About our Project

(feat. Listen, Attend and Spell)

Winter Vacation Capstone Study

TEAM Kai.Lib

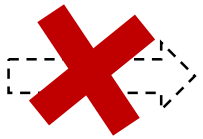
발표자 : 김수환

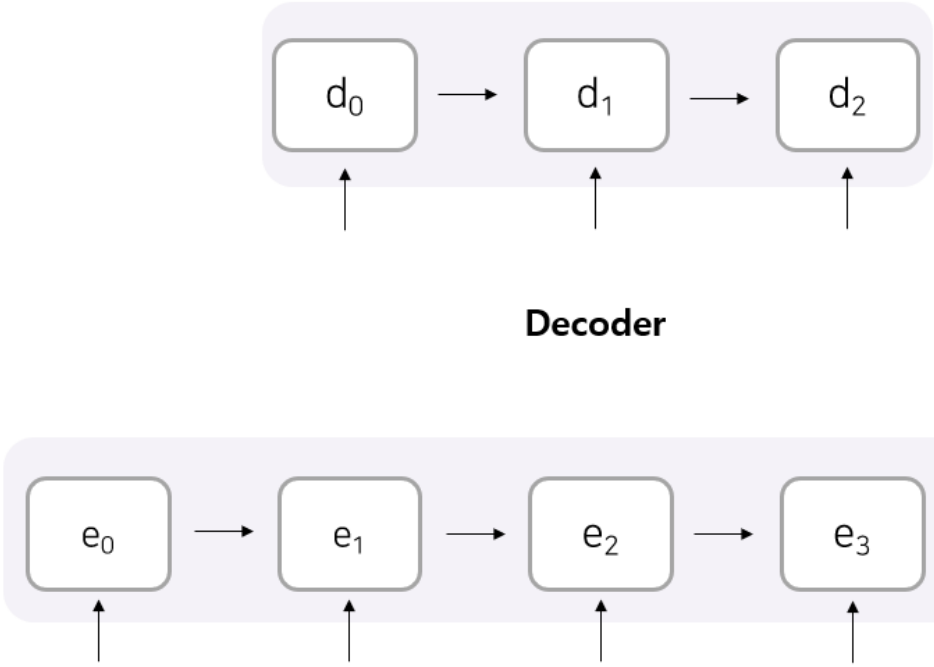
2020.02.03 (MON)

내 머리속 Debug

- 잘못 이해하고 있던 점

```python
def __init__(self):
    self.use_bidirectional = True
    self.use_attention = True
    self.input_reverse = True
    self.use_augmentation = True
    self.use_pickle = True
    self.augment_ratio = 0.3
    self.hidden_size = 256
    self.dropout = 0.5
    self.encoder_layer_size = 5
    self.decoder_layer_size = 3
    self.batch_size = 6
    self.worker_num = 1
    self.max_epochs = 40
    self.lr = 0.0001
    self.teacher_forcing = 0.99
    self.seed = 1
    self.max_len = 80
    self.no_cuda = False
    self.save_name = 'model'
    self.mode = 'train'
    self.load_model = False
    self.model_path = './weight_file/epoch2'
```

