9
$M^{50,50,50}_{4,9,15}$	4,9,15	50	53-190	17.1
F^{400}_{160}	160	400	125	18.1
I^{50}_{15}	15	50	190	18.3

Baseline: FBANK

Best Single-Span

CHiME4 – Multi-Span

- Multi-span with optimal setting outperforms Fbank & single
- Multi-resolution processing
  - Variable L, fixed S
- Multi-rate sampling
  - Fixed L, variable S

ID	S	L	T	dev
$M^{50,100,400}_{15,15,15}$	15	50,100,400	190-212	18.4
$M^{50,100,400}_{4,9,15}$	4,9,15	50,100,400	53-212	17.9
$M^{50,50,50}_{4,9,15}$	4,9,15	50	53-190	17.1
$\mathbf{F}^{400}_{160}$	160	400	125	18.1
I^{50}_{15}	15	50	190	18.3

Baseline: FBANK
Best Single-Span

CHiME4 – Multi-Span

- Multi-span under optimal setting outperforms Fbank & single-span
- Multi-resolution processing
  - HERE, Fbank and Single-span are better!!!
- Multi-rate sampling
  - Optimal performance

ID	S	L	T	dev
$M^{50,100,400}_{15,15,15}$	15	50,100,400	190-212	18.4
$M^{50,100,400}_{4,9,15}$	4,9,15	50,100,400	53-212	17.9
$M^{50,50,50}_{4,9,15}$	4,9,15	50	53-190	17.1
F^{400}_{160}	160	400	125	18.1
I^{50}_{15}	15	50	190	18.3

Baseline: FBANK
Best Single-Span

AMI-IHM – Single and Multi-Span

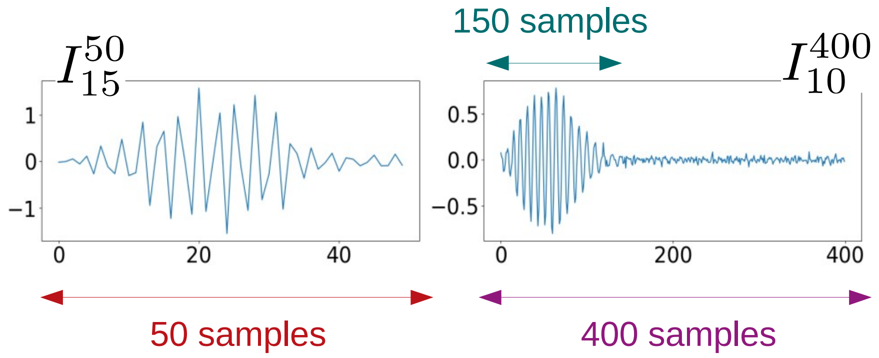
- Optimal single & multi-spam outperform Fbank
  - Single-span: 0.3% abs
  - Multi-span: 1.8%
  - Single was worse for CHiME4
- Optimal setup
  - Single: L=50, S=15
  - Multi: L=50, S=4,9,15

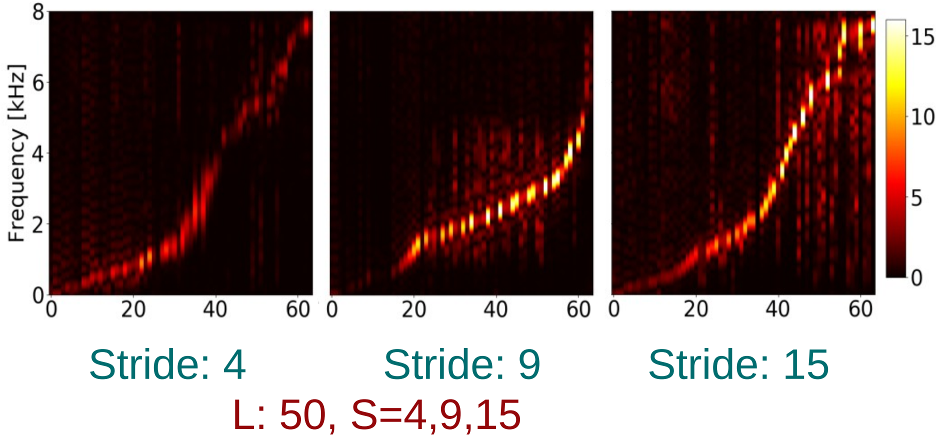
ID	System	dev	eval
F_{160}^{400}	FBANK-DNN	28.3	31.1
I_{10}^{400}	Single-Span-DNN	29.1	31.9
I_{15}^{50}	Single-Span-DNN	28.1	30.8
$M_{4,9,15}^{50,50,50}$	Multi-Span-DNN	27.2	29.3

Learned Filters

- Model tends to learn short filters (HERE)

- Filters do not seem to follow an audiological distribution
  - For L=50, S=4,9,15 ...
    - S=4 → emphasis on low freq
    - S=15 → emphasis on high freq
    - Why?





