# 졸업 논문

2020학년도 2학기 재료공학부 재료종합실험

**학 번** : 2017 – 16621

**이 름** : 안 소 현

**신청강좌** : 재료종합실험

**강좌번호** : 001 **담당교수명** : 김 기 범

**실험제목** : AI 기반 이차전지 소재 탐색 연구

**실험실명** : Advanced Energy Materials Lab

**실험실 지도 교수명** : 강 기 석 (인)

# < 초 록 >


스마트폰, 전기자동차의 발달에 따라 고용량, 고에너지밀도를 가진 배터리에 대한 수요가 증가하고 있다. 그러나 기존 Li ion 배터리의 대표적인 양극 소재 물질인 LCO($LiCoO_2$ )와 같은 전이금속 산화물의 경우, 전이 금속을 이용함에 따라 specific capacity의 한계가 존재할 수밖에 없고 전이 금속으로 인한 유해성을 무시할 수 없다. 또한 이 전극 물질들의 합성 과정에서 고온이 요구되므로 $CO_2$ 배출 또한 문제가 되고 있다. 음극 소재로 사용되는 흑연의 경우 안정적인 구조를 유지할 수 있으나 6 C atoms당 1 Li atom을 수용할 수 있으므로 energy density 측면에서 불리하다.

이러한 기존 배터리의 단점을 보완할 수 있는 차세대 배터리 전극 물질로서 유기물 전극 물질이 연구되고 있다. 유기물 전극 물질은 산화·환원에도 구조적 안정성을 유지하는 유기물을 배터리 소재로 사용하는데, 유기물의 구조를 tuning함으로써 energy density와 theoretical specific capacity를 조절할 수 있고, 상온에서 합성이 가능하므로 $CO_2$ 배출을 감소시킬 수 있다.

이러한 유기물 배터리를 차세대 배터리로서 사용하기 위해서는 적합한 소재를 탐색하는 것이 필수적이다. 유기물 분자 구조와 배터리 소재로서 성능의 상관관계를 파악하는 것은 이를 위한 한 방법이 될 수 있다. 배터리 소재로서의 가장 중요한 성능 중 하나인 전압을 결정짓는 소재의 물성으로 IE(Ionization Energy)와 EA(Electronic Affinity)가 있다. 이를 파악하기 위해서 경험론적인 방법과 DFT(Density Functional Theory)와 같은 이론적 방법이 이용되지만, 전자의 경우 한정된 화학종에 대해서만 연구가 진행되었고 후자의 경우 calculation cost가 매우 크다.

유기물 분자 구조로부터 IE와 EA를 파악하는데 있어서 ML(Machine Learning) 기법을 활용한다면 분자 구조로부터 배터리의 성능을 파악하는 것이 단순화되므로 위 한계를 해결해줄 수 있다. 즉, 유기물 분자 구조에 대한 정보만으로 폭넓은 chemical space에서 후보군 물질을 선정하는 것이 가능해진다.

ML 기법은 FCNN(Fully Connected Neural Network)를 이용하고 Supervised learning을 통해 ML을 진행한다. 이 때, 유기물 분자 구조를 computer-understand vector로 만들기 위해 ECFPs(Extended-Connectivity Fingerprints)를 이용한다.

또한 FCNN Training에 있어, 유기물 분자에 대한 label은 DFT 계산 결과에 기반한 database를 이용하고 k-fold Cross Validation을 실행할 것이다. 다양한 FCNN model에 대해 Training과 Validation에 대한 결과를 분석하여 Validation에 대한 loss값

이 최소화된 model을 얻어내고자 한다. 이후 전체 dataset과 k-fold Cross Validation 기법을 통해 Training시킨 FCNN을 가지고 유기물 분자들에 대한 Inference를 진행하여 IE와 EA값을 예측해본 뒤, 이에 대한 오차를 확인해볼 계획이다.

**주요어 : 유기물, Ionizaion Energy, Electron Affinity, DFT, Machine Learning, FCNN**

# < 본문 목차 >

# < 표 목차 >

# < 그림 목차 >

# 1. 서론

## 1.1. 연구 배경

스마트폰, 전기자동차의 발달에 따라 고용량, 고에너지밀도를 가진 배터리에 대한 수요가 증가하고 있다. 그러나 기존 Li ion 배터리의 양극 소재로 사용되고 있는 대표적인 물질인 LCO($LiCoO_2$ )와 같은 전이금속 산화물의 경우, 무거운 전이 금속 원소를 이용함에 따라 specific capacity의 한계가 존재할 수밖에 없고 전이 금속으로 인한 유해성을 무시할 수 없다. 또한 이 전극 물질들의 합성 과정에서 고온이 요구되므로 합성 과정에서의 $CO_2$ 배출 또한 문제가 되고 있다. 음극 소재로 사용되는 흑연의 경우 안정적인 구조를 유지할 수 있다는 장점이 있으나 6 C atoms당 1 Li atom을 수용할 수 있으므로 energy density 측면에서 불리하다.

이러한 기존 배터리의 단점을 보완할 수 있는 차세대 배터리 전극 물질로서 유기물 전극 물질이 연구되고 있다. 유기물 전극 물질은 산화·환원에도 구조적 안정성을 유지하는 유기물을 배터리 소재로 사용하는데, 유기물의 구조를 tuning함으로써 energy density와 theoretical specific capacity를 조절할 수 있고, 상온에서 합성이 가능하므로 $CO_2$ 배출을 감소시킬 수 있다.

이러한 유기물 배터리를 차세대 배터리로서 사용하기 위해서는 적합한 소재를 탐색하는 것이 필수적이다. 따라서 유기물 분자 구조와 배터리 소재로서 성능의 상관관계를 파악해야 한다. 배터리 소재로서의 가장 중요한 성능 중 하나인 전압을 결정짓는 소재의 물성으로  IE(Ionization Energy)와 EA(Electronic Affinity)가 있다. 이를 파악하기 위해서 경험론적인 방법과 DFT(Density Functional Theory)와 같은 이론적 방법이 이용되지만, 전자의 경우 한정된 화학종에 대해서만 연구가 진행되었고 후자의 경우 calculation cost가 매우 크다.

유기물 분자 구조로부터 IE와 EA를 파악하는데 있어서 ML(Machine Learning) 기법을 활용한다면 분자 구조로부터 배터리의 성능을 파악하는 것이 단순화되므로 위 한계를 해결해줄 수 있다. 즉, 유기물 분자 구조에 대한 정보만으로 후보군 물질을 선정하는 것이 가능해지므로 기존의 방법보다 증대된 효율을 기대할 수 있다.

