

# LSTM 예제 및 실습

# 수업목표

- 예제를 통한 MLP 와 LSTM 비교

# Example

- 나비야 음계 학습

나 비 야

The musical notation for the Na Bi Ya scale is presented in four lines, each containing four measures. The notes are written in treble clef with a 2/4 time signature. The notes are labeled with their letter names and octave numbers (e.g., g8, e8, e4, f8, d8, d4, c8, d8, e8, f8, g8, g8, g4).

Line 1: g8 e8 e4 f8 d8 d4 c8 d8 e8 f8 g8 g8 g4

Line 2: g8 e8 e8 e8 f8 d8 d4 c8 e8 g8 g8 e8 e8 e4

Line 3: d8 d8 d8 d8 d8 e8 f4 e8 e8 e8 e8 e8 f8 g4

Line 4: g8 e8 e4 f8 d8 d4 c8 e8 g8 g8 e8 e8 e4

\* 김태영의 케라스 블로그에서 발췌

# Example

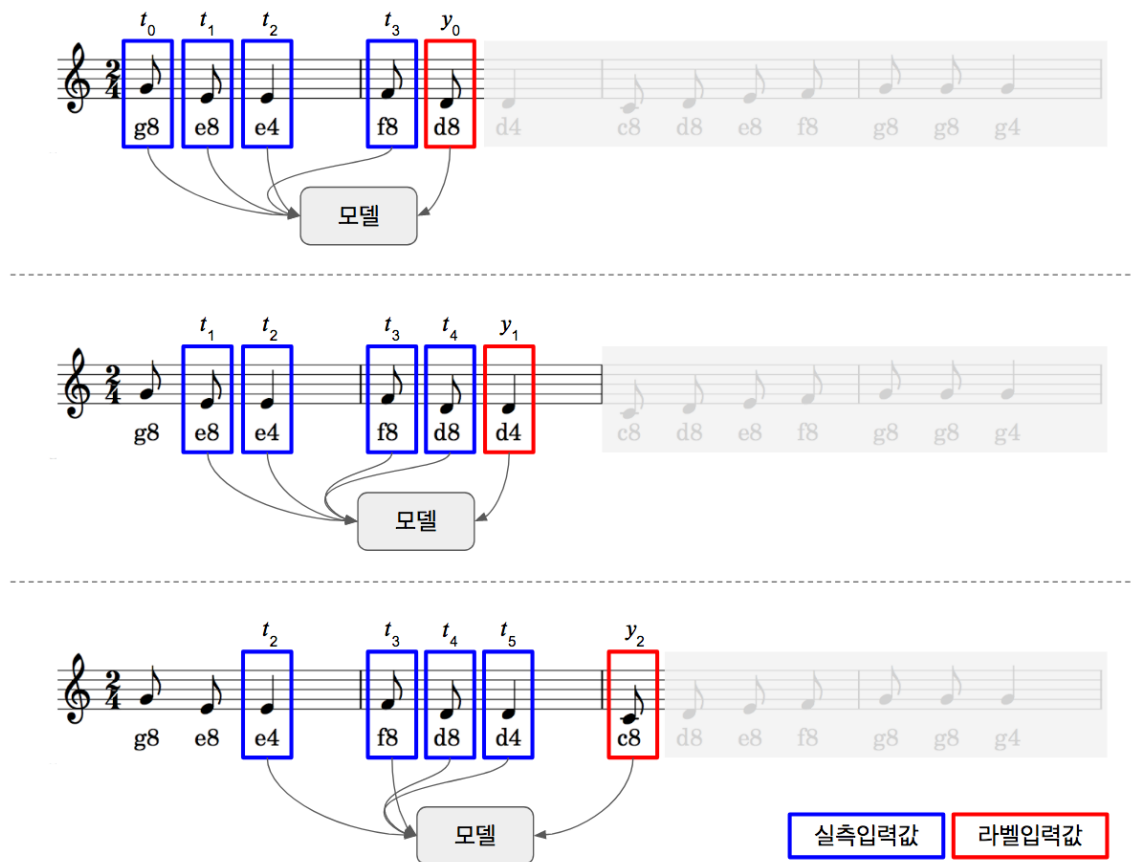
## - 나비야 음계 학습

- 코드화
- 시계열 자료(sequential)
- 결과 확인이 쉬움(악보, 연주)
- 도(C), 레(D), 미(E), 파(F), 솔(G), 라(A), 시(B) 7음계
- 4분 음표(4), 8분 음표(8)