CENTER FOR LANGUAGE AND SPEECH PROCESSING

Acoustic modelling from the signal domain using CNNs

Pegah Ghahremani[1], *Vimal Manohar*[1], *Daniel Povey*[1,2], *Sanjeev Khudanpur*[1,2]

[1]Center of Language and Speech Processing
[2]Human Language Technology Center Of Excelence,
Johns Hopkins University, Baltimore, MD
{pghahre1,vmanoha1,khudanpur}@jhu.edu, dpovey@gmail.com

INTERSPEECH 2016

Idea and Contribution

- Using a modified NIN architecture
- Feature/Data pre-processing
  - MVN, speed and shift perturbation
- Speaker adaptation (iVector bias)
- Filter interpretation

Idea and Contribution

- Using a modified NIN architecture
- Feature/Data pre-processing
  - MVN, speed and shift perturbation
- Speaker adaptation (iVector bias)
- Filter interpretation

Network in Network (NIN)

Network In Network

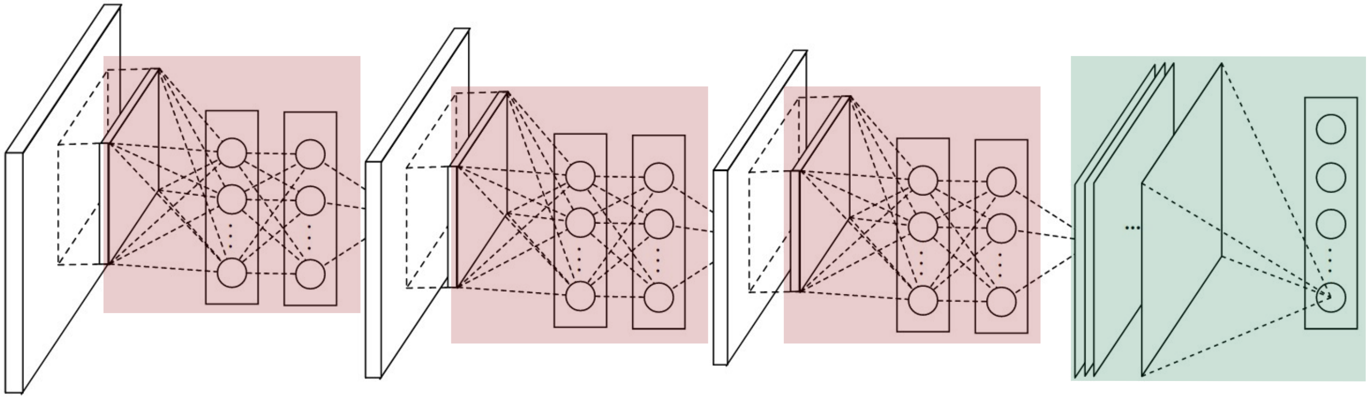
Min Lin[1,2], Qiang Chen[2], Shuicheng Yan[2]
[1]Graduate School for Integrative Sciences and Engineering
[2]Department of Electronic & Computer Engineering
National University of Singapore, Singapore
{linmin,chenqiang,eleyans}@nus.edu.sg

ICLR
2014

E. Loweimi

Network in Network (NIN)

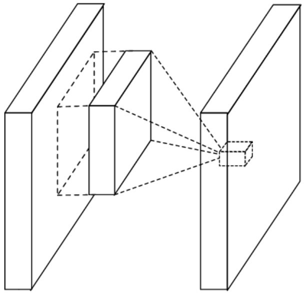
- NIN has two main components:
  - Micro NN, e.g. MLP
    - Each adjacent layer pair has their own Micro NN
  - Global Average Pooling

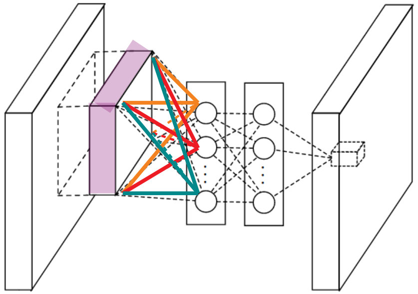


NIN – **MLPconv** Layer

- A non-linear filtering, allows complex and learnable interaction between channels
  - Cross channel parametric pooling structure
  - Comparable to linear channel combination via 1x1 Conv
- Channels' response to each input patch is computed, then non-linearly combined through MLP