## 1.2. 연구 목적

ML 기법을 이용하여 유기물 분자 구조로부터 IE와 EA를 파악하는 ANN을 구축

한다. 이로 인해 유기물 분자 구조로부터 배터리 후보군 물질을 선정하는 것이 가능해지도록 하는 것이 목적이다. 이를 위해 ANN의 경우, FCNN(Fully Connected Neural Network)을 이용하고 Supervised learning을 통해 ML을 진행할 계획이다. 이때, 유기물 분자 구조를 computer-understand vector로 만들기 위해 ECFPs(Extended-Connectivity Fingerprints)를 이용한다.

또한 FCNN Training에 있어, 유기물 분자에 대한 label은 DFT database를 이용하고 k-fold Cross Validation을 실행할 것이다. 다양한 FCNN model에 대해 Training과 Validation에 대한 결과를 분석하여 Validation에 대한 loss값이 최소화된 model을 얻어내고자 한다. 이후 전체 dataset과 k-fold Cross Validation 기법을 통해 Training시킨 FCNN을 가지고 유기물 분자들에 대한 Inference를 진행하여 IE와 EA값을 예측해본 뒤, 이에 대한 오차를 확인해볼 계획이다.

# 2. 이론적 배경

## 2.1. Machine Learning[1)]

### 2.1.1. Machine Learning의 의미

ML이란 computer에게 explicit programming을 하지 않고 learning할 수 있는 능력을 갖게 하는 방법론이다. 이는 다양한 data set에서 특정 문제를 위한 패턴을 찾아내는 일반적인 알고리즘이 존재한다는 아이디어에서 시작되었다. 이를 이용하여 유기물 분자 구조로부터 IE와 EA를 파악하는데 있어, calculation cost가 큰 DFT와 같은 이론적 방법 대신 ML 기법을 활용한다면 분자 구조로부터 배터리의 성능을 파악하는 것이 단순화시킬 수 있게 된다. 이러한 ML은 크게 supervised learning, unsupervised learning, 그리고 reinforcement learning으로 분류되는데 본 연구에서는 supervised learning 기법을 활용한다.

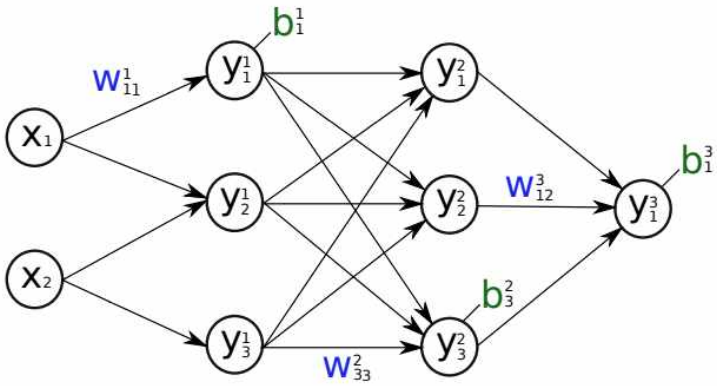


그림 1 A fully-connected feed-forward neural network

### 2.1.2. Neural Network의 다양한 특성

[그림 1]은 Artificial Neural Network의 대표적인 예시인 fully-connected feed-forward neural network의 예시를 나타내며 2개의 hidden layer와 1개의 output layer로 구성되어있다. Node(y)는 원으로 나타내었으며 화살표는 information의 방향과 더불어

1) John-Anders Stende, 『Constructing high-dimensional neural network potentials for molecular dynamics』, Faculty of Mathematics and Natural Sciences University of Oslo, September 2017, pp.19-43.

neuron들의 connection을 의미하기도 한다. 각 connection은 weight 값을 가지고 있으며 각각의 node들은 output y와 bias b로 나타내어져 있다. hidden neuron들은 NN(Neural Network)의 functional form을 나타내기 위한 목적으로 존재한다.

같은 layer안의 node들이 모두 같은 activation function을 갖고 있다고 하면 layer l 안의 node i 에 대해 output 값은 다음과 같다.

$$y_i^l = f_l(\sum_{j=1}^{N_{l-1}} w_{ij}^l y_j^{l-1} + b_i^l)$$

어떤 layer의 모든 node에 대해 output 값이 계산되었다면 그 다음 layer에 대한 계산도 마찬가지로 수행하면 된다. 이를 표현하면 아래와 같다.

$$y_1^{l+1} = f_{l+1}[\sum_{j=1}^{N_l} w_{1j}^3 f_l(\sum_{k=1}^{N_{l-1}} w_{jk}^2 f_{l-1}(\cdots f_1(\sum_{n=1}^{N_0} w_{mn}^1 x_n + b_m^1)\cdots) + b_k^2) + b_1^3]$$

이를 통해 NN에서 independent variable은 오직 input value인 $x_n$이 유일함을 알 수 있다.

이러한 NN의 특성을 결정하는 중요한 요인 중 하나는 activation function(f)이다. FCNN model의 경우 activation function은 다음과 같은 조건을 만족하여야 한다.

1. Non-constant
2. Bounded
3. Monotonically-increasing
4. Continuous

위 조건을 모두 만족하는 몇 가지 non-linear function들에 대해 살펴보겠다. 우선, sigmoid function이라고 하는 것이 있으며 수식으로는 다음과 같이 표현된다.

$$f(x) = \frac{1}{1+e^{-x}}$$

이외에도 [그림 2]에서 볼 수 있듯, hyperbolic tangent($f(x) = \tanh(x)$), rectifier function($f(x) = \max(0, x)$), relu function 등과 같은 다양한 activation function들이 존재한다.

다음으로는 전반적인 Model Training에 대해 살펴보도록 하겠다. NN은 network에게 data feeding을 반복적으로 수행하는 과정을 통해 learning이 이루어진다. NN은

