Ai Paper and Project

PAPER

[graduated thesis] :

Advanced model prediction by hyperparameter optimization in chemistry

PROJECT

- **3D Camera project**
- **Mass-Spectrum Prediction**

Seung Jin Lee

PAPER

[graduated thesis] :

Advanced model prediction by hyperparameter optimization in chemistry

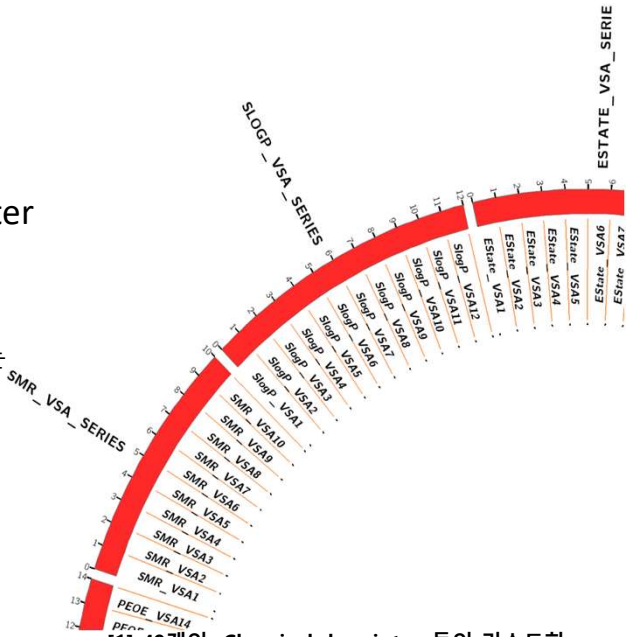
(Lab2. 2020.10 – 2022.02)

요약 : 신약개발, 단백질 도킹, 농약 및 독 등에 사용되어지는 용해도(solubility)를 예측하는 것을 기존의 방법론들 보다 R2 점수 기준으로 0.2-0.3 이상으로 올라가게 되었으며 제일 높은 점수는 0.991이라는 성과를 올렸습니다.

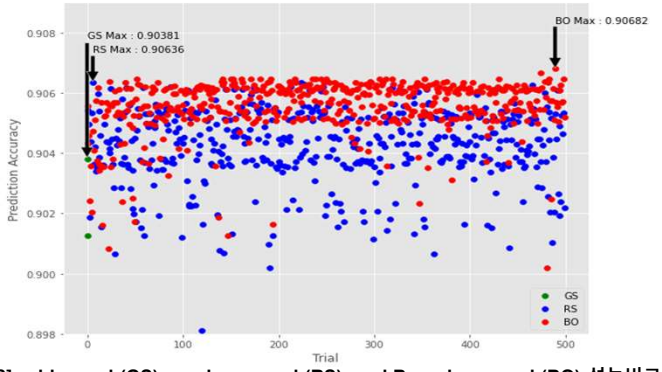
방법론(AFO) : 2가지 접근방식으로 계산 또는 분석화학에서 사용되어지는 대표적인 **49**개의 chemical descriptors(특징)들을 입력데이터에 반영하여 성능을 최대한 높이는 것을 탐색. 여기서 사용되어지는 방법론은 Search method의 Bayesian Optimization을 이용하지만 surrogate model을 TPE(Tree-Parzen Estimator)를 이용하여 Gaussian보다 높은 처리속도와 성능을 이용.
두번째 방법에서는 똑같은 TPE-Bayesian Optimization을 이용한 model의 hyperparameter optimization을 이용하여 성능의 극대화를 시도합니다.

