**모델 1**

def create\_model():

model = keras.Sequential() # Sequential 모델 시작

model.add(keras.layers.Conv2D(filters=48, kernel\_size=3, strides=1, activation=tf.nn.relu, padding='SAME',

input\_shape=(256, 256, 3)))

model.add(keras.layers.MaxPool2D(padding='SAME'))

model.add(keras.layers.Conv2D(filters=54, kernel\_size=3, strides=1, activation=tf.nn.relu, padding='SAME'))

model.add(keras.layers.MaxPool2D(padding='SAME'))

model.add(keras.layers.Conv2D(filters=60, kernel\_size=3, strides=1, activation=tf.nn.relu, padding='SAME'))

model.add(keras.layers.MaxPool2D(padding='SAME'))

model.add(keras.layers.Conv2D(filters=66, kernel\_size=3, strides=1, activation=tf.nn.relu, padding='SAME'))

model.add(keras.layers.MaxPool2D(padding='SAME'))

model.add(keras.layers.Conv2D(filters=72, kernel\_size=3, strides=1, activation=tf.nn.relu, padding='SAME'))

model.add(keras.layers.MaxPool2D(padding='SAME'))

model.add(keras.layers.Conv2D(filters=80, kernel\_size=3, strides=1, activation=tf.nn.relu, padding='SAME'))

model.add(keras.layers.MaxPool2D(padding='SAME'))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(256, activation=tf.nn.relu))

model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(256, activation=tf.nn.relu))

model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(6, activation=tf.nn.softmax)) # Dense층

return model

1)

learning\_rate = 0.005 # 러닝레이트

training\_epochs = 40 # 에폭

batch\_size = 32 # 배치사이즈

Buffer\_size = 100 # 인스턴스 버퍼 사이즈

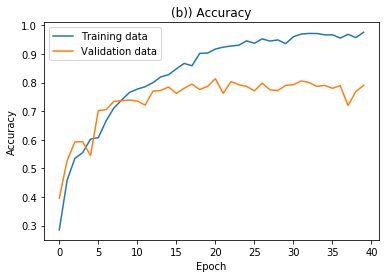
model.compile(loss='categorical\_crossentropy',

optimizer = 'Adam',

metrics = ['accuracy'])

VAl loss : 1.3153951887791215

VAl ACCURACY : 0.7904125



**loss : 2.0503601941201177**

**ACCURACY : 0.716569**

true positive\_each label: (301, 177, 181, 272, 217, 197)

zero\_negative: (5, 31, 3, 5, 30)

one\_negative: (6, 11, 28, 52, 26)

two\_negative: (49, 28, 8, 24, 9)

three\_negative: (7, 8, 1, 5, 7)

four\_negative: (1, 53, 11, 8, 13)

five\_negative: (12, 37, 13, 17, 24)

2)

learning\_rate = 0.007 # 러닝레이트

training\_epochs = 40 # 에폭

batch\_size = 32 # 배치사이즈

Buffer\_size = 100 # 인스턴스 버퍼 사이즈

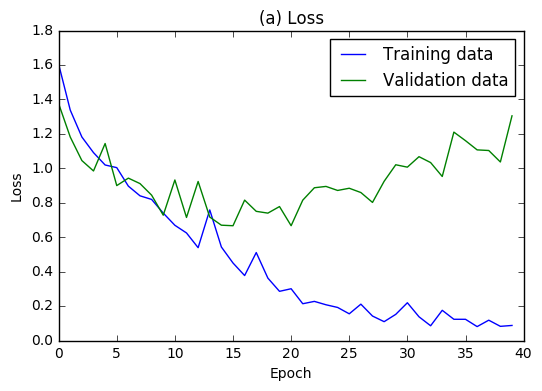
model.compile(loss='categorical\_crossentropy',

optimizer = 'Adam',

metrics = ['accuracy'])

VAl loss : 1.3045391977298486

VAl ACCURACY : 0.77814937



**loss : 1.965650053404085**

**ACCURACY : 0.68193924**

true positive\_each label: (283, 163, 198, 272, 157, 207)

zero\_negative: (4, 49, 6, 2, 31)

one\_negative: (8, 17, 32, 44, 36

two\_negative: (60, 17, 4, 10, 10)

three\_negative: (12, 1, 1, 5, 9)

four\_negative: (7, 73, 19, 17, 30)

five\_negative: (20, 30, 12, 15, 16)

