Recent Advances in End-to-End Speech Recognition

Abbas Khosravani

Postdoc Researcher
Idiap Research Institute

Amirkabir Artificial Intelligent Summer Summit (AAISS)

August 2020

About Us

About Me

- **Education**
  - B.Sc. in Software Engineering
    - Shiraz University - 2005-2009
  - M.Sc. in Artificial Intelligence
    - Shiraz University - 2009-2011
  - Ph.D. in Artificial Intelligence
    - Amirkabir University of Technology - 2012-2018
- **Research Interests**
  - Speech Analysis & Modeling
  - Speech Recognition
  - Speaker Recognition & Diarization
  - Speech Enhancement

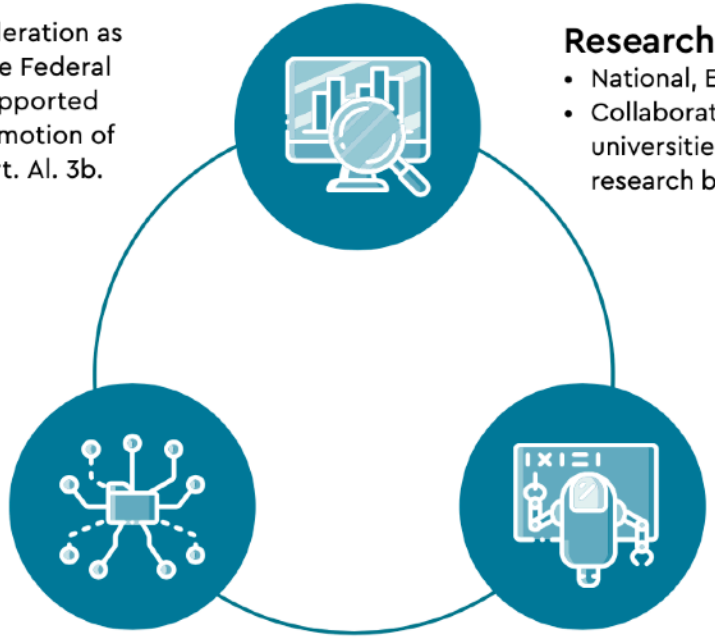
About Me

- **Scientific Experiences**
  - Postdoc Researcher
    - Idiap research Institute - 2019-Now
  - Research Scientist
    - Intelligent Voice - 04/2016-09/2017
  - Research Internship
    - CEA Nano-INNOV - Télécom SudParis - 04/2015-09/2015

Idiap Research Institute

3 missions

Idiap is recognized by the Confederation as part of the strategic domain of the Federal Institutes of Technology and is supported under the Federal Law on the Promotion of Research and Innovation (LERI), art. Al. 3b.



Research

- National, European, and worldwide
- Collaborations with prestigious universities and public and private research bodies

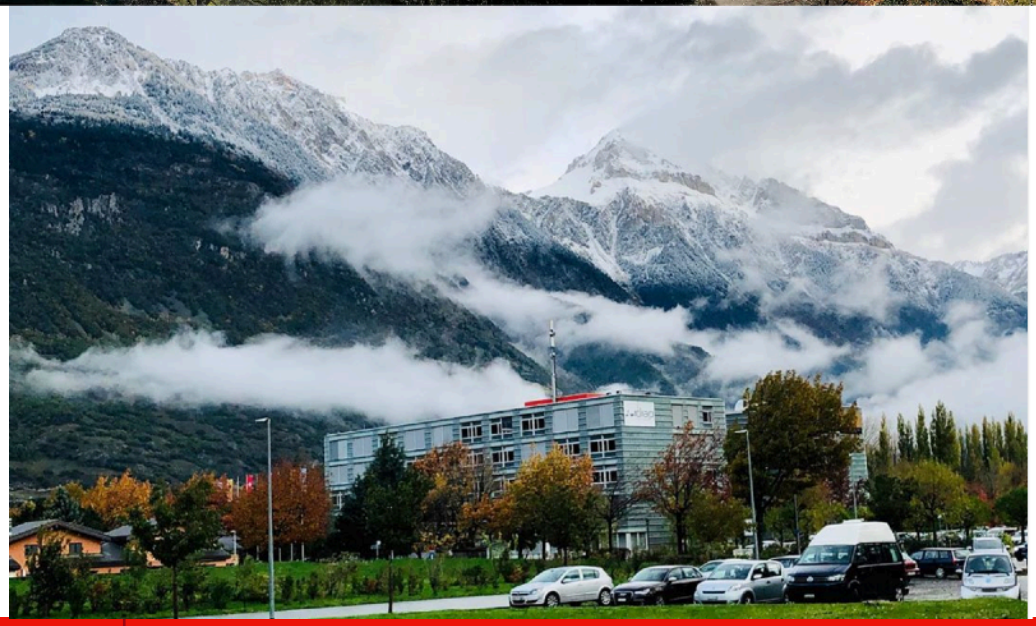
Innovation

- Technology transfer
- Start-up creation
- Dedicated incubator—IdeArk
- Patents, licenses, and open source

Training

- Numerous courses at EPFL and internally
- Master's in Artificial Intelligence incorporating guaranteed work experience
- Encouraging the next generation of young researchers

Idiap Research Institute



Human Resources

172 individuals in total
and more than 50 posts in the start-up ecosystem

Scientists

- 5 professors
- 1 senior scientist (MER)
- 14 permanent senior researchers
- 28 postdocs
- 48 research assistants
- 14 students
- 27 trainees/visitors

