Spec Augmentation

「SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition」 논문

Winter Vacation Capstone Study

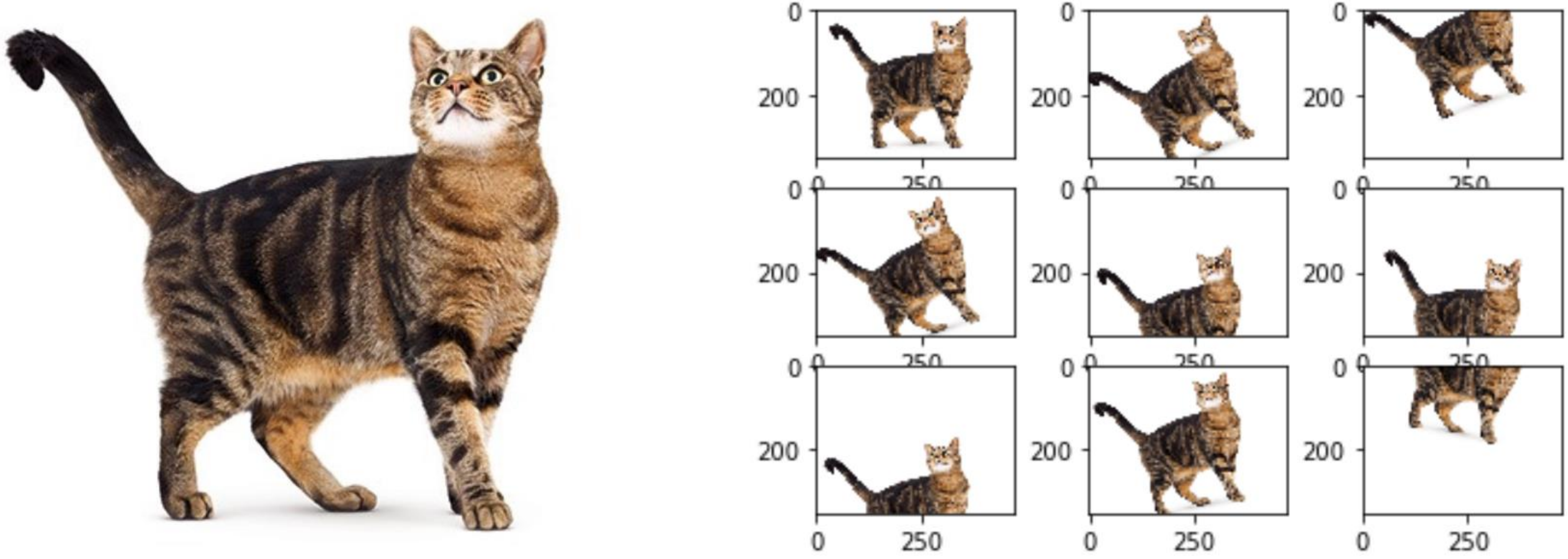
TEAM Kai.Lib

발표자 : 배세영

2020.01.13 (MON)

Spec Augmentation이란

- **Augmentation (증강)**



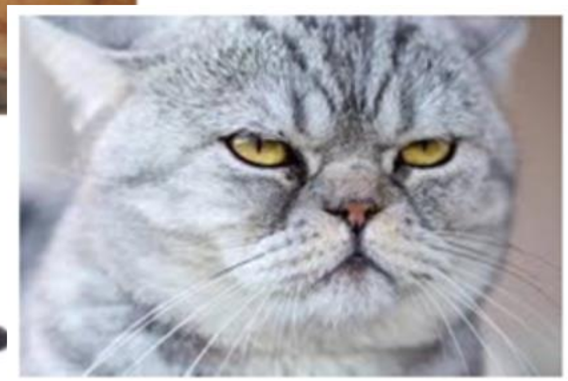
Original Image and Augmented Images

Spec Augmentation이란

- **Augmentation (증강)**

  데이터를 부풀려서 모델의 성능을 향상시키는 기법
  이미지 인식 분야에서 활용하던 방식 (좌우 반전, 일부 발췌, 밝기 조절 등)
  한정된 데이터를 조금씩 변형시켜 새로운 데이터처럼 활용하는 것으로, Data Augmentation이라고도 한다






Augmentation을 하는 중요한 이유

- Preprocessing 및 Augmentation을 하면, 대부분의 경우 성능이 향상된다.
- 원본 데이터를 활용하여 추가하는 개념이므로 성능이 저하될 염려가 없다.
- 방법이 간단하며 패턴이 정해져 있다.

"단기간에 성능 향상을 원한다면, Transfer Learning / Augmentation을 활용하라."

AlexNet의 Augmentation

- 좌우 반전
- 224*224px -> 256*256px -> 224*224px 영역으로 random하게 2048번 잘라 학습 (Training)
- 224*224px -> 256*256px -> 좌상/좌하/중앙/우상/우하 영역 잘라 5배 -> 좌우 반전 통해 10배 늘려 각각의 결과값을 예측한 후 평균을 내어 최종 결정 (Predict)

VGGNet의 Augmentation

- **이미지 데이터에서 가장 많이 활용하는 Preprocessing / Augmentation 기법이 적용된 모델**

- **Preprocessing**
  - **RGB값의 평균을 빼주어 평균을 0으로 만든다**

    성능 개선보다는 학습 속도를 빠르게 하는 효과가 있다.

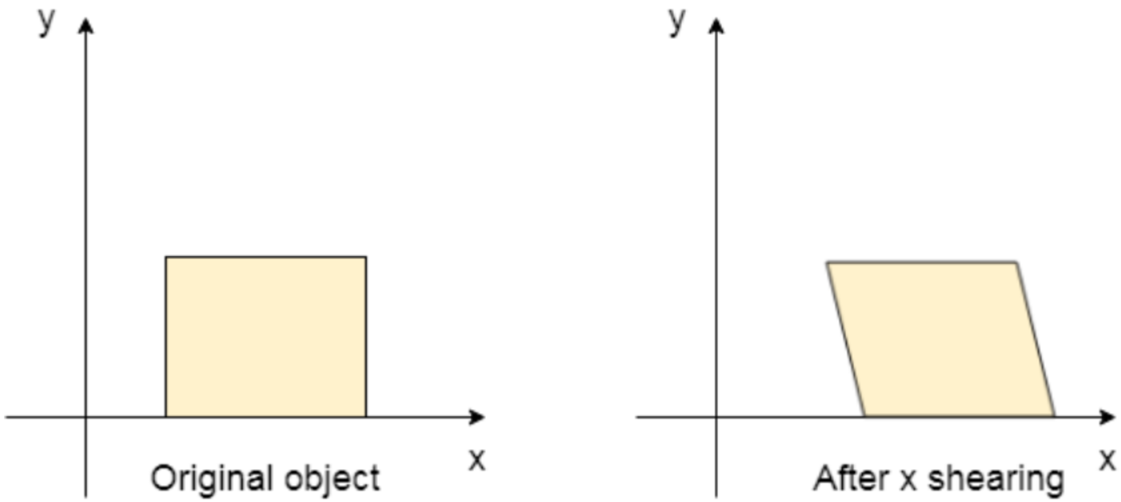
- **Augmentation**
  - **이미지를 256*256px, 384*384px, 512*512px의 3가지 버전으로 만든 후 224*224로 랜덤 crop**

