Современный DL в обработке аудио: Wav2Vec2.0

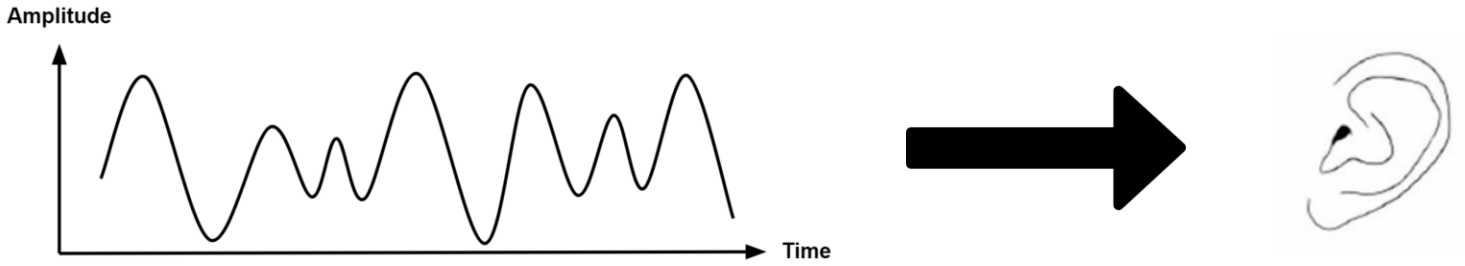
Федоров Илья, 517 группа

План доклада

- Ликбез по работе со звуком
- Классические подходы в DL для классификации аудио
- Архитектура Wav2Vec2.0 (лето 2020, FAIR)

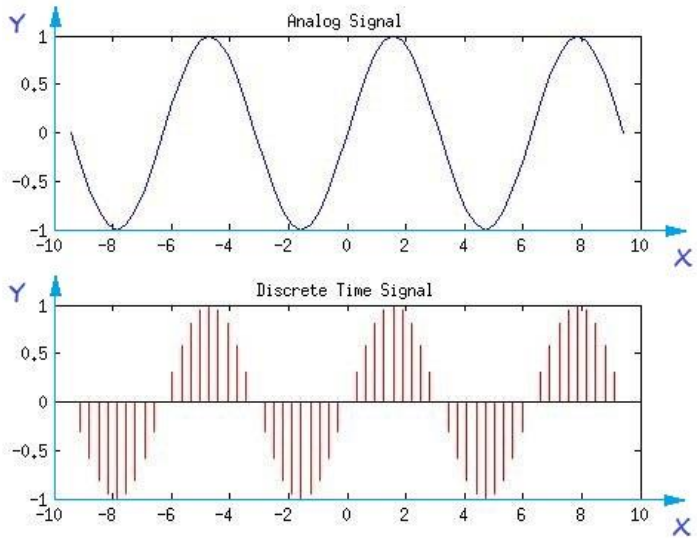
Ликбез по работе с аудио

Звук - возмущения воздушной среды (волны)



Аналоговый сигнал (источник изображения)

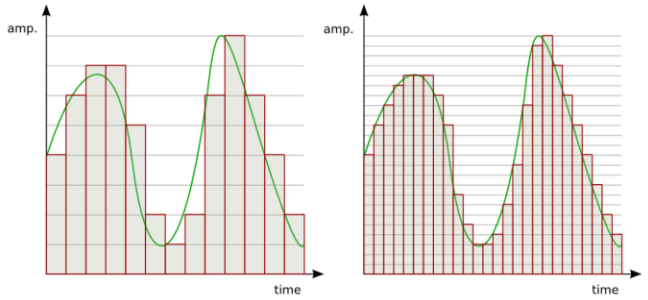
Для передачи/обработки сигнала компьютером, его необходимо дискретизировать:



Дискретизация сигнала (источник)

Ликбез по работе с аудио

Основной параметр - частота дискретизации (sampling rate, sr)



