Protein Discovery With Discrete Walk-Jump Sampling

ICLR 2024, Outstanding Papers

Presenter: 김지환

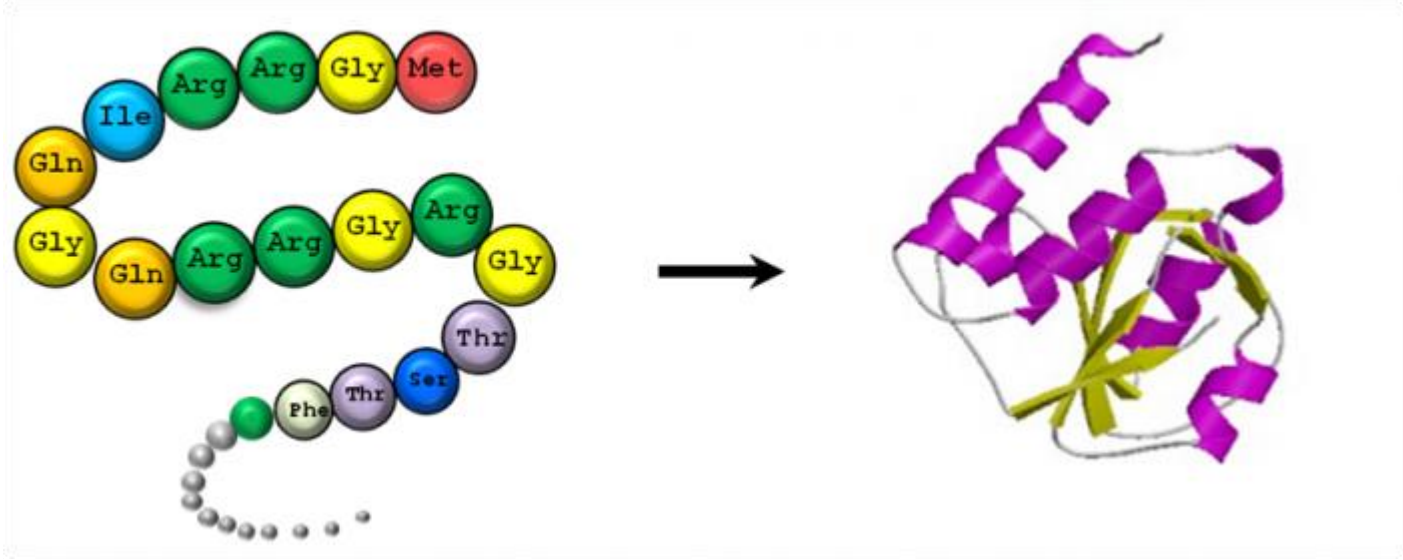
목차

Introduction

- 단백질 설계는 discrete sequence generation 문제

  20개의 아미노산으로 이루어진 이산(discrete) 조합 공간에서 유효한 서열을 탐색

  이 공간의 크기는 20^L (L은 서열 길이)로 지수적으로 큼 → 매우 sparse 하고 복잡한 공간



https://cityu-bioinformatics.netlify.app/too2/new_proteo/pro_seq/

Introduction

- 항체(anti body) 관련 단백질은 구조적 제약이 크고 데이터 수가 적어 모델링이 어려움

  항체는 면역 시스템의 주요 단백질로, 외부 항원과 결합하는 **CDR**(complementarity determining regions)를 가짐
  항체는 conserved framework regions (FRs) 와 highly variable CDRs 를 가지기 때문에
  단백질 설계 문제 중에서도 더욱 복잡하고 제약 조건이 많음

