한국소음진동공학회 AI혁신위원회 주관
2022년 인공지능 강습회

AI 기계이상진단을 위한 인공지능 학습 기법

제 8강 소음 데이터를 이용한 이상진단 (실습)

한국과학기술원 전기및전자공학부

최정우
jwoo@kaist.ac.kr

KAIST EE

목차

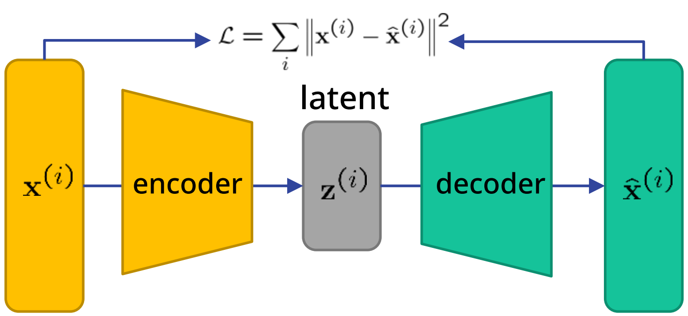
- 소음 진동 기반 이상진단을 위한 모델들

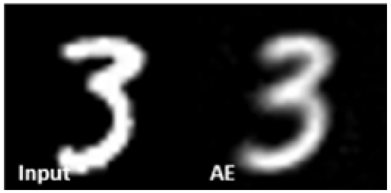
- 시간 맥락 이해를 위한 WaveNet

- 효율적인 분류기 구성을 위한 ResNet

Autoencoder를 사용한 재현기의 문제점

- Imperfect decoding
  - Reconstruction from low-resolution data using transposed convolution
  - Image degradation is inevitable: high reconstruction error bias

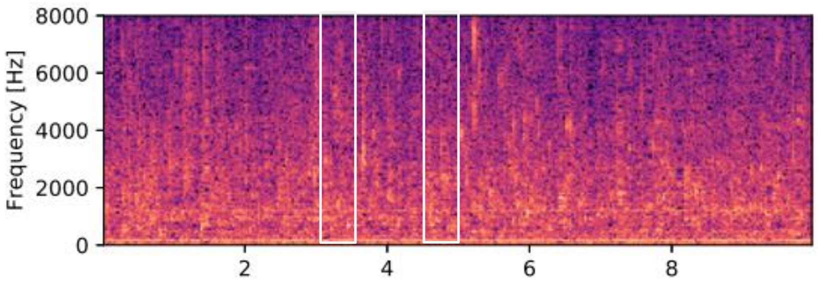




- Heuristic selection of latent dimension
  - Representation can be overfitted or underfitted
    - e.g.) too small latent vars. → insufficient representation
      too many latent vars. → redundant representation

Autoencoder의 대안 모델

- **Understanding time series data**
  - Most of real systems are causal
  - Future and past data are linked by some relations
  - Can future data be predicted by past & present data? (or vice versa)



  - c.f.) **masked prediction task** of language processing model

Training data: the man went to the store, he bought a gallon of milk.

Input: the man went to the [____], he bought a [_____] of milk.

Autoencoder의 대안 모델

- **Architectures to extract temporal context**

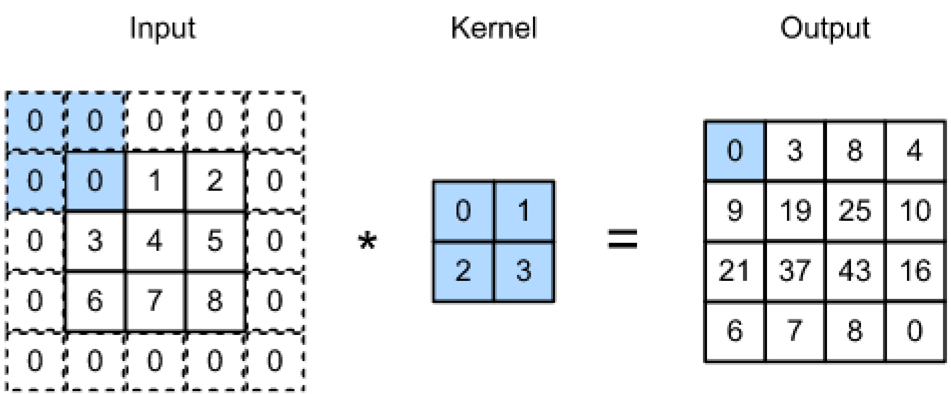
- Recurrent neural network (RNN), Long-term short-term memory (LSTM)
- Masked autoencoder
- WaveNet

