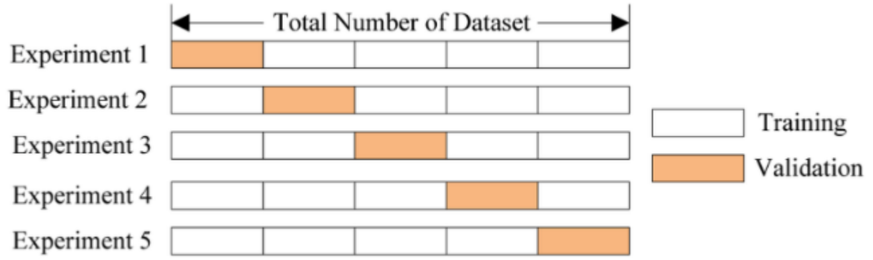
PE 프로그램 진행과정 (8.26)

01

K-Fold Cross Validation

- Stratified K-Fold Cross Validation
- 총 데이터의 수가 적을때 정확도를 향상시킬 수 있는 방법
- 각 데이터 셋이 K번 train/test set으로 사용됨
- target 속성값을 비슷하게 배치해
데이터가 편향되는 것을 방지
- cross_val_score 에서 자동으로
Stratified k-fold 방식 선택

K-Fold Cross Validation




```
model = KerasClassifier(build_fn=mlp_model_k, epochs=150, batch_size=32, verbose=0)
```

```
kfold = KFold(n_splits = 5, shuffle = True, random_state = 50)
results = cross_val_score(model, X, y, cv=kfold)
```

```
results, round(np.mean(results),3)
```

```
(array([0.82360947, 0.8196972 , 0.81391394, 0.82088792, 0.80370814]), 0.816)
```

MLP model (k = 5)
2 hidden layer
relu & dropout

Average accuracy = 0.816

PE 프로그램 진행과정 (8.26)

02

Various RNN model

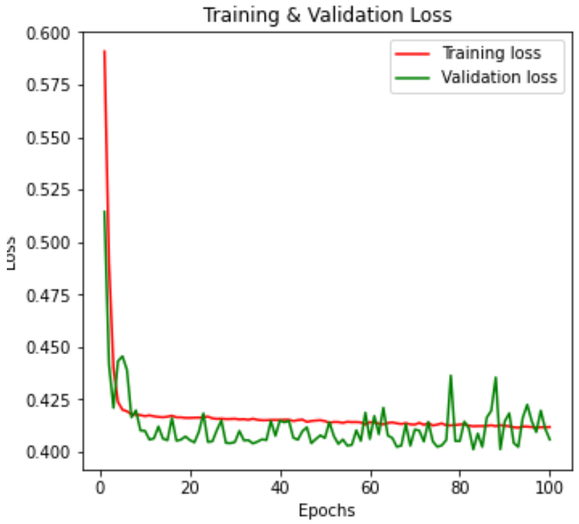
- **Simple RNN**
- Stacked RNN
- LSTM
- Deep LSTM
- Bidirectional LSTM
- GRU

1 X 13의 모양으로 24985개의 세트가 입력

```
def simple_rnn3():
    model = Sequential()
    model.add(SimpleRNN(32, input_shape = (1,13), return_sequences = False))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'Adam', metrics = ['acc'])

    return model
```




```
results = model3.evaluate(X_test, y_test)
print('Test accuracy: ', results[1])
```

```
138/138 [==============================] -
Test accuracy:  0.8156462311744469
```

*** 가장 기본적인 RNN model**

활성화 함수 : sigmoid

손실 함수 : binary crossentropy

최적화 함수 : Adam

MLP model보다 약간 개선된 loss & accuracy

Test accuracy = 0.82

PE 프로그램 진행과정 (8.26)