Introduction

- 딥러닝 기반 생성모델

실험 기반 탐색은 시간과 비용이 큼

딥러닝 기반 생성 모델을 활용해, 표본 효율적이고 생물학적으로 유효한 후보를 빠르게 생성

그러나 기존 모델들은 discrete space의 특성 때문에 sampling/optimization에 한계가 있음

Introduction

- 기존 모델

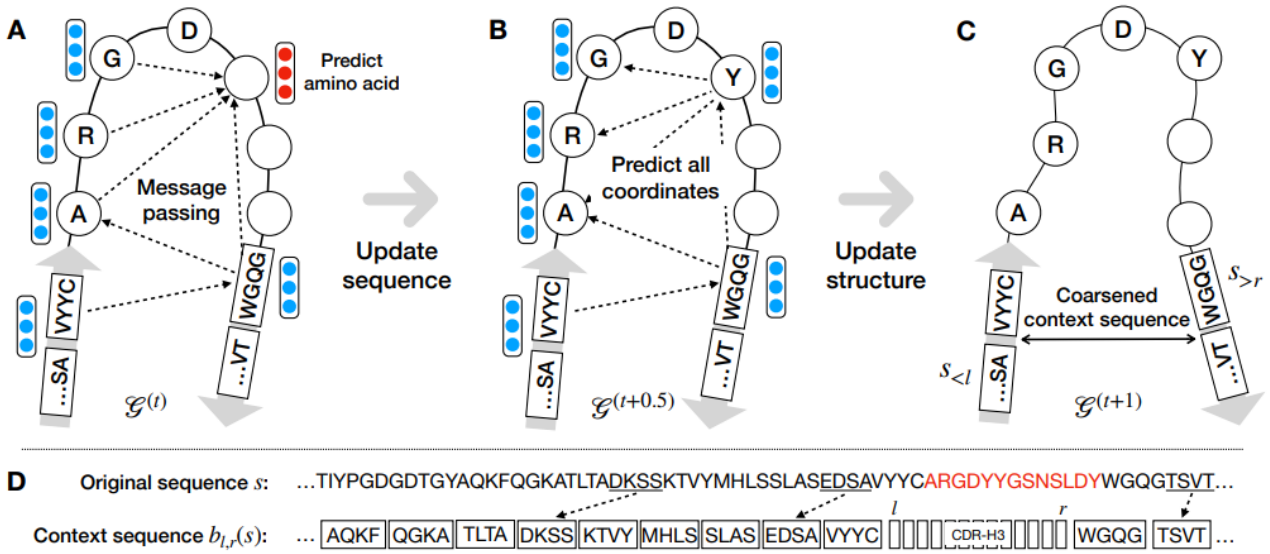


Figure 2: (A-C) One generation step of RefineGNN. Each circle represents a CDR residue and each square represents a residue block in a coarsened context sequence. (D) Sequence coarsening.

RefineGNN: Inefficient, can suffer from accumulation of errors and high inference latency.