# MLP

# MLP

## 1. Overview



# MLP

## 2. Import library

- `import keras`
- `import keras.models as Sequential`
- `import keras.layers as Dense`
- `import keras.utils as np_utils`
- `import numpy as np`

# MLP

## 3. 손실 이력 클래스 정의

```
1 # 손실 이력 클래스 정의
2 class LossHistory(keras.callbacks.Callback):
3     def init(self):
4         self.losses = []
5
6     def on_epoch_end(self, batch, logs={}):
7         self.losses.append(logs.get('loss'))
```



# MLP

## 4. 데이터셋 생성 함수

```
1 # 데이터셋 생성 함수
2 def seq2dataset(seq, window_size):
3     dataset = []
4     for i in range(len(seq)-window_size):
5         subset = seq[i:(i+window_size+1)]
6         dataset.append([note2idx[item] for item in subset])
7     return np.array(dataset)
```

# MLP

## 4. 데이터셋 생성 함수



A musical score for the song "나 비 야" (Na Bi Ya) in 2/4 time. The score consists of four staves, each containing eight notes. The notes are labeled with pitch classes and octave numbers (e.g., g8, e8, e4, f8, d8, d4). A red box highlights the first six notes of the first staff: g8, e8, e4, f8, d8, and d4. A blue arrow points from the text "4. 데이터셋 생성 함수" to the red box.

나 비 야

g8 e8 e4 f8 d8 d4 c8 d8 e8 f8 g8 g8 g4

5 g8 e8 e8 e8 f8 d8 d4 c8 e8 g8 g8 e8 e8 e4

9 d8 d8 d8 d8 d8 e8 f4 e8 e8 e8 e8 e8 f8 g4

13 g8 e8 e4 f8 d8 d4 c8 e8 g8 g8 e8 e8 e4

# MLP

## 4. 음계 및 인덱스 사전(dictionary) 정의

```
1 # 2. 데이터 준비하기(note, index)
2
3 # 음계 및 인덱스 사전 정의
4
5 note2idx = {'c4':0, 'd4':1, 'e4':2, 'f4':3, 'g4':4, 'a4':5, 'b4':6,
6            'c8':7, 'd8':8, 'e8':9, 'f8':10, 'g8':11, 'a8':12, 'b8':13}
7
8 idx2note = {0:'c4', 1:'d4', 2:'e4', 3:'f4', 4:'g4', 5:'a4', 6:'b4',
9            7:'c8', 8:'d8', 9:'e8', 10:'f8', 11:'g8', 12:'a8', 13:'b8'}
10
11 # 시퀀스 데이터 정의
12
13 seq = ['g8', 'e8', 'e4', 'f8', 'd8', 'd4', 'c8', 'd8', 'e8', 'f8', 'g8', 'g8', 'g4',
14        'g8', 'e8', 'e8', 'e8', 'f8', 'd8', 'd4', 'c8', 'e8', 'g8', 'g8', 'e8', 'e8', 'e4',
15        'd8', 'd8', 'd8', 'd8', 'd8', 'e8', 'f4', 'e8', 'e8', 'e8', 'e8', 'e8', 'f8', 'g4',
16        'g8', 'e8', 'e4', 'f8', 'd8', 'd4', 'c8', 'e8', 'g8', 'g8', 'e8', 'e8', 'e4']
```

# MLP

## 6. 데이터셋 생성

```
1 # 3. 데이터셋 생성하기
2 dataset = seq2dataset(seq, window_size = 4)
3
4 print(dataset.shape)
5 print(dataset)
6
7 # 입력(X)과 출력(Y) 변수로 분리하기
8 x_train = dataset[:,0:4]
9 y_train = dataset[:,4]
10
11 max_idx_value = 13
12
13 # 입력값 정규화 시키기
14 x_train = x_train / float(max_idx_value)
15
16 # 라벨값에 대한 one-hot 인코딩 수행
17 y_train = np_utils.to_categorical(y_train)
18
19 one_hot_vec_size = y_train.shape[1]
20
21 print("one hot encoding vector size is ", one_hot_vec_size)
```

# MLP

## 6. 데이터셋 생성

### to\_categorical

```
keras.utils.to_categorical(y, num_classes=None)
```

Converts a class vector (integers) to binary class matrix.

E.g. for use with categorical\_crossentropy.

