Speaker and Speech Recognition from raw waveform with SincNet

Mirco Ravanelli

Mila
On Processing Waveforms...

Problem:

- Speech/Audio sequences are very high-dimensional.

Raw Waveform: 1 second = 16000 features

FBANKs/MFCCs: 1 second ≈ 4000 features

- *Hand-crafted* features (e.g. MFCCs, PLPs, or FBANKs) are still employed to achieve a more compact representation.

https://github.com/mravanelli/SincNet

M. Ravanelli, Y. Bengio, "Speaker Recognition from raw waveform with SincNet", in Proc. of SLT 2018.
On Processing Waveforms...

Problem:
- *Hand-crafted* features are designed from *perceptual evidence*.

Not necessarily optimal for all audio/speech tasks!
(e.g., Speaker recognition)

We smooth the spectrum, possibly hindering the extraction of pitch and formants.

---

https://github.com/mravanelli/SincNet

Standard Approach

Problem:
• Recent works have proposed directly feeding CNNs with raw waveforms.

Convolution:
\[ y[n] = x[n] * h[n] = \sum_{l=0}^{L-1} x[l] \cdot h[n - l] \]

Critical Part:
• High Dimensionality
• Vanishing Gradient

https://github.com/mravanelli/SincNet
M. Ravanelli, Y. Bengio, "Speaker Recognition from raw waveform with SincNet", in Proc. of SLT 2018.
Problem:
- Recent works have proposed directly feeding CNNs with raw waveforms.

CNN filters:

---

GitHub: https://github.com/mravanelli/SincNet

M. Ravanelli, Y. Bengio, "Speaker Recognition from raw waveform with SincNet", in Proc. of SLT 2018.
**Problem:**
- Recent works have proposed directly feeding CNNs with raw waveforms.

*Can we help the CNN discover more meaningful filters?*

*A simple idea: Inject prior knowledge on the filter shape*

M. Ravanelli, Y. Bengio, "Speaker Recognition from raw waveform with SincNet", in Proc. of SLT 2018.

[GitHub link: https://github.com/mravanelli/SincNet]
SincNet

Standard CNN:

\[ y[n] = x[n] \ast h[n] \]

We learn all the elements of each filter

SincNet:

\[ y[n] = x[n] \ast g[n, \theta] \]

We only learn the \( \theta \) parameters of the predefined kernel

What could be a good choice for \( g(\cdot) \)?
SincNet

- We can choose $g(\cdot)$ to implement a bank of **band-pass filters** where low and high **cutoff frequencies** are the only parameters learned.

\[
G[f, f_1, f_2] = \text{rect} \left( \frac{f}{2f_2} \right) - \text{rect} \left( \frac{f}{2f_1} \right)
\]

**Frequency Domain:**

**Time Domain:**

\[
g[n, f_1, f_2] = 2f_2 \text{sinc}(2\pi f_2 n) - 2f_1 \text{sinc}(2\pi f_1 n)
\]

For each filter we only learn $f_1$ and $f_2$

M. Ravanelli, Y. Bengio, "Speaker Recognition from raw waveform with SincNet", in Proc. of SLT 2018.

https://github.com/mravanelli/SincNet
SincNet

CNN Filters:

SincNet Filters:

https://github.com/mravanelli/SincNet

M. Ravanelli, Y. Bengio, "Speaker Recognition from raw waveform with SincNet", in Proc. of SLT 2018.
Model Properties

- **Few Parameters:**
  F = Number of filters (e.g. 80)
  L = Length of each filter (e.g. 100)

<table>
<thead>
<tr>
<th>Standard CNN</th>
<th>F · L parameters (8k)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SincNet</strong></td>
<td>2F parameters (160)</td>
</tr>
</tbody>
</table>

The number of parameters doesn’t depend on L.