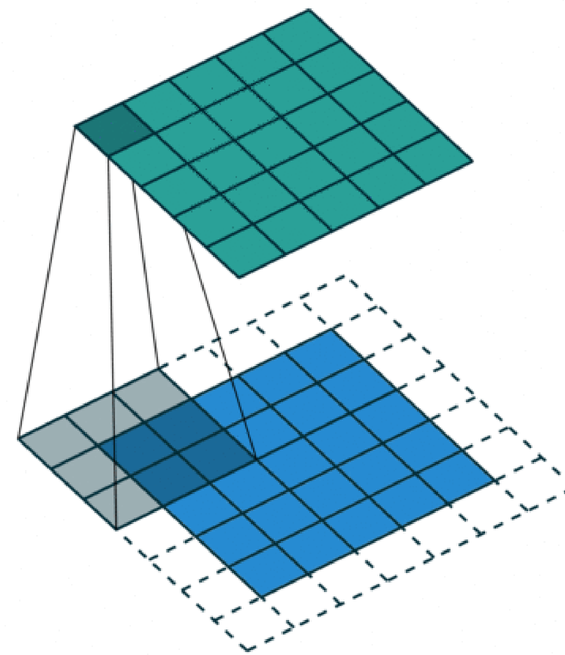
WaveNet

Dilated causal convolution

Convolution layer

- Review: convolution & kernel

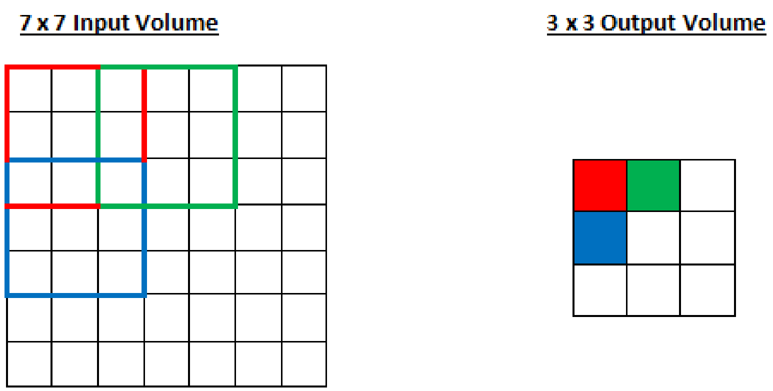


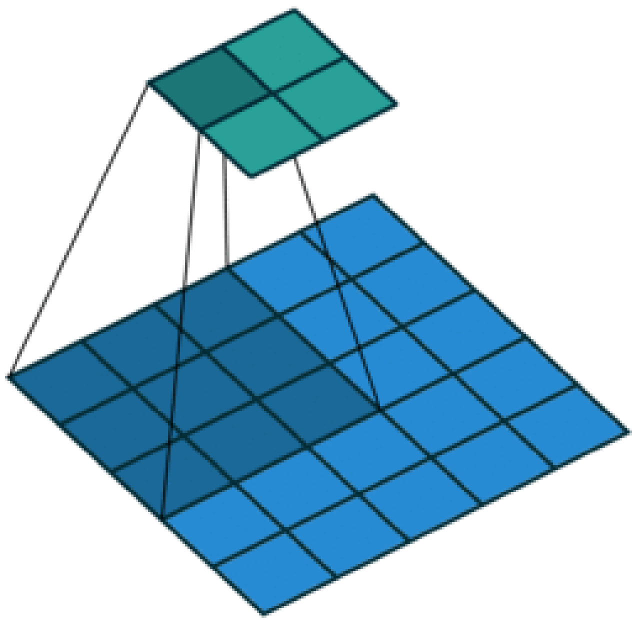


2D convolution using a kernel size of 3, stride of 1 and padding 1

Convolution layer

- Review: stride

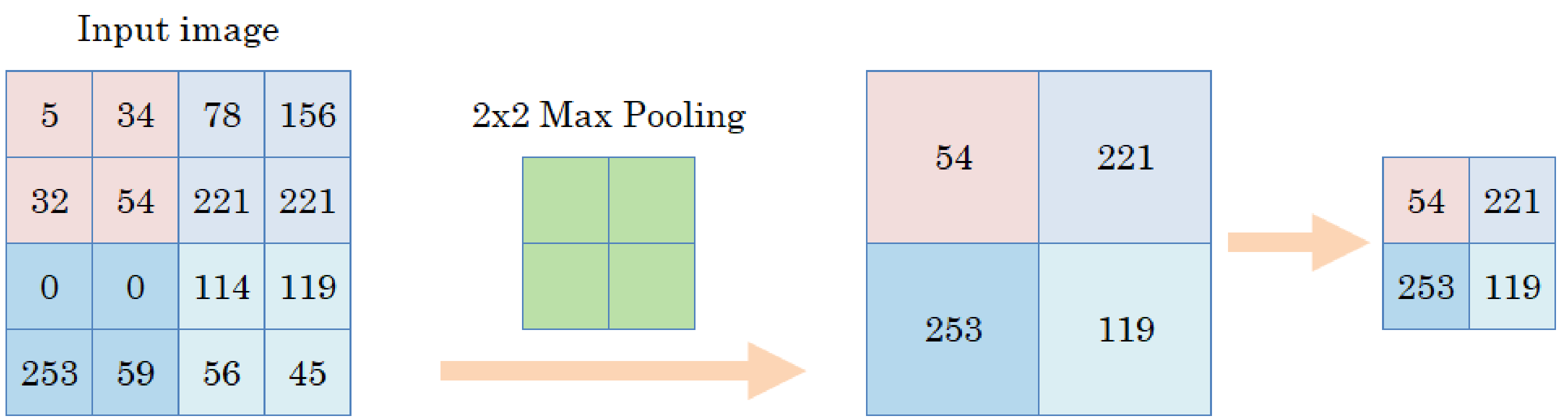




2D convolution with no padding, stride of 2 and kernel of 3

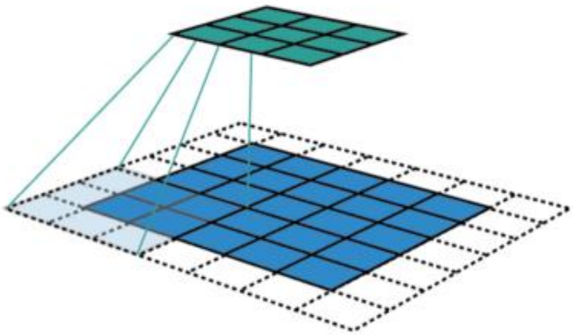
Convolution layer

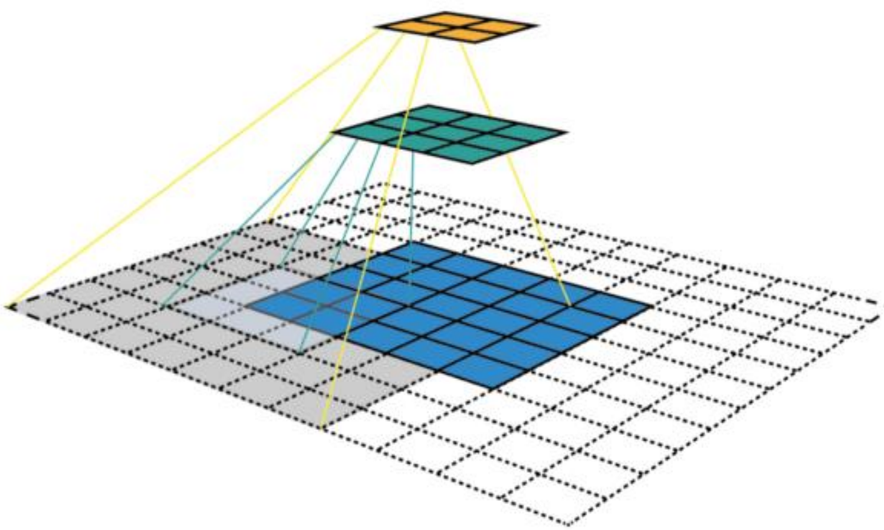
- Review: pooling



Convolution layer

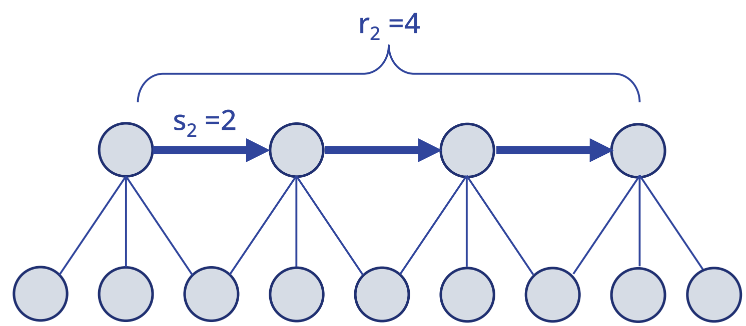
- Receptive field





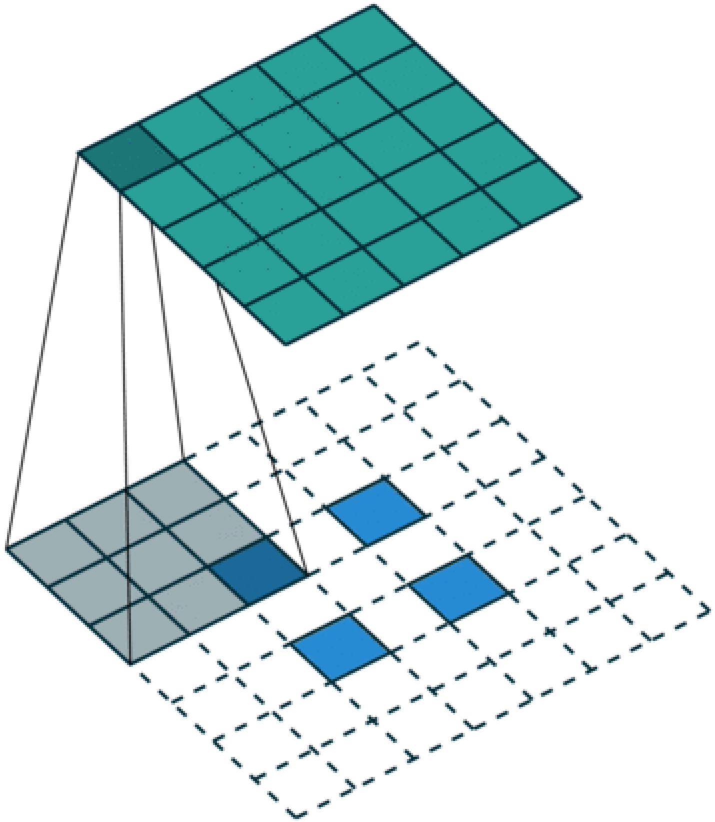
$$r_{i-1} = s_i \times (r_i - 1) + k_i$$