03

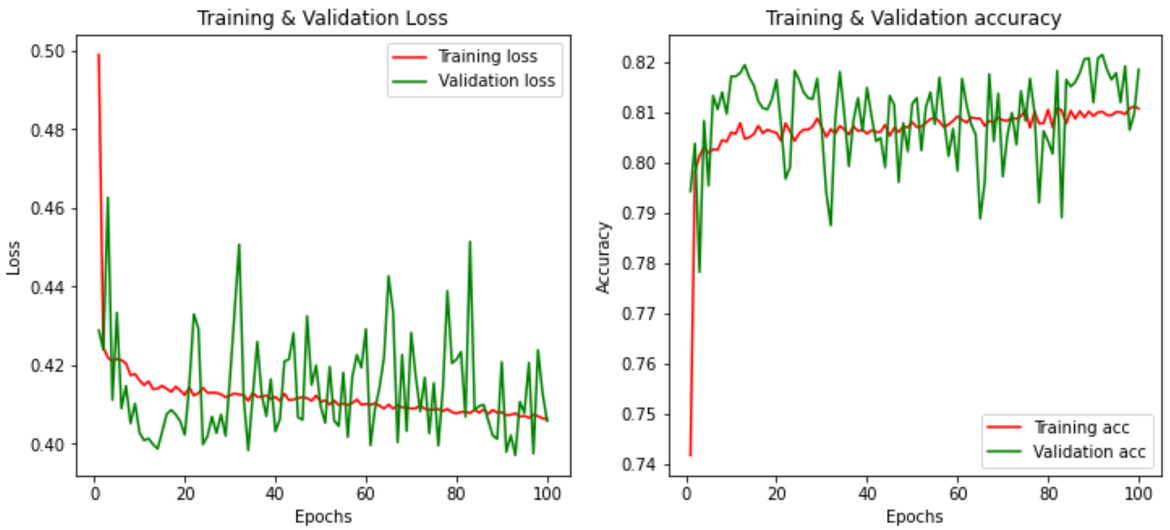
Various RNN model

- Simple RNN
- **Stacked RNN**
- LSTM
- Deep LSTM
- Bidirectional LSTM
- GRU

```
def simple_rnn4():
    model = Sequential()
    model.add(SimpleRNN(50, input_shape = (1,13), return_sequences = True))
    model.add(SimpleRNN(32, return_sequences = False))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'Adam', metrics = ['acc'])

    return model
```



*** 다중 RNN model**

활성화 함수 : sigmoid

손실 함수 : binary crossentropy

최적화 함수 : Adam

Simple RNN셀로 이루어진 layer을 쌓았지만

Overfitting 및 성능 저하

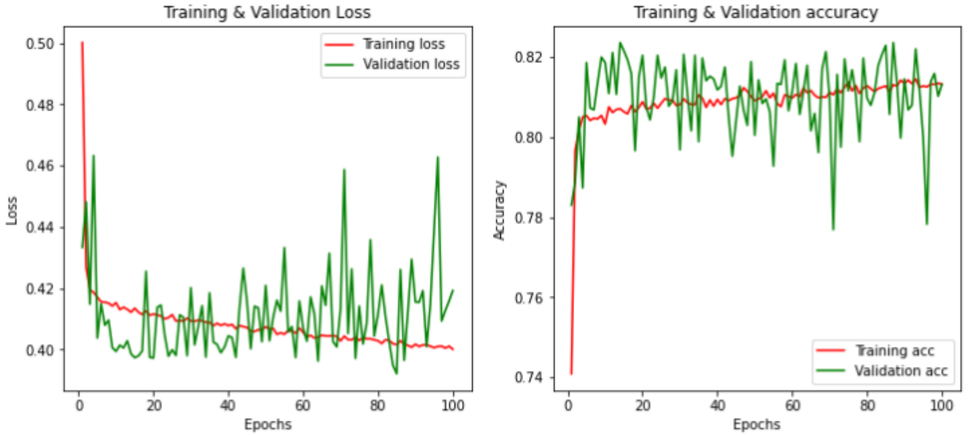
PE 프로그램 진행과정 (8.26)