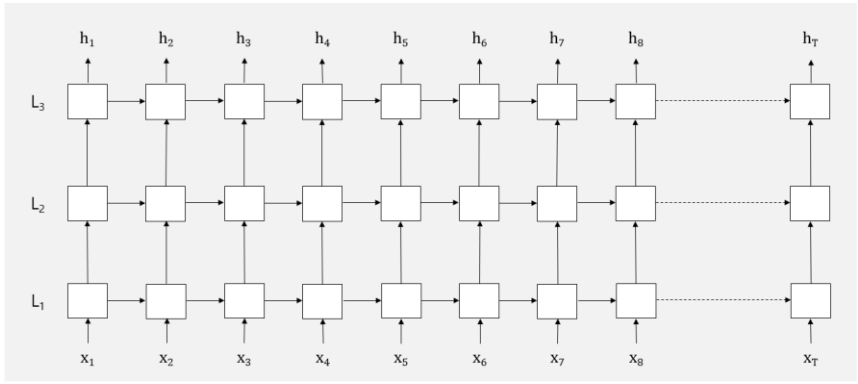




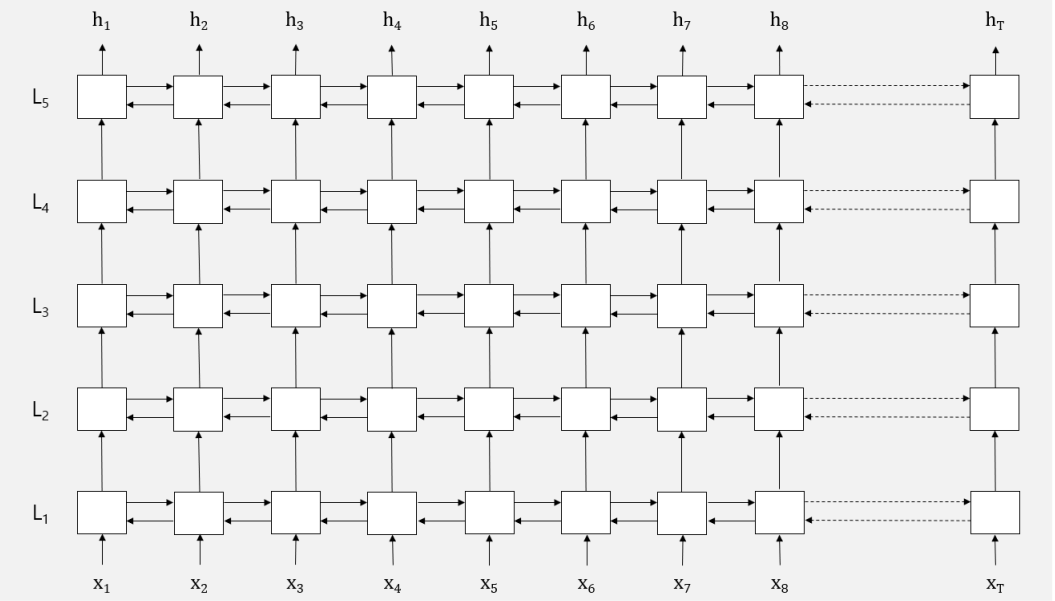
내 머리속 Debug

- 잘못 이해하고 있던 점

```
def __init__(self):
    self.use_bidirectional = True
    self.use_attention = True
    self.input_reverse = True
    self.use_augmentation = True
    self.use_pickle = True
    self.augment_ratio = 0.3
    self.hidden_size = 256
    self.dropout = 0.5
    self.encoder_layer_size = 5
    self.decoder_layer_size = 3
    self.batch_size = 6
    self.worker_num = 1
    self.max_epochs = 40
    self.lr = 0.0001
    self.teacher_forcing = 0.99
    self.seed = 1
    self.max_len = 80
    self.no_cuda = False
    self.save_name = 'model'
    self.mode = 'train'
    self.load_model = False
    self.model_path = './weight_file/epoch2'
```