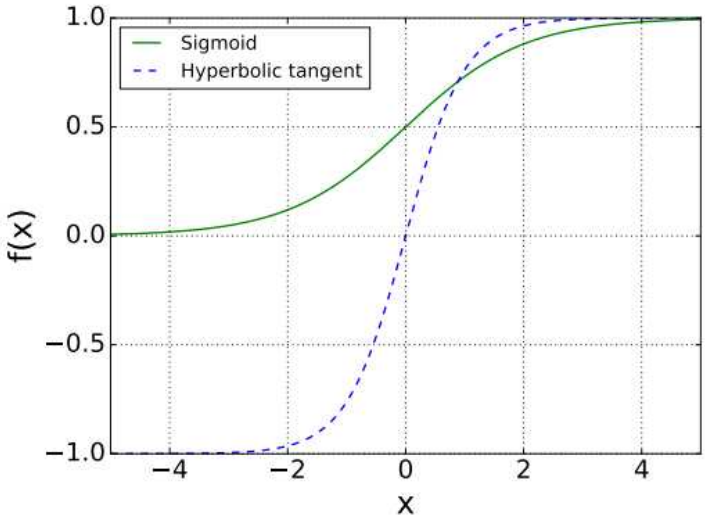


그림 2 Two of the most common activation functions for neural networks

learning 알고리즘을 통해 weight와 bias와 같은 parameter들을 조절함으로써 data 안의 pattern을 찾고 정확도를 향상시키게 된다. 우리의 목적은 분자의 구조를 나타내는 input data(ECFPs)를 통해 IE 또는 EA 를 output으로 mapping하는 function을 나타내는 NN을 구축하는 것이다. 즉, random하게 초기화된 weight와 bias값들을 반복적으로 수정하면서 원하는 output값을 낼 때까지 값을 조정하여 error를 최소화하는 것을 “Training”이라고 한다. 그리고 Training에 대한 결과의 척도인 error는 loss function 또는 cost function이라고 하는 것에 의해 정의된다. NN에서의 standard loss function은 Mean Square Error로, 다음과 같다.

$$\Gamma = \frac{1}{2N}\sum_{i=1}^{N}(Y_i - y_i)^2$$

$N$은 Training input의 개수, $Y$는 desired output, $y_i$는 Training example에 대해 NN에 의해 예측된 값을 의미한다. 더불어, Mean Absolute Error 는 모든 절대 오차의 평균으로 다음과 같다.

$$\Gamma = \frac{1}{N}\sum_{i=1}^{N}|Y_i - y_i|$$

Activation function뿐만 아니라 NN을 대표하는 또 다른 특성에는 optimization 알고리즘이 있다. Loss function을 최소화하기 위한 NN parameter set을 얻기 위해서 다양한 알고리즘이 사용된다. 다양한 Update rule 중에서 특히나, NN 연구에서 널리 사

용되는 gradient descent-based methods 중에는 momentum, adagrad, adadelta, adam 등이 있다. 이들 모두 적절한 learning rate을 찾는 것이 중요하며, 만약 learning rate이 너무 작으면 convergence가 매우 느려지고 learning rate이 너무 크면 loss function이 minimum 근처에서 fluctuation하거나 심지어는 diverge를 초래할 수 있다.

위에서 설명한 Training 알고리즘을 간단하게 정리하면 다음과 같으며 NN을 학습시키기 위해 이를 반복한다.

1. n개의 Training example들의 set을 input으로 지정한다.
2. 각각의 Training example에 대해 ($\vec{x} = X_{i*}$)

(a) input layer를 initialization한다.

$$\overrightarrow{y_0} = \vec{x}$$

(b) forward 방향으로 propagation하고 $\overrightarrow{u_l}$과 $\overrightarrow{y_l}$을 저장한다.

(For $l = 1, ..., L+1$)

$$\overrightarrow{u_l} = W_l \overrightarrow{y_{l-1}} + \vec{b_l}$$

$$\overrightarrow{y_l} = f_l(\overrightarrow{u_l})$$

(c) output layer에서 error를 계산하고 저장한다.

$$\overrightarrow{\delta_{L+1}} = \vec{Y} - \overrightarrow{y_{L+1}}$$

(d) error를 backpropagation하고 $\vec{\delta_l}$을 저장한다. (For $l = 1, ..., L+1$)

$$\vec{\delta_l} = \vec{y_l} \odot (W_{l+1}^T \overrightarrow{\delta_{l+1}})$$

(e) weight와 bias gradient를 계산하고 저장한다. (For $l = 1, ..., L+1$)

$$G_{l,i} = \vec{\delta_l} \overrightarrow{y_{l-1}^T}, \ \overrightarrow{H_{l,i}} = \vec{\delta_l}$$

3. 모든 Training example i에 대해 total gradient를 계산한다.

$$G_l = \sum_i^n G_{l,i} \text{ and } \overrightarrow{H_l} = \sum_i^n \overrightarrow{H_{l,i}}$$

4. 선택한 update rule에 의해 weight와 bias를 update한다.

## 2.2. Extended-Connectivity Fingerprints[2)]

ECFPs란 분자의 특성을 표현하는 topological fingerprints라고 할 수 있다. 특히나 분자의 structure-activity modeling에 중점을 두고 개발된 fingerprints이다. ECFPs는 특정 분자에 대해 빠르게 계산될 수 있으며  무수히 많은 다른 분자 특성을 나타낼

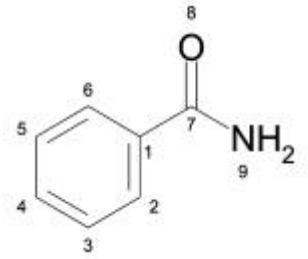


그림 3 Benzoic acid amide atom numbering (of non-hydrogen atoms)

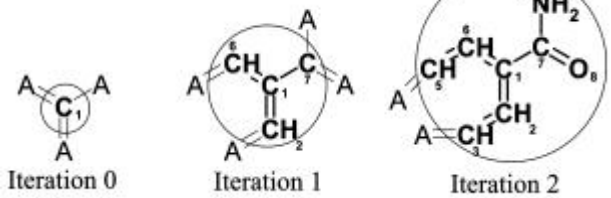


그림 4 Illustration of the effect of iterative updating on the information represented by an atom identifier

수 있다. 분자의 구조로부터 ECFPs로 변환되는 과정은 다음과 같다.

1. Initial assignment stage

분자 내의 각 원자들은 integer identifier를 할당받는다. 예를 들어, 각 원자들은 원자 번호를 할당받을 수 있다. 이 initial atom identifier들은 initial fingerprint set이 된다.

2. Iterative updating stage

각각의 원자 identifier들은 각각의 이웃 원자들의 identifier를 반영하도록 update된다. 이 때, 이웃 원자들은 order-dependence를 피하기 위해 그들의 identifier들과 attaching bond의 순서를 통해 ordering된다. Hash function은 ECFPs가 이전과 다른 single-integer identifier의 array로 되돌아가는 것을 막아준다. Update되면서 분자 내의 모든 atom들이 새로운 identifier들을 생성해내면, old identifier들은 new identifier로 대체된다. 그리고 이러한 새로운 identifier들은 fingerprint set에 추가된다. 본 단계에서는 이러한 iteration이 사전에 정해진 횟수만큼 반복된다.

3. Duplicate identifier removal stage

정해진 횟수의 iteration이 모두 완료되면, 반복된 identifier들은 제거되고 남아있는

2) David Rogers and Mathew Hahn, 『Extended-Connectivity Fingerprints』, J. Chem. Inf. Model., 50, 742-754, February 4, 2010, pp.742-754.

integer identifier set이 ECFPs가 된다.