04

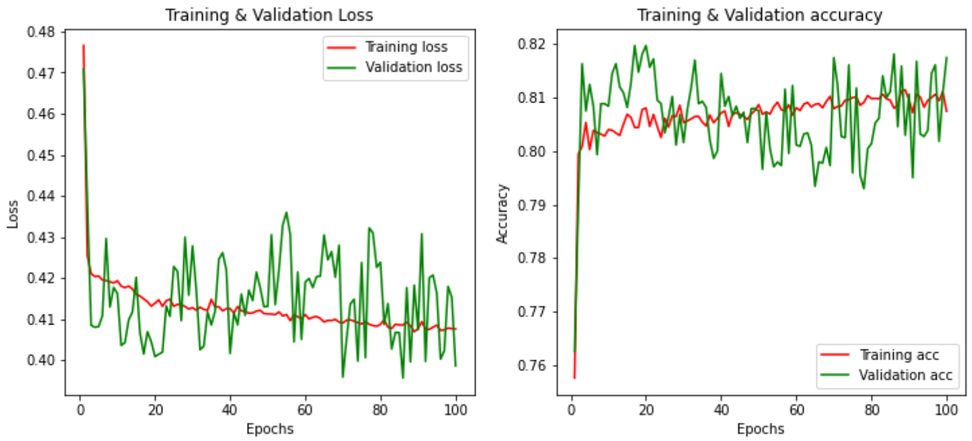
Various RNN model

- Simple RNN
- **Stacked RNN**
- LSTM
- Deep LSTM
- Bidirectional LSTM
- GRU

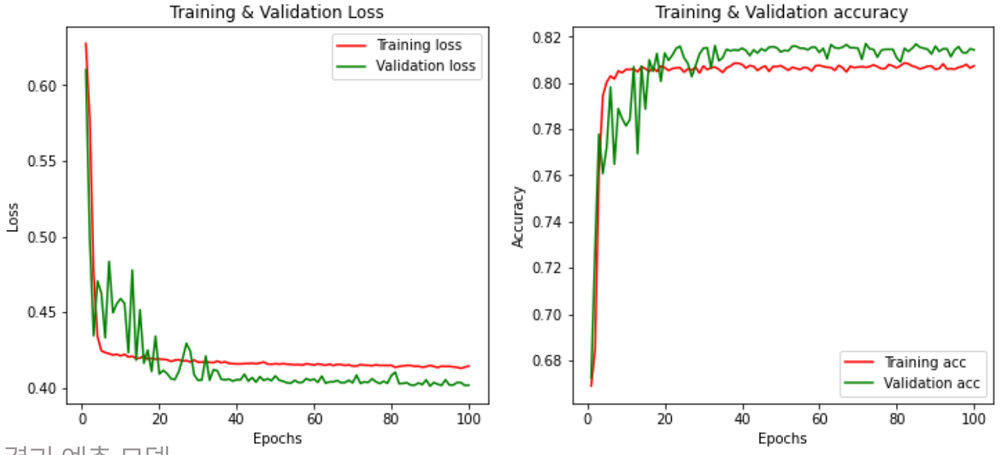
1. 활성화 함수 : relu & sigmoid



2. 활성화 함수 : tanh & sigmoid



3. 활성화 함수 : sigmoid



활성화 함수로 relu, tanh를 사용했을 때
너무나 불안정한 학습과정

sigmoid를 사용했을 때 비로소 안정적

하지만, 단층 RNN과 크게 달라지지 않은
Test accuracy = 0.81

PE 프로그램 진행과정 (8.26)