источник

Для работы с человеческим голосом обычно используется 16000 Гц

Чем выше sr - тем "чище" звук. Аналог из CV - разрешение изображения

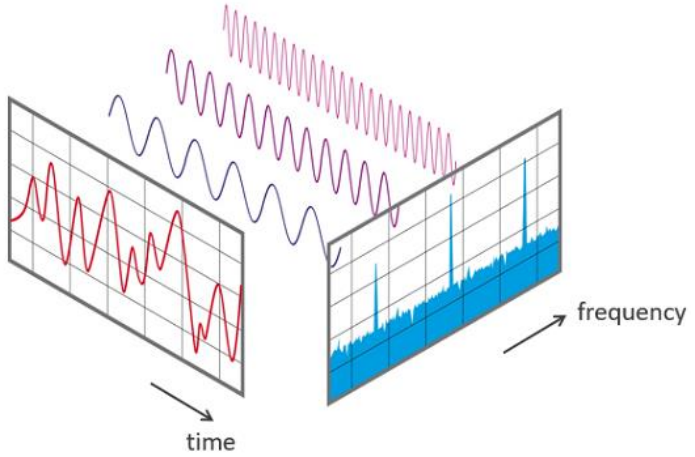

High Resolution Image


Low Resolution Image

источник

Классические подходы в DL для классификации аудио

Стандартный подход для извлечения фичей из аудио - преобразование Фурье

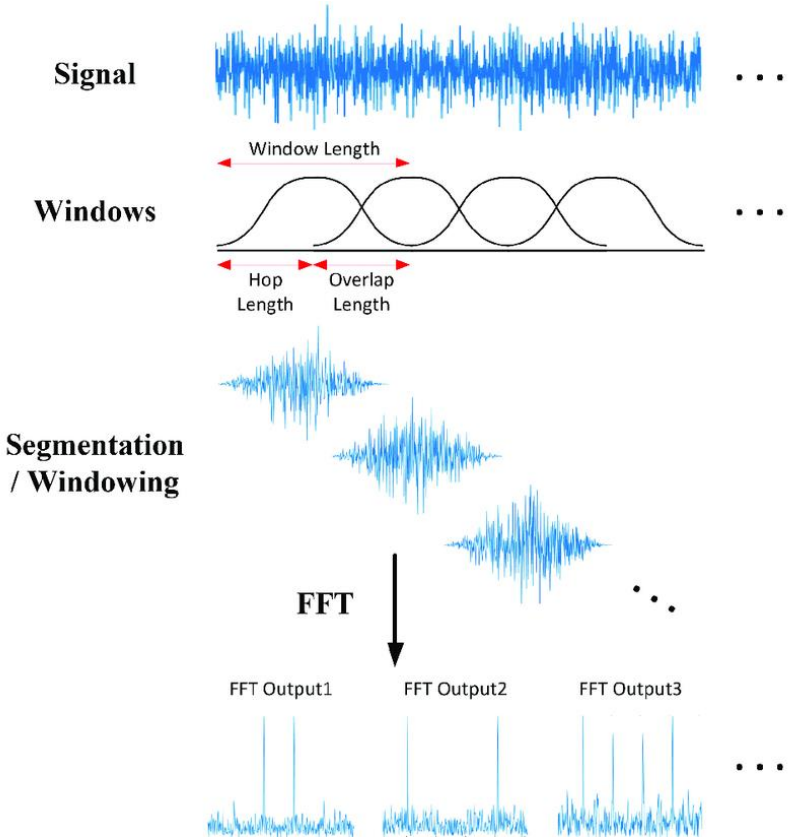


Time Domain and Frequency Domain (источник)

Проблема - "реальный" звук меняет свои характеристики с течением времени

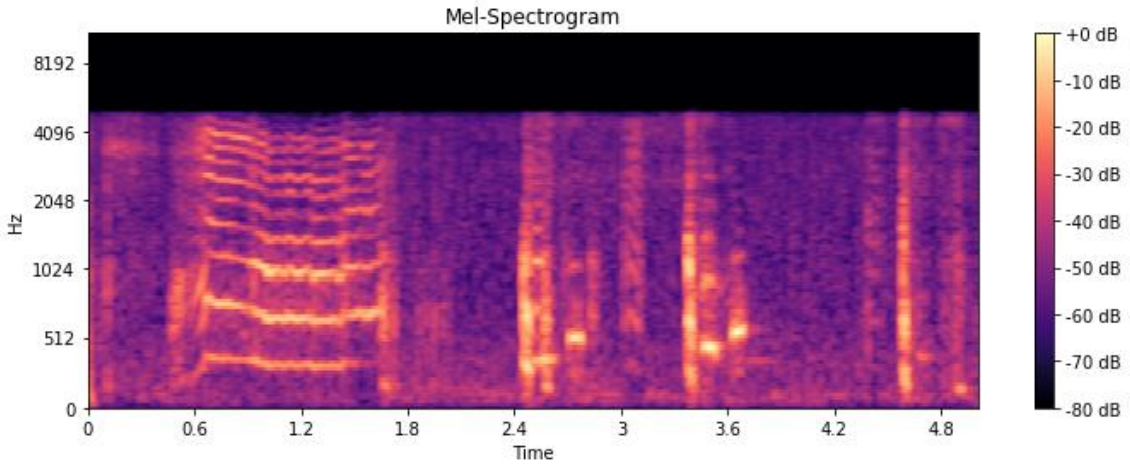
Классические подходы в DL для классификации аудио

