Digital Processing of Speech Signals

Yoon Joong Kim
Hanbat National University

Music21과 심층신경망 피아노연주기

Music21 사용법
DNN(심층신경망) 피아노연주기

Yoonjoong Kim

Department of Computer Engineering, Hanbat National University

yjkim@hanbat.ac.kr

yjkim@hanbat.ac.kr

yjkim@hanbat.ac.kr

Contents

1.1 Music21 패키지의 소개

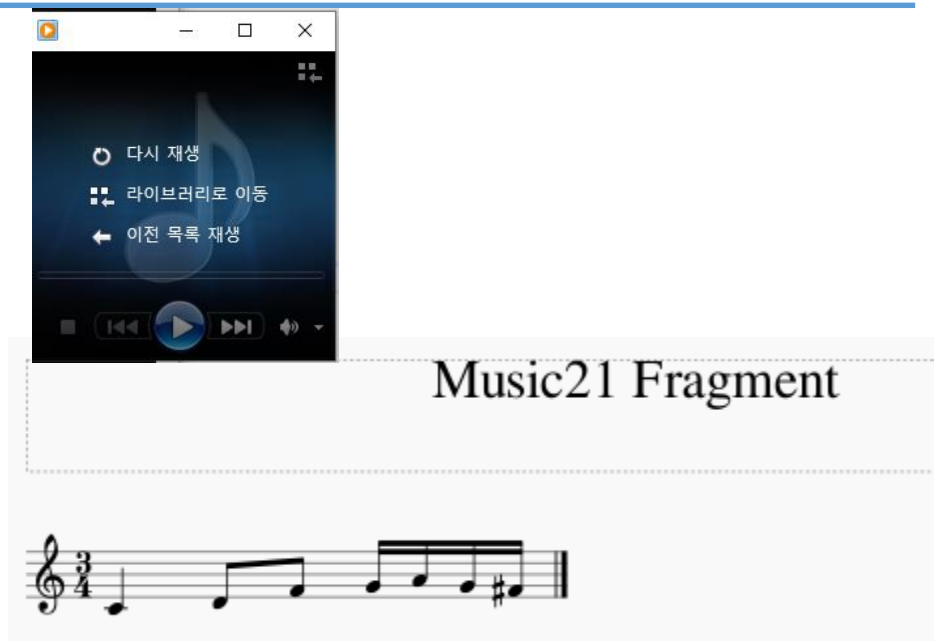
- 1.1 Music21 패키지의 소개
  - Music21은 음악에 대한 질문에 빠르고 간단하게 답변 할 수 있도록 학자 등을 도와주는 도구 모음입니다.
  - 여기서는 음악을 연주하는 방법을 분석하고 인공지능으로 피아노를 연주 하는 방법을 학습한다.
  - Music 21의 MIT에서 시작 된 프로젝트로 시작되었다. MIT의 음악 학과는 Harvard, Smith 및 Mount Holyoke Colleges와 함께 이 툴킷을 개발하고 발전시켜왔다.
  - 이 시스템은 2008 년부터 사용되어 왔으며 지속적으로 성장하고 확장되고 있습니다. music21의 접근과 전통은 많은 이전 소프트웨어 시스템에서 사용되었습니다.
- Web site
  - http://web.mit.edu/music21/

1.2. Music 21 환경설정 설정

- Music 21 환경설정 설정
  - music 21 설치
    - pip install music21
  - 5선지 악보 패키지 설치
    - #xml 악보가 아닌 오선지 악보를 이용하기 위하여 MusicScore 패키지를 다운로드 받아서 설치한다.
    - https://musescore.org 에서 FreeDownload 클릭
    - MuseScore-3.5.2.3.. msi 실행

1.3. Music21 맛보기

```
import numpy as np
from music21 import *     #pip install music21

#xml 악보가 아닌 오선지 악보를 이용하기 위하여 MusicScore 패키지를 다운로드 받아서 설치한다.
# https://musescore.org 에서   FreeDownload
# MuseScore-3.5.2.3.. msi  실행

# path 설정 한번 만 샐행
us=environment.UserSettings()
us["musescoreDirectPNGPath"]=r"C:\Program Files\MuseScore 3\bin\MuseScore3.exe"
us["musicxmlPath"]=r"C:\Program Files\MuseScore 3\bin\MuseScore3.exe"

s=converter.parse("tinynotation: 3/4 c4 d8 f g16 a g f#") #<music21.stream.Part 0x21cacfd32c8>
s.show('midi', fp='sample.mid')          #음악재생 및 sample.mid로 저장
s.show()                                 #오선악보 출력
```

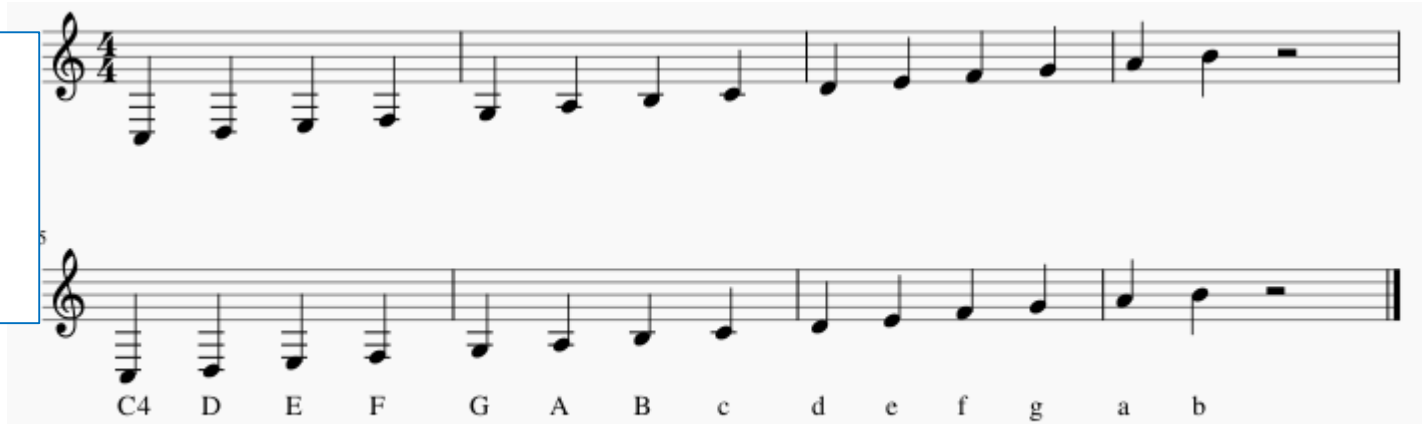



```
p=converter.parse("tinynotation: 3/4 c4 d8 f g16 a g f#") #<music21.stream.Part 0x21cacfd3cc8>
for n in list(p.flat.notes) :   #음표출력
      print("Note: {}{} {:0.1f}".format(n.pitch.name, n.pitch.octave, n.duration.quarterLength))
#Note: C4 1.0
#Note: D4 0.5
#Note: F4 0.5
#Note: G4 0.2
#Note: A4 0.2
#Note: G4 0.2
#Note: F#4 0.2
```

