ESPnet: End-to-End Speech Processing Toolkit

Yuan-Fu Liao

National Yang Ming Chiao Tung University

yfliao@nycu.edu.tw

ESPnet Tutorial Series

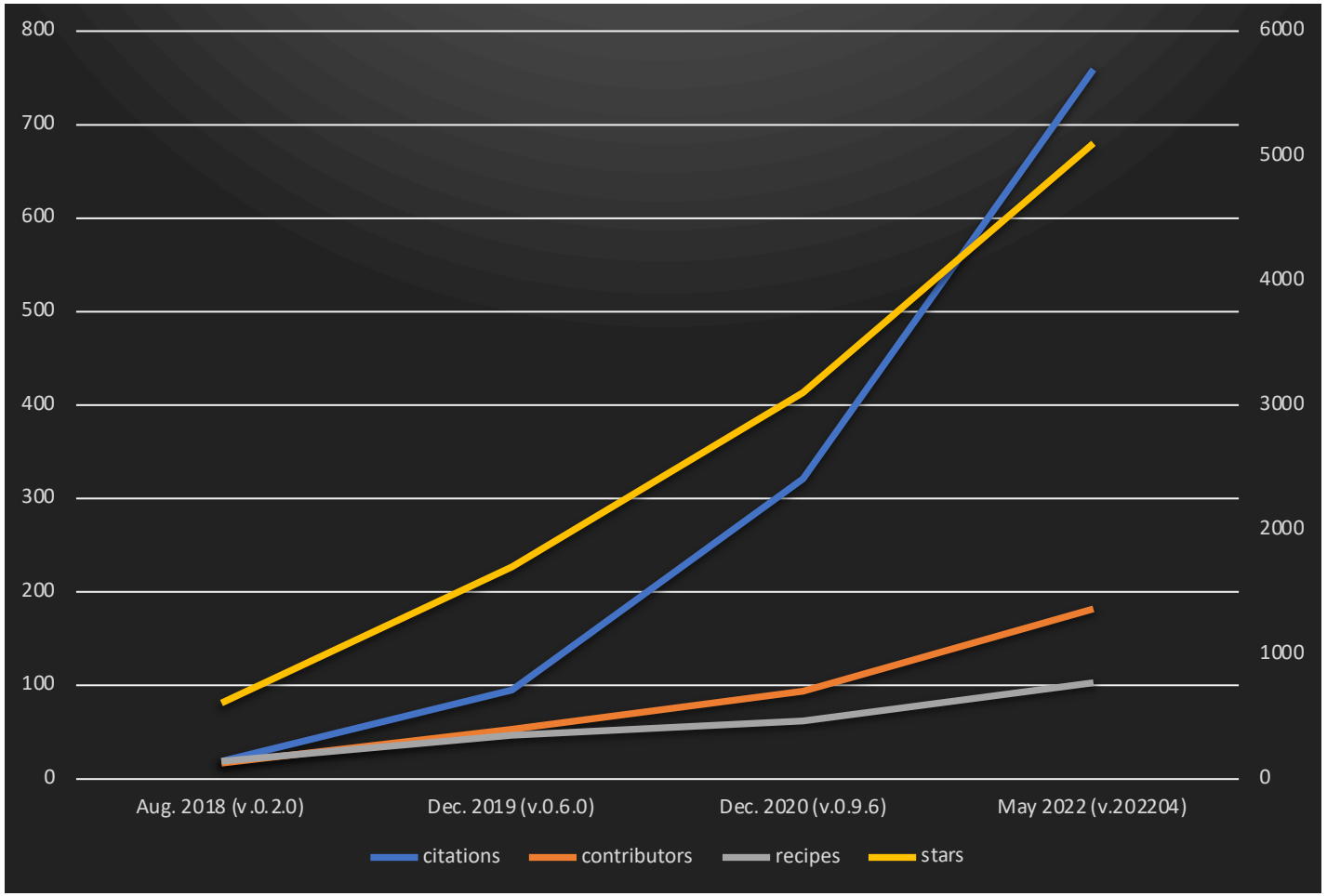
- 2019 Tutorial at Interspeech
  - Material
- 2021 Tutorial at CMU
  - Online video
  - Material
- 2022 Tutorial at CMU
  - Usage of ESPnet (ASR as an example)
    - Online video
    - Material
  - Add new models/tasks to ESPnet
    - Online video
    - Material

ESPnet  , launched in December 2017

https://github.com/espnet/espnet

- **Open source** (Apache2.0) end-to-end speech processing toolkit
- Major concept: **Reproducibility**
  - Leverage our end-to-end ASR experience to the community
  - Accelerates end-to-end research for speech researchers
- Initially Chainer but later PyTorch based dynamic neural network toolkit as an engine
  - Easily develop novel neural network architecture
- Follows the **famous speech recognition toolkit, Kaldi, style**
  - Smoothly port ASR experiments from Kaldi to ESPnet with the common data processing, feature extraction/format
  - Recipes to provide a complete setup for speech processing experiments
- **The project is greatly accelerated in these four years**

Activity statistics (from 2018 to 2022)



- **Citations**, **contributors**, **recipes** (examples), and **stars** are all growing

i.e.,

- Developers have increasingly supported the development of ESPnet
- has been used in various research groups and contributed a lot to speech research activities
- ESPnet 5.2K stars (today)
  - Kaldi 11.7K stars
  - Nvidia NeMo 4.4K stars
  - SpeechBrain 4.1K stars
  - Google Lingvo 2.5K stars

Major change in the internal framework
From ESPnet1 to ESPnet2

ESPnet2: a new system for DNN training to extend our system from v.0.7.0

Mostly refactoring, but which enables major update to deal with

- **Distributed training**
- **On-the-fly feature extraction** from the raw waveform
- Improved the **scalability**

Improved software workflow by enhancing the **continuous integration**, **enriching documentation**, supporting the **docker**, **pip install**, and **model zoo** functions via HuggingFace.

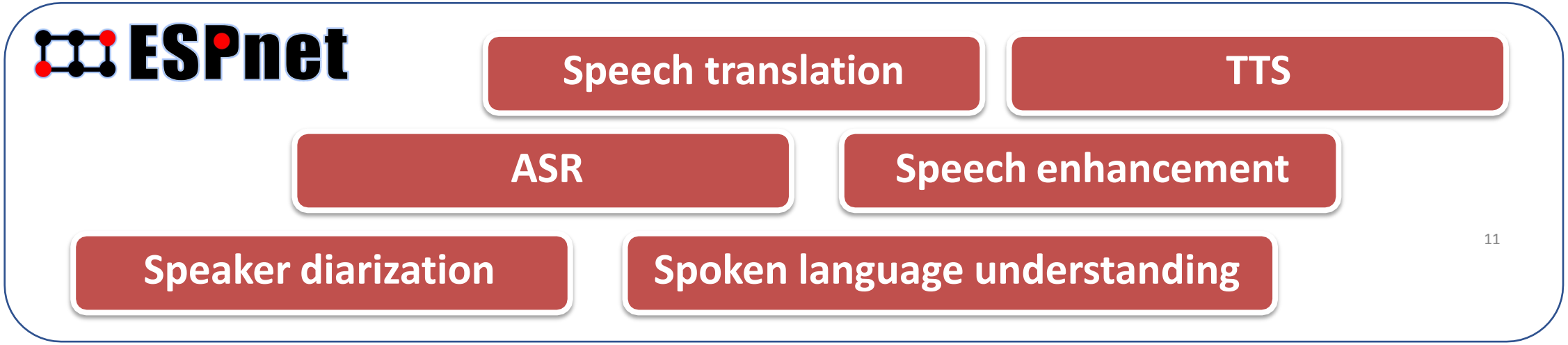
What kind of research topics in speech research?

- Speech recognition
- Speech synthesis
- Voice conversion
- Speaker recognition
- Language recognition
- Speech emotion recognition
- Speaker diarization
- Speech coding
- Speech perception
- Speech enhancement
- Microphone array processing
- Audio event classification and detection
- Speech separation
- Spoken language understanding
- Spoken dialogue systems
- Speech translation
- Multimodal processing
- Speech corpus

From the topic list in Interspeech

Broadened Applications

- ESPnet (**ASR+X**) covers the following topics complementally



- Why can one toolkit support such wide-ranges of applications?

What kind of research topics in speech research?

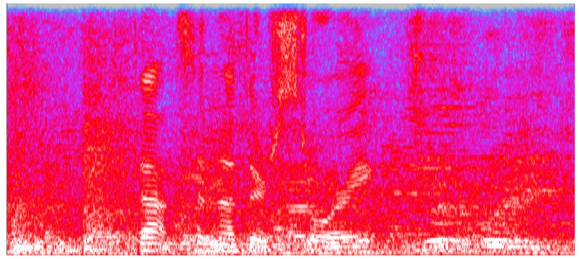
- **Speech recognition**
- **Speech synthesis**
- **Voice conversion**
- Speaker recognition
- Language recognition
- **Speech emotion recognition**
- **Speaker diarization**
- Speech coding
- Speech perception
- **Speech enhancement**
- **Microphone array processing**
- **Audio event classification and detection**
- **Speech separation**
- **Spoken language understanding**
- Spoken dialogue systems
- **Speech translation**
- **Multimodal processing**
- Speech corpus

Automatic speech recognition (ASR)

- Mapping **speech** sequence to ***character*** sequence

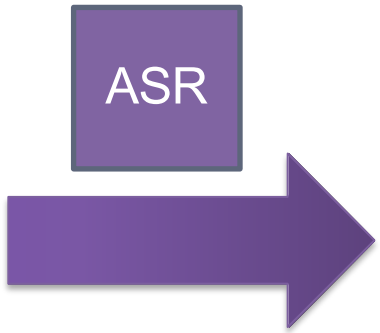
$$X = (x(l) \in \mathbb{Z} | l = 1, \ldots, L)$$

$$L = 43263$$



$$X = (\mathbf{x}_t \in \mathbb{R}^D | t = 1, \ldots, T)$$

$$T = 268$$



"That's another story"

$$W = (\mathbf{w}_n \in \mathcal{V} | n = 1, \ldots, N)$$

$$N = 18$$

Speech to text translation (ST)

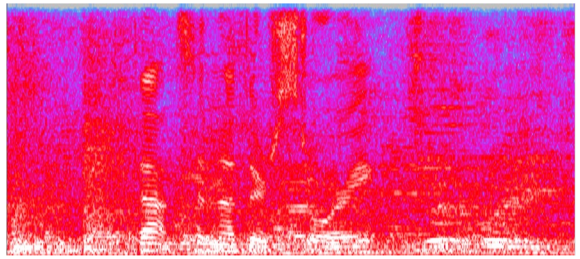
- Mapping **speech** sequence in a **source** language to ***character*** sequence in a **target** language

ST

That's another story

$$X = (x(l) \in \mathbb{Z} | l = 1, \ldots, L)$$

$$L = 43263$$



$$X = (\mathbf{x}_t \in \mathbb{R}^D | t = 1, \ldots, T)$$

$$T = 268$$

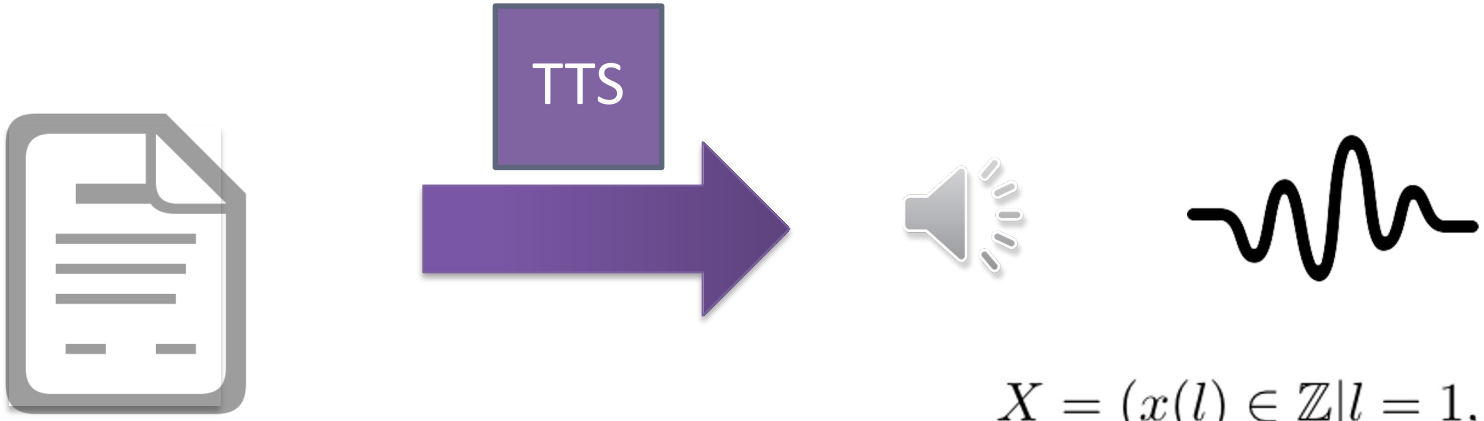
"Das ist eine andere Geschichte"

$$W = (\mathbf{w}_n \in \mathcal{V} | n = 1, \ldots, N)$$

N=31

Text to speech (TTS)