Decoder RNN Layer



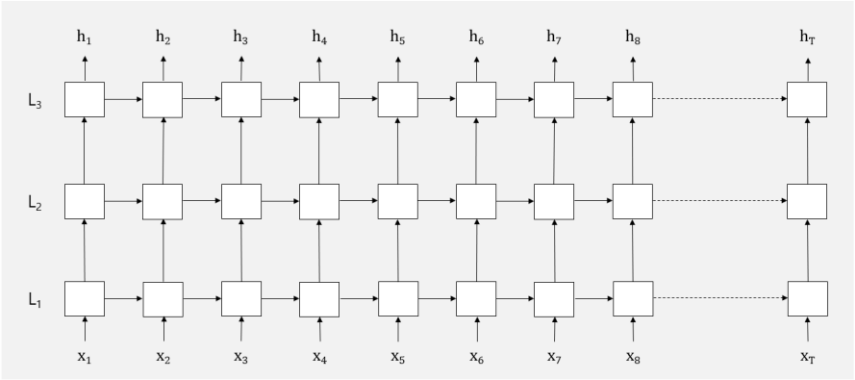
Encoder RNN Layer

내 머리속 Debug

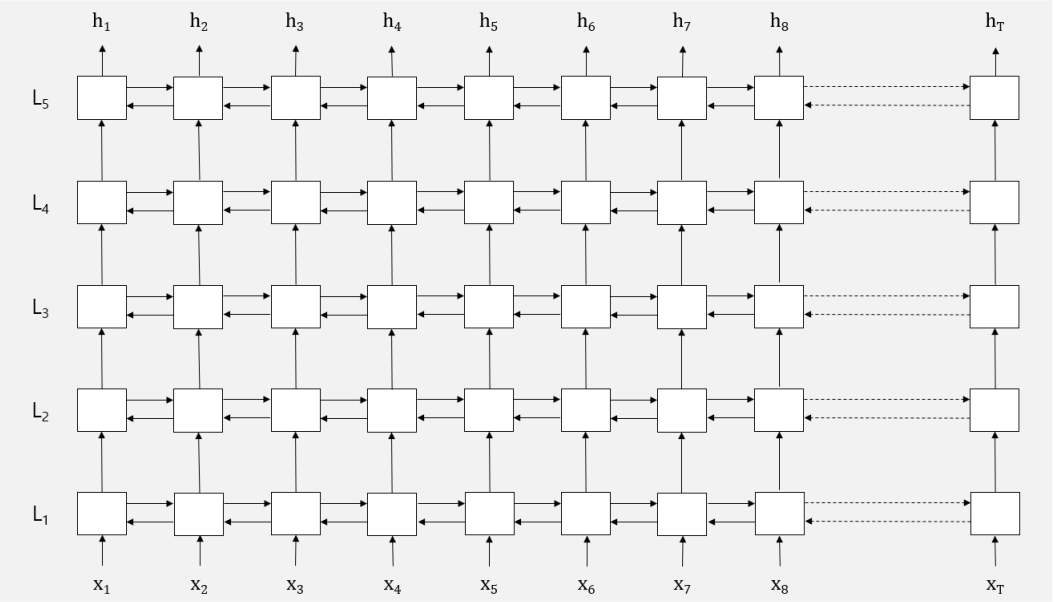
- 잘못 이해하고 있던 점

```
def __init__(self):
    self.use_bidirectional = True
    self.use_attention = True
    self.input_reverse = True
    self.use_augmentation = True
    self.use_pickle = True
    self.augment_ratio = 0.3
    self.hidden_size = 256
    self.dropout = 0.5
    self.encoder_layer_size = 5
    self.decoder_layer_size = 3
    self.batch_size = 6
    self.worker_num = 1
    self.max_epochs = 40
    self.lr = 0.0001
    self.teacher_forcing = 0.99
    self.seed = 1
    self.max_len = 80
    self.no_cuda = False
    self.save_name = 'model'
    self.mode = 'train'
    self.load_model = False
    self.model_path = './weight_file/epoch2'
```

상당히 깊은 구조였다...