→ Search method optimization을 위해서 Grid Search (GS) / Random Search (RS) / Bayesian Optimization (BO)을 비교. 이후에는 BO에서 Tree-Parzen Estimator(TPE) 환경으로 반영하여 기존보다 빠른 속도로 후보 값을 검색 (Guassian Process는 $O(n^3)$ 자원 요구)

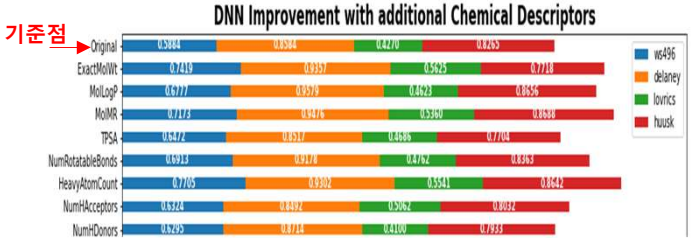
language : Python, C++, CUDA
Library : tensorflow, pytorch, scikit-learn, sqlite, postgresql, RDKit, optuna



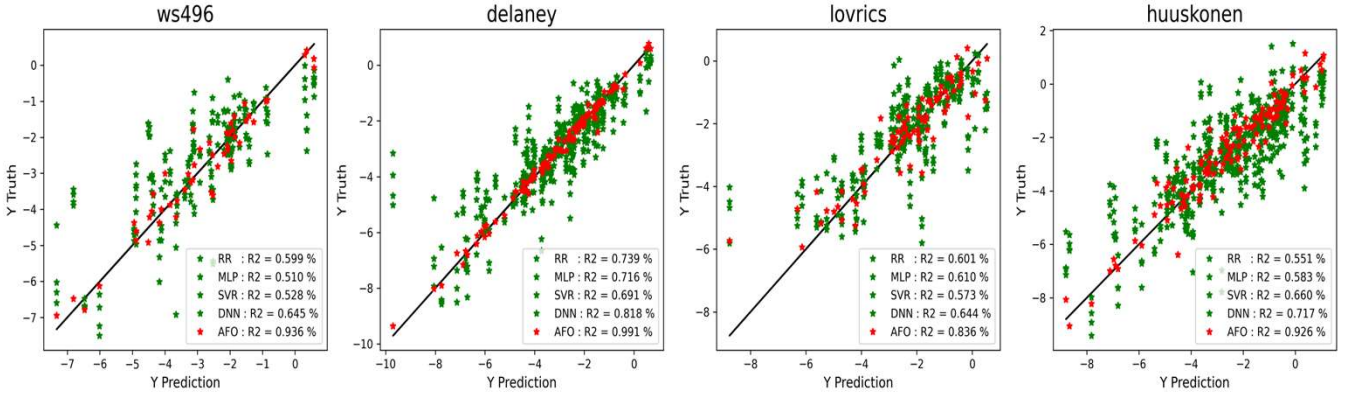
[1] 49개의 Chemical descriptors들의 리스트화



[2] grid search(GS), random search(RS), and Bayesian search(BO) 성능비교



[3]입력데이터에 개별적으로 chemical descriptor들을 반영한 결과



[4] 서로 다른 데이터들의 각 예측 결과. AFO(our method, 빨강색 점)가 모든 데이터에서 최고점수를 얻을 것을 알 수 있음.

PROJECT

❖ 3D Camera project

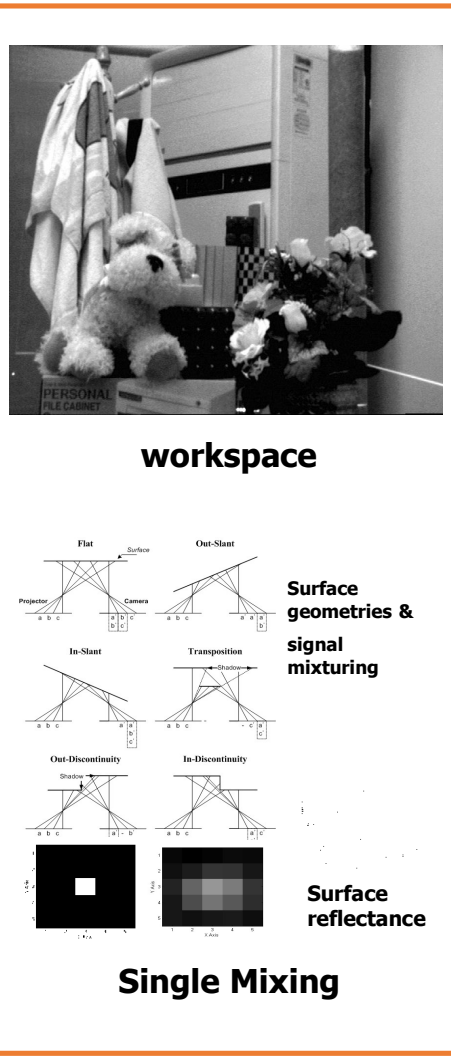
(Lab1. 2018.10 – 2019.06)