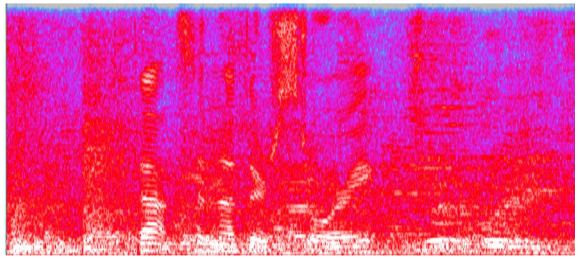
- Mapping **character** sequence to **speech** sequence



"That's another story"

$$W = (\mathbf{w}_n \in \mathcal{V} | n = 1, \ldots, N)$$
$$N = 18$$

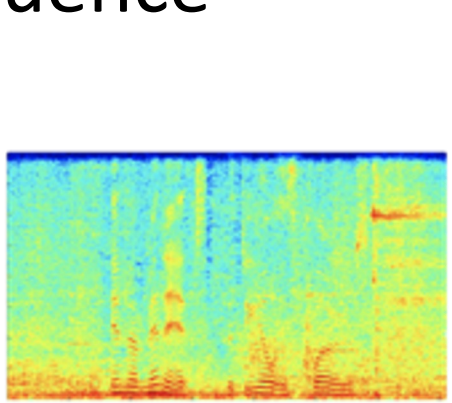
$$X = (x(l) \in \mathbb{Z} | l = 1, \ldots, L)$$
$$L = 43263$$



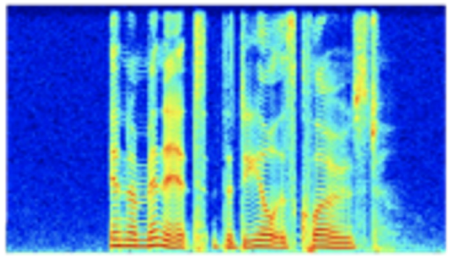
$$X = (\mathbf{x}_t \in \mathbb{R}^D | t = 1, \ldots, T)$$
$$T = 268$$

Speech enhancement (SE)

- Mapping **noisy** speech sequence to **clean** speech sequence



SE



$$X = (\mathbf{x}_t \in \mathbb{R}^D | t = 1, \ldots, T)$$
$$T = 268$$

$$X' = (\mathbf{x}'_t \in \mathbb{R}^D | t = 1, \ldots, T)$$
$$T = 268$$

All of the problems

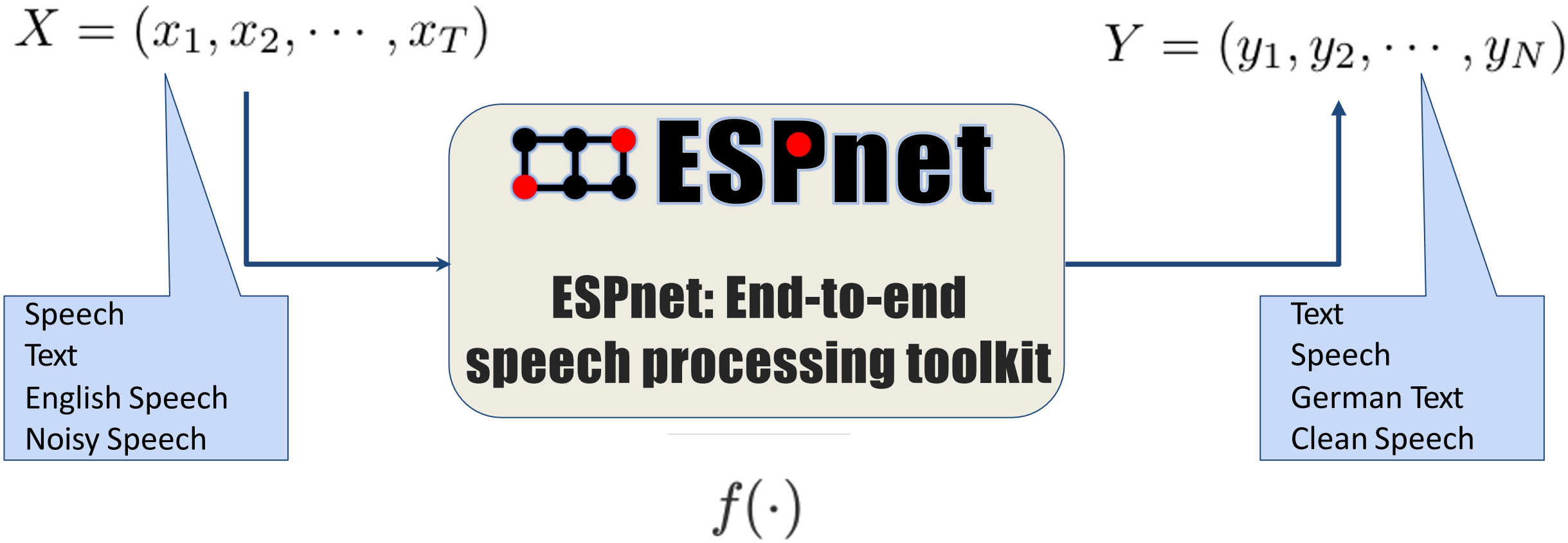
$$X = (x_1, x_2, \cdots, x_T) \xrightarrow{f} Y = (y_1, y_2, \cdots, y_N)$$

Unified form → Unified software design

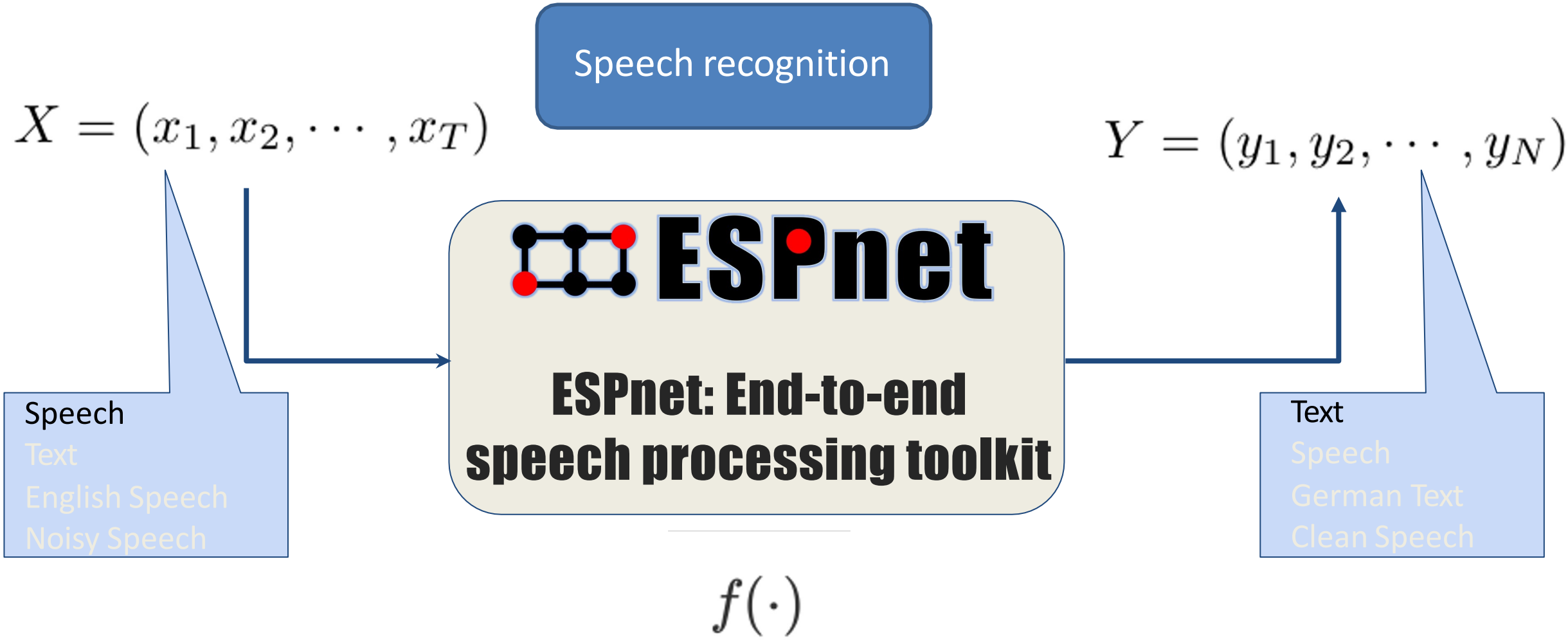
We design ESPnet by leveraging a **unified** mathematical **form** of **sequence (X) to sequence (Y) transformation f**