3)

learning\_rate = 0.0001 # 러닝레이트

training\_epochs = 40 # 에폭

batch\_size = 32 # 배치사이즈

Buffer\_size = 100 # 인스턴스 버퍼 사이즈

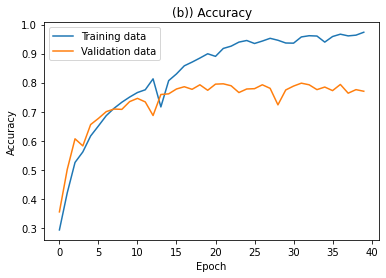
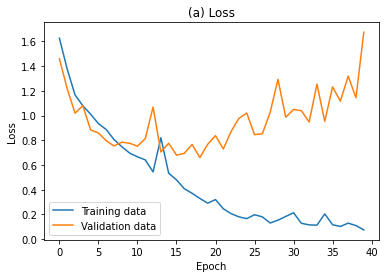
model.compile(loss='categorical\_crossentropy',

optimizer = 'Adam',

metrics = ['accuracy'])

VAl loss : 1.6744884017983674

VAl ACCURACY : 0.77034557



**loss : 2.254451348734588**

**ACCURACY : 0.700586**

true positive\_each label: (288, 120, 217, 268, 231, 191)

zero\_negative: (4, 63, 2, 3, 15)

one\_negative: (3, 26, 29, 91, 31)

two\_negative: (36, 10, 5, 27, 4)

three\_negative: (6, 5, 2, 7, 12)

four\_negative: (5, 19, 18, 15, 15)

five\_negative: (23, 27, 22, 13, 24)

4)

learning\_rate = 0.0005 # 러닝레이트

training\_epochs = 40 # 에폭

batch\_size = 32 # 배치사이즈

Buffer\_size = 100 # 인스턴스 버퍼 사이즈

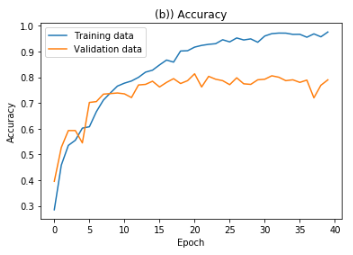
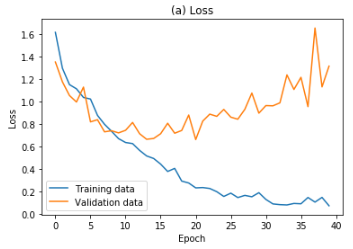
model.compile(loss='categorical\_crossentropy',

optimizer = 'Adam',

metrics = ['accuracy'])

VAl loss : 1.3153951887791215

VAl ACCURACY : 0.7904125



**loss : 2.0503601941201177**

**ACCURACY : 0.716569**

true positive\_each label: (301, 177, 181, 272, 217, 197)

zero\_negative: (5, 31, 3, 5, 30)

one\_negative: (6, 11, 28, 52, 26)

two\_negative: (49, 28, 8, 24, 9)

three\_negative: (7, 8, 1, 5, 7)

four\_negative: (1, 53, 11, 8, 13)

five\_negative: (12, 37, 13, 17, 24)

5)

learning\_rate = 0.01 # 러닝레이트

training\_epochs = 40 # 에폭

batch\_size = 32 # 배치사이즈

Buffer\_size = 100 # 인스턴스 버퍼 사이즈

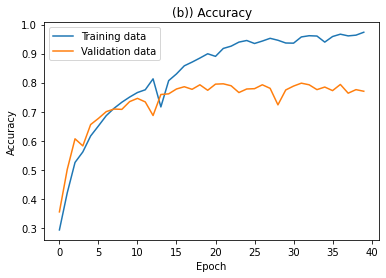
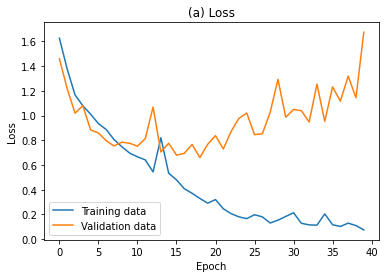
model.compile(loss='categorical\_crossentropy',

optimizer = 'Adam',

metrics = ['accuracy'])

