가우시안 오류 선형 단위(GELUS)

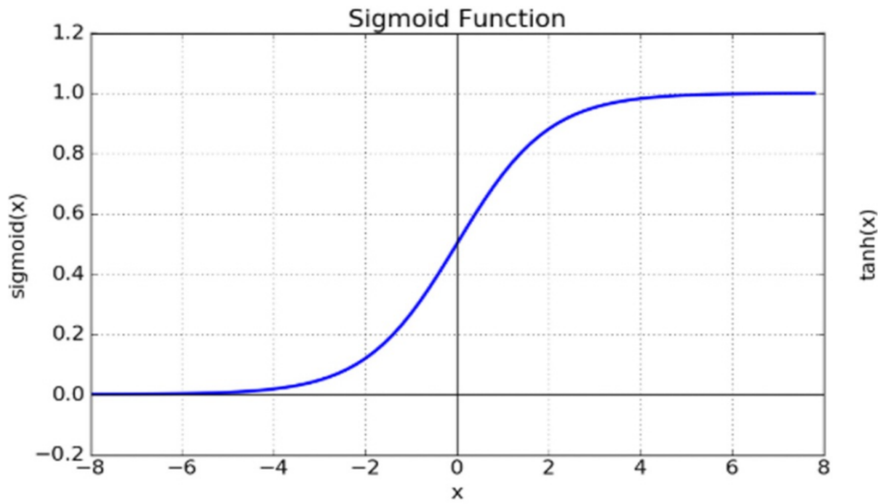
GAUSSIAN ERROR LINEAR UNITS (GELUS)