https://arxiv.org/pdf/2110.04624

Introduction

- 기존 모델

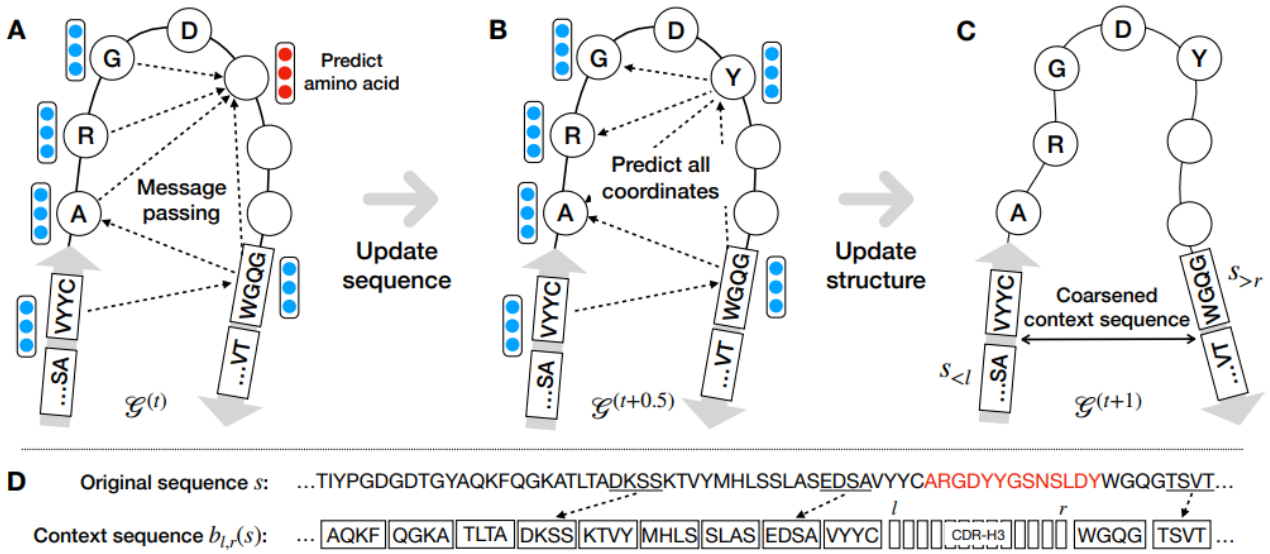


Figure 2: (A-C) One generation step of RefineGNN. Each circle represents a CDR residue and each square represents a residue block in a coarsened context sequence. (D) Sequence coarsening.

Autoregressive: 느린 속도와 높은 계산량

Error accumulation: 초기의 잘못된 오류가 전체 단백질의 구조적 불안정성 초래 가능

https://arxiv.org/pdf/2110.04624

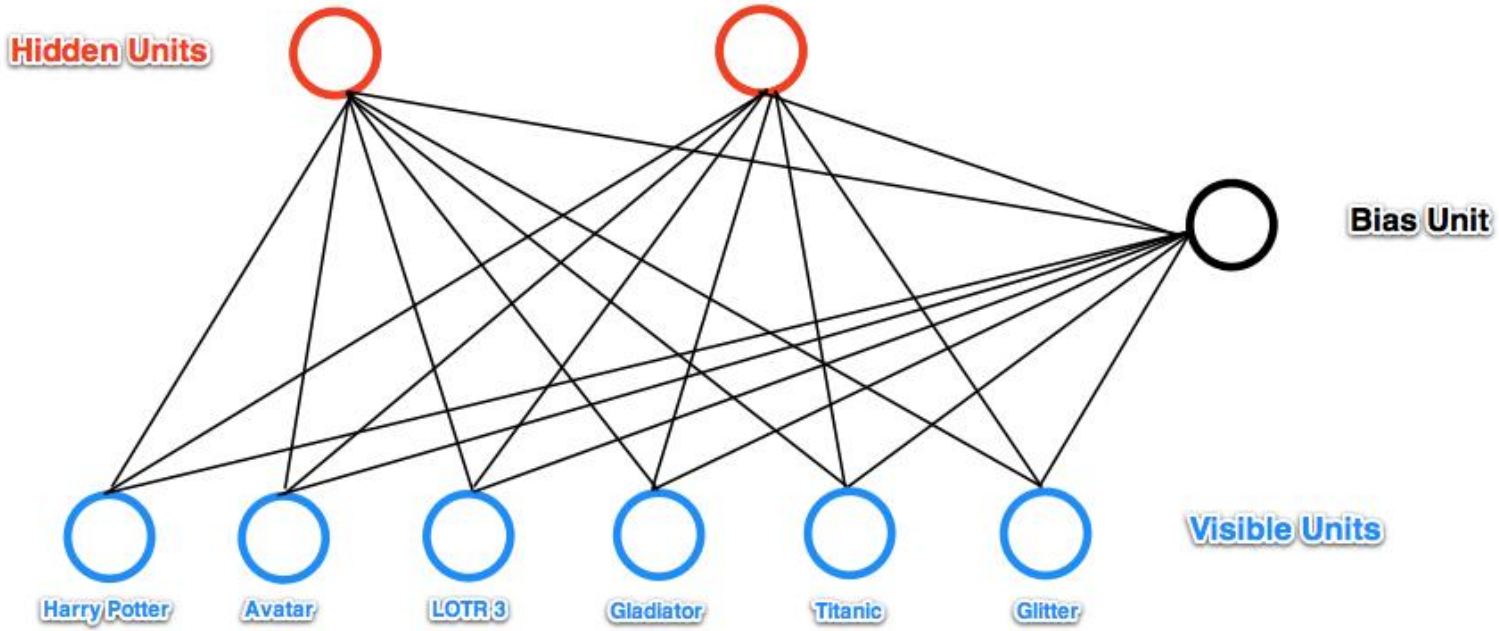
Introduction

- 제안

We introduce Smoothed Discrete Sampling (SDS)
Discrete Walk-Jump Sampling (dWJS), a method building on the Neural Empirical Bayes (NEB) formalism,
that addresses the brittleness of discrete EBMs and diffusion models