VAl loss : 1.6744884017983674

VAl ACCURACY : 0.77034557



**loss : 2.254451348734588**

**ACCURACY : 0.700586**

true positive\_each label: (288, 120, 217, 268, 231, 191)

zero\_negative: (4, 63, 2, 3, 15)

one\_negative: (3, 26, 29, 91, 31)

two\_negative: (36, 10, 5, 27, 4)

three\_negative: (6, 5, 2, 7, 12)

four\_negative: (5, 19, 18, 15, 15)

five\_negative: (23, 27, 22, 13, 24)

**모델 2**

def create\_model():

model = keras.Sequential() # Sequential 모델 시작

model.add(keras.layers.Conv2D(filters=64, kernel\_size=3, strides=1, activation=tf.nn.relu, padding='SAME', input\_shape=(256, 256, 3)))

model.add(keras.layers.MaxPool2D(padding='SAME'))

model.add(keras.layers.Conv2D(filters=64, kernel\_size=3, strides=1, activation=tf.nn.relu, padding='SAME'))

model.add(keras.layers.MaxPool2D(padding='SAME'))

model.add(keras.layers.Conv2D(filters=64, kernel\_size=3, strides=1, activation=tf.nn.relu, padding='SAME'))

model.add(keras.layers.MaxPool2D(padding='SAME'))

model.add(keras.layers.Conv2D(filters=64, kernel\_size=3, strides=1, activation=tf.nn.relu, padding='SAME'))

model.add(keras.layers.MaxPool2D(padding='SAME'))

model.add(keras.layers.Conv2D(filters=64, kernel\_size=3, strides=1, activation=tf.nn.relu, padding='SAME'))

model.add(keras.layers.MaxPool2D(padding='SAME'))

model.add(keras.layers.Conv2D(filters=64, kernel\_size=3, strides=1, activation=tf.nn.relu, padding='SAME'))

model.add(keras.layers.MaxPool2D(padding='SAME'))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(256, activation=tf.nn.relu))

model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(256, activation=tf.nn.relu))

model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(6, activation=tf.nn.softmax)) # Dense층

return model

1) 최종 모델

learning\_rate = 0.01 # 러닝레이트

training\_epochs = 60 # 에폭

batch\_size = 32 # 배치사이즈

Buffer\_size = 100 # 인스턴스 버퍼 사이즈

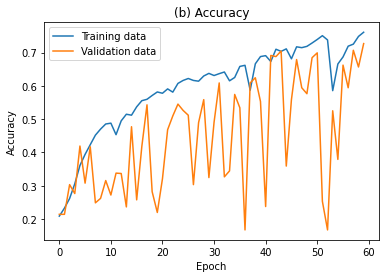
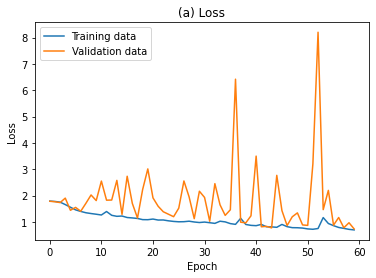
model.compile(loss='categorical\_crossentropy',

optimizer = 'SGD',

metrics = ['accuracy'])

VAl loss : 0.7298346757888794

VAl ACCURACY : 0.7268673181533813



**loss : 0.7757940888404846**

**ACCURACY : 0.7144379615783691**

true positive\_each label: (289, 183, 200, 265, 228, 176)

zero\_negative: (5, 64, 1, 3, 13)

one\_negative: (5, 13, 11, 82, 6)

two\_negative: (38, 18, 1, 37, 5)

three\_negative: (16, 3, 1, 11, 4)

four\_negative: (4, 39, 19, 2, 11)

five\_negative: (28, 34, 18, 12, 32)