3조 20 유하정 21 신혜빈 21 우진주

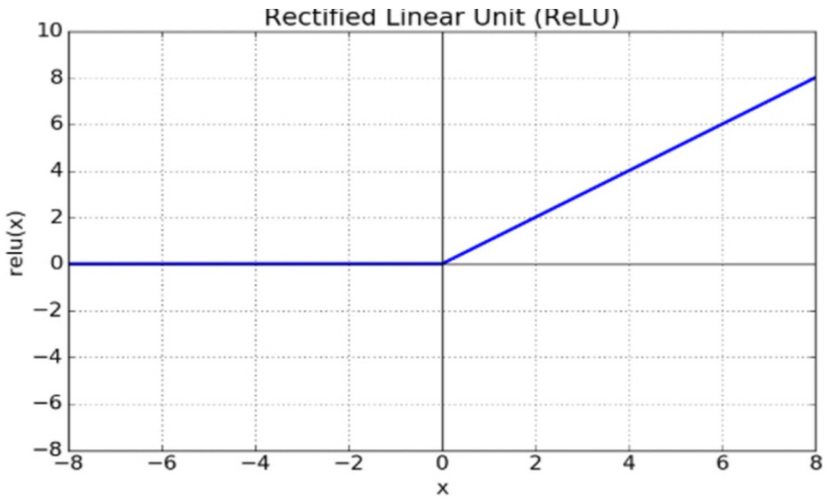
Index

01 Introduction

Sigmoid **vs** ReLU



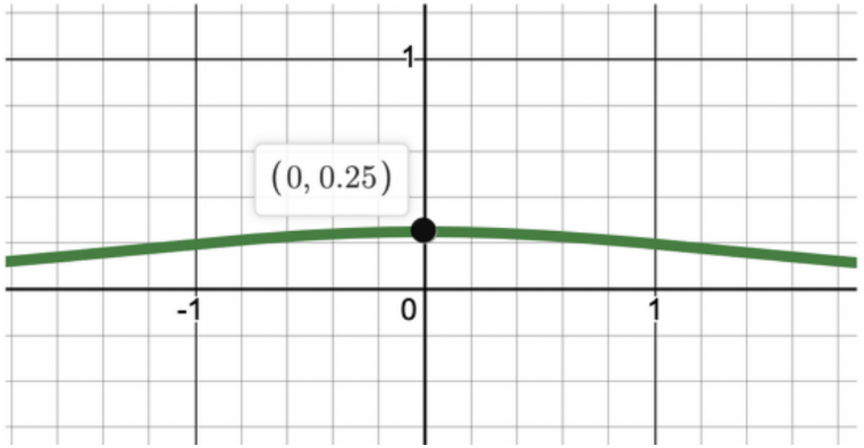
< Sigmoid 함수 >



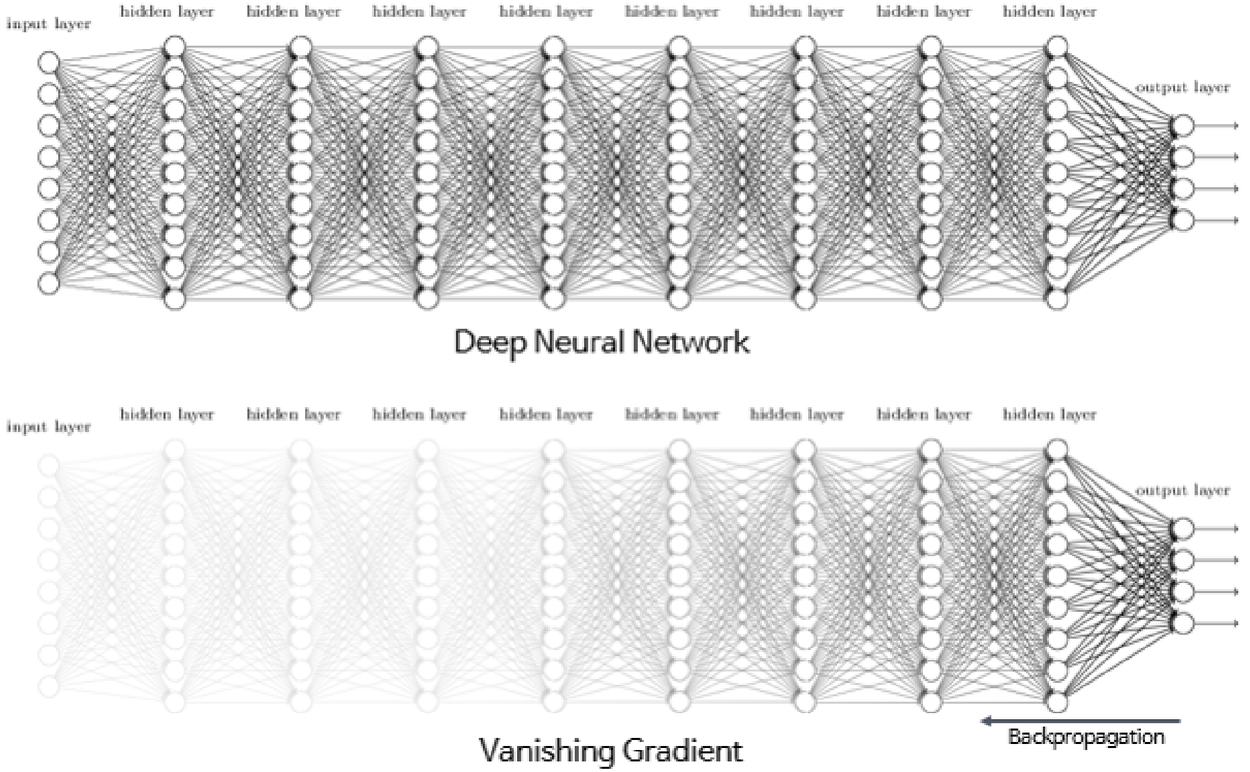
< ReLU 함수 >

01 Introduction

Sigmoid**의 Vanishing Gradient**

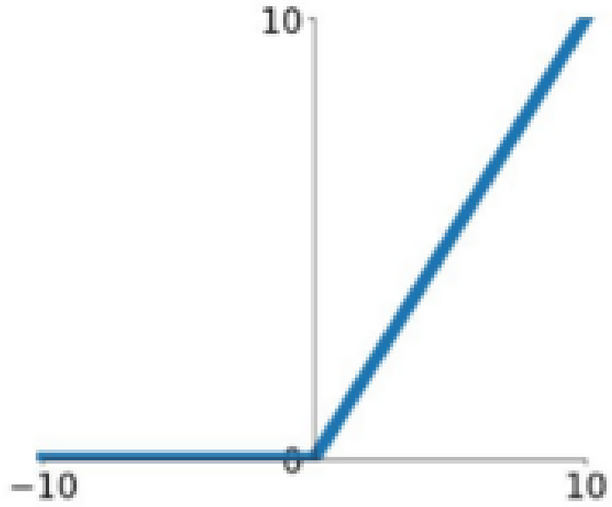


< Sigmoid 기울기 >

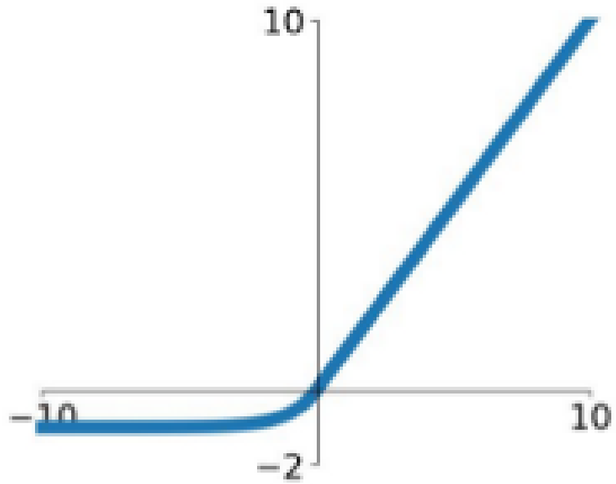


01 Introduction

ReLU **vs** ELU



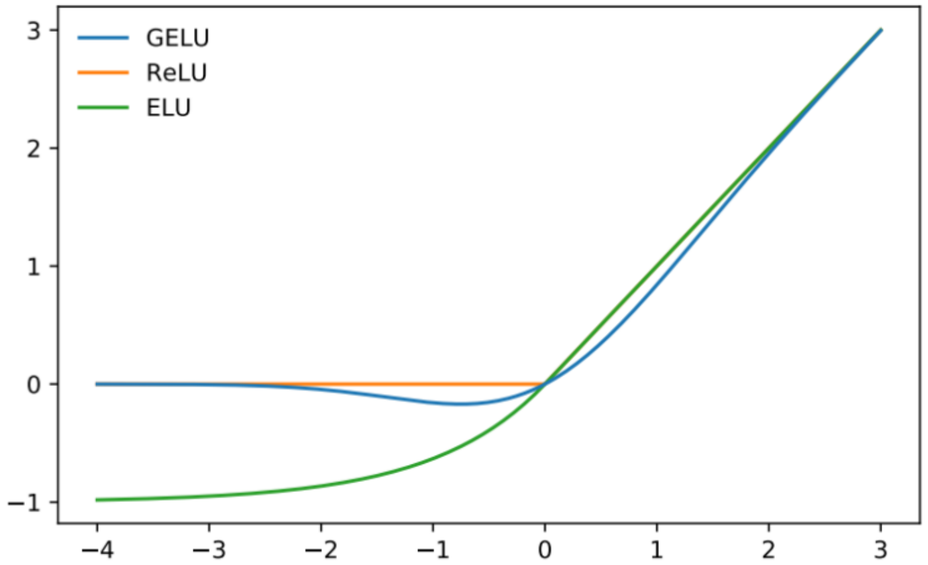
< ReLU 함수 >



< ELU 함수 >

01 Introduction

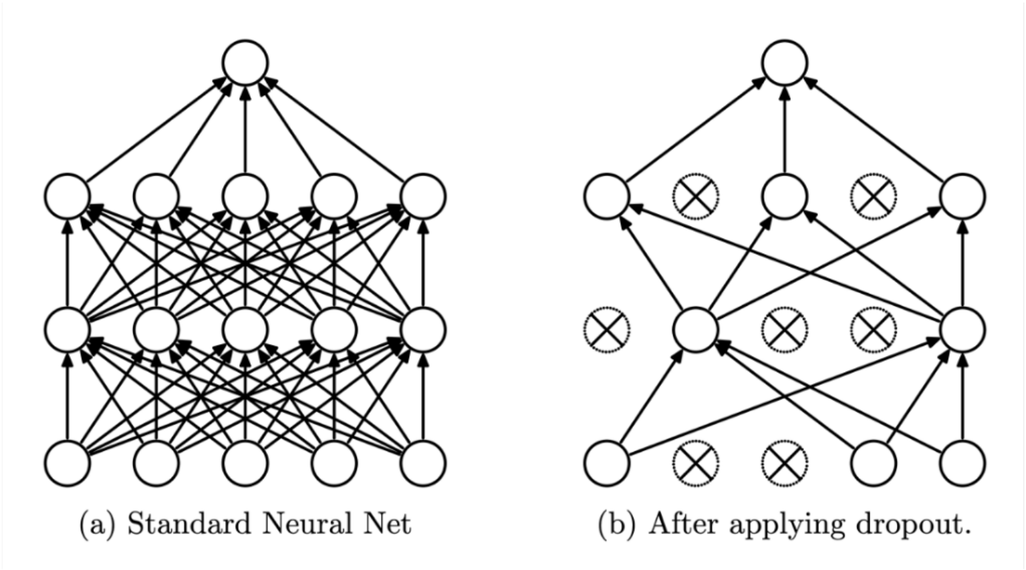
GELU의 등장



< ReLU vs ELU vs GELU >

01 Introduction

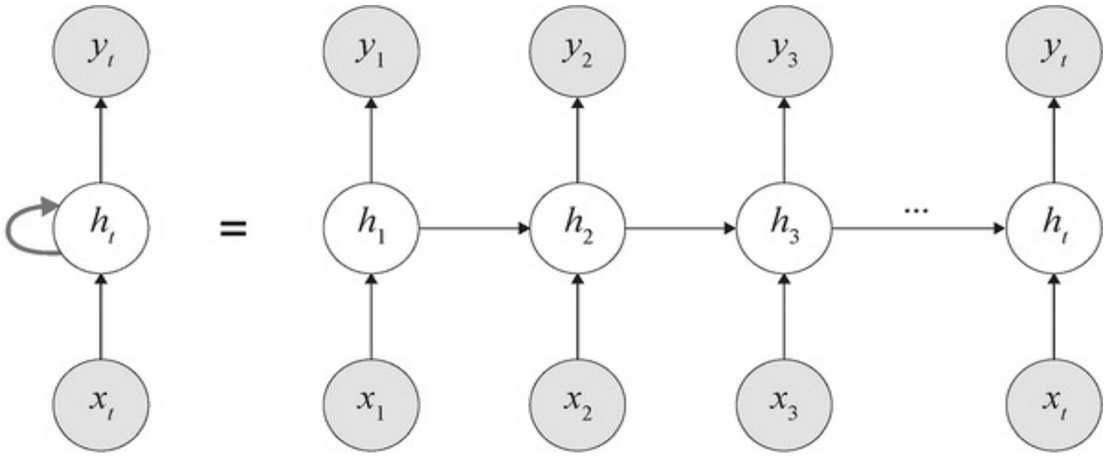
확률적 정규화(**노이즈, 드롭아웃**)



신경망의 각 층에 대해서 Dropout rate에 따라 랜덤으로 뉴런 제거 → 유사 앙상블처럼 네트워크 작동

02 GELU Formulation

ReLU + Dropout + Zoneout



Zoneout

순환신경망(RNN)에서 사용되는 드롭아웃의 변형

랜덤으로 뉴런 비활성화

ht = mask * f(ht-1, xt) + (1 - mask) * ht-1

02 GELU Formulation

ReLU **+** Dropout **+** Zoneout

ReLU + **Dropout** + **Zoneout**

GELU

표준정규분포의 누적분포함수(phi(X))에 베르누이 함수를 씌운 형태 → **Bernoulli** (phi(X))

02 GELU Formulation

뉴런 입력값의 **특성**

배치정규화를 통해 정규분포를 따르는 경향성 有

X가 감소함에 따라 삭제될 확률 ↑

⇓

무작위성, 입력값에 대한 의존성 유지

배치정규화(Batch Normalization)

미니 배치의 평균과 분산을 기준으로 입력값을 정규화하는 기법

02 GELU Formulation

GELU **수식**

GELU = Bernoulli(Phi(x))

= Phi(x)*I(x) + (1-Phi(x))*0x

= xPhi(x)