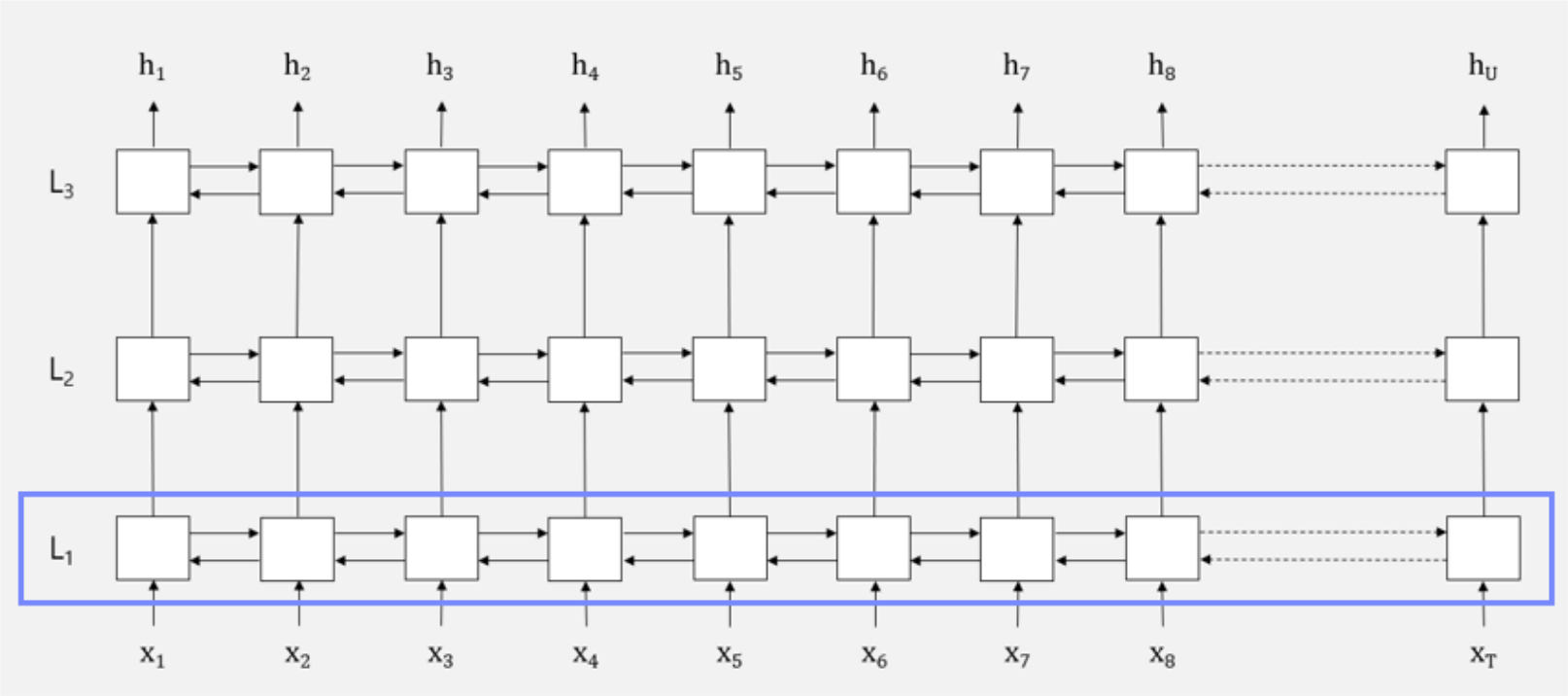
Decoder RNN Layer

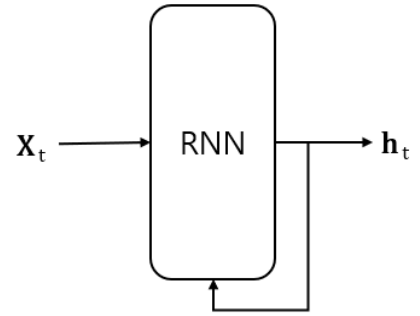


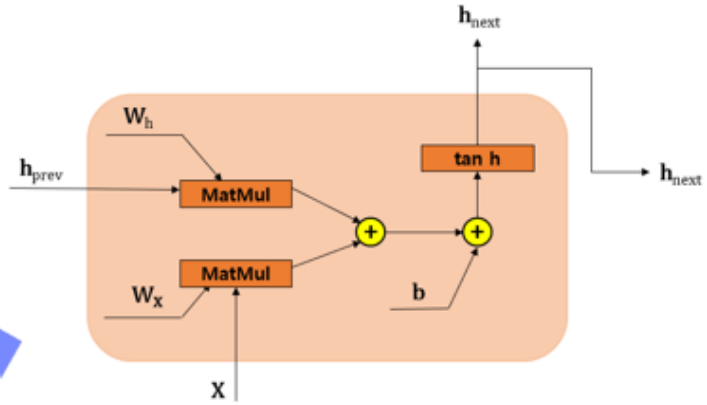
Encoder RNN Layer