receptive field   stride   kernel size



Transposed convolution

- Deconvolution for recovering original image size
- Using strides to input image

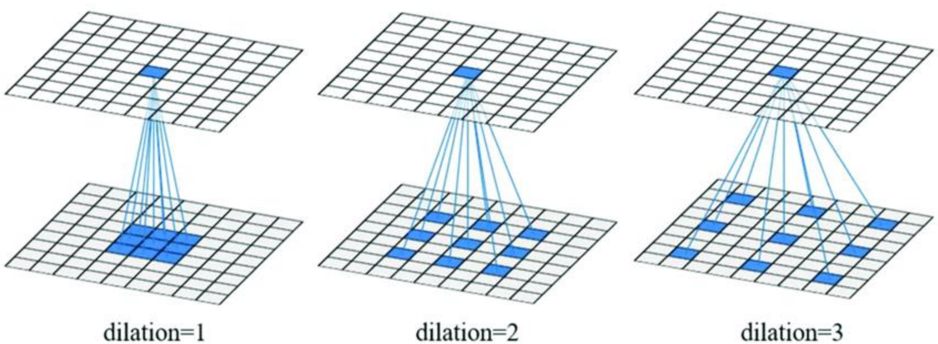


Transposed 2D convolution with no padding, stride of 2 and kernel of 3

- Checkerboard artifacts

Dilated convolution

- Using sparse kernel instead of stride or pooling
  - Stride = 1 (no loss of information): all input pixels are processed
  - Use sparse kernels
  - Wide receptive field without increase of parameters

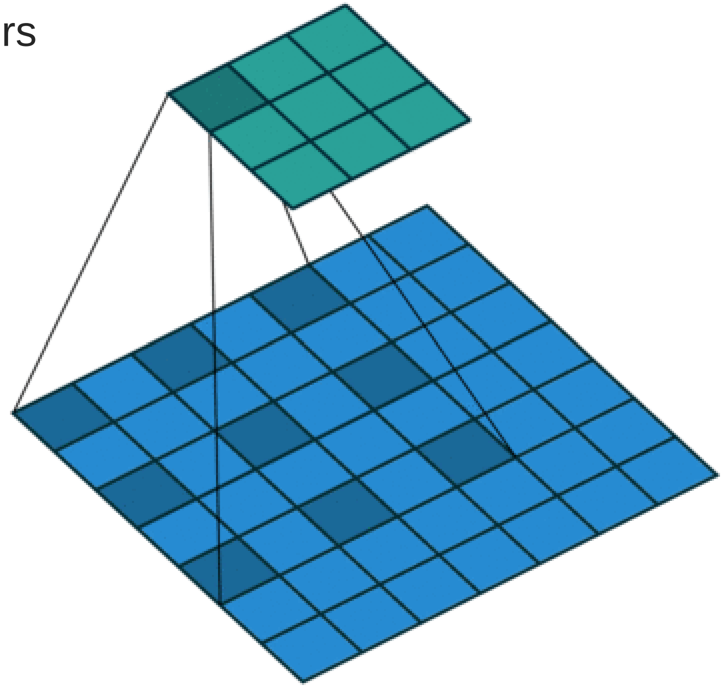


$$k_{i,e} = f_i \times (k_i - 1) + 1$$

effective kernel size = dilation factor * (kernel size -1) +1

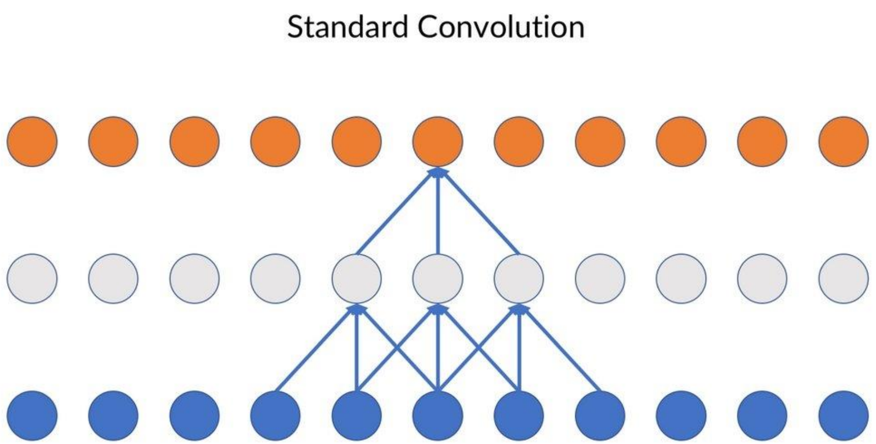
$$r_{i-1} = 1 \times (r_i - 1) + k_{i,e}$$