We can achieve **high frequency selectivity** without wasting parameters!
SincNet

Model Properties

- Few Parameters
- Fast Convergence

Speaker Classification

- Softmax
- CNN/DNN layers
- Dropout
- Leaky ReLU
- Layer Norm
- Pooling

Speech Waveform

Model Evaluation

SincNet vs. CNN
SincNet

Model Properties

- Few Parameters
- Fast Convergence
- Interpretability

Fig. 3: Cumulative frequency response of the SincNet filters.
SincNet

Speaker Classification

Softmax
CNN/DNN layers
Dropout
Leaky ReLU
Layer Norm
Pooling

• Interpretability

M. Ravanelli, Y. Bengio, "Interpretable Convolutional Filters with SincNet", in Proc. of NIPS@IRASL 2018.

GitHub https://github.com/mravanelli/pytorch-kaldi

arXiv.org
Speaker Recognition Results

**Training:** 12-15 seconds for each speaker  
**Test:** short sentences (from 2 to 6 seconds)

### Speaker Identification Performance:

<table>
<thead>
<tr>
<th></th>
<th>TIMIT</th>
<th>LibriSpeech</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN-MFCC</td>
<td>0.99</td>
<td>2.02</td>
</tr>
<tr>
<td>CNN-FBANK</td>
<td>0.86</td>
<td>1.55</td>
</tr>
<tr>
<td>CNN-Raw</td>
<td>1.65</td>
<td>1.00</td>
</tr>
<tr>
<td>SINCNET</td>
<td>0.85</td>
<td>0.96</td>
</tr>
</tbody>
</table>

*Tab 1: Classification Error Rate (CER%) on TIMIT (462 spks) and Librispeech (2484 spks).*

### Speaker Verification Performance:

<table>
<thead>
<tr>
<th></th>
<th>d-vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN-MFCC</td>
<td>0.88</td>
</tr>
<tr>
<td>CNN-FBANK</td>
<td>0.60</td>
</tr>
<tr>
<td>CNN-Raw</td>
<td>0.58</td>
</tr>
<tr>
<td>SINCNET</td>
<td>0.51</td>
</tr>
</tbody>
</table>

*Tab 2: Equal Error Rate (SER%) on Librispeech with the d-vector approach*

I-Vector EER = 1.1 %

https://github.com/mravanelli/SincNet

M. Ravanelli, Y. Bengio, "Speaker Recognition from raw waveform with SincNet", in Proc. of SLT 2018.
### Speech Recognition Results

<table>
<thead>
<tr>
<th></th>
<th>TIMIT</th>
<th>DIRHA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-FBANK</td>
<td>18.3</td>
<td>40.1</td>
</tr>
<tr>
<td>CNN-Raw waveform</td>
<td>18.3</td>
<td>40.5</td>
</tr>
<tr>
<td>SincNet-Raw waveform</td>
<td><strong>18.0</strong></td>
<td><strong>38.2</strong></td>
</tr>
</tbody>
</table>

Tab 3: Speech Recognition error rates (%) obtained for TIMIT and for the DIRHA dataset.

😊 SincNet works for ASR as well!

😊 SincNet works in noisy and reverberant conditions

[GitHub](https://github.com/mravanelli/pytorch-kaldi)

M. Ravanelli, Y. Bengio, "Interpretable Convolutional Filters with SincNet", in Proc. of NIPS@IRASL 2018.
Conclusion

Summary:

• SincNet has shown **promising** on speaker and speech recognition task.

• Analysis of the SincNet filters reveals that the learned filter-bank is tuned to address the specific task.

M. Ravanelli, Y. Bengio, "Speaker Recognition from raw waveform with SincNet", in Proc. of SLT 2018.

https://github.com/mravanelli/SincNet
Towards Unsupervised Learning of Speech Representations

Mirco Ravanelli

Mila
Outline

• Why unsupervised learning?

• Self-Supervised Learning

• Local Info Max (LIM)

• Problem-Agnostic Speech Encoder (PASE)

• Conclusion
Why Unsupervised Learning?

- **Deep learning** = Learning hierarchical representations.
Why Unsupervised Learning?

Unsupervised Learning = Learning Without a Teacher

- Unsupervised learning is actually how humans/animals learn.
- Rapid generalization to a new task.
- Targets/rewards can be difficult/expensive to obtain or define.
Why Unsupervised Learning?

Some popular approaches:

• Deep Belief Nets
• Autoencoders
• Variational Autoencoders
• Generative Adversarial Networks
Self-Supervised Learning

A field that is gaining popularity in computer vision is self-supervised learning.