이러한 모든 과정이 [그림 3]과 [그림 4]에 잘 요약되어있다. [그림 3]은 benzoic acid amide의 atom numbering 예시를 나타내며 atom 1에 대한 iteration 효과에 대한 결과가 [그림 4]에 나타나 있다. [그림 4]에서 볼 수 있듯이, iteration 0에서 initial atom identifier들은 각각의 atom과 attached bond만을 나타낸다. Iteration 1이 되면 atom 1 바로 옆의 이웃 atom들에 대한 정보까지 포함된다. Iteration 2가 되면 이 분자의 aromatic ring 상당 부분뿐만 아니라 amide group에 대한 정보를 포함할 수 있게 된다.

## 2.3. Ionization Energy and Electron Affinity of Organic Molecules[3)]

Organic molecule들의 IE(Ionization Energy)와 EA(Electron Affinity)는 physico-chemical 성질의 고유한 fundamental이라고 할 수 있다. 따라서 organic electronics materials의 연구에서 energy level을 결정하는 데 있어 중요한 특성이다.
IE (EA) 는 neutral N-electron system과 positively-(negatively-)charged N−1 (N+1)-electron system 간의 energy 차이 크기를 의미한다. 이러한 IE와 EA는 각각 direct photoemission과 inverse photoemission technique들을 통해 측정되고는 했다. 즉, sample에 주입된 hole이나 electron의 energy를 측정했던 것이다. 이러한 방법에서 나아가, Enrico Fermi가 기본적인 이론을 도입하고 Walter Kohn이 발전시킨 DFT라는 computational quantum mechanical modelling은 분자 내부에 전자가 들어있는 모양과 그 에너지를 양자역학으로 계산하기 위한 이론의 하나이다. DFT를 통해 어떤 분자가 세상에 존재할 수 있는지 없는지의 여부, 특정 분자의 모양과 성질, IE, EA 등을 예측할 수 있다. 컴퓨터를 사용하는 과학 계산들 중에서, 가장 널리 쓰이는 양자역학 계산 분야 중 하나이나 calculation cost가 매우 크다는 단점이 있다.

3) Susumu Yanagisawa, 『Determination of the ionization energy and the electron affinity of organic molecular crystals from first-principles: dependence on the molecular orientation at the surface』, Department of Physics and Earth Sciences, Faculty of Science, University of the Ryukyus, 1 Senbaru, Nishihara, Okinawa 903-0213, Japan, November 27, 2019, pp.1-9.

## 2.4. Redox Flow Battery 구성 요소와 Screening 시 고려해야 할 점

RFB(Redox Flow Battery)는 기존 2차 전지와는 달리 electrolyte 내의 active material이 산화-환원되어 충·방전되는 시스템으로, 전기에너지를 electrolyte의 화학적 에너지로 저장시키는 전기화학적 축전장치이다. 실제 전기 화학적 반응은 stack에서 일어나고 electrolyte를 유체펌프를 이용하여 stack 내부에 지속적으로 순환시킴으로써 작동한다. Active material로 사용되는 Redox pair로는 V/V, Zn/Br, Fe/Cr, Zn/air 등이 있는데 이 중 V/V, Zn/Br Redox pair가 가장 널리 사용되고 있다.

양극과 음극 모두 V을 사용한 RFB를 VRFB라고 하며, 이를 가지고 RFB의 기본적인 구조 및 충·방전 원리를 살펴보겠다. 아래 [그림 5]와 같이, 전지 충전 시에 stack 내부의 양극에서는 V 4가 이온이 5가 이온으로 산화되고, stack 내부의 음극에서는 V 3가 이온이 2가 이온으로 환원되며 방전 시에는 이와 반대로 양극에서는 환원 반응, 음극에서는 산화 반응이 일어난다. 이러한 RFB는 크게 stack과 electrolyte, 그리고 electrolyte를 순환시키기 위한 pump로 구성된다. 추가적으로 양극과 음극에서 전자를 외부로 전달하기 위한 current collector가 존재한다.[4)]

RFB의 음극, 양극 소재를 select하고자 할 때, 고려해야할 사항은 다음과 같다.

1. 다른 구성 요소와의 반응성
2. Solubility in electrolyte
3. 전압

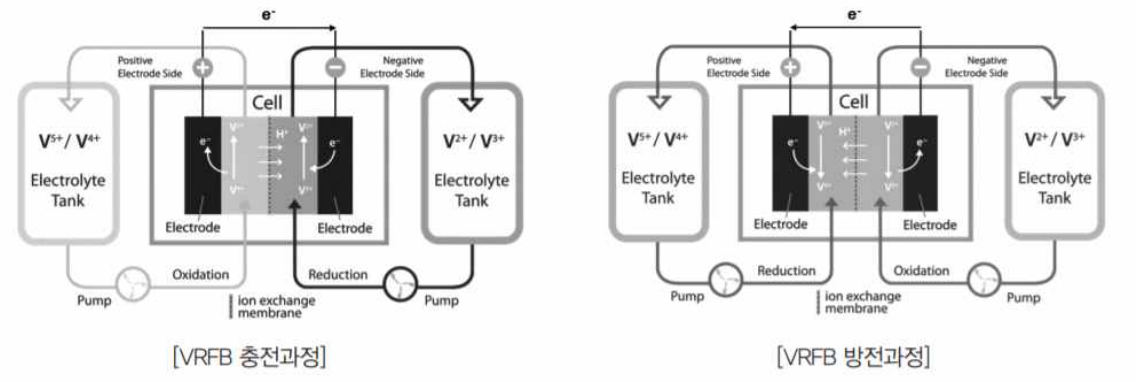


그림 5 VRFB의 충전 및 방전과정

3번 전압의 경우를 살펴보면, 배터리가 높은 에너지 저장 능력을 갖추기 위해서

4) 한 신, 김유종, 허지향, 『Vanadium Redox Flow Battery 개발 및 국내 실증』, Journal of the Electric World / Monthly Magazine, Special Issues 4, June, 2014, p. 50.

는 음극재의 potential은 낮고, 양극재의 potential은 높아야 한다. 하지만 이것만 고려하여 무조건적으로 전압을 높일 수 없다. 이 외에도 electrolyte에 대한 solubility를 고려해야하며, 배터리의 다른 구성 요소와 반응해서는 안 된다. 이러한 것들을 종합적으로 고려하여 potential window를 설정하고 그 안에서 높은 전압을 가질 수 있도록 음극/양극 소재를 screening해야 한다. 현재 RFB에서 가장 널리 쓰이는 Redox pair인 V/V와 Zn/Br은 독성이 존재하는 원소들이므로 그 위험성을 무시할 수 없다. Redox pair를 유기물 분자로 대체한다면, 독성 문제를 해결할 수 있고 다양한 유기물들이 존재하므로 전압 tunability가 증가할 뿐만 아니라 가격이 저렴해진다는 장점이 존재한다. 따라서 본 연구에서는 음극/양극 소재의 후보 물질로서 유기물을 기반으로 screening한다.