Решение - применяем преобразование Фурье скользящим окном, стакаем получаемые векторы коэффициентов Фурье по второй оси



Short Time Fourier Transform (STFM), [источник]

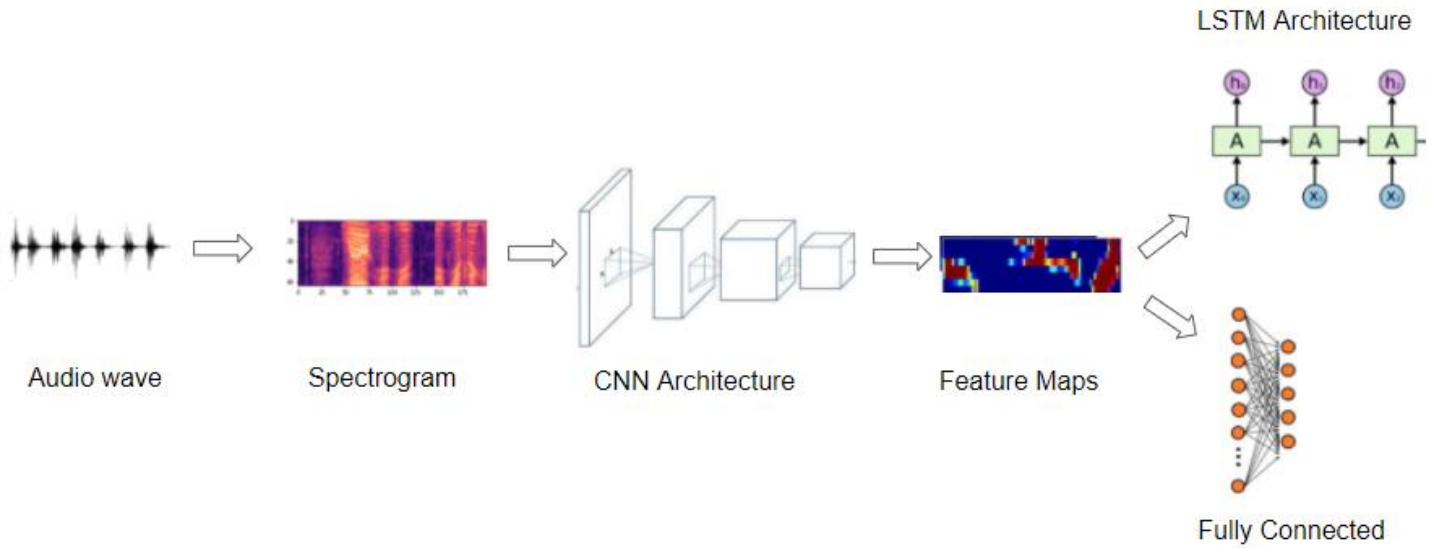
Классические подходы в DL для классификации аудио



(Мел) спектрограмма, [источник]