    256*256에서 잘라낸 이미지에는 전체 이미지가 거의 다 들어가겠지만, 384*384, 512*512에서 잘라낸 이미지는 원본 이미지의 일부만이 발췌되어 들어가 있음. 일부만 보고 전체를 인식하도록 학습하는 효과

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**

기타 Image Augmentation 방식

- Rotation
  - 0'~360' 임의로 회전
- Shifting
  - 임의의 방향으로 정해진 px씩 사진을 이동
- Rescaling
  - 이미지의 크기를 줄이거나 키움 (VGGNet)
- Flipping
  - 상하/좌우 반전
- Shearing
  - 이미지를 찌그러뜨림
- Stretching
  - 이미지를 잡아 늘림



Shearing

Data Augmentation for Audio (Traditional Way)

- Noise Injection
  - 잡음 첨가

- Shifting Time
  - 시간축 이동

- Changing Pitch
  - 주파수 변조

- Changing Speed
  - 속도 변조

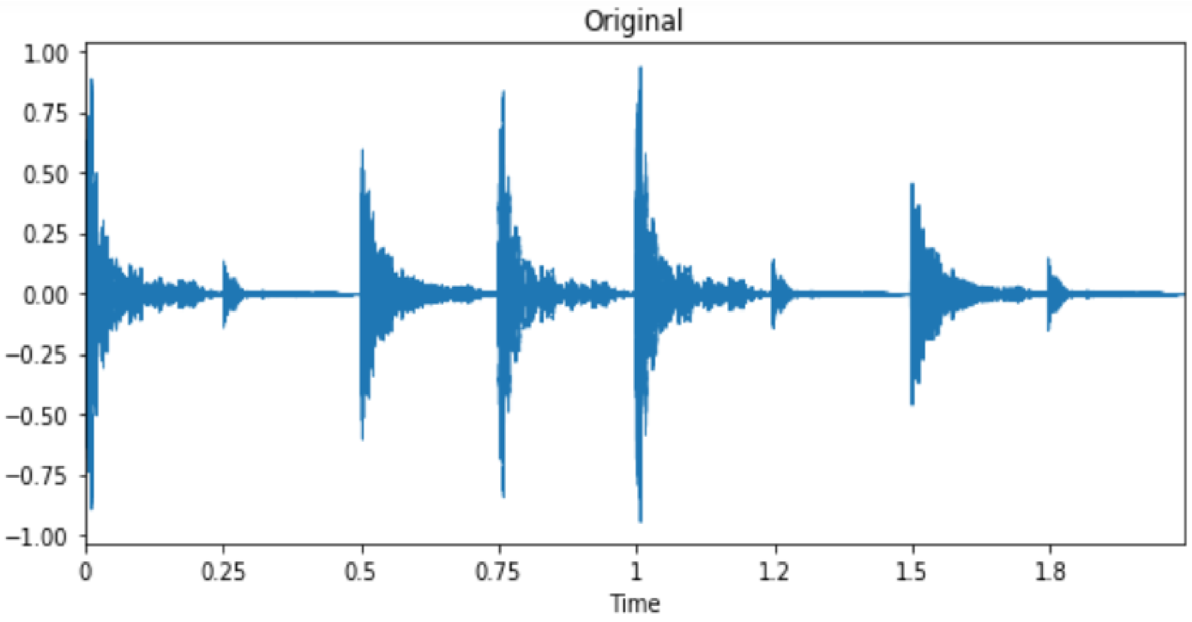
Data Augmentation for Audio (Traditional Way)

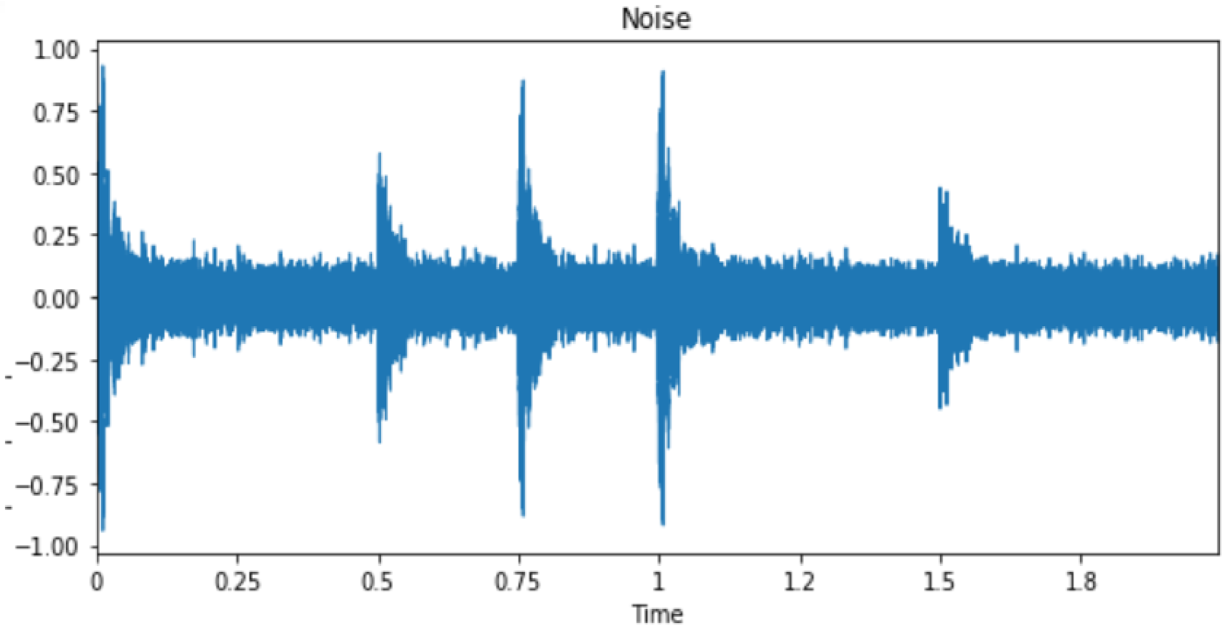
- Noise Injection
  - numpy array 이용. 난수 값을 Data에 더해준다

```
import numpy as np

def manipulate(data, noise_factor):
    noise = np.random.randn(len(data))
    augmented_data = data + noise_factor * noise
    # Cast back to same data type
    augmented_data = augmented_data.astype(type(data[0]))
    return augmented_data
```

Data Augmentation for Audio (Traditional Way)

- **Noise Injection**
  - **numpy array 이용, 난수 값을 Data에 더해준다**





Data Augmentation for Audio (Traditional Way)