ConvLayer





MLPconv

NIN – Global Average Pooling

- IDEA: Replace the FC NN with a Conv Layer
  - Channel ≡ Class, #Channels = #Classes
- HOW:
  - Compute and Average the feature map for each channel
  - Pass the averages to softmax
- ADVANTAGES:
  - Fewer parameters than FC + Some translation invar

Idea and Contribution

- Using a modified NIN architecture
- Feature/Data pre-processing
  - MVN, speed and shift perturbation
- Speaker adaptation (iVector bias)
- Filter interpretation

Features Pre-processing: MVN

- Raw waveform is MVNed at utterance level
  - DC removal and loudness equalisation
  - Stabilise the training
    - Put numbers in similar range
    - Slightly faster convergence
    - Identical final performance on WSJ

Data Perturbation: Speed & Shift

- **Speed** → Articulation speed invariant → MFCC & Raw
  - Speed factor: 0.9, 1.0, 1.1
- **Shift** → translation invariant
  - |FFT|-based features are shift invariant, BUT Raw is NOT
  - Randomly shift raw frames to right (≤ 0.2 frame-len)
  - Improves CE on Train and Dev

Perturbation method	Training CE	Validation CE
No random shift	-0.96	-1.22
With random shift	-0.88	-1.13

Data Perturbation: Speed & Shift

- **Speed** → Articulation speed invariant → MFCC & Raw
  - Speed factor: 0.9, 1.0, 1.1
- **Shift** → translation invariant
  - |FFT|-based features are shift invariant, BUT Raw is NOT
  - Randomly shift raw frames to right (≤ 0.2 frame-len)
  - Improves CE on Train and Dev

Perturbation method	Training CE	Validation CE
No random shift	-0.96	-1.22
With random shift	-0.88	-1.13

Data Perturbation: Speed & Shift

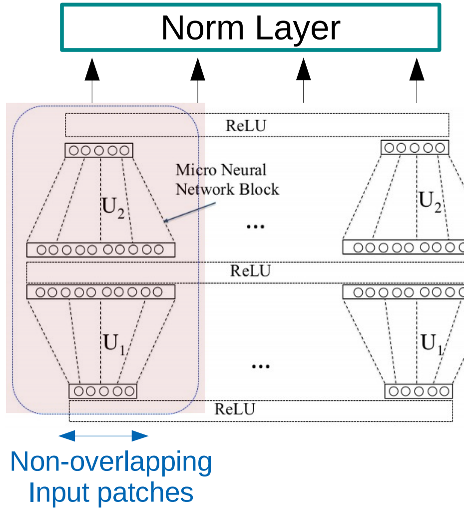
- **Speed** → Articulation speed invariant → MFCC & Raw
  - Speed factor: 0.9, 1.0, 1.1
- **Shift** → translation invariant
  - |FFT|-based features are shift invariant, BUT Raw is NOT
  - Randomly shift raw frames to right (≤ 0.2 frame-len)
  - Improves CE on Train and Dev

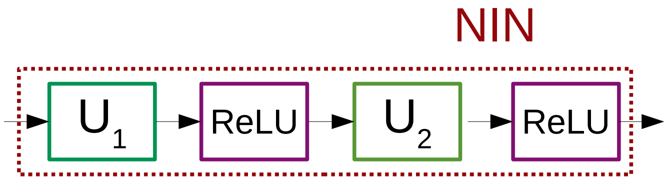
Perturbation method	Training CE	Validation CE
No random shift	-0.96	-1.22
With random shift	-0.88	-1.13

– Can CE become negative?
– Should be Log-likelihood ...

NIN Architecture

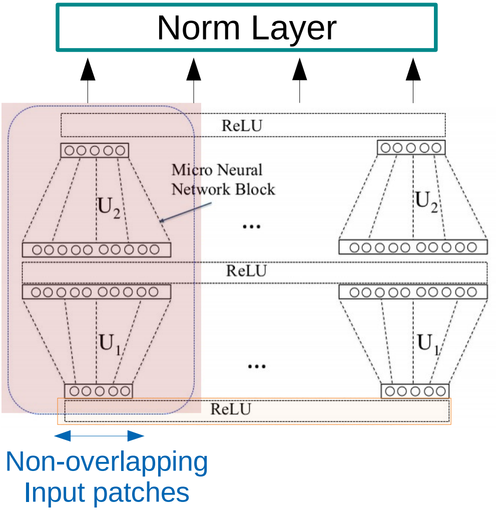
- Layers are interleaved with *Micro NN*
- NIN is interpretable as a *pooling block* or a *many-to-many non-linearity*
- HERE: *μNN*: $U_1 \rightarrow \text{ReLU} \rightarrow U_2 \rightarrow \text{ReLU}$
  - $U_1 \rightarrow$ m x k linear mapping
  - $U_2 \rightarrow$ k x n linear mapping
  - m: in-dim; n:out-dim
  - k: NIN hidden dim ($k \approx 5m$)