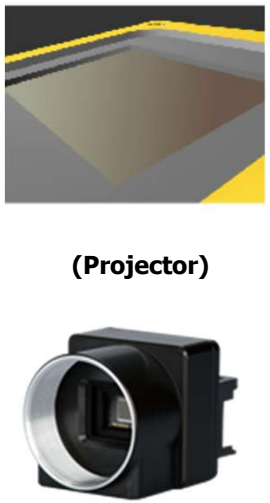
요약 : 3D카메라의 성능을 높이기 위해서 정확도와 계산속도를 높이는 것을 책임지게 되었으며, 기존 error calibration 0.097에서 **0.081**로 개선, 계산시간 37sec에서 **24sec**로 개선

방법론 : 3D카메라 기법은 2개의 카메라를 이용한 겹치는 현상을 이용한 촬영기법. 여기서 겹치는 부분에서 생기는 **error calibration**을 최소화하는 작업이 필요. 개선방안으로 **Hierarchical Orthogonal** 코덱의 불필요한 loop문 및 register설정 단순화 과정과 OpenCV와 OpenGL의 forced fixed value를 부여하여 error rate를 축소. 추가로 Nvidia GPU CUDA를 이용하여 시간을 더 축약이 가능해짐.

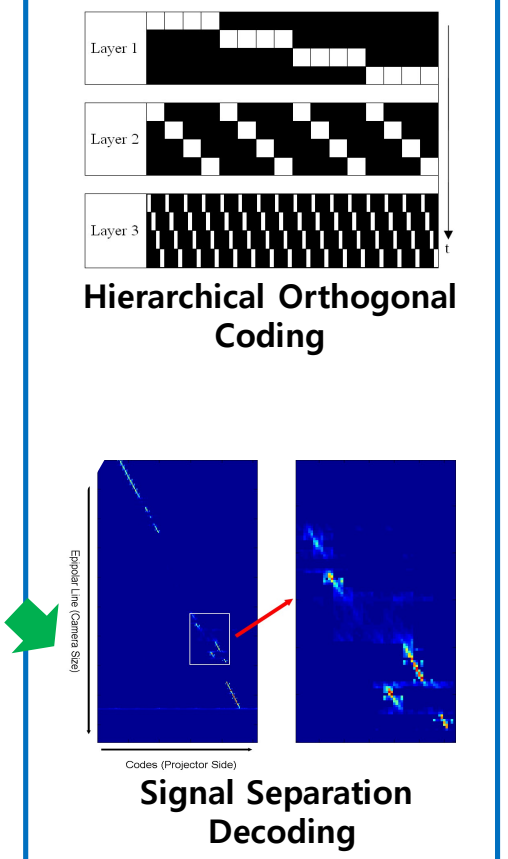
language : Python, C++, CUDA

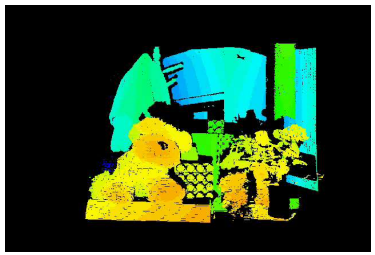
Library : OpenCV, OpenGL, Camera SW





CODEC algorithm





3D Point Cloud

PROJECT

❖ Mass-Spectrum Prediction

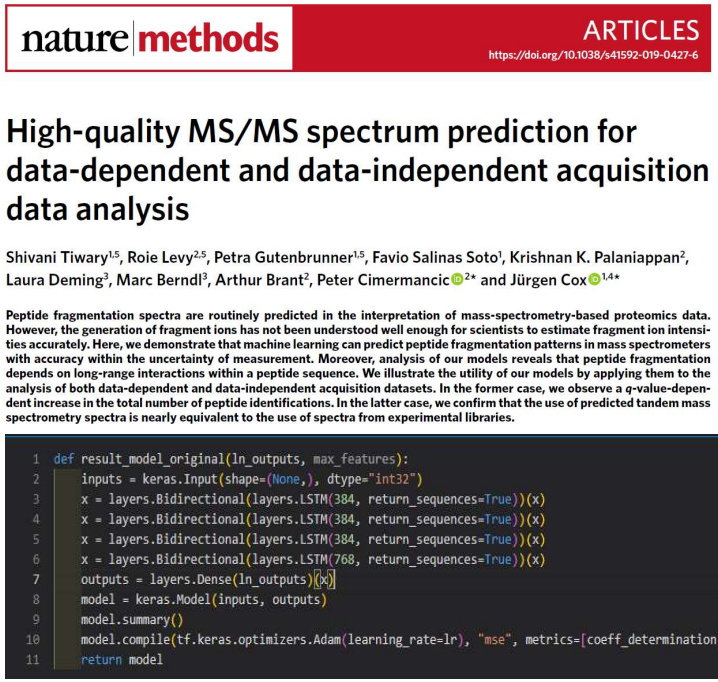
(2022.05 – 2022.06)

요약 : Mass Spectrum (MS) 데이터를 이용한 모델 예측도를 구현하여 최신 논문의 평가지수인 PCC 0.957 보다 높은 0.998를 구현.