$$X = (x_1, x_2, \cdots, x_T) \xrightarrow{f} Y = (y_1, y_2, \cdots, y_N)$$

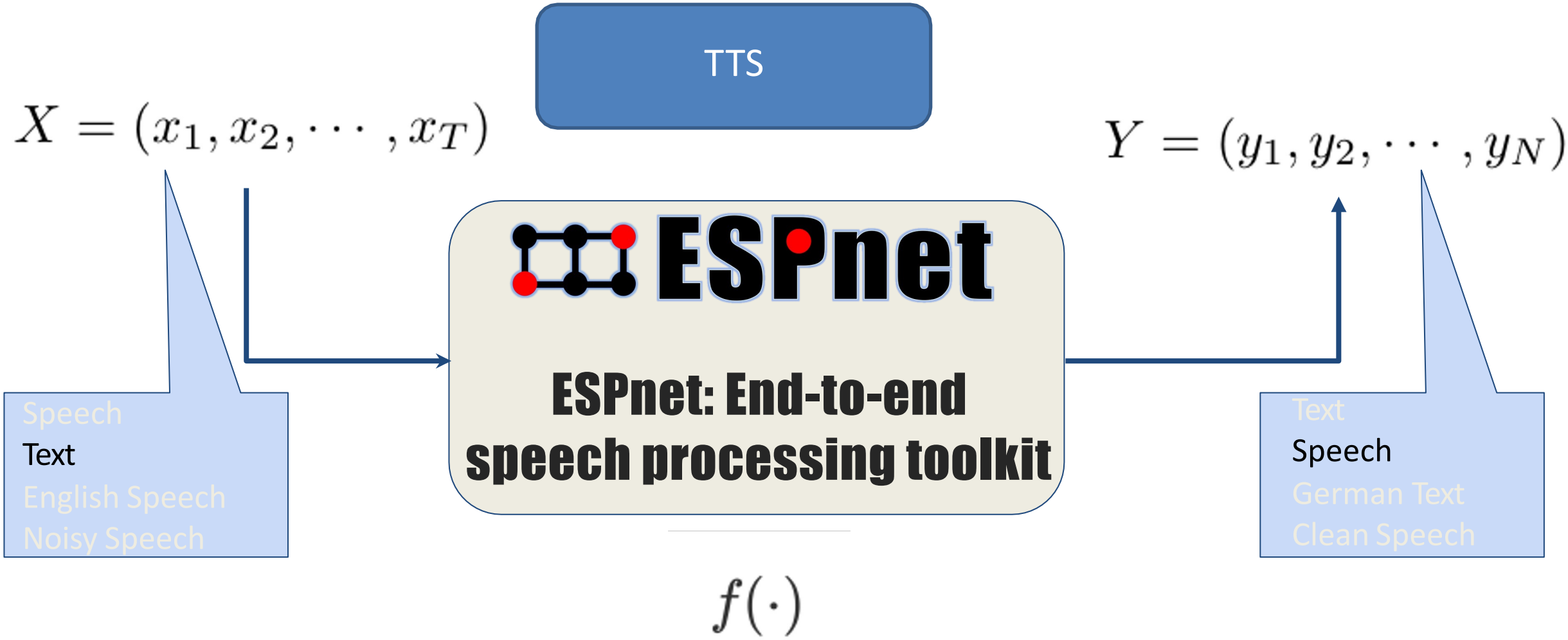
Unified view → Unified software design



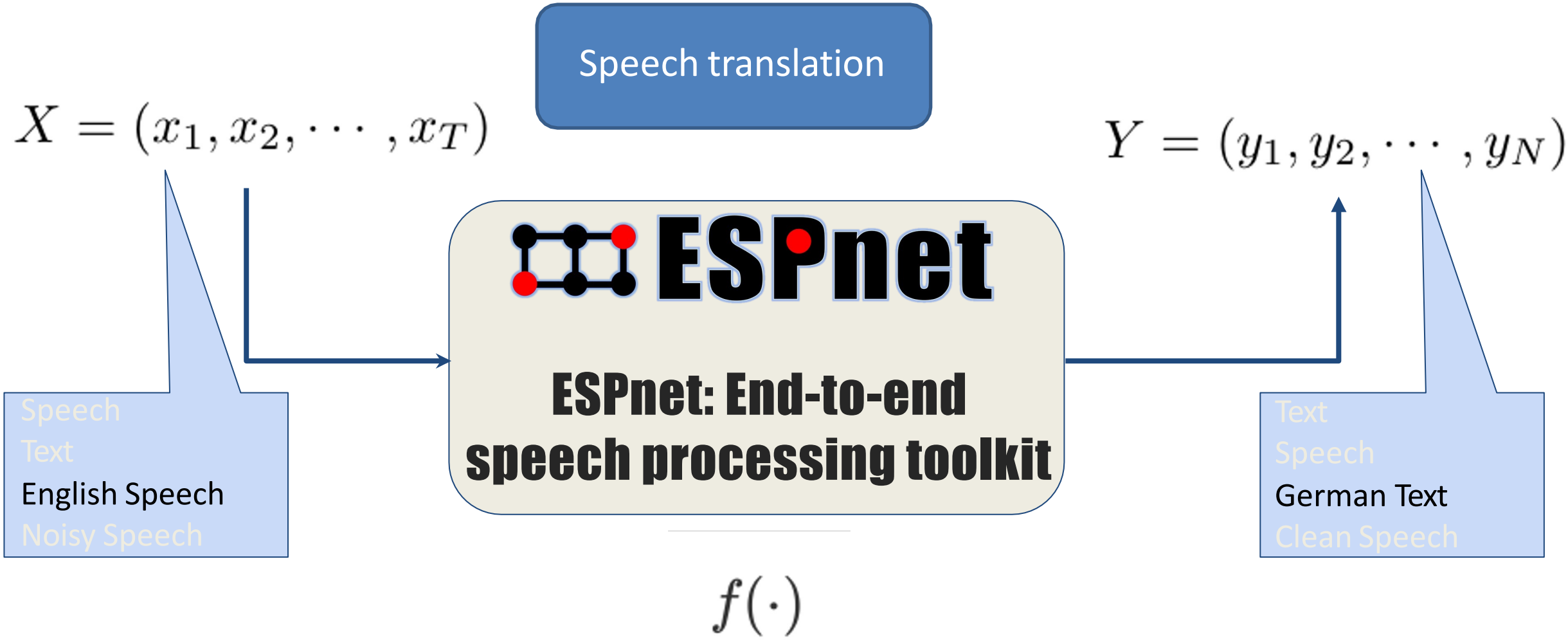
Unified view → Unified software design



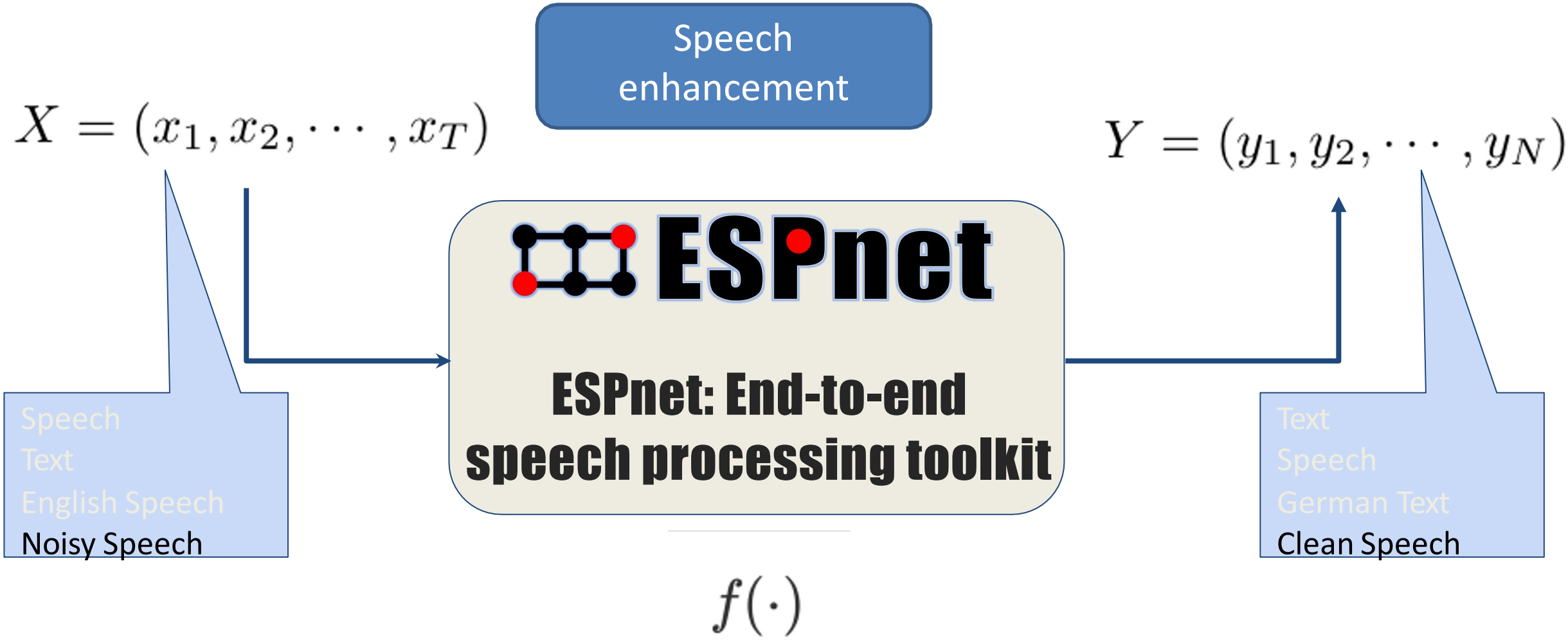
Unified view → Unified software design



Unified view → Unified software design



Unified view → Unified software design



Unified form → Unified software design

$$X = (x_1, x_2, \cdots, x_T)$$

$$Y = (y_1, y_2, \cdots, y_N)$$

CTC
Attention
Joint
CTC/Attention
RNN-T
Transformer, etc.

$$f(\cdot)$$

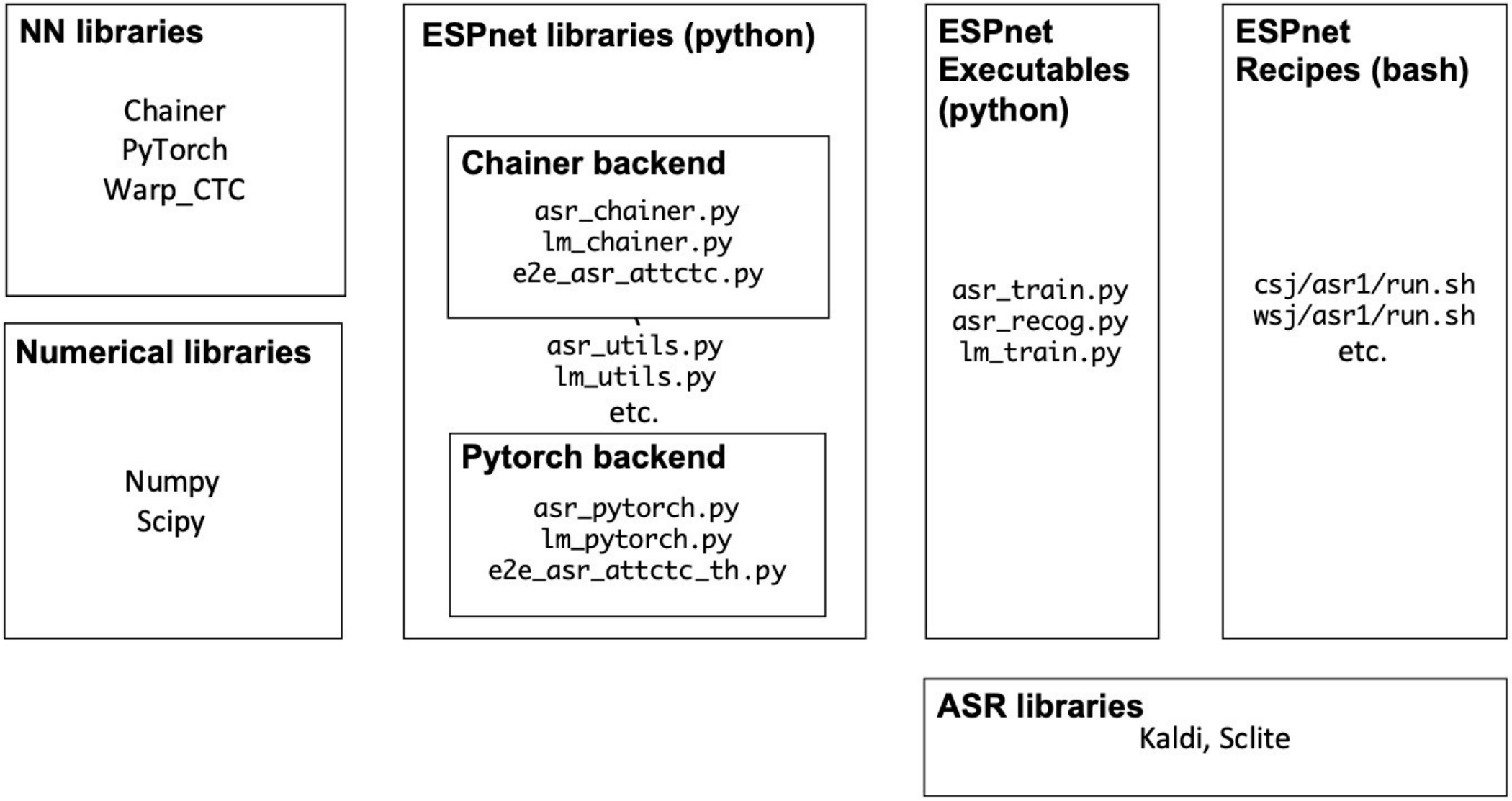
Unified form → Unified software design

$$X = (x_1, x_2, \cdots, x_T)$$

$$Y = (y_1, y_2, \cdots, y_N)$$

- Many speech processing applications can be **unified** based on seq2seq
- **Nemo, Fairseq, Lingvo, Espresso, SpeechBrain, Asteroid** and other toolkits also fully make use of these functions
- We are closely collaborating/interacting with them

Software Architecture of ESPnet



Software Requirements

- Python 3.6.1+
- gcc 4.9+ for PyTorch1.0.0+
- Cuda 8.0, 9.0, 9.1, 10.0 depending on each DNN library
- Cudnn 6+, 7+
- NCCL 2.0+ (for the use of multi-GPUs)

Ubuntu System Packages

- $ sudo apt-get install cmake
- $ sudo apt-get install sox
- $ sudo apt-get install libsndfile1-dev
- $ sudo apt-get install ffmpeg
- $ sudo apt-get install flac

PyTorch Environment

- $ git clone https://github.com/espnet/espnet
- $ cd <espnet-root>/tools
- $ CONDA_TOOLS_DIR=$(dirname ${CONDA_EXE})/..
- $ ./setup_anaconda.sh ${CONDA_TOOLS_DIR} espnet 3.9
- $ make

Directory structure of ESPnet

```
espnet/                 # Python modules
utils/                  # Utility scripts of ESPnet
test/                   # Unit test
test_utils/             # Unit test for executable scripts
egs/                    # The complete recipe for each corpora
    an4/                # AN4 is tiny corpus and can be obtained freely, so it might be suitable f
      asr1/             # ASR recipe
          - run.sh      # Executable script
          - cmd.sh      # To select the backend for job scheduler
          - path.sh     # Setup script for environment variables
          - conf/       # Containing Configuration files
          - steps/      # The steps scripts from Kaldi
          - utils/      # The utils scripts from Kaldi
      tts1/             # TTS recipe
    ...
```

Directory structure of ESPnet

```
espnet/  # Python modules of espnet1
espnet2/ # Python modules of espnet2
egs/     # espnet1 recipes
egs2/    # espnet2 recipes
```

```
egs2/an4/asr1/
 - conf/      # Configuration files for training, inference, etc.
 - scripts/   # Bash utilities of espnet2
 - pyscripts/ # Python utilities of espnet2
 - steps/     # From Kaldi utilities
 - utils/     # From Kaldi utilities
 - db.sh      # The directory path of each corpora
 - path.sh    # Setup script for environment variables
 - cmd.sh     # Configuration for your backend of job scheduler
 - run.sh     # Entry point
 - asr.sh     # Invoked by run.sh
```

run.sh

```
train_set=train
valid_set=dev
test_sets="dev test"

asr_config=conf/train_asr_branchformer.yaml
inference_config=conf/decode_asr_branchformer.yaml

lm_config=conf/train_lm_transformer.yaml
use_lm=false
use_wordlm=false

# speed perturbation related
# (train_set will be "${train_set}_sp" if speed_perturb_factors is specified)
speed_perturb_factors="0.9 1.0 1.1"

./asr.sh \
--nj 32 \
--inference_nj 32 \
--ngpu 4 \
--lang zh \
--audio_format "flac.ark" \
--feats_type raw \
--token_type char \
--use_lm ${use_lm} \
--use_word_lm ${use_wordlm} \
--lm_config "${lm_config}" \
--asr_config "${asr_config}" \
--inference_config "${inference_config}" \
--train_set "${train_set}" \
--valid_set "${valid_set}" \
--test_sets "${test_sets}" \
--speed_perturb_factors "${speed_perturb_factors}" \
--asr_speech_fold_length 512 \
--asr_text_fold_length 150 \
--lm_fold_length 150 \
--lm_train_text "data/${train_set}/text" "$@"
```

ESPnet Model Zoo

- https://github.com/espnet/espnet_model_zoo
- https://huggingface.co/models?filter=espnet