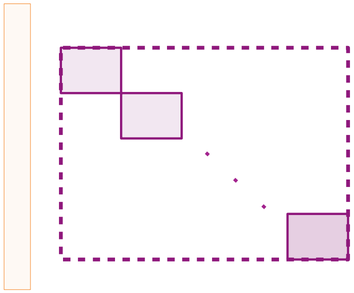




NIN Architecture

- Interpretable as a FC layer with block diagonal weight matrix
- Sharing U_is across a NIN ≡ 1D-Conv
- U_is operate on non-overlapping patches
  - m = Filter length = Stride
  - A FC layer with shared block diagonal W





Normalisation Layers

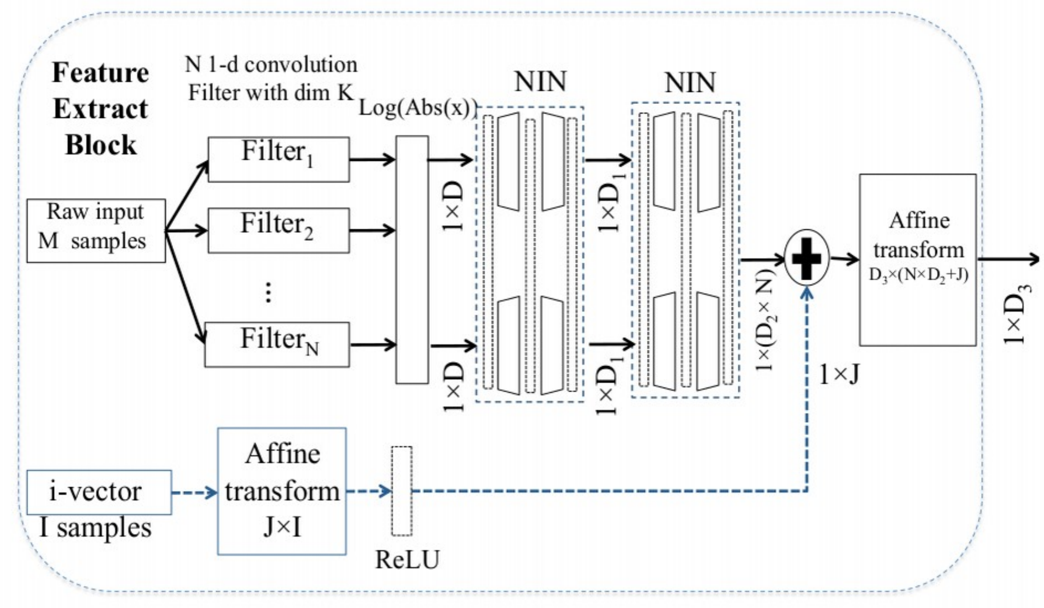
- Normalisation layer is put after each NIN
- **Goal**: Scale down the whole set of activations and stabilises training
- **Application**: For unbounded-output non-linearities
- **How**:
  - $y_i = x_i / \sigma$ if $\sigma > 1$ else x_i # σ is *uncentered* STD of layer X (x_i: i^{th} unit)

Statistics Extraction Layers

- Statistics Extraction layer
  - Computes 1st and 2nd (STD) order statistics from hidden layer activation
    - Stats computed over a moving win of ≤ 200 frames (2 sec)
    - Stats are appended to the input of the next hidden layer (bias)
  - Advantages:
    - Capture long-term effect (speaker, channel, environment)
    - Hopefully helpful in alleviating sensitivity to them

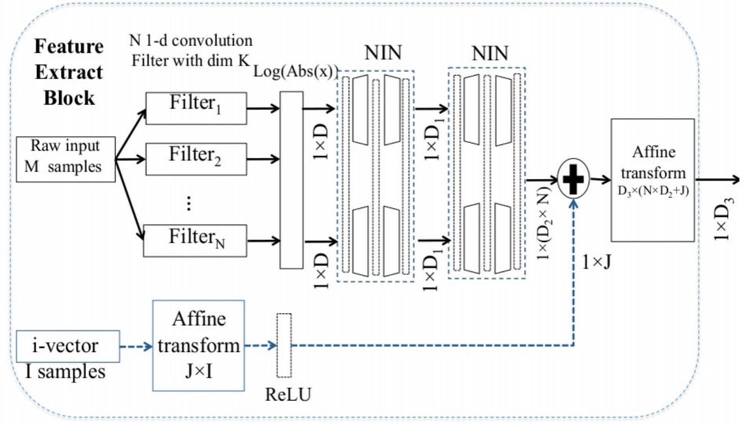
Feature Extraction Block

- 1D-Conv with #Ch = N
- Log(Abs)
- 2 NINs
- Append with iVector
- Affine Transform
  - Output dim: D_3



