Model train & evaluate (추가 연구 필요)

input data

- X : mfcc, mfcc_delta, stft 를 합친 3차원 데이터
- y: label

mfcc12_0130_total_label.json	1.5 MB
mfcc12_0130_total_stft.json	1.6 GB
mfcc12_0130_total_mfcc_delta.json	1.6 GB
mfcc12_0130_total_mfcc.json	1.6 GB

```
# load data
X, y, X_train, X_validation, X_test, y_train, y_validation, y_test = cnn.load_data()
print(X_train.shape, y_train.shape)
input_shape = (X_train.shape[1], X_train.shape[2], X_train.shape[3])
```

```
File loaded!
File loaded!
File loaded!
File loaded!
100%|██████████| 185566/185566 [00:02<00:00, 69028.81it/s]
(111339, 12, 22, 3, 1) (111339,)
```

68개의 라벨

- 일단 있는대로 다 넣어서 학습시켜보고, 혼동을 주는 라벨은 삭제하는 방식으로 시도

68개의 라벨

```
{   "0": "moaning",
    "1": "  ",
    "2": "teeth-chattering",
    "3": "baby_crying",
    "4": "tongue-clicking",
    "5": "crying",
    "6": "nose-blowing",
    "7": "coughing",
    "8": " ",
    "9": "sighing",
    "10": "throat-clearing",
    "11": "shout",
    "12": "panting",
    "13": "clap",
    "14": "laughing",
    "15": "teeth-grinding",
    "16": "cat",
    "17": "bird",
    "18": "dog",
    "19": "lion",
    "20": "horse",
    "21": "read_men",
    "22": "original_clean",
```

```
    "23": "read_women",
    "24": "",
    "25": " ",
    "26": " ",
    "27": " ",
    "28": " ",
    "29": "",
    "30": "",
    "31": "  ",
    "32": " ",
    "33": " ",
    "34": "  ",
    "35": " ",
    "36": " ",
    "37": " ",
    "38": " ",
    "39": "  ",
    "40": " ",
    "41": " ",
    "42": "  ",
    "43": " ",
    "44": " ",
    "45": " ",
    "46": " ",
    "47": " ",
    "48": " ",
    "49": " ",
    "50": " ",
    "51": " ",
    "52": " ",
    "53": " ",
    "54": "  ",
    "55": " ",
    "56": " ",
    "57": "  ",
    "58": " ",
    "59": " ",
    "60": "  ",
    "61": " ",
    "62": " ",
    "63": " ",
    "64": "  ",
    "65": "  ",
    "66": " ",
    "67": "bgm" }
```


modeling

build

```
# compile model
optimiser = keras.optimizers.Adam(learning_rate=0.0001)
model.compile(optimizer=optimiser,
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 10, 20, 32)        896

 max_pooling2d (MaxPooling2D  (None, 10, 20, 32)       0
 )

 batch_normalization (BatchN  (None, 10, 20, 32)       128
 ormalization)

 conv2d_1 (Conv2D)           (None, 8, 18, 128)        36992

 max_pooling2d_1 (MaxPooling  (None, 4, 9, 128)        0
 2D)

 batch_normalization_1 (Batc  (None, 4, 9, 128)        512
 hNormalization)

 conv2d_2 (Conv2D)           (None, 3, 8, 128)         65664

 max_pooling2d_2 (MaxPooling  (None, 3, 8, 128)        0
 2D)

 batch_normalization_2 (Batc  (None, 3, 8, 128)        512
 hNormalization)

 flatten (Flatten)           (None, 3072)              0

 dense (Dense)               (None, 64)                196672

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 68)                4420

=================================================================
Total params: 305,796
Trainable params: 305,220
Non-trainable params: 576
```

model fit

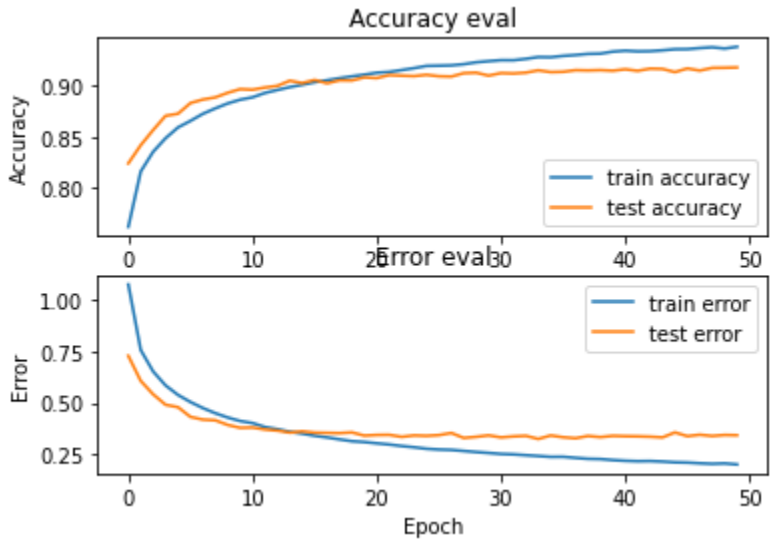
- colab GPU로 하다가 리소스 부족으로 강제 종료당해서 epochs 50으로 GPU없이 fit 진행