vs

Zoneout

ht = mask*f(ht-1,xt) + (1 - mask)*ht-1

02 GELU Formulation

GELU **수식**

$$\mathbf{GELU(x)} = xP\,(X \leq x) = x\Phi(x)$$

$$= x \cdot 1/2[\,1 + \mathrm{erf}(x/\sqrt{2})]$$

근사

$$\Rightarrow \quad 0.5x(1 + \tanh[\sqrt{2}/\pi(x + 0.044715x^3)])$$

03 GELU Experiment - MNIST Classification

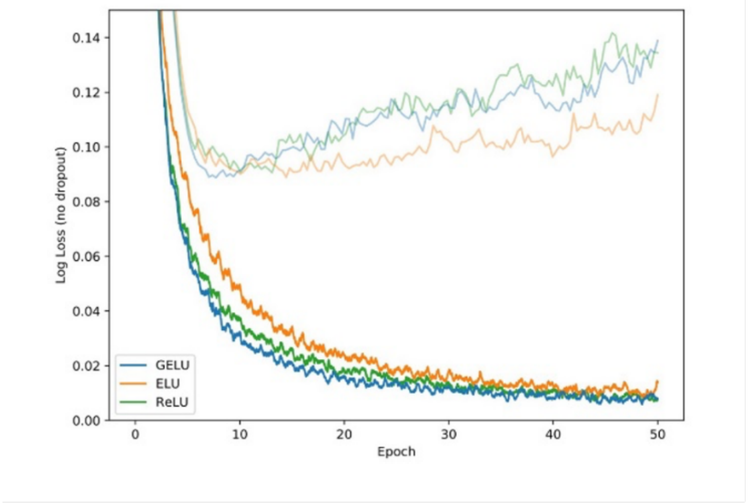
MNIST Classification

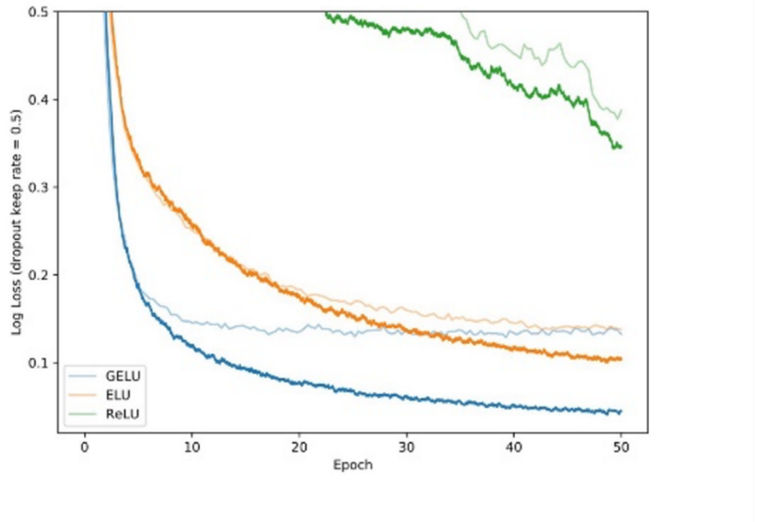
- Fully connected neural network GELU, ReLU, ELU
- 각 레이어마다 128개의 뉴런, batch size 128, 총 50회 학습

1. MNIST 분류 실험
2. MNIST 분류 정확도 실험

03 GELU Experiment - MNIST Classification

MNIST 분류 실험





- 진한 색 : training set, 연한 색 : validation set

  왼 : 드롭아웃을 적용하지 않은 모델은 학습데이터에서만 낮은 손실(과적합되었음)

  오 : 드롭아웃 적용한 모델은 검증데이터에 대해서도 낮은 손실을 보여줌(과적합 완화)

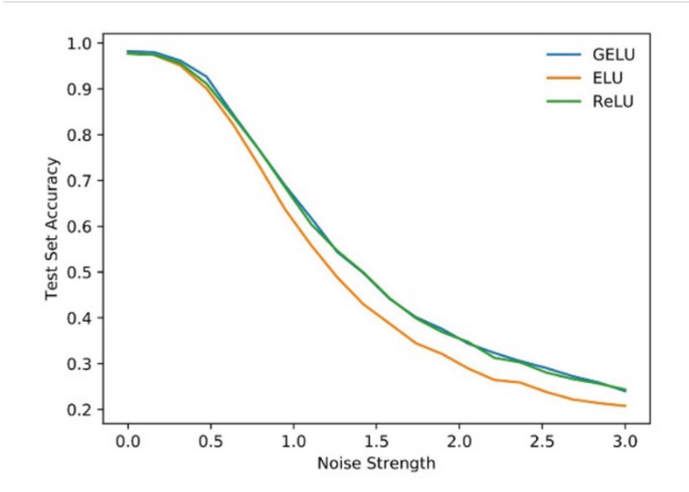
03 GELU Experiment - MNIST Classification

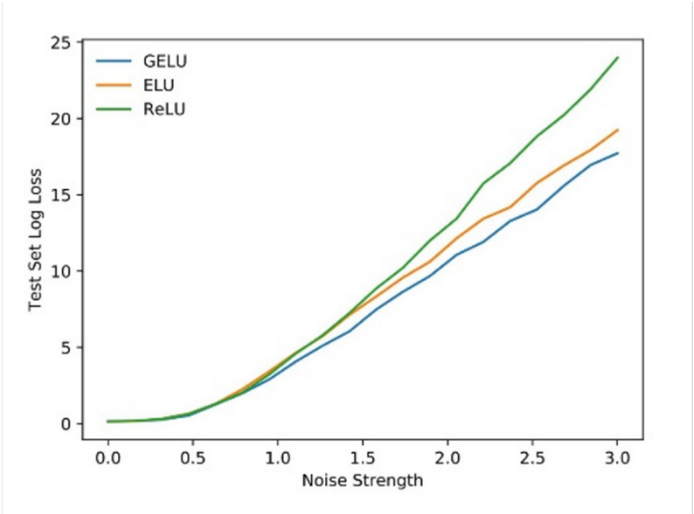
MNIST 분류 정확도 실험

- **Robustness** : 입력데이터에 noise가 추가되었을 때의 성능
- a(최대 noise)를 3으로 설정
- training set에서 5000개의 validation 데이터로 learning rates {10^(-3), 10^(-4), 10^(-5)} 5번씩 실행
- learning rate마다 5개 결과의 **중앙값** 중 가장 좋은 결과를 가지는 learning rate 선택