# 3. 연구 방법

모든 coding은 python 언어 기반으로 연구를 진행했으며 ML관련 tool은 tensorflow.keras[5]를 이용하여 구현하였다.

## 3.1. Ionization Energy와 Electron Affinity data set 수집

DFT database로부터 IE와 EA의 data set을 수집한다. 이후 유기물 분자를 SMILES(Simplified Molecular Input Line Entry System, 분자구조 문자열 표현식) 로 변환해준다. Pandas tool을 이용하여 유기물 분자 각각에 대해 SMILES와 IE, EA를 정리한 표를 제작한다. 이 표는 추후 ANN Training에 이용할 data set이 된다. NN에서의 input data는 유기물 분자의 ECFPs가 되어야하므로 SMILES로부터 ECFPs로 변환해주어야 하는데, 이를 위해 RDKIT package를 이용하였다. [6]

## 3.2. ANN Model 구축

IE와 EA 각각에 대해 적절한 ANN Model을 구축한다. 이를 위해서 다양한 ANN Model에 대해, 위 data set의 일부를 이용하여 Training과 Validation을 반복한 뒤, MAE(Mean Absolute Error) 값이 가장 작게 나오는 ANN Model을 select한다. 다양한 ANN Model에 대한 변인으로는 radius의 수, ECFP의 bit 수, layer 수, 각 layer에서의 neuron의 수 등이 있다. ANN Model이 Training data set에 특화되어 성능이 높아지지 않고 오히려 감소하는 overfitting을 방지하기 위해, L2 Regularization, Drop out을 추가적으로 진행해준다. 이 때, l2_value = 0.003, dropout_rate = 0.1, learning_rate = 0.003, batch_size = 30000, epochs = 5000, activation function은 relu function을 이용하며 k-fold split의 수는 2로 하여 진행하였다.

표 1 다양한 ANN Model

| dir_name | radius | n_bits | layer 1 | layer 2 | layer 3 |
|---|---|---|---|---|---|
| 0 | 1 | 1024 | 1024 | 512 | 256 |
| 1 | 1 | 1024 | 512 | 256 | 128 |
| 2 | 1 | 1024 | 1024 | 512 | |
| 3 | 1 | 1024 | 1024 | 256 | |

5) Martin Abadi et al., 『TensorFlow: Large-scale machine learning on heterogeneous systems』,Software available from tensorflow.org, 2015.

6) Open-Source cheminformatics : http://www.rdkit.org

| | | | | | |
|---|---|---|---|---|---|
| 4 | 2 | 1024 | 1024 | 512 | 256 |
| 5 | 2 | 1024 | 512 | 256 | 128 |
| 6 | 2 | 1024 | 1024 | 512 | |
| 7 | 2 | 1024 | 1024 | 256 | |
| 8 | 3 | 1024 | 1024 | 512 | 256 |
| 9 | 3 | 1024 | 512 | 256 | 128 |
| 10 | 3 | 1024 | 1024 | 512 | |
| 11 | 3 | 1024 | 1024 | 256 | |
| 12 | 1 | 2048 | 1024 | 512 | 256 |
| 13 | 1 | 2048 | 512 | 256 | 128 |
| 14 | 1 | 2048 | 1024 | 512 | |
| 15 | 1 | 2048 | 1024 | 256 | |
| 16 | 2 | 2048 | 1024 | 512 | 256 |
| 17 | 2 | 2048 | 512 | 256 | 128 |
| 18 | 2 | 2048 | 1024 | 512 | |
| 19 | 2 | 2048 | 1024 | 256 | |
| 20 | 3 | 2048 | 1024 | 512 | 256 |
| 21 | 3 | 2048 | 512 | 256 | 128 |
| 22 | 3 | 2048 | 1024 | 512 | |
| 23 | 3 | 2048 | 1024 | 256 | |
| 24 | 1 | 1024 | 512 | 256 | |

## 3.3. ANN Model Optimization 및 Training & Validation

IE에 대해서는 dir2, EA에 대해서는 dir24 model을 select했으며 전체 data set을 가지고 Training과 Validation을 진행해준다. 이 때, k-fold split의 수를 8로 하고 나머지 조건은 2번 과정과 같으나 dropout_rate = 0.1로 해주었다. 관련 python code는 부록에 첨부해 놓았다. k-fold Cross Validation 결과를 보고 Validation Mean MAE 값이 가장 낮은 경우를 최종 model로 select한다.

## 3.4. Inference & ANN Model을 이용한 Screening

최종적으로 select한 model과 전체 dataset을 가지고 각각 IE, EA 값을 inference 해본다. 이후 learning curve를 plotting해보고 $R^2$, MAE, standard deviation 값들을 구해본다. 신뢰도를 고려하여, 정해 놓은 potential window를 가지고 양극, 음극 물질을 screening한다.

# 4. 연구 결과 및 논의

## 4.1. Inference 결과

IE, EA dataset은 materialsproject.org의 molecular explorer에서 2020/08/04에 기재된 것을 수집했다. k-fold Cross Validation 후, 가장 학습이 잘 된 model을 기준으로 최종 model을 select했다. 최종 model을 가지고 inference를 진행한 결과는 아래 표와 같다.

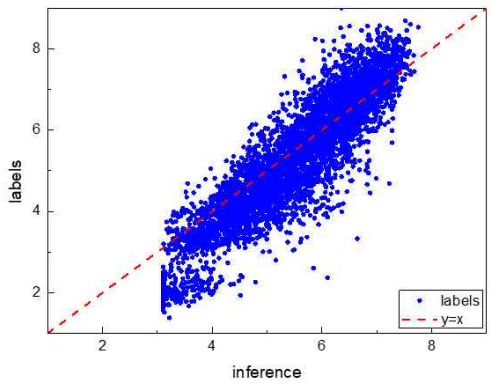


그림 6 IE Inference results

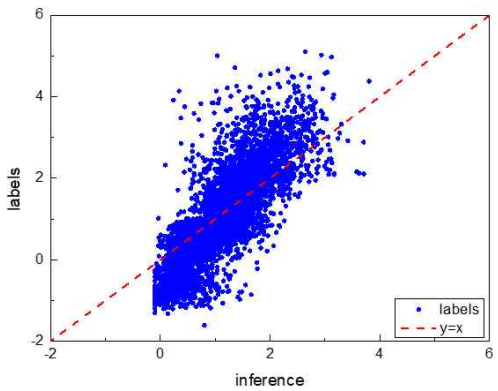


그림 7 EA Inference results

표 2 Inference results

| | IE | EA |
|---|---|---|
| $R^2$ | 0.723 | 0.445 |
| MAE(eV) | 0.51 | 0.528 |
| standard deviation | 0.452 | 0.478 |