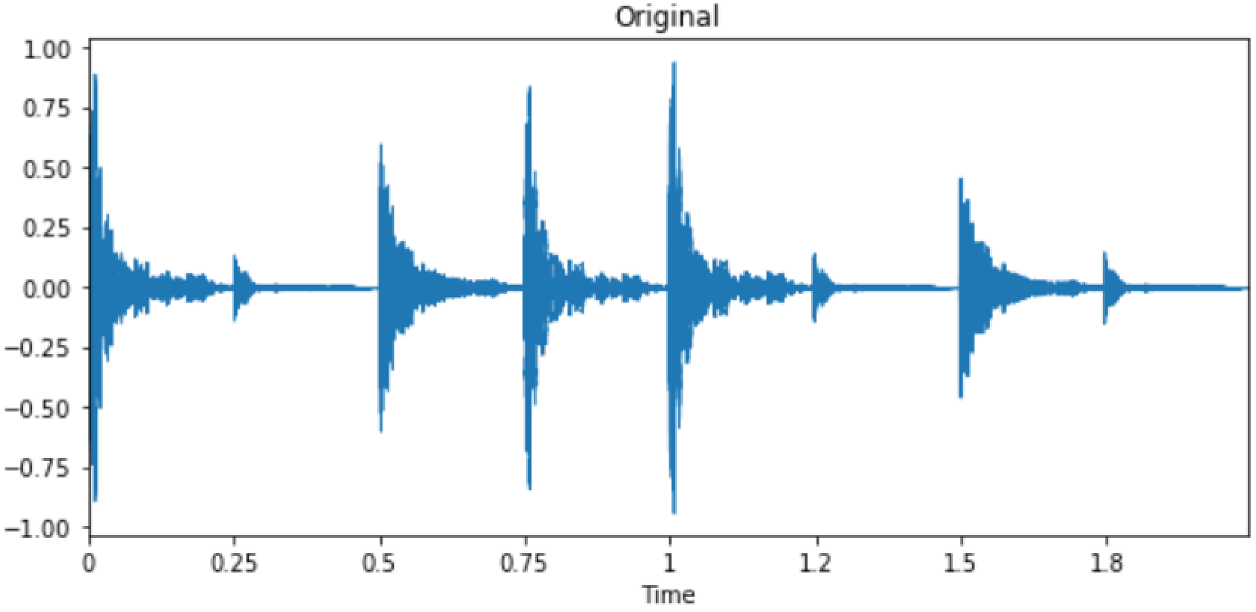
- **Shifting Time**
  - **numpy array 이용. 임의의 값만큼 좌/우로 shift하고 빈 공간은 0으로 채운다**

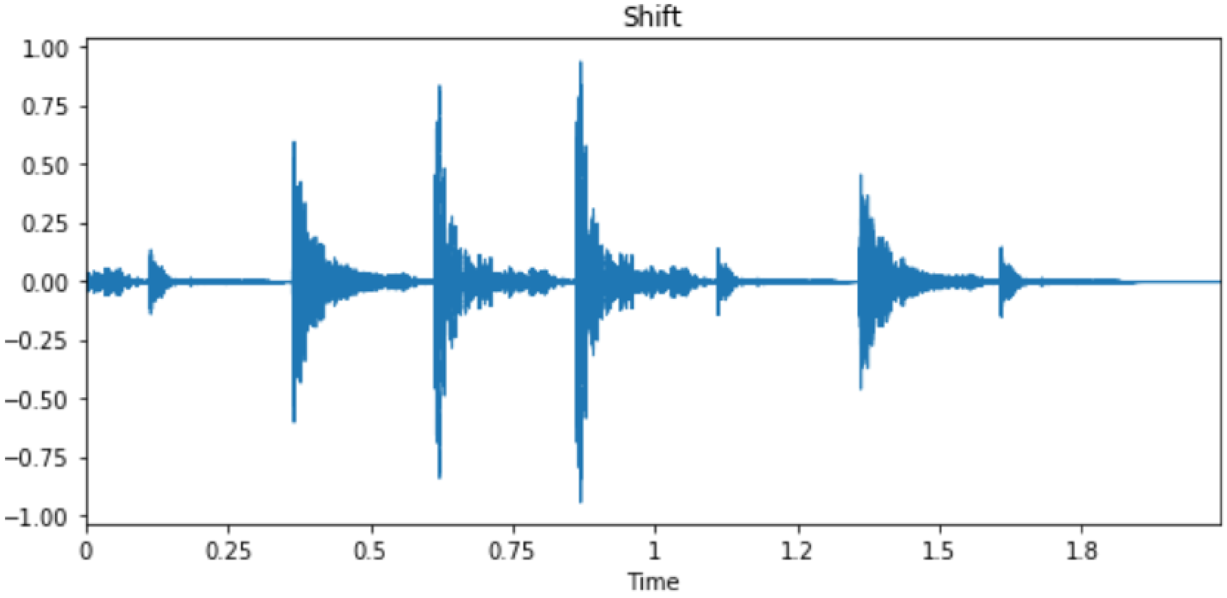
```
import numpy as np

def manipulate(data, sampling_rate, shift_max, shift_direction):
    shift = np.random.randint(sampling_rate * shift_max)
    if shift_direction == 'right':
        shift = -shift
    elif self.shift_direction == 'both':
        direction = np.random.randint(0, 2)
        if direction == 1:
            shift = -shift

    augmented_data = np.roll(data, shift)
    # Set to silence for heading/ tailing
    if shift > 0:
        augmented_data[:shift] = 0
    else:
        augmented_data[shift:] = 0
    return augmented_data
```

```
>>> x = np.arange(10)
>>> np.roll(x, 2)
array([8, 9, 0, 1, 2, 3, 4, 5, 6, 7])
```

Data Augmentation for Audio (Traditional Way)

- Shifting Time
  - numpy array 이용, 임의의 값만큼 좌/우로 shift하고 빈 공간은 0으로 채운다





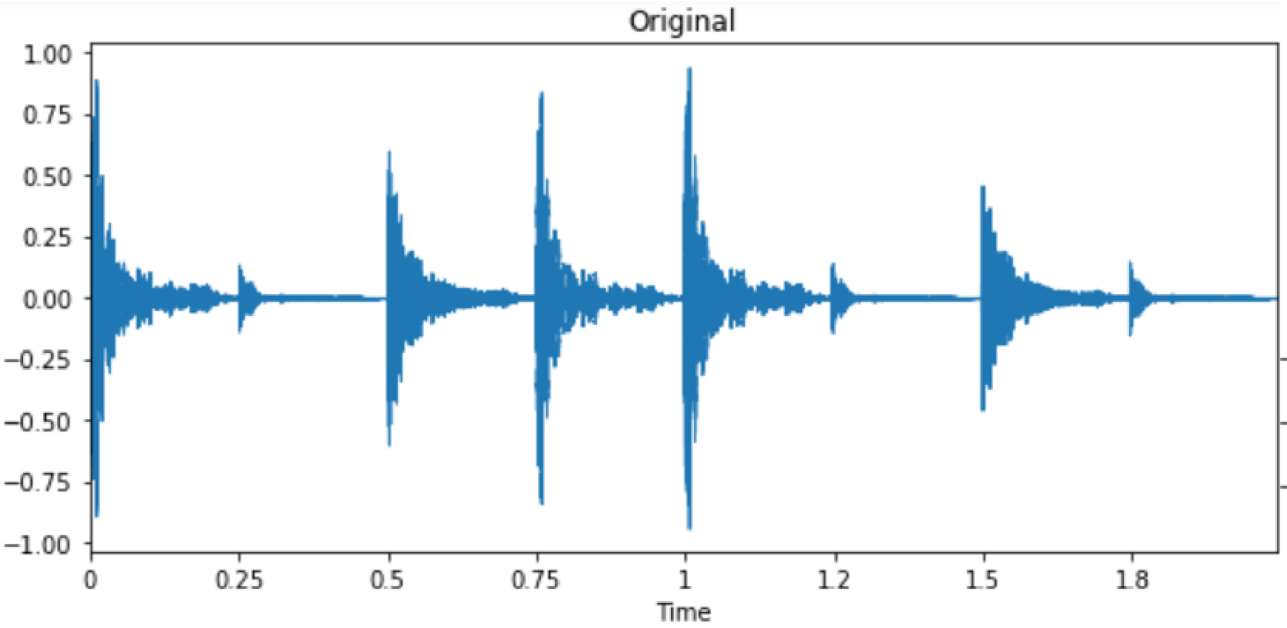
Data Augmentation for Audio (Traditional Way)