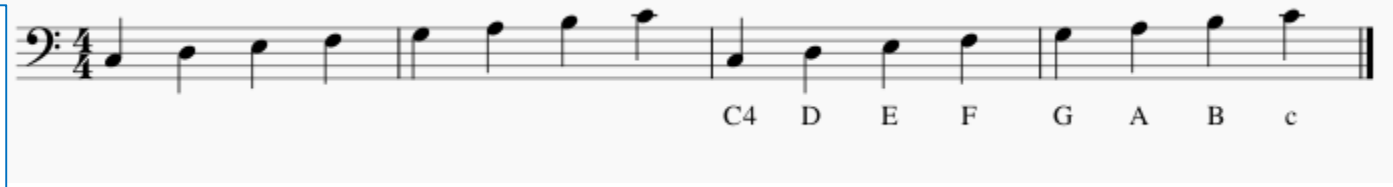
yjkim@hanbat.ac.kr

1.4. tinyNotation

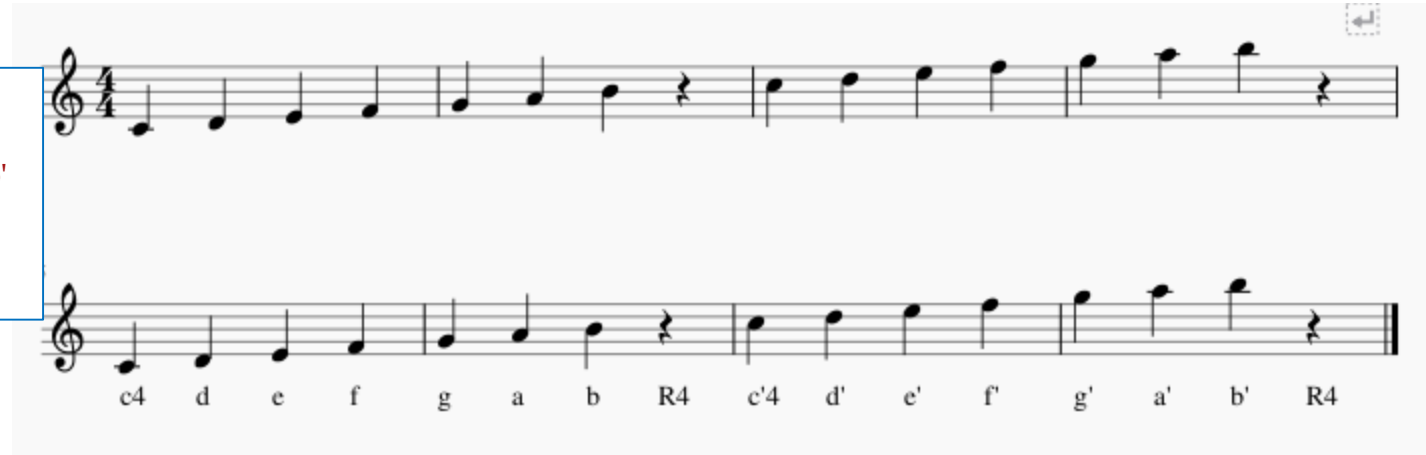
```
from music21 import *

c= converter.parse("""tinynotation: 4/4
C4 D E F   G A B c   d e f g   a b r2
C4_C4 D_D E_E F_F G_G A_A B_B c_c d_d e_e f_f g_g a_a b_b r2
""")
c.show()
```




```
c= converter.parse("""tinynotation: 4/4
C4 D E F   G A B c
C4_C4 D_D E_E F_F G_G A_A B_B c_c
""")
c.show()
```

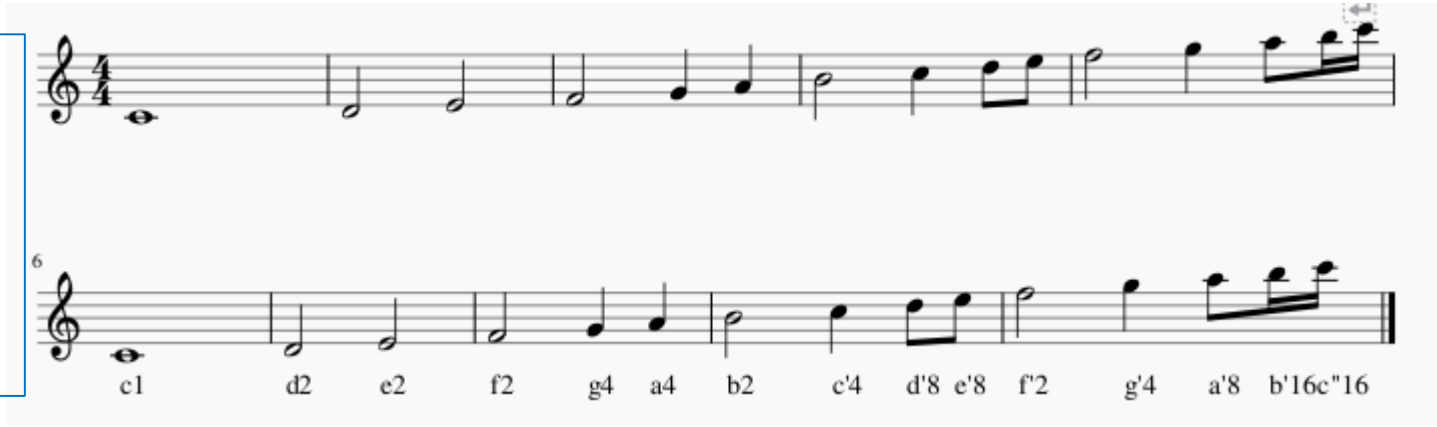



```
c= converter.parse("""tinynotation: 4/4
c4 d e f g a b r4                    c'4 d' e' f' g' a' b' r4
c4_c4 d_d e_e f_f g_g a_a b_b r4_R4          c'4_c'4 d'_d' e'_e' f'_f' g'_g' a'_a' b'_b'
r4_R4
""")
c.show()
```



1.4. tinyNotation(cont.)

```
c= converter.parse("""tinynotation: 4/4
c1d2 e2f2 g4 a4   b2 c'4 d'8 e'8   f'2  g'4 a'8 b'16  c"16
c1_c1d2_d2 e2_e2f2_f2 g4_g4 a4_a4   b2_b2 c'4_c'4 d'8_d'8 e'8_e'8   f'2_f'2
g'4_g'4 a'8_a'8 b'16_b'16  c"16_c"16
""")
c.show()
```



yjkim@hanbat.ac.kr

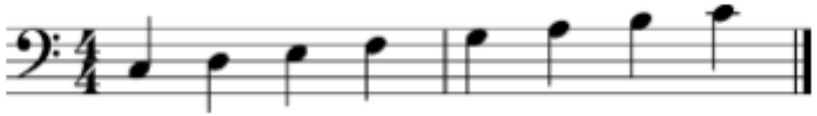
1.4. tinyNotation(cont.)