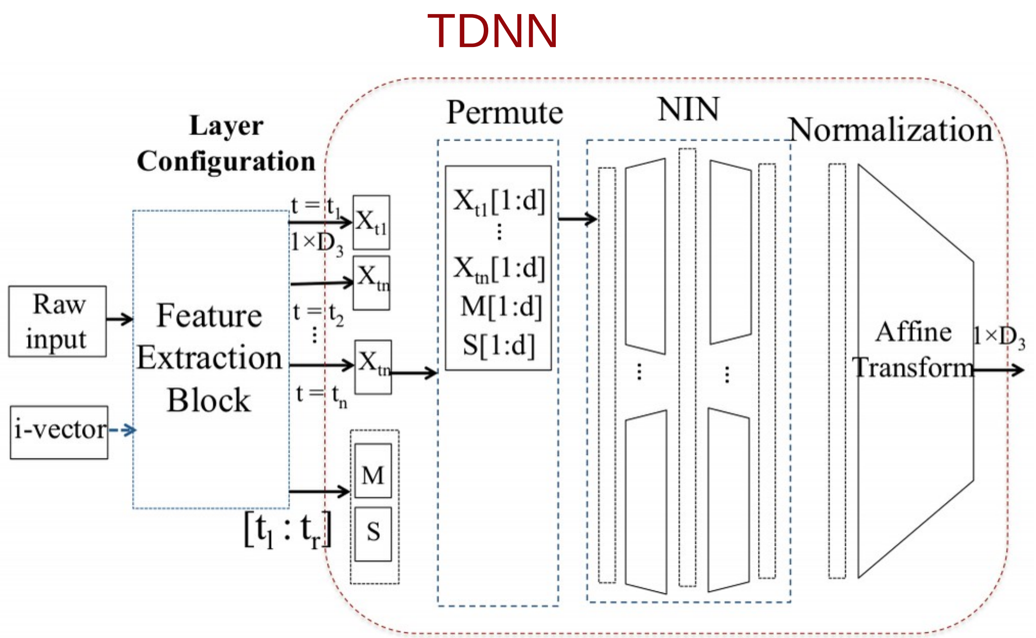
Feature Extraction Block

- 1D-Conv, N filters, Kernel_len K, Stride S
  - NIN shared across all N filters/bands
- Log (|ConvOut|) ≡ log-Fbank
- 2 NINs + norm layers
- Speaker adaptation using iVector
  - iVector → Affine trans. → ReLU → Append
- Affine projection after augmentation by iVector



Classification Block

- Appended X_{t1}, …, X_{tn} with moving stats (M & S) extracted by StatsExt layer
- Splice the features via TDNN
- One NIN layer
- Affine transformation
  - Dim reduction to D_3
- MLP → 6 HiddLayers (ReLU)