03 GELU Experiment - MNIST Classification

MNIST 분류 정확도 실험





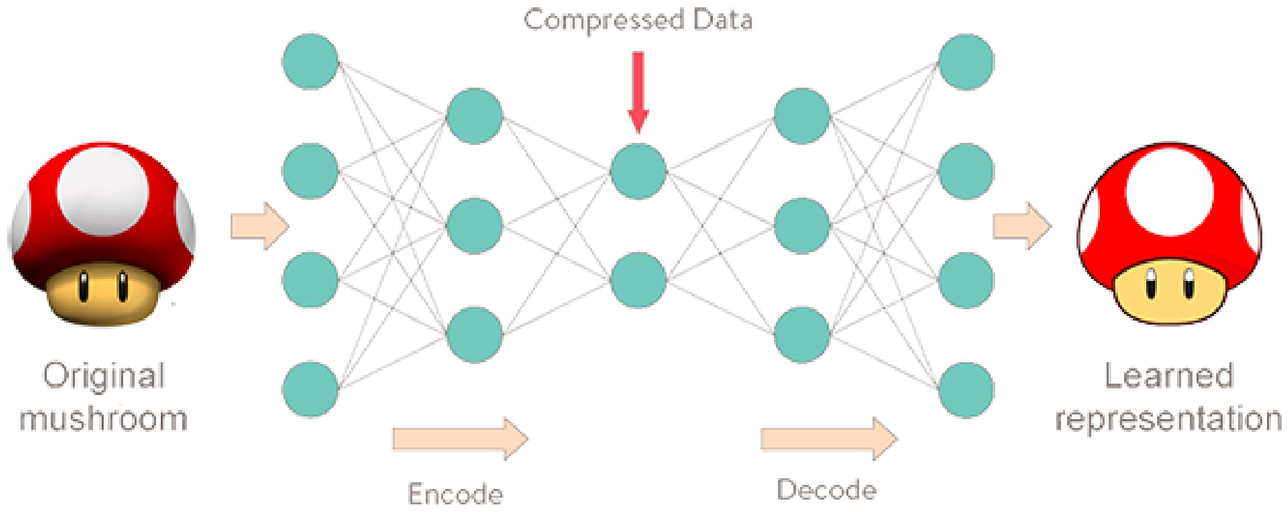
왼 : test set의 정확도면에서 GELU가 가장 높음

오 : test set의 log 손실이 GELU가 가장 낮음

-> GELU가 noise에 강함

03 GELU Experiment - MNIST Autoencoder

MNIST Autoencoder



Autoencoder : 비지도학습 인공지능 신경망

03 GELU Experiment - MNIST Autoencoder

MNIST Autoencoder

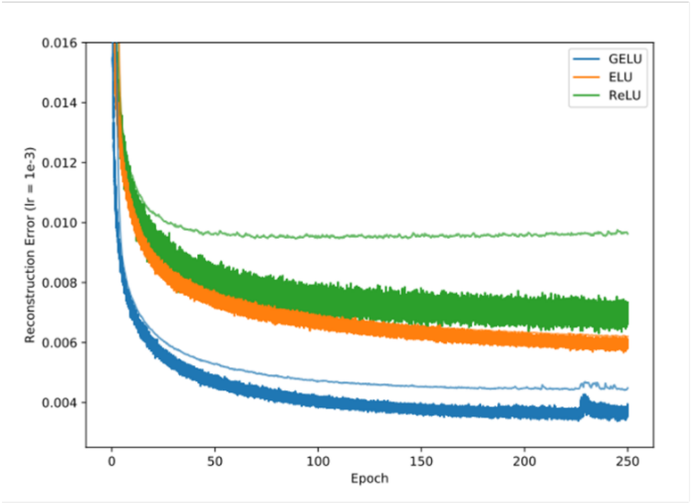
- 레이어 너비 : 1000, 500, 250, 30, 250, 500,1000
- Adam optimizer 사용
- batch size 64
- learning rate {10^(-3), 10^(-4)}(10^(-2)는 ELU에서 분산, GELU, ReLU 수렴하지 않음)

03 GELU Experiment - MNIST Autoencoder

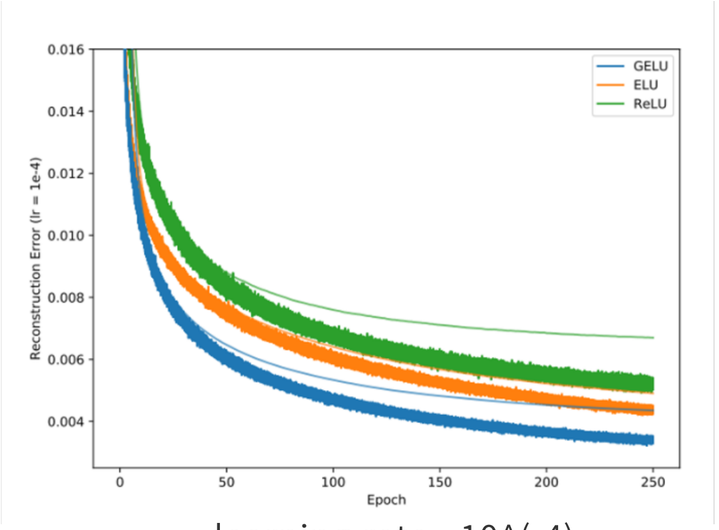
MNIST Autoencoder

log loss : 모델의 예측값과 실제값의 차이 -> 주로 분류 문제에서 사용

reconstruction error : 원본 입력과 재구성된 입력값의 차이 -> autoencoder에서 사용



learning rate : 10^(-3)



learning rate : 10^(-4)

- 얇은 선 : test set의 log loss, 두꺼운 선 : training set의 log loss
- 각 곡선은 3번 실행의 중앙값 의미

  GELU의 loss가 가장 낮음 -> 다양한 learning rate 수용

03 GELU Experiment - Twitter Pos Tagging

Twitter Pos Tagging

- POS : part - of - speech (문장에서 단어들의 품사를 주석으로 붙인 것)
- 트윗태거 : 레이어 2개, 트윗 텍스트를 품사태깅하는 작업 수행. 5,600만개의 트윗 말뭉치에 대해 사전 훈련된 단어 벡터 기반