17.7% women

Engineers

- 15 research and development engineers
- 7 system engineers

9.1% women

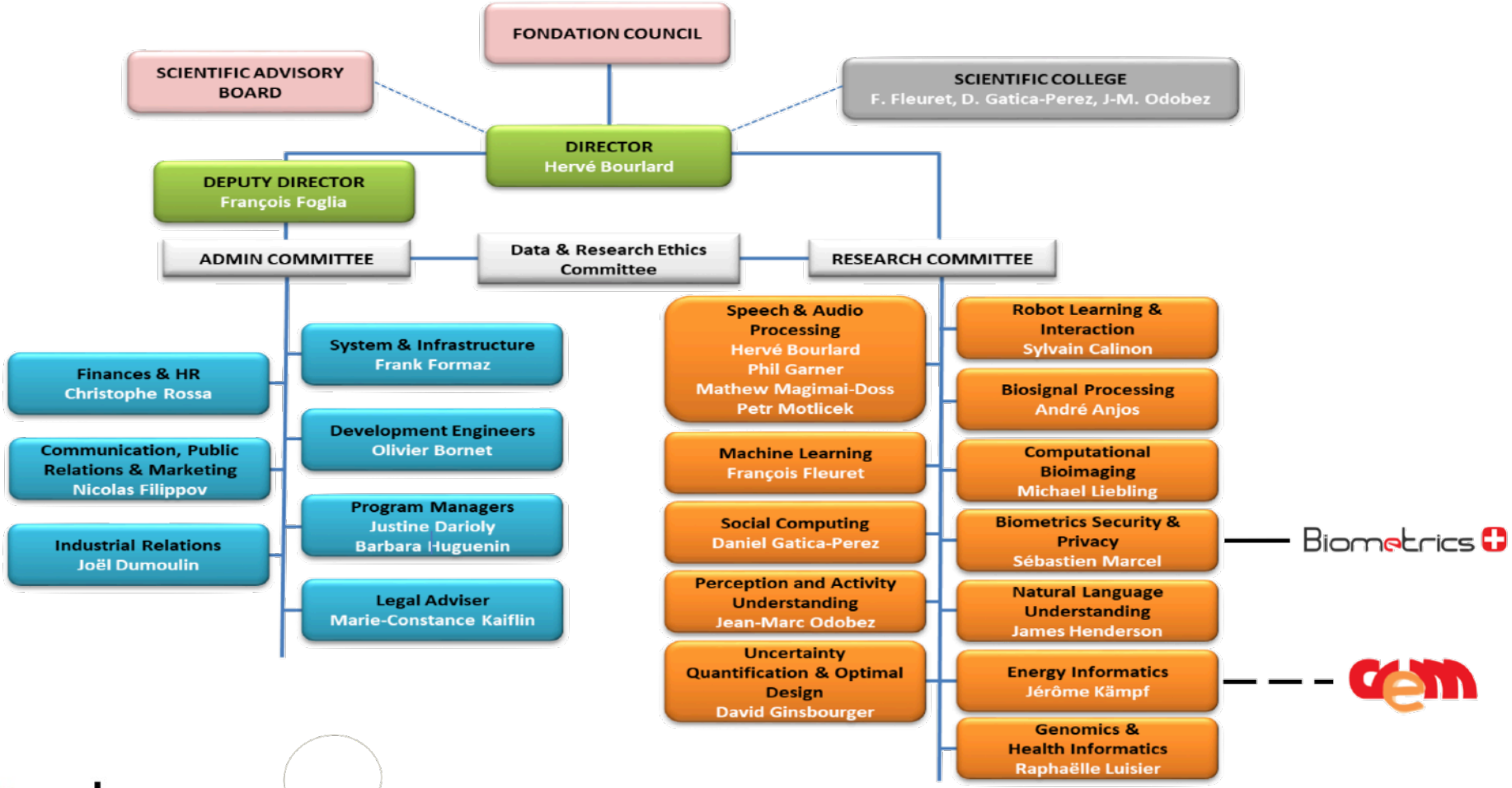
Administrative staff

- 13 administrative staff

61.5% women

20.3% of Idiap employees are women, **79.7%** are men

Organization Chart



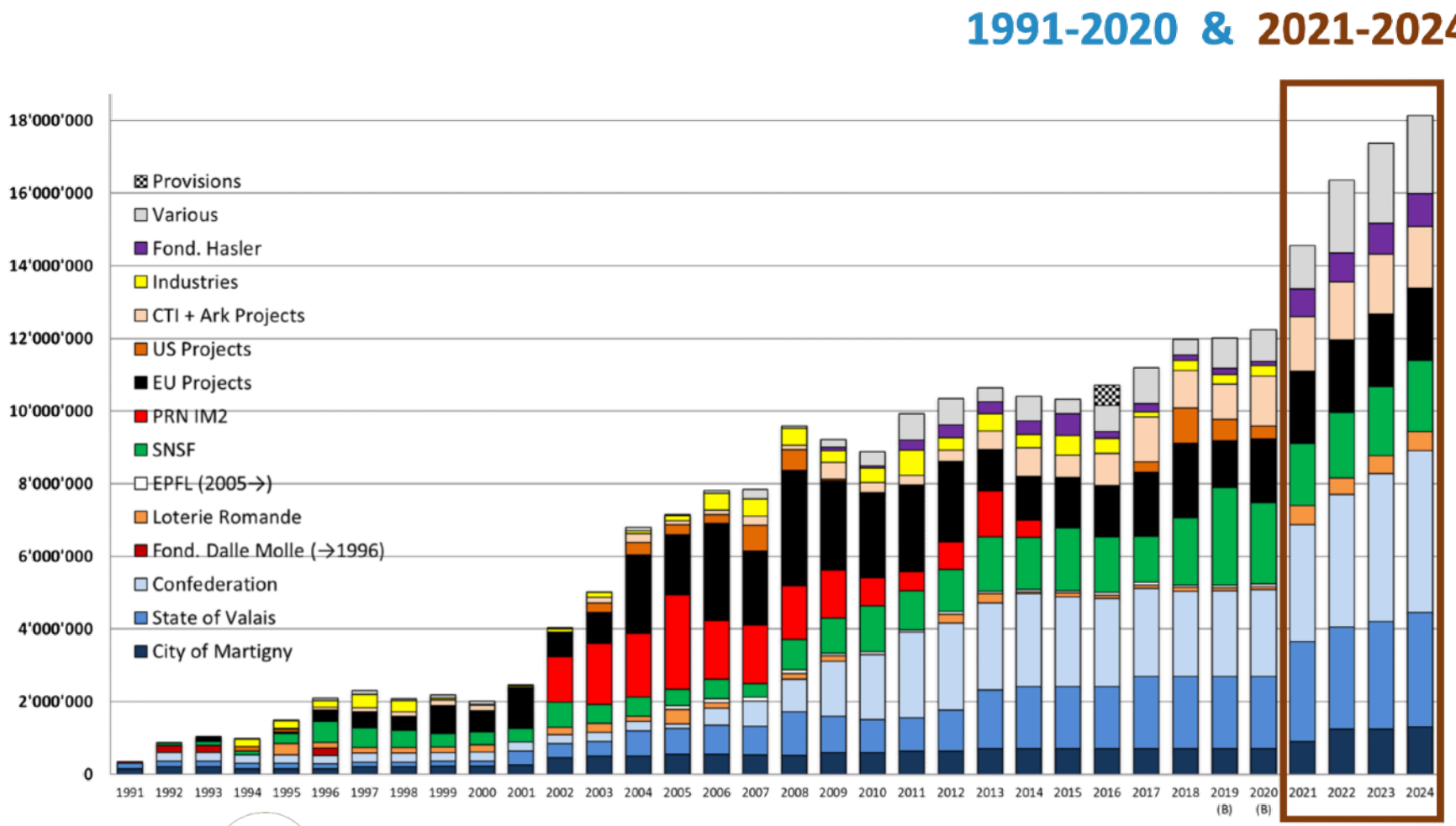
idiap RESEARCH INSTITUTE AAISS

Open Positions

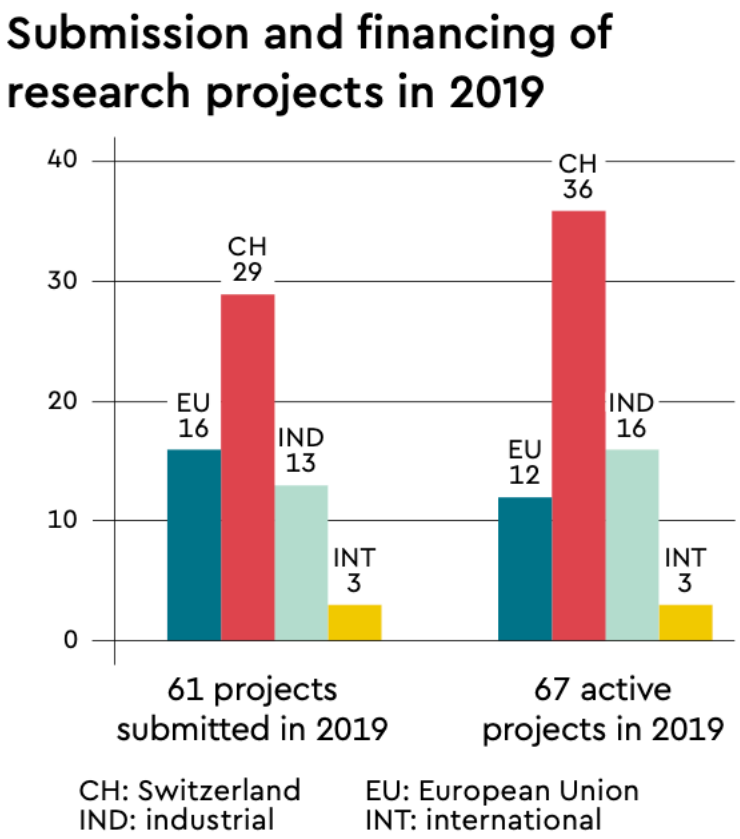
- **Job Opportunity:**
  - https://www.idiap.ch/en/join-us/job-opportunities
- **As of February 2019:**

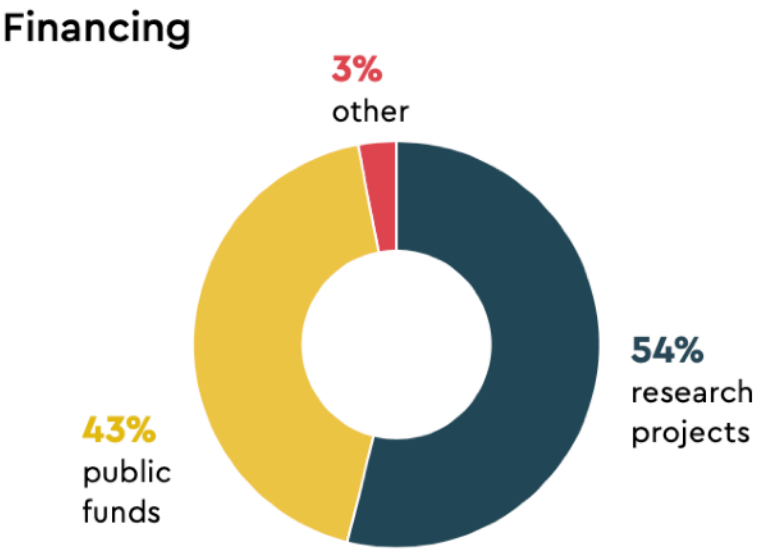
Year	Open positions	Applications received
2007	17	600
2008	40	847
2009	41	917
2010	42	848
2011	36	763
2012	42	951
2013	29	713
2014	30	936
2015	32	893
2016	43	1002
2017	39	1270
2018	40	1607