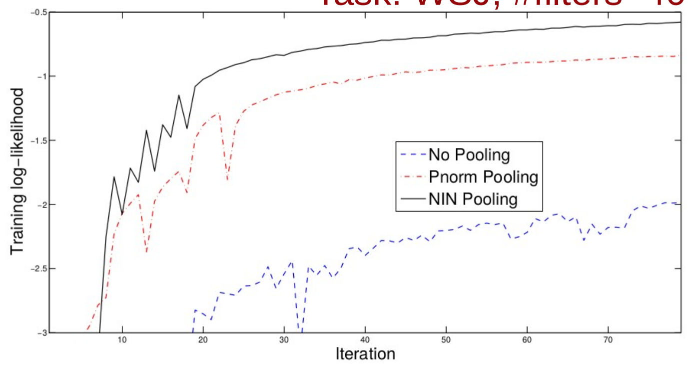
Experimental Setup

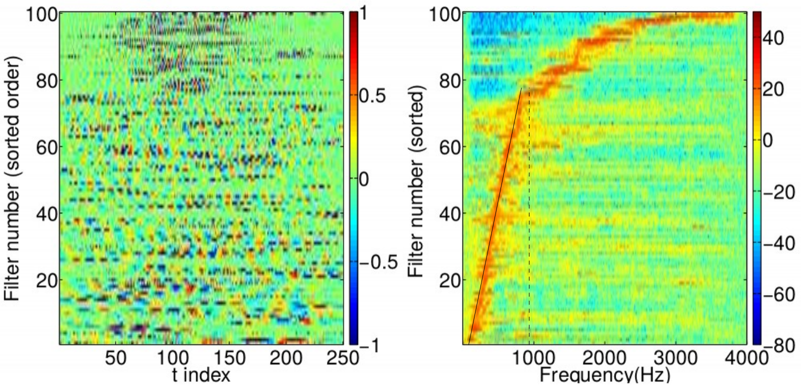
- MFCC: 40-dim, iVector: 100-dim
- Raw waveform length: M = 50ms, MVN on utterance level
- WSJ:
  - #filters=40, filter_len=30ms, stride=0.625ms (10 samples)
  - m=16, k=300, n=32, 6HL-750-ReLU
- SWB:
  - #filters=100, filter_len=31.25, stride=1.25ms (10 samples)
  - FeatureExtraction block: m=16, k=120, n=18, D_3=500, 100 micro NIN
  - Classification block: m=5, k=75, n=18, 100 micro NIN, 6HL-600-ReLU
  - lattice-free MMI
  - StatsExtractor MVN → 99 frames on either sides

First Layer's Learned Filters

- NIN effect vs p-norm pooling
  - Faster convergence
  - Higher log-likelihood
  - Max-pooling ???
- Learned filters @ L_1
  - Bandpass filters
  - Linear < 1 kHz
  - Non-linear > 1 kHz

Task: WSJ; #filters=40





Task: SWB; #filters=100

Experimental Results – WSJ

- WER_{MFCC} - WER_{Raw} ≈ 1% abs
- Raw → used p-norm instead NIN
- Raw+NIN vs Raw
  - Worse WER, better log-like
- iVector speaker adaptation
  - improves MFCC (+9.4% RWERR)
  - degrades Raw (-5.9% RWERR*)

Table 2: *WER (%) Results on WSJ LVCSR task.*

Model	*Nov'92 eval*	*Nov'93 dev*
MFCC	5.28	8.29
Raw	3.95	7.34
Raw + NIN	3.92	7.6
MFCC + iVector	4.52	7.51
Raw + iVector	4.06	7.80

* RWERR: Relative WER Reduction

Experimental Results – SWB

- Raw slightly (0.1% abs) outperforms MFCC
- Using StatsExt layer is useful
  - More useful for Raw
- iVector useful for both
  - It should be "+Stats+iVector"
  - Slightly useful for Raw
  - More useful for MFCC

Table 4: *WER (%) Results on Switchboard LVCSR task.*

	Hub5'00		*RT'03*	
Model	Total	SWBD	Total	SWBD
MFCC	17.5	11.6	22.1	26.6
Raw	17.4	11.5	21.7	26.5
MFCC + Stats	16.4	11.0	20.0	24.3
Raw + Stats	16.3	10.6	19.1	23.3
MFCC + iVector	15.7	10.4	19.2	23.5
Raw + iVector	16.1	10.5	18.9	23.1

– MFCC ↔ ReLU
– Raw ↔ NIN

* "*… but only a little improvement in the raw waveform setup ...*"

CONVOLUTIONAL, LONG SHORT-TERM MEMORY, FULLY CONNECTED DEEP NEURAL NETWORKS

Tara N. Sainath, Oriol Vinyals, Andrew Senior, Haşim Sak

Google, Inc., New York, NY, USA
{tsainath, vinyals, andrewsenior, hasim}@google.com

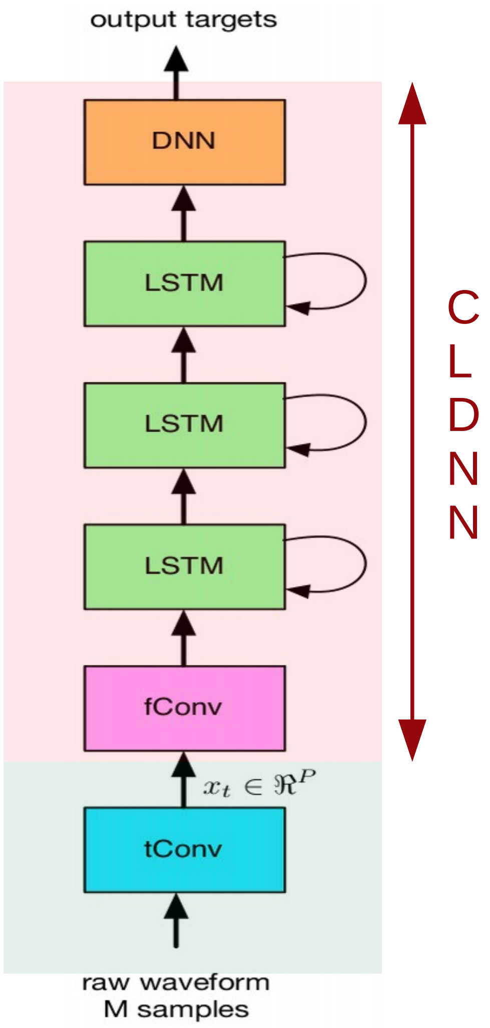
ICASSP
2015

Learning the Speech Front-end With Raw Waveform CLDNNs

Tara N. Sainath, Ron J. Weiss, Andrew Senior, Kevin W. Wilson, Oriol Vinyals

Google, Inc. New York, NY, U.S.A
{tsainath, ronw, andrewsenior, kwwilson, vinyals}@google.com