#### Arguments

- **y**: class vector to be converted into a matrix (integers from 0 to num\_classes).
- **num\_classes**: total number of classes.

#### Returns

A binary matrix representation of the input. The classes axis is placed last.

# MLP

## 7. 모델 구성

```
1 # 4. 모델 구성하기
2 model = Sequential()
3 model.add(Dense(128, input_dim=4, activation='relu'))
4 model.add(Dense(128, activation='relu'))
5 model.add(Dense(one_hot_vec_size, activation='softmax'))
```

# MLP

## 8. 학습 과정 설정

```
1 # 5. 모델 학습과정 설정하기
2 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
3
4 history = LossHistory() # 손실 이력 객체 생성
5 history.init()
```

# MLP

## 9. 학습

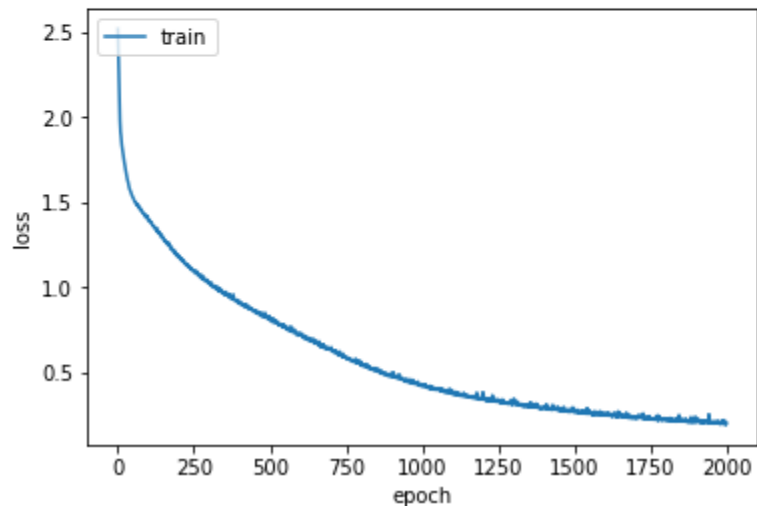
```
1 # 6. 모델 학습시키기  
2 model.fit(x_train, y_train, epochs=2000, batch_size=10, verbose=2, callbacks=[history])
```



# MLP

## 10. 학습 과정 확인

```
1 # 7. 학습과정 살펴보기
2 %matplotlib inline
3 import matplotlib.pyplot as plt
4
5 plt.plot(history.losses)
6 plt.ylabel('loss')
7 plt.xlabel('epoch')
8 plt.legend(['train'], loc='upper left')
9 plt.show()
```



# MLP

## 11. 평가

```
1 # 8. 모델 평가하기
2 scores = model.evaluate(x_train, y_train)
3 print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

50/50 [=====] - 0s 2ms/step .....  
acc: 92.00%

# MLP

## 12. 결과 확인