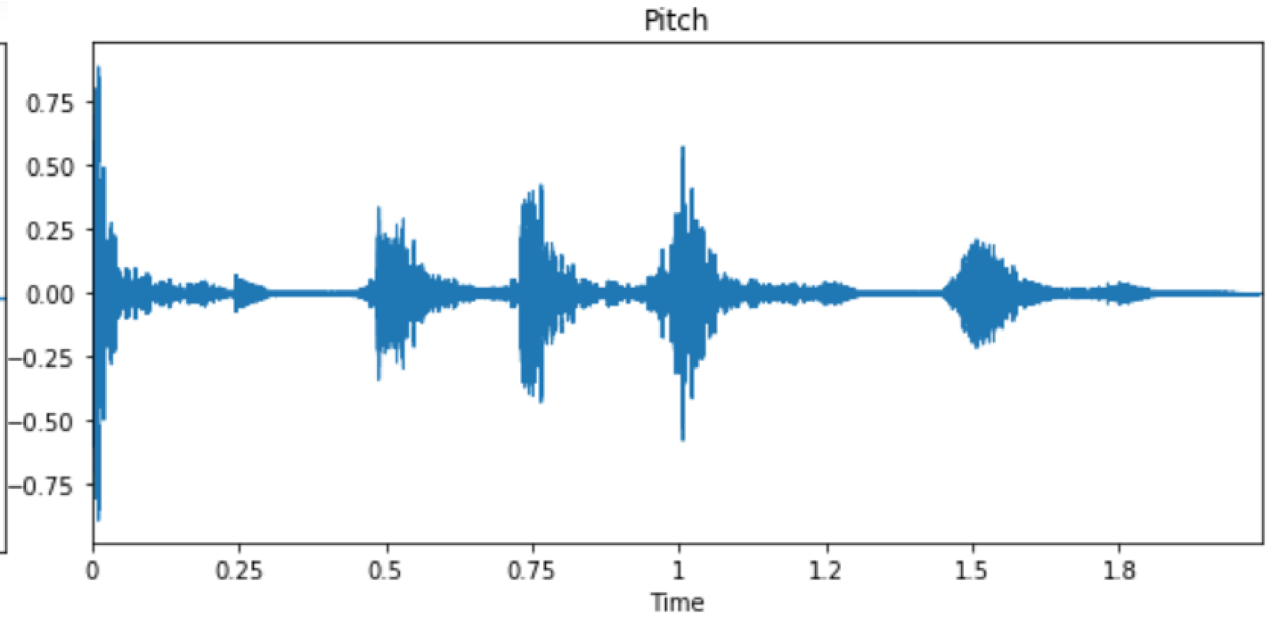
- Changing Pitch
  - librosa 라이브러리에서 제공, Pitch(음높이, 주파수)를 random하게 변경

```
import librosa


def manipulate(data, sampling_rate, pitch_factor):
    return librosa.effects.pitch_shift(data, sampling_rate,
pitch_factor)
```

Data Augmentation for Audio (Traditional Way)

- Changing Pitch
  - librosa 라이브러리에서 제공, Pitch(음높이, 주파수)를 random하게 변경





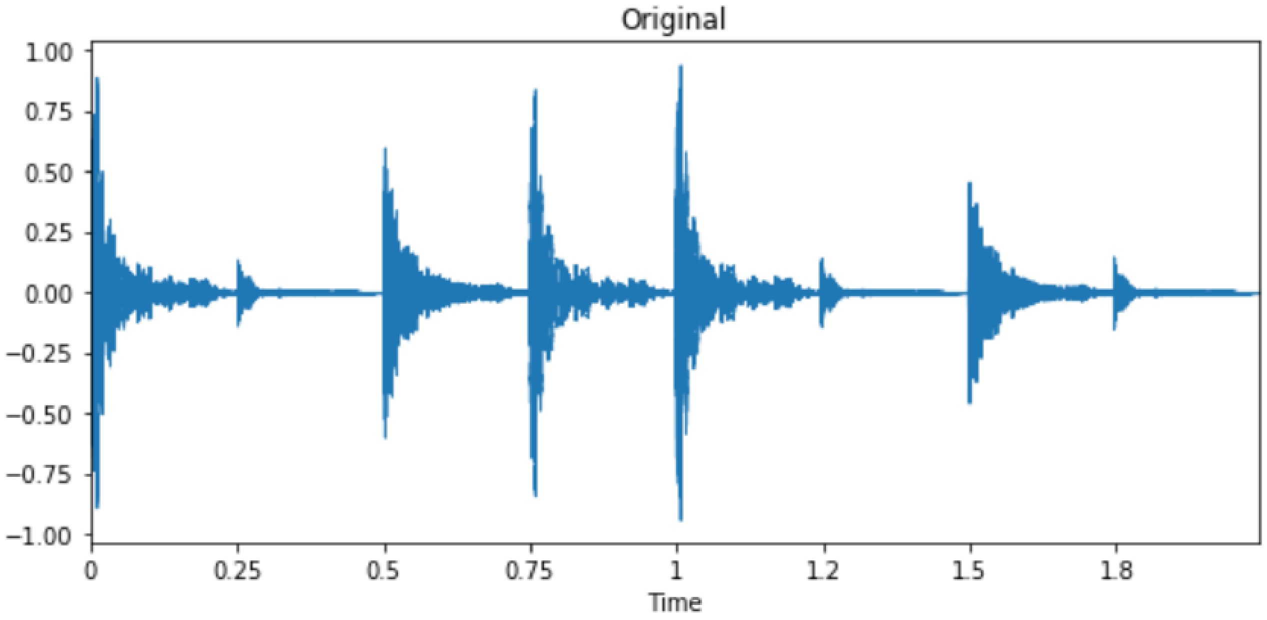
Data Augmentation for Audio (Traditional Way)