**Self-supervised Learning** = the supervision is extracted from the signal itself.

- In general, this is performed by applying known transforms to the input data and using the resulting outcomes as targets.

**Relative Positioning**

*(Example:)*

- (Doersch et al., ICCV 2015)

**Colourization**

*(Zhang et al., ECCV 2016)*

**Correct Rotation**

*(Gidaris et al., ICLR 2018)*
Self-Supervised Learning

Some recent works have used self-supervised learning to learn speech representations:

![Diagram showing encoder, resampling, decoder, and discriminator]

We learn independent features for speech separation.

P. Brakel, Y. Bengio, “Learning independent features with adversarial nets for non-linear ICA”, 2017

Constructive Predicting Coding (CPC)

We learn features that are “predictable” about the future.

A. van den Oord, Y. Li, O. Vinyals, “Representation Learning with Contrastive Predictive Coding”, 2018
Self-Supervised Learning

Self-supervised learning on speech: why is challenging?

- High-dimensionality
- Long sequences
- Variable-length
- Complex hierarchical structure

![Diagram showing the hierarchy of speech components from raw audio samples to meaning.](image)
Local Info Max (LIM)

**Goal:** Learn good speaker representations with Mutual Information.

\[
MI(z_1, z_2) = \int_{z_1} \int_{z_2} p(z_1, z_2) \log \left( \frac{p(z_1, z_2)}{p(z_1)p(z_2)} \right) dz_1 dz_2 \\
= D_{KL} (p(z_1, z_2) \mid \mid p(z_1)p(z_2))
\]

- MI can capture complex **non-linear relationships** between random variables.
- MI is **difficult to compute** in high-dimensional spaces.
- MINE (*Belghazi, 2018*) found that it is possible to maximize or minimize the MI within a framework that closely resembles that of GANs.

M. Ravanelli, Y. Bengio, “Learning Speaker Representations with Mutual Information”, 2018
Local Info Max (LIM)

**Sampling strategy:**
1. Choose a random chunk from a random sentence $C_a$ (anchor).
2. Choose another random chunk from the same sentence $C_p$ (positive).
3. Choose a random chunk from another random sentence $C_n$ (negative).

**The game we play:**
1. Process $C_a$, $C_p$, $C_n$ with an encoder.
   - $(Z_a, Z_p)$: sample from the joint distribution (positive sample).
   - $(Z_a, Z_n)$: sample from the product of marginal distribution (negative sample).
2. We feed the discriminator with positive or negative samples.
3. The discriminator should figure out if their two inputs come from the **same** or **different** sentences.
Local Info Max (LIM)

- The discriminator loss is set to maximize the mutual information MI.
- Different choices are possible (MINE, Info-NCE, BCE).
- Encoder and discriminator are jointly trained.
- Cooperative game, not adversarial!
- The representations discovered by the encoder can be later used for the supervised speaker recognition task.

M. Ravanelli, Y. Bengio, “Learning Speaker Representations with Mutual Information”, 2018
Local Info Max (LIM)

- **Loss Comparison**

<table>
<thead>
<tr>
<th>Loss</th>
<th>Librispeech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triplet Loss</td>
<td>1.33%</td>
</tr>
<tr>
<td>MINE</td>
<td>0.94%</td>
</tr>
<tr>
<td>Info-NCE</td>
<td>0.82%</td>
</tr>
<tr>
<td>Binary Cross Entropy (BCE)</td>
<td>0.75%</td>
</tr>
</tbody>
</table>

*Tab. 1 Classification Error Rate obtained the speaker-id tasks (2484 spks) using LIM with various losses (the lower the better)*.

**Insights:**
- Mutual information losses (MINE, Info-NCE, BCE) **outperform** the triplet loss.
- Better embeddings can thus be derived with a divergence measure more meaningful than the simple **cosine distance** used in triplet loss.
- The best performance is achieved with the standard binary **cross-entropy**.

- Similar to *(D. Hjelm et. al, 2018)*, we have observed that this **bounded** metric is more **stable** and **easier** to optimize.