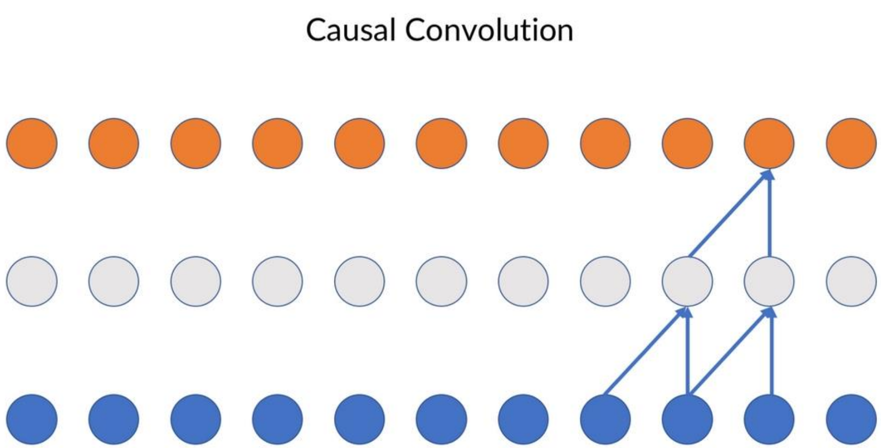
receptive field



Dilated convolution with dilation factor 2, kernel size 3

Causal / non-causal convolutions

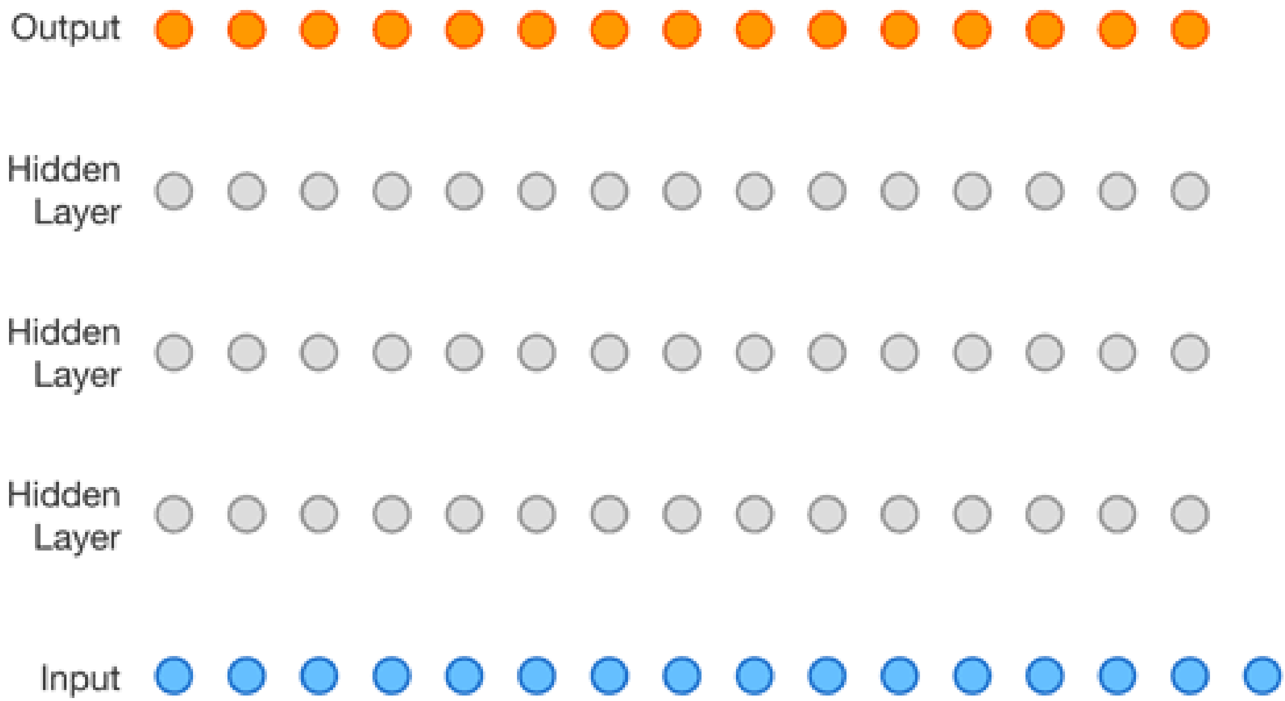




images from the book, "Machine Learning for Finance," by Jannes Klaas, Packt

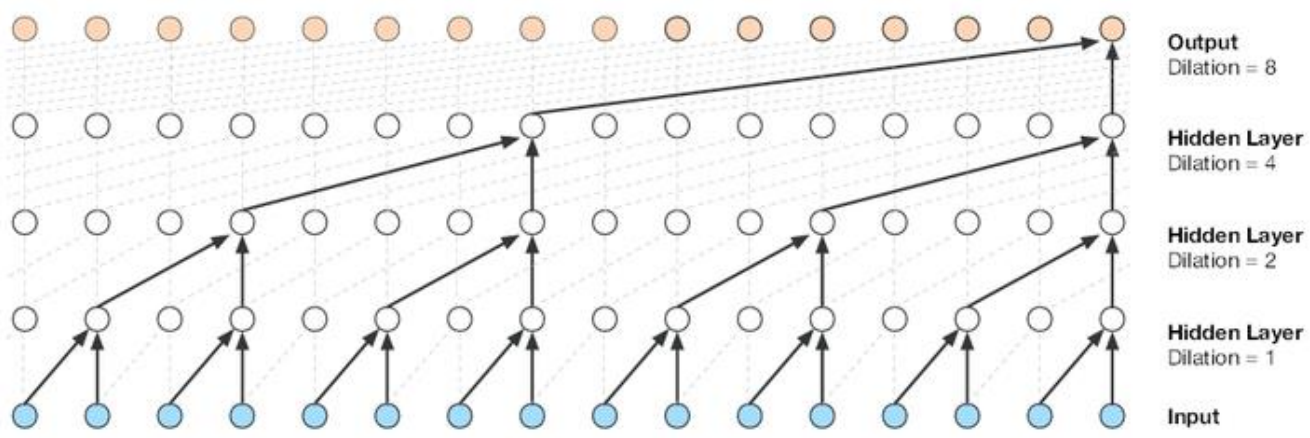
Causal dilated convolution

- Causal + Dilated convolution

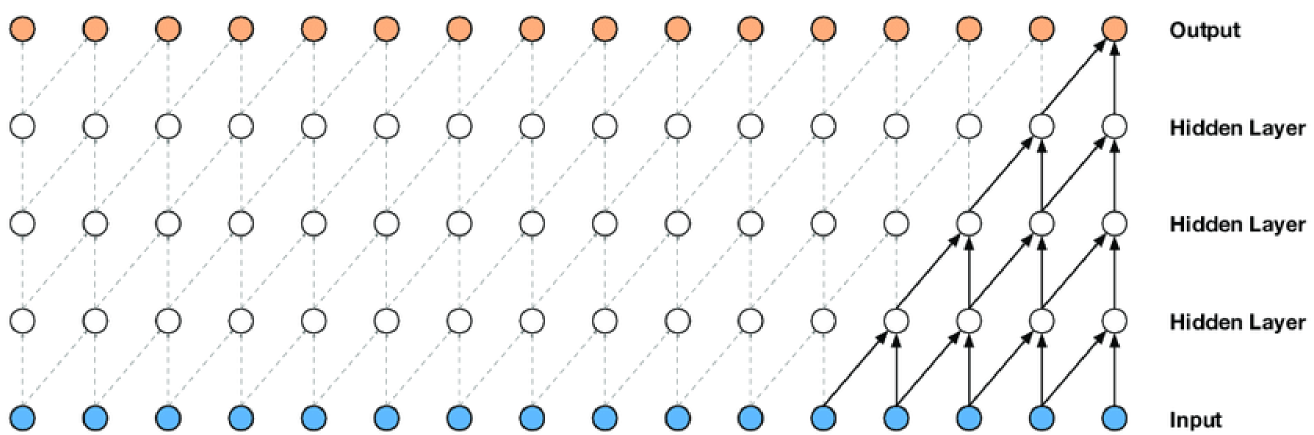


Causal dilated convolution

- Causal + Dilated convolution

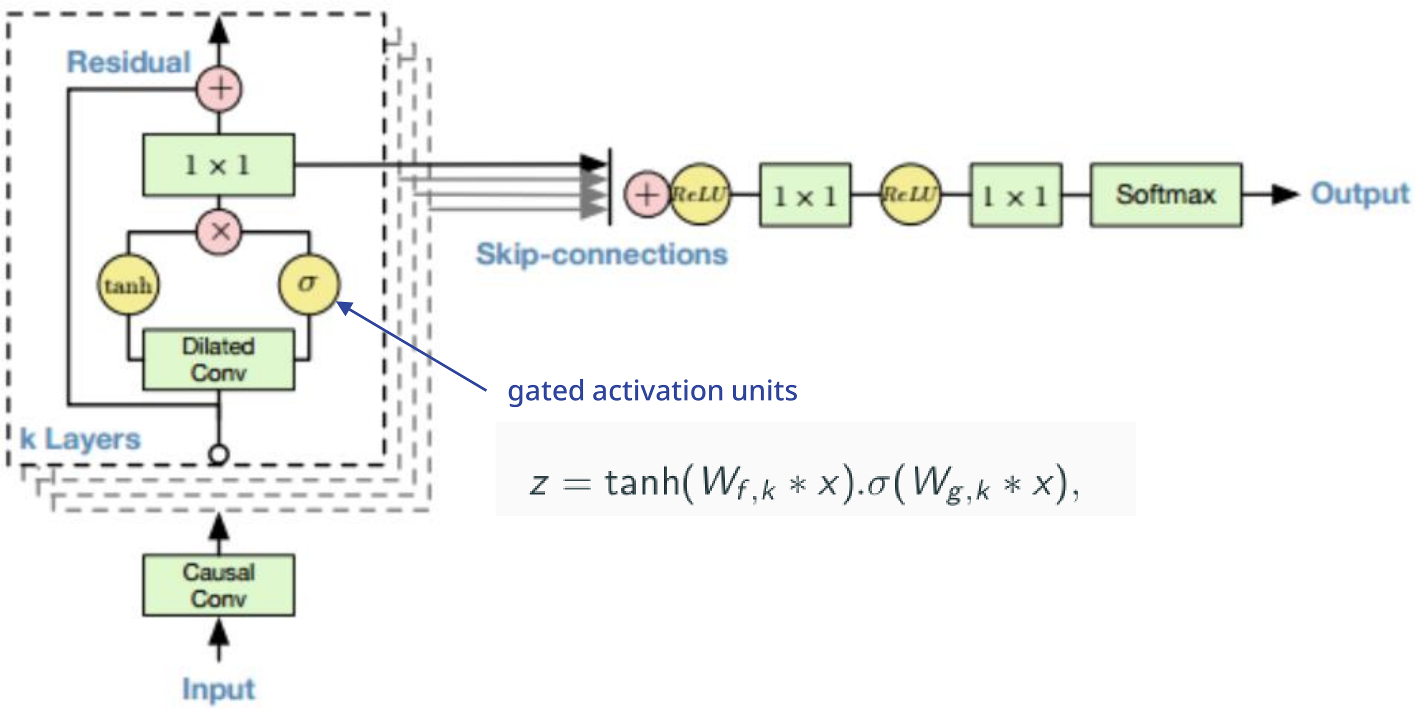


c.f.) causal only



WaveNet architecture

- Causal dilated convolution for auto-regressive prediction
- Residual block & skip connections to s
- Gated activation unit



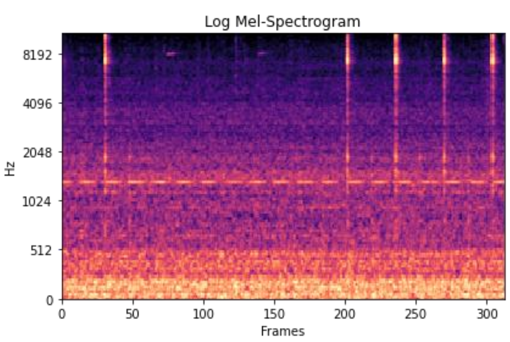
gated activation units

$$z = \tanh(W_{f,k} * x) . \sigma(W_{g,k} * x),$$

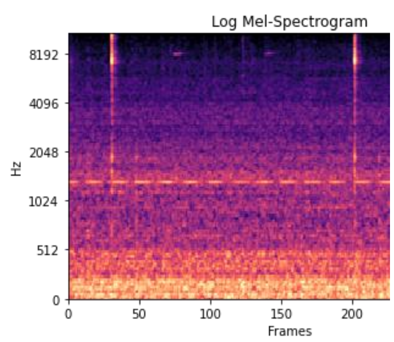
Freak WaveNet architecture

- Future spectrogram prediction task using WaveNet
  - Frequencies of mel-spectrogram → channel dimension
  - GroupNorm → LayerNorm

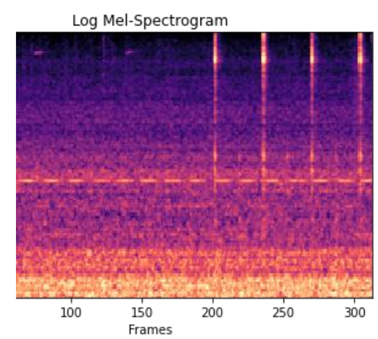
- Implementation (63 frames to predict 1 future frame)
  - Target: spectrogram for time frames from (63) to (end)
  - Input: spectrogram for time frames from (0) to (end-63)
  - Causal dilated convolution's receptive field is set to 63