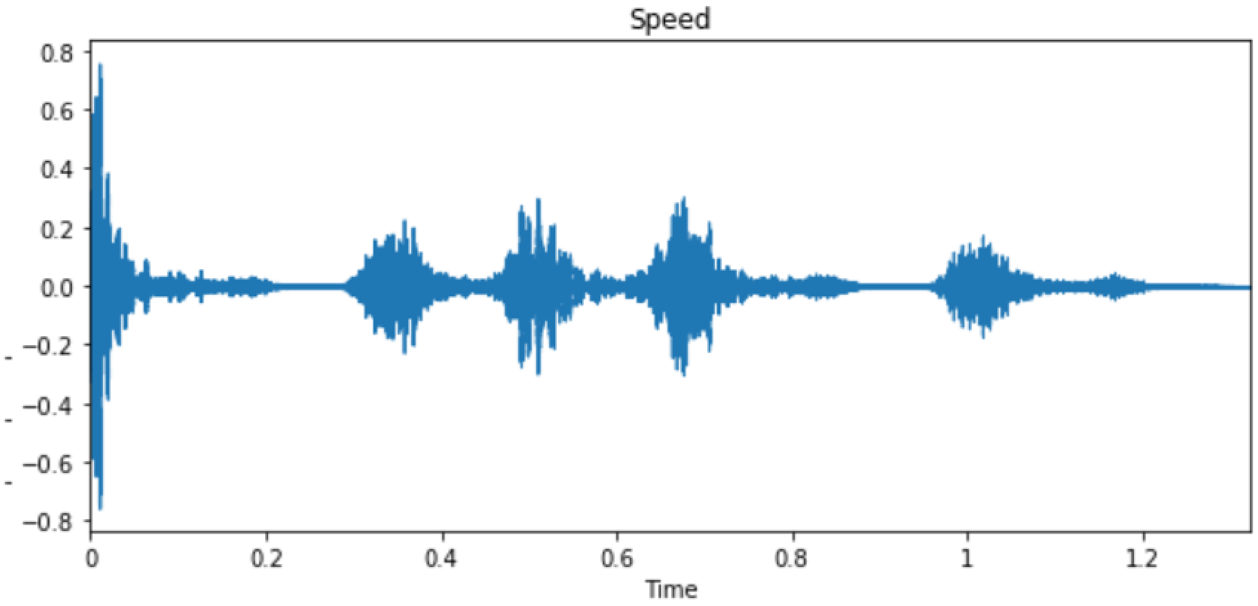
- **Changing Speed**
  - **librosa 라이브러리에서 제공, time series를 고정된 값만큼 stretch**

```
import librosa


def manipulate(data, speed_factor):
    return librosa.effects.time_stretch(data, speed_factor)
```

Data Augmentation for Audio (Traditional Way)

- Changing Speed
  - librosa 라이브러리에서 제공, time series를 고정된 값만큼 stretch





Data Augmentation for Audio (SOTA)

- Google Brain Team에서 2019.12.03 발행한 논문
  - 기존의 waveform 기반 data augmentation 방식을 대체할 세 가지 augmentation 방식 제안
  - 기존 방식에 비해 계산적으로 가벼우며, 추가적인 Data가 필요하지 않음

SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, Quoc V. Le*

Google Brain

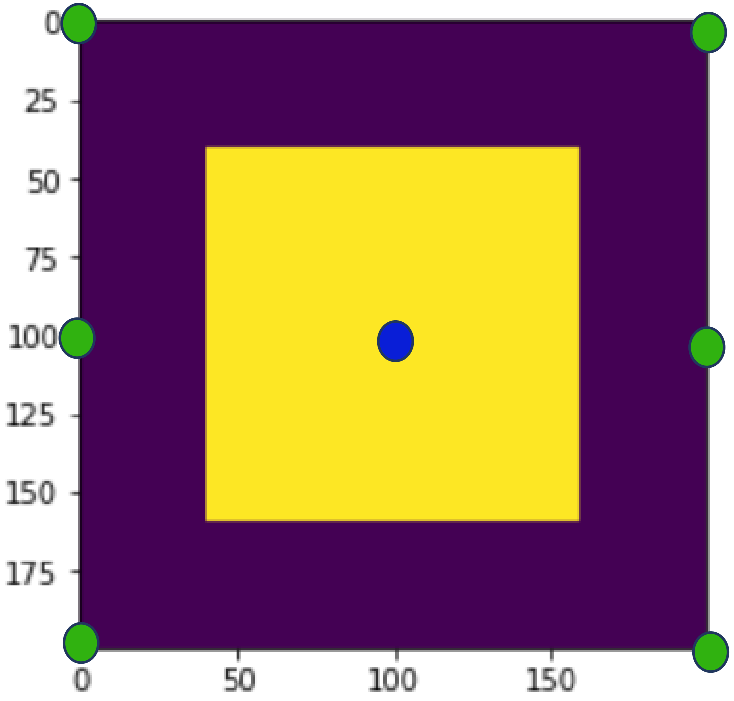
{danielspark, williamchan, ngyuzh, chungchengc, barretzoph, cubuk, qvl}@google.com

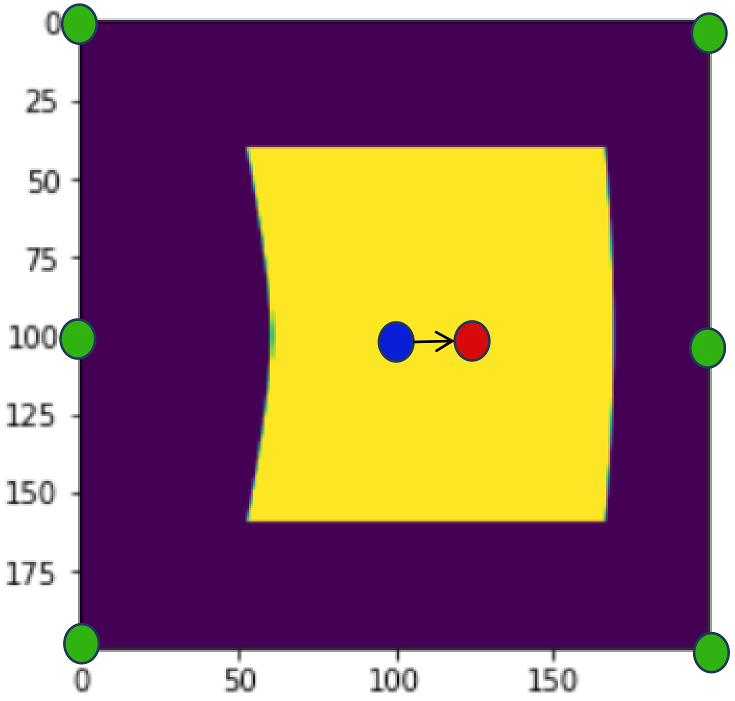
Data Augmentation for Audio (SOTA)

- Spectrogram 기반의 Data Augmentation Methods
  - Time Warping
  - Frequency Masking
  - Time Masking

Data Augmentation for Audio (SOTA)

- Spectrogram 기반의 Data Augmentation Methods
  - Time Warping
    - 이미지에서 사용하는 image warping을 응용 (Spectrogram은 Image처럼 처리 가능하므로)

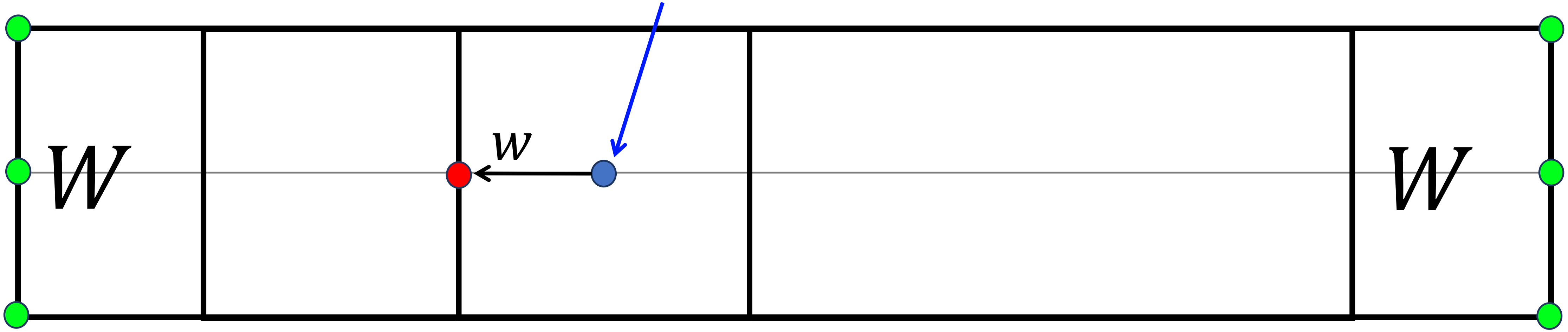




Data Augmentation for Audio (SOTA)