M. Ravanelli, Y. Bengio, “Learning Speaker Representations with Mutual Information”, 2018
Local Info Max (LIM)

- **Speaker Identification on Librispeech (2484 spks)**

<table>
<thead>
<tr>
<th>Model</th>
<th>Clean</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>0.80</td>
<td>17.1</td>
</tr>
<tr>
<td>LIM (Frozen)</td>
<td>0.75</td>
<td>15.2</td>
</tr>
<tr>
<td>LIM (FineTuned)</td>
<td>0.56</td>
<td>9.6</td>
</tr>
<tr>
<td>LIM (joint Training)</td>
<td>0.52</td>
<td>9.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Insights:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- LIM outperforms a <strong>fully-supervised classifier</strong>.</td>
</tr>
<tr>
<td>- The gap becomes more evident when <strong>pre-training</strong> the encoder with LIM and fine-tune it with the classifier (<strong>LIM-FineTuned</strong>).</td>
</tr>
</tbody>
</table>

Tab. 2 Classification Error Rate (CER%) obtained on speaker-id in clean and reverberant conditions (the lower the better).

- **Jointly training** from scratch **encoder**, **discriminator**, and **classifier** (**LIM-joint Training**) yields the best performance.

M. Ravanelli, Y. Bengio, “Learning Speaker Representations with Mutual Information”, 2018
Local Info Max (LIM)

**Strengths**

- LIM highlights high-quality speaker representations.
- LIM is simple and efficient (local information only).

**Issue**

- The LIM representations are very task-specific.

All previous approaches were based on single self-supervised tasks only.

*But, Is it really possible to capture the complex structure of speech with a single-tasks only?*

The risk is to focus on “specific” aspects of the speech signal only.
Problem-agnostic Speech Encoder (PASE)

**Idea:** jointly tackle multiple self-supervised tasks

Where an ensemble of neural networks must cooperate to discover good speech representations.

**Intuition:**

- Each self-supervised task brings a different “view” on the speech signal.

- A consensus across these different “views” is needed, imposing several “soft constraints” to the representation.

- This way, our approach is more likely to learn general, robust, and transferable features.

S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, Y. Bengio "Learning Problem-agnostic Speech Representations from Multiple Self-supervised Tasks“, 2019.
Problem-agnostic Speech Encoder (PASE)

https://github.com/santi-pdp/pase
Problem-agnostic Speech Encoder (PASE)

**Regression Tasks**
- **Prosody**: we predict fundamental freq., voiced/unvoiced probability, zero-crossing rate, and energy.
- **Waveform**: we predict the input waveform in an auto-encoder fashion.
- **Log power spectrum (LPS)**: we compute it using 1024 frequency bins.
- **Mel-frequency cepstral coefficients (MFCC)**: we extract 20 coefficients from 40 mel filter banks.
- **Prosody**: we predict fundamental freq., voiced/unvoiced probability, zero-crossing rate, and energy.

**Known speech representations**
- Waveform
- LPS
- MFCC
- Prosody

We inject prior knowledge into the encoder!
1. We sample three speech chunks (i.e., anchor, positive, negative chunks) according to a predefined strategy.

2. We process all the chunks with PASE.

3. Given the anchor, the discriminator should figure out if the other input is the positive or the negative one.

Intuition:
- Positive + Anchor => “Close”
- Negative + Anchor => “Distant”
Problem-agnostic Speech Encoder (PASE)

Local Info Max (LIM)

This way we highlight speaker identities.

Global Info Max (GIM) \( (Hjelm, 2018) \)

This way we highlight “global” information.

---

S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, Y. Bengio "Learning Problem-agnostic Speech Representations from Multiple Self-supervised Tasks", 2019.
Problem-agnostic Speech Encoder (PASE)

Sequence Predictive Coding (SPC)

Sampling strategy:
1. Choose a random chunk from a random sentence (anchor).
2. Choose another random chunk from the future of the same sentence (positive).
3. Choose another random chunk from the past of the same sentence (negative).

This way we want to capture longer contextual information.

Similar to CPC (van den Oord, 2018), but samples here are all from the same sentence.