Preliminary: Energy based model

Restricted boltzmann machine



https://angeloyeo.github.io/2020/10/02/RBM.html

Preliminary: Energy based model

- $F(\mathbf{v}) = -\log \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$
- $Z = \sum_{\mathbf{v}} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$

Restricted boltzmann machine

- 주어진 데이터 분포에 대해 모델의 로그 우도를 최대화

$$E(v,h) = -v^\top W h - a^\top v - b^\top h$$

에너지 함수

$$P(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$$

특정한 visible vector v에 대한 확률

$$\log P(\mathbf{v}) = \log \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} - \log Z \longrightarrow \log P(\mathbf{v}) = -F(\mathbf{v}) - \log Z$$

로그 우도 → Free-energy term 사용

https://angeloyeo.github.io/2020/10/02/RBM.html

Preliminary: Energy based model

Restricted boltzmann machine

- Z 는 intractable, 계산이 어려움

$$\max_{\theta} \log P(\mathbf{v}) \quad \Longleftrightarrow \quad \min_{\theta} \mathcal{L}(\theta) = -\log P(\mathbf{v}) = F(\mathbf{v}) + \log Z$$

https://angeloyeo.github.io/2020/10/02/RBM.html

Preliminary: Energy based model

Restricted boltzmann machine

- 로그 우도를 미분해서 gradient 를 계산
- Constastive Divergence 를 줄이는 방식으로 학습 (진짜 데이터는 에너지를 낮추고, 가짜 샘플은 에너지를 높임)

$$\nabla_\theta \log P(\mathbf{v}) = -\nabla_\theta F(\mathbf{v}) + \mathbb{E}_{P_{\text{model}}}[\nabla_\theta F(\mathbf{v})]$$

$$loss = F(v) - F(v^{(1)})$$

Gibbs Sampling 1번 (CD-1)

https://angeloyeo.github.io/2020/10/02/RBM.html

Preliminary: Neural Empirical Bayes

- 관측 데이터 X를 가우시안 커널로 스무딩 하여 Y를 얻음

$$Y = X + \mathcal{N}(0, \sigma^2 I)$$

- Empirical Bayes는 Y에서 X를 추정하는 베이즈 추정 방법

$$\hat{x}(y) = y + \sigma^2 \boxed{\nabla \log f_Y(y)} \leftarrow \text{Score}$$

- Score-function parameterized with a neural network

$$\hat{x}_\phi(y) = y + \sigma^2 g_\phi(y)$$

$$\boxed{L(\theta) = \mathbb{E}_{X,Y}\left[\|X - Y + \sigma^2 \nabla \phi(Y;\theta)\|^2\right]}$$

Loss function

Preliminary: Neural Empirical Bayes

- 새로운 샘플 생성방법

  Walk-Jump Sampling
- 고차원 분포를 따라 샘플을 생성 -> 원래 데이터 manifold 근처로 이동

$$y_{t+1} = y_t - \delta^2 \nabla \phi(y_t) + \sqrt{2}\delta\epsilon_t$$

Walk: Langevin MCMC

$$\hat{x}(y_t) = y_t - \sigma^2 \nabla \phi(y_t)$$

Jump: Empirical Bayes

Discrete Walk-Jump Sampling (dWJS)

본 논문에서의 방법론

- 노이즈가 추가된 데이터에 대해 EBM 학습
- NEB를 이용해 Score-based Denoising 모델 학습
- Walk는 학습된 EBM의 gradient, Jump는 Denoising 모델사용

Discrete Walk-Jump Sampling (dWJS)

EBM f(y) 학습

$$\arg\max_{\theta} \mathbb{E}_{y\sim p_Y}[\log p_\theta(y)] = \arg\max_{\theta} \left(\mathbb{E}_{y^-\sim p_\theta(y)}[f_\theta(y^-)] - \mathbb{E}_{y^+\sim p_Y}[f_\theta(y^+)]\right)$$

where y^+ are noisy training data and y^- are noisy data sampled from the model.