Budget & Funding Resource



Projects & Finance



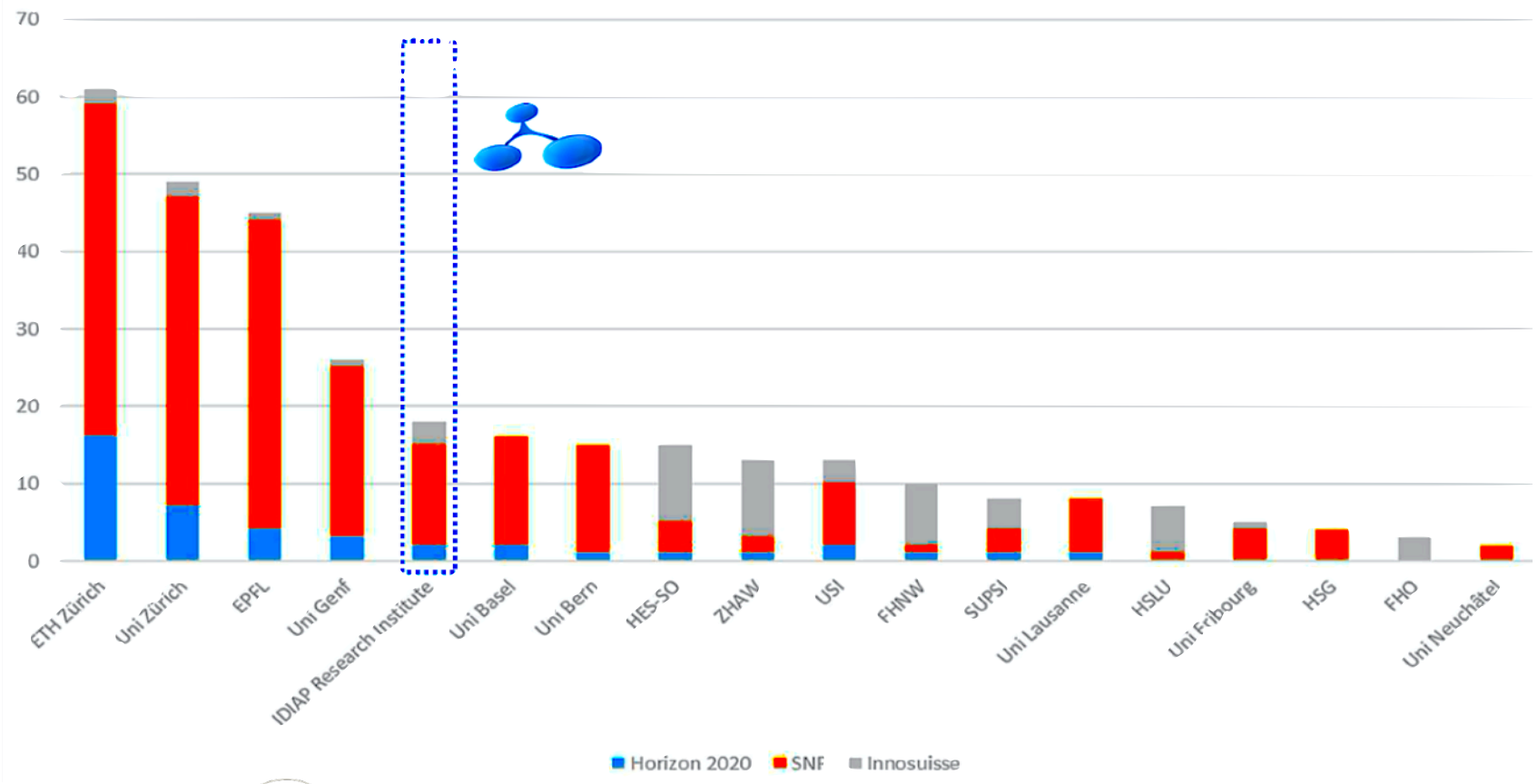


Salary

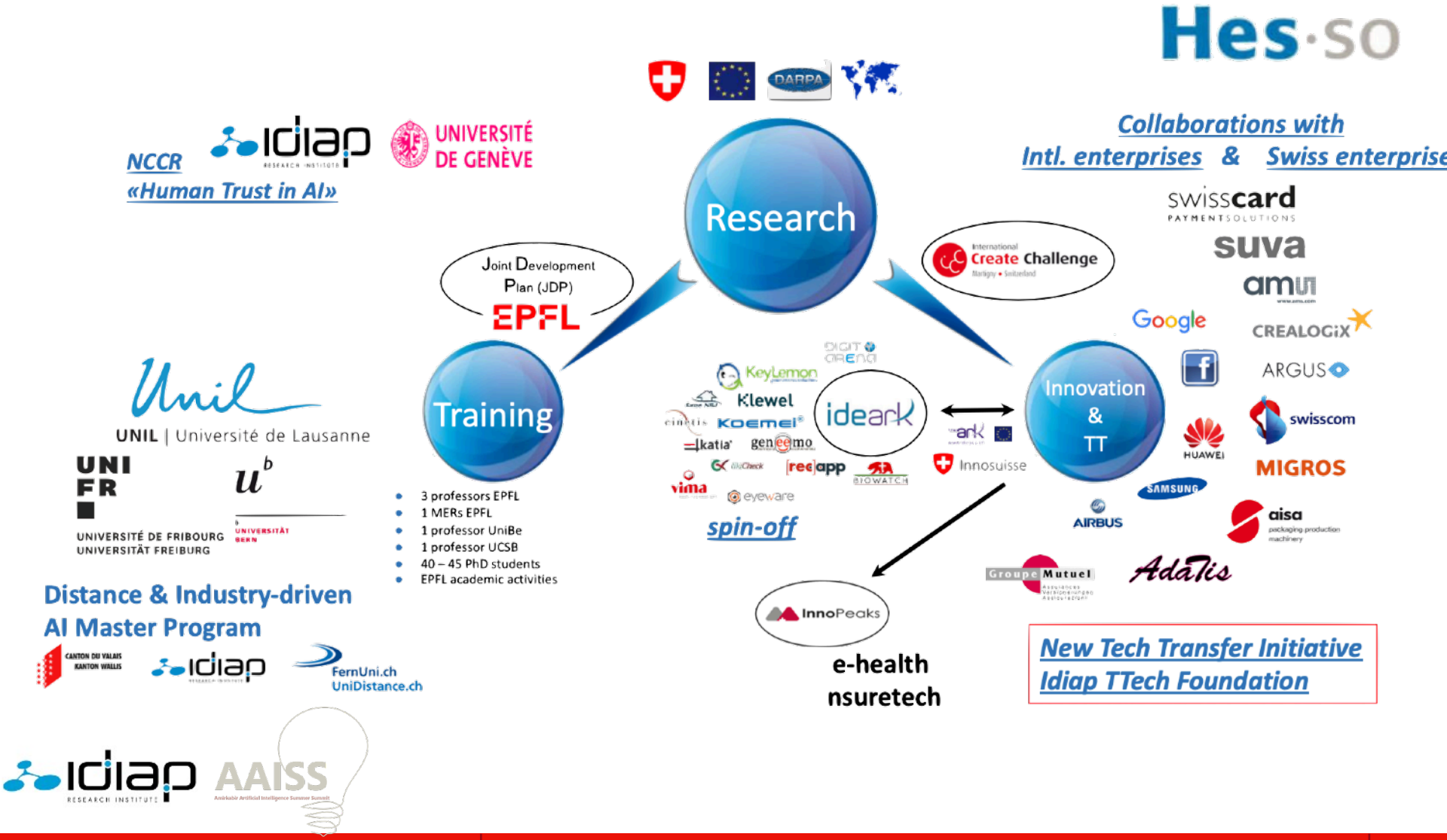
- **Salary Mass:**

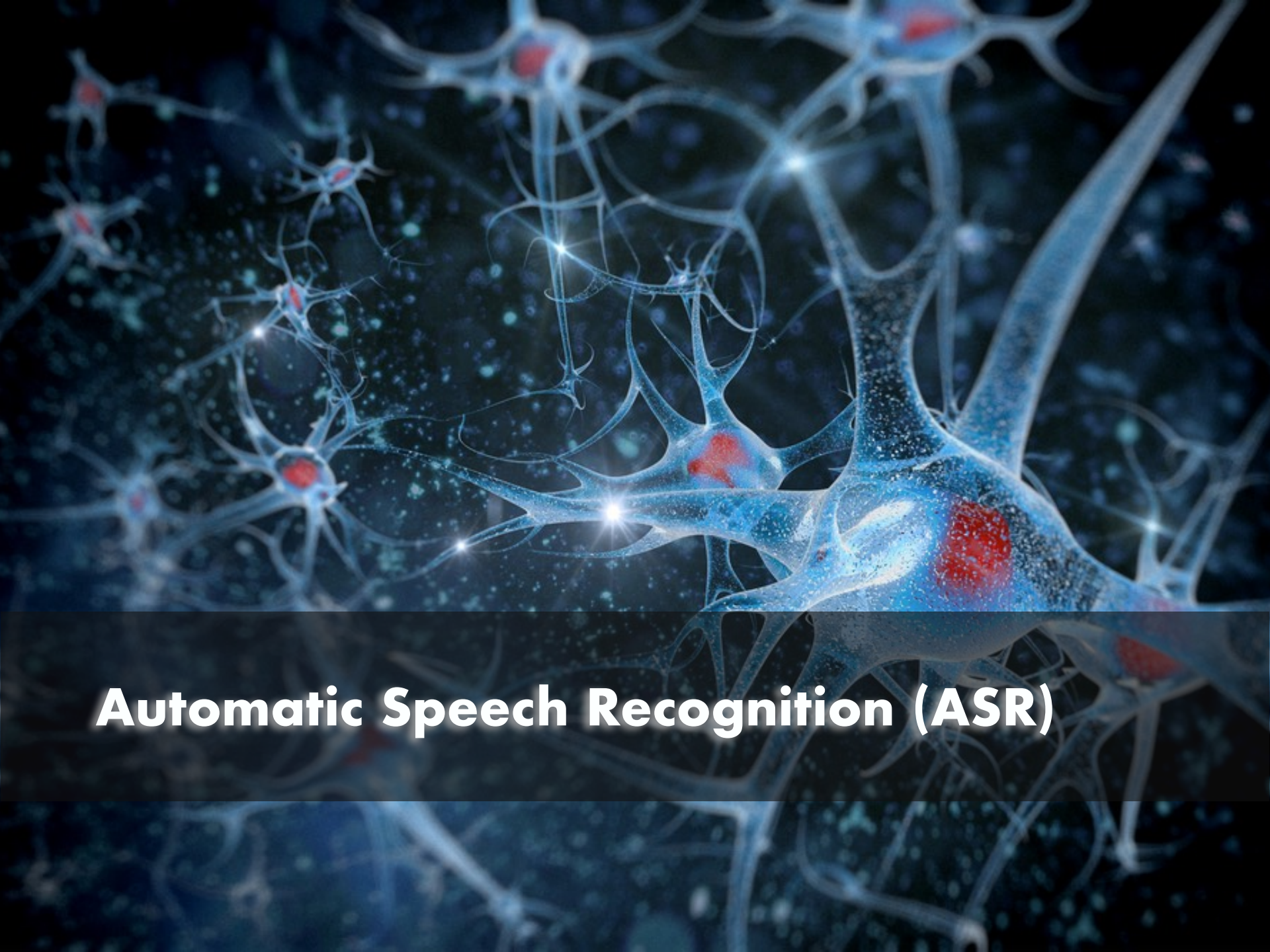
	Salary Mass		
Version 08.04.2019	**2018**	**2017**	**2016**
Administration	9.1%	10.6%	9.8%
Administration Service	3.3%	2.0%	2.7%
Administration System	7.2%	8.5%	8.2%
R&D Engineer	10.8%	14.1%	14.8%
Permanent Researcher	23.4%	19.9%	20.6%
PhD	19.7%	17.9%	18.2%
Post-Doc	20.4%	20.5%	18.0%
Stagiaire/Intern	2.4%	2.3%	1.0%
Research Associate	2.8%	4.2%	6.7%
Other	1.0%	0.0%	0.0%
Total	**100.0%**	**100.0%**	**100.0%**

Ranking


70
60
50
40
30
20
10
0
ETH Zürich
Uni Zürich
EPFL
Uni Genf
IDIAP Research Institute
Uni Basel
Uni Bern
HES-SO
ZHAW
USI
FHNW
SUPSI
Uni Lausanne
HSLU
Uni Fribourg
HSG
FHO
Uni Neuchâtel
Horizon 2020
SNF
Innosuisse