다시 정리

- **RNN Layer 개념 다시 정리**



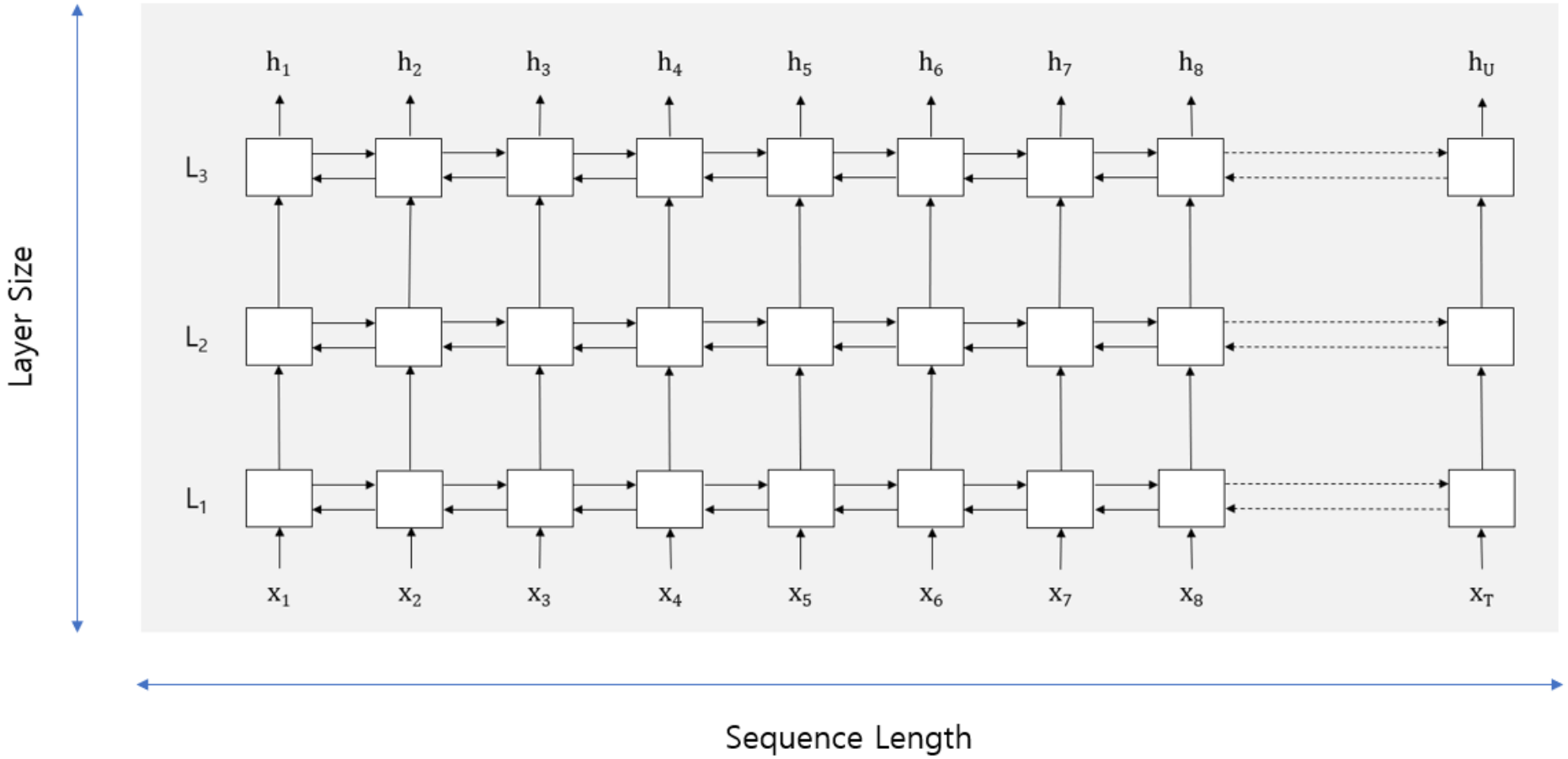




W_h, W_x는 고정된 채로,
X와 h_{prev}만 바뀌면서 Hidden State를 생성
(입력이 끝날 때까지 반복)
(가변 길이의 입력이 가능한 이유)