A. van den Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding” 2018
Problem-agnostic Speech Encoder (PASE)

- The total loss that is computed as the average of each worker cost.
- Encoder and workers are jointly trained (using Librispeech).
- The encoder parameters will be updated pointing to a direction that is a compromise among all the worker losses.
- The representations discovered by the encoder can be later used for supervised classification:
  - **PASE-Frozen**: we keep the encoder frozen during supervised training.
  - **PASE-FineTuned**: fine-Tuning the encoder during supervised training.
Problem-agnostic Speech Encoder (PASE)

- **Which self-supervised tasks are needed?**

Table 1: Accuracies using PASE and an MLP as classifier. Rows below the “all workers” model report absolute accuracy loss when discarding each worker for self-supervised training.

<table>
<thead>
<tr>
<th>Model</th>
<th>Classification accuracy [%]</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Speaker-ID (VCTK)</td>
<td>Emotion (INTERFACE)</td>
<td>ASR (TIMIT)</td>
<td></td>
</tr>
<tr>
<td>PASE (All workers)</td>
<td>97.5</td>
<td>88.3</td>
<td>81.1</td>
<td></td>
</tr>
<tr>
<td>- Waveform</td>
<td>-1.3</td>
<td>-3.9</td>
<td>-0.3</td>
<td></td>
</tr>
<tr>
<td>- LPS</td>
<td>-1.5</td>
<td>-5.3</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>- MFCC</td>
<td>-2.4</td>
<td>-3.2</td>
<td>-0.7</td>
<td></td>
</tr>
<tr>
<td>- Prosody</td>
<td>-0.5</td>
<td>-5.3</td>
<td>-0.1</td>
<td></td>
</tr>
<tr>
<td>- LIM</td>
<td>-0.8</td>
<td>-1.3</td>
<td>-0.0</td>
<td></td>
</tr>
<tr>
<td>- GIM</td>
<td>-0.6</td>
<td>-0.5</td>
<td>-0.3</td>
<td></td>
</tr>
<tr>
<td>- SPC</td>
<td>-0.4</td>
<td>-1.6</td>
<td>-0.0</td>
<td></td>
</tr>
</tbody>
</table>

**Insights:**

- No worker is dispensable (the best results are achieved with all workers).
- Some workers are helpful for all the speech tasks (e.g., Waveform, LPS, and MFCC).
- Others turn out to be more application-dependent (e.g., Prosody, LIM, GIM, SPC).

S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, Y. Bengio "Learning Problem-agnostic Speech Representations from Multiple Self-supervised Tasks", 2019.
Problem-agnostic Speech Encoder (PASE)

• **Comparison with Standard Features**

<table>
<thead>
<tr>
<th>Model</th>
<th>Classification accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Speaker-ID (VCTK)</td>
</tr>
<tr>
<td></td>
<td>MLP</td>
</tr>
<tr>
<td>MFCC</td>
<td>96.9</td>
</tr>
<tr>
<td>FBANK</td>
<td>98.4</td>
</tr>
<tr>
<td>PASE-Supervised</td>
<td>97.0</td>
</tr>
<tr>
<td>PASE-Frozen</td>
<td>97.3</td>
</tr>
<tr>
<td>PASE-FineTuned</td>
<td><strong>99.3</strong></td>
</tr>
</tbody>
</table>

**Insights:**

• PASE features are **often better** than MFCCs and FBANKs, even when freezing the encoder (PASE-Frozen).

• The improvement is more evident when **pre-training the encoder and fine-tuning it** with the supervised task of interest (PASE-FineTuned).

• This approach consistently provides the **best performance over all the tasks and classifiers** considered here.
Problem-agnostic Speech Encoder (PASE)

- **Transferability**

Table 3: Word error rate (WER) obtained on the DIRHA corpus.

<table>
<thead>
<tr>
<th>System</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFCC</td>
<td>35.8</td>
</tr>
<tr>
<td>FBANK</td>
<td>34.0</td>
</tr>
<tr>
<td>PASE-Supervised</td>
<td>33.5</td>
</tr>
<tr>
<td>PASE-Frozen</td>
<td>32.5</td>
</tr>
<tr>
<td><strong>PASE-FineTuned</strong></td>
<td><strong>29.8</strong></td>
</tr>
</tbody>
</table>

**Insights:**

- Finally, we study the exportability of PASE to acoustic conditions that are very different from the clean one used to train it.

- Interestingly, PASE clearly outperforms the other systems even if it is not specifically designed to address noise and reverberation.