Here's a bunch of quarter notes in 4/4:

Notice that the last "c" is lowercase, while the rest of the notes are uppercase. Case determines octave: "C" = the c in bass clef (C3) while "c" = middle C (C4). Here are some other octaves:

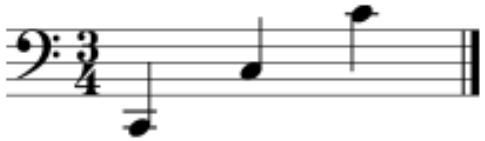
Here's a bunch of quarter notes in 4/4:

```
s = converter.parse('tinyNotation: 4/4 C4 D4 E4 F4 G4 A4 B4 c4')
s.show()
```

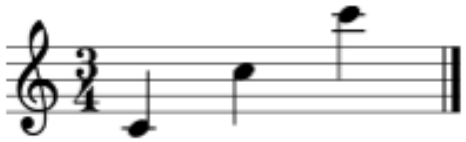


Notice that the last "c" is lowercase, while the rest of the notes are uppercase. in bass clef (C3) while "c" = middle C (C4). Here are some other octaves:

```
s = converter.parse("tinyNotation: 3/4 CC4 C4 c4")
s.show()
```



```
s = converter.parse("tinyNotation: 3/4 c4 c'4 c''4")
s.show()
```



1.4. tinyNotation(cont.)

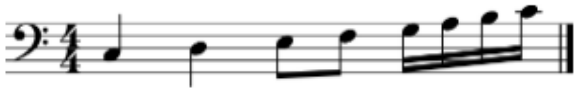
And, yes, CCC is the C below CC, and c''' is the c above c''. Remember when you use higher notes to make sure to enclose your string in double quotes, not single quotes.
Typing all those "4"s for each of the quarter notes got tedious, so if the number for a duration is omitted, then the next note uses the previous note's duration:

Periods signify dots, "r" is for a rest, and "~" indicates a tie:

Sharps, flats, and, if desired for clarity, naturals are indicated with #, - (not b) and, n, respectively:

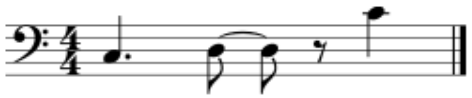
A lyric syllable is specified by appending it after the note with an underscore:

```
s = converter.parse('tinyNotation: 4/4 C4 D E8 F G16 A B c')
s.show()
```



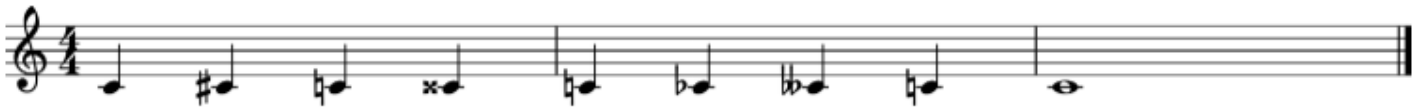
Periods signify dots, "r" is for a rest, and "~" indicates a tie:

```
s = converter.parse('tinyNotation: 4/4 C.4 D8~ D8 r c4')
s.show()
```



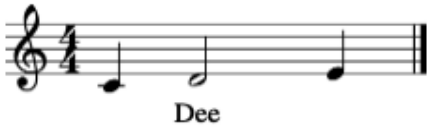
Sharps, flats, and, if desired for clarity, naturals are indicated with #, - (not b) and, n, respectively:

```
s = converter.parse('tinyNotation: 4/4 c4 c# c c## cn c- c-- c c1')
s.show()
```



A lyric syllable is specified by appending it after the note with an underscore:

```
s = converter.parse('tinyNotation: 4/4 c4 d2_Dee e4')
s.show()
```



1.4. tinyNotation(cont.)

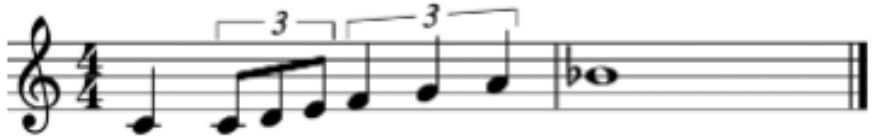
And, finally, triplets are possible by enclosing the triplet notes in curly brackets along with a special trip prefix:

Okay – so what if you want to do something more complex? Apply an id to a note with the "=" tag, and then make changes to it using music21:

And that's how I use TinyNotation, about 90% of the time. But when I need to, I can make something more complex…

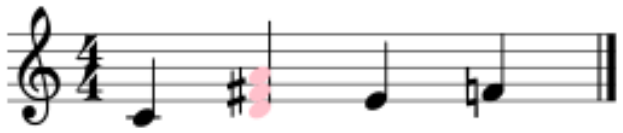
And, finally, triplets are possible by enclosing the triplet notes in curly

```
s = converter.parse('tinyNotation: 4/4 c4 trip{c8 d e} t
s.show()
```



Okay – so what if you want to do something more complex? Apply an changes to it using music21:

```
s = converter.parse('tinyNotation: 4/4 c4 d=id2 e f')
n = s.recurse().getElementById('id2')
ch = chord.Chord('D4 F#4 A4')
ch.style.color = 'pink'
n.activeSite.replace(n, ch)
s.show()
```



And that's how I use TinyNotation, about 90% of the time. But when I complex...