```
1 # 9. 모델 사용하기
2
3 pred_count = 50 # 최대 예측 개수 정의
4
5 # 한 스텝 예측
6
7 seq_out = ['g8', 'e8', 'e4', 'f8']
8 pred_out = model.predict(x_train)
9
10 for i in range(pred_count):
11     idx = np.argmax(pred_out[i]) # one-hot 인코딩을 인덱스 값으로 변환
12     seq_out.append(idx2note[idx]) # seq_out는 최종 악보이므로 인덱스 값을 코드로 변환하여 저장
13
14
15 # 결과를 저장할 변수 선언
16 note_onestep = seq_out
17
18 # 곡 전체 예측
19
20 seq_in = ['g8', 'e8', 'e4', 'f8']
21 seq_out = seq_in
22 seq_in = [note2idx[it] / float(max_idx_value) for it in seq_in] # 코드를 인덱스값으로 변환
23
24
25 for i in range(pred_count):
26     sample_in = np.array(seq_in)
27     sample_in = np.reshape(sample_in, (1, 4)) # batch_size, feature
28     pred_out = model.predict(sample_in)
29     idx = np.argmax(pred_out)
30     seq_out.append(idx2note[idx])
31     seq_in.append(idx / float(max_idx_value))
32     seq_in.pop(0)
33
34
35 # 결과를 저장할 변수 선언
36 note_fullsong = seq_out
```

# MLP

## 12. 결과 확인 – one step prediction

실제 음표 4개를 순차적 입력

나 비 야

The musical score consists of four staves, each containing 12 notes. The notes are labeled with pitch names and octave numbers. The first four notes of the first staff are highlighted with a red box, and the fifth note is highlighted with a blue box. A blue arrow points from the text '실제 음표 4개를 순차적 입력' to the blue box.