S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, Y. Bengio "Learning Problem-agnostic Speech Representations from Multiple Self-supervised Tasks", 2019.
Conclusion

• PASE is neural speech encoder trained using multiple self-supervised tasks.

• The discovered embeddings turn out to carry important information related to, at least, speaker identity, phonemes, and emotional cues.

• It is designed to be efficient and fully parallelizable.

• PASE can be used a standard feature extractor or as a pre-trained model (as commonly done in computer vision).

• It can be seen as a first step towards a universal speech feature extractor.

https://github.com/santi-pdp/pase

S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, Y. Bengio "Learning Problem-agnostic Speech Representations from Multiple Self-supervised Tasks", 2019.
Other Research Directions: SincNet

- SincNet is a convolutional architecture for efficiently processing raw audio samples.

**Standard CNN:**
\[ y[n] = x[n] \ast h[n] \]

- We perform the convolution with a set of FIR filters.

- We learn all the taps of each filter.

**SincNet:**
\[ y[n] = x[n] \ast g[n, \theta] \]

- We perform the convolution with sinc-based kernels that implement band-pass filters.

- We learn only low and high cut-off frequencies of each filter.

[GitHub](https://github.com/mravanelli/SincNet)
M. Ravanelli, Y. Bengio, "Speaker Recognition from raw waveform with SincNet", in Proc. of SLT 2018.
Other Research Directions

Cooperative Neural Networks

M. Ravanelli, T. Parcollet, Y. Bengio, "The PyTorch-Kaldi Speech Recognition Toolkit", in Proc. of ICASSP 2019.

The PyTorch-Kaldi Project

https://github.com/mravanelli/pytorch-kaldi

M. Ravanelli, T. Parcollet, Y. Bengio, "The PyTorch-Kaldi Speech Recognition Toolkit", in Proc. of ICASSP 2019.
Montreal: the silicon valley of AI
Cooperative Networks of Deep Neural Networks

Cooperative Networks of DNNs

DNN Competition:

- **Competition** played a crucial role for the evolution of living forms.
- Several deep learning systems are inspired by the “principle of competition”

Cooperative Networks of DNNs

What about cooperation?

• **Cooperation** played a crucial role for the evolution of living forms as well.

• Can we train **multiple DNNs** that learn how to cooperate?

• Cooperation can be helpful to **counteract uncertainty**.

• This paradigm can be exploited to solve **challenging problems**.

• **Distant speech recognition** represents the natural application field for this approach!

---

Distant Speech Recognition (DSR)

Applications:
- Home Automation
- Smart TV
- Meeting Transcriptions
- Healthcare
- Robotics

DSR is very challenging due to noise $n(t)$ and reverberation $h(t)$:

$$y(t) = x(t) \ast h(t) + n(t)$$

Cooperative Networks of DNNs

Breaking the pipeline

- Lack of matching
- Lack of communication

- Improved matching
- Full-communication

Cooperative Networks of DNNs
How we train it?

Input features $x$ → SPEECH ENHANCEMENT $SE$-DNN → Phone predictions $\hat{y}_{SR}$

Unfold

Level N

$\hat{x}_{SE_N}$

$SE$ DNN-N

$\hat{y}_{SR_N}$

$SR$ DNN-N

$x_{clean}$

$NLL_N$

$y_{SR}$

Level 1

$\hat{x}_{SE_1}$

$SE$ DNN-1

$\hat{y}_{SR_1}$

$SR$ DNN-1

$x_{clean}$

$NLL_1$

$y_{SR}$

Level 0

$\hat{x}_{SE_0}$

$SE$ DNN-o

$\hat{y}_{SR_0}$

$SR$ DNN-o

$x_{clean}$

$NLL_0$

$y_{SR}$

Input features $x$ → SPEECH ENHANCEMENT $SE$-DNN → Phone predictions $\hat{y}_{SR}$
## Network of DNNs

### ASR results

<table>
<thead>
<tr>
<th>Systems</th>
<th>TIMIT Rev</th>
<th>DIRHA WSJ Rev</th>
<th>DIRHA WSJ Rev+Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single DNN</td>
<td>31.9</td>
<td>8.1</td>
<td>14.3</td>
</tr>
<tr>
<td>Joint SE-SR training</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network of DNNs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Features** → **Context-Dependent targets**
## Network of DNNs