학습된 model이 label값을 정확하게 추론하는지 나타내는 $R^2$값은 IE 예측 model의 경우 0.723, EA 예측 model의 경우 0.445로 나타났다.([표 4]) 학습된 model의 IE와 EA의 Inference 결과를 살펴보면, IE의 경우가 EA의 경우보다 더 잘 예측했음을 확인할 수 있었다. 한편으로 MAE는 0.5 eV로 작지 않은 값이 얻어졌다. 이 원인에 대해서는 2가지를 생각해 볼 수 있다. 우선, 본 연구에서 학습시키는 데에 사용한 input data는 산화/환원되지 않은 분자의 ECFPs로, 산화/환원되지 않은 분자의 정보만으로는 산화/환원되는 과정에서 얻어지는 IE/EA를 학습하기가 어려웠을 수 있다. 그러나 분자의 IE와 EA가 각각 HOMO, LUMO와 깊은 연관이 있다는 Koopmans' Theorem을 생각해 보았을 때 이 원인이 주요한 원인은 아닐 것이다. 다른 이유로는 학습 model의 구조, 학습 과정에서의 hyperparameter의 적합성을 생각해 볼 수 있다. [그림 6]과 [그림 7]에서 학습된 model은 특정 값 이하의 IE/EA로는 추론을 하지 못한 것을 확인할 수 있었다. l2 regularizer, dropout rate 등 hyperparameter를 보다 최적화한다면 이러한 문제를 해결할 수 있을 것으로 기대한다.

## 4.2. Screening

배터리의 음극, 양극 소재를 screening할 때 potential window를 결정짓는 요소에는 여러 가지가 존재하는데, 그 중 전극 구성 요소와 전해질 구성 요소, 이 두 가지 요소가 결정적인 역할을 한다.

배터리 전해질의 solvent로 많이 사용하는 전해질에는 DMC(dimethyl carbonate), DEC(diethyl carbonate), EC(ethylene carbonate), 그리고 PC(propylene carbonate) 등이 있다. 그 중 RFB에서 종종 사용되는 solvent인 ACN(Acetonitrile)을 기준으로 screening을 진행하였다. ACN의 redox potential은, reduction 0.44V, oxidation이 6.54V이다.($VS.\,Li/Li^+$) 또한 일반적인 배터리에서, 양극의 current collector로 사용하는 금속은 Al이며 이는 4.7V ($VS.\,Li/Li^+$)에서 산화가 일어난다. 따라서 양극의 경우 최대 oxidation potential은 4.7V가 된다. 즉 potential window는, 음극의 경우 0.44V, 양극의 경우는 4.7V가 된다.

표 3 ACN의 Redox potential

| $E^0$ of ACN | Reduction (V) | Oxidization (V) |
|---|---|---|
| $VS.\,SHE$ | -2.6 | 3.5 |
| $VS.\,Li/Li^+$ | 0.44 | 6.54 |
| Absolute | 1.68 | 7.78 |

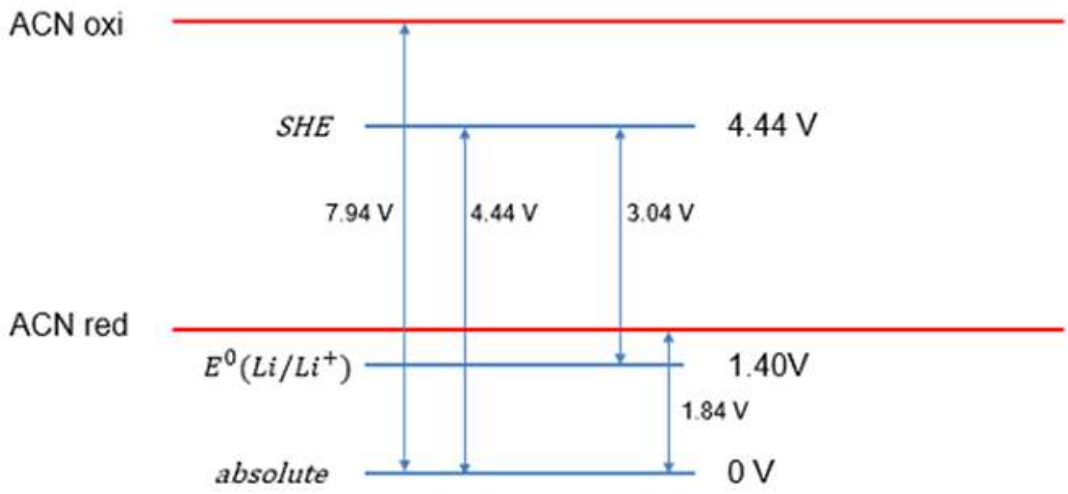


그림 8 ACN의 Redox potential

어떤 물질의 IE − 1.40V는 그 물질의 oxidation potential ($VS.Li/Li^+$) 이 된다. Inference 결과를 참고하여, 오차가 정규분포를 따른다고 가정했을 때 신뢰도 95%로 추정하면 1.41 V의 uncertain한 영역이 생긴다. 대부분의 물질이 ACN과 함께 사용 가능하다. 하지만 높은 에너지 효율을 위해선 최대한 높은 voltage 영역에 있는 물질을 사용하는 것이 유리할 것이다. 이러한 관점에서 3.0V~3.29V 사이의 범위에서 oxidation이 일어나는 물질만을 screening 한다면 2758개의 물질이 screening 결과가 얻어진다.([그림 9])

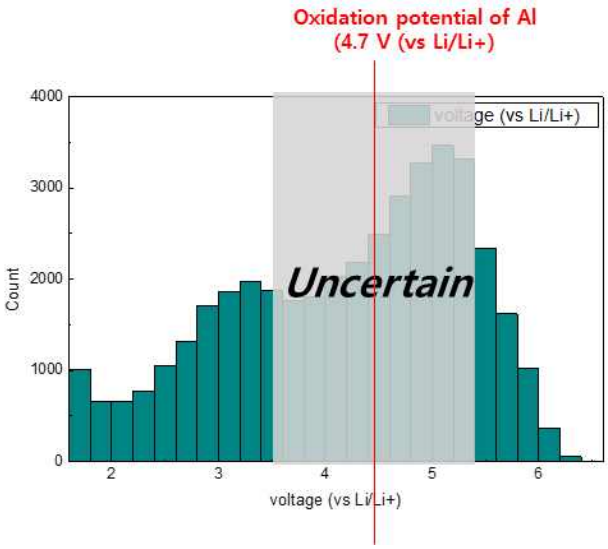


그림 9 IE Screening results

어떤 물질의 EA − 1.40V는 그 물질의 reduction potential ($VS.Li/Li^+$) 이 된다.[7], [8] Inference 결과를 참고하여, 오차가 정규분포를 따른다고 가정했을 때 신뢰도 95%

7) Y. Ding et al., Chem. Soc. Rev. 47, 69, 2018.

