'''

chord\_nn\_tester.py

version 0.1

author: R2\_B09

'''

**import** sys

**import** keras, h5py, pygame

**import** numpy **as** NP

**import** loc\_note\_MIDI

**from** pygame **import** midi

**from** pygame.locals **import** \*

**import** moira, soleil

**def** NNtest(INPUT, EXPECTED\_OUTPUT):

'''

    Runs a single NN test pass given input and expected outs as numpy arrays.

    :param EXPECTED\_INPUT: Numpy array for expected input.

    :param EXPECTED\_OUTPUT: Numpy array for expected output.

    :return:

    '''

# TODO: ADD NN (TEST FOR INDIVIDUAL SAMPLE) CODE HERE

# SAFETY NET: If nothing is running through realtime input then pass

**if** INPUT **is** None **or** len(INPUT) < 24:

**pass**

**else**:

# TODO: Add NN code here

**def** realTimeTest():

'''

    Displays the mappings of the notes currently being played on MIDI and runs them through the NN,

    (kushalbhabra, 2013; Tang, n.d.)

    '''

# display a list of MIDI devices connected to the computer

# this function written by Stephen Tang (n.d.), stephentang.io

**def** print\_device\_info():

**for** i **in** range(pygame.midi.get\_count()):

r = pygame.midi.get\_device\_info(i)

(interf, name, input, output, opened) = r

in\_out = ""

**if** input:

in\_out = "(input)"

**if** output:

in\_out = "(output)"

**print**("%2i: interface: %s, name: %s, opened: %s %s" %

(i, interf, name, opened, in\_out))

pygame.init()

pygame.fastevent.init()

event\_get = pygame.fastevent.get

event\_post = pygame.fastevent.post

pygame.midi.init()

**print**("Available MIDI devices:")

print\_device\_info()

MI = int(input("Select device ID: ")) # 2 the default for Project-X99

input\_id = MI

midi\_in = pygame.midi.Input(input\_id)

pygame.display.set\_caption("midi test")

screen = pygame.display.set\_mode((400, 300), RESIZABLE, 32)

**print**("Initializing realtime poll process.")

realtime = True

currentNoteNumbersInChord = [] # Current MIDI note #s in chord

currentNotesInChord = [] # Binary mapping of notes in chord

currentNoteNamesInChord = [] # Current note names in chord

**for** i **in** range(24):

currentNotesInChord.append(0)

**print**("initialize:\n", currentNotesInChord, "length:" + str(len(currentNotesInChord)))

**while** realtime:

# pygame event handling

events = event\_get()

**for** e **in** events:

**if** e.type **in** [QUIT]:

realtime = False

**if** e.type **in** [KEYDOWN]:

realtime = False

**if** midi\_in.poll():

midi\_events = midi\_in.read(10)

# MIDI NOTE ON

**if** 0x90 <= midi\_events[0][0][0] <= 0x9F:

mod = 1

**if** len(currentNotesInChord) != 24:

**print**("Error - length of array is not 24! Emergency stop.")

realtime = False

**break**

# Add note number to current note numbers in chord

currentNoteNumbersInChord.append(midi\_events[0][0][1])

# Add note to binary list

root = currentNoteNumbersInChord[0] % 12

**try**:

currentNotesInChord[(root) + (currentNoteNumbersInChord[-1] - currentNoteNumbersInChord[0])] = 1

**except** IndexError:

currentNotesInChord[midi\_events[0][0][1] % 12] = 1

# Add note being played

currentNoteNamesInChord.append(loc\_note\_MIDI.Loc\_Note\_MIDI.midiNumToNote(midi\_events[0][0][1]))

**print**(currentNotesInChord)

**print**(currentNoteNumbersInChord)

**print**(currentNoteNamesInChord)

# MIDI NOTE OFF

**elif** 0x80 <= midi\_events[0][0][0] <= 0x8F:

**print**("Purging variables.")

currentNotesInChord = []

**for** i **in** range(24): currentNotesInChord.append(0)

currentNoteNumbersInChord = []

currentNoteNamesInChord = []

# Aftertouch

**elif** 0xD0 <= midi\_events[0][0][0] <= 0xDF:

**pass**

# Something else except aftertouch

**else**:

**print**("Something else was tinkered.")

**print**(midi\_events[0][0][0])

# TODO: FILL PARAMETERS FOR NNtest()

NNtest(currentNotesInChord, [])

**print**("Exiting.")

midi\_in.close()

pygame.midi.quit()

pygame.quit()

exit()