

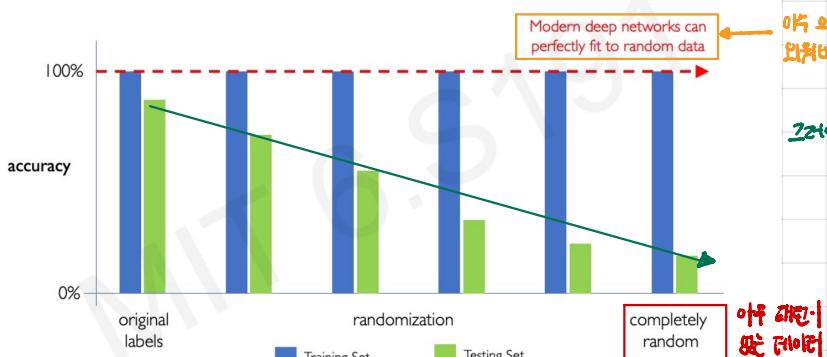
Universal Approximation Theorem

→ 어떤 함수(임의적인 간단한 형태로도 예측 가능)가 주어지면 해당 함수에 대한 일련의 층계이 주어지면, 해당 함수를 극복할 수 있는 **Single layer neuron network**가 존재합니다.

→ 해당 Single layer neuron 개수 또한 사용하는 parameter를 통해 정확히 알 수는 있지만, 그 이후의 수법을 수 있는 Single layer neuron 개수를 반드시 정확히 알 수는 없습니다.

Limitations of Deep Learning algorithm

1. Rethinking Generalization : Deep learning 와 일반화를 잘 하는 것인가?

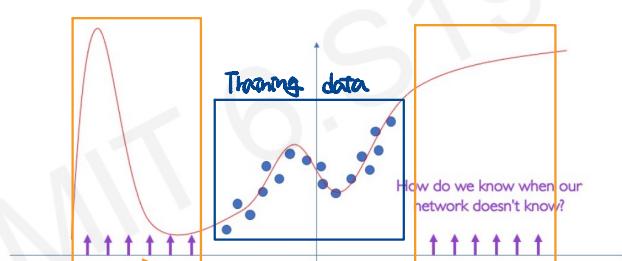


이것은 Deep Learning Training data를 어떤 것에 대처하여 학습하는 것처럼 보입니다.

그러나 실제로 일반화 능력이 훨씬 낮습니다.

아무 대입·
없는 데이터

Neural networks are excellent function approximators
...when they have training data



모델이 확률적 예측을 못합니다. → Uncertainty는 어떻게 예측할 수 있는가?

2. Neural network failure modes

→ Uncertainty 예측에 불행하는 경우가 대부분입니다.

→ Adversarial Perturbation : 사용으로 공격하기 위한 작은 임의 변화에 비해, 모델의 예측값은 차이를 불러옵니다.

3. Algorithmic Bias

→ 데이터 자체, 학습 방식에서 기인한 Bias

Neural Network Limitations...

- Very **data hungry** (eg. often millions of examples)
- **Computationally intensive** to train and deploy (tractably requires GPUs)
- Easily fooled by **adversarial examples**
- Can be subject to **algorithmic bias**
- Poor at **representing uncertainty** (how do you know what the model knows?)
- Uninterpretable **black boxes**, difficult to trust
- Often require **expert knowledge** to design, fine tune architectures
- Difficult to **encode structure** and prior knowledge during learning
- **Extrapolation**: struggle to go beyond the data

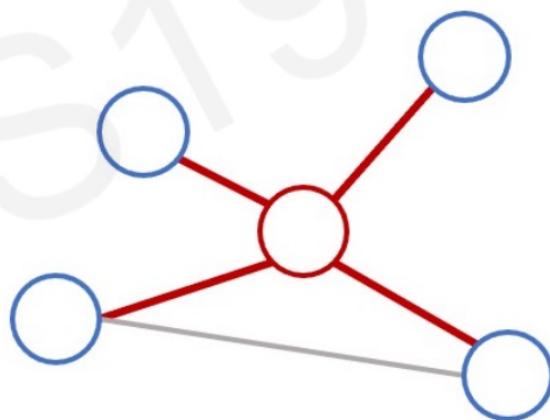
lecture 5

In this lecture

1. Encoding Structure into Deep Learning

⇒ 시각의 물리적인 Graph로 이해할 때, Graph Encoding하기 어렵다.

Graph Convolutional Networks (GCNs)



- CNN에서처럼 kernel 사용

- 연결여부는 adjacency matrix로 표현된다.

- 각 Node의 feature vector가 존재하고, 모든 노드의 feature matrix가 존재한다.

- kernel은 노드를 통하여 각각의 feature를 전파로 전달한다.

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

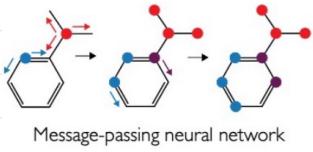
- $H^{(l)}$: l 번째 layer의 feature matrix

- \tilde{A} : Adjacency matrix

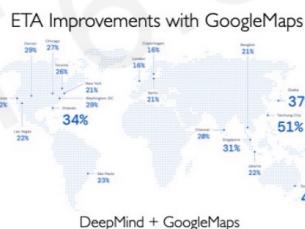
- \tilde{D} : degree of node

Applications of Graph Neural Networks

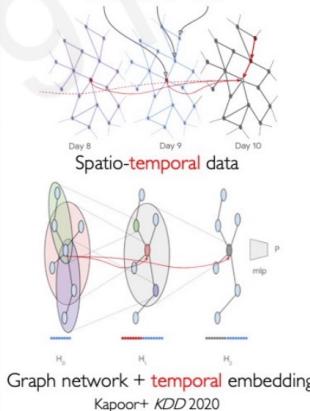
Molecular Discovery



Traffic Prediction



COVID-19 Forecasting



2D object의 차트
3D에서 물체의 위치를 Graph의 노드처럼 생각하는 방식
학습할 수 있도록 한다.