Staff	Note 1	Note 2	Note 3	Note 4	Note 5	Note 6	Note 7	Note 8	Note 9	Note 10	Note 11	Note 12
1	g8	e8	e4	f8	d8	d4	c8	d8	e8	f8	g8	g8
5	g8	e8	e8	e8	f8	d8	d4	c8	e8	g8	g8	e8
9	d8	d8	d8	d8	d8	e8	f4	e8	e8	e8	e8	f8
13	g8	e8	e4	f8	d8	d4	c8	e8	g8	g8	e8	e8

# MLP

## 12. 결과 확인 – full song prediction

예측한 값을 토대로 다음 값을 예측

나 비 야

g8 e8 e4 f8 d8 d4 c8 d8 e8 f8 g8 g8 g4

5 g8 e8 e8 e8 f8 d8 d4 c8 e8 g8 g8 e8 e8 e4

9 d8 d8 d8 d8 d8 e8 f4 e8 e8 e8 e8 e8 f8 g4


13 g8 e8 e4 f8 d8 d4 c8 e8 g8 g8 e8 e8 e4

# MLP

## 12. 결과 확인

나비야

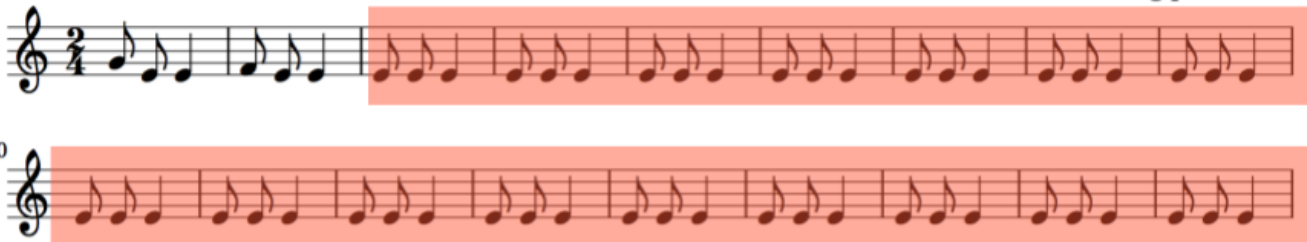
MLP one-step prediction



9

나비야

MLP full song prediction

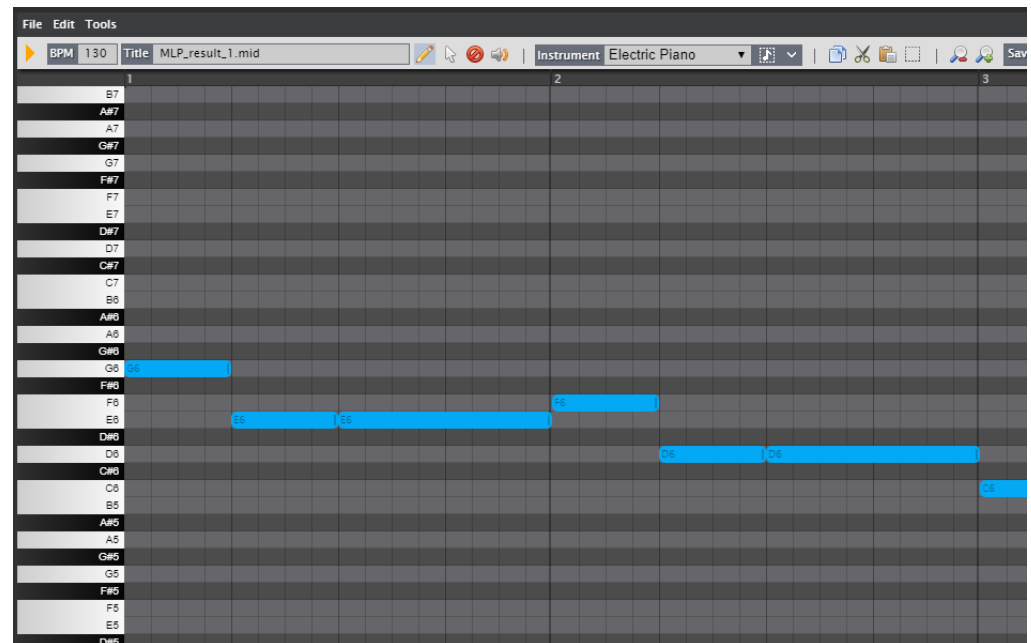
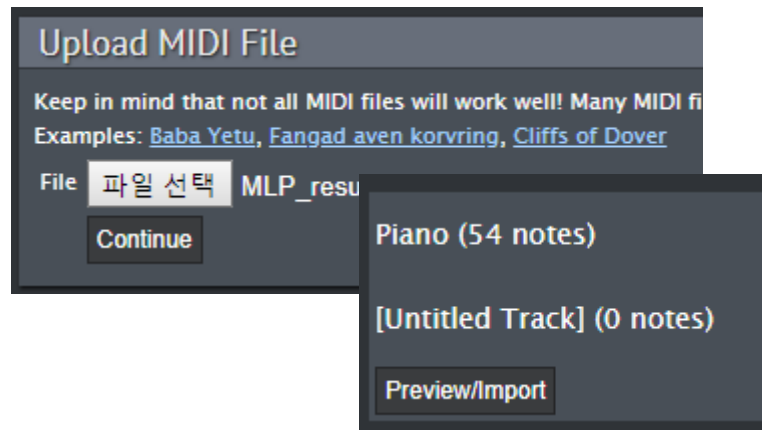


10

# MLP

## 12. 결과 확인 - 들어보기

- <https://onlinesequencer.net/import> 로 접속
- [파일 선택] → [Continue] → [Preview/Import]

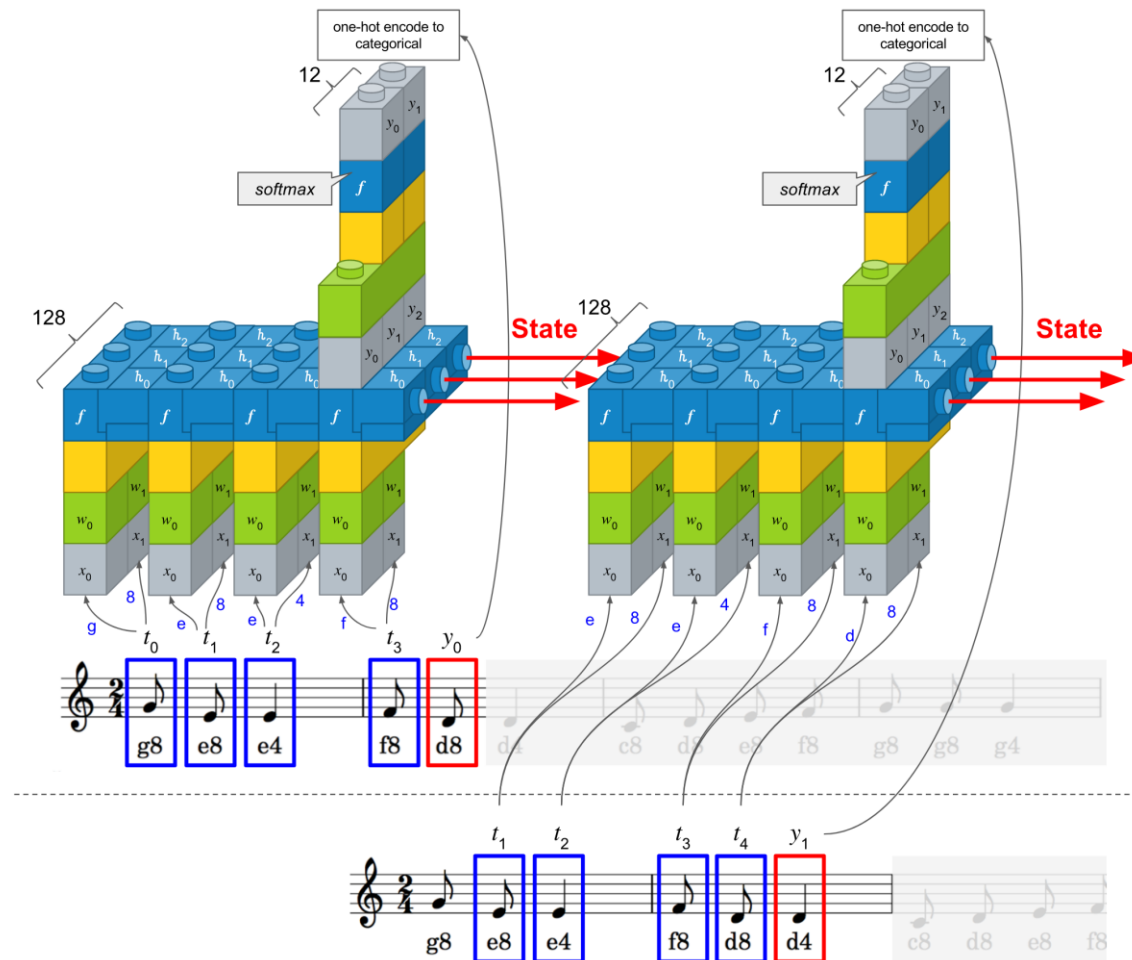


# LSTM



# LSTM

## 1. Overview



# LSTM

## 1. Overview

- 음계와 음표의 길이를 특성으로 취함
- 128개의 메모리 셀을 가진 LSTM layer 1개 + Dense layer
- 입력 샘플 50개 + 타임 스텝 4개 + 속성(특성) 2개

# LSTM

## 2. Import library

- `import keras`
- `import keras.models as Sequential`
- `import keras.layers as Dense`
- `import keras.layers as LSTM`
- `import keras.utils as np_utils`
- `import numpy as np`

# LSTM

## 3. 손실 이력 클래스 정의