05

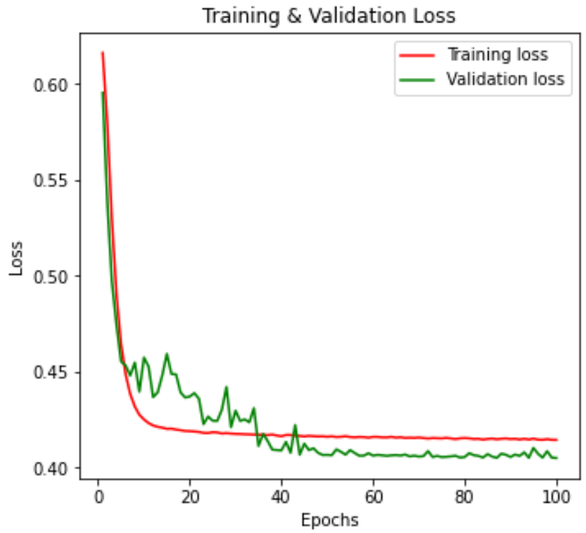
Various RNN model

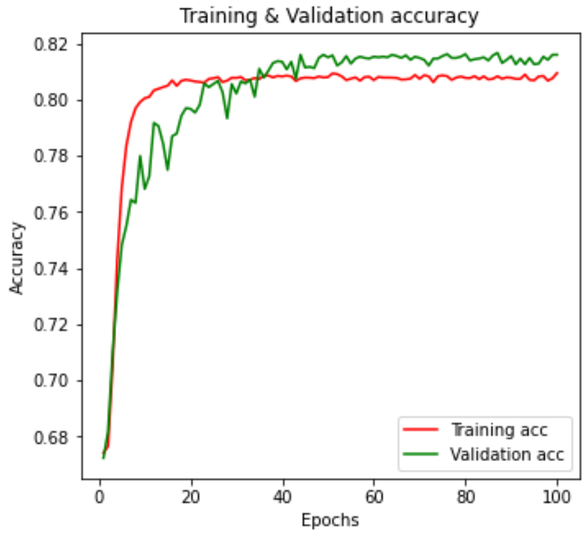
- Simple RNN
- Stacked RNN
- LSTM
- Deep LSTM
- Bidirectional LSTM
- GRU

```
def lstm3():
    model = Sequential()
    model.add(LSTM(50, input_shape = (1,13), activation='sigmoid'))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'Adam', metrics = ['acc'])

    return model
```






```
results = ltsm_model3.evaluate(X_test, y_test)
print('Test accuracy: ', results[1])
```

```
138/138 [==============================] - 0s 1
Test accuracy:  0.8160997629165649
```

*** LSTM model**

활성화 함수 : sigmoid

손실 함수 : binary crossentropy

최적화 함수 : Adam

40 epochs를 넘으면서 학습 안정화

Test accuracy = 0.82

PE 프로그램 진행과정 (8.26)

06

Various RNN model

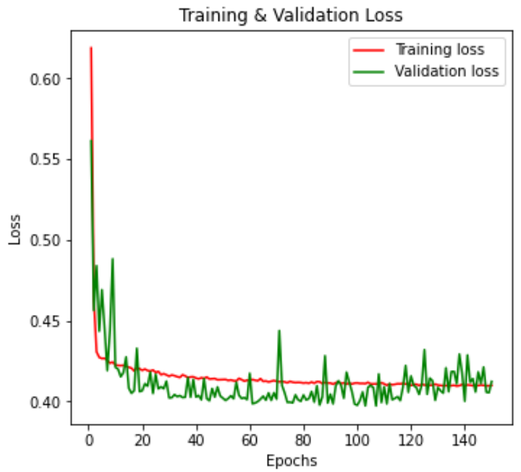
- Simple RNN
- Stacked RNN
- LSTM
- **Deep LSTM**
- Bidirectional LSTM
- GRU

다중 LSTM (layer : 2)

```
def lstm4():
    model = Sequential()
    model.add(LSTM(50, input_shape = (1,13), return_sequences = True, activation='sigmoid'))
    model.add(LSTM(50, return_sequences = False))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'Adam', metrics = ['acc'])

    return model
```