- epoch당 250초 정도 소요

```
history = model.fit(X_train, y_train, validation_data=(X_validation,
y_validation),batch_size=32, epochs=50)
```

```
Epoch 1/50
3480/3480 [==============================] - 287s 82ms/step - loss: 1.0720 - accuracy: 0.7618 - val_loss: 0.7274 - val_accuracy: 0.8240
Epoch 2/50
3480/3480 [==============================] - 248s 71ms/step - loss: 0.7553 - accuracy: 0.8160 - val_loss: 0.6062 - val_accuracy: 0.8417
Epoch 3/50
3480/3480 [==============================] - 250s 72ms/step - loss: 0.6517 - accuracy: 0.8353 - val_loss: 0.5416 - val_accuracy: 0.8566
Epoch 4/50
3480/3480 [==============================] - 258s 74ms/step - loss: 0.5845 - accuracy: 0.8486 - val_loss: 0.4911 - val_accuracy: 0.8705
Epoch 5/50
3480/3480 [==============================] - 253s 73ms/step - loss: 0.5386 - accuracy: 0.8592 - val_loss: 0.4795 - val_accuracy: 0.8725
Epoch 6/50
3480/3480 [==============================] - 251s 72ms/step - loss: 0.5045 - accuracy: 0.8655 - val_loss: 0.4327 - val_accuracy: 0.8830
Epoch 7/50
3480/3480 [==============================] - 253s 73ms/step - loss: 0.4753 - accuracy: 0.8725 - val_loss: 0.4195 - val_accuracy: 0.8864
Epoch 8/50
3480/3480 [==============================] - 254s 73ms/step - loss: 0.4505 - accuracy: 0.8778 - val_loss: 0.4162 - val_accuracy: 0.8885
Epoch 9/50
3480/3480 [==============================] - 254s 73ms/step - loss: 0.4297 - accuracy: 0.8826 - val_loss: 0.3939 - val_accuracy: 0.8930
Epoch 10/50
3480/3480 [==============================] - 254s 73ms/step - loss: 0.4127 - accuracy: 0.8863 - val_loss: 0.3799 - val_accuracy: 0.8966
Epoch 40/50
3480/3480 [==============================] - 293s 84ms/step - loss: 0.2242 - accuracy: 0.9333 - val_loss: 0.3411 - val_accuracy: 0.9146
Epoch 41/50
3480/3480 [==============================] - 298s 86ms/step - loss: 0.2208 - accuracy: 0.9342 - val_loss: 0.3395 - val_accuracy: 0.9160
Epoch 42/50
3480/3480 [==============================] - 303s 87ms/step - loss: 0.2188 - accuracy: 0.9336 - val_loss: 0.3387 - val_accuracy: 0.9145
Epoch 43/50
3480/3480 [==============================] - 302s 87ms/step - loss: 0.2195 - accuracy: 0.9338 - val_loss: 0.3370 - val_accuracy: 0.9167
Epoch 44/50
3480/3480 [==============================] - 298s 86ms/step - loss: 0.2170 - accuracy: 0.9346 - val_loss: 0.3336 - val_accuracy: 0.9164
Epoch 45/50
3480/3480 [==============================] - 305s 88ms/step - loss: 0.2134 - accuracy: 0.9357 - val_loss: 0.3576 - val_accuracy: 0.9133
Epoch 46/50
3480/3480 [==============================] - 292s 84ms/step - loss: 0.2121 - accuracy: 0.9358 - val_loss: 0.3396 - val_accuracy: 0.9168
Epoch 47/50
3480/3480 [==============================] - 294s 85ms/step - loss: 0.2087 - accuracy: 0.9368 - val_loss: 0.3465 - val_accuracy: 0.9147
Epoch 48/50
3480/3480 [==============================] - 293s 84ms/step - loss: 0.2062 - accuracy: 0.9376 - val_loss: 0.3406 - val_accuracy: 0.9174
Epoch 49/50
3480/3480 [==============================] - 298s 86ms/step - loss: 0.2080 - accuracy: 0.9363 - val_loss: 0.3453 - val_accuracy: 0.9175
Epoch 50/50
3480/3480 [==============================] - 302s 87ms/step - loss: 0.2031 - accuracy: 0.9380 - val_loss: 0.3441 - val_accuracy: 0.9178
```

plot result



overfitting 인 듯 합니다..

- 92% 의 정확도라니 딥러닝은 다르구나 하면서 probability도 확인해보니
- 전체 데이터 18.5만 개 중 사람 목소리가 12만 개여서 사람 목소리로 심하게 몰아서 추정해서 높은 정확도 인 것 같이 보입니다.

	index
0	
23	92848
21	29688
11	7418
13	6669
67	5213
...	...
62	102
44	78
57	71
58	54
41	32

- 라벨별 데이터 개수를 조정하고, 너무 세세히 나눠진 라벨은 병합시켜서 카테고리를 재구성할 필요가 있겠습니다.