yjkim@hanbat.ac.kr

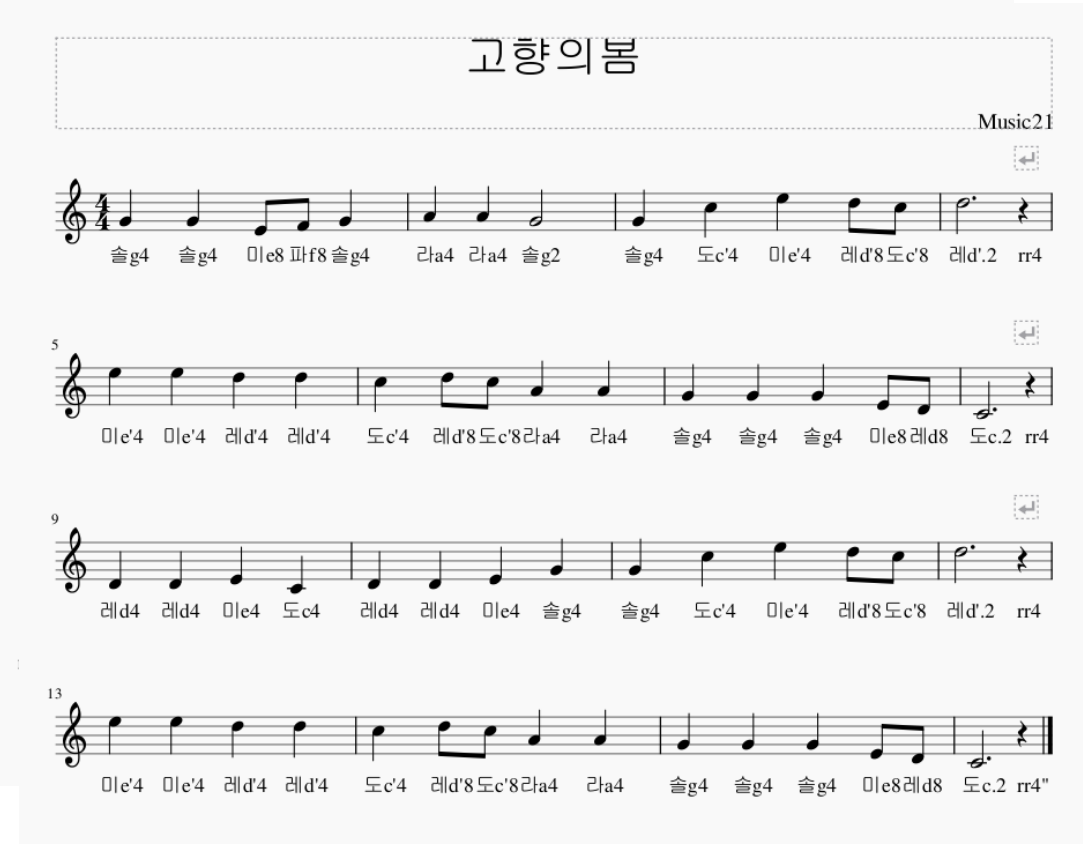
1.4. tinyNotation(cont.)

```
c=converter.parse("tinynotation: 4/4 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")
c.show('midi',fp='나비야.mid')
c.show()
```

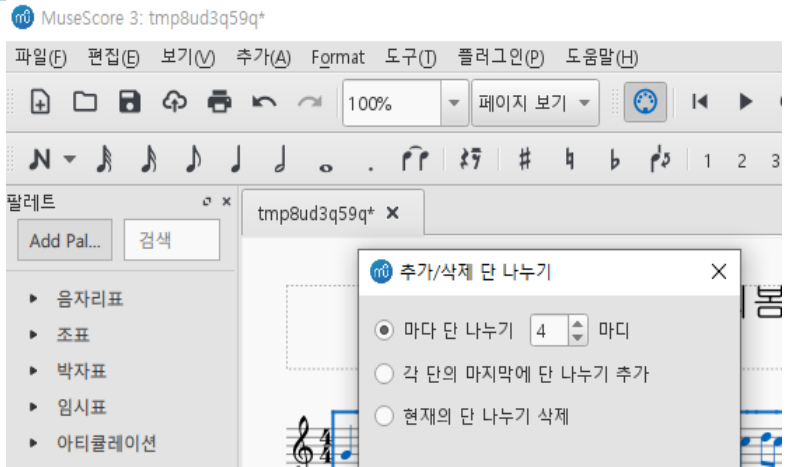
나비야.mid


```
c=converter.parse("""tinynotation: 4/4
  g4 g e8 f8 g4 a a g2          g4 c' e' d'8 c'8 d'.2 r4
  e'4 e'4 d' d' c' d'8 c'8 a4 a4   g4 g g e8 d8 c.2 r4
  d4 d e c  d4 d e g            g4 c' e' d'8 c'8 d'.2 r4
  e'4 e' d' d' c' d'8 c'8 a4 a4    g4 g g e8 d8 c.2 r4"
  """)
c.show('midi',fp='고향의봄.mid')
c.show()
```

고향의봄.mid



Musescore 단 마디맞추기
Format > 단나누기추가/제거
단나누기 4마디



2. DNN Model -피아노 연주기

- DNN(deep Neural Network) Model -피아노 연주기
  - Packages 소개
1. 고향의봄 음계 악보 데이터 준비하기
2. Dataset 생성
3. DNN 모델의 구성
4. 학습방법 설정
5. 모델 학습시키기 및 모델 저장
6. 학습과정 살펴보기
7. 학습된 모델의 성능평가하기
8. 모델을 사용하여 음을 예측하기
    1. 한 스텝 예측계산으로 전체음악의 악보 생성
    2. 한 음계 씩 기억을 더듬어 전체음악의 악보 생성
    3. 악보의 출력 및 “고향의봄”음악 연주