로 추정하면 1.484 V의 uncertain한 영역이 생긴다. 이 영역 바깥에서 ACN과 함께 사용할 수 있는 물질은 33개이다.([그림 10])

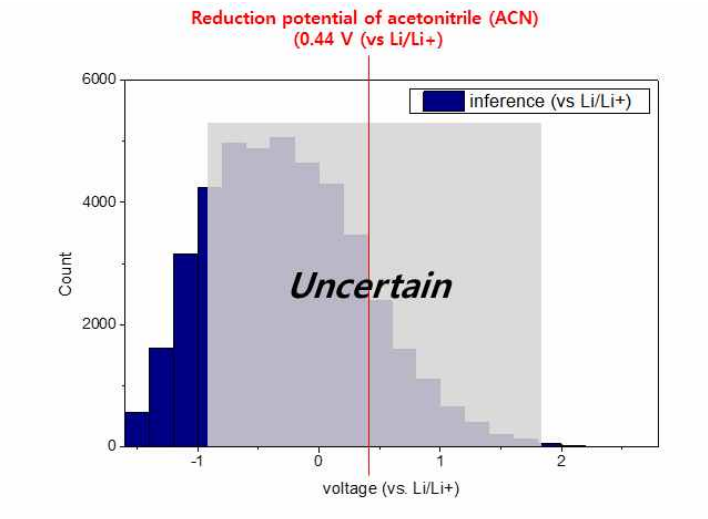


그림 10 EA Screening results

이처럼 양극 후보군으로서 screening된 물질은 3~3.29 V 영역에서 2758개, 음극 후보군으로서 screening된 물질은 1.8 V이상 영역에서 33개였다. 현재 organic battery system에서 연구되고 있는 물질들의 전압 영역대가 양극에서 2 V 후반~ 3 V 초반, 음극에서 1 V 중반~ 2 V 초반임을 생각해 보았을 때[9] 현재 연구되고 있는 물질과 유사하거나 더 좋은 에너지 효율을 기대할 수 있는 분자들을 screening할 수 있었다. 또한 본 연구에서는 전해질 용매를 ACN을 가정하여 screening을 진행하였는데 다른 용매를 사용하는 경우에도 용매의 redox potential에 따라 screening을 진행할 수 있을 것으로 기대된다.

8) N. Dardenne et al., J. Phys. Chem. C 119, 23373, 2015.
9) Sechan Lee†, Giyun Kwon†, Kyojin Ku†, Kyungho Yoon, Sung-Kyun Jung, Hee-Dae Lim, Kisuk Kang, 『Recent progress in organic electrodes for Li and Na rechargeable batteries』, JAdvanced Materials, Vol 30, 1704682, 2017.

# 5. 결론 및 논의

기존 Li 배터리의 단점을 극복하기 위한 유기물 배터리를 차세대 배터리로 사용하기 위해서는 적합한 소재를 탐색해야 한다. 그 중 배터리의 전압을 결정짓는 소재의 물성으로 IE와 EA를 파악하는데 있어 ML기법을 응용해보았다. ML 기법을 이용하여 유기물 분자 구조로부터 IE와 EA를 파악하는 ANN을 구축하는 것이 최종 목표였다.

이를 위해 먼저 DFT database를 통해 유기물 분자들에 대한 IE/EA 값을 수집했고 다양한 후보 ANN model들 중 Training & Validation을 반복하고 hyperparameter optimization을 진행하며 MAE 값이 가장 낮은 model을 select했다. 이후 각각 model에 대해 k-fold Cross Validation을 진행하여 MAE 값이 가장 낮은 최종 model을 select한 뒤, 이 model에 대해 전체 dataset을 가지고 Inference를 진행하였다. 최종적으로, RFB에서 solvent가 ACN일 때 음극/양극으로 사용될 수 있는 유기물 분자들을 screening하였다.

그 결과, $R^2$값은 IE 예측 model의 경우 0.723, EA 예측 model의 경우 0.445로 나타났다. 즉, IE의 경우가 EA의 경우보다 더 잘 예측했음을 확인할 수 있었다. 한편으로 IE/EA 모두 MAE는 약 0.5 eV로 작지 않은 값이 얻어졌다.

$R^2$값이 1보다 작게 나오고 MAE 값이 다소 큰 이유는 다음과 같이 분석할 수 있었다. Input data는 산화/환원되지 않은 분자의 ECFPs인 반면, IE/EA는 분자가 산화/환원되는 과정에서 얻어지기 때문에 다소 차이가 존재할 수 있다. 하지만 Koopmans' Theorem을 생각해 보았을 때 이 원인이 주요한 원인은 아닐 것이고, ANN의 hyperparameter optimization이 충분히 이루어지지 않은 것이 영향을 미쳤을 것이다. 학습된 model은 특정 값 이하의 IE/EA로는 추론을 하지 못한 것이 그 증거이다. L2 regularizer, dropout rate 등 hyperparameter optimization을 충분히 진행해주면 이러한 문제를 해결할 수 있을 것이다.

Inference 결과를 참고하여, 오차가 정규분포를 따른다고 가정했을 때 신뢰도 95%로 추정하고, ACN과 함께 사용할 수 있는 물질을 screening하였다. Potential window는, 음극의 경우 0.44V, 양극의 경우는 4.7V였으므로 이를 기반으로 screening하면 양극 후보군으로서 screening된 물질은 3~3.29 V 영역에서 2758개, 음극 후보군으로서 screening된 물질은 1.8 V이상 영역에서 33개였다.

현재 organic battery system에서 연구되고 있는 물질들의 전압 영역대를 고려해보았을 때, 현재 연구되고 있는 물질과 유사하거나 더 좋은 에너지 효율을 기대할 수

있는 분자들을 screening할 수 있었다.

본 연구는 ML을 통해 유기물 분자의 IE/EA를 파악할 수 있다는 가능성을 제기했다는 점에서 큰 의의가 있다. 하지만 uncertain region이 꽤나 광범위하고 screening된 물질 후보군들은 여전히 매우 많다. 이를 해결하기 위해, 산화/환원된 유기물 분자의 ECFPs를 input data로 사용하여 정확도를 높이거나, 더 많은 dataset에 대해 Training을 진행시켜볼 수 있을 것이다. 더욱이, 충분한 hyperparameter optimization에 더불어, 후보군 ANN model을 더 다양화하여 수많은 ANN 구조에 대해 Training을 진행하면 fitting이 더 잘되는 model을 찾을 수 있을 것이다.