방법론 : 데이터 x에서는 주어진 protein sequence 기준으로 추가적인 메타데이터를 포함시켜 학습을 시키고, y값은 주어진 x값에 따라 분자들의 강도 (mass intensity)를 매칭시켜 학습을 시키게 됩니다.

최상위 논문의 모델에서는 Bi-LSTM를 이용한 encoding방법으로 접근하여 대용량에 따른 sequence학습에 유용하게 접근을 하였으나, 여기서 최신기법으로 Attention algorithm, hyperparameter optimization 기법을 추가하여 기존 모델의 가능성을 높이는 것과 동시에 데이터의 quality를 높이는 작업으로 feature들에 대한 분석을 시도합니다. 덕분에 이러한 작업을 토대로한 여러 시도 끝에 기존보다 더 높은 점수인 PCC 0.998를 얻게 되었습니다.

language : Python, C++, CUDA
Library : tensorflow, pytorch, scikit-learn, sqlite, optuna



nature methods ARTICLES
https://doi.org/10.1038/s41592-019-0427-6

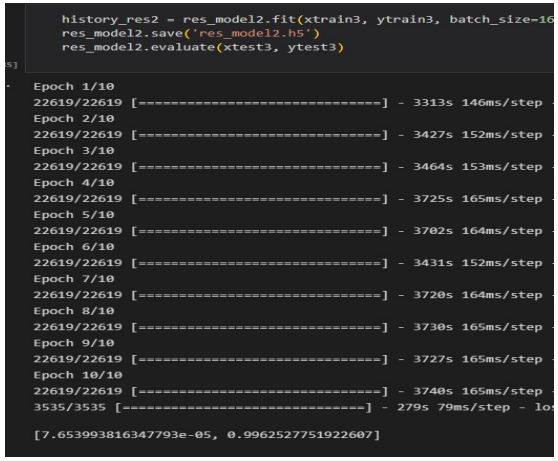
High-quality MS/MS spectrum prediction for data-dependent and data-independent acquisition data analysis

Shivani Tiwary[1,5], Roie Levy[2,5], Petra Gutenbrunner[1,5], Favio Salinas Soto[1], Krishnan K. Palaniappan[2], Laura Deming[3], Marc Berndl[3], Arthur Brant[2], Peter Cimermancic[2*] and Jürgen Cox[1,4*]