심층신경망 동영상자료-선형회귀모델
https://www.youtube.com/watch?v=i6z11lutAPs&feature=youtu.be

yjkim@hanbat.ac.kr

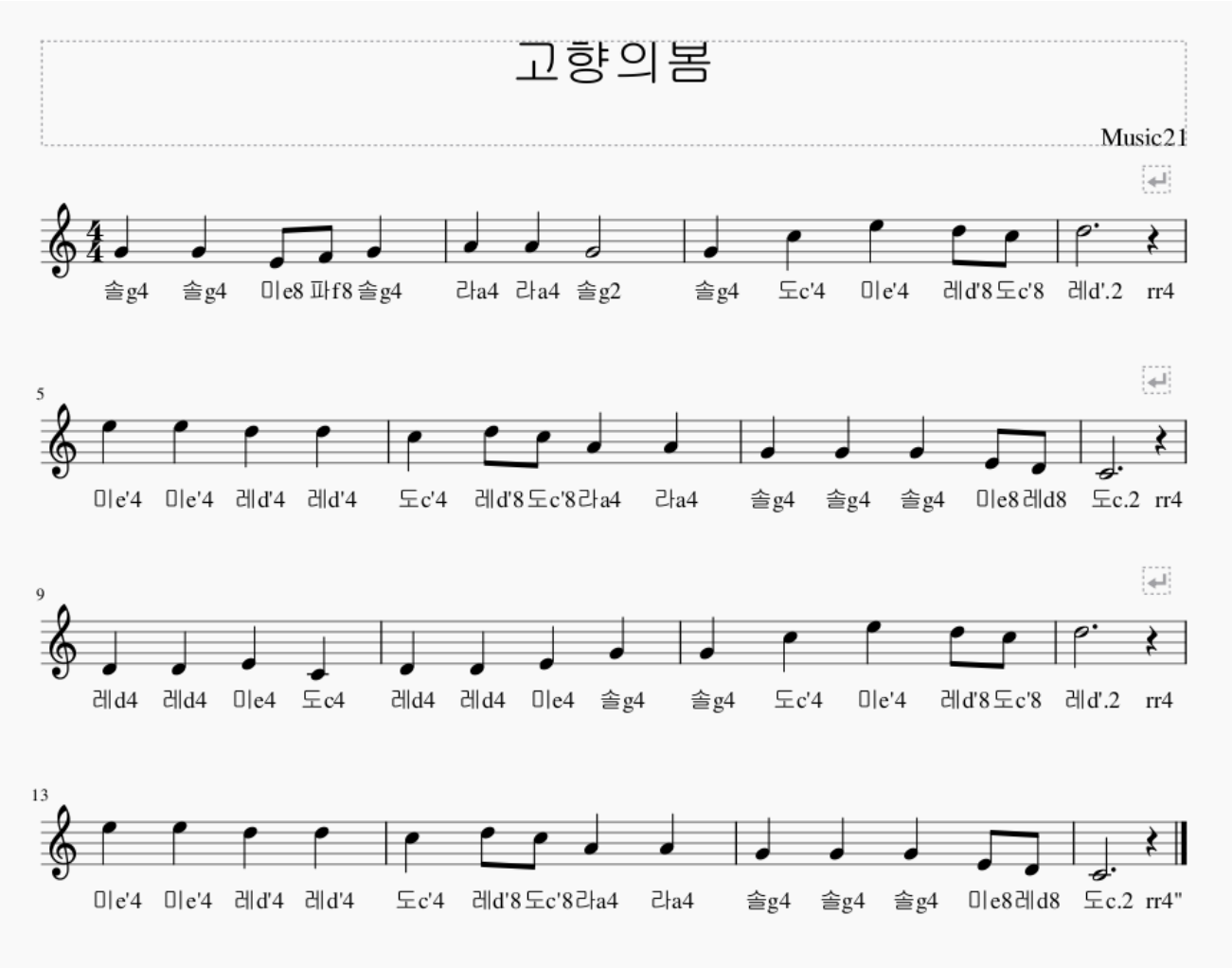
2. DNN Model -피아노 연주기(cont.)

Packages

```
import numpy as np
from music21 import *
import tensorflow.keras as keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from keras.utils import np_utils
from music21 import *
import json

TIME_STEP=5
```

1. 데이터준비하기

```
고향의봄="""
 g4 g4 e8 f8 g4 a4 a4 g2          g4 c'4 e'4 d'8 c'8 d'.2 r4
 e'4 e'4 d'4 d'4 c'4 d'8 c'8 a4 a4   g4 g4 g4 e8 d8 c.2 r4
 d4 d4 e4 c4  d4 d4 e4 g4          g4 c'4 e'4 d'8 c'8 d'.2 r4
 e'4 e'4 d'4 d'4 c'4 d'8 c'8 a4 a4   g4 g4 g4 e8 d8 c.2 r4"
 """
```



2. DNN Model -피아노 연주기(cont.)

 # 2. dataset 생성

```
# 악보로부터 학습용 데이터 X와 라벨 Y 작성
dataset,n2i=seq2dataset(고향의봄)
i2n={n2i[n]:n for n in n2i.keys()}                #index to nate code사전

open('n2i.json','w').write(json.dumps(n2i))       #save  n2i
#n2i = json.loads(open('n2i.json').read())        #load n2i
# dataset :
#    인덱스 코드                음계코드
#   [[15 15 12 13 15  0]       g4 g4 e8 f8 g4 a4
#    [15 12 13 15  0  0]       g4 e8 f8 g4 a4 a4
#    ...
#    [15 15 15 12  9  3]       g4 g4 g4 e8 d8 c.2
#    [15 15 12  9  3 17]]      g4 g4 e8 d8 c.2 r4
#n2i : {'a4': 0, "c'4": 1, "c'8": 2, 'c.2': 3, 'c4': 4, "d'.2": 5, "d'4": 6, "d'8": 7, 'd4': 8, 'd8': 9, "e'4": 10,
'e4': 11, 'e8': 12, 'f8': 13, 'g2': 14, 'g4': 15, 'r4': 16, 'r4"': 17}

X=dataset[:,:TIME_STEP]                           #입력 (57,5)
Y=dataset[:,TIME_STEP]                            #출력 (57,)

no_of_vocaburary=len(n2i.keys())                  #음계의 수          18
X=X/float(no_of_vocaburary)                       #정규화된 X          (57,5)
Y_ohe=np_utils.to_categorical(                    #one_hot_encoding of Y (57,18)
   Y,num_classes=no_of_vocaburary)
```

```
def seq2dataset(seq) :
   #입력 :    seq ,      악보음계코드
   #출력 :    dataset,   훈련용 sample 집합
   #          n2i,       음계코드사전 : note code to index
   #seq : "g4 g4 e8 f8 g4 a4 a4 g2 ..."
   #=>['g4', 'g4', 'e8', 'f8', 'g4', 'a4', 'a4', 'g2',...]
   song=[]
   for s in seq.split(' '):
      s=s.strip()
      if s!='' : song.append(s)
   vocaburary=sorted(set(song))              #['a4', "c'4", "c'8", 'c.2',...]
   n2i={n:i for i,n in enumerate(vocaburary)}  #{'a4': 0, "c'4": 1, "c'8": 2, ...}
   song_decoded=[n2i[n] for n in song]       #[15, 15, 12, 13, 15, 0, 0, 14,
15,..., 9, 3, 17]
   #[15, 15, 12, 13, 15, 0, 0, 14, 15, 1, 10, 7, 2, 5, 16, 10, 10, 6, 6, 1, 7, 2, 0, 0,
15, 15, 15, 12, 9, 3, 16, 8, 8, 11, 4, 8, 8, 11, 15, 15, 1, 10, 7, 2, 5, 16, 10, 10, 6,
6, 1, 7, 2, 0, 0, 15, 15, 15, 12, 9, 3, 17]
   dataset=[]
   for i in np.arange(TIME_STEP,len(song_decoded)) :
      dataset.append(song_decoded[i-TIME_STEP:i+1])
   dataset=np.array(dataset)
   return dataset,n2i
```