# 참고 문헌

[1] John-Anders Stende, 『Constructing high-dimensional neural network potentials for molecular dynamics』, Faculty of Mathematics and Natural Sciences University of Oslo, September 2017, pp.19-43.

[2] David Rogers and Mathew Hahn, 『Extended-Connectivity Fingerprints』, J. Chem. Inf. Model., 50, 742−754, February 4, 2010, pp.742-754.

[3] Susumu Yanagisawa, 『Determination of the ionization energy and the electron affinity of organic molecular crystals from first-principles: dependence on the molecular orientation at the surface』, Department of Physics and Earth Sciences, Faculty of Science, University of the Ryukyus, 1 Senbaru, Nishihara, Okinawa 903-0213, Japan, November 27, 2019, pp.1-9.

[4] 한 신, 김유종, 허지향, 『Vanadium Redox Flow Battery 개발 및 국내 실증』, Journal of the Electric World / Monthly Magazine, Special Issues 4, June, 2014, p. 50.

[5] Martin Abadi et al., 『TensorFlow: Large-scale machine learning on heterogeneous systems』,Software available from tensorflow.org, 2015.

[6] Open-Source cheminformatics : http://www.rdkit.org

[7] Y. Ding et al., Chem. Soc. Rev. 47, 69, 2018.

[8] N. Dardenne et al., J. Phys. Chem. C 119, 23373, 2015.

[9] Sechan Lee, Giyun Kwon, Kyojin Ku, Kyungho Yoon, Sung-Kyun Jung, Hee-Dae Lim, Kisuk Kang, 『Recent progress in organic electrodes for Li and Na rechargeable batteries』, Advanced Materials, Vol 30, 1704682, 2017.

```
from rdkit import Chem
from rdkit.Chem import AllChem
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras import backend as K
from sklearn.model_selection import KFold
import os
from tqdm.notebook import tqdm
gpunum =1
gpus = tf.config.list_physical_devices('GPU')
if gpus:
    try:
        tf.config.set_visible_devices(gpus[gpunum], 'GPU')
        tf.config.experimental.set_virtual_device_configuration(gpus[gpunum],

[tf.config.experimental.VirtualDeviceConfiguration(memory_limit =5000)])
        logical_gpus = tf.config.list_logical_devices('GPU')
        print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
    except RuntimeError as e:
        print(e)
radius =1
n_bits =2048
l2_value =0.003
dropout_rate =0.1
lr =0.003
batch_size =30000
epochs =5000

df = pd.read_excel('/home/thgus4425/data/EA_fromMPdata.xlsx', index_col =0)
smiles = df['smiles']
labels = df['EA']
fplist = []
done_index = []
```

```
for i,s in enumerate(tqdm(smiles)):
    try:
        mol = Chem.MolFromSmiles(s)
        fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius, n_bits)
        fplist.append(fp)
        done_index.append(i)
    except:
        pass
labels = np.array(labels[done_index])
fps = np.array(fplist)
maxval, minval = (9.867671107321804, -3.4570409156131063)
np.max(labels), np.min(labels)

def build_model():
    l2 = keras.regularizers.l2(l=0.003)
    randomnormal = keras.initializers.RandomNormal
    inputs = layers.Input((2048,))
    out = layers.Dense(1024,
                       activation ='relu',
                       kernel_regularizer = l2,
                       kernel_initializer = randomnormal)(inputs)
    out = layers.Dropout(0.2)(out)
    out = layers.Dense(512,
                       activation ='relu',
                       kernel_regularizer = l2,
                       kernel_initializer = randomnormal)(out)
    out = layers.Dropout(0.2)(out)
    out = layers.Dense(256,
                       activation ='relu',
                       kernel_regularizer = l2,
                       kernel_initializer = randomnormal)(out)
    out = layers.Dropout(0.2)(out)
    out = layers.Dense(1,
                       kernel_regularizer = l2,
                       kernel_initializer = randomnormal)(out)
    model = models.Model(inputs,out)
    return model
kf = KFold(n_splits =8, random_state =123, shuffle =True)
kf_results = kf.split(fps)
```

```
def TrueMAE(y_true, y_pred):
    truemae = tf.math.reduce_mean(tf.math.abs((tf.squeeze(y_true)-tf.squeeze(y_pred))*(maxval-minval)))
    return truemae
for i, (train_idx, validation_idx) in enumerate(kf.split(fps)):
    K.clear_session()
    model = build_model()
    if i ==0:
        model.save_weights('./initial_weights.hdf5')
    else:
        model.load_weights('./initial_weights.hdf5')

    model.compile(keras.optimizers.Adam(lr),
                  loss = keras.losses.MeanSquaredError(),
                  metrics = [TrueMAE])

    def train_generator():
        for fp, l in zip(fps[train_idx], labels[train_idx]):
            yield fp, l
    def val_generator():
        for fp, l in zip(fps[validation_idx], labels[validation_idx]):
            yield fp, l
    def minmax_norm(inputs, labels):
        labels = (labels-minval)/(maxval-minval)
        return inputs, labels
    types = (tf.int32, tf.float32)
    shapes = ((2048,),())
    train_data = tf.data.Dataset.from_generator(train_generator,
                                        output_types = types,
                                        output_shapes = shapes)
    train_data = train_data.map(minmax_norm)
    train_data = train_data.batch(batch_size)
    val_data = tf.data.Dataset.from_generator(val_generator,
                                              output_types = types,
                                              output_shapes = shapes)
    val_data = val_data.map(minmax_norm)
    val_data = val_data.batch(batch_size)
```

```
tensorboard = keras.callbacks.TensorBoard('./logs{}'.format(i))
    if not os.path.isdir('./checkpoint{}'.format(i)):
        os.mkdir('./checkpoint{}'.format(i))
    checkpointcallback =
tf.keras.callbacks.ModelCheckpoint('./checkpoint{}'.format(i)+'/epoch{epoch:05d}_val_
TrueMAE{val_TrueMAE:.4f}.hdf5', monitor ='val_TrueMAE', save_best_only =True,
save_weights_only =True, mode ='min')
    def schedule(epoch, _):
                    if epoch<500:
                        return lr
                    elif epoch>=500:
                        return lr * tf.math.exp(3* (500- epoch)/(epochs-500))
    learningratecallback = tf.keras.callbacks.LearningRateScheduler(schedule,
verbose=0)
    model.fit(train_data,
              validation_data = val_data,
             epochs = epochs,
             callbacks = [tensorboard, checkpointcallback, learningratecallback])
```