a) 입력 : 미리 학습된 워드 벡터, 품사 태깅하려는 단어와 주변 왼쪽과 오른쪽 단어들의 벡터를 이어 붙인 형태

b) 출력 : 입력 부분 벡터들을 하나로 이어붙여 문장을 구성하는 벡터 형태

03 GELU Experiment - Twitter Pos Tagging

Twitter Pos Tagging

- 실험 목적 : 25개 태그 포함 POS 주석이 달린 데이터셋을 사용하여 활성화 함수 비교

 -> 적은 예제에서도 잘 일반화되는 비선형성 찾고자 함

- 2개의 레이어, 각 레이어에 256개의 뉴런, Dropout rate 0.8

- Adam optimizer, 세가지 모델 5번씩 learning rate {10^(-3), 10^(-4), 10^(-5)} 변화학습

- 각 학습률에 대해 test set error 측정, 중앙값으로 활성화 함수 비교

03 GELU Experiment - Twitter Pos Tagging

Twitter Pos Tagging

GELU 활성화 함수가

가장 낮은 test set error 중앙값인 12.57%,

ReLU 12.67%, ELU 12.91%

⇓

GELU가 가장 우수

03 GELU Experiment - TIMIT Classification

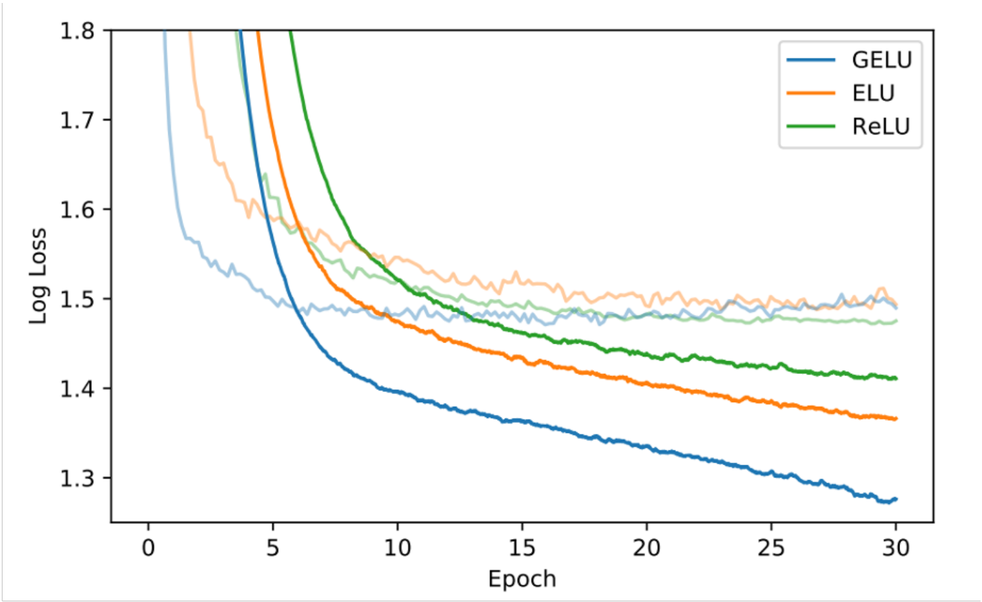
TIMIT Classification

목적 : 각 frame마다 MFCC, energy, derivative를 고려한 26개의 특징으로 중앙 frame의 phone 예측

- 5개의 레이어, 레이어마다 2048개의 뉴런
- Dropout rate 0.5, Adam optimizer 사용
- validation set으로 learning rate {10^(-3), 10^(-4), 10^(-5)} 중 가장 작은 error로 조정
- 5번 실행 후 test error 중앙 값 GELU 29.3%, ReLU 29.5%, ELU 29.6%
- input : 11개의 프레임 시계열데이터, output : 39개의 phone labels

03 GELU Experiment - TIMIT Classification

TIMIT Classification



- 진한 선 : training set의 log loss, 연한 선 : validation set의 log loss
  training set, validation set 모두 log loss 감소, 수렴
  -> 모델이 training set에만 의존하지 않고 새로운 데이터에도 잘 일반화된 것을 의미

03 GELU Experiment - CIFAR-10/100 Classification

Shallower CNN

Table 1: Neural network architecture for CIFAR-10.

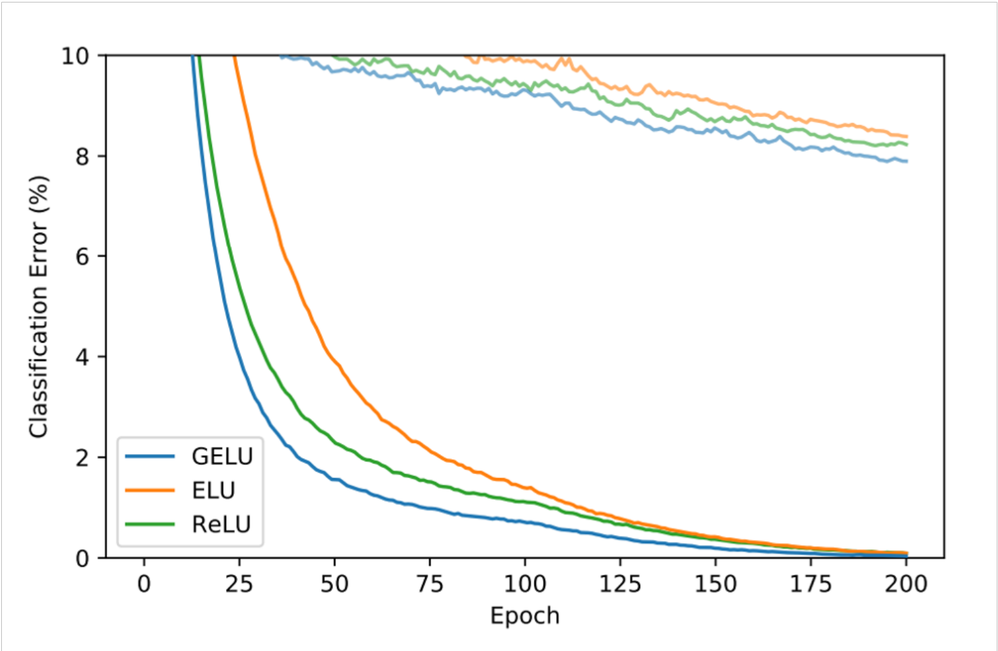
Layer Type	# channels	x, y dimension
raw RGB input	3	32
ZCA whitening	3	32
Gaussian noise $\sigma = 0.15$	3	32
3×3 conv with activation	96	32
3×3 conv with activation	96	32
3×3 conv with activation	96	32
2×2 max pool, stride 2	96	16
dropout with $p = 0.5$	96	16
3×3 conv with activation	192	16
3×3 conv with activation	192	16
3×3 conv with activation	192	16
2×2 max pool, stride 2	192	8
dropout with $p = 0.5$	192	8
3×3 conv with activation	192	6
1×1 conv with activation	192	6
1×1 conv with activation	192	6
global average pool	192	1
softmax output	10	1

- 레이어 9개, Batch normalization
- 데이터 증강 사용 X
- Adam optimizer, 200 epochs
- validation set {10^(-3), 10^(-4), 10^(-5)}

  성능 튜닝, 전체 training set으로 평가

03 GELU Experiment - CIFAR-10/100 Classification

Shallower CNN



각 곡선 : GELU, ELU, ReLU 각각 3번씩 실행 후 중앙값
진한 색 : training set error rates, 연한 색 : test set error rates

100번째 epoch에서 학습률 선형적으로 0 수렴 -> 모델 안정성 높임
GELU error rate 중앙값 : 7.89%, ReLU : 8.16%, ELU : 8.41%