- **Transformer**

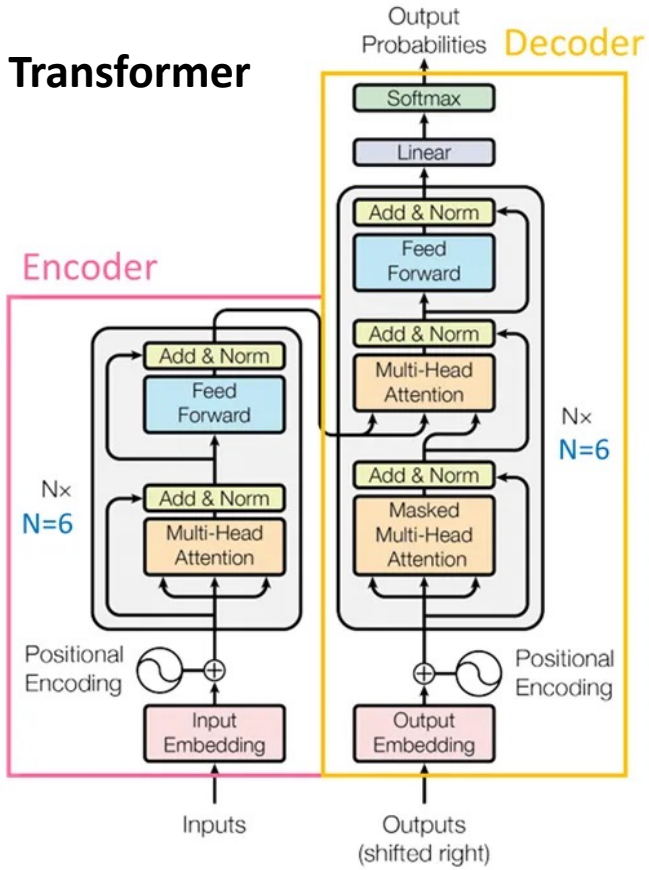
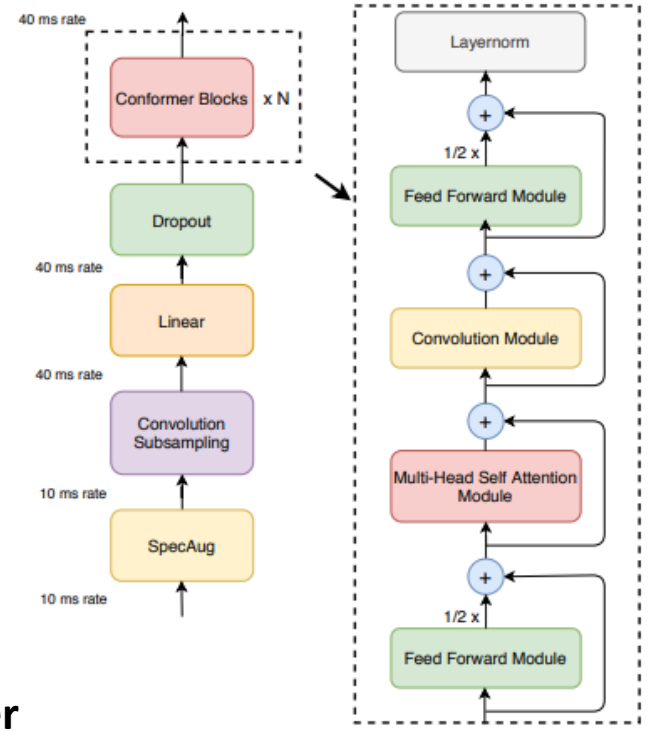
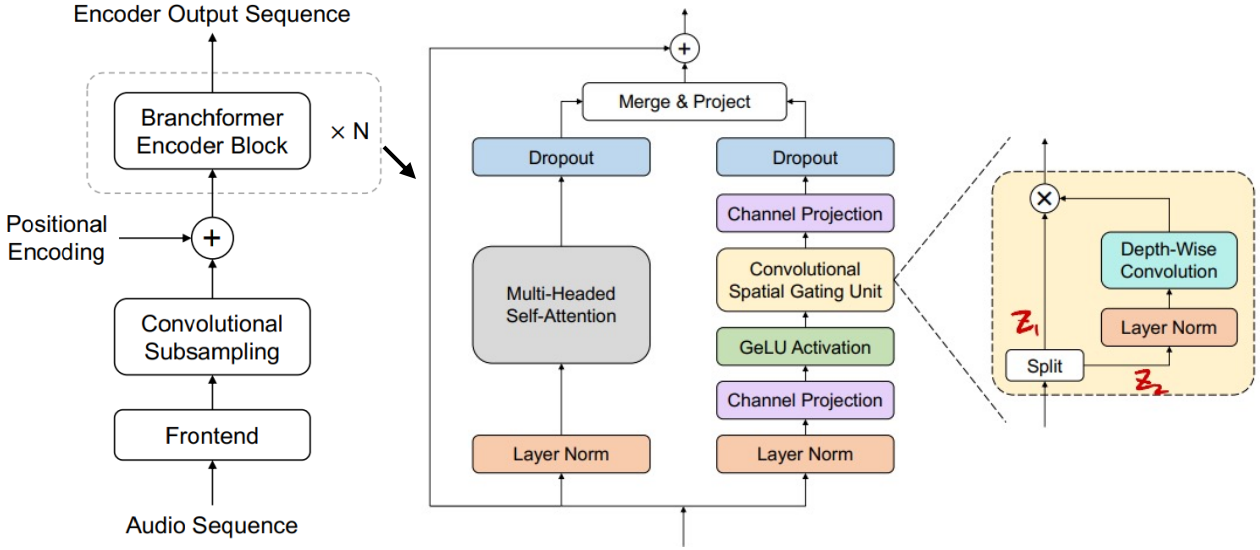


Figure 1: The Transformer - model architecture.

- **Conformer**



- **Branchformer**



Conformer, Branchformer & E-Branchformer

Conformer

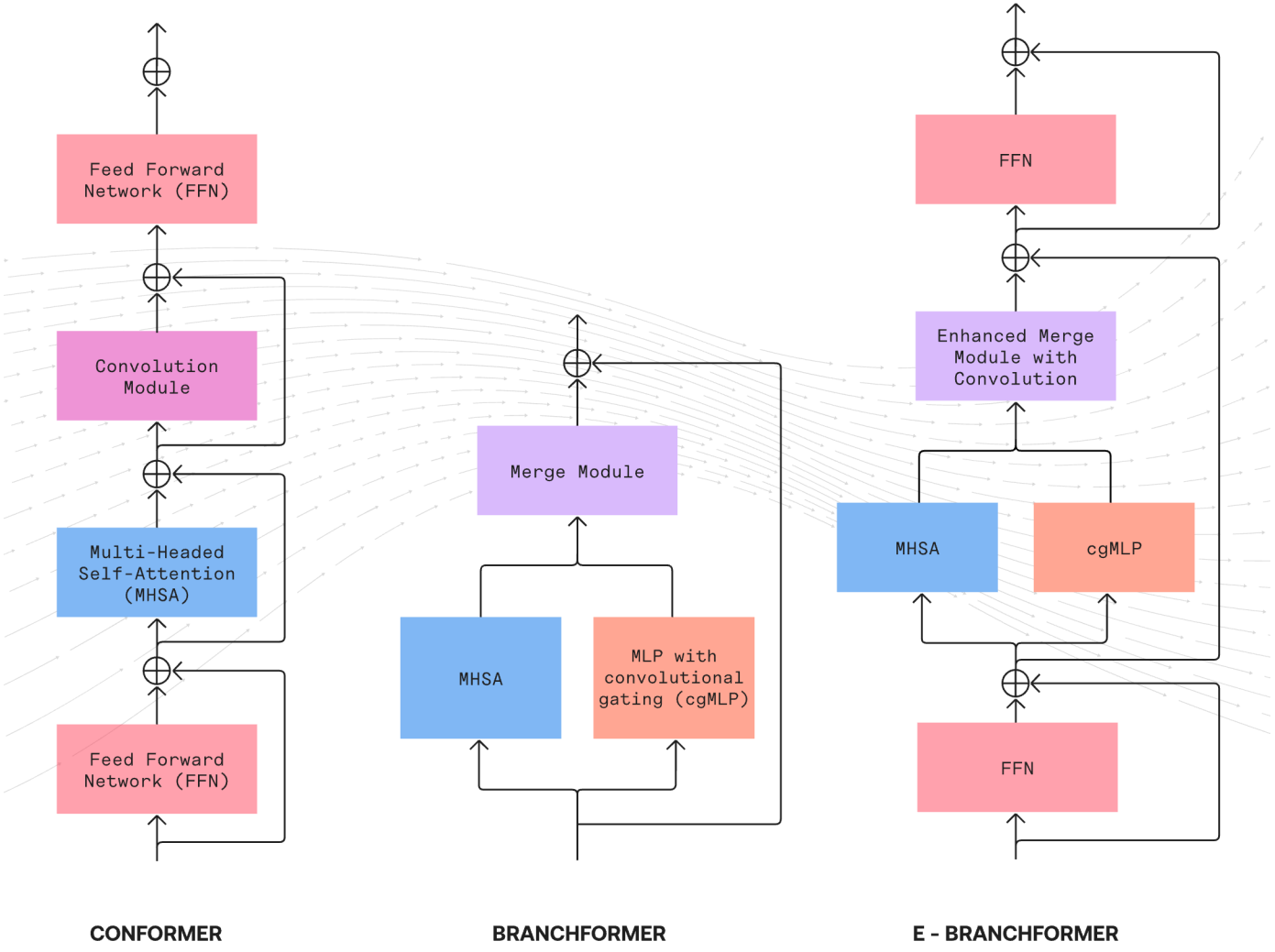
- applying convolution to model the local context

Branchformer

- Local-context branch using MLP with convolutional gating (cgMLP)
- Global-context branch using multi-headed self-attention
- Merging the module with a linear projection layer

E-Branchformer

- enhanced the merging module by introducing additional depth-wise convolution
- stacking FFN together to improve the model's capacity.



egs2/aishell/asr1/conf/tuning/train_asr_branchformer_e24_amp.yaml

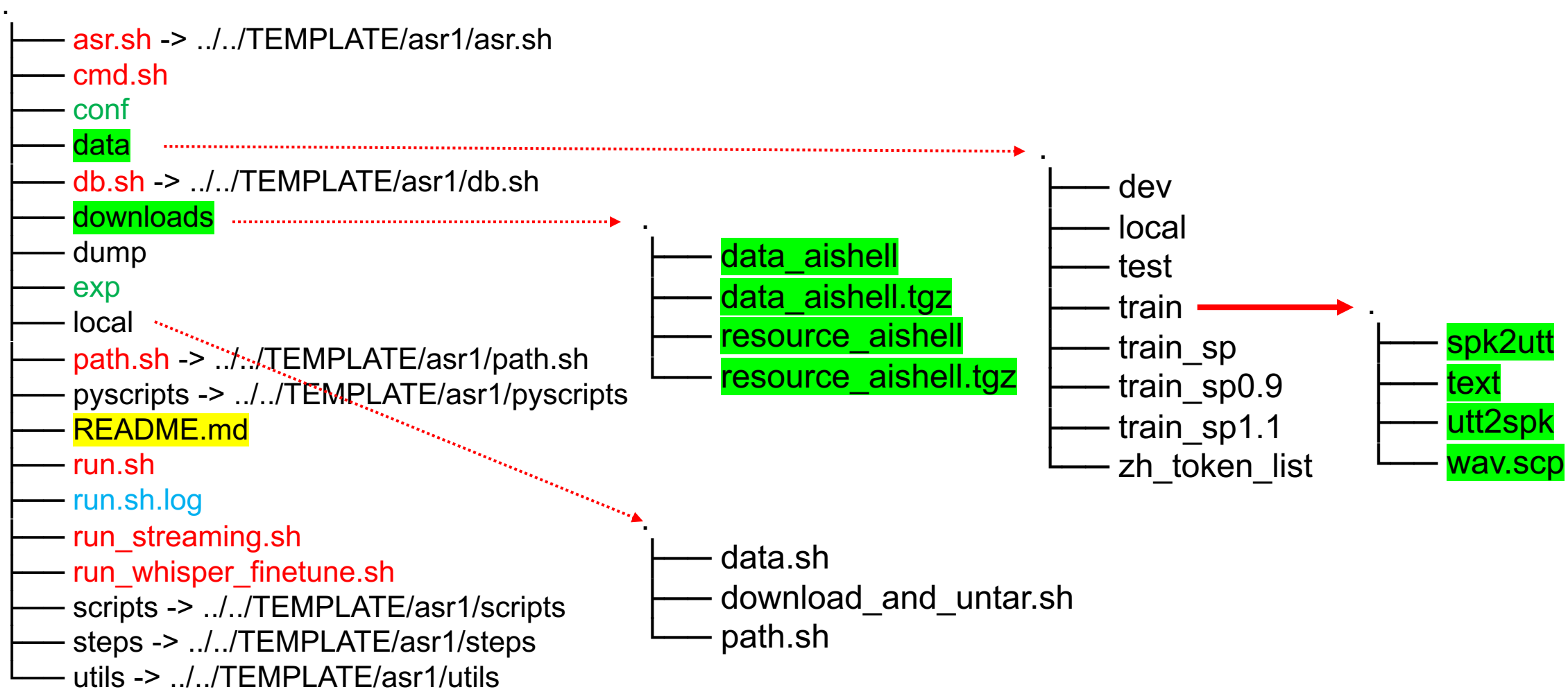
yaml

```yaml
# network architecture
# encoder related
encoder: branchformer
encoder_conf:
output_size: 256
use_attn: true
attention_heads: 4
attention_layer_type: rel_selfattn
pos_enc_layer_type: rel_pos
rel_pos_type: latest
use_cgmlp: true
cgmlp_linear_units: 2048
cgmlp_conv_kernel: 31
use_linear_after_conv: false
gate_activation: identity
merge_method: concat
cgmlp_weight: 0.5
attn_branch_drop_rate: 0.0
num_blocks: 24
dropout_rate: 0.1
positional_dropout_rate: 0.1
attention_dropout_rate: 0.1
input_layer: conv2d
stochastic_depth_rate: 0.0

# decoder related
decoder: transformer
decoder_conf:
attention_heads: 4
linear_units: 2048
num_blocks: 6
dropout_rate: 0.1
positional_dropout_rate: 0.1
self_attention_dropout_rate: 0.
src_attention_dropout_rate: 0.

# hybrid CTC/attention
model_conf:
ctc_weight: 0.3
lsm_weight: 0.1 # label smoothing option
length_normalized_loss: false

# minibatch related
batch_type: numel
batch_bins: 25000000

# optimization related
accum_grad: 1
grad_clip: 5
max_epoch: 60
val_scheduler_criterion:
- valid
- acc
best_model_criterion:
- - valid
- acc
- max
keep_nbest_models: 10

optim: adam
optim_conf:
lr: 0.001
weight_decay: 0.000001
scheduler: warmuplr
scheduler_conf:
warmup_steps: 35000

num_workers: 4 # num of workers of data loader
use_amp: true # automatic mixed precision
unused_parameters: false # set as true if some params are unused in DDP

# SPECAUG
specaug: specaug
specaug_conf:
apply_time_warp: true
time_warp_window: 5
time_warp_mode: bicubic
apply_freq_mask: true
freq_mask_width_range:
- 0
- 27
num_freq_mask: 2
apply_time_mask: true
time_mask_width_ratio_range:
- 0.
- 0.05
num_time_mask: 10
```