Peptide fragmentation spectra are routinely predicted in the interpretation of mass-spectrometry-based proteomics data. However, the generation of fragment ions has not been understood well enough for scientists to estimate fragment ion intensities accurately. Here, we demonstrate that machine learning can predict peptide fragmentation patterns in mass spectrometers with accuracy within the uncertainty of measurement. Moreover, analysis of our models reveals that peptide fragmentation depends on long-range interactions within a peptide sequence. We illustrate the utility of our models by applying them to the analysis of both data-dependent and data-independent acquisition datasets. In the former case, we observe a q-value-dependent increase in the total number of peptide identifications. In the latter case, we confirm that the use of predicted tandem mass spectrometry spectra is nearly equivalent to the use of spectra from experimental libraries.

```python
def result_model_original(ln_outputs, max_features):
    inputs = keras.Input(shape=(None,), dtype="int32")
    x = layers.Bidirectional(layers.LSTM(384, return_sequences=True))(x)
    x = layers.Bidirectional(layers.LSTM(384, return_sequences=True))(x)
    x = layers.Bidirectional(layers.LSTM(384, return_sequences=True))(x)
    x = layers.Bidirectional(layers.LSTM(768, return_sequences=True))(x)
    outputs = layers.Dense(ln_outputs)(x)
    model = keras.Model(inputs, outputs)
    model.summary()
    model.compile(tf.keras.optimizers.Adam(learning_rate=lr), "mse", metrics=[coeff_determination])
    return model
```

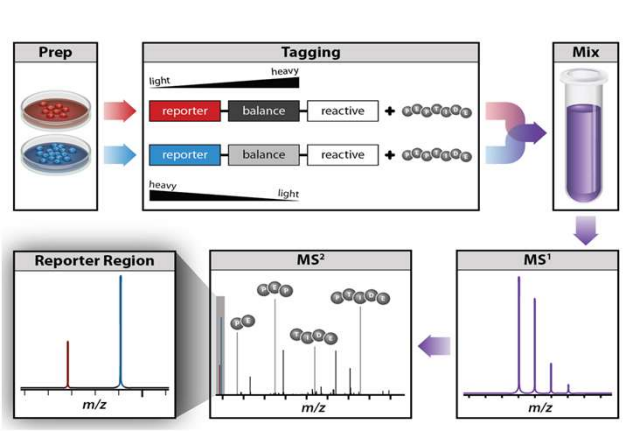
[1] 목표로 한 상위급 논문과 구현

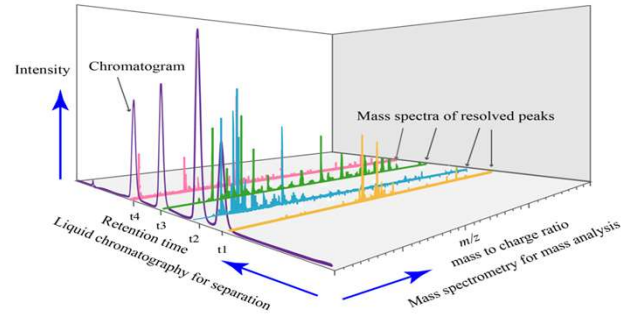


```
history_res2 = res_model2.fit(xtrain3, ytrain3, batch_size=16
res_model2.save('res_model2.h5')
res_model2.evaluate(xtest3, ytest3)

Epoch 1/10
22619/22619 [==============================] - 3313s 146ms/step
Epoch 2/10
22619/22619 [==============================] - 3427s 152ms/step
Epoch 3/10
22619/22619 [==============================] - 3464s 153ms/step
Epoch 4/10
22619/22619 [==============================] - 3725s 165ms/step
Epoch 5/10
22619/22619 [==============================] - 3702s 164ms/step
Epoch 6/10
22619/22619 [==============================] - 3431s 152ms/step
Epoch 7/10
22619/22619 [==============================] - 3720s 164ms/step
Epoch 8/10
22619/22619 [==============================] - 3730s 165ms/step
Epoch 9/10
22619/22619 [==============================] - 3727s 165ms/step
Epoch 10/10
22619/22619 [==============================] - 3740s 165ms/step
3535/3535 [==============================] - 279s 79ms/step - lo

[7.653993816347793e-05, 0.9962527751922607]
```

R2 score evaluation (with test data) : **0.99625, (PCC 0.998)**

[3] 모델 최적화와 전처리 과정 간소화 결과





Given data :

x = (**modified sequences**, retention time, mass to charge ratio)

y = (relative**) Intensity**

[2] MS 실험과정과 분석되는 데이터의 시각화 그리고 입력데이터의 x와 y값