INTERSPEECH
2015



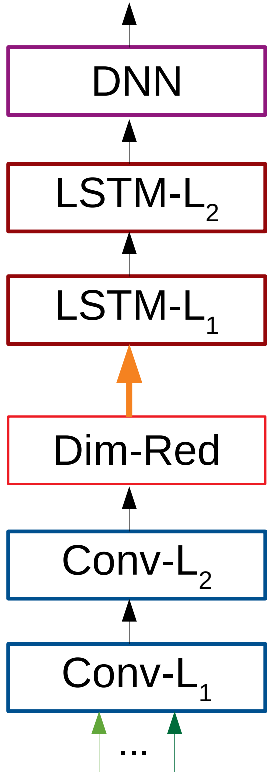
CNN, LSTM and DNN (MLP) ...

- ... are limited in their modelling capabilities ...
  - CNN → Efficient feature extraction; Invariant to ...
  - LSTM → Temporal/Sequential processing
  - DNN (MLP) → Abstract representation extraction
    - Linearly separable → class discrimination

- What is an optimal combination?
  - GMM/HMM: MFCC → HMM → GMM

CLDNN: CNN + LSTM + DNN

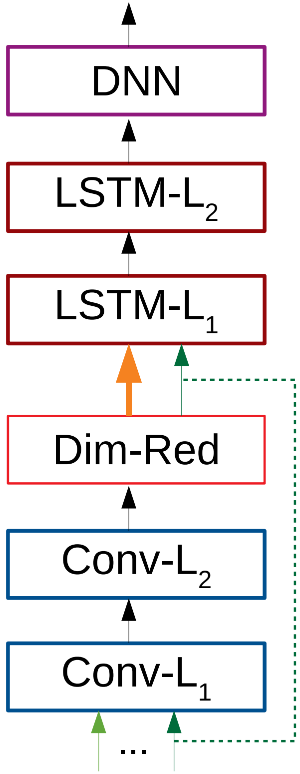
- fConv: 2 Layers 2D-Conv
  - Max-pooling: non-overlapping, only L_1, only in frequency
- Linear dim reduction (from flatten to 256)
- LSTM: 2 layers, 832 cells, projected to 512
- DNN → 2 layers, 1024 ReLU units



$\mathbf{X}_t = [x_{t-L}, \ldots, x_t]$

CLDNN: CNN + LSTM + DNN

- fConv: 2 Layers 2D-Conv
  - Max-pooling: non-overlapping, only L_1, only in frequency
- Linear dim reduction (from flatten to 256)
- LSTM: 2 layers, 832 cells, projected to 512
- DNN → 2 layers, 1024 ReLU units
- Multi-scale addition → concatenate long-term representation $f(x_{t-l},\ldots,x_t)$ with x_t



$\mathbf{X}_t = [x_{t-L}, \ldots, x_t]$

Experimental Setup

- Data: Voice Search task,
  - 200h and 2000h, clean and noisy
- Optimisation:
  - Asynchronous SGD (ASGD) + exp learning rate decay
- Architecture: variable for different experiments
  - #filters=256, max-pooling@L_1=3
- Initialisation:
  - CNN & DNN: Glorot-Bengio (Gaussian)
  - LSTM → zero-mean, var: 1/#inputs

Experimental Results – Baselines

- DNN: FC-6L-1024-ReLU; Context: [-20,+5]
- CNN: 2LConv + FC-4L-1024-ReLU; Context: [-20,+5]
- LSTM: 2L, unroll:20, context: [-/=0,0]
- LSTM & CNN works equally well

Feature: FBank

Method	WER
DNN	18.4
CNN	18.0
LSTM	18.0

Experimental Results – Baselines

- DNN: FC-6L-1024-ReLU; Context: [-20,+5]
- CNN: 2LConv + FC-4L-1024-ReLU; Context: [-20,+5]
- LSTM: 2L, unroll:20, context: [-*l*,0]
  - Adding left context [-l,0] is not required!
  - Unroll=30 is not optimal
  - Adding third Layer was not useful