asr.sh

- **Stage 1: Data preparation**: download raw data, split the entire set into train/dev/test, and prepare them in the Kaldi format
- **Stage 2: Speed perturbation** (one of the data augmentation methods)
- **Stage 3: Format wav.scp**: data/ -> dump/raw (dumps such pipe-style-wav to real audio file, and it can also change the audio-format and sampling rate.)
- **Stage 4: Remove long/short data**: dump/raw/org -> dump/raw
- **Stage 5: Generate token_list** from dump/raw/train_nodev/text using BPE.
- **Stage 6: LM collect stats**
- **Stage 7: LM Training**
- **Stage 8: Calc perplexity**
- **Stage 9: Ngram Training**
- **Stage 10: ASR collect stats**: train_set=dump/raw/train_nodev, valid_set=dump/raw/train_dev
- **Stage 11: ASR Training**: train_set=dump/raw/train_nodev, valid_set=dump/raw/train_dev
- **Stage 12: Decoding**
- **Stage 13: Scoring**
- **Stage 14: Pack model**
- **Stage 15: Upload model to Zenodo**
- **Stage 16: Upload model to HuggingFace**

~/espnet/egs2/aishell/asr1



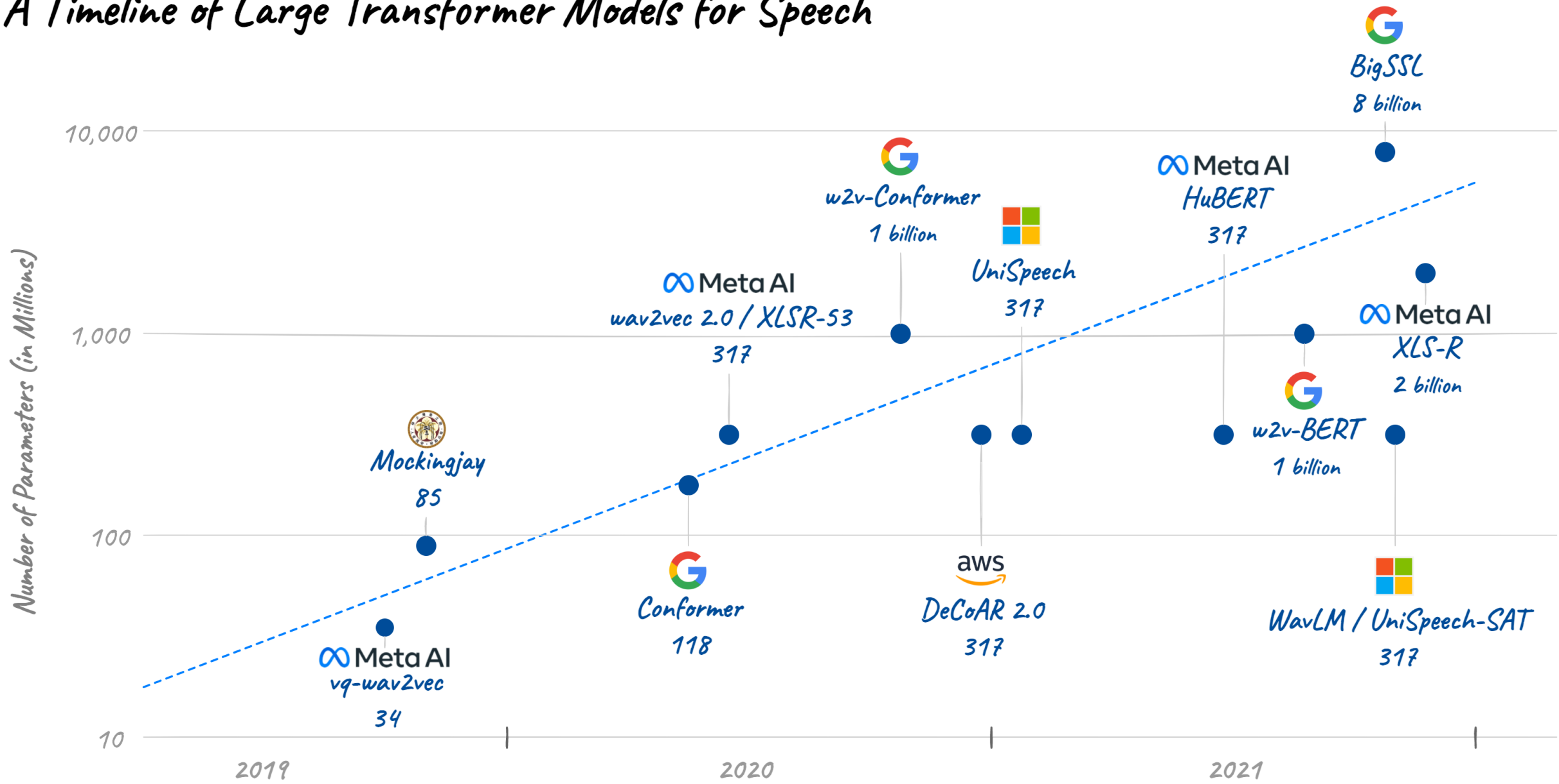
data/train/, test/, dev/

- wav.scp
- Speaker0001-0 ~/kaldi-data/OC16-CE80/Training_Set/train/Speaker0001/0.wav
- Speaker0001-1 ~/kaldi-data/OC16-CE80/Training_Set/train/Speaker0001/1.wav
- Speaker0001-10 ~/kaldi-data/OC16-CE80/Training_Set/train/Speaker0001/10.wav
-
- text
- Speaker0001-0 打开 Notepad 编辑 文件
- Speaker0001-1 结束 teleconference
- Speaker0001-10 Capital Hotel 你 觉 的 怎么样
-
- utt2spk
- Speaker0001-0 Speaker0001
- Speaker0001-1 Speaker0001
- Speaker0001-10 Speaker0001
-
- spk2utt
- Speaker0001 Speaker0001-0 Speaker0001-1 Speaker0001-10 Speaker0001-11 … … …
- Speaker0003 Speaker0003-1 Speaker0003-10 Speaker0003-11 Speaker0003-115 … … …
- Speaker0004 Speaker0004-0 Speaker0004-1 Speaker0004-10 Speaker0004-11 … … …

Stage 11: ASR Training

```
${python} -m espnet2.bin.launch \
--cmd "${cuda_cmd} --name ${jobname}" \
--log "${asr_exp}"/train.log \
--ngpu "${ngpu}" \
--num_nodes "${num_nodes}" \
--init_file_prefix "${asr_exp}"/.dist_init_ \
--multiprocessing_distributed true -- \
${python} -m espnet2.bin.${asr_task}_train \
--use_preprocessor true \
--bpemodel "${bpemodel}" \
--token_type "${token_type}" \
--token_list "${token_list}" \
--non_linguistic_symbols "${nlsyms_txt}" \
--cleaner "${cleaner}" \
--g2p "${g2p}" \
--valid_data_path_and_name_and_type "${_asr_valid_dir}/${_scp},speech,${_type}" \
--valid_shape_file "${asr_stats_dir}/valid/speech_shape" \
--resume true \
${pretrained_model:+--init_param $pretrained_model} \
--ignore_init_mismatch ${ignore_init_mismatch} \
--fold_length "${_fold_length}" \
--output_dir "${asr_exp}" \
${_opts} ${asr_args}
```

A Timeline of Large Transformer Models for Speech



jonathanbgn.com

Use self-supervised pre-trained models as the front-end

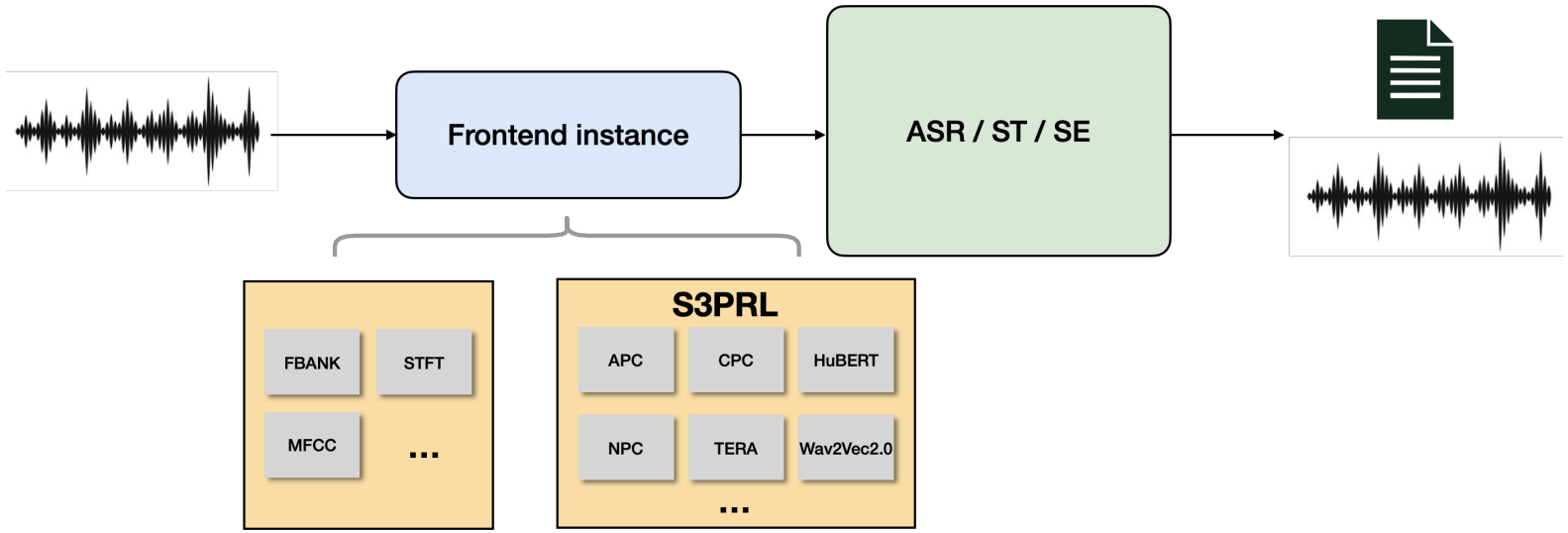
- SUPERB benchmark
  https://superbbenchmark.org/

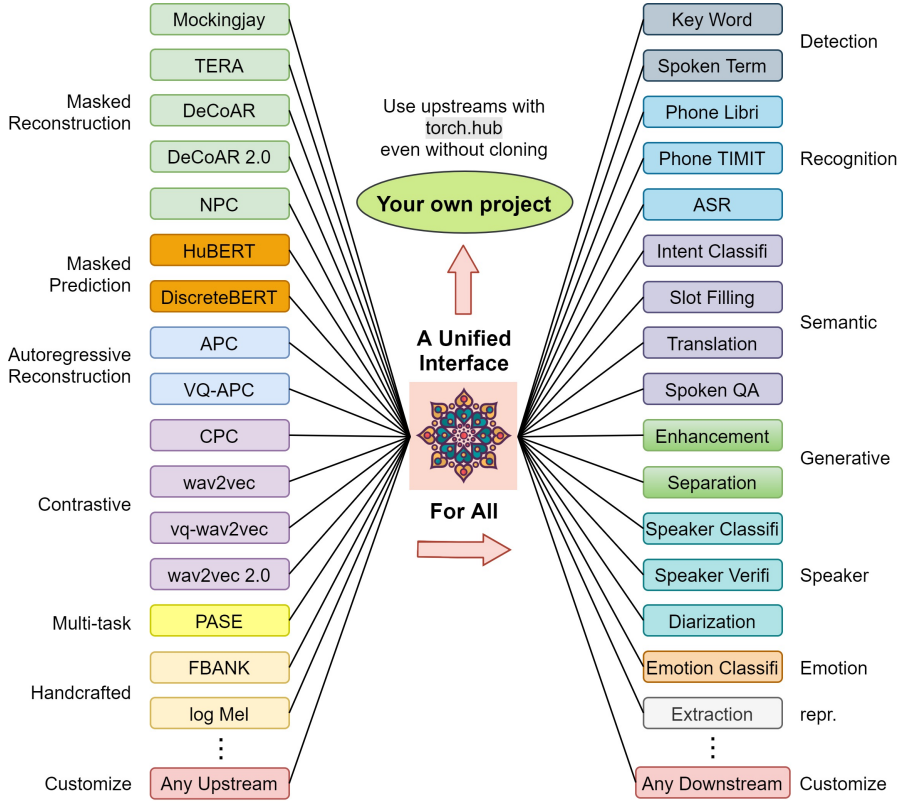
Table 1. Universal speech representation evaluation on SUPERB benchmark. ParaL denote Paralinguistics aspect of speech.