```
1 # 손실 이력 클래스 정의
2 class LossHistory(keras.callbacks.Callback):
3     def init(self):
4         self.losses = []
5
6     def on_epoch_end(self, batch, logs={}):
7         self.losses.append(logs.get('loss'))
```

# MLP

## 4. 데이터셋 생성 함수

```
# 데이터셋 생성 함수
def seq2dataset(seq, window_size):
    dataset = []
    for i in range(len(seq)-window_size):
        subset = seq[i:(i+window_size+1)]
        dataset.append([code2idx[item] for item in subset])
    return np.array(dataset)
```

```
# 데이터셋 생성 함수
def seq2dataset(seq, window_size):
    dataset_X = []
    dataset_Y = []
    for i in range(len(seq)-window_size):
        subset = seq[i:(i+window_size+1)]
        for si in range(len(subset)-1):
            features = note2features(subset[si])
            dataset_X.append(features)
        dataset_Y.append([note2idx[subset[window_size]]])
    return np.array(dataset_X), np.array(dataset_Y)

# 속성 변환 함수
def note2features(note):
    features = []
    features.append(note2scale[note[0]]/float(max_scale_value))
    features.append(note2length[note[1]])
    return features
```

# MLP

## 5. 데이터 준비하기

*# 2. 데이터 준비하기*

*# 코드 사전 정의*

```
note2scale = {'c':0, 'd':1, 'e':2, 'f':3, 'g':4, 'a':5, 'b':6}  
note2length = {'4':0, '8':1}
```

```
note2idx = {'c4':0, 'd4':1, 'e4':2, 'f4':3, 'g4':4, 'a4':5, 'b4':6,  
           'c8':7, 'd8':8, 'e8':9, 'f8':10, 'g8':11, 'a8':12, 'b8':13}
```

```
idx2note = {0:'c4', 1:'d4', 2:'e4', 3:'f4', 4:'g4', 5:'a4', 6:'b4',  
            7:'c8', 8:'d8', 9:'e8', 10:'f8', 11:'g8', 12:'a8', 13:'b8'}
```

```
max_scale_value = 6.0
```

*# 시퀀스 데이터 정의*

```
seq = ['g8', 'e8', 'e4', 'f8', 'd8', 'd4', 'c8', 'd8', 'e8', 'f8', 'g8', 'g8', 'g4',  
       'g8', 'e8', 'e8', 'e8', 'f8', 'd8', 'd4', 'c8', 'e8', 'g8', 'g8', 'e8', 'e8', 'e4',  
       'd8', 'd8', 'd8', 'd8', 'd8', 'e8', 'f4', 'e8', 'e8', 'e8', 'e8', 'e8', 'f8', 'g4',  
       'g8', 'e8', 'e4', 'f8', 'd8', 'd4', 'c8', 'e8', 'g8', 'g8', 'e8', 'e8', 'e4']
```

# LSTM

## 6. 데이터 생성하기

```
# 3. 데이터셋 생성하기

x_train, y_train = seq2dataset(seq, window_size = 4)