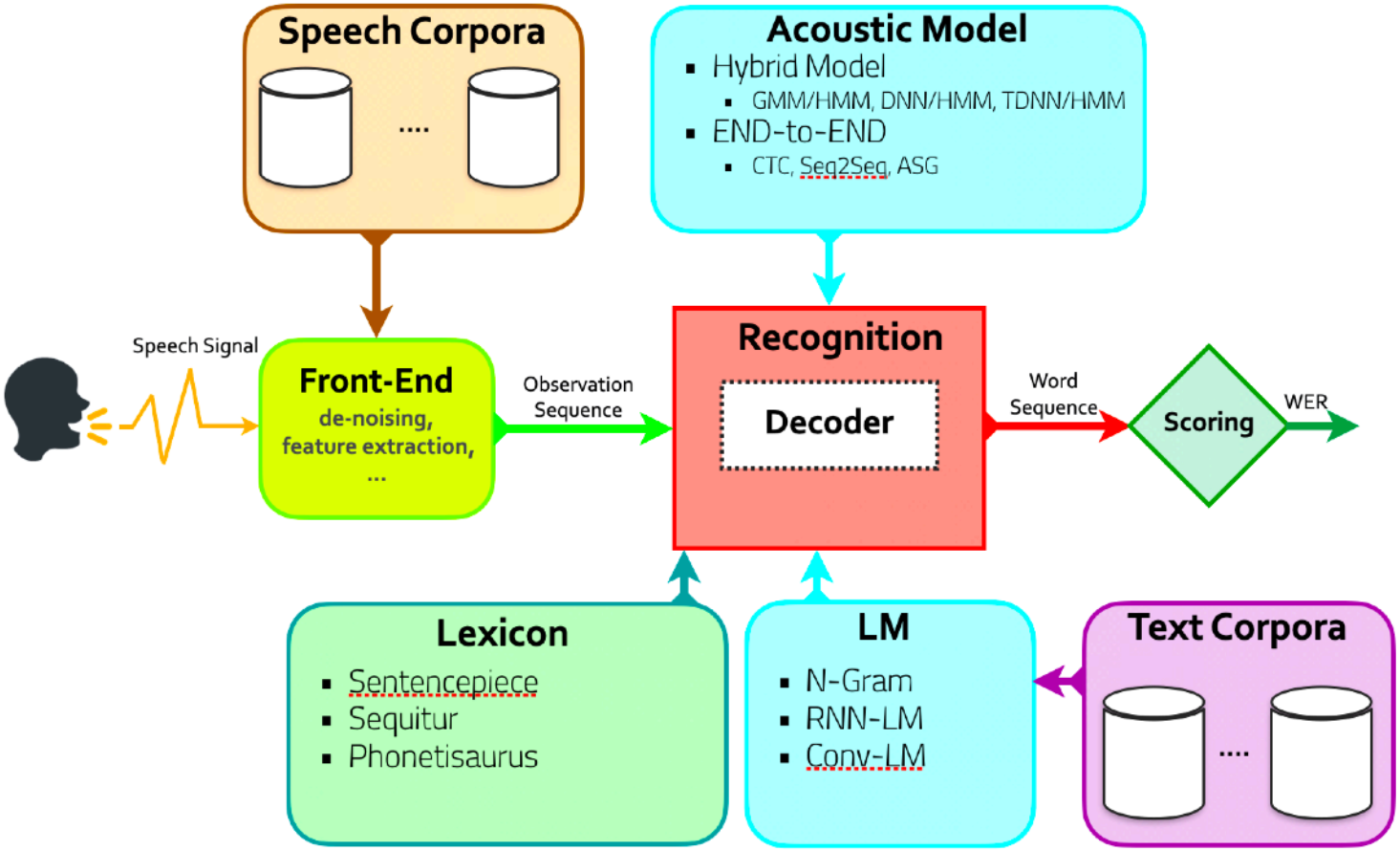
Ecosystem


Hes·so
NCCR
«Human Trust in AI»
idiap RESEARCH INSTITUTE
UNIVERSITÉ DE GENÈVE
DARPA
Collaborations with
Intl. enterprises & Swiss enterprise
Research
Joint Development Plan (JDP)
EPFL
Training
International Create Challenge
Innovation & TT
ideark
Innosuisse
spin-off
InnoPeaks
e-health
nsuretech
swisscard PAYMENTSOLUTIONS
suva
amm
Google
CREALOGIX
ARGUS
swisscom
MIGROS
HUAWEI
SAMSUNG
AIRBUS
aisa
Groupe Mutuel
AdaTis
KeyLemon
Klewel
KOEMEI
katia
genemo
recapp
BIOWATCH
eyeware
vima
UNIL | Université de Lausanne
UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG
UNIVERSITÄT BERN
3 professors EPFL
1 MERs EPFL
1 professor UniBe
1 professor UCSB
40 – 45 PhD students
EPFL academic activities
Distance & Industry-driven
AI Master Program
CANTON DU VALAIS
KANTON WALLIS
idiap
FernUni.ch
UniDistance.ch
New Tech Transfer Initiative
Idiap TTech Foundation
idiap RESEARCH INSTITUTE
AAISS



Automatic Speech Recognition (ASR)

Speech Recognition



Hybrid ASR

- HMM-based hybrid model remains a competitive ASR approach especially in low/medium-resource settings
- A two-stage training process is required, in which the older GMM approach is still used as a starting point.
- A typical ASR system is factorized into several modules including acoustic, lexicon, and language models.
- The acoustic model is not directly trained to minimize the final objective of interest.

Hybrid ASR

- **Linguistic information**
  - Requires a handcrafted pronunciation dictionary
  - Since phonemes are designed using linguistic knowledge, they are subject to human error that a fully data-driven system might avoid.
  - Problematic for multi-lingual ASR
    - One popular strategy is to make all languages share the same phonemic representations through a universal phonetic alphabet such as International Phonetic Alphabet (IPA) phone set.
  - Recently a hybrid ASR system using graphemic lexicon has been proposed.

Multilingual Graphemic Hybrid ASR with Massive Data Augmentation

Hybrid ASR

- **Complex decoding**
  - Inference/decoding has to be performed by integrating all modules.
- **Incoherence in optimization**
  - The above modules are optimized separately with different objectives, which may result in incoherence in optimization, where each module is not trained to match the other modules.
- **Stepwise refinement**
  - Many module-specific processes are required to build an accurate module.

End-to-End ASR

- **End-to-end ASR has the goal of simplifying the above module-based architecture into a single network architecture within a deep learning framework.**
- **All ASR models aim to model *p(W|X)*, the posterior distribution of a word sequence *W*, given a speech feature sequence *X*.**
- **End-to-end methods directly carry this out whereas conventional models factorize *p(W|X)* into modules such as**
  - the language model, p(W), which can be trained on pure language data,
  - and an acoustic model likelihood, p(X|W), which is trained on acoustic data with the corresponding language labels.

End-to-End ASR

- **There are two major types of end-to-end architectures for ASR:**
  - Connectionist temporal classification (CTC)
    - Uses Markov assumptions to efficiently solve sequential problems by dynamic programming

the ground truth of the word sequence — acoustic frames

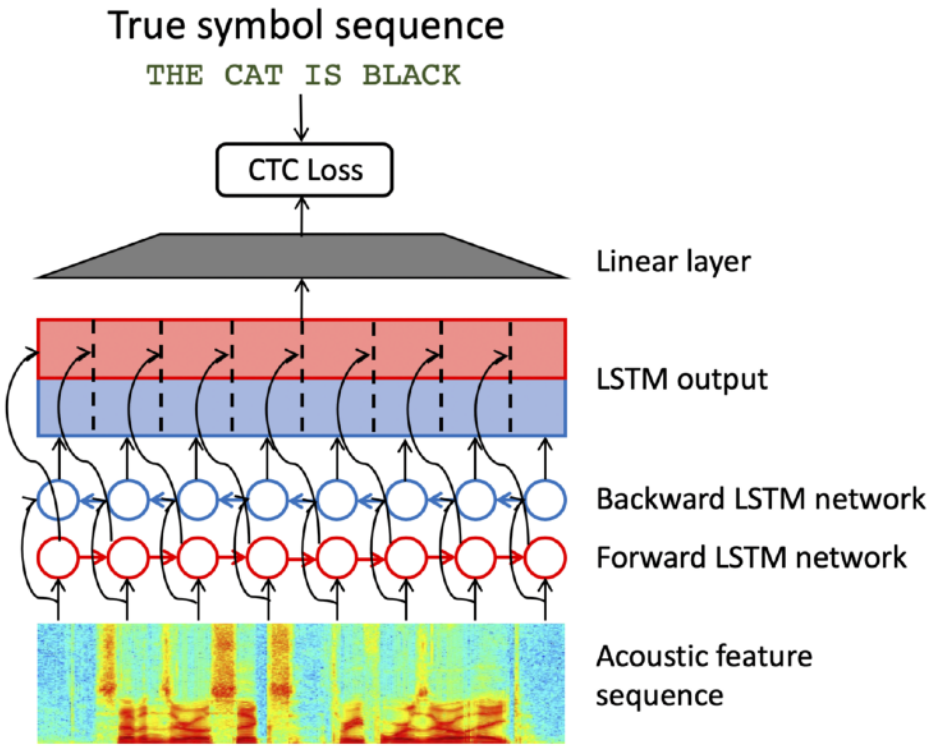
$$\mathcal{L}_{CTC} = -\log P(\boldsymbol{S}|\boldsymbol{X})$$

$$P(\boldsymbol{S}|\boldsymbol{X}) = \sum_{\boldsymbol{C} \in A(\boldsymbol{S})} P(\boldsymbol{C}|\boldsymbol{X})$$

sum over all possible paths (e.g. ccɛaaɛtt, cccɛaɛtt, cɛaɛtttt, ...)