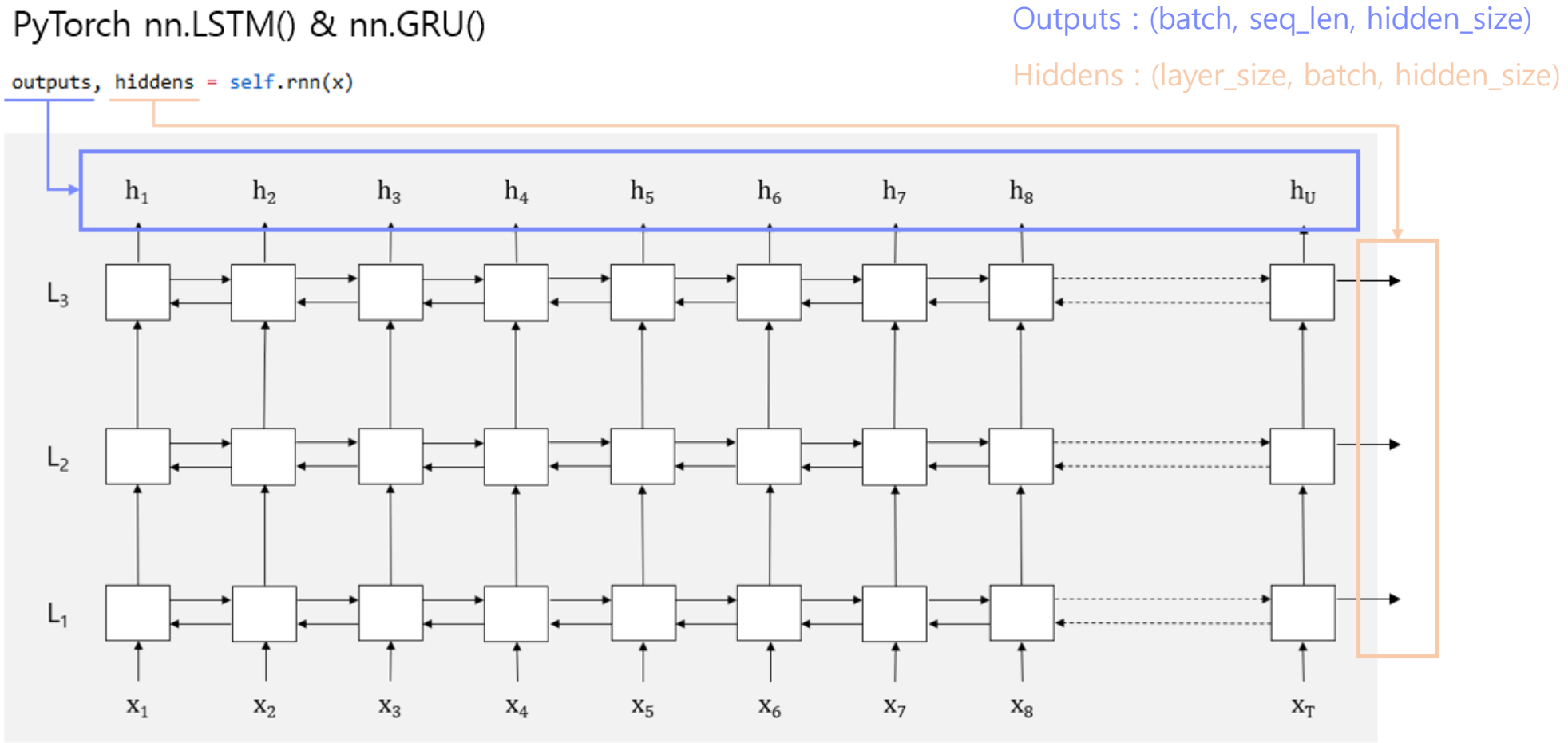
다시 정리

- **layer_size & seq_len**

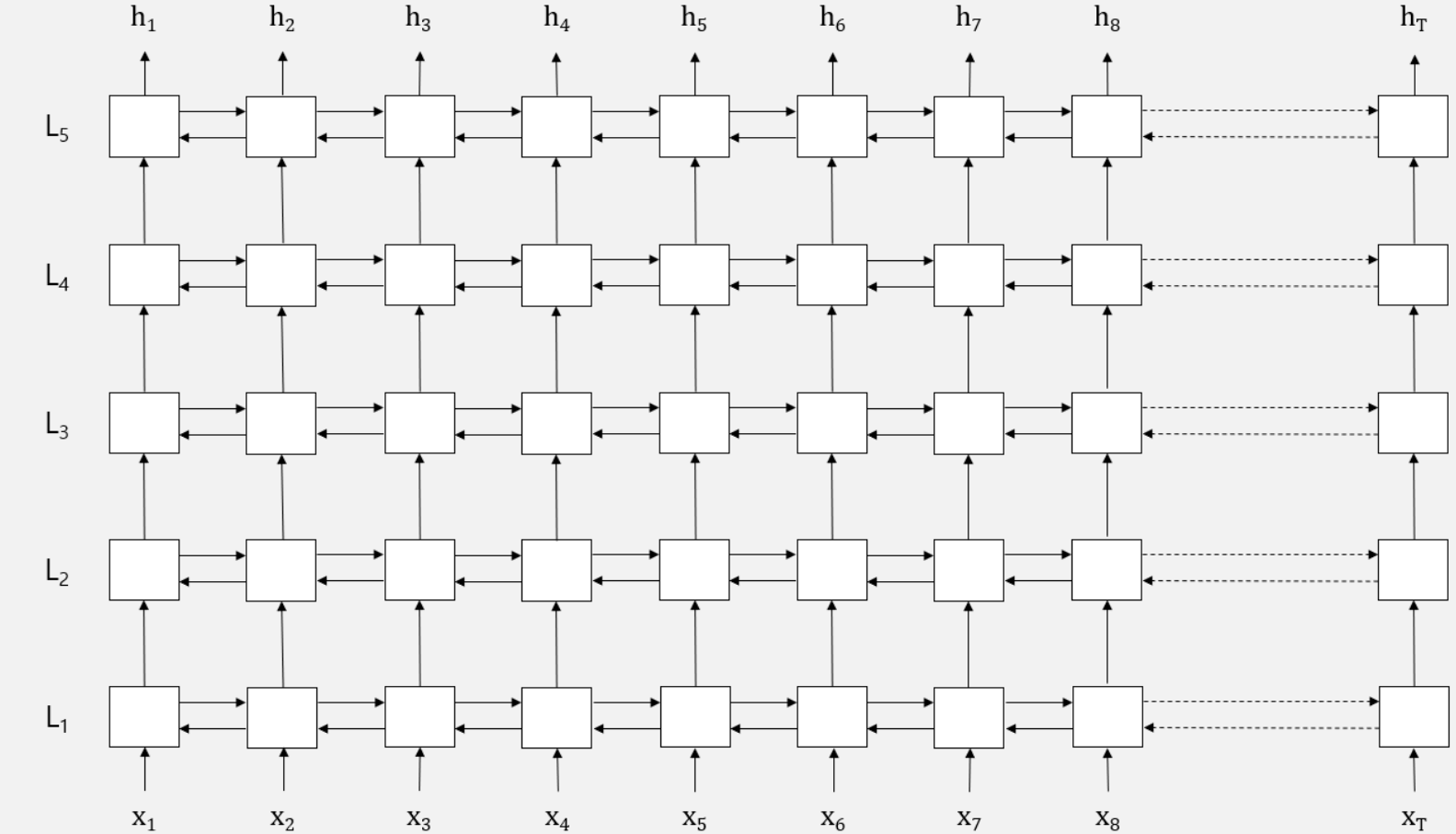


다시 정리

- **PyTorch nn.LSTM() & nn.GRU()**

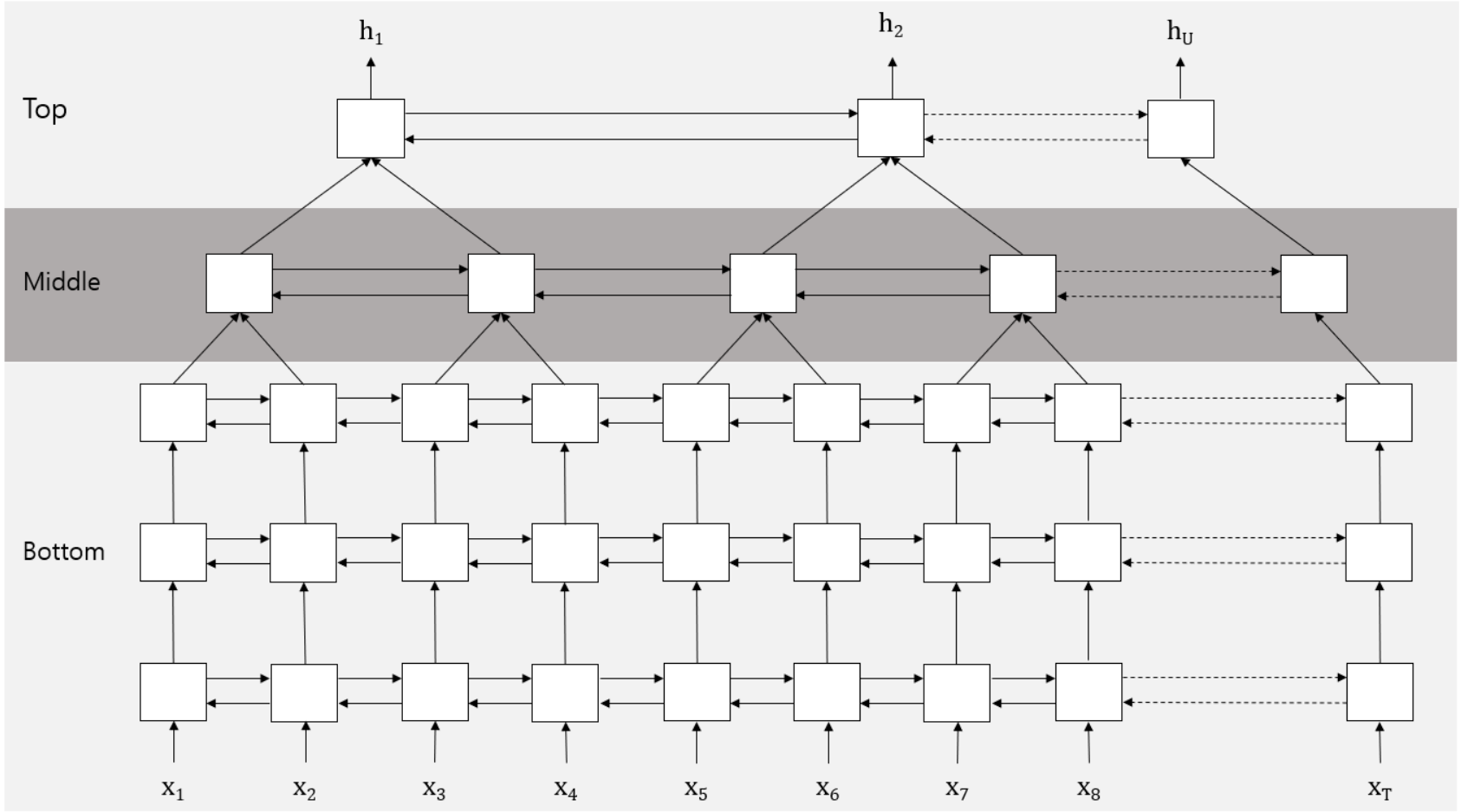


Proposal



:: use_pyramidal == False

Proposal



:: use_pyramidal == True

Proposal

- **Implement**

```
if use_pyramidal:
    self.bottom_layer_size = layer_size - 2
    self.bottom_rnn = self.rnn_cell(feature_size, hidden_size, self.bottom_layer_size, batch_first=True, bidirectional=bidirectional, dropout=dropout_p)
    self.middle_rnn = self.rnn_cell(hidden_size * 4, hidden_size, 1, batch_first=True, bidirectional=bidirectional, dropout=dropout_p)
    self.top_rnn = self.rnn_cell(hidden_size * 4, hidden_size, 1, batch_first=True, bidirectional=bidirectional, dropout=dropout_p)
else:
    self.rnn = self.rnn_cell(feature_size, hidden_size, layer_size, batch_first=True, bidirectional=bidirectional, dropout=dropout_p)
```

[use_pyrimidal시, bottom, middle, top으로 RNN 셀을 나누어 생성]

```
if self.use_pyramidal:
    bottom_outputs, _ = self.bottom_rnn(x)
    middle_inputs = self._make_pyramid(bottom_outputs)
    middle_outputs, _ = self.middle_rnn(middle_inputs)