Discrete Walk-Jump Sampling (dWJS)

Score network Denoiser g(y) 학습

$$\hat{x}_\phi(y) = y + \sigma^2 g_\phi(y)$$

$$\mathcal{L}(\phi) = \mathbb{E}_{x\sim p(x), y\sim p(y|x)} \|x - \hat{x}_\phi(y)\|^2$$

Discrete Walk-Jump Sampling (dWJS)

Sampling

Algorithm 1: Discrete Walk-Jump Sampling

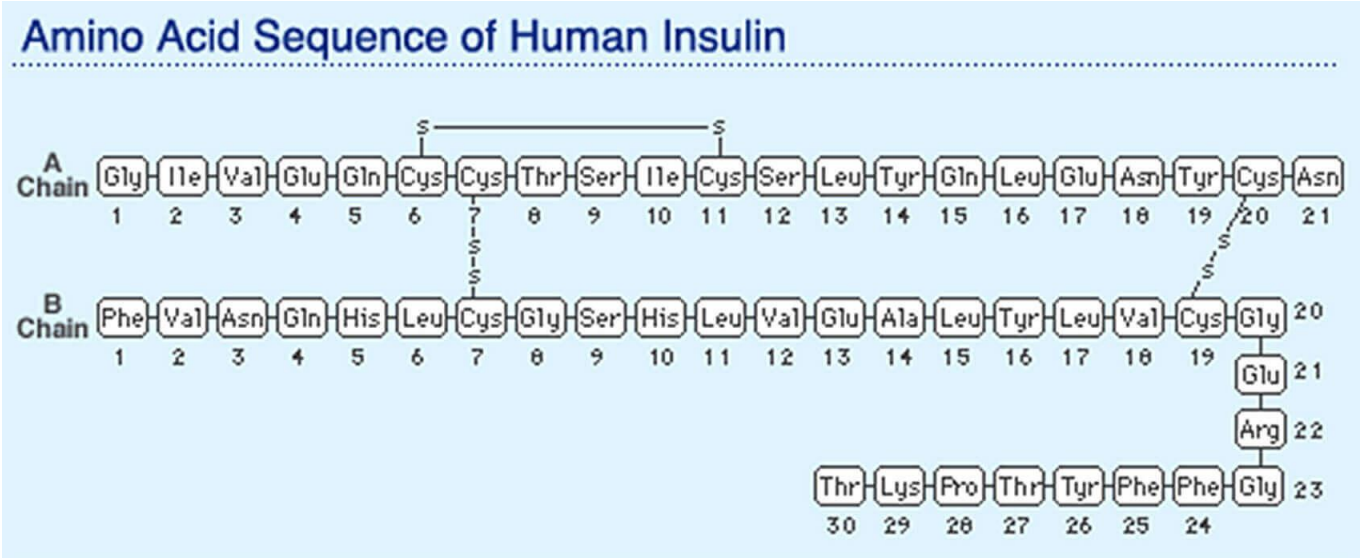
Input: Denoiser, $g_\phi(y)$, energy-based model, $f_\theta(y)$
Output: Noisy samples $y \sim p(y)$, denoised samples $\hat{x}(y)$

$y_0 \sim \text{Unif}([0,1]^d) + \mathcal{N}(0, \sigma^2 I_d)$
for $t = 0, \ldots, T-1$ **do**
  $y_{t+1} \leftarrow y_t - \delta \nabla_y f_\theta(y_t) + \sqrt{2\delta}\varepsilon_t \,, \varepsilon_t \sim \mathcal{N}(0, I_d)$
end
$\hat{x}_T \leftarrow y_T + \sigma^2 g_\phi(y_T)$
return $\arg\max \hat{x}_T$ `// to recover one-hot encoding`