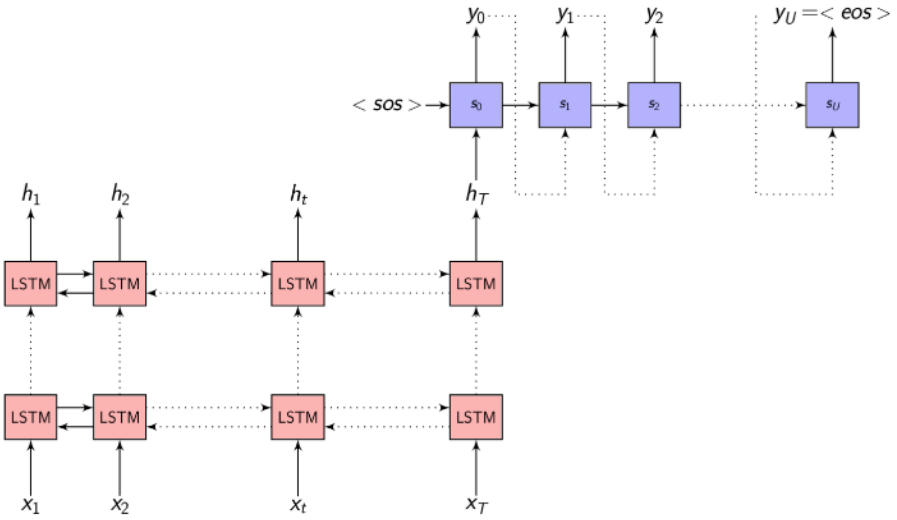
$$P(\boldsymbol{C}|\boldsymbol{X}) = \prod_{t=1}^{T} y(c_t, t)$$

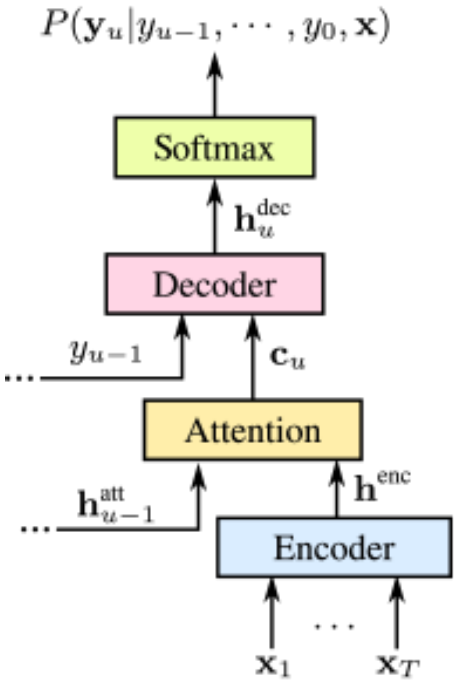
joint probability of a path (e.g. ccɛaaɛtt)



End-to-End ASR

- Attention-based methods
  - It solves the problem as a sequence mapping from speech feature sequences to text by using an encoder–decoder architecture.
  - The decoder network uses an attention mechanism to find an alignment between each element of the output sequence and the hidden states generated by the acoustic encoder network for each frame of acoustic input.
  - For an attention-based system, we can look into the whole input again for each step but it will be replaced with the attention.





ASR Challenges

- **Language diversity**
  - Multi-lingual ASR, Code-Switching
- **Dialects**
  - Multi-dialect ASR
- **Labeled data**
  - Unsupervised, self-supervised training, Low-resourced languages
- **Domain mismatch**
  - Domain adaptation
- **Out-of-Vocabulary (OOV)**
  - Subword-unit approach
- **…**

Subword-unit approach

- We can use either characters of words as the output of the model.
- A common approach for dealing with the open vocabulary issue is to break up rare words into subword units.
  - Byte-Pair-Encoding (BPE)
  - Unigram language model

Subword-unit approach

- **Byte-Pair-Encoding (BPE)**
  - BPE first splits the whole sentence into individual characters.
  - The most frequent adjacent pairs of characters are then consecutively merged until reaching a desired vocabulary size.
  - Words consisting of rare character combinations will be split into smaller units, e.g., substrings or characters.

persönlichkeitsrechte : _persön lichkeit s rechte

Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units." Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2016.

Subword-unit approach

- **Unigram Language Model**
  - The unigram language model makes an assumption that each subword occurs independently, and consequently, the probability of a subword sequence is formulated as the product of the subword occurrence probabilities

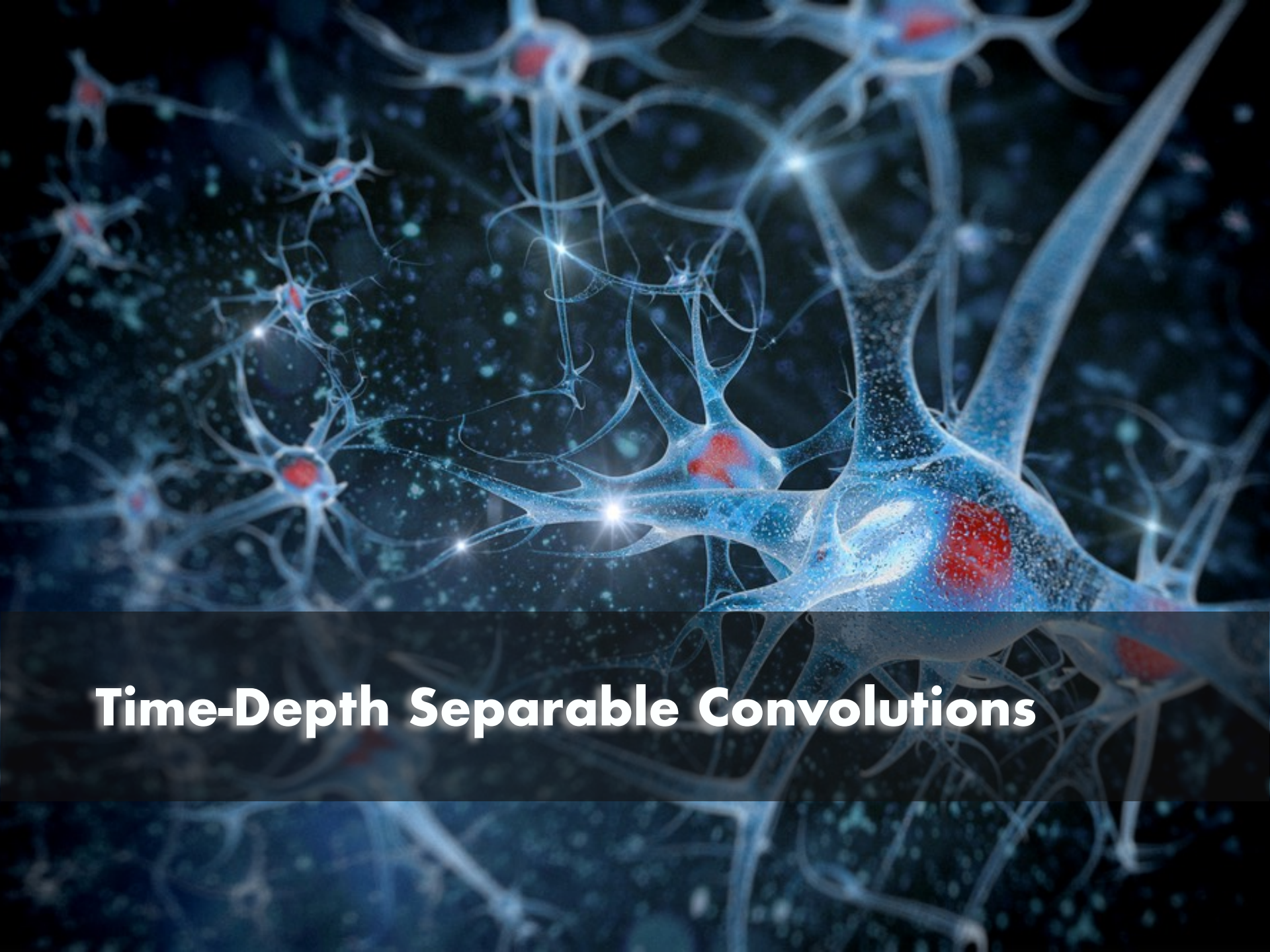
$$P(\mathbf{x}) = \prod_{i=1}^{M} p(x_i), \qquad \mathbf{x} = (x_1, \ldots, x_M)$$

$$\forall i \; x_i \in \mathcal{V}, \quad \sum_{x \in \mathcal{V}} p(x) = 1, \qquad \mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{S}(X)} P(\mathbf{x}),$$

spersönlichkeitsrechte : _persönlich keit s recht e

spersönlichkeitsrechte : _persönlich keit s re cht e

Kudo, Taku. "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates." Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018.


Time-Depth Separable Convolutions

Facebook AI Research

INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Sequence-to-Sequence Speech Recognition with Time-Depth Separable Convolutions

Awni Hannun, Ann Lee, Qiantong Xu, Ronan Collobert

Facebook AI Research, USA
awni@fb.com, annl@fb.com, qiantong@fb.com, locronan@fb.com

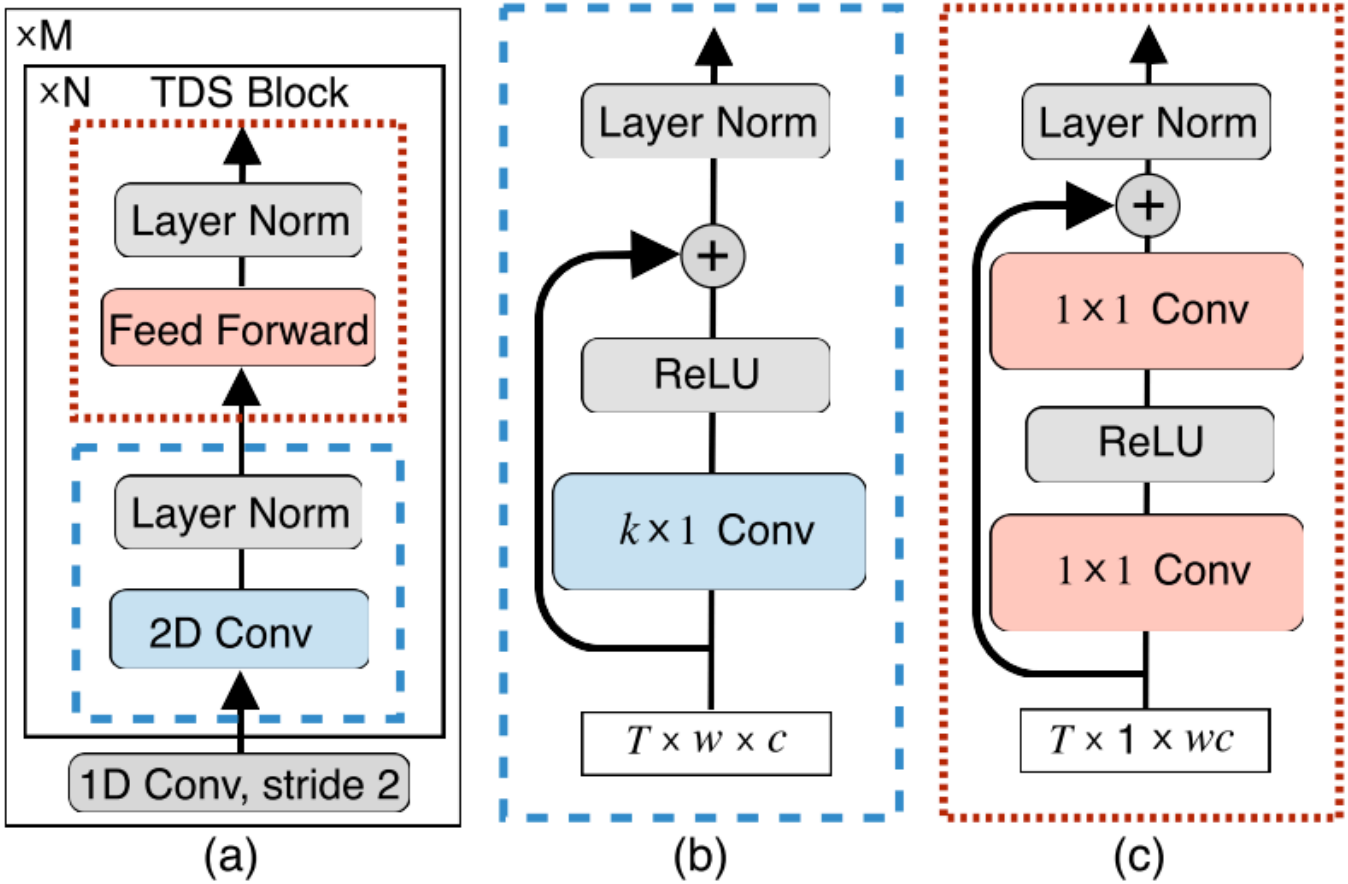
https://www.isca-speech.org/archive/Interspeech_2019/abstracts/2460.html