    top_inputs = self._make_pyramid(middle_outputs)
    outputs, hiddens = self.top_rnn(top_inputs)
```

[Bottom → Middle → Top → outputs, hiddens]

```
def _cat_consecutive(self, prev_layer_outputs):
    """concatenate the outputs at consecutive setps of  each layer before feeding it to the next layer"""
    if prev_layer_outputs.size(1) % 2:
        """if prev_layer_outputs`s seq_len is odd, concatenate zeros"""
        zeros = torch.zeros((prev_layer_outputs.size(0), 1, prev_layer_outputs.size(2)))
        prev_layer_outputs = torch.cat([prev_layer_outputs, zeros], 1)
    return torch.cat([prev_layer_outputs[:, 0::2], prev_layer_outputs[:, 1::2]], 2)
```

[concatenate 2 layer]

그 외 진행사항

- **SpecAugment 적용 코드**

```
train_dataset_list.append(BaseDataset(audio_paths=audio_paths[train_begin_idx:train_end_idx],
                                      label_paths=label_paths[train_begin_idx:train_end_idx],
                                      bos_id=SOS_token, eos_id=EOS_token, target_dict=target_dict,
                                      input_reverse=hparams.input_reverse, use_augment=hparams.use_augment))
```

[BaseDataset 생성시 augment 여부 옵션으로 설정]

audio_paths	label_paths	is_augment
KaiSpeech_012358.pcm	KaiSpeech_012358.txt	False
KaiSpeech_524365.pcm	KaiSpeech_524365.txt	True
KaiSpeech_215678.pcm	KaiSpeech_215678.txt	False
KaiSpeech_012358.pcm	KaiSpeech_012358.txt	True

[멤버변수 is_augment[idx]가 True면 오그멘테이션 적용하도록 설정]
전체 데이터셋은 False로 해두고, 전체중 augment_ratio만큼
골라낸 다음 is_augment를 True로 설정

```
def get_item(self, idx):
    label = get_label(self.label_paths[idx], self.bos_id, self.eos_id, self.target_dict)
    feat = get_librosa_mfcc(self.audio_paths[idx], n_mfcc=33, del_silence=False, input_r
    # exception handling
    if feat.size(0) == 1:
        logger.info("Delete label_paths : %s" % self.label_paths[idx])
        label = ''
        return feat, label
    if self.is_augment[idx]:
        feat = spec_augment(feat, T=40, F=15, time_mask_num=2, freq_mask_num=2)
    return feat, label
```

is_augment[idx] == True면 spec_augment 적용

그 외 진행사항

- **test.py**

  모델 성능 테스트를 할 수 있는 코드 작성 (기존 테스트 코드는 nsml상에서 테스트를 하는 코드였음)

  => test.py : https://github.com/sh951011/Korean-Speech-Recognition/blob/master/test.py

  학습시 : main.py 실행

  테스트시 : test.py 실행