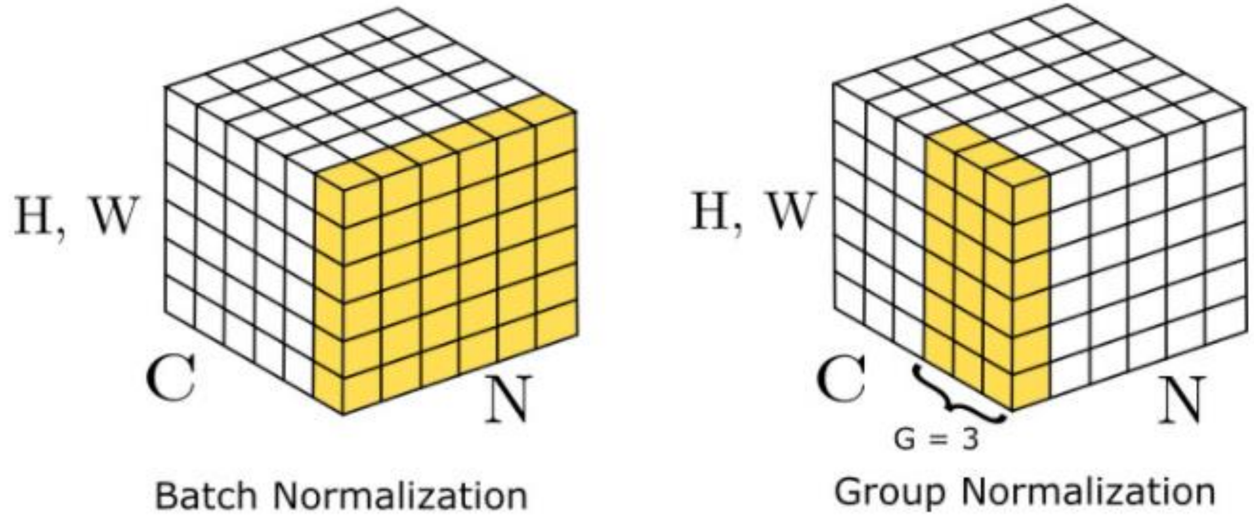
input



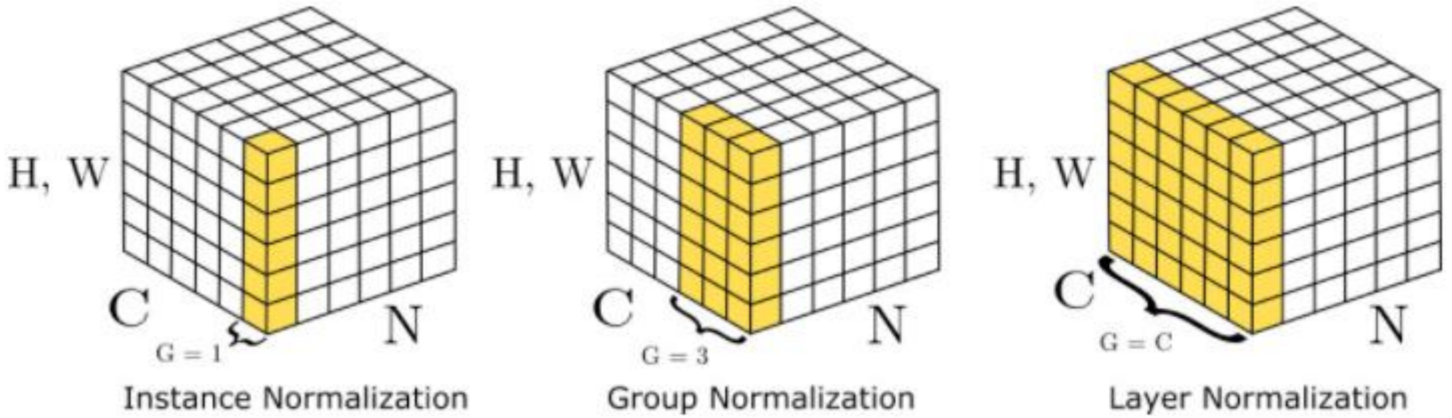
target



GroupNorm



H, W
C
N
Batch Normalization
H, W
C
N
G = 3
Group Normalization

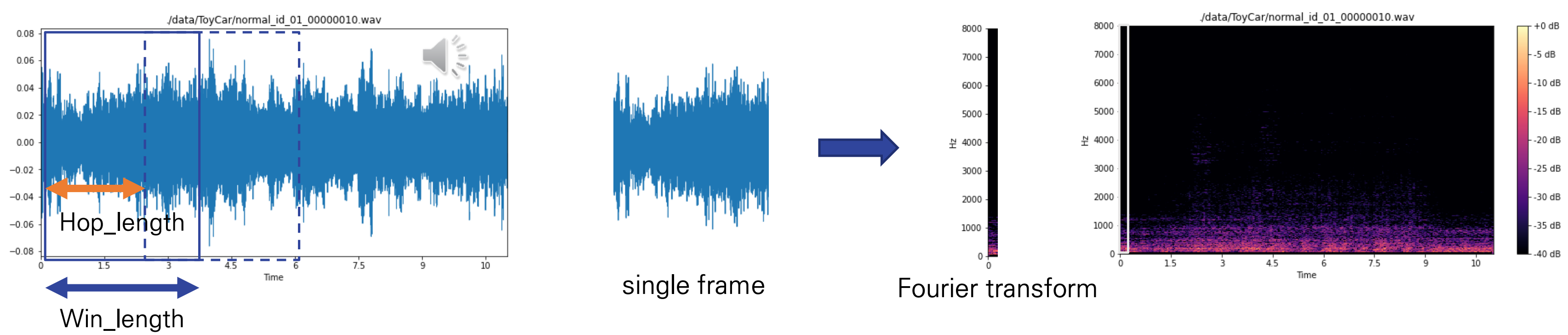


H, W
C
N
G = 1
Instance Normalization
H, W
C
N
G = 3
Group Normalization
H, W
C
N
G = C
Layer Normalization

Baseline System

- Acoustic features

```
# Parameters for mel-spectrogram
n_fft = 2048
hop_length = 512
n_mels = 128
power = 2

# Parameters for WaveNet
n_mul = 4
kernel_size = 3

# Training parameters
EPOCHS = 20
BATCH = 32
```



Function definition

For the convenience of the code, we define some functions below here.

- `file_load` : reads one 10-sec sound file and returns 1D array `y` and sampling rate `sr`
- `file_list_generator` : returns file_name list in `target_dir`
- `file_to_log_mel` : convert one file to log-mel spectrogram for use as input to the model
- `list_to_dataset` : returns a total dataset of train data

[ ]

```
def file_load(wav_name):
  try:
    return librosa.load(wav_name, sr=None, mono=False)
  except:
    print('file_broken or not exists!! : {}'.format(wav_name))


def file_list_generator(target_dir):
  training_list_path = os.path.abspath('{dir}/*.wav'.format(dir=target_dir))
  files = sorted(glob.glob(training_list_path))
  if len(files) == 0:
    print('no_wav_file!!')
  return files
```

Function definition

```
def file_to_log_mel(file_name, n_mels, n_fft, hop_length, power):
  y, sr = file_load(file_name)
  mel_spectrogram = librosa.feature.melspectrogram(y=y,
                                                   sr=sr,
                                                   n_fft=n_fft,
                                                   hop_length=hop_length,
                                                   n_mels=n_mels,
                                                   power=power)
  log_mel_spectrogram = 20.0 / power * np.log10(mel_spectrogram + sys.float_info.epsilon)
  return log_mel_spectrogram


def list_to_dataset(file_list, n_mels, n_fft, hop_length, power):
  for idx in tqdm(range(len(file_list))):
    log_mel = file_to_log_mel(file_list[idx],
                              n_mels=n_mels,
                              n_fft=n_fft,
                              hop_length=hop_length,
                              power=power)
    if idx == 0:
      dataset = np.zeros((len(file_list), 1, len(log_mel[:,0]), len(log_mel[0,:])), float)
    dataset[idx, 0, :, :] = log_mel
  return dataset
```

WaveNet code

- CausalConv1d

```python
class CausalConv1d(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size, dilation=1):
        super(CausalConv1d, self).__init__()
        self.in_channels = in_channels
        self.out_channels = out_channels
        self.kernel_size = kernel_size
        self.dilation = dilation
        self.conv1 = self.causal_conv(self.in_channels, self.out_channels, self.kernel_size, self.dilation)
        self.padding = self.conv1.padding[0]

    def causal_conv(self, in_channels, out_channels, kernel_size, dilation):
        pad = (kernel_size - 1) * dilation
        return nn.Conv1d(in_channels, out_channels, kernel_size, padding=pad, dilation=dilation)

    def forward(self, x):
        x = self.conv1(x)
        x = x[:, :, :-self.padding]
        return x
```

WaveNet code

- WaveNet class

```
class WaveNet(nn.Module):
    def __init__(self, n_channel, n_mul, kernel_size):
        super(WaveNet, self).__init__()
        self.n_channel = n_channel
        self.n_mul = n_mul
        self.kernel_size = kernel_size

        self.n_filter = self.n_channel * self.n_mul
        self.group_norm1 = nn.GroupNorm(1, self.n_channel)
        self.conv1 = nn.Conv1d(self.n_channel, self.n_filter, 1)

        self.block1 = ResidualBlock(self.n_channel, self.n_mul, self.kernel_size, 1)
        self.block2 = ResidualBlock(self.n_channel, self.n_mul, self.kernel_size, 2)
        self.block3 = ResidualBlock(self.n_channel, self.n_mul, self.kernel_size, 4)
        # self.block4 = ResidualBlock(self.n_channel, self.n_mul, self.kernel_size, 8)
        # self.block5 = ResidualBlock(self.n_channel, self.n_mul, self.kernel_size, 16)

        self.relu1 = nn.ReLU()
        self.group_norm2 = nn.GroupNorm(1, self.n_channel)
        self.conv2 = nn.Conv1d(self.n_channel, self.n_channel, 1)
        self.relu2 = nn.ReLU()
        self.group_norm3 = nn.GroupNorm(1, self.n_channel)
        self.conv3 = nn.Conv1d(self.n_channel, self.n_channel, 1)
```

WaveNet code

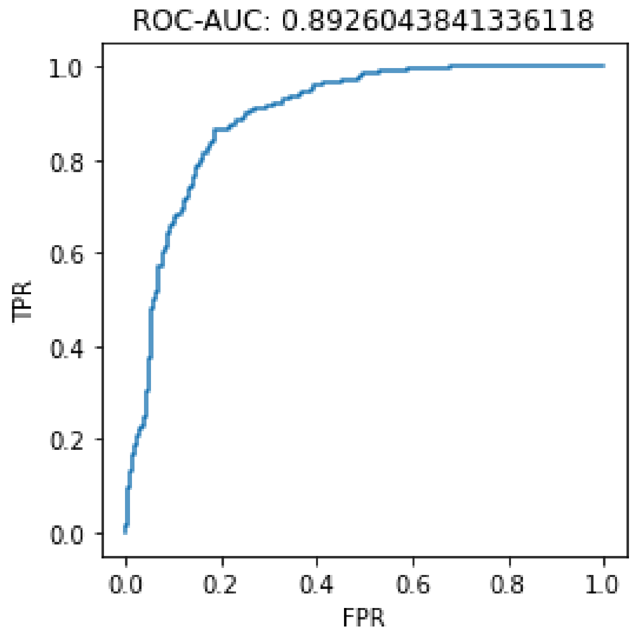
- Residual block class

```
class ResidualBlock(nn.Module):
    def __init__(self, n_channel, n_mul, kernel_size, dilation_rate):
        super(ResidualBlock, self).__init__()
        self.n_channel = n_channel
        self.n_mul = n_mul
        self.kernel_size = kernel_size
        self.dilation_rate = dilation_rate
        self.n_filter = self.n_channel * self.n_mul

        self.sigmoid_group_norm = nn.GroupNorm(1, self.n_filter)
        self.sigmoid_conv = CausalConv1d(self.n_filter, self.n_filter, self.kernel_size, self.dilation_rate)
        self.tanh_group_norm = nn.GroupNorm(1, self.n_filter)
        self.tanh_conv = CausalConv1d(self.n_filter, self.n_filter, self.kernel_size, self.dilation_rate)

        self.skip_group_norm = nn.GroupNorm(1, self.n_filter).to(device)
        self.skip_conv = nn.Conv1d(self.n_filter, self.n_channel, 1)
        self.residual_group_norm = nn.GroupNorm(1, self.n_filter)
        self.residual_conv = nn.Conv1d(self.n_filter, self.n_filter, 1)

    def forward(self, x):
        x1 = self.sigmoid_group_norm(x)
        x1 = self.sigmoid_conv(x1)
        x2 = self.tanh_group_norm(x)
        x2 = self.tanh_conv(x2)
        x1 = nn.Sigmoid()(x1)
        x2 = nn.Tanh()(x2)
        x = x1 * x2
```

```
        x1 = self.skip_group_norm(x)
        skip = self.skip_conv(x1)
        x2 = self.residual_group_norm(x)
        residual = self.residual_conv(x2)
        return skip, residual
```