Architecture

- **A fully convolutional sequence-to-sequence encoder architecture with a simple and efficient decoder.**
- **Key to this approach is the fully convolutional encoder with a time-depth separable (TDS) block structure.**
  - TDS convolution block generalizes much better than other deep convolutional architectures and needs fewer parameters.
  - It can be implemented efficiently using a standard 2D convolution.

This model gives state-of-the-art results for non speaker adapted models on both LibriSpeech test sets at the time of publication.

Architecture



Architecture

```
(0): View (-1 80 1 0)
(1): Conv2D (1->10, 21x1, 2,1, SAME,SAME, 1, 1) (with bias)out (0.200000)
(4): LayerNorm ( axes : { 3 } )
(5): Time-Depth Separable Block (21, 80, 10) [800 -> 800 -> 800]
(6): Time-Depth Separable Block (21, 80, 10) [800 -> 800 -> 800]
(7): Conv2D (10->14, 21x1, 2,1, SAME,SAME, 1, 1) (with bias)
(8): ReLU
(9): Dropout (0.200000)
(10): LayerNorm ( axes : { 3 } )
(11): Time-Depth Separable Block (21, 80, 14) [1120 -> 1120 -> 1120]
(12): Time-Depth Separable Block (21, 80, 14) [1120 -> 1120 -> 1120]
(13): Time-Depth Separable Block (21, 80, 14) [1120 -> 1120 -> 1120]
(14): Conv2D (14->18, 21x1, 2,1, SAME,SAME, 1, 1) (with bias)
(15): ReLU
(16): Dropout (0.200000)
(17): LayerNorm ( axes : { 3 } )
(18): Time-Depth Separable Block (21, 80, 18) [1440 -> 1440 -> 1440]
(19): Time-Depth Separable Block (21, 80, 18) [1440 -> 1440 -> 1440]
(20): Time-Depth Separable Block (21, 80, 18) [1440 -> 1440 -> 1440]
(21): Time-Depth Separable Block (21, 80, 18) [1440 -> 1440 -> 1440]
(22): Time-Depth Separable Block (21, 80, 18) [1440 -> 1440 -> 1440]
(23): Time-Depth Separable Block (21, 80, 18) [1440 -> 1440 -> 1440]
(24): View (0 1440 1 0)
(25): Reorder (1,0,3,2)
(26): Linear (1440->1024) (with bias)
```

Model

- **Efficient Decoder**
  - The decoder is sequential in nature since to compute the next output requires the previous prediction
  - At training time a teacher forcing technique is used– the previous ground truth is used in place of the previous prediction. This allows to compute all output frames simultaneously.
  - An inner-product key-value attention is used which can be implemented much more efficiently than a neural attention.
  - An open-vocabulary beam search decoder is used which optimizes the following objective

$$\log P_{\text{s2s}}(Y \mid X) + \alpha \log P_{\text{LM}}(Y) + \beta |Y|. \qquad (7)$$

The term $|Y|$ counts the number of tokens in Y, α is the LM weight, and β is the token insertion term.

Model

- **Soft Window Pre-training**
  - Propose a simple soft attention window pre-training scheme to enable the training of very deep convolutional encoders
  - Encourage the model to align the output at uniform intervals along the input by penalizing attention values which are too far from the desired locations.
  - Window pre-training is used for the first few epochs and then switch it off. This is sufficient to enable the model to learn an alignment and converge.

Model

- **Regularization**
  - Dropout
  - Label Smoothing: Label smoothing is used to reduce over-confidence in predictions. As in machine translation, label smoothing hurts the loss on the dev set but improves WER.
  - Word-Piece Sampling: Word pieces as outputs following the Unigram Language Model approach.

Results

Table 1: *A comparison of the TDS conv model to other models on the Librispeech Dev and Test sets.*

Model	Dev WER		Test WER	
	clean	other	clean	other
hybrid, speaker adapted				
CAPIO (single) [33] + RNN	3.12	8.28	3.51	8.58
CAPIO (ensemble) [33] + RNN	2.68	7.56	3.19	7.64
CNN ASG [31] + ConvLM	3.16	10.05	3.44	11.24
RNN S2S [23]	4.87	14.37	4.87	15.39
RNN S2S [23] + 4-gram	4.79	14.31	4.82	15.30
RNN S2S [23] + LSTM	3.54	11.52	3.82	12.76
TDS conv	5.04	14.45	5.36	15.64
TDS conv + 4-gram	3.75	10.70	4.21	11.87
TDS conv + ConvLM	3.01	8.86	3.28	9.84


Wav2Letter

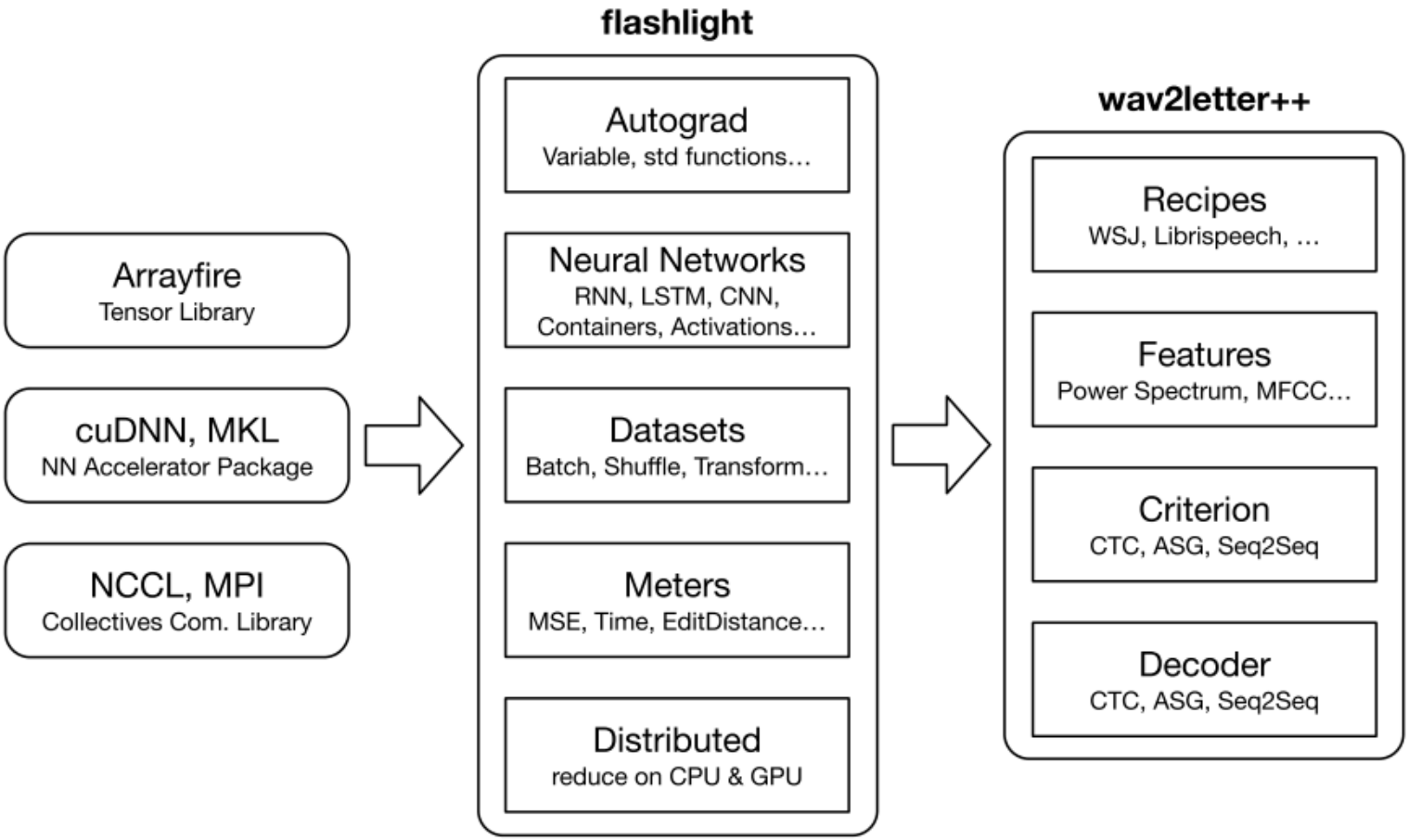
Wav2letter

- Wav2letter++ is the first open-source speech recognition system written entirely in C++
- Uses the ArrayFire tensor library for maximum efficiency
- It relies on end-to-end acoustic modeling based on graphemes rather than phonemes.
- End-to-end models are significantly simpler and the gap in accuracy to hybrid systems is rapidly closing.