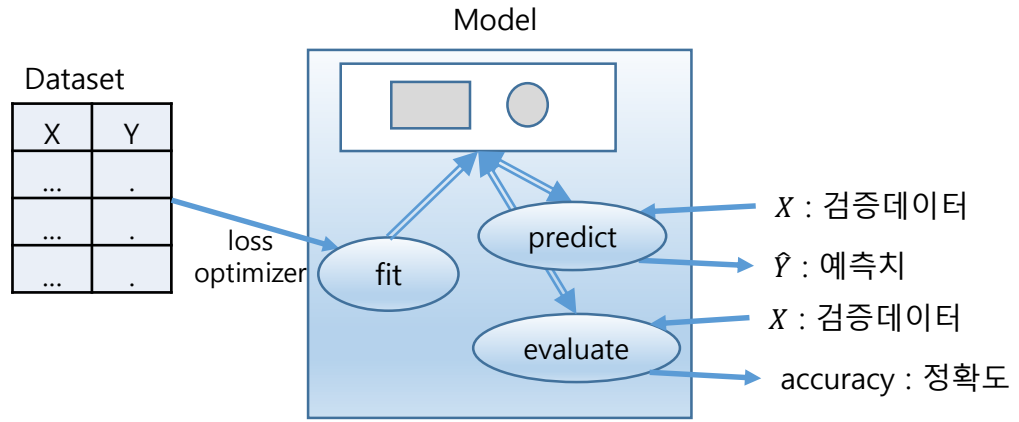
심층신경망 동영상자료-선형회귀모델
https://www.youtube.com/watch?v=i6z11lutAPs&feature=youtu.be

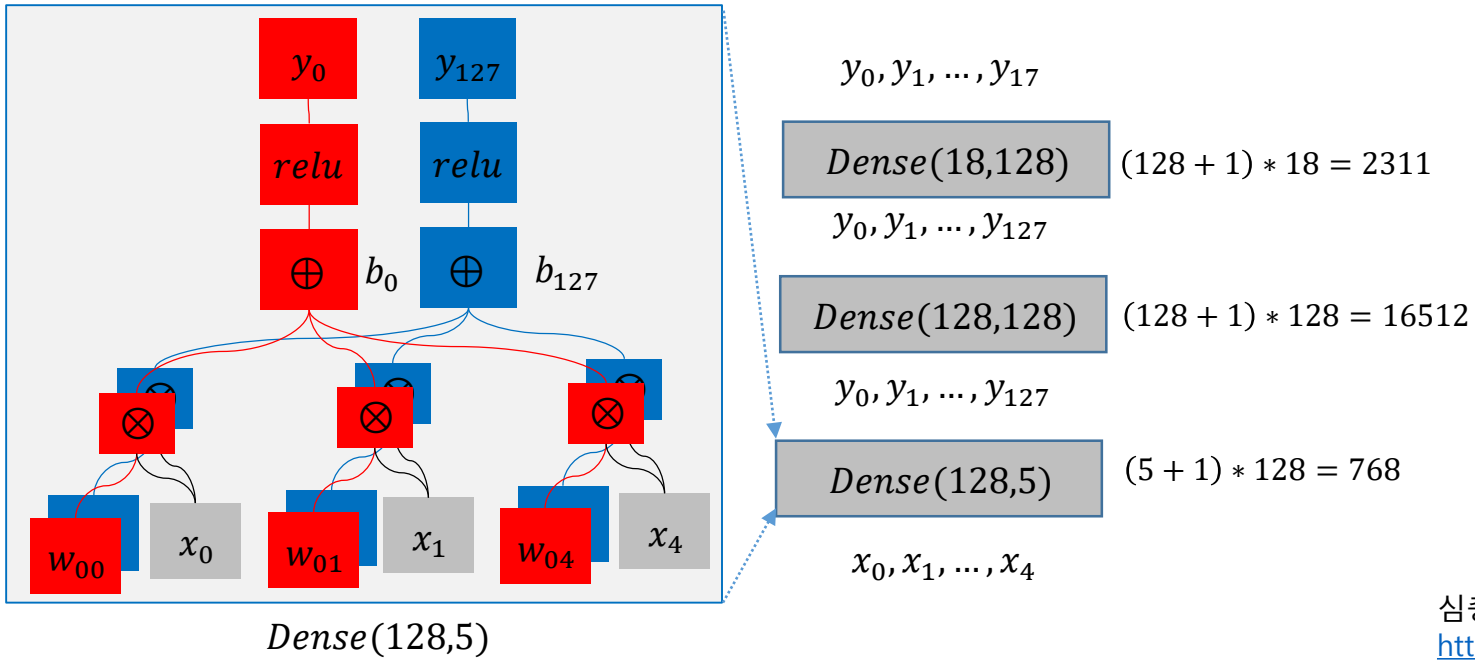
yjkim@hanbat.ac.kr

2. DNN Model -피아노 연주기(cont.)

#3. 모델의 구성

```
model = Sequential(name="DNN_MUSIC_SONG")
model.add(Dense(128, input_dim=5STEP, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(no_of_vocaburary, activation='softmax'))
model.summary()
```





$Dense(128,5)$

Model: "DNN_MUSIC21_SONG"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	768
dense_1 (Dense)	(None, 128)	16512
dense_2 (Dense)	(None, 18)	2322

Total params: 19,602
Trainable params: 19,602
Non-trainable params: 0

심층신경망 동영상자료-선형회귀모델
https://www.youtube.com/watch?v=i6z11lutAPs&feature=youtu.be

2. DNN Model -피아노 연주기(cont.)

#3. 모델의 구성

```
model = Sequential(name="DNN_MUSIC_SONG")
model.add(Dense(128, input_dim=TIME_STEP, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(no_of_vocaburary, activation='softmax'))
model.summary()
```

#4. 학습방법 설정

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

#5. 모델 학습시키기

```
hist=model.fit(X, Y_ohe, epochs=500, batch_size=60, verbose=2,
validation_data=(X,Y_ohe))
model.save('model')                    #학습된 모델저장
```

```
Model: "DNN_MUSIC21_SONG"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               768
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_2 (Dense)              (None, 18)                2322
=================================================================
Total params: 19,602
Trainable params: 19,602
Non-trainable params: 0
_________________________________________________________________
```