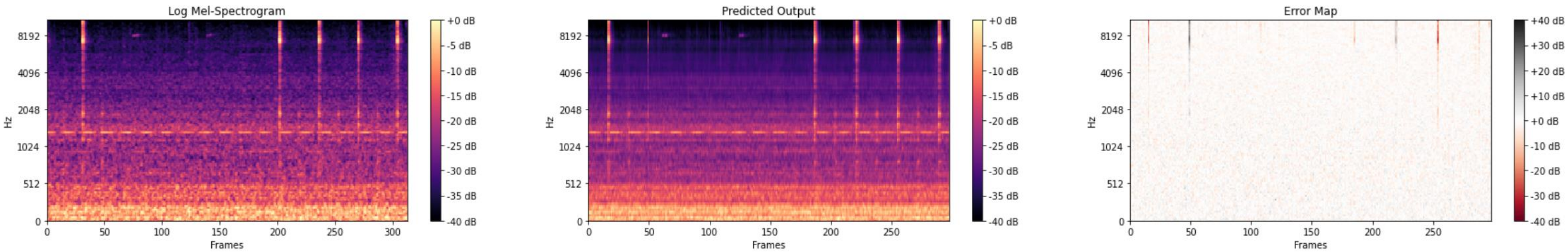
WaveNet results

```
Epoch 1
-------------------------------
loss: 649.934082  [    0/ 3291]
loss: 503.684448  [  960/ 3291]
loss: 328.259644  [ 1920/ 3291]
loss: 220.738785  [ 2880/ 3291]
Epoch 2
-------------------------------
loss: 159.997742  [    0/ 3291]
loss: 88.883598  [  960/ 3291]
loss: 42.684723  [ 1920/ 3291]
loss: 27.946882  [ 2880/ 3291]

Epoch 99
-------------------------------
loss: 4.521351  [    0/ 3291]
loss: 4.461485  [  960/ 3291]
loss: 4.709872  [ 1920/ 3291]
loss: 4.805975  [ 2880/ 3291]
Epoch 100
-------------------------------
loss: 4.695333  [    0/ 3291]
loss: 4.460145  [  960/ 3291]
loss: 4.430577  [ 1920/ 3291]
loss: 4.708189  [ 2880/ 3291]
```

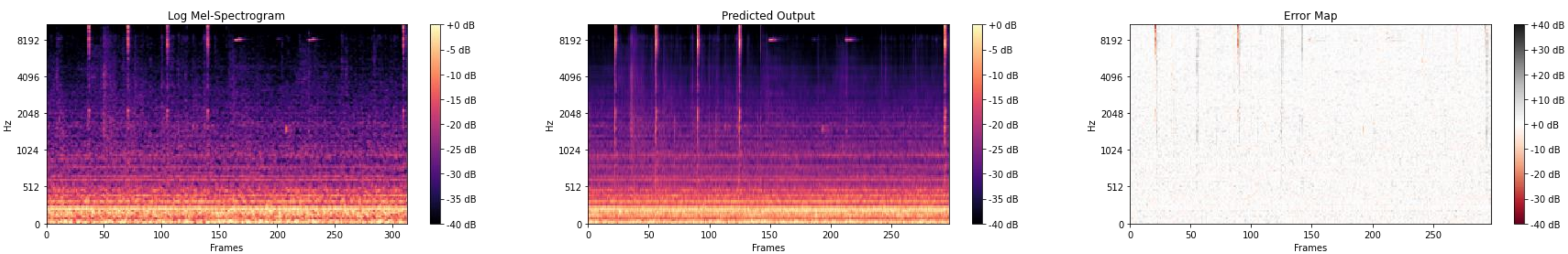


WaveNet results

- Reconstruction of normal data



- Reconstruction of abnormal data

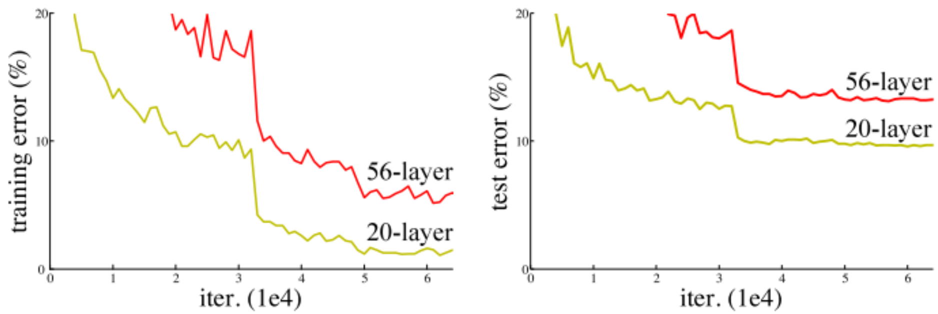


ResNet

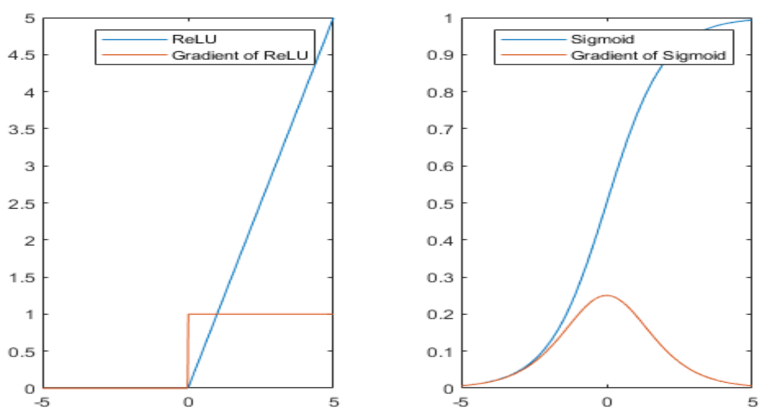
Deeper convolutional network
for Classification-based Anomaly Detection

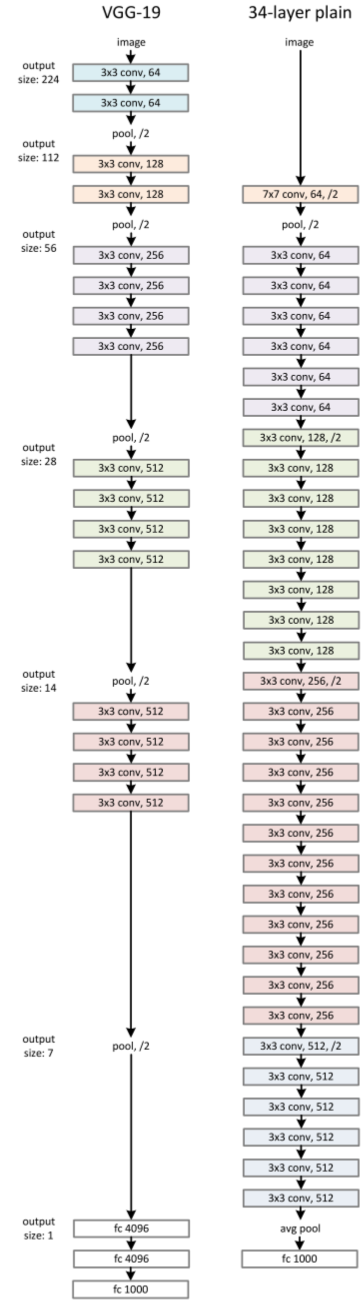
Problem of deeper network

- Vanishing gradient issue for deeper network



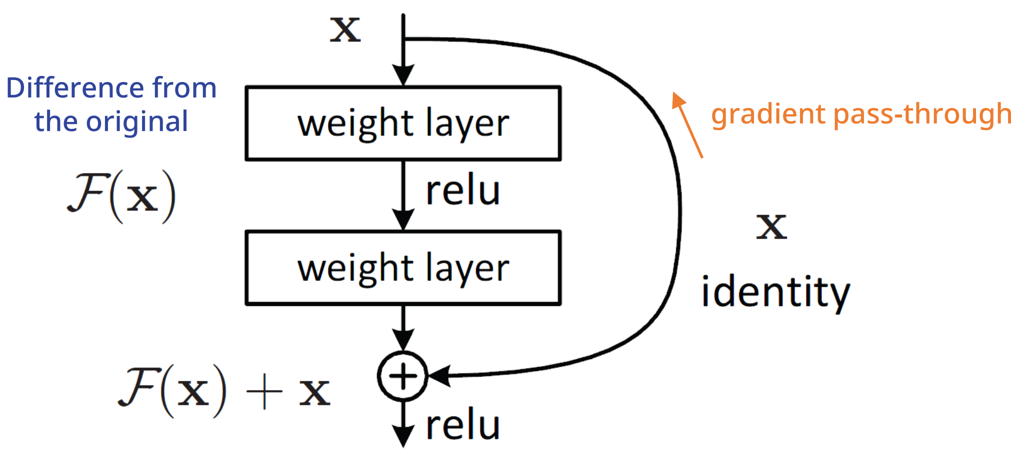
https://arxiv.org/pdf/1512.03385.pdf





ResNet

- Residual connection
  - Hypothesis: it is easier to optimize the residual mapping than to optimize the original
  - Learning the difference from original