Method	#Params	Corpus	Speaker			Content				Semantics			ParaL	Overall
			SID	ASV	SD	PR	ASR	KS	QbE	IC	SF		ER	
			Acc ↑	EER ↓	DER ↓	PER ↓	WER ↓	Acc ↑	MTWV ↑	Acc ↑	F1 ↑	CER ↓	Acc ↑	Score ↑
FBANK	0	-	8.5E-4	9.56	10.05	82.01	23.18	8.63	0.0058	9.10	69.64	52.94	35.39	40.5
PASE+ (Ravanelli et al., 2020)	7.83M	LS 50 hr	37.99	11.61	8.68	58.87	25.11	82.54	0.0072	29.82	62.14	60.17	57.86	55.1
APC (Chung et al., 2019)	4.11M	LS 360 hr	60.42	8.56	10.53	41.98	21.28	91.01	0.0310	74.69	70.46	50.89	59.33	66.0
VQ-APC (Chung et al., 2020)	4.63M	LS 360 hr	60.15	8.72	10.45	41.08	21.20	91.11	0.0251	74.48	68.53	52.91	59.66	65.6
NPC (Liu et al., 2020a)	19.38M	LS 360 hr	55.92	9.40	9.34	43.81	20.20	88.96	0.0246	69.44	72.79	48.44	59.08	65.2
Mockingjay (Liu et al., 2020c)	85.12M	LS 360 hr	32.29	11.66	10.54	70.19	22.82	83.67	6.6E-04	34.33	61.59	58.89	50.28	53.5
TERA (Liu et al., 2020b)	21.33M	LS 360 hr	57.57	15.89	9.96	49.17	18.17	89.48	0.0013	58.42	67.50	54.17	56.27	62.0
modified CPC (Rivière et al., 2020)	1.84M	LL 60k hr	39.63	12.86	10.38	42.54	20.18	91.88	0.0326	64.09	71.19	49.91	60.96	63.2
wav2vec (Schneider et al., 2019)	32.54M	LS 960 hr	56.56	7.99	9.9	31.58	15.86	95.59	0.0485	84.92	76.37	43.71	59.79	69.9
vq-wav2vec (Baevski et al., 2020a)	34.15M	LS 960 hr	38.80	10.38	9.93	33.48	17.71	93.38	0.0410	85.68	77.68	41.54	58.24	67.7
wav2vec 2.0 Base (Baevski et al., 2020b)	95.04M	LS 960 hr	75.18	6.02	6.08	5.74	6.43	96.23	0.0233	92.35	88.30	24.77	63.43	79.0
HuBERT Base (Hsu et al., 2021a)	94.68M	LS 960 hr	81.42	5.11	5.88	5.41	6.42	96.30	0.0736	98.34	88.53	25.20	64.92	80.8
WavLM Base	94.70M	LS 960 hr	84.51	4.69	4.83	4.84	6.21	96.79	0.0870	98.63	89.38	22.86	65.94	81.9
- w/o utterance mixing	94.70M	LS 960 hr	84.39	4.91	6.03	4.85	6.08	96.79	0.0799	98.42	88.69	23.43	65.55	81.5
- w/o structure modification	94.68M	LS 960 hr	84.74	4.61	4.72	5.22	6.80	96.79	0.0956	98.31	88.56	24.00	65.60	81.7
WavLM Base+	94.70M	Mix 94k hr	86.84	4.26	4.07	4.07	5.64	96.69	**0.0990**	**99.16**	89.73	21.54	67.98	82.8
wav2vec 2.0 Large (Baevski et al., 2020b)	317.38M	LL 60k hr	86.14	5.65	5.62	4.75	3.75	96.66	0.0489	95.28	87.11	27.31	65.64	80.8
HuBERT Large (Hsu et al., 2021a)	316.61M	LL 60k hr	90.33	5.98	5.75	3.53	3.62	95.29	0.0353	98.76	89.81	21.76	67.62	82.2
WavLM Large	316.62M	Mix 94k hr	**95.25**	**4.04**	**3.47**	**3.09**	**3.51**	**97.40**	0.0827	99.10	**92.25**	**17.61**	**70.03**	**84.6**

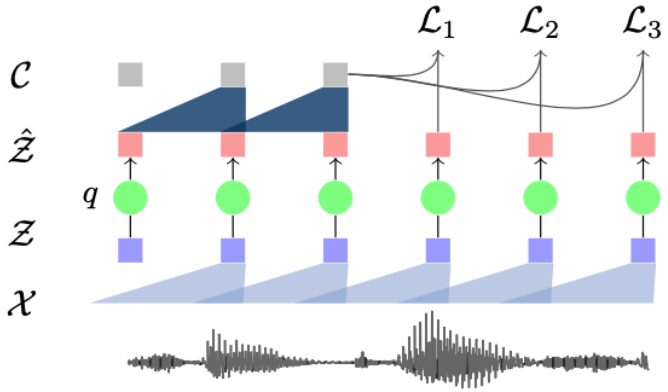
Combine ESPnet with S3PRL [Yang+ 2021, Chang+ 2021]

- Self-supervised pretraining on speech data have achieved a lot of progress, like wav2vec2.0 [Baevski+ 2020], Hubert [Hsu+ 2021], etc.
- S3PRL* toolkit provides an integration of pretrained speech representation models and speech tasks, e.g., wav2vec2.0 + LSTM acoustic model for ASR.
- Support combine the pretrained models with advanced end-to-end speech processing models in a simple way.

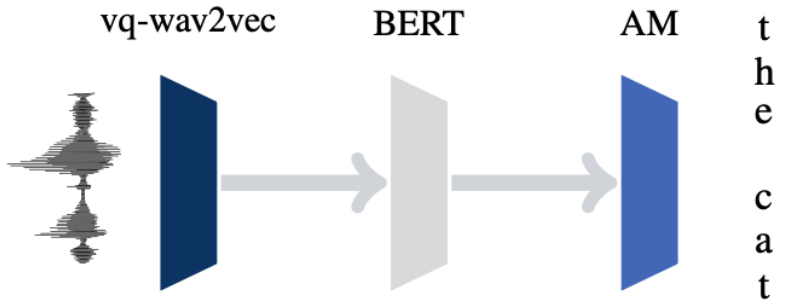




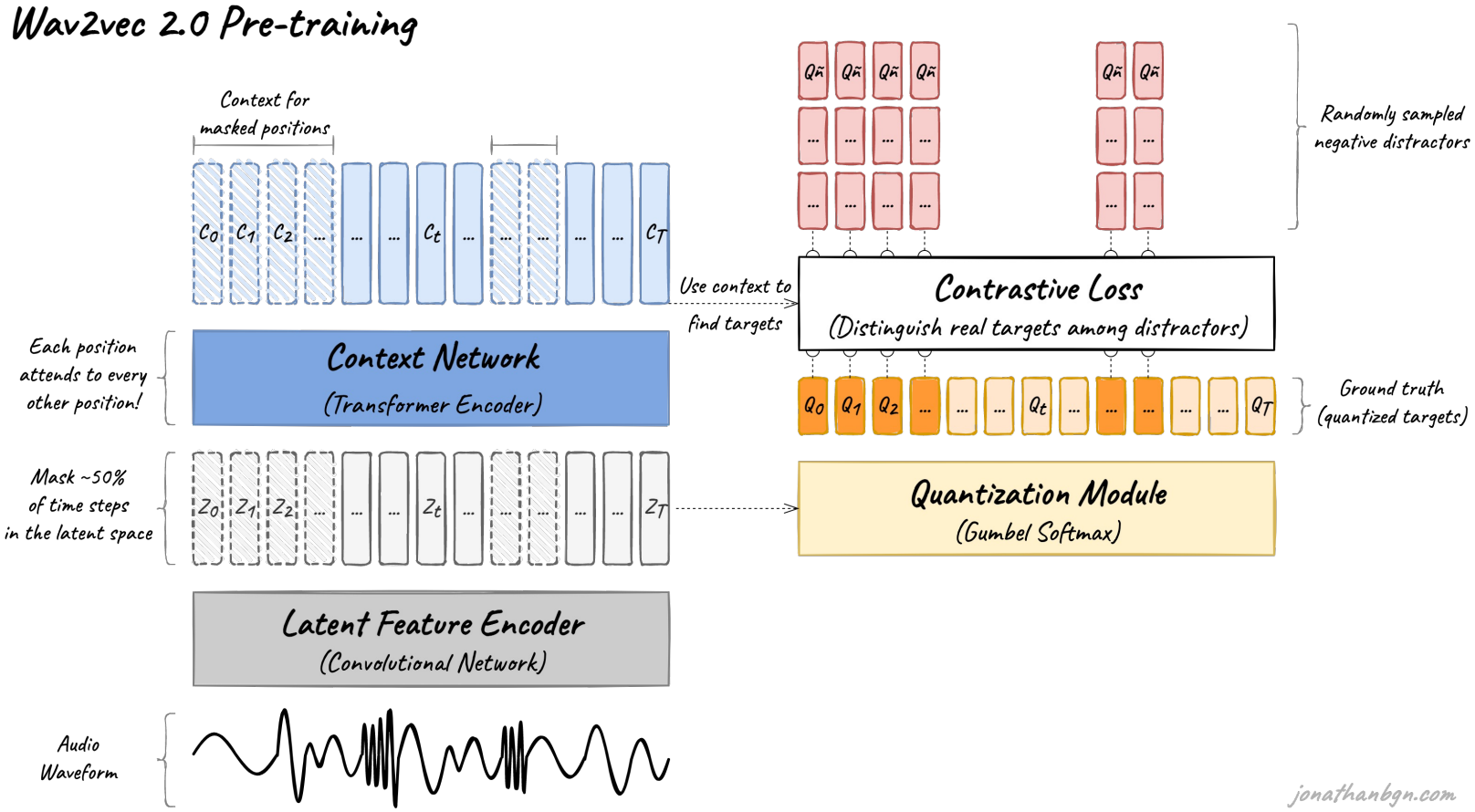
*: https://github.com/s3prl/s3prl



(a) vq-wav2vec



(b) Discretized speech training pipeline



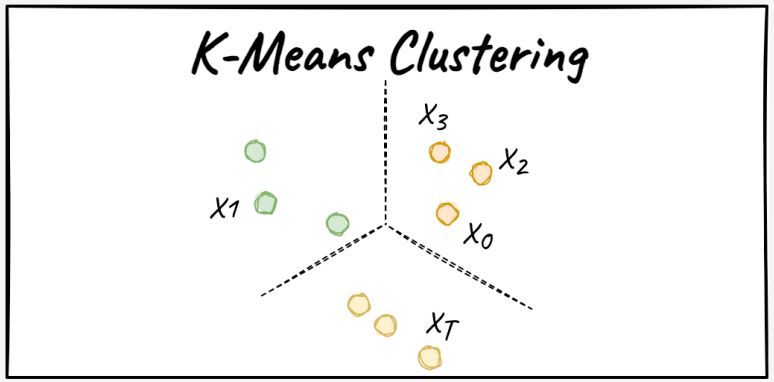
HuBERT Training Process

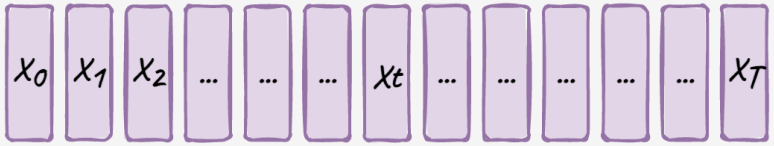
Alternate between two steps

STEP 1: Discover "hidden units" targets

Hidden units embeddings

Assign each feature vector X_t to a hidden unit cluster





Clustering Feature Extraction

Audio Waveform

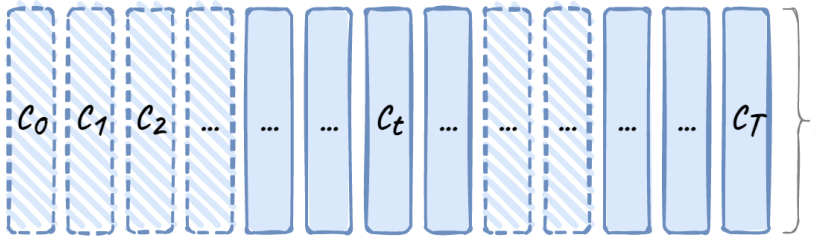
Use hidden units as targets to predict

Re-use intermediate layer features for better clustering

Compute directly from raw waveform

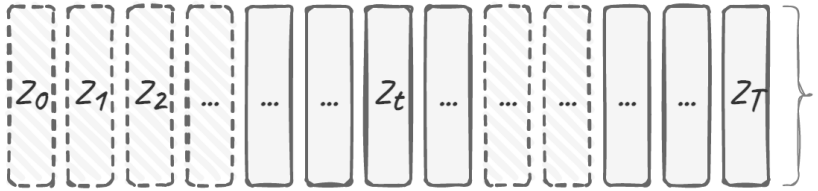
STEP 2: Predict targets at masked positions