Классические подходы в DL для классификации аудио

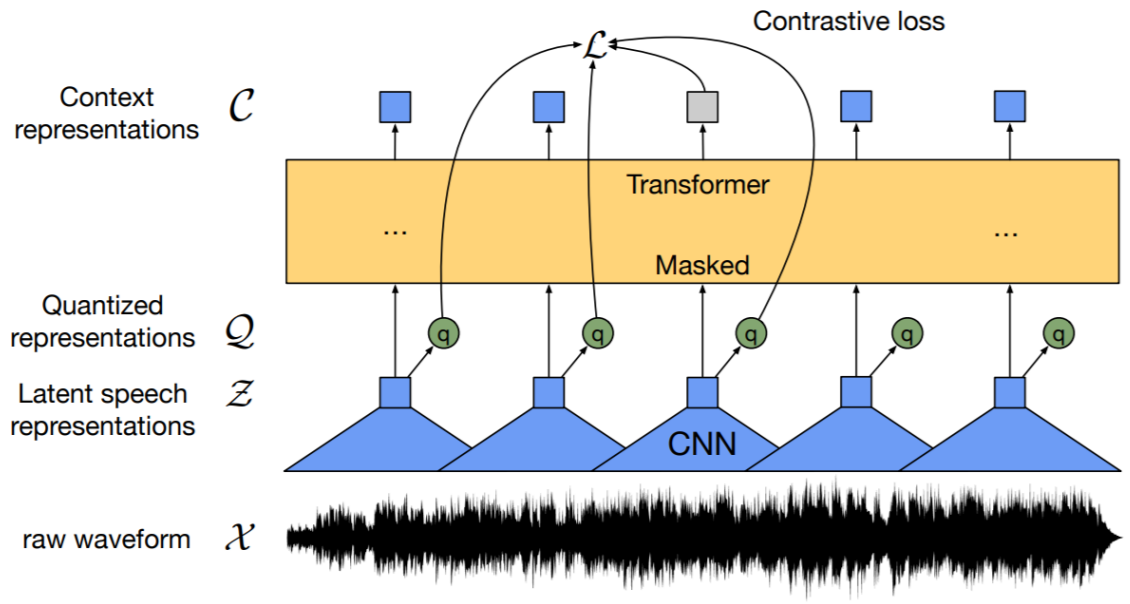
Мы перешли от сигналов к 2d массивам (почти изображениям), следовательно мы можем применять обычные сверточные нейросети для выделения фичей и классификации (ResNet, EfficientNet, ViT и прочее)



Вопрос - какое основное отличие спектрограмм от изображений?

Архитектура Wav2Vec2.0

Основная идея - использование self-supervised подхода применительно к аудио



Архитектура довольно навороченная, нужно прояснить несколько идей

Архитектура Wav2Vec2.0

Напоминание из курса нейробайеса: Gumbel Softmax

Хотим использовать дискретные случайные величины внутри нейросети =>
они должны пропускать градиенты =>
нужен reparametrization trick

В случае популярных непрерывных распределений часто делается довольно легко:

$$\mathcal{N}(x \mid \mu, \sigma^2) \Longleftrightarrow x = y\sigma + \mu, \quad y \sim \mathcal{N}(y \mid 0, 1)$$

Рассмотим дискретное распределение:

$$P(x = i) = \pi_i, \quad i = 1 \dots n$$

Стандартная схема сэмплирования из дискретных распределений:

$$x = \max(k \mid \pi_1 + \pi_2 + \cdots + \pi_{k-1} < y), \quad y \sim \mathrm{U}[0, 1]$$

Как видим, такая схема не дает возможности репараметризации

Архитектура Wav2Vec2.0

Напоминание из курса нейробайеса: Gumbel Softmax

Оказывается, сэмплировать из дискретного распределения можно и по-другому:

$$x \sim \mathrm{Cat}(\pi_1, \ldots, \pi_n) \Longleftrightarrow x = \mathrm{OneHot}(\arg\max_i(G_i + \log \pi_i)), \quad G_i \sim \mathrm{Gumbel}(0, 1)$$

Остается лишь одна проблема: argmax - не дифференцируемая операция. Заменим её на softmax:

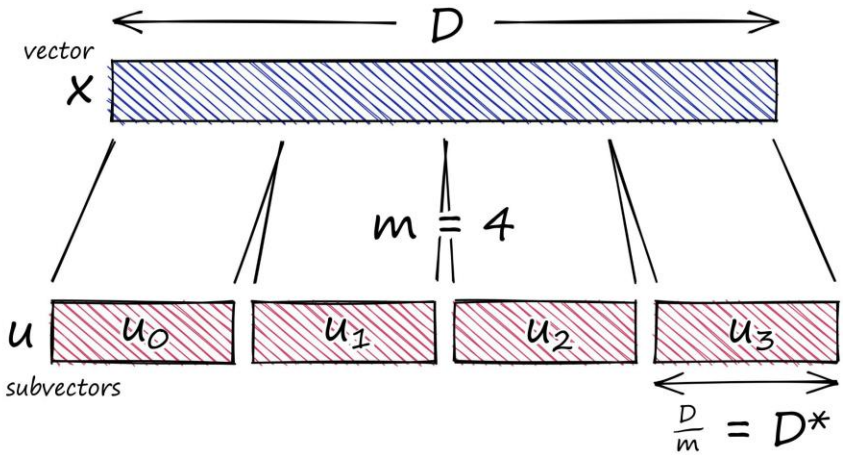
$$x = \mathrm{softmax}(G_i + \log \pi_i), \quad G_i \sim \mathrm{Gumbel}(0, 1)$$

На этапе обучения будем использовать softmax, а во время инференса - честный argmax.

Зачем это все было нужно? Чтобы получить возможность пропускать градиенты в π_i

Архитектура Wav2Vec2.0

Product Quantization

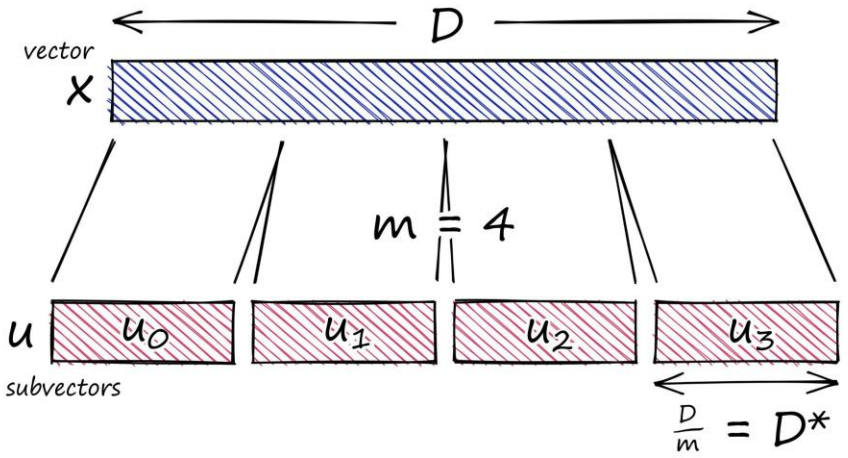


[источник]

Часто применяется для приближенного поиска ближайших соседей (к примеру в FAISS)

Архитектура Wav2Vec2.0

Product Quantization



Каждый из u_i берется из "таблицы" (codebook) векторов.
Всего имеется G таблиц, каждая из которых состоит из V векторов.

Выбор происходит случайным образом, но в соответствии с распределением, закрепленным за каждой из таблиц (для этого и нужен был Gumbel Softmax)

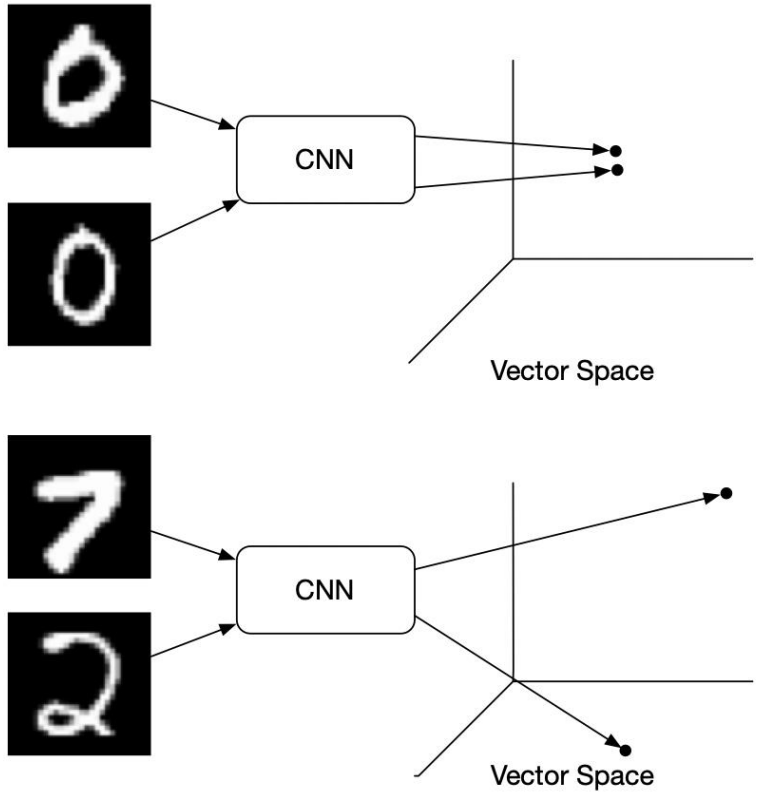
Таким образом, большой исходный вектор v заменяется конкатенаций из G небольших векторов, каждый из которых был взят из соответствующей таблицы.