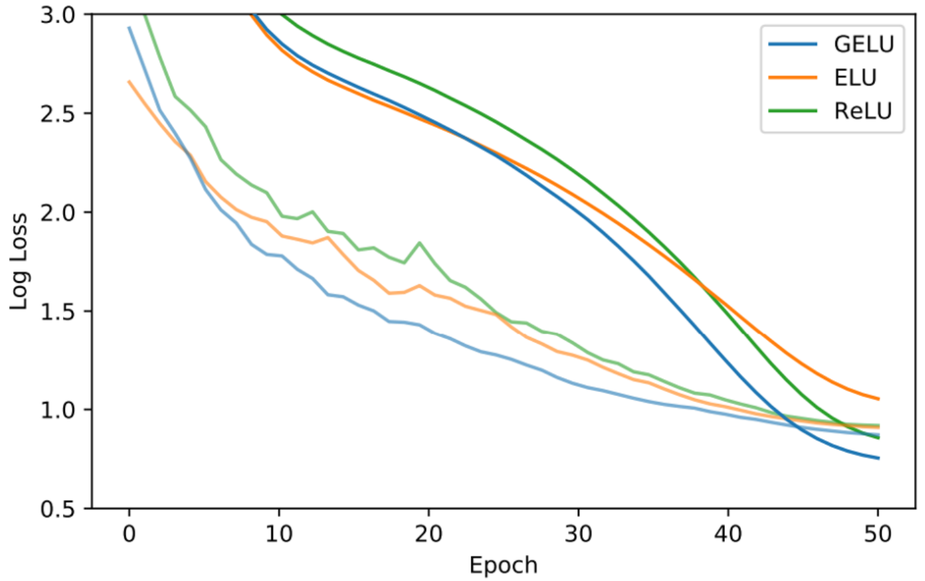
03 GELU Experiment - CIFAR-10/100 Classification

CIFAR - 100 Classification

- 40개의 레이어, 레이어마다 채널수 4배(뉴런 개수 4배) -> 네트워크 너비 증가
- 초기 학습률 10^-1, 50 epochs
- validation set으로 learning rate {10^(-3), 10^(-4), 10^(-5)} 성능 튜닝, learning rate으로 평가
- Nesterov Momentum, Dropout rate 0.7
- ELU 활성화 함수 : 잔차 블록 끝에 배치 정규화 사용

03 GELU Experiment - CIFAR-10/100 Classification

CIFAR - 100 Classification



진한 색 : training set(dropout O), 연한 색 : test set(dropout X)
3번 실행결과 중앙값

수렴하는 모습 보임
GELU error rate 중앙값 : 20.74%, ReLU : 22.89%, ELU :22.98%

04 Discussion

GELU와 ReLU

< 특수한 경우 >

σ : "tanh" 함수의 기울기 → 0,

μ : GELU 함수 내에서 등장하는 상수 = 0

GELU(x) = 0.5 * x * (1 + tanh(√(2/π) * (x + 0.044715 * x^3)))

GELU(x) ≈ 0.5 * x * (1 + tanh(0)) = 0.5 * x * (1 + 0) = 0.5 * x

-> GeLU 함수의 식이 단순화 되면서 비선형성을 잃고 ReLU와 동일한 형태가 됨.

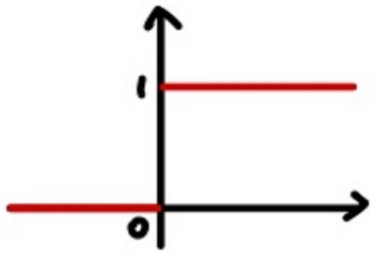
04 Discussion

GELU와 ReLU

ReLU(x) = max(x, 0) = x1(x > 0)

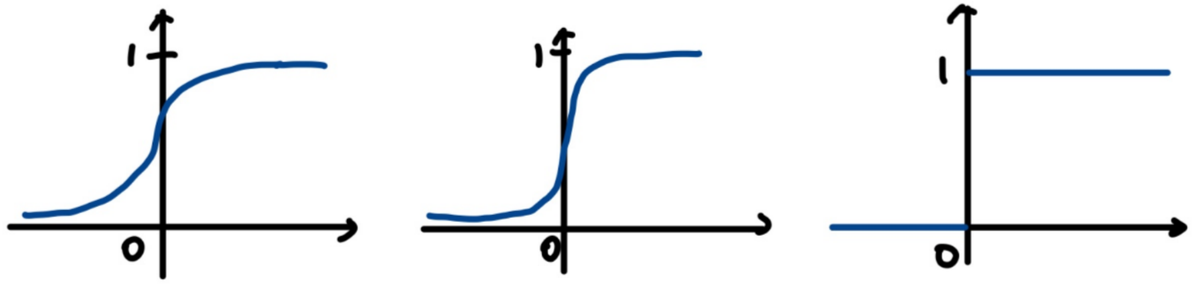
$$ReLU(x) = x \times \mathbb{1}$$
$$= \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$

$\mathbb{1}$: x > 0이면 1, x <= 0이면 0을 출력하는 ReLU의 indicator 함수



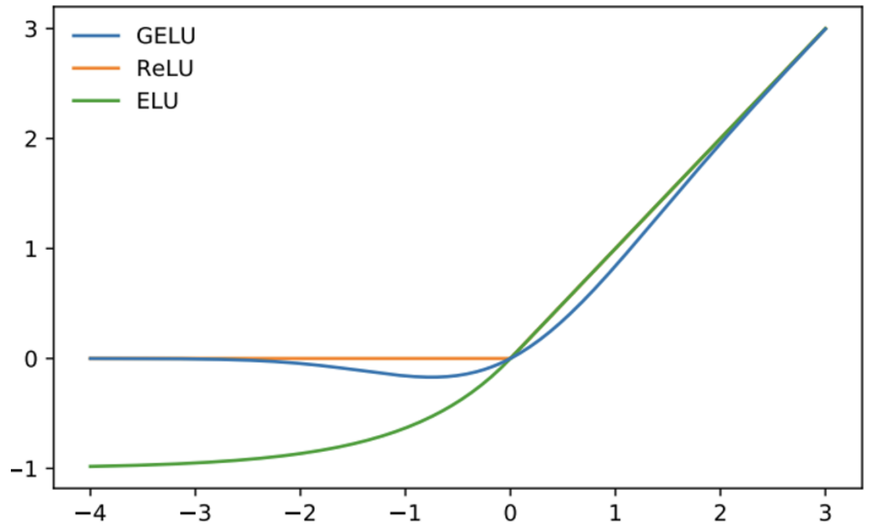
GELU(x) = xΦ(x)