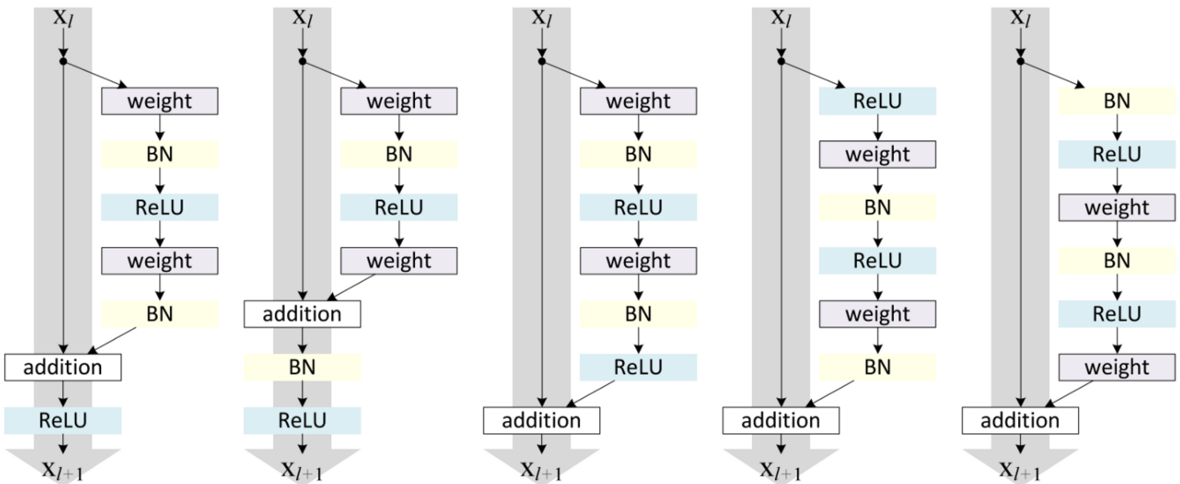


$$\frac{\partial(\mathcal{F}(\mathbf{x}) + \mathbf{x})}{\partial \mathbf{x}} = \frac{\partial \mathcal{F}(\mathbf{x})}{\partial \mathbf{x}} + 1$$

  - Backprop through residual connection: resolving gradient vanishing problem

ResNet

- Residual connection



(a) original

(b) BN after addition

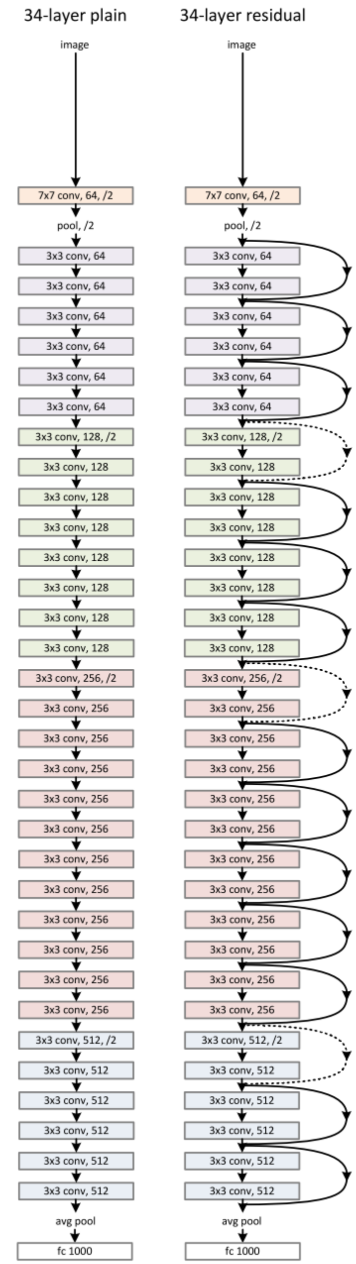
(c) ReLU before addition

(d) ReLU-only pre-activation

(e) **full pre-activation**

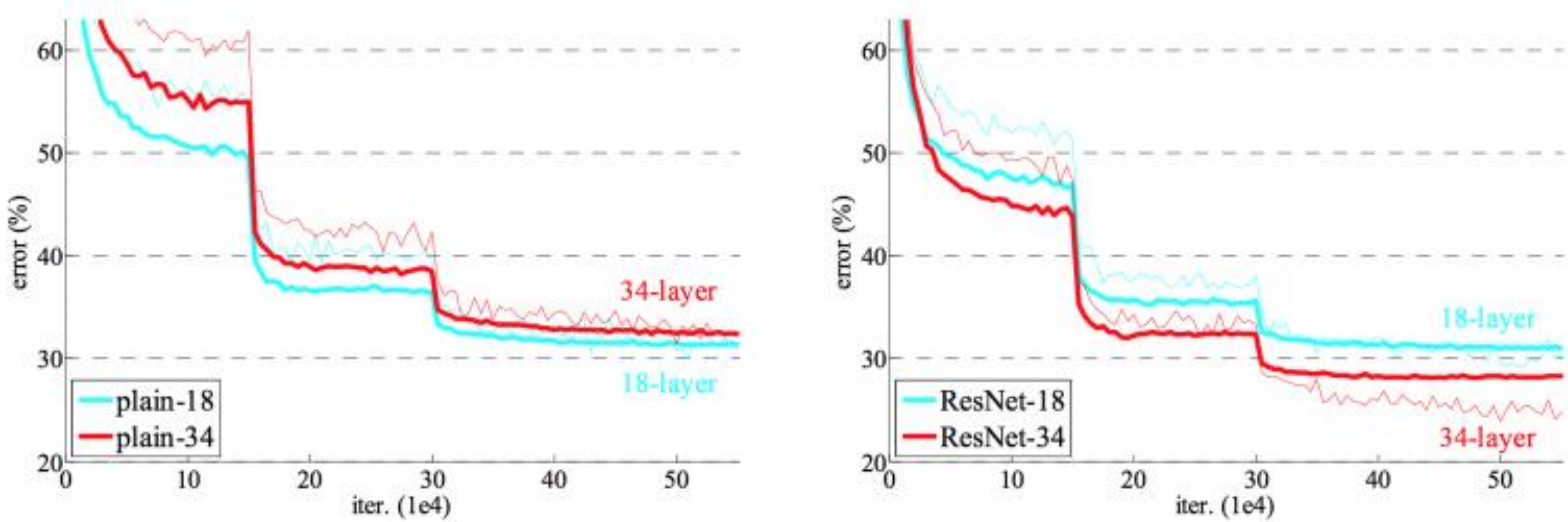
batch norm prohibits backprop

ReLU only produces positive values



ResNet

- Training on ImageNet



  - ResNet is not a SOTA technique but rather a de-facto baseline

Code

- Generating train labels

4-1. Generate Train Label

Create a data label. In `valve` dataset, there exists four classes: `id_00`, `id_02`, `id_04`, `id_06`. It can be converted to a one-hot vector.

```
label_list = ['id_00', 'id_02', 'id_04', 'id_06']
train_label = torch.LongTensor([idx for file_name in files for idx, label_idx in enumerate(label_list) if label_idx in file_name])
train_label = nn.functional.one_hot(train_label, num_classes=len(label_list))
```

```
train_data = torch.Tensor(train_data)
train_dataset = TensorDataset(train_data, train_label)
train_dataloader = DataLoader(train_dataset, batch_size=BATCH, shuffle=True)
```

Code

- Residual block

```
class ResidualBlock(nn.Module):
    def __init__(self, in_channel, out_channel, projection=False):
        super(ResidualBlock, self).__init__()
        self.in_channel = in_channel
        self.out_channel = out_channel
        self.projection = projection
        if self.projection:
            self.conv1 = nn.Conv2d(self.in_channel, self.out_channel, kernel_size=3, stride=2, padding=(1, 1))
        else:
            self.conv1 = nn.Conv2d(self.in_channel, self.out_channel, kernel_size=3, padding=(1, 1))
        self.bn1 = nn.BatchNorm2d(self.out_channel)
        self.relu = nn.ReLU()
        self.conv2 = nn.Conv2d(self.out_channel, self.out_channel, kernel_size=3, padding='same')
        self.bn2 = nn.BatchNorm2d(self.out_channel)
        if self.projection:
            self.downsample = nn.Conv2d(self.in_channel, self.out_channel, stride=2, kernel_size=1)
        else:
            self.downsample = nn.Conv2d(self.in_channel, self.out_channel, kernel_size=1)
```

Code

- Forward propagation (residual block)

```
def forward(self, x):
    out = self.conv1(x)
    out = self.bn1(out)
    out = self.relu(out)
    out = self.conv2(out)
    out = self.bn2(out)
    if self.projection:
        skip = self.downsample(x)
    else:
        skip = x
    out += skip
    out = self.relu(out)
    return out
```

Code

- ResNet

```
class ResNet(nn.Module):
    def __init__(self, n_class):
        super(ResNet, self).__init__()
        self.n_channel = 8
        self.n_class = n_class
        self.conv1 = nn.Conv2d(1, self.n_channel, kernel_size=7, stride=2, padding=(3, 2))
        self.bn1 = nn.BatchNorm2d(self.n_channel)
        self.relu = nn.ReLU()
        self.pooling1 = nn.MaxPool2d(kernel_size=3, stride=2, padding=(1, 1))
        self.block1 = ResidualBlock(self.n_channel, self.n_channel)
        self.block2 = ResidualBlock(self.n_channel, self.n_channel)
        self.block3 = ResidualBlock(self.n_channel, self.n_channel * 2, True)
        self.block4 = ResidualBlock(self.n_channel * 2, self.n_channel * 2)
        self.block5 = ResidualBlock(self.n_channel * 2, self.n_channel * 4, True)
        self.block6 = ResidualBlock(self.n_channel * 4, self.n_channel * 4)
        self.block7 = ResidualBlock(self.n_channel * 4, self.n_channel * 8, True)
        self.block8 = ResidualBlock(self.n_channel * 8, self.n_channel * 8)
        self.gap1 = nn.AdaptiveAvgPool2d((1, 1))
        self.fc = nn.Linear(self.n_channel * 8, self.n_class)
```