- **Spectrogram 기반의 Data Augmentation Methods**
  - **Time Warping**
    - **이미지에서 사용하는 image warping을 응용 (Spectrogram은 Image처럼 처리 가능하므로)**

Random Point between (W, t-W)



W : Hyper Parameter
w : $0, W$ 사이에서 random하게 뽑은 값

Data Augmentation for Audio (SOTA)

- **Spectrogram 기반의 Data Augmentation Methods**
  - **Time Warping**

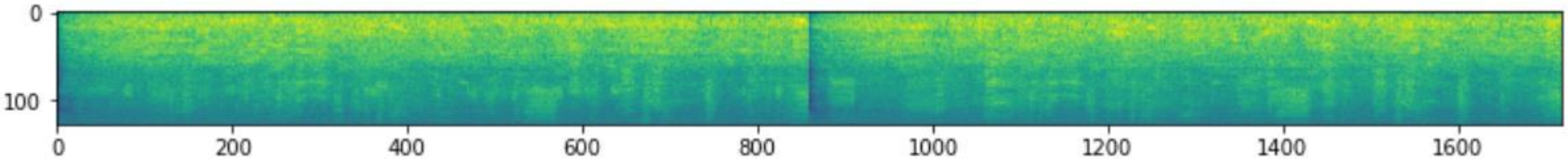
```python
#Export
def time_warp(spec, W=5):
    num_rows = spec.shape[1]
    spec_len = spec.shape[2]
    device = spec.device

    y = num_rows//2
    horizontal_line_at_ctr = spec[0][y]
    assert len(horizontal_line_at_ctr) == spec_len

    point_to_warp = horizontal_line_at_ctr[random.randrange(W, spec_len - W)]
    assert isinstance(point_to_warp, torch.Tensor)

    # Uniform distribution from (0,W) with chance to be up to W negative
    dist_to_warp = random.randrange(-W, W)
    src_pts, dest_pts = (torch.tensor([[[y, point_to_warp]]], device=device),
                         torch.tensor([[[y, point_to_warp + dist_to_warp]]], device=device))
    warped_spectro, dense_flows = sparse_image_warp(spec, src_pts, dest_pts)
    return warped_spectro.squeeze(3)
```

```python
lists = [1, 3, 6, 3, 8, 7, 13, 23, 13, 2, 3.14, 2, 3, 7]

def test(t):
    assert type(t) is int, '정수 아닌 값이 있네'

for i in lists:
    test(i)
#결과
AssertionError: 정수 아닌 값이 있네
```

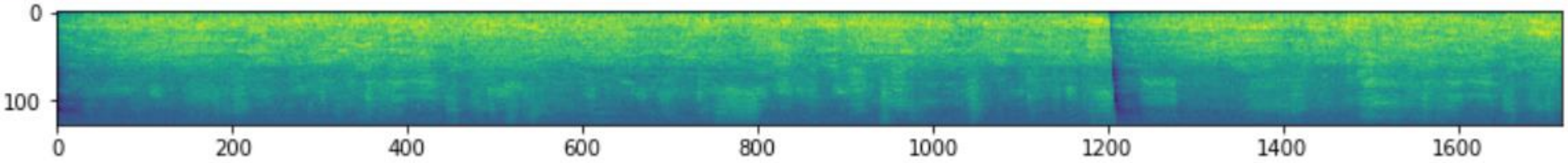
https://github.com/zcaceres/spec_augment

Data Augmentation for Audio (SOTA)

- **Spectrogram 기반의 Data Augmentation Methods**
  - **Time Warping**




```
torch.Size([1, 128, 1718])
/home/jupyter/git/spec_augment/exp/nb_SparseImageWarp.py:300: UserWarning: To copy construct from a tensor, it
is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(True), rath
er than torch.tensor(sourceTensor).
  alpha = torch.tensor((queries - floor), dtype=grid_type, device=grid_device)
```

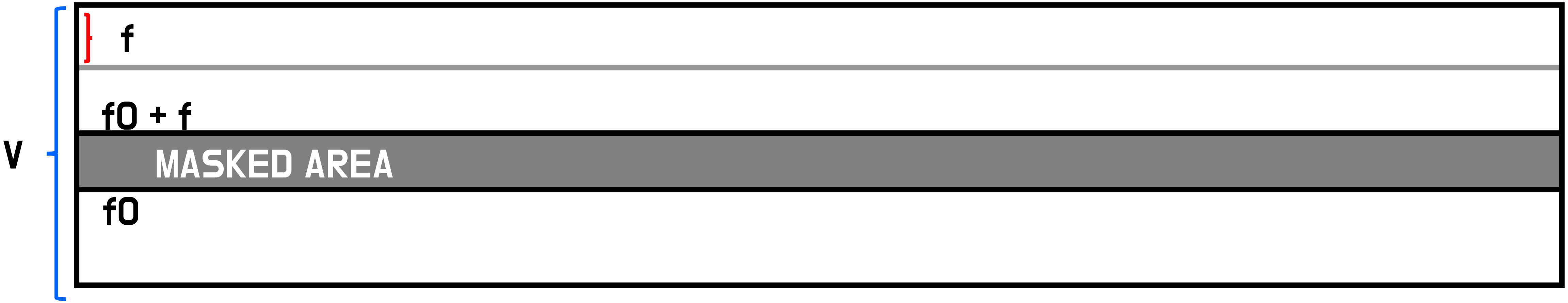



```
torch.Size([1, 128, 1718])
```

https://github.com/zcaceres/spec_augment

Data Augmentation for Audio (SOTA)

- Spectrogram 기반의 Data Augmentation Methods
  - Frequency Masking
    - Spectrogram 상의 Frequency 축을 따라 일정 영역을 0으로 마스킹



F : Hyper Parameter
f : 0, F 사이에서 random하게 뽑은 값
f0 : 0, V-f 사이에서 random하게 뽑은 값

Data Augmentation for Audio (SOTA)