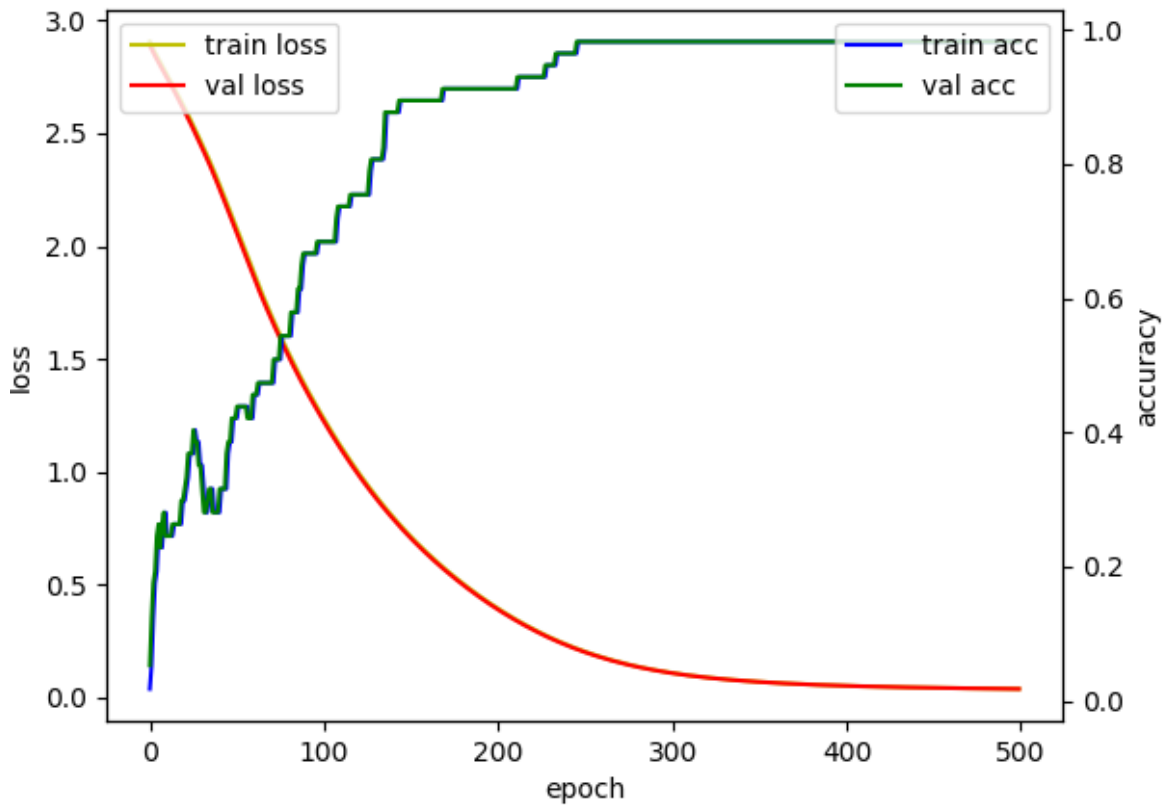
```
1/1 - 0s - loss: 2.9064 - accuracy: 0.0175 - val_loss: 2.8895 - val_accuracy: 0.0526
Epoch 2/500
1/1 - 0s - loss: 2.8895 - accuracy: 0.0526 - val_loss: 2.8733 - val_accuracy: 0.1228
Epoch 3/500
1/1 - 0s - loss: 2.8733 - accuracy: 0.1228 - val_loss: 2.8577 - val_accuracy: 0.1754
Epoch 4/500
1/1 - 0s - loss: 2.8577 - accuracy: 0.1754 - val_loss: 2.8427 - val_accuracy: 0.1930
Epoch 5/500
```

```
Epoch 496/500
1/1 - 0s - loss: 0.0375 - accuracy: 0.9825 - val_loss: 0.0375 - val_accuracy: 0.9825
Epoch 497/500
1/1 - 0s - loss: 0.0375 - accuracy: 0.9825 - val_loss: 0.0374 - val_accuracy: 0.9825
Epoch 498/500
1/1 - 0s - loss: 0.0374 - accuracy: 0.9825 - val_loss: 0.0373 - val_accuracy: 0.9825
Epoch 499/500
1/1 - 0s - loss: 0.0373 - accuracy: 0.9825 - val_loss: 0.0372 - val_accuracy: 0.9825
Epoch 500/500
1/1 - 0s - loss: 0.0372 - accuracy: 0.9825 - val_loss: 0.0371 - val_accuracy: 0.9825
```

심층신경망 동영상자료-선형회귀모델
https://www.youtube.com/watch?v=i6z11lutAPs&feature=youtu.be

2. DNN Model -피아노 연주기(cont.)

```
# 6. 학습과정 살펴보기

model = keras.models.load_model("model")#저장된 모델로딩 사용

#학습과정의 loss화 accuracy의 변화 추이를 분석한다.
import matplotlib.pyplot as plt
fig, loss_ax = plt.subplots()
acc_ax = loss_ax.twinx()

loss_ax.plot(hist.history['loss'], 'y', label='train loss')
loss_ax.plot(hist.history['val_loss'], 'r', label='val loss')
loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
loss_ax.legend(loc='upper left')

acc_ax.plot(hist.history['accuracy'], 'b', label='train acc')
acc_ax.plot(hist.history['val_accuracy'], 'g', label='val acc')
acc_ax.set_ylabel('accuracy')
acc_ax.legend(loc='upper right')
plt.show()
```



2. DNN Model -피아노 연주기(cont.)

```
# 7. 한습된 모델의 성능평가하기

scores = model.evaluate(X,Y_ohe)    #Score  accuracy: 98.25%
print("\nScore  %s: %.2f%%".format(model.metrics_names[1], scores[1]*100))


# 8. 모델을 사용하여 음을 예측하기

seq_out = ['g4','g4','e8','f8','g4']      #출력의 시작 5(TIME_STEP)개 ,seed가 필요
pred_count = len(X)                        #최대 예측 개수 정의

# 한 스텝 예측계산으로 모든 악보를 계산
pred_out = model.predict(X)          # 모든 X에 대한 Y_ohe을 계산
for i in range(pred_count):          # 모든 예측의 개수만큼 계산
    idx = np.argmax(pred_out[i])     # one-hot 인코딩을 인덱스 값으로 변환
    seq_out.append(i2n[idx])         # seq_out에 예측된 index를 음계로변호나하여 악보를 반든다.
print("one step prediction : ", seq_out) #예측된 악보출력
```

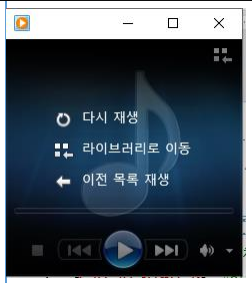
```
Score  accuracy: 98.25
```

```
one step prediction :  ['g4', 'g4', 'e8', 'f8', 'g4', 'a4', 'a4', 'g2', 'g4',
"c'4", "e'4", "d'8", "c'8", "d'.2", 'r4', "e'4", "e'4", "d'4", "d'4", "c'4",
d'8", "c'8", 'a4', 'a4', 'g4', 'g4', 'g4', 'e8', 'd8', 'c.2', 'r4"', 'd4', 'd4
', 'e4', 'c4', 'd4', 'd4', 'e4', 'g4', 'g4', "c'4", "e'4", "d'8", "c'8", "d'.2
", 'r4', "e'4", "e'4", "d'4", "d'4", "c'4", "d'8", "c'8", 'a4', 'a4', 'g4', 'g
4', 'g4', 'e8', 'd8', 'c.2', 'r4"']
```