Code

- Forward propagation (ResNet)

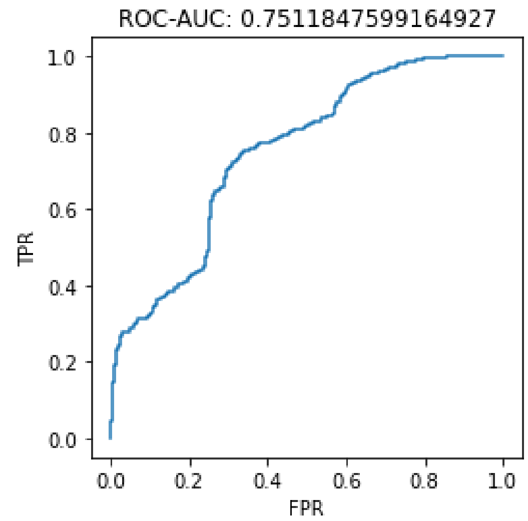
```
def forward(self, x):
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.pooling1(x)
    x = self.block1(x)
    x = self.block2(x)
    x = self.block3(x)
    x = self.block4(x)
    x = self.block5(x)
    x = self.block6(x)
    x = self.block7(x)
    x = self.block8(x)
    x = self.gap1(x)
    x = torch.flatten(x, 1)
    x = self.fc(x)
    return x
```

```
model = ResNet(len(label_list)).to(device)
summary(model, input_size=(BATCH, 1, n_mels, 313))
```

ResNet results

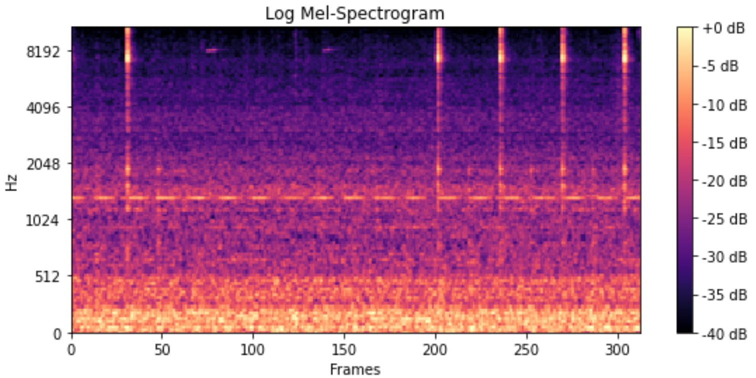
```
Epoch 1
-------------------------------
loss: 1.563473  [    0/ 3291]
loss: 0.226422  [  960/ 3291]
loss: 0.064789  [ 1920/ 3291]
loss: 0.117298  [ 2880/ 3291]
Epoch 2
-------------------------------
loss: 0.015003  [    0/ 3291]
loss: 0.006293  [  960/ 3291]
loss: 0.005425  [ 1920/ 3291]
loss: 0.002784  [ 2880/ 3291]
```

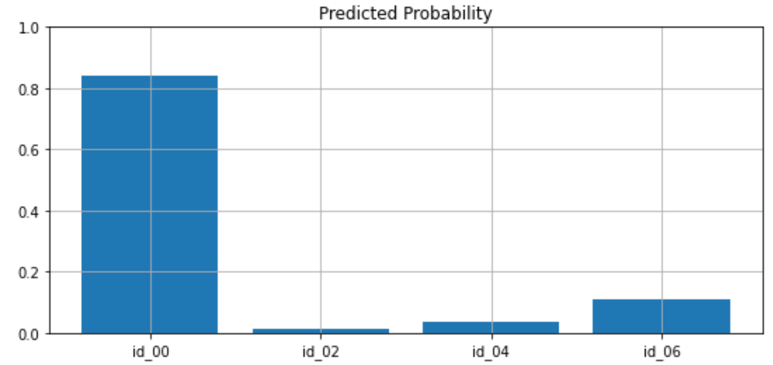
```
Epoch 19
-------------------------------
loss: 0.000008  [    0/ 3291]
loss: 0.000020  [  960/ 3291]
loss: 0.000082  [ 1920/ 3291]
loss: 0.000016  [ 2880/ 3291]
Epoch 20
-------------------------------
loss: 0.000059  [    0/ 3291]
loss: 0.000018  [  960/ 3291]
loss: 0.000033  [ 1920/ 3291]
loss: 0.000041  [ 2880/ 3291]
```



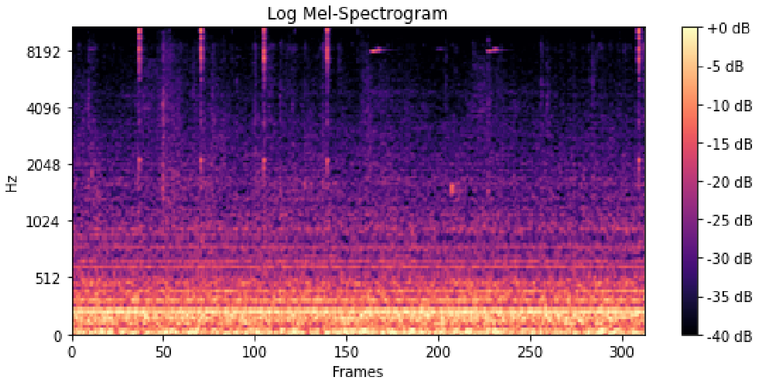
ResNet results

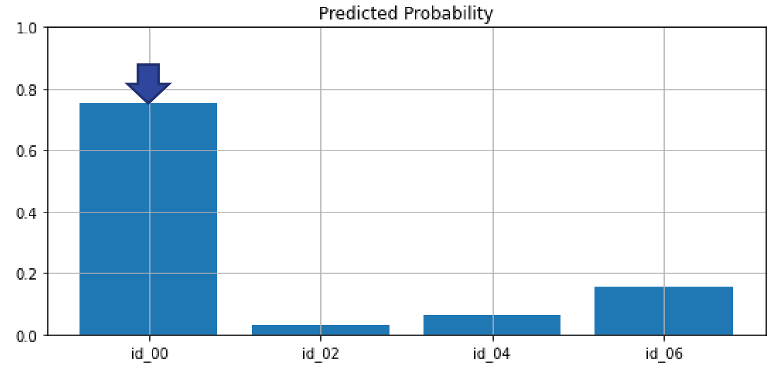
- Normal data





- Anomalous data

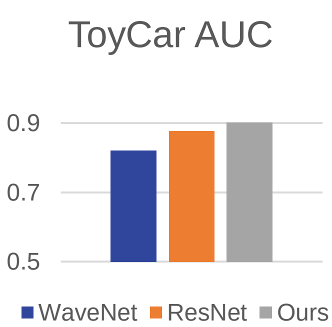


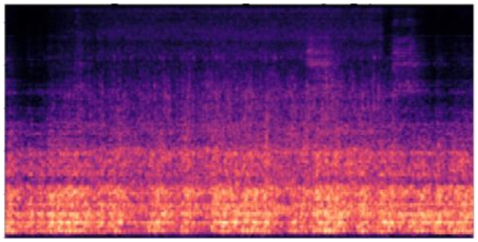


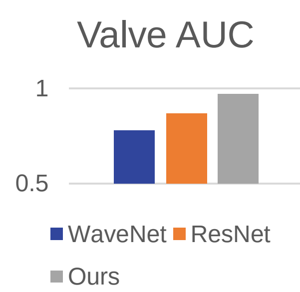
요약 및 결론

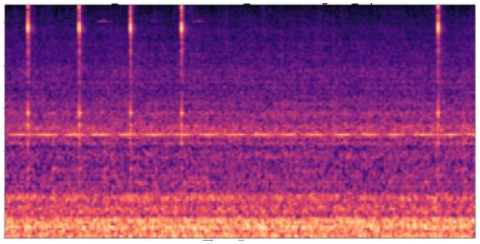
- **WaveNet 을 사용한 미래 예측 (재현) 태스크**
  - 한정된 frame의 과거 데이터를 사용하여 미래 frame 예측
  - Dilated convolution을 사용한 receptive field 확장

- **재현 태스크의 한계**
  - Stationary signal의 경우, 재현을 통해 배울 수 있는 지식이 없음 (현재 데이터의 단순 copy)
  - 시계열 맥락이 없는 경우, 미래 예측에 한계가 있음
  - 데이터 특성에 따라 필요한 receptive field 길이가 달라짐









요약 및 결론

- ResNet을 **사용한 자체 레이블 분류 태스크**
  - 본 문제와 관련 없는 내부의 레이블을 사용하여 분류 학습
  - 데이터 분류를 위해 정상 데이터의 특징을 추출하도록 학습

- **분류 태스크의 한계**
  - 정상 데이터 내부 레이블이 반드시 필요 (작동 조건이나 기계 ID)
  - 레이블 별 데이터의 다양성에 따라 학습할 수 있는 정보량이 다름
  - Overfitting, over-confidence의 위험이 상존함

마침

DNN에 어떤 창의적인 문제를 출제할 것인가?