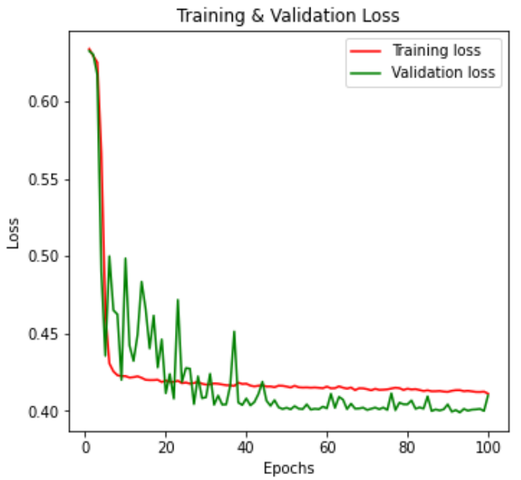


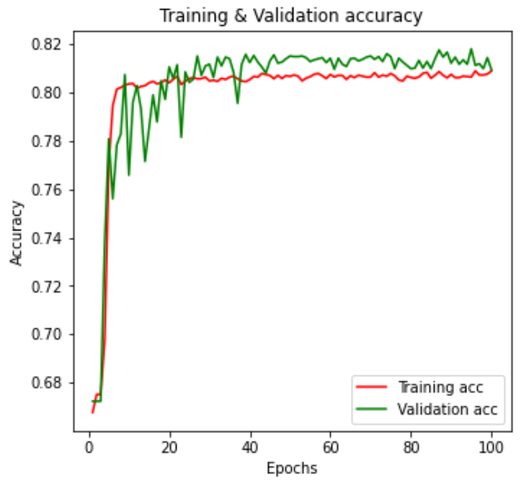
다중 LSTM (layer : 4)

```
def lstm7():
    model = Sequential()
    model.add(LSTM(40, input_shape = (1,13), return_sequences = True, activation='sigmoid'))
    model.add(LSTM(40, return_sequences = True, activation='sigmoid'))
    model.add(LSTM(30, return_sequences = True, activation='sigmoid'))
    model.add(LSTM(20, return_sequences = False, activation='sigmoid'))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'Adam', metrics = ['acc'])

    return model
```





*** Deep LSTM model**

단층 LSTM에 비해 오히려 낮은 성능

Test accuracy = 0.80

PE 프로그램 진행과정 (8.26)

07

Various RNN model

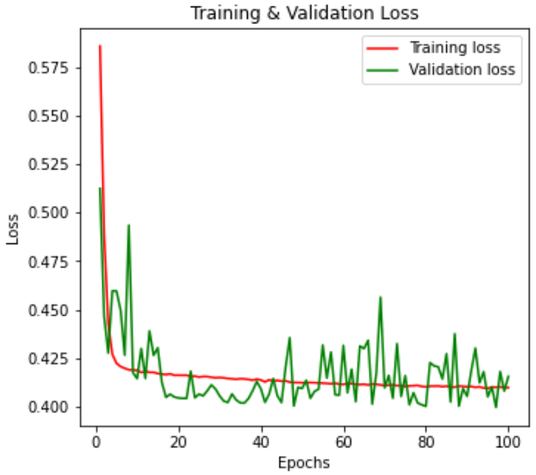
- Simple RNN
- Stacked RNN
- LSTM
- Deep LSTM
- **Bidirectional LSTM**
- GRU