Discrete Walk-Jump Sampling (dWJS)

Variable length protein sequence generation

- 단백질은 아미노산(amino acid, AA)들이 특정 순서로 연결된 이산 시퀀스
- 단백질마다 길이가 다르고, 중간에 삽입하거나 삭제되는 경우도 있음
- 아미노산의 종류는 20종



https://www.ajinomoto.com/ko/amino-acids/how-the-sequencing-of-amino-acids-in-insulin-revealed-the-structure-of-proteins

Discrete Walk-Jump Sampling (dWJS)

Variable length protein sequence generation

- AHo numbering scheme을따라 항체 데이터베이스 (OAS)를 정렬함
- Heavy chain은 길이 149로 안정화
- Light chain은 길이 148로 안정화
- Multiple Sequence Alignment 을 통해 모두 같은 길이로 맞춰주면 고정된 입력 벡터로 만들수 있음 (gap token 사용)

데이터의 차원: one hot encoded (149+148)x21 = 6237

Discrete Walk-Jump Sampling (dWJS)

Protein Design vs Discovery

- Protein Desgin: 출발점 (sequence)를 주고 나머지만 변경하는 방식
- Langevin MCMC 매 스텝마다 고정 된 자리(P로 정함)를 starting sequence s로 덮어쓰기

$$P^\top \operatorname{argmax} \hat{x}(y,t) = P^\top s$$

Discrete Walk-Jump Sampling (dWJS)

Optimal Noise Level for Discrete sequence data

- SDS 방식에서 학습과 샘플링을 위한 적절한 노이즈 수준을 선택하는 것이 중요
- 작으면 다양성 저하, 너무 크면 데이터 구조가 무너짐

Discrete Walk-Jump Sampling (dWJS)

Optimal Noise Level for Discrete sequence data

- 데이터 샘플 쌍 사이의 평균 거리를 계산해서 Critical noise level 측정
- 0.5 이상이 이산 서열 데이터의 SDS 학습에 적합한 노이즈 레벨

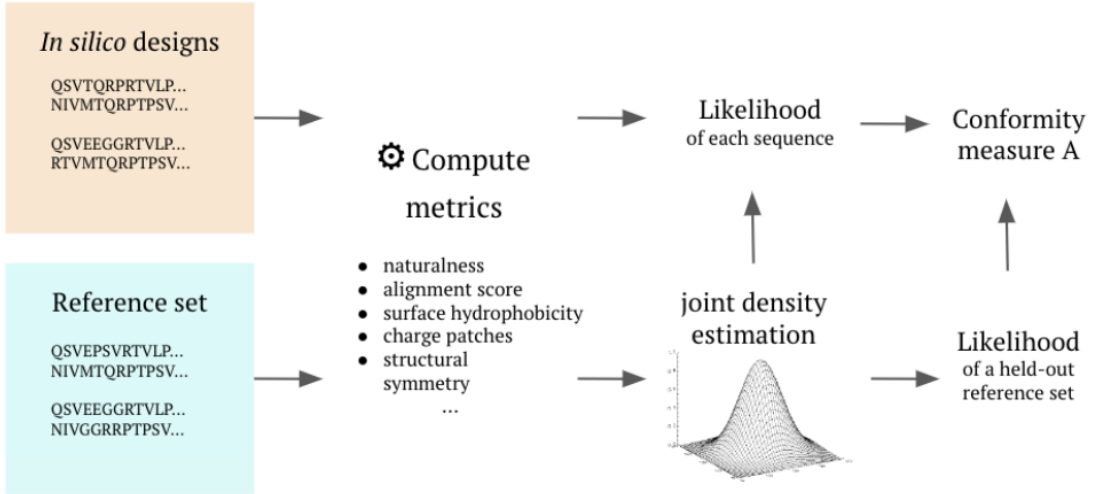
$$\chi_{ii'} = \frac{||X_i - X_{i'}||}{2\sqrt{d}}$$

$$\sigma_c = \max_{ii'} \chi_{ii'}$$

Discrete Walk-Jump Sampling (dWJS)