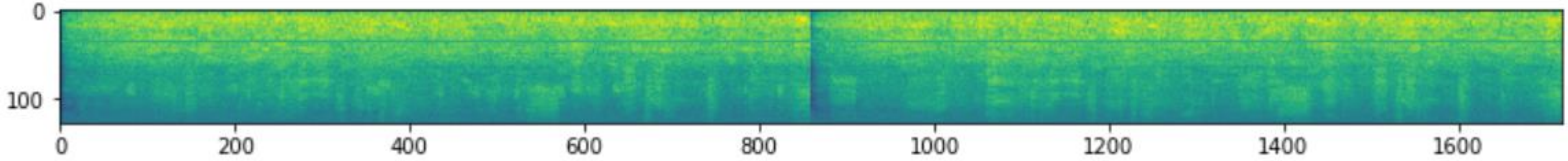
- **Spectrogram 기반의 Data Augmentation Methods**
  - **Frequency Masking**

```python
#Export
def freq_mask(spec, F=30, num_masks=1, replace_with_zero=False):
    cloned = spec.clone()
    num_mel_channels = cloned.shape[1]

    for i in range(0, num_masks):
        f = random.randrange(0, F)
        f_zero = random.randrange(0, num_mel_channels - f)

        # avoids randrange error if values are equal and range is empty
        if (f_zero == f_zero + f): return cloned

        mask_end = random.randrange(f_zero, f_zero + f)
        if (replace_with_zero): cloned[0][f_zero:mask_end] = 0
        else: cloned[0][f_zero:mask_end] = cloned.mean()

    return cloned
```

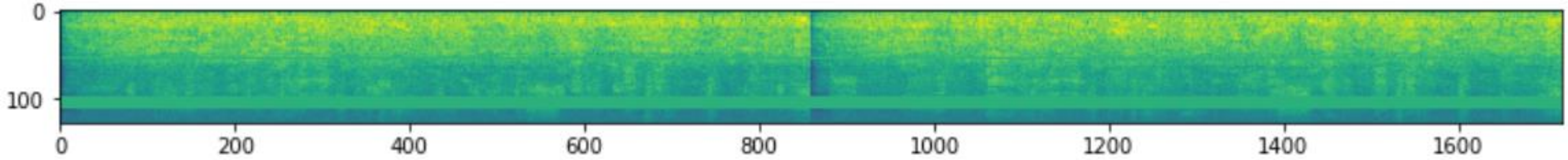
https://github.com/zcaceres/spec_augment

Data Augmentation for Audio (SOTA)

- **Spectrogram 기반의 Data Augmentation Methods**
  - **Frequency Masking**



torch.Size([1, 128, 1718])

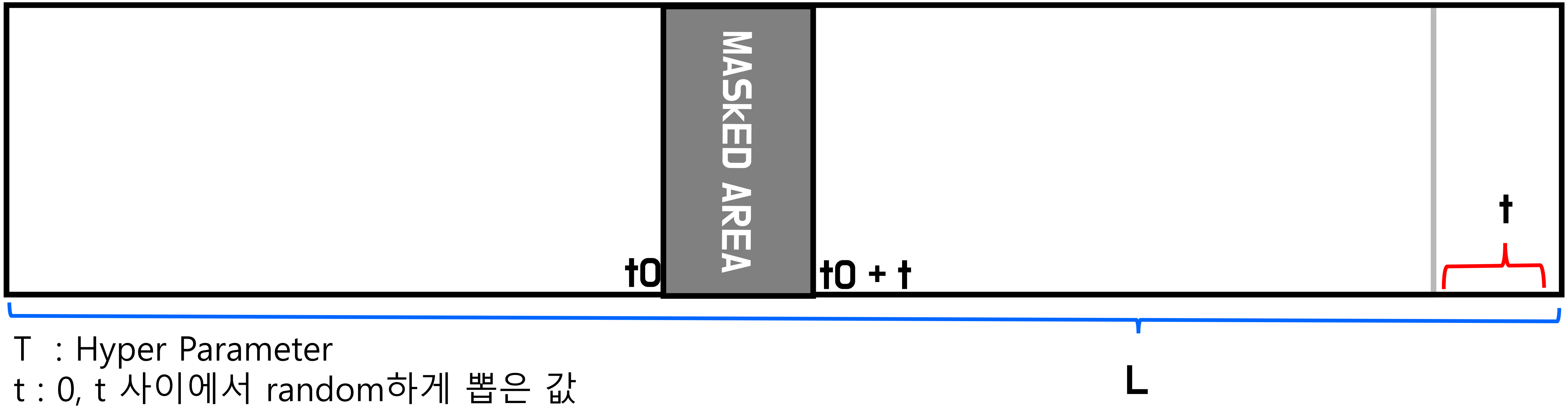


torch.Size([1, 128, 1718])

https://github.com/zcaceres/spec_augment

Data Augmentation for Audio (SOTA)

- Spectrogram 기반의 Data Augmentation Methods
  - Time Masking
    - Spectrogram 상의 Time 축을 따라 일정 영역을 0으로 마스킹



T : Hyper Parameter
t : 0, t 사이에서 random하게 뽑은 값
t0 : 0, L-t 사이에서 random하게 뽑은 값

Data Augmentation for Audio (SOTA)

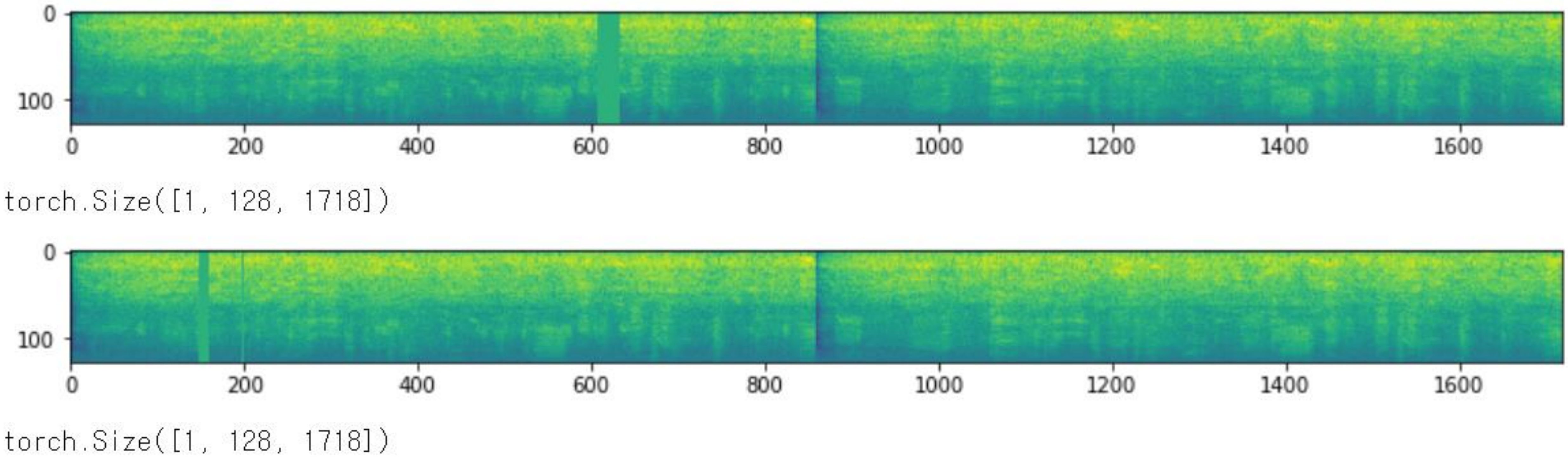
- **Spectrogram 기반의 Data Augmentation Methods**
  - **Time Masking**

```python
#Export
def time_mask(spec, T=40, num_masks=1, replace_with_zero=False):
    cloned = spec.clone()
    len_spectro = cloned.shape[2]

    for i in range(0, num_masks):
        t = random.randrange(0, T)
        t_zero = random.randrange(0, len_spectro - t)

        # avoids randrange error if values are equal and range is empty
        if (t_zero == t_zero + t): return cloned

        mask_end = random.randrange(t_zero, t_zero + t)
        if (replace_with_zero): cloned[0][:,t_zero:mask_end] = 0
        else: cloned[0][:,t_zero:mask_end] = cloned.mean()
    return cloned
```

https://github.com/zcaceres/spec_augment

Data Augmentation for Audio (SOTA)