Таблицы являются обучаемыми параметрами.

Архитектура Wav2Vec2.0

Contrastive loss

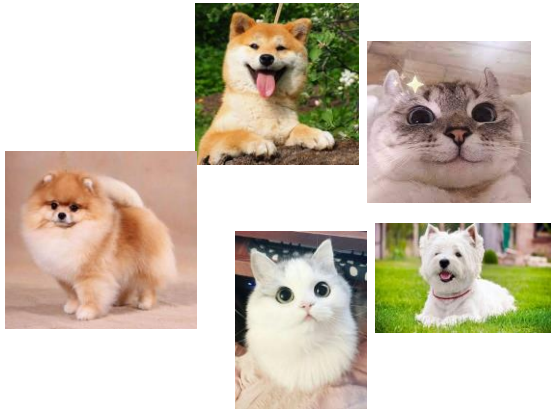
Классический (наряду с Triplet Loss) лосс из области metric learning



Архитектура Wav2Vec2.0

Contrastive loss

Пусть имеется фиксированный объект x, а также набор других объектов, среди которых есть один похожий на x, а все остальные - не похожие на x.

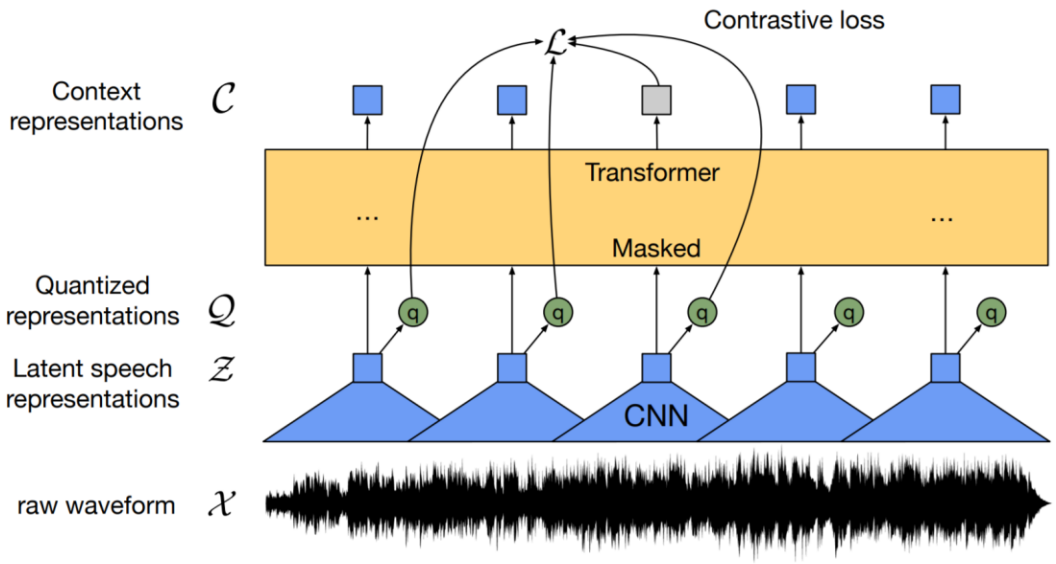


- Прогоняем все объекты через нейросеть
- Получаем эмбеддинги для каждого объекты
- Измеряем расстояния между x и всем и остальными объектами
- Применяем softmax к полученному вектору расстояний
- Требуем, чтобы похожие объекты давали единицу

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N}{}_{[k\neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)},$$

источник

Архитектура Wav2Vec2.0



- По входному сигналу с некоторым шагом проходится сверточный энкодер.
- Все представления z квантуются с помощью Product Quantization + Gumbel Softmax
- Часть представлений z маскируется (единым обучаемым токеном) и прогоняются через трансформер
- Для каждого замаскированного представления считается contrastive loss: требуется найти среди всех квантованных представлений замаскированных объектов то, которое соответствовало ему до прогонки через трансформер

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

Архитектура Wav2Vec2.0

После предобучения нейросеть использовалась для решения задачи ASR с использованием CTC лосса.
В оригинале использовался датасет Librispeech. WER – World Error Rate, ниже - лучше.