Distributional Conformity Score (DCS)

- 기존의 이미지/자연어 생성분야에서 사용하는 평가 지표들이 단백질 생성 평가에 적합하지 않음 (FID, BLEU)
- 생성된 시퀀스가 좋은 분포를 따르는가에 대한 평가: 실제 항체 데이터셋에서 측정된 다양한 property
- Conformity measure A: 다양한 통계적/생물물리적 특성의 결합된 확률 밀도 기반 측도
- DCS > 0.3은 거의 100% 발현율



Results

Model	$W_{\text{property}} \downarrow$	Unique ↑	$E_{\text{dist}} \uparrow$	IntDiv ↑	DCS ↑
dWJS (energy-based)	**0.056**	**1.0**	58.4	55.3	0.38
dWJS (score-based)	0.065	0.97	**62.7**	**65.1**	0.49
SeqVDM	0.062	**1.0**	60.0	57.4	0.40
DEEN	0.087	0.99	50.9	42.7	0.41
GPT 3.5	0.14	0.66	55.4	46.1	0.23
IgLM	0.08	**1.0**	48.6	34.6	**0.533**
ESM2	0.15	**1.0**	70.99*	77.56*	0.061

- $W_{property}$: 생성된 샘플의 물리/통계적 특성이 validation set랑 비슷한지 측정: biophysical plausibility
- Unique: 생성된 샘플 중 중복 없는 서열 비율
- Edist: validation set과 얼마나 다른지. 클수록 새롭고 참신: novelty
- IntDiv: 샘플들 사이의 다양성
- DCS: 생성된 샘플이 validation set 분포에서 얼마나 자연스럽고 가능성 있는지

Results

Model	$W_{\text{property}} \downarrow$	Unique ↑	$E_{\text{dist}} \uparrow$	IntDiv ↑	DCS ↑
dWJS (energy-based)	**0.056**	**1.0**	58.4	55.3	0.38
dWJS (score-based)	0.065	0.97	**62.7**	**65.1**	0.49
SeqVDM	0.062	**1.0**	60.0	57.4	0.40
DEEN	0.087	0.99	50.9	42.7	0.41
GPT 3.5	0.14	0.66	55.4	46.1	0.23
IgLM	0.08	**1.0**	48.6	34.6	**0.533**
ESM2	0.15	**1.0**	70.99*	77.56*	0.061

- dWJS가 기존 ref 와 가장 유사한 물리/통계적 특성을 가지며 기존과 다른 참신한 디자인 생성.
- IgLM은 아주 높은 자연스러움(DCS)을 가졌지만 다양성이 부족함
- ESM2는 다양하고 참신한 단백질 생성하지만 발현 가능성이 현저히 낮음

Results

발현율

- 기존 EBM 학습방법은 42% 발현율로 저조

Table 3: Measured protein synthesis.

Model	$\text{total}_{\text{expressed}} \uparrow$
dWJS (score-based)	**1.0**
dWJS (energy-based)	0.97
EBM	0.42

Results

결합 확률

- p_{bind}: 생성된 항체 서열이 HER2 항원에 결합할 확률, 분류기를 통해 추정
- $total_{bind}$: 실험적으로 확인된 결합률, 실제로 실험에서 HER2에 결합한 비율

Table 4: Predicted and measured antibody binding affinity.

Model	p_{bind} ↑	$\text{total}_{\text{bind}}$ ↑
dWJS (energy-based) (Ours)	**0.96**	**0.70**
dWJS (score-based) (Ours)	0.95	N/A
LaMBO-2 (Gruver et al., 2023)	N/A	0.25
AbDiffuser (Martinkus et al., 2023)	0.94	0.22 (0.57)
SeqVDM	0.75	N/A
GPT 4	0.74	N/A
Transformer	0.60	N/A
EGNN	0.58	N/A