Cross-Entropy Loss
(Predict hidden units at masked locations)



Use context representations for prediction

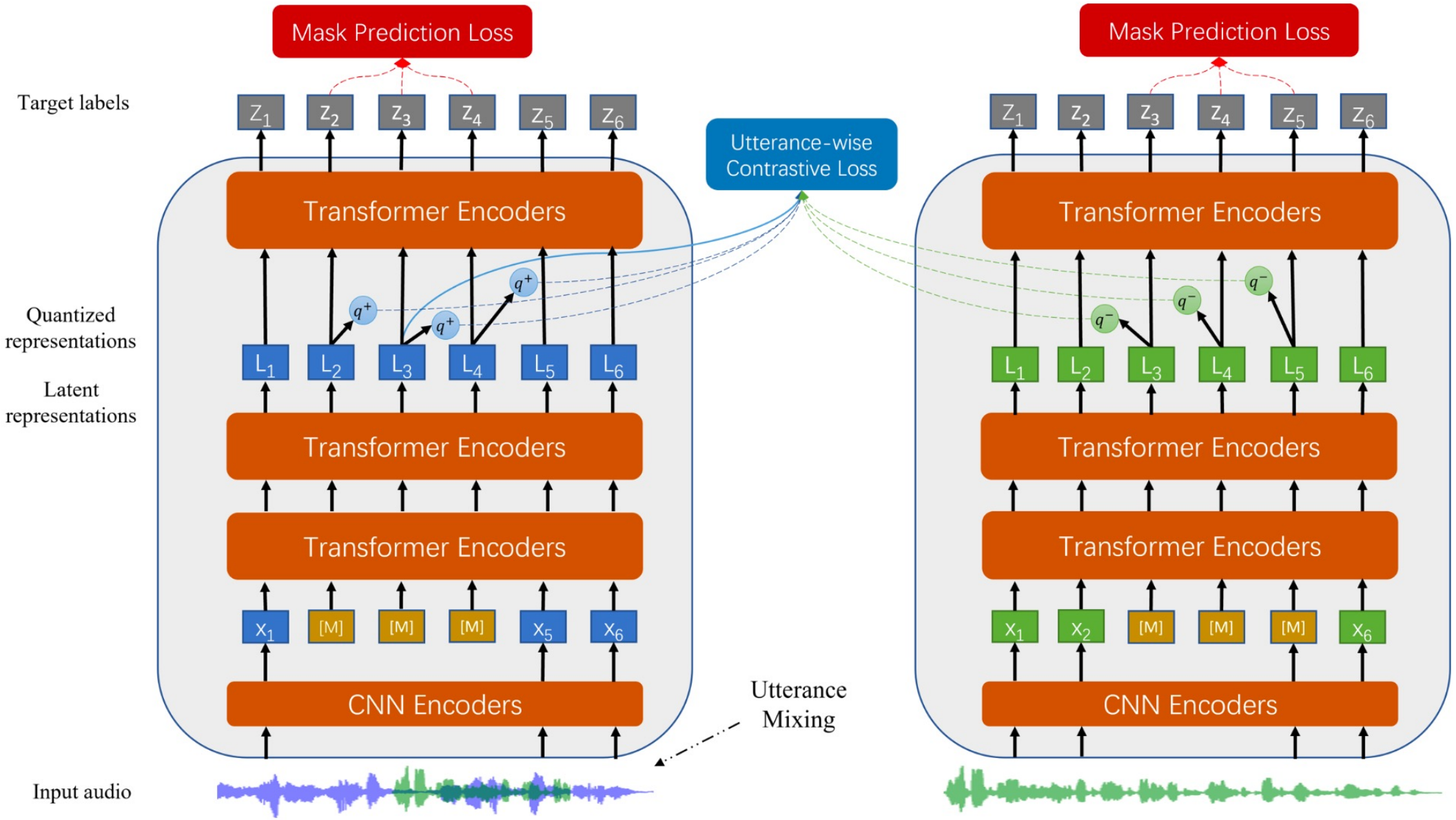
Context Network
(Transformer Encoder)



Mask ~50% of time steps

Latent Feature Encoder
(Convolutional Network)

UniSpeech-SAT/WavLM



egs2/librispeech/asr1/conf/tuning/train_asr_conformer7_wav2vec2_960hr_large.yaml

```
batch_type: numel
batch_bins: 40000000
accum_grad: 3
max_epoch: 35
patience: none
init: none
best_model_criterion:
-   - valid
    - acc
    - max
keep_nbest_models: 10
unused_parameters: true
freeze_param: [
"frontend.upstream"
]

frontend: s3prl
frontend_conf:
    frontend_conf:
        upstream: wav2vec2_large_960 ←
    download_dir: ./hub
    multilayer_feature: True

preencoder: linear
preencoder_conf:
    input_size: 1024 ←
    output_size: 80

encoder: conformer
encoder_conf:
    output_size: 512
    attention_heads: 8
    linear_units: 2048
    num_blocks: 12
    dropout_rate: 0.1
    positional_dropout_rate: 0.1
    attention_dropout_rate: 0.1
    input_layer: conv2d2
    normalize_before: true
    macaron_style: true
    pos_enc_layer_type: "rel_pos"
    selfattention_layer_type: "rel_selfattn"
    activation_type: "swish"
    use_cnn_module:  true
    cnn_module_kernel: 31

decoder: transformer
decoder_conf:
    attention_heads: 8
    linear_units: 2048
    num_blocks: 6
    dropout_rate: 0.1
    positional_dropout_rate: 0.1
    self_attention_dropout_rate: 0.1
    src_attention_dropout_rate: 0.1

model_conf:
    ctc_weight: 0.3
    lsm_weight: 0.1
    length_normalized_loss: false
    extract_feats_in_collect_stats: false

optim: adam
optim_conf:
    lr: 0.0025
scheduler: warmuplr
scheduler_conf:
    warmup_steps: 40000

specaug: specaug
specaug_conf:
    apply_time_warp: true
    time_warp_window: 5
    time_warp_mode: bicubic
    apply_freq_mask: true
    freq_mask_width_range:
    - 0
    - 30
    num_freq_mask: 2
    apply_time_mask: true
    time_mask_width_range:
    - 0
    - 40
    num_time_mask: 2
```

egs2/librispeech/asr1/conf/tuning/train_asr_conformer7_hubert_ll60k_large.yaml

```
batch_type: numel
batch_bins: 40000000
accum_grad: 3
max_epoch: 35
patience: none
init: none
best_model_criterion:
-   - valid
    - acc
    - max
keep_nbest_models: 10
unused_parameters: true
freeze_param: [
"frontend.upstream"
]

frontend: s3prl
frontend_conf:
    frontend_conf:
        upstream: hubert_large_ll60k ←
    download_dir: ./hub
    multilayer_feature: True

preencoder: linear
preencoder_conf:
    input_size: 1024 ←
    output_size: 80

encoder: conformer
encoder_conf:
    output_size: 512
    attention_heads: 8
    linear_units: 2048
    num_blocks: 12
    dropout_rate: 0.1
    positional_dropout_rate: 0.1
    attention_dropout_rate: 0.1
    input_layer: conv2d2
    normalize_before: true
    macaron_style: true
    pos_enc_layer_type: "rel_pos"
    selfattention_layer_type: "rel_selfattn"
    activation_type: "swish"
    use_cnn_module:  true
    cnn_module_kernel: 31

decoder: transformer
decoder_conf:
    attention_heads: 8
    linear_units: 2048
    num_blocks: 6
    dropout_rate: 0.1
    positional_dropout_rate: 0.1
    self_attention_dropout_rate: 0.1
    src_attention_dropout_rate: 0.1

model_conf:
    ctc_weight: 0.3
    lsm_weight: 0.1
    length_normalized_loss: false
    extract_feats_in_collect_stats: false

optim: adam
optim_conf:
    lr: 0.0025
scheduler: warmuplr
scheduler_conf:
    warmup_steps: 40000

specaug: specaug
specaug_conf:
    apply_time_warp: true
    time_warp_window: 5
    time_warp_mode: bicubic
    apply_freq_mask: true
    freq_mask_width_range:
    - 0
    - 30
    num_freq_mask: 2
    apply_time_mask: true
    time_mask_width_range:
    - 0
    - 40
    num_time_mask: 2
```

~/espnet/egs2/egs2/librispeech/asr1/conf/tuning/train_asr_conformer7_wavlm_large.yaml

```
batch_type: numel
batch_bins: 40000000
accum_grad: 3
max_epoch: 35
patience: none
init: none
best_model_criterion:
-   - valid
    - acc
    - max
keep_nbest_models: 10
unused_parameters: true
freeze_param: [
"frontend.upstream"
]

frontend: s3prl
frontend_conf:
    frontend_conf:
        upstream: wavlm_large ←
    download_dir: ./hub
    multilayer_feature: True

preencoder: linear
preencoder_conf:
    input_size: 1024 ←
    output_size: 80
```

```
encoder: conformer
encoder_conf:
    output_size: 512
    attention_heads: 8
    linear_units: 2048
    num_blocks: 12
    dropout_rate: 0.1
    positional_dropout_rate: 0.1
    attention_dropout_rate: 0.1
    input_layer: conv2d2
    normalize_before: true
    macaron_style: true
    pos_enc_layer_type: "rel_pos"
    selfattention_layer_type: "rel_selfattn"
    activation_type: "swish"
    use_cnn_module:  true
    cnn_module_kernel: 31

decoder: transformer
decoder_conf:
    attention_heads: 8
    linear_units: 2048
    num_blocks: 6
    dropout_rate: 0.1
    positional_dropout_rate: 0.1
    self_attention_dropout_rate: 0.1
    src_attention_dropout_rate: 0.1
```

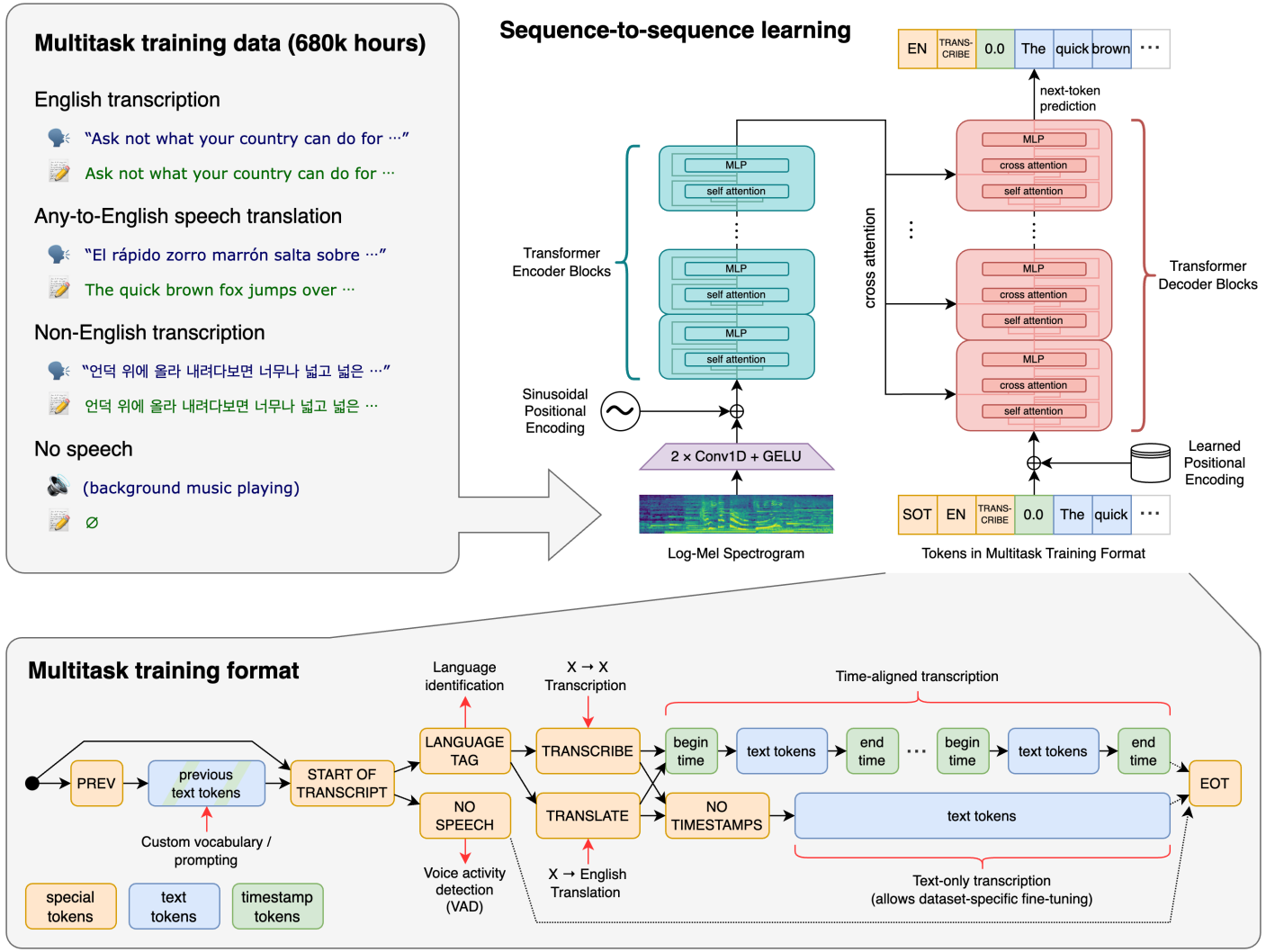
```
model_conf:
    ctc_weight: 0.3
    lsm_weight: 0.1
    length_normalized_loss: false
    extract_feats_in_collect_stats: false

optim: adam
optim_conf:
    lr: 0.0025
scheduler: warmuplr
scheduler_conf:
    warmup_steps: 40000


specaug: specaug
specaug_conf:
    apply_time_warp: true
    time_warp_window: 5
    time_warp_mode: bicubic
    apply_freq_mask: true
    freq_mask_width_range:
    - 0
    - 30
    num_freq_mask: 2
    apply_time_mask: true
    time_mask_width_range:
    - 0
    - 40
    num_time_mask: 2
```