Feature: FBank

Method	WER
DNN	18.4
CNN	18.0
LSTM	18.0

Method	WER
LSTM, l=0, unroll=20	18.0
LSTM, l=10, unroll=20	18.0
LSTM, l=0, unroll=30	18.2

DNN-LSTM vs CNN-LSTM

- CNN+LSTM
  - Better than LSTM
- DNN+LSTM
  - Worse than LSTM

Method	WER
DNN	18.4
CNN	18.0
LSTM	18.0

Input Context	# Steps Unroll	WER CNN	WER DNN
l=0,r=0	20	17.8	18.2
l=10,r=0	20	**17.6**	18.2
l=20,r=0	20	17.9	18.5

- CNN is a better feature extraction

CNN → LSTM
DNN → LSTM

DNN-LSTM vs CNN-LSTM

- CNN+LSTM vs DNN+LSTM
  - CNN is a better

Method	WER
DNN	18.4
CNN	18.0
LSTM	18.0

- Optimal context: [-10,0]
  - CNN & DNN need context!
    - NOT LSTM!

Input Context	# Steps Unroll	WER CNN	WER DNN
l=0,r=0	20	17.8	18.2
l=10,r=0	20	**17.6**	18.2
l=20,r=0	20	17.9	18.5

CNN → LSTM
DNN → LSTM

LSTM + DNN

- LSTM+DNN outperform LSTM
  - Contrary to DNN+LSTM ...
  - Gain saturated after 2 FC layers

# DNN Layers	WER
0	18.0 (LSTM)
1	17.8
2	**17.6**
3	17.6

200h data, FC-1024-ReLU

- Both CNN+LSTM & LSTM+DNN work well; combine them ...
  - CNN+LSTM → LSTM+DNN
  - CNN → LSTM→ DNN = CLDNN

Method	WER
LSTM	18.0
CNN+LSTM	17.6
LSTM+DNN	17.6
CLDNN	**17.3**

Effect of Other Factors

- Initialisation effect:
  - Uniform vs Gauss
  - Uniform is better (WER: 17.3 → 17.0)

200h

Method	WER - Gaussian Init	WER - Uniform Init
LSTM	18.0	17.7
CLDNN	17.3	**17.0**

- Multi-scale addition is useful (16.8)
- Passing CNN output to both LSTM & DNN is NOT useful

200h

Method	WER
LSTM (Uni Init)	17.7
CLDNN, long-term feature to LSTM	17.0
+ short-term feature to LSTM	**16.8**
+ CNN to LSTM and DNN layers	17.0

Training on 2000 hour + Seq Training

- Advantages of CLDNN carry over to 2000h data
- LSTM → CLDNN, CE RWERR
  - Clean: 4.1%; Multi: 4.4%
- LSTM → CLDNN, Seq RWERR
  - Clean: 4.4%; Multi: 7.4%
- CE → Seq, RWERR: 6% → 10%
- Multi-scale useful only for CE

Clean

Method	WER-CE	WER-Seq
LSTM	14.6	13.7
CLDNN	14.0	13.1
multi-scale CLDNN	**13.8**	**13.1**

Multi

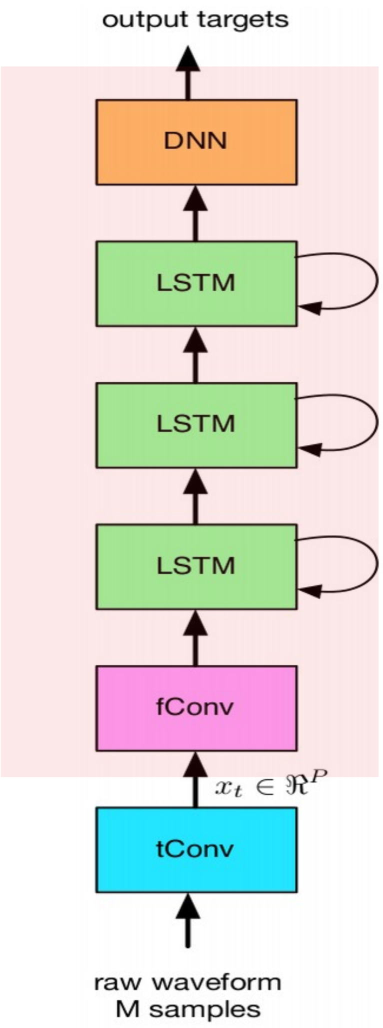
Method	WER-CE	WER-Seq
LSTM	20.3	18.8
CLDNN	19.4	**17.4**
multi-scale CLDNN	**19.2**	**17.4**

Next Session …

Raw waveform modelling using CLDNN + Beamforming

+

Parametric CNNs for Raw Waveform Modelling



That's It!

- Thanks for Your Attention!
- Q/A