Disscussion

1. Score-based, Energy-based dWJS의 차이점은 뭐고 왜 energy function을 사용해야 했는가?
2. 그래서 이 논문의 novelty는 뭘까?

Disscussion

dWJS: score-based vs energy-based

- 둘 다 같은 walk jump sampling 프레임워크 안에서 작동
- score-based는 score-network의 output으로 walk, jump 둘 다 수행
- energy-based 모델은 energy function의 gradient로 walk, score-network로 jump

Disscussion

dWJS: score-based vs energy based

- EBM 모델 (dEBM) 과 Denoising 모델 (ByteNet)의 구조적 차이로 인한 sampling 속도 차이
- ByteNet: 35 Layer with hidden dimension of 128 (dilated convolution)
- dEBM: 3 Conv1D layers, 128 size linear

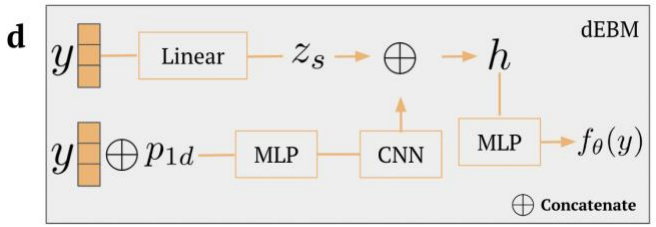


Table 7: Model size, sampling time and memory consumption.

Model	Parameters (M)	GPU time / sample (ms) ↓	GPU memory (MB) ↓
dWJS (energy-based)	9.87	64.89	4734.76
dWJS (score-based)	1.03	327.60	**16.7**
SeqVDM	12.31	940.40	246.39
DEEN	1.03	976.99	16.72
IgLM	12.89	2800	—
ESM2	7.8	**5.25**	—

Disscussion

dWJS: score-based vs energy based

- Efficient Sampling (5배 이상) > fast mixing (샘플들이 빠르게 다양한 모드를 탐색)
- walk, jump 를 완전히 분리. 각각의 기능을 잘 하도록 학습

Disscussion

Novelty?

1. Discrete EBM 학습의 어려움을 SDS 프레임워크로 극복

이산 공간에서는 gradient가 정의되지 않으므로 Langevin MCMC를 사용할수 없음
이로 인해 EBM은 물론 DDPM, NCSN도 학습이 어려움
논문에서는 노이즈를 추가해 Continous space로 smooth시켜 EBM을 잘 학습시킴

Disscussion

Novelty?

2. 다중 noise level을 사용하는 모델 대비 단일 noise level로 효율적인 샘플링

기존 score-based generative models (DDPM, Noise Conditional Score Network)은 noise schedule을 포함하거나
다중 noise level을 사용해야 하며, 각 noise level마다 Langevin step을 반복 수행하므로 inference가 느림
본 논문은 Neural Empirical Bayes (NEB)를 활용해 단일 noise level만 사용하여도 학습 및 고품질 샘플 생성이 가능함을 보임

Thank You
For Your Attention

Preliminary: Energy based model

Restricted boltzmann machine

```
def forward(self,v):
    # RBM에서 forward propagation은 visible -> h
    # hidden unit과 visible unit을 샘플링 하는 과
    pre_h1,h1 = self.v_to_h(v)

    h_ = h1
    for _ in range(self.k): # Contrastive Diverg
        pre_v_,v_ = self.h_to_v(h_)
        pre_h_,h_ = self.v_to_h(v_)

    # 아래의 v는 입력으로 들어온 v이고 v_는 sampl
    return v,v_
```

```
def free_energy(self,v):
    # Free Energy 계산
    vbias_term = v.mv(self.v_bias) # mv: matrix - vector product
    wx_b = F.linear(v,self.W,self.h_bias)

    temp = torch.log(
        torch.exp(wx_b) + 1
        )
    hidden_term = torch.sum(temp, dim = 1)

    return (-hidden_term - vbias_term).mean()
```

```
v,v1 = rbm(sample_data)
loss = rbm.free_energy(v) - rbm.free_energy(v1)
loss_.append(loss.data.item())
```

https://angeloyeo.github.io/2020/10/02/RBM.html