# 입력을 (샘플 수, 타임스텝, 특성 수)로 형태 변환
x_train = np.reshape(x_train, (50, 4, 2))

# 라벨값에 대한 one-hot 인코딩 수행
y_train = np_utils.to_categorical(y_train)

one_hot_vec_size = y_train.shape[1]

print("one hot encoding vector size is ", one_hot_vec_size)
```

# LSTM

## 7. 모델 구성

```
# 4. 모델 구성하기
model = Sequential()
model.add(LSTM(128, batch_input_shape = (1, 4, 2), stateful=True))
model.add(Dense(one_hot_vec_size, activation='softmax'))
```



# LSTM

## 8. 학습 과정 설정

# 5. 모델 학습과정 설정하기

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

# LSTM

## 9. 학습

```
# 8. 모델 학습시키기
```

```
num_epochs = 2000
```

```
history = LossHistory() # 손실 이력 객체 생성
```

```
history.init()
```

```
for epoch_idx in range(num_epochs):
```

```
    print('epochs : ' + str(epoch_idx) )
```

```
    model.fit(x_train, y_train, epochs=1, batch_size=1, verbose=2, shuffle=False, callbacks=[history]) # 50 is A
```

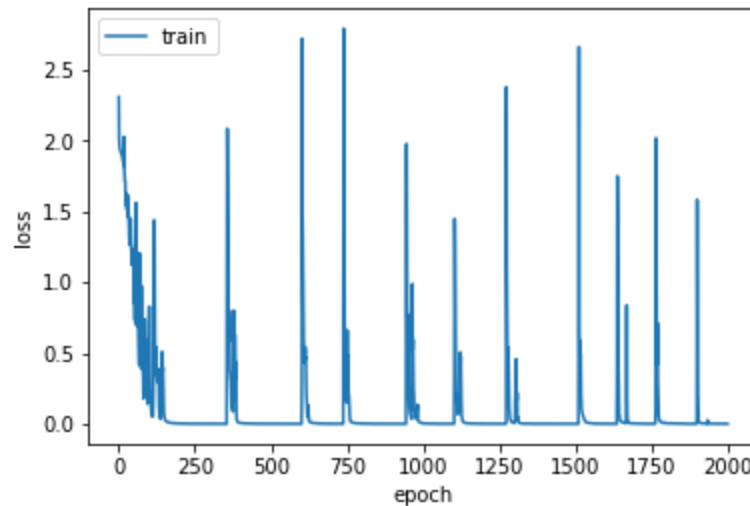
```
    model.reset_states()
```

# LSTM

## 10. 학습 과정 확인

```
# 7. 학습과정 살펴보기
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(history.losses)
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train'], loc='upper left')
plt.show()
```



# LSTM

## 11. 평가

```
# 8. 모델 평가하기
scores = model.evaluate(x_train, y_train, batch_size=1)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
model.reset_states()
```

50/50 [=====] - 0s 10ms/ste

100.00%

acc: 100.00%

# LSTM

## 12. 결과 확인

```
# 9. 모델 사용하기

pred_count = 50 # 최대 예측 개수 정의

# 한 스텝 예측

seq_out = ['g8', 'e8', 'e4', 'f8']
pred_out = model.predict(x_train, batch_size=1)

for i in range(pred_count):
    idx = np.argmax(pred_out[i]) # one-hot 인코딩을 인덱스 값으로 변환
    seq_out.append(idx2note[idx]) # seq_out는 최종 악보이므로 인덱스 값을 코드로 변환하여 저장

note_onestep = seq_out

model.reset_states()

# 곡 전체 예측

seq_in = ['g8', 'e8', 'e4', 'f8']
seq_out = seq_in

seq_in_feats = []

for si in seq_in:
    features = note2features(si)
    seq_in_feats.append(features)

for i in range(pred_count):
    sample_in = np.array(seq_in_feats)
    sample_in = np.reshape(sample_in, (1, 4, 2)) # 샘플 수, 타임스텝 수, 특성 수
    pred_out = model.predict(sample_in)
    idx = np.argmax(pred_out)
    seq_out.append(idx2note[idx])

    features = note2features(idx2note[idx])
    seq_in_feats.append(features)
    seq_in_feats.pop(0)

model.reset_states()

note_fullsong = seq_out
```

# LSTM

## 12. 결과 확인

나비야

Stateful LSTM two features one-step prediction



나비야

Stateful LSTM two features full song prediction