### ASR results

<table>
<thead>
<tr>
<th>Systems</th>
<th>TIMIT Rev</th>
<th>DIRHA WSJ Rev</th>
<th>DIRHA WSJ Rev+Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single DNN</td>
<td>31.9</td>
<td>8.1</td>
<td>14.3</td>
</tr>
<tr>
<td>Joint SE-SR training</td>
<td>29.1</td>
<td>7.8</td>
<td>12.7</td>
</tr>
<tr>
<td>Network of DNN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

# Network of DNNs

## ASR results

<table>
<thead>
<tr>
<th>Systems</th>
<th>TIMIT Rev</th>
<th>DIRHA WSJ Rev</th>
<th>DIRHA WSJ Rev+Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single DNN</td>
<td>31.9</td>
<td>8.1</td>
<td>14.3</td>
</tr>
<tr>
<td>Joint SE-SR training</td>
<td>29.1</td>
<td>7.8</td>
<td>12.7</td>
</tr>
<tr>
<td>Network of DNNs</td>
<td>28.7</td>
<td>7.6</td>
<td>12.3</td>
</tr>
</tbody>
</table>
Conclusion

• Networks of DNNs can counteract noise with cooperation.

• To better exploit this paradigm further studies are needed in the future.

• We can apply this paradigm to many other fields!

The PyTorch-Kaldi Speech Recognition Toolkit

Mirco Ravanelli
What is PyTorch-Kaldi?

- PyTorch-Kaldi is an open-source toolkit for developing state-of-the-art DNN/HMM speech recognition systems.
- The PyTorch-Kaldi project aims to bridge the gap between the Kaldi and the PyTorch toolkits.
- It inherits the efficiency of Kaldi and the flexibility of PyTorch.
- The toolkit is released under a Creative Commons Attribution 4.0 International license.

GitHub: https://github.com/mravanelli/pytorch-kaldi

M. Ravanelli, T. Parcollet, Y. Bengio, “The PyTorch Kaldi Speech Recognition Toolkit”, 2018
It’s more than a simple interface...

PyTorch-Kaldi is not only a simple interface between these toolkits, but it embeds several useful features and utilities for developing modern speech recognizers:

• Several **pre-implemented models** (MLP, CNN, LSTM, GRU, Li-GRU, SincNet).

• Easy and **flexible configuration files**.

• Natural implementation of **complex models** based on multiple features, labels, and neural architectures.

GitHub [https://github.com/mravanelli/pytorch-kaldi](https://github.com/mravanelli/pytorch-kaldi)

M. Ravanelli, T. Parcollet, Y. Bengio, “The PyTorch Kaldi Speech Recognition Toolkit”, 2018
It’s more than a simple interface...

- Easy plug-in of **user-defined models**.

```python
class my_NN(nn.Module):
    def __init__(self, options):
        super(my_NN, self).__init__()
        # Definition of Model Parameters
        # Parameter Initialization

    def forward(self, minibatch):
        # Definition of Model Computations
        return [output_prob]
```

GitHub  https://github.com/mravanelli/pytorch-kaldi

M. Ravanelli, T. Parcollet, Y. Bengio, “The PyTorch Kaldi Speech Recognition Toolkit", 2018
It’s more than a simple interface...

- Designed to work locally or on **HPC clusters**.

- **Automatic recovery** from the last processed chunk.

- **Multi-GPU** training.

- **Easy hyperparameter tuning**.