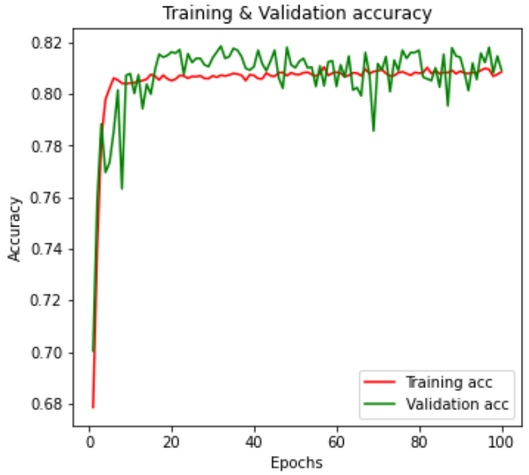
양방향 LSTM

```
def bidirectional_lstm1():
    model = Sequential()
    model.add(Bidirectional(LSTM(13, return_sequences = False), input_shape = (1,13)))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'rmsprop', metrics = ['acc'])

    return model
```



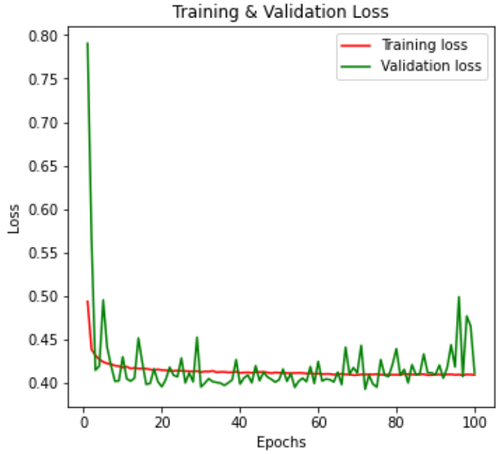


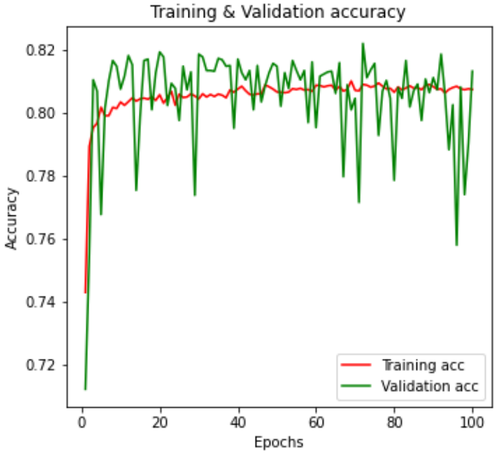
심층 양방향 LSTM

```
def bidirectional_lstm4():
    model = Sequential()
    model.add(Bidirectional(LSTM(40, return_sequences = True), input_shape = (1,13)))
    model.add(Bidirectional(LSTM(40, return_sequences = True)))
    model.add(Bidirectional(LSTM(40, return_sequences = False)))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'rmsprop', metrics = ['acc'])

    return model
```





*** (Deep) Bidirectional LSTM model**

layer을 추가하면 더욱 검증 결과가 악화

> 복잡한 구조의 인공신경망을 학습시킬 땐 여러가지 고려

Test accuracy = 0.80

PE 프로그램 진행과정 (8.26)

08

Various RNN model

- Simple RNN
- Stacked RNN
- LSTM
- Deep LSTM
- Bidirectional LSTM
- GRU

LSTM 보다 적은 수의 파라미터로 학습이 가능한 GRU

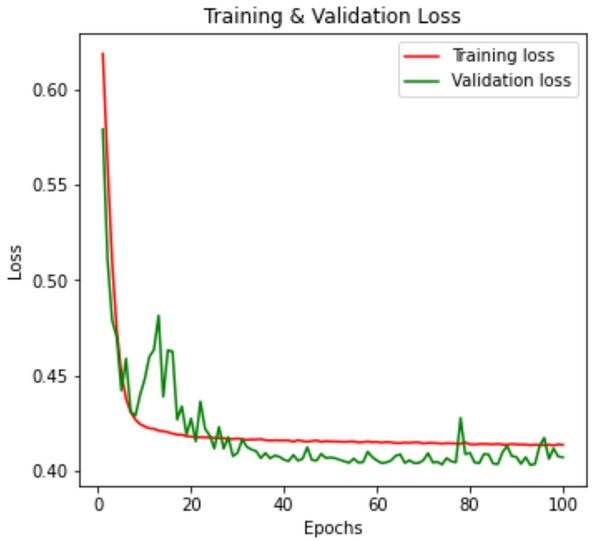
```
def gru1():
    model = Sequential()
    model.add(GRU(50, input_shape = (1,13), return_sequences = True, activation='sigmoid'))
    model.add(GRU(1, return_sequences = False, activation='sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'rmsprop', metrics = ['acc'])

    return model
```

```
results = gru_model1.evaluate(X_test, y_test)
print('Test accuracy: ', results[1])
```

```
138/138 [==============================] - 0s
Test accuracy:  0.8108843564987183
```