2)

learning\_rate = 0.007 # 러닝레이트

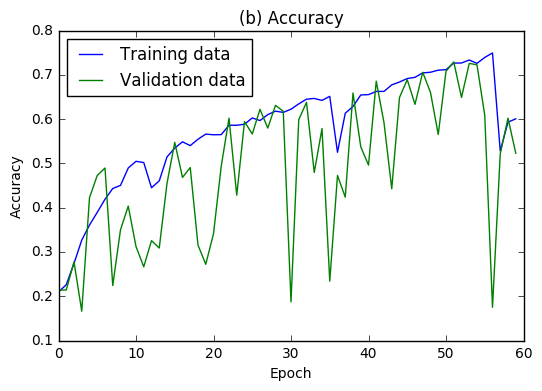
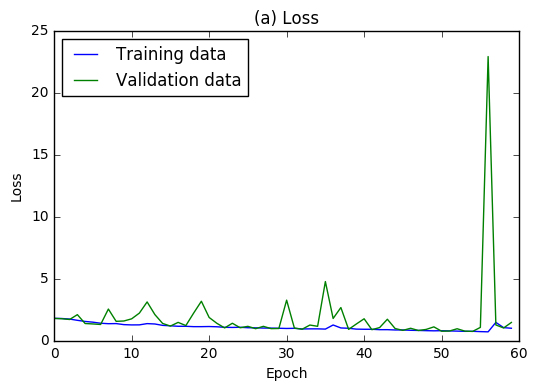
training\_epochs = 60 # 에폭

batch\_size = 32 # 배치사이즈

Buffer\_size = 100 # 인스턴스 버퍼 사이즈

VAl loss : 1.4545011618730084

VAl ACCURACY : 0.522854



**loss : 1.7095658278046102**

**ACCURACY : 0.48428342**

true positive\_each label: (274, 43, 44, 268, 78, 202)

zero\_negative: (1, 32, 1, 0, 67)

one\_negative: (69, 6, 82, 25, 75)

two\_negative: (185, 8, 14, 13, 35)

three\_negative: (21, 0, 0, 1, 10)

four\_negative: (51, 10, 8, 47, 109)

five\_negative: (51, 19, 6, 14, 8)

3)

learning\_rate = 0.005 # 러닝레이트

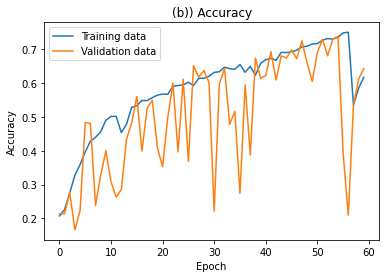
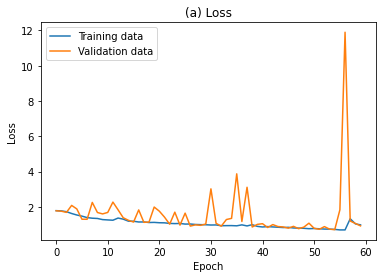
training\_epochs = 60 # 에폭

batch\_size = 32 # 배치사이즈

Buffer\_size = 100 # 인스턴스 버퍼 사이즈

VAl loss : 0.9250698352737171

VAl ACCURACY : 0.6432553



**loss : 0.989334642601318**

**ACCURACY : 0.6164092**

true positive\_each label: (280, 152, 111, 275, 177, 162)

zero\_negative: (5, 37, 4, 1, 48)

one\_negative: (11, 6, 51, 62, 18)

two\_negative: (95, 49, 6, 32, 6)

three\_negative: (13, 3, 0, 2, 7)

four\_negative: (10, 61, 7, 21, 27)

five\_negative: (51, 40, 5, 20, 22)

4)

learning\_rate = 0.0005 # 러닝레이트

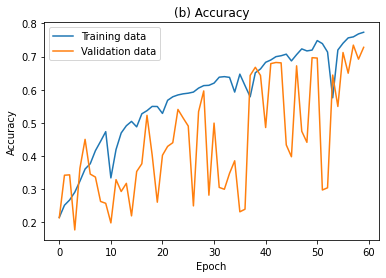
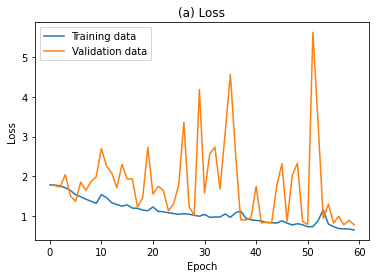
training\_epochs = 60 # 에폭

batch\_size = 32 # 배치사이즈

Buffer\_size = 100 # 인스턴스 버퍼 사이즈

VAl loss : 0.7710262537002563

VAl ACCURACY : 0.7279821634292603



**loss : 0.8426657915115356**

**ACCURACY : 0.705380916595459**

true positive\_each label: (299, 166, 193, 273, 231, 162)

zero\_negative: (7, 47, 2, 6, 14)

one\_negative: (6, 10, 19, 90, 9)

two\_negative: (44, 13, 1, 41, 7)

three\_negative: (14, 2, 1, 7, 3)

four\_negative: (4, 34, 21, 8, 5)

five\_negative: (34, 40, 14, 11, 39)