- **Spectrogram 기반의 Data Augmentation Methods**
  - **Time Masking**

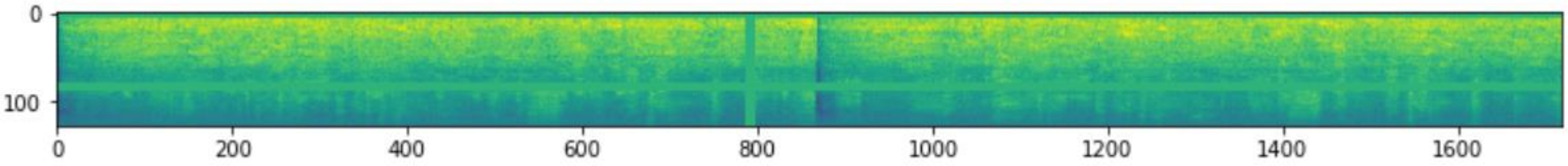


https://github.com/zcaceres/spec_augment

Data Augmentation for Audio (SOTA)

- **Spectrogram 기반의 Data Augmentation Methods**
  - **Combined**

```
combined = time_mask(freq_mask(time_warp(spectro), num_masks=2), num_masks=2)
tensor_to_img(combined)
```




```
torch.Size([1, 128, 1718])
```

https://github.com/zcaceres/spec_augment

Data Augmentation for Audio (SOTA)

- Spectrogram 기반의 Data Augmentation Methods 적용 결과

Table 3: *LibriSpeech 960h WERs (%).*

Method	No LM		With LM	
	clean	other	clean	other
HMM				
Panayotov et al., (2015) [19]			5.51	13.97
Povey et al., (2016) [29]			4.28	
Han et al., (2017) [30]			3.51	8.58
Yang et al. (2018) [31]			2.97	7.50
CTC/ASG				
Collobert et al., (2016) [32]	7.2			
Liptchinsky et al., (2017) [33]	6.7	20.8	4.8	14.5
Zhou et al., (2018) [34]			5.42	14.70
Zeghidour et al., (2018) [35]			3.44	11.24
Li et al., (2019) [36]	3.86	11.95	2.95	8.79
LAS				
Zeyer et al., (2018) [23]	4.87	15.39	3.82	12.76
Zeyer et al., (2018) [37]	4.70	15.20		
Irie et al., (2019) [24]	4.7	13.4	3.6	10.3
Sabour et al., (2019) [38]	4.5	13.3		
Our Work				
LAS	4.1	12.5	3.2	9.8
LAS + SpecAugment	**2.8**	**6.8**	**2.5**	**5.8**

Table 5: *Switchboard 300h WERs (%).*

Method	No LM		With LM	
	SWBD	CH	SWBD	CH
HMM				
Veselý et al., (2013) [40]			12.9	24.5
Povey et al., (2016) [29]			9.6	19.3
Hadian et al., (2018) [41]			9.3	18.9
Zeyer et al., (2018) [23]			8.3	17.3
CTC				
Zweig et al., (2017) [42]	24.7	37.1	14.0	25.3
Audhkhasi et al., (2018) [43]	20.8	30.4		
Audhkhasi et al., (2018) [44]	14.6	23.6		
LAS				
Lu et al., (2016) [45]	26.8	48.2	25.8	46.0
Toshniwal et al., (2017) [46]	23.1	40.8		
Zeyer et al., (2018) [23]	13.1	26.1	11.8	25.7
Weng et al., (2018) [47]	12.2	23.3		
Zeyer et al., (2018) [37]	11.9	23.7	11.0	23.1
Our Work				
LAS	11.2	21.6	10.9	19.4
LAS + SpecAugment (SM)	**7.2**	14.6	**6.8**	14.1
LAS + SpecAugment (SS)	7.3	**14.4**	7.1	**14.0**

Data Augmentation for Audio (SOTA)

- 왜 효과가 있는 걸까?
  - **Time Warping**
    Training Data의 다양성 증가, Test Case로 등장했을 때 예측 가능한 Data의 범위가 넓어지게 됨
  - **Masking**
    주파수 관점에서 보면, N개의 Feature를 통해 음소를 구분해 내야 하지만 N-dim feature vector는 Sparse함
    이 vector를 보다 작게 쪼개서 학습 가능, 음소 구분에 필요한 n개 feature가 동시에 존재해야 할 필요가 없어짐
    Curse of Dimensionality(차원의 저주)를 어느 정도 해소 가능

Phonetic Symbol	Example Word	F_1 (Hz)	F_2 (Hz)	F_3 (Hz)
/ow/	bought	570	840	2410
/oo/	boot	300	870	2240
/u/	foot	440	1020	2240
/a/	hot	730	1090	2440
/uh/	but	520	1190	2390
/er/	bird	490	1350	1690
/ae/	bat	660	1720	2410
/e/	bet	530	1840	2480
/i/	bit	390	1990	2550
/iy/	beet	270	2290	3010

And More…

- **LAS(Listen, Attend and Spell) Model**
- **Learning Rate Schedules**
- **Shallow Fusion with LM(Language Model)**

3.1. LAS Network Architectures

We use Listen, Attend and Spell (LAS) networks [6] for end-to-end ASR studied in [25], for which we use the notation LAS-d-w. The input log mel spectrogram is passed in to a 2-layer Convolutional Ne
stride of 2. The
coder consisting
size w to yield a
tors are fed into a
which yields the t
using a Word Piec
for LibriSpeech a
riSpeech 960h is
For the Switchboa
are combined wit

3.3. Shallow Fusion with Language Models

While we are able to get state-of-the-art results with augmentation, we can get further improvements by using a language model. We thus incorporate an RNN language model by shallow fusion for both tasks. In shallow fusion, the "next token" $\mathbf{y}^*$ in the decoding process is determined by

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}} \left(\log P(\mathbf{y}|\mathbf{x}) + \lambda \log P_{LM}(\mathbf{y}) \right) , \quad (1)$$

i.e., by jointly scoring the token using the base ASR model and the language model. We also use a coverage penalty c [29].

For LibriSpeech, we use a two-layer RNN with embedding dimension 1024 used in [25] for the LM, which is trained on the LibriSpeech LM corpus. We use identical fusion parameters ($\lambda = 0.35$ and $c = 0.05$) used in [25] throughout.

For Switchboard, we use a two-layer RNN with embedding dimension 256, which is trained on the combined transcripts of the Fisher and Switchboard datasets. We find the fusion parameters via grid search by measuring performance on RT-03 (LDC2007S10). We discuss the fusion parameters used in individual experiments in section 4.2.

3.2. Learning Rate Schedules

The learning rate schedule turns out to be an importa
in determining the performance of ASR networks, es
so when augmentation is present. Here, we introduce
schedules that serve two purposes. First, we use these s
to verify that a longer schedule improves the final perf
of the network, even more so with augmentation (Table
ond, based on this, we introduce very long schedules
used to maximize the performance of the networks.