*** GRU model**

활성화 함수 : sigmoid

손실 함수 : binary crossentropy

최적화 함수 : rmsprop

Test accuracy = **0.81**

PE 프로그램 진행과정 (8.26)

09

Various RNN model

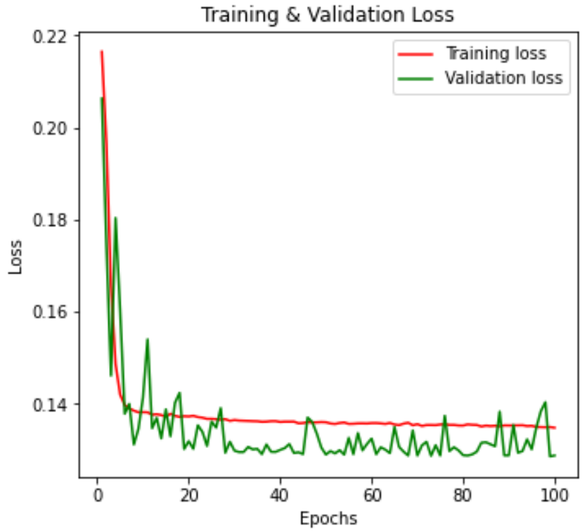
- Simple RNN
- Stacked RNN
- LSTM
- Deep LSTM
- Bidirectional LSTM
- GRU

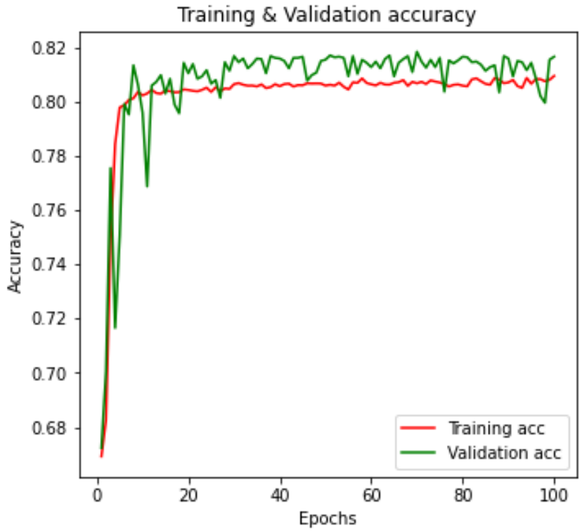
한 층의 layer을 더 쌓은 GRU

```
def gru4():
    model = Sequential()
    model.add(GRU(50, input_shape = (1,13), return_sequences = True, activation='sigmoid'))
    model.add(GRU(50, return_sequences = True, activation='sigmoid'))
    model.add(GRU(1, return_sequences = False, activation='sigmoid'))

    model.compile(loss = 'mean_squared_error', optimizer = 'rmsprop', metrics = ['acc'])

    return model
```






```
results = gru_model4.evaluate(X_test, y_test)
print('Test accuracy: ', results[1])
```

```
138/138 [==============================] - 0s
Test accuracy:  0.8165532946586609
```

*** Deep GRU model**

활성화 함수 : sigmoid

손실 함수 : mean_squared_error

최적화 함수 : rmsprop

Test accuracy = **0.82**