2. Generative Model & Diffusion Models

- Problem of VAEs and GANs

a. 학습이 어려움

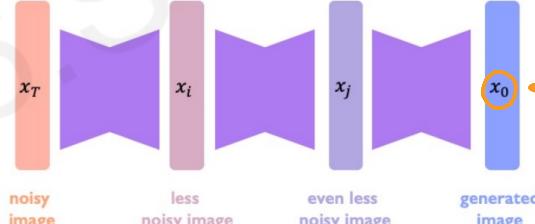
b. 학습된 결과가 항상 X

c. Generative가 잘 되지 않는다.

Diffusion Model

Diffusion: Generating images iteratively by repeatedly refining and removing noise

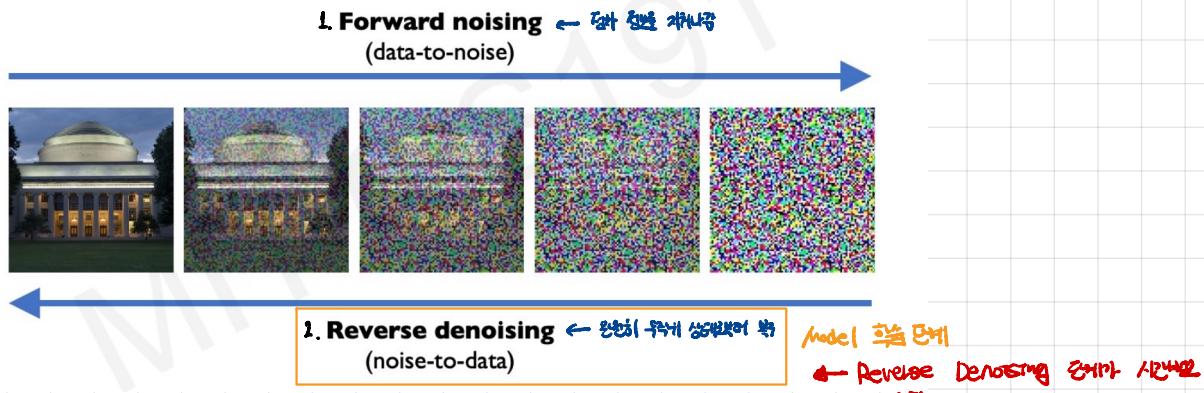
Noise \downarrow \downarrow \downarrow



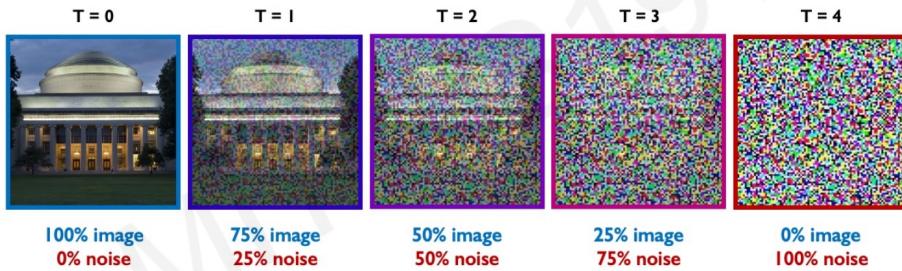
- VAE와 같이 noise vector로 하기 때문에 생략하기 쉽다
(one-shot)

Noise \downarrow \downarrow \downarrow \downarrow

The Diffusion Process



1. Forward Noising



- 시간이 지남수록 점차 Noise를 추가한다.

$$x_t = \sqrt{\alpha_t} \cdot x_0 + \sqrt{1 - \alpha_t} \cdot \epsilon, \epsilon \sim N(0, I), \bar{x}_t = \frac{1}{T} \sum_{t=1}^T x_t: \text{Noise 증가}$$

한번의 흔적에 α_t 를 더해 Noise

Noise distribution

2. Reverse Denoising: Forward Noising 단계에서 한 차운(t) = (Image + Noise)가 주어지면 ($t-1$) 차운의 이미지를 어떻게 찾는가?



- 어떻게 보여줄 것인가?

Loss?:
→ 가장 흔하게 pixel별로 비교하여 평균이나 다른 손실함수!

$$\hat{x}_{t-1} = \frac{1}{\sqrt{1 - \beta_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \cdot \epsilon_t(x_t, t) \right) + \underbrace{\text{noise}}_{\sim N(0, I)}, \beta_t: t 단계의 noise \geq 1 (= 1 - \alpha_t)$$

증거로, Random Sampling 사용

$$\text{Loss} = E_{x, \epsilon, t} [\epsilon_t(x_t, t) - \hat{\epsilon}]^2 \leftarrow \text{Pixel 별로 Forwarding시 Noise와 다른 Noise를 비교}$$

- 각 단계별로 $P(x_t | x_{t-1})$ 라는 확률분포를 따른다.

⇒ 첫 번째 흔적(100% noise)에서는 각 예상된 prediction이 이전과는 다른 예상하는 상태로 바뀐다.

Maximum Uncertainty
 $t=T$

→ Maximum diversity
 $t=0$

* 예상된 Random은 상관관계 사용
상관관계로 수 있는 것의 최대 정점이다.

Applications: Text to Image

→ 각 단계별로 다양한 정보를 통해
다양성이 있다.