OpenAI Whisper

- https://github.com/openai/whisper
- trained on 680,000 hours of multilingual and multitask supervised data collected from the web.
- enables transcription in multiple languages, as well as translation from those languages into English

Size	Layers	Width	Heads	Parameters	English-only	Multilingual
tiny	4	384	6	39 M	✓	✓
base	6	512	8	74 M	✓	✓
small	12	768	12	244 M	✓	✓
medium	24	1024	16	769 M	✓	✓
large	32	1280	20	1550 M	x	✓

- Finetuning
  - ESPnet
    - https://github.com/espnet/espnet/tree/master/egs2/aishell/asr1
  - Hugging Face Transformer
    - https://colab.research.google.com/github/sanchit-gandhi/notebooks/blob/main/fine_tune_whisper.ipynb



egs2/aishell/asr1/run_whisper_finetune.sh

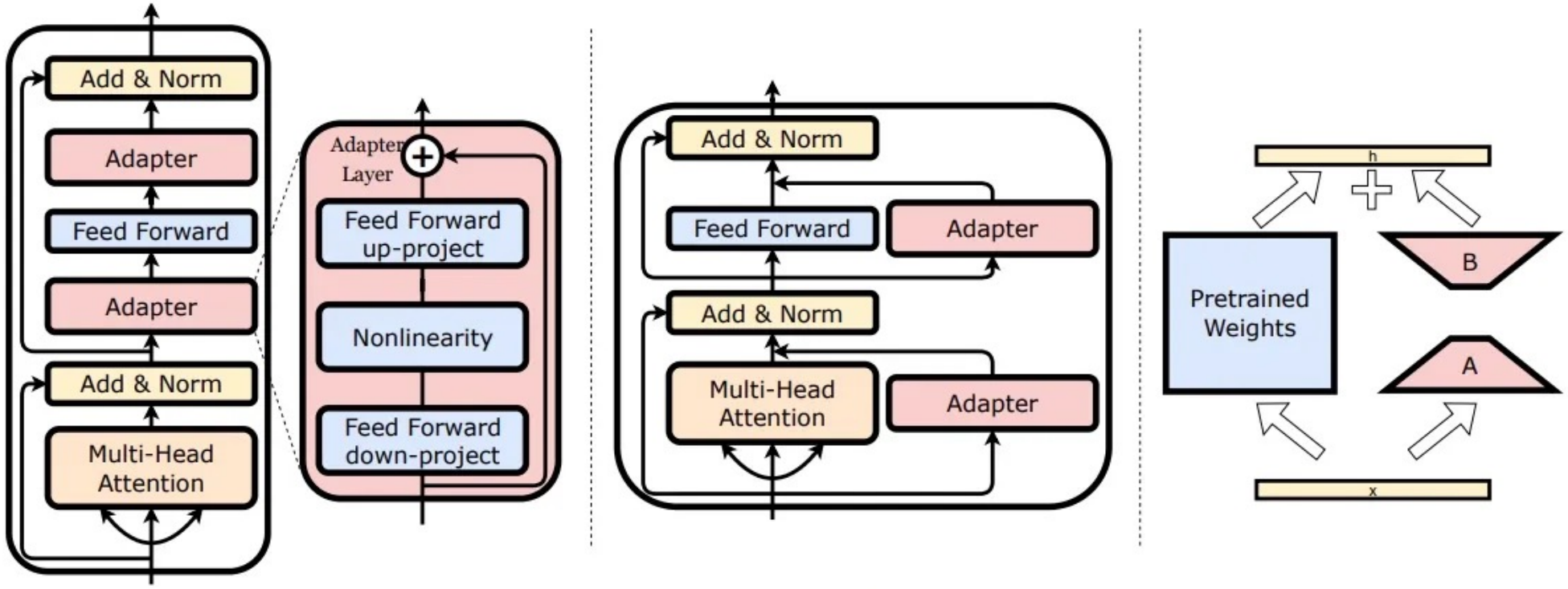
```
#!/usr/bin/env bash
# Set bash to 'debug' mode, it will exit on :
# -e 'error', -u 'undefined variable', -o ... 'error in pipeline', -x 'print commands',
set -e
set -u
set -o pipefail

train_set=train
valid_set=dev
test_sets="dev test"

asr_config=conf/tuning/train_asr_whisper_medium_finetune.yaml
inference_config=conf/tuning/decode_asr_whisper_noctc_beam10.yaml

lm_config=conf/train_lm_transformer.yaml
use_lm=false
use_wordlm=false

# speed perturbation related
# (train_set will be "${train_set}_sp" if speed_perturb_factors is specified)
speed_perturb_factors="0.9 1.0 1.1"
```

```
./asr.sh \
    --nj 32 \
    --gpu_inference true \
    --inference_nj 1 \
    --lang zh \
    --token_type whisper_multilingual \
    --feats_normalize "" \
    --audio_format "flac.ark" \
    --feats_type raw \
    --use_lm ${use_lm}                          \
    --use_word_lm ${use_wordlm}                     \
    --lm_config "${lm_config}"                   \
    --cleaner whisper_basic                      \
    --asr_config "${asr_config}"                  \
    --inference_config "${inference_config}"         \
    --train_set "${train_set}"                   \
    --valid_set "${valid_set}"                   \
    --test_sets "${test_sets}"                   \
    --speed_perturb_factors "${speed_perturb_factors}" \
    --asr_speech_fold_length 512 \
    --asr_text_fold_length 150 \
    --lm_fold_length 150 \
    --lm_train_text "data/${train_set}/text" "$@"
```

egs2/aishell/asr1/conf/tuning/train_asr_whisper_medium_finetune.yaml

```
normalize: null

encoder: whisper
encoder_conf:
    whisper_model: medium
    dropout_rate: 0.0
    use_specaug: true
    specaug_conf:
      apply_time_warp: true
      time_warp_window: 5
      time_warp_mode: bicubic
      apply_freq_mask: true
      freq_mask_width_range:
      - 0
      - 40
      num_freq_mask: 2
      apply_time_mask: true
      time_mask_width_ratio_range:
      - 0.
      - 0.12
      num_time_mask: 5

decoder: whisper
decoder_conf:
    whisper_model: medium
    dropout_rate: 0.0

preprocessor: default
preprocessor_conf:
    whisper_language: "zh"
    whisper_task: "transcribe"

model_conf:
    ctc_weight: 0.0
    lsm_weight: 0.1
    length_normalized_loss: false
    extract_feats_in_collect_stats: false
    sym_sos: "<|startoftranscript|>"
    sym_eos: "<|endoftext|>"
    # do_pad_trim: true # should be set when doing zero-shot inference

frontend: null
input_size: 1             # to prevent build_model() from complaining

seed: 2022
log_interval: 100
num_att_plot: 0
num_workers: 4
sort_in_batch: descending     # how to sort data in making batch
sort_batch: descending        # how to sort created batches
batch_type: numel
batch_bins: 12000000          # good for 8 * RTX 3090 24G
accum_grad: 4
max_epoch: 3
patience: none
init: none
best_model_criterion:
-   - valid
    - acc
    - max
keep_nbest_models: 3

use_amp: true
cudnn_deterministic: false
cudnn_benchmark: false
```

Adapter-Transformers

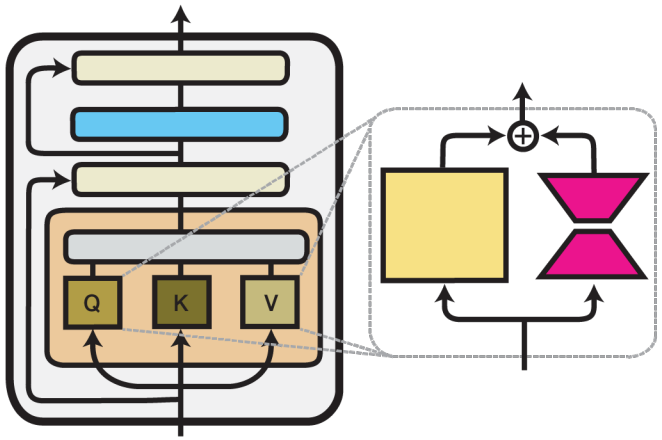


(a) Series Adapter (b) Parallel Adapter (c) LoRA

Figure 1: A detailed illustration of the model architectures of three different adapters: (a) Series Adapter (Houlsby et al., 2019), (b) Parallel Adapter (He et al., 2021), and (c) LoRA (Hu et al., 2021).

egs2/aishell/asr1/conf/tuning/train_asr_whisper_medium_lora_finetune.yaml

```
normalize: null

encoder: whisper
encoder_conf:
    whisper_model: medium
    dropout_rate: 0.0
    use_specaug: true
    specaug_conf:
        apply_time_warp: true
        time_warp_window: 5
        time_warp_mode: bicubic
        apply_freq_mask: true
        freq_mask_width_range:
        - 0
        - 40
        num_freq_mask: 2
        apply_time_mask: true
        time_mask_width_ratio_range:
        - 0.
        - 0.12
        num_time_mask: 5

decoder: whisper
decoder_conf:
    whisper_model: medium
    dropout_rate: 0.0

preprocessor: default
preprocessor_conf:
    whisper_language: "zh"
    whisper_task: "transcribe"

model_conf:
    ctc_weight: 0.0
    lsm_weight: 0.1
    length_normalized_loss: false
    extract_feats_in_collect_stats: false
    sym_sos: "<|startoftranscript|>"
    sym_eos: "<|endoftext|>"
    # do_pad_trim: true
```




```
frontend: null
input_size: 1             # to prevent build_model() from complaining

seed: 2022
log_interval: 100
num_att_plot: 0
num_workers: 4
sort_in_batch: descending    # how to sort data in making batch
sort_batch: descending       # how to sort created batches
batch_type: numel
batch_bins: 70000000         # good for 8 * RTX 3090 24G
accum_grad: 2
max_epoch: 10
patience: none
init: none
best_model_criterion:
-   - valid
    - acc
    - max
keep_nbest_models: 5

use_amp: true
cudnn_deterministic: false
cudnn_benchmark: false

# LoRA finetune related
use_lora: true
lora_conf:
    rank: 8
    alpha: 16
    dropout_rate: 0.05
    target_modules: ["query", "key", "value", "attn.out"]

optim: adamw
grad_clip: 1.0
optim_conf:
    lr: 5.0e-04
    weight_decay: 0.01
    betas:
    - 0.9
    - 0.99
    eps: 1.0e-06
scheduler: warmuplr
scheduler_conf:
    warmup_steps: 1500
```

Fine-Tune Whisper using Hugging Face Transformers

- Blog
  - https://huggingface.co/blog/fine-tune-whisper
  - https://huggingface.co/learn/audio-course/chapter5/fine-tuning
- Colab
  - https://colab.research.google.com/github/sanchit-gandhi/notebooks/blob/main/fine_tune_whisper.ipynb
- GitHub
  - https://github.com/yeyupiaoling/Whisper-Finetune
  - https://github.com/vasistalodagala/whisper-finetune
  - https://github.com/yfliao/whisper-hakka

Make a new recipe

- Create a new directory
- Download data
- Finish the script for data preparation
- Create a script as the entry point
- Create the training config
- Execute the script
- Print the results

```
%cd /content/espnet
!egs2/TEMPLATE/asr1/setup.sh egs2/tidigits/asr1
%cd egs2/tidigits/asr1
!ls
```

```
db.sh
```

```
local/data.sh ← local/data_prep.py
```

```
run.sh
```

```
conf/train_asr_demo_branchformer.yaml
```