2. DNN Model -피아노 연주기(cont.)

```
# 차례로 한 음씩 기억을 더듬어 전체 악보를 예측
seq_in = ['g4','g4','e8','f8','g4']   #입력 최기화, 시작을 위한 5(TIME_STEP)개 음으로 시작
seq_out = seq_in                      #출력 악보 초기화
seq_in = [n2i[it] / float(no_of_vocaburary) for it in seq_in]
                                      # 음계(코드)를 인덱스값으로 변환하고 정규화
# 모든 악보의 음을 차례로 계산
for i in range(pred_count):
    sample_in = np.array(seq_in)    # 5개음의 인덱스  [0.8, 0.8, 0.6, 0.7, 0.8] (5,)
    sample_in = np.reshape(
        sample_in, (1, TIME_STEP))  # [[0.8, 0.8, 0.6, 0.7, 0.8]] (1,5)
    pred_out = model.predict(sample_in) # [[0.9,0.3,..]]  (1,18)
    idx = np.argmax(pred_out)         # 0
    seq_out.append(i2n[idx])          # ['g4', 'g4', 'e8', 'f8', 'g4', 'a4']
    seq_in.append(                    #[0.8, 0.8, 0.6, 0.7, 0.8, 0.0] (6,)
        idx / float(no_of_vocaburary))
    seq_in.pop(0)                     #[0.8, 0.6, 0.7, 0.8, 0.0]      (5,)

print("full song prediction : ", seq_out)                          #예측한 악보 음코드 출력
converter.parse("tinynotation: 4/4 "+' '.join(seq_out)).show()     #예측한 악보 출력
converter.parse("tinynotation: 4/4 "+' '.join(seq_out)).show('midi') #예측한 악보의 음악 재생
```

```
full song prediction :  ['g4', 'g4', 'e8', 'f8', 'g4', 'a4', 'a4', 'g2', 'g4',
 'c'4", 'e'4", 'd'8", 'c'8", 'd'.2", 'r4", 'e'4", 'e'4", 'd'4", 'd'4", 'c'4",
"d'8", "c'8", 'a4', 'a4', 'g4', 'g4', 'g4', 'e8', 'd8', 'c.2', 'r4"', 'd4', 'c
4', 'e4', 'c4', 'd4', 'd4', 'e4', 'g4', 'g4', "c'4", "e'4", "d'8", "c'8", "d'
2", 'r4', "e'4", "e'4", "d'4", "d'4", "c'4", "d'8", "c'8", 'a4', 'a4', 'g4',
g4', 'g4', 'e8', 'd8', 'c.2', 'r4"']
Press any key to continue . . .
```

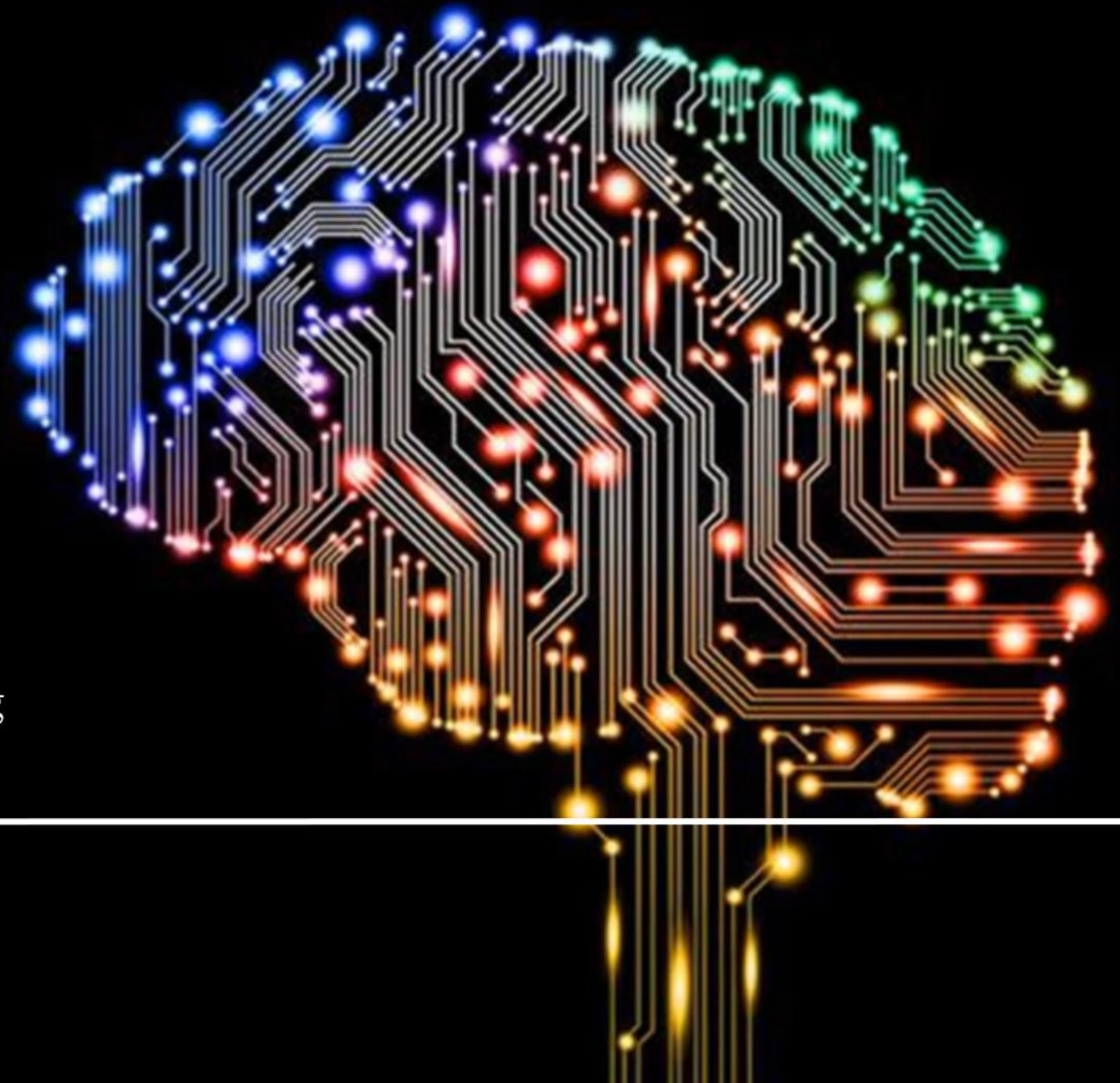


고향의봄.mid

yjkim@hanbat.ac.kr

Contents

- 심층신경망 동영상자료-선형회귀모델 동영상(37분) pdf
- 소프트맥스분류 동영상(48분) pdf



Digital processing of Speech Signals

Yoon Joong Kim,
Hanbat National University