- **Rich Documentation** with tutorials

GitHub  https://github.com/mravanelli/pytorch-kaldi

M. Ravanelli, T. Parcollet, Y. Bengio, “The PyTorch Kaldi Speech Recognition Toolkit”, 2018
PyTorch-Kaldi Architecture

GitHub: https://github.com/mravanelli/pytorch-kaldi

M. Ravanelli, T. Parcollet, Y. Bengio, “The PyTorch Kaldi Speech Recognition Toolkit”, 2018
## Baselines

**Table 1:** PER(%) obtained for the test set of TIMIT with various neural architectures.

<table>
<thead>
<tr>
<th></th>
<th>MFCC</th>
<th>FBANK</th>
<th>fMLLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>18.2</td>
<td>18.7</td>
<td>16.7</td>
</tr>
<tr>
<td>RNN</td>
<td>17.7</td>
<td>17.2</td>
<td>15.9</td>
</tr>
<tr>
<td>LSTM</td>
<td>15.1</td>
<td>14.3</td>
<td>14.5</td>
</tr>
<tr>
<td>GRU</td>
<td>16.0</td>
<td>15.2</td>
<td>14.9</td>
</tr>
<tr>
<td>Li-GRU</td>
<td>15.3</td>
<td>14.6</td>
<td><strong>14.2</strong></td>
</tr>
</tbody>
</table>

**Table 2:** PER(%) obtained on TIMIT when progressively applying some techniques implemented within PyTorch-Kaldi.

<table>
<thead>
<tr>
<th></th>
<th>RNN</th>
<th>LSTM</th>
<th>GRU</th>
<th>Li-GRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>16.5</td>
<td>16.0</td>
<td>16.6</td>
<td>16.3</td>
</tr>
<tr>
<td>+ Incr. Seq. length</td>
<td>16.6</td>
<td>15.3</td>
<td>16.1</td>
<td>15.4</td>
</tr>
<tr>
<td>+ Recurrent Dropout</td>
<td>16.4</td>
<td>15.1</td>
<td>15.4</td>
<td>14.5</td>
</tr>
<tr>
<td>+ Batch Normalization</td>
<td>16.0</td>
<td>14.8</td>
<td>15.3</td>
<td>14.4</td>
</tr>
<tr>
<td>+ Monophone Reg.</td>
<td>15.9</td>
<td>14.5</td>
<td>14.9</td>
<td><strong>14.2</strong></td>
</tr>
</tbody>
</table>
Baselines

Table 3: PER(%) obtained by combining multiple neural networks and acoustic features.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Features</th>
<th>PER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Li-GRU</td>
<td>fMLLR</td>
<td>14.2</td>
</tr>
<tr>
<td>MLP+Li-GRU+MLP</td>
<td>MFCC+FBANK+fMLLR</td>
<td>13.8</td>
</tr>
</tbody>
</table>

Table 4: PER(%) obtained with standard convolutional and with the SincNet architectures.

<table>
<thead>
<tr>
<th>Model</th>
<th>Features</th>
<th>PER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>FBANK</td>
<td>18.3</td>
</tr>
<tr>
<td>CNN</td>
<td>Raw waveform</td>
<td>18.3</td>
</tr>
<tr>
<td>SincNet</td>
<td>Raw waveform</td>
<td>18.1</td>
</tr>
</tbody>
</table>
**Baselines**

**Table 5:** WER(%) obtained for the DIRHA, CHiME, and LibriSpeech (100h) datasets with various neural architectures.

<table>
<thead>
<tr>
<th></th>
<th>DIRHA</th>
<th>CHiME</th>
<th>LibriSpeech</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>26.1</td>
<td>18.7</td>
<td>6.5</td>
</tr>
<tr>
<td>LSTM</td>
<td>24.8</td>
<td>15.5</td>
<td>6.4</td>
</tr>
<tr>
<td>GRU</td>
<td>24.8</td>
<td>15.2</td>
<td>6.3</td>
</tr>
<tr>
<td>Li-GRU</td>
<td><strong>23.9</strong></td>
<td><strong>14.6</strong></td>
<td><strong>6.2</strong></td>
</tr>
</tbody>
</table>

[GitHub](https://github.com/mravanelli/pytorch-kaldi)

M. Ravanelli, T. Parcollet, Y. Bengio, “The PyTorch Kaldi Speech Recognition Toolkit", 2018
Conclusion and Future Work

• PyTorch-Kaldi is a novel toolkit to design state-of-the-art ASR systems.

• The project is still in its initial phase and we invite all potential contributors to participate in it.

• We hope to build a community of developers larger enough to progressively maintain, improve, and expand the functionalities of our current toolkit.

GitHub  https://github.com/mravanelli/pytorch-kaldi
arXiv.org  M. Ravanelli, T. Parcollet, Y. Bengio, “The PyTorch Kaldi Speech Recognition Toolkit”, 2018