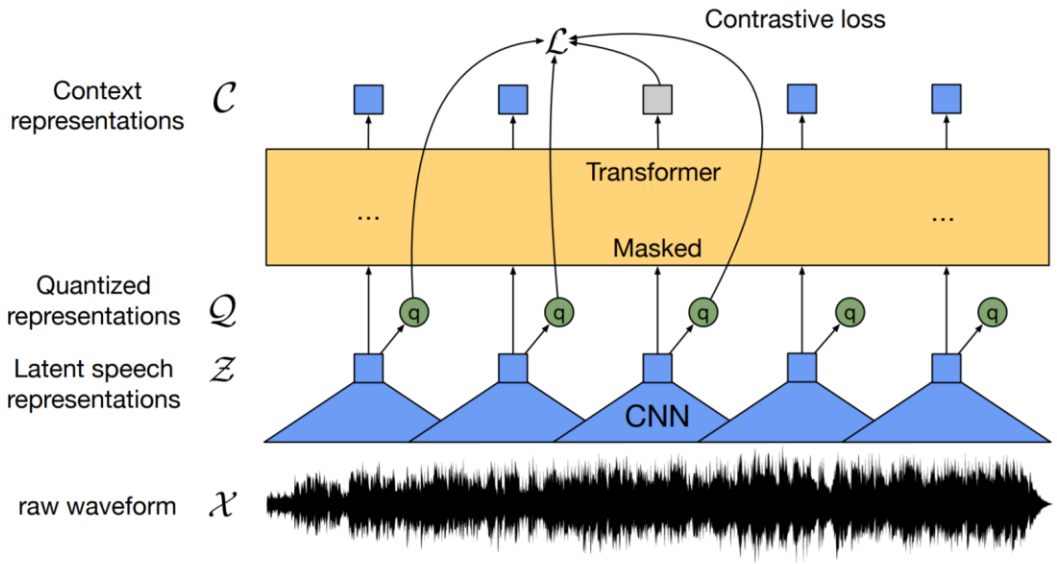
Table 1: WER on the Librispeech dev/test sets when training on the Libri-light low-resource labeled data setups of 10 min, 1 hour, 10 hours and the clean 100h subset of Librispeech. Models use either the audio of Librispeech (LS-960) or the larger LibriVox (LV-60k) as unlabeled data. We consider two model sizes: BASE (95m parameters) and LARGE (317m parameters). Prior work used 860 unlabeled hours (LS-860) but the total with labeled data is 960 hours and comparable to our setup.

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.	4.6	7.9	4.8	8.2
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.	2.9	5.4	2.9	5.8
10h labeled						
Discrete BERT [4]	LS-960	4-gram	5.3	13.2	5.9	14.1
Iter. pseudo-labeling [58]	LS-960	4-gram+Transf.	23.51	25.48	24.37	26.02
	LV-60k	4-gram+Transf.	17.00	19.34	18.03	19.92
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.	2.4	4.8	2.6	4.9
100h labeled						
Hybrid DNN/HMM [34]	-	4-gram	5.0	19.5	5.8	18.6
TTS data augm. [30]	-	LSTM			4.3	13.5
Discrete BERT [4]	LS-960	4-gram	4.0	10.9	4.5	12.1
Iter. pseudo-labeling [58]	LS-860	4-gram+Transf.	4.98	7.97	5.59	8.95
	LV-60k	4-gram+Transf.	3.19	6.14	3.72	7.11
Noisy student [42]	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.	1.9	4.0	2.0	4.0

Архитектура Wav2Vec2.0

Можно использовать для классификации, сделав пуллинг выходных представлений (к примеру, усреднение)

Затем применить стандартный classifier head:
FC + Activation -> Dropout -> FC + Activation -> softmax



Спасибо за внимание!