5)

learning\_rate = 0.0001 # 러닝레이트

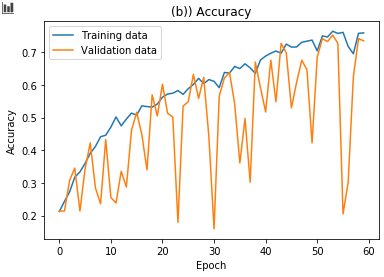
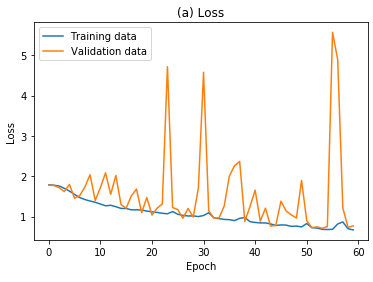
training\_epochs = 60 # 에폭

batch\_size = 32 # 배치사이즈

Buffer\_size = 100 # 인스턴스 버퍼 사이즈

VAl loss : 0.7703278287996018

VAl ACCURACY : 0.7346711



loss : 0.9017369590925929

ACCURACY : 0.68034095

true positive\_each label: (303, 147, 174, 279, 221, 153)

zero\_negative: (5, 35, 5, 4, 23)

one\_negative: (7, 5, 51, 77, 13)

two\_negative: (57, 19, 6, 41, 2)

three\_negative: (14, 1, 1, 2, 3)

four\_negative: (6, 32, 10, 22, 12)

five\_negative: (38, 33, 13, 24, 39)

---------------------------------------------------------------

**Train : val : test = 7:1:2**

**Train 데이터 비율을 늘렸을 때의 결과와 비교해보고 싶어서 진행함.**

**모델 2번의 lr = 0.01을 최종 선정했으며, 최종 모델을 Adam, SGD의 Optimizer로 각각 적용해 결과를 비교해봄.**

learning\_rate = 0.01 # 러닝레이트

training\_epochs = 60 # 에폭

batch\_size = 32 # 배치사이즈

Buffer\_size = 100 # 인스턴스 버퍼 사이즈

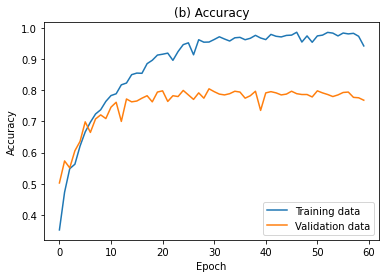
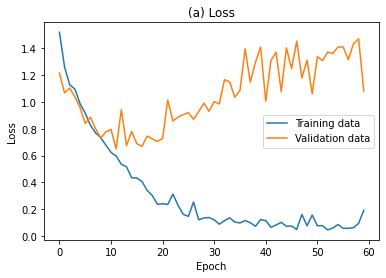
model.compile(loss='categorical\_crossentropy',

optimizer = 'Adam',

metrics = ['accuracy'])

VAl loss : 1.081316397764296

VAl ACCURACY : 0.7683246



**loss : 1.6158767998593409**

**ACCURACY : 0.7437107**

true positive\_each label: (207, 147, 151, 172, 127, 142)

zero\_negative: (4, 40, 4, 1, 19)

one\_negative: (1, 9, 4, 29, 10)

two\_negative: (18, 7, 1, 12, 10)

three\_negative: (3, 9, 3, 7, 6)

four\_negative: (1, 44, 10, 2, 14)

five\_negative: (11, 23, 6, 4, 14)

learning\_rate = 0.01 # 러닝레이트

training\_epochs = 60 # 에폭

batch\_size = 32 # 배치사이즈

Buffer\_size = 100 # 인스턴스 버퍼 사이즈

model.compile(loss='categorical\_crossentropy',

optimizer = 'SGD',

metrics = ['accuracy'])

VAl loss : 0.7646214962005615

VAl ACCURACY : 0.735602080821991

**loss : 0.84325557947**

**ACCURACY : 0.7130503**