Design

- **The design of wav2letter++ is motivated by three requirements:**
  - The toolkit must be able to efficiently train models on datasets containing many thousands of hours of speech.
  - Expressing and incorporating new network architectures, loss functions, and other core operations should be simple.
  - The path from model research to deployment should be straightforward, requiring as little new code as possible while maintaining the flexibility needed for research.

Design


flashlight
Arrayfire
Tensor Library
cuDNN, MKL
NN Accelerator Package
NCCL, MPI
Collectives Com. Library
Autograd
Variable, std functions…
Neural Networks
RNN, LSTM, CNN,
Containers, Activations…
Datasets
Batch, Shuffle, Transform…
Meters
MSE, Time, EditDistance…
Distributed
reduce on CPU & GPU
wav2letter++
Recipes
WSJ, Librispeech, …
Features
Power Spectrum, MFCC…
Criterion
CTC, ASG, Seq2Seq
Decoder
CTC, ASG, Seq2Seq

Design

- **ArrayFire Tensor Library**
  - ArrayFire is a highly optimized tensor library that can execute on multiple back-ends including a CUDA GPU back-end and a CPU back-end.
  - ArrayFire also uses just-in-time code generation to combine series of simple operations into a single kernel call. This results in faster execution for memory bandwidth bound operations and can reduce peak memory use.
- **Flashlight Machine Learning Library**
  - The wav2letter++ library is built on top of flashlight
  - For certain operations we extend the core ArrayFire CUDA back-end with more efficient cuDNN operations such as convolutions and RNN routines, among others.

Design

- **Data Preparation and Feature Extraction**
  - Feature extraction supports multiple audio file formats (e.g. wav, flac... / mono, stereo / int, float) and several feature types including the raw audio, a linearly scaled power spectrum, log-Mels (MFSC) and MFCCs. We use the FFTW library to compute discrete Fourier transforms
  - Data loading in wav2letter++ computes features on the fly prior to each network evaluation. This allows for dynamic data augmentation and makes deploying models easier since the full pipeline can run from a single binary
  - We load the audio and compute the features asynchronously and in parallel. For the models and batch sizes we tested, the time spent in data loading is negligible

Design

- **Models**
  - It support several end-to-end sequence models. Every model is divided into a network and criterion.
    - Connectionist Temporal Classification (CTC)
    - AutoSegCriterion (ASG)
    - Sequence-to-Sequence models with attention (S2S)
    - Transformers
  - Adding new sequence criteria is particularly easy given that loss functions like ASG and CTC can be efficiently implemented in C++.

Design

- **Training and Scale**
  - The training pipeline gives maximum flexibility for the user to experiment with different features, architectures and optimization parameters.
  - Training can be run in three modes
    - train (flat-start training)
    - continue (continuing with a checkpoint state), and
    - fork (for e.g. transfer learning).
  - wav2letter++ scale to larger datasets with data parallel, synchronous SGD.
  - NVIDIA Collective Communication Library (NCCL2) is used for inter-process communication

Design

- **Optimization functions:**

```
constexpr const char* kSGDOptimizer = "sgd";
constexpr const char* kAdamOptimizer = "adam";
constexpr const char* kRMSPropOptimizer = "rmsprop";
constexpr const char* kAdadeltaOptimizer = "adadelta";
constexpr const char* kAdagradOptimizer = "adagrad";
constexpr const char* kAMSgradOptimizer = "amsgrad";
constexpr const char* kNovogradOptimizer = "novograd";
constexpr const char* kCtcCriterion = "ctc";
constexpr const char* kAsgCriterion = "asg";
constexpr const char* kSeq2SeqCriterion = "seq2seq";
constexpr const char* kTransformerCriterion = "transformer";
```

Decoder

- **Beam-Search Decoding**
  - The decoder interface accepts as input the emissions and (if relevant) transitions from the acoustic model.
  - We also give the decoder a Trie which contains the word dictionary and a language model.
  - We support any type of language model which exposes the interface required by our decoder including n-gram LMs and any other stateless parametric LM

Writing ARC file

```
V -1 NFEAT 1 0
C2 1 10 21 1 2 1 -1 -1
R
DO 0.05
LN 0 1 2
TDS 10 21 80 0.05
C2 10 14 21 1 2 1 -1 -1
R
DO 0.1
LN 0 1 2
TDS 14 21 80 0.1
TDS 14 21 80 0.1
TDS 14 21 80 0.15
C2 14 18 21 1 2 1 -1 -1
R
DO 0.2
LN 0 1 2
TDS 18 21 80 0.2
TDS 18 21 80 0.2
TDS 18 21 80 0.25
TDS 18 21 80 0.25
V 0 1440 1 0
RO 1 0 3 2
L 1440 1024
```

Writing ARC file

Conv2D

```
C2 [inputChannels] [outputChannels] [xFilterSz] [yFilterSz] [xStride]
[yStride] [xPadding <OPTIONAL>] [yPadding <OPTIONAL>] [xDilation <OPTIONAL>]
[yDilation <OPTIONAL>]
```

Linear

```
L [inputChannels] [outputChannels]
```

BatchNorm

```
BN [totalFeatSize] [firstDim] [secondDim <OPTIONAL>] [thirdDim <OPTIONAL>]
```

Dropout

```
DO [dropProb]
```

TDSBlock

```
TDS [inputChannels] [kernelWidth] [inputWidth] [dropoutProb <OPTIONAL,
DEFAULT=0>] [innerLinearDim <OPTIONAL, DEFAULT=0>] [rightPadding <OPTIONAL,
DEFAULT=-1>] [lNormIncludeTime <OPTIONAL, DEFAULT=True>]
```

RNN

```
LSTM [inputSize] [outputSize] [numLayers] [isBidirectional] [dropProb]
GRU [inputSize] [outputSize] [numLayers] [isBidirectional] [dropProb]
RNN [inputSize] [outputSize] [numLayers] [isBidirectional] [dropProb]
```

Data Preparation

```
// Example input file format

[~/speech/data] head train.lst
train001 /tmp/000000000.flac 100.03  this is sparta
train002 /tmp/000000001.flac 360.57  coca cola
train003 /tmp/000000002.flac 123.53  hello world
train004 /tmp/000000003.flac 999.99  quick brown fox jumped
...
...
```

- We use sndfile for loading the audio files.
- A token dictionary file consists of a list of all subword units (graphemes / phonemes /...) used.
- A lexicon file consists of mapping from words to their sequence of tokens representation.
- You can use KenLM for training an N-gram language model.

Train a model

```
<train_cpp_binary> [train|continue|fork] \
--datadir <path/to/data/> \
--tokensdir <path/to/tokens/file/> \
--archdir <path/to/architecture/files/> \
--rundir <path/to/save/models/> \
--arch   <name_of_architecture.arch> \
--train  <train/datasets/ds1.lst,train/datasets/ds2.lst> \
--valid  <validation/datasets/ds1.lst,validation/datasets/ds2.lst> \
--lexicon <path/to/lexicon.txt> \
--lr=0.0001 \
--lrcrit=0.0001
```

Decoding

- To get the greedy path, one should use the Test binary in the following way

```
wav2letter/build/Test \
  --am path/to/train/am.bin \
  --maxload 10 \
  --test path/to/test/list/file \
  --tokensdir path/to/tokens/dir \
  --tokens tokens.txt \
  --lexicon path/to/the/lexicon/file
```

- The **Test binary** can be used also to generate an **Emission Set** including the emission matrix as well as other target-related information for each sample. All flags are also stored in the **Emission Set.** Specifically, the emission matrix of the CTC/ASG model is the posterior, while for seq2seq models, it is an encoded audio with a series of embeddings. The **Emission Set** can be fed into the **Decode binary** directly to generate transcripts without running AM forwarding again.

Decoding

- **It support lexicon-base beam-search decoder and lexicon-free beam-search decoder for acoustic models.**
- **Lexicon-based beam-search decoder (uselexicon=true)**
  - For a lexicon-based decoder it restrict our search by the lexicon provided by a user. In other words, generated transcriptions only contain words from the lexicon.
- **Lexicon-free beam-search decoder (uselexicon=false)**
  - The lexicon-free beam-search decoder considers any possible token as candidates and there is no notion of words during decoding.
  - In this case, a word separator should be set by `--wordseparator` and included into tokens set for AM training.
  - The word separator is treated and predicted as all the other normal tokens.
  - After obtaining the transcription in tokens, word separator is used to split the sequence into words.

Decoding

```
wav2letter/build/Decode \
  --am path/to/train/am.bin \
  --test path/to/test/list/file \
  --maxload 10 \
  --nthread_decoder 2 \
  --show \
  --sholetters \
  --lexicon path/to/the/lexicon/file \
  --uselexicon [true, false] \
  --lm path/to/lm/file \
  --lmtype [kenlm, convlm] \
  --decodertype [wrd, tkn] \
  --beamsize 100 \
  --beamsizetoken 100 \
  --beamthreashould 20 \
  --lmweight 1 \
  --wordscore 0 \
  --eosscore 0 \
  --silscore 0 \
  --unkscore 0 \
  --smearing max
```

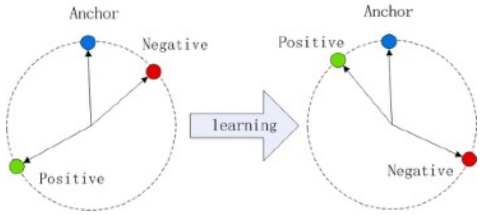

Wav2Vec 2.0

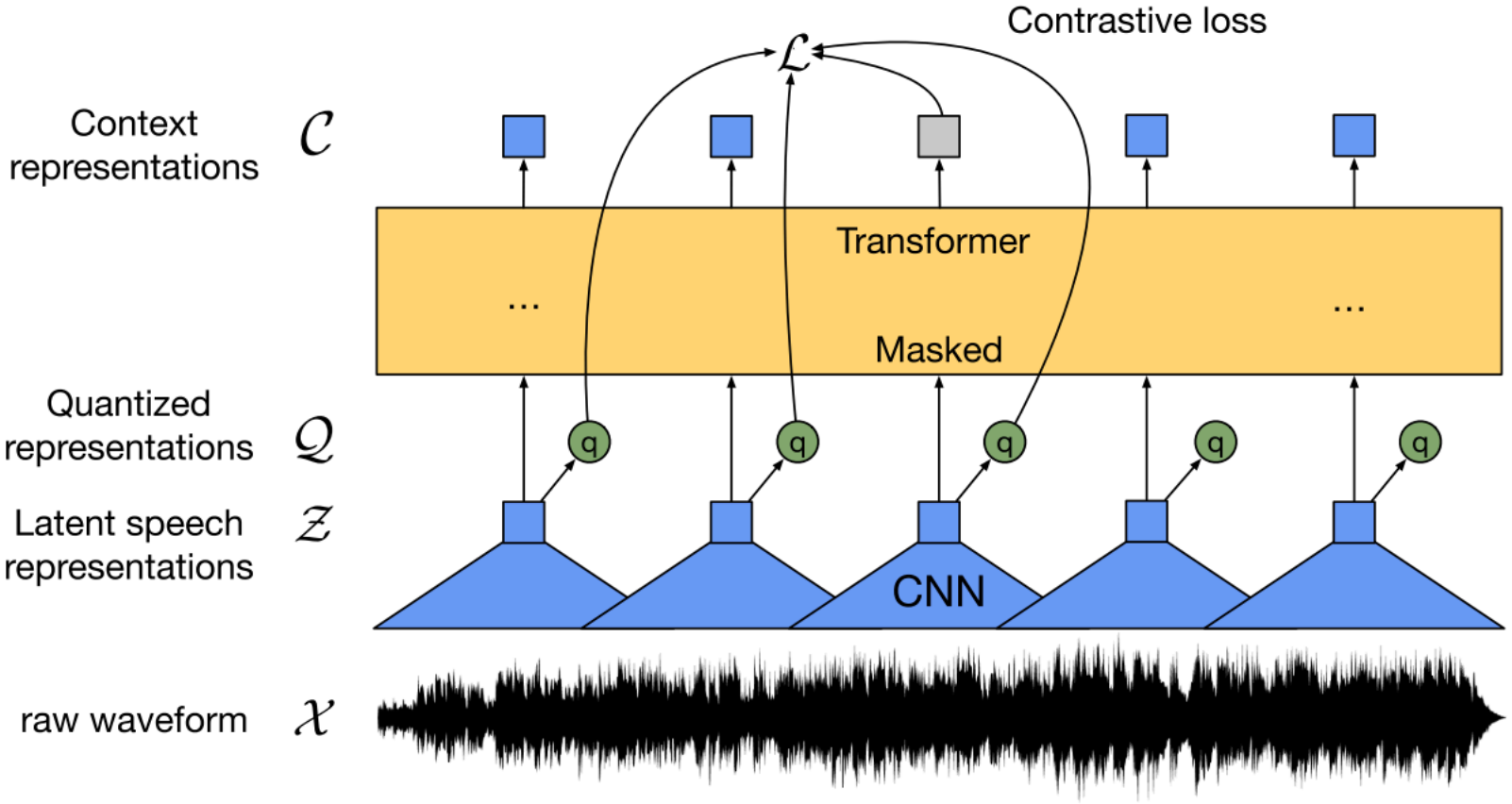
Wav2Vec 2.0

- **A framework for Self-Supervised Learning of Speech Representation**
  - Learning powerful representations for speech audio using self-supervision
  - It masks the speech input in the latent space and solves a contrastive task defined over a quantization of latent representations which are jointly learned.
  - Learn discrete linguistic units via a gumbel softmax to represent the latent representations in the contrastive task.
  - Using just 10min of labeled data, and pre-training on 53h hours of unlabeled data still achieves 5.7/10.1 WER on the clean/noisy test sets of Librispeech.
- **Facebook AI team**
  - https://arxiv.org/abs/2006.11477

https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/README.md

Model Architecture





Model

- **Feature Encoder**
  - The encoder consists of several blocks containing a temporal convolution followed by a GELU (Gaussian Error Linear Units) activation function.
  - The first block maps raw audio to a feature representation
  - Add a group normalization before the GELU to normalize each output channel over the sequence.
  - Apply layer normalization to the output channel of this network.
- **Quantization Module**
  - Discretize the output of the feature encoder z to a finite set of speech representations via product quantization.
  - This amounts to choosing quantized representations from multiple codebooks and concatenating them.
- **Contextual Representation with Transformers**
  - The output of feature encoder is fed to a context network which follows the Transformer architecture.
  - Relative positional embedding (using convolution layer)

Training

- **Masking**
  - Masking a proportion (~50%) of the feature encoder output or time-steps before feeding them to the context network and replace them with a trained feature vector shared between all masked time steps.
- **Objective**

$$\mathcal{L} = \mathcal{L}_m + \alpha \mathcal{L}_d + \beta \mathcal{L}_f$$

- **Contrastive Loss**

$$\mathcal{L}_m = -\log \frac{\exp(sim(\mathbf{c}_t, \mathbf{q}_t))/\kappa}{\sum_{\tilde{\mathbf{q}} \sim \mathbf{Q}_t} \exp(sim(\mathbf{c}_t, \tilde{\mathbf{q}}))/\kappa}$$

Training

- **Diversity Loss and L2 penalty**
  - The contrastive task depends on the codebook to represent both positive and negative examples and the diversity loss is designed to increase the use of the quantized codebook representations.
  - Encourage the equal use of the V entries in each of the G codebooks by maximizing the entropy of the averaged softmax distribution.
- **Fine-Tuning**
  - Pre-trained models are fine-tuned for speech recognition by adding a randomly initialized linear projection on top of the context network into C classes representing the vocabulary of the task.
  - For Librispeech 29 tokens for character targets plus a word boundary token is used.
  - Model is optimized using CTC loss.
  - SpecAug is used for masking time and frequency during training.

Models & Datasets

- **Datasets**
  - Unlabeled data
    - Librispeech (~960h), LibriVox (~53,200h)
  - Labeled data for fine-tuning
    - Librispeech: All (~960h), train-clean-100 (~100h), train-10h, train-1h, train-10min
- **Base Model**
  - 12 transformer blocks, model dimension 768, inner dimension (FFN) 3,072 and 8 attention heads.
  - 95m parameters
- **Large Model**
  - 24 transformer blocks with model dimension 1,024, inner dimension 4,096 and 16 attention heads.
  - 317m parameters

Results

Model	Unlabeled data	LM	dev		test	
			clean	other	clean	other
10 min labeled						
Discrete BERT [4]	LS-960	4-gram	15.7	24.1	16.3	25.2
BASE	LS-960	4-gram	8.9	15.7	9.1	15.6
		Transf.	6.6	13.2	6.9	12.9
LARGE	LS-960	Transf.	6.6	10.6	6.8	10.8
	LV-60k	Transf.	5.4	9.9	5.7	10.1
1h labeled						
Discrete BERT [4]	LS-960	4-gram	8.5	16.4	9.0	17.6
BASE	LS-960	4-gram	5.0	10.8	5.5	11.3
		Transf.	3.8	9.0	4.0	9.3
LARGE	LS-960	Transf.	3.8	7.1	3.9	7.6
	LV-60k	Transf.	3.3	6.4	3.4	6.8
10h labeled						
Discrete BERT [4]	LS-960	4-gram	5.3	13.2	5.9	14.1
BASE	LS-960	4-gram	3.8	9.1	4.3	9.5
		Transf.	2.9	7.4	3.2	7.8
LARGE	LS-960	Transf.	2.9	5.7	3.2	6.1
	LV-60k	Transf.	2.5	5.2	2.6	5.2

Results

100h labeled						
Hybrid DNN/HMM [33]	-	4-gram	5.0	19.5	5.8	18.6
TTS data augm. [29]	-	LSTM			4.3	13.5
Discrete BERT [4]	LS-960	4-gram	4.0	10.9	4.5	12.1
Iter. pseudo-labeling [56]	LS-860	4-gram+Transf.	5.0	8.72	5.37	9.51
Iter. pseudo-labeling [56]	+LV-60k	4-gram+Transf.	3.19	6.14	3.72	7.11
Noisy student [41]	LS-860	LSTM	3.9	8.8	4.2	8.6
BASE	LS-960	4-gram	2.7	7.9	3.4	8.0
		Transf.	2.2	6.3	2.6	6.3
LARGE	LS-960	Transf.	2.1	4.8	2.3	5.0
	LV-60k	Transf.	2.0	4.1	2.1	4.4

Results

Table 2: WER on Librispeech when using all labeled data of 960 hours (cf. Table 1).

Model	Unlabeled data	LM	dev		test	
			clean	other	clean	other
Supervised						
CTC Transf [49]	-	CLM+Transf.	2.20	4.94	2.47	5.45
S2S Transf. [49]	-	CLM+Transf.	2.10	4.79	2.33	5.17
Transf. Transducer [58]	-	Transf.	-	-	2.0	4.6
ContextNet [16]	-	LSTM	1.9	3.9	1.9	4.1
Semi-supervised						
CTC Transf. + PL [49]	LV-60k	CLM+Transf.	2.10	4.79	2.33	4.54
S2S Transf. + PL [49]	LV-60k	CLM+Transf.	2.00	3.65	2.09	4.11
Iter. pseudo-labeling [56]	LV-60k	4-gram+Transf.	1.85	3.26	2.10	4.01
Noisy student [41]	LV-60k	LSTM	1.6	3.4	1.7	3.4
This work						
LARGE - from scratch	-	Transf.	1.7	4.3	2.1	4.6
BASE	LS-960	Transf.	1.8	4.7	2.1	4.8
LARGE	LS-960	Transf.	1.7	3.9	2.0	4.1
	LV-60k	Transf.	1.6	3.2	1.9	3.5

Thank You