Φ(x) : 표준 정규분포의 누적분포함수(CDF)



04 Discussion

활성화 함수 비교



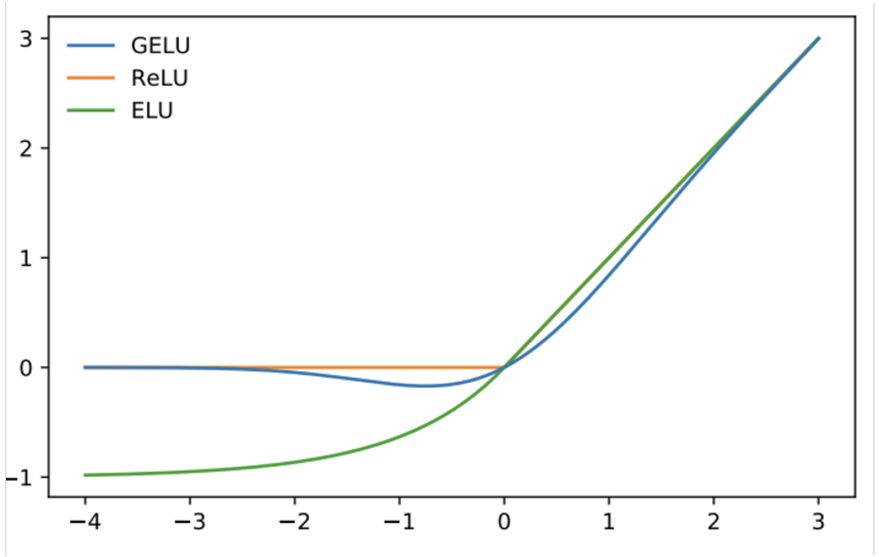
곡률, 비단조성 증가

-> 복잡한 함수 쉽게 근사 가능

활성화 함수	GELU	ReLU	ELU
함수의 값	양수, 0,음수	양수, 0	양수, 0,음수
함수의 형태	블록 X, 비단조	블록 O, 단조	블록 O, 단조
선형	양수 : 비선형, 모든 부분에서 곡률 나타냄	양수 : 선형, 곡률 부족할 수 있음	양수 : 선형, 곡률 부족할 수 있음

04 Discussion

활성화 함수 비교



활성화 함수	GELU	ReLU
입력값에 따른 반응	다른 입력값들과 비교하여 가중치 부여	부호에 따라 gate

GELU : 입력값의 상대적인 크기에 따라 변화하는 비선형성

-> 입력 값들 간의 상대적인 차이에 더 민감하게 반응.

->입력 값들의 상대적인 중요성을 반영할 수 있음

04 Discussion

활성화 함수 비교

$$GELU(x) = x * \Phi(x)$$

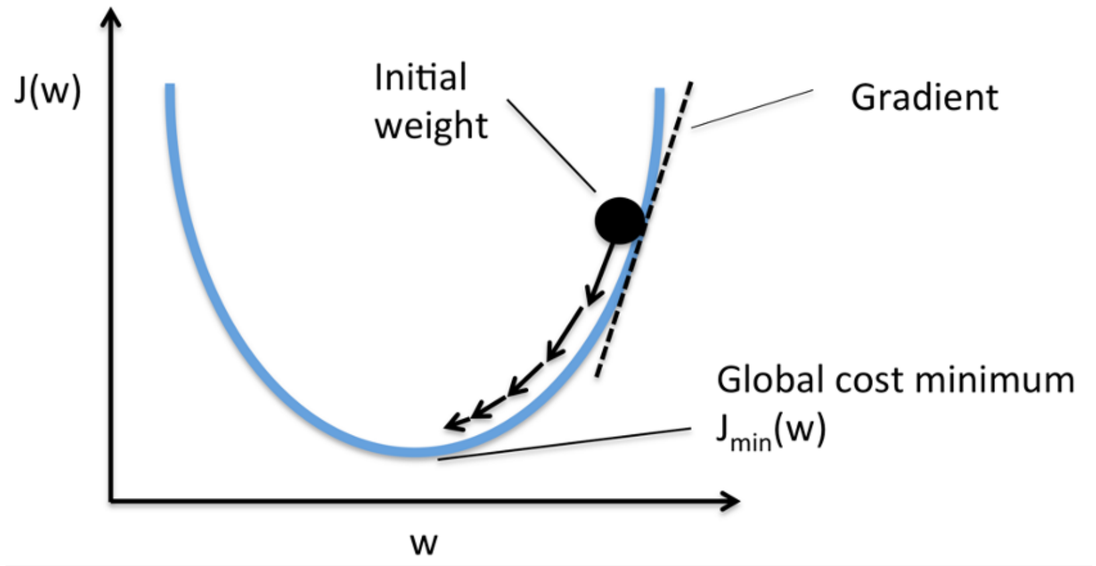
입력 값 x에 대한 확률 변수 Φ(x)의 기댓값을 계산

확률적 규제의 기댓값이라는 점에서 확률적 해석 가능

-> 과적합 방지, 일반화 개선

04 Discussion

GELU를 사용할 때 Tip 1



Momentum 갖는 Obtimizer 사용 권장
(논문에서의 실험은 항상 Adam optimizer 사용)

GELU : 활성화 함수로 딥 러닝 모델에서 많은 비선형성을 가지기 때문에 학습에서의 어려움

-> 모멘텀 최적화를 사용하여 기울기 업데이트 조절하면 학습이 더 안정적.

04 Discussion

GELU를 사용할 때 Tip 2

가우시안(Gaussian) 분포의 누적 분포 함수에 대한 근사값을 사용

GELU(x) = x * Φ(x)

Φ(x) : 가우시안 분포의 누적 분포 함수

-> 평균 0, 표준편차 1 확률 분포로서 무한대의 구간에 걸쳐 정의,

이를 정확히 계산하는 것은 매우 어려움.

-> GELU 함수 구현 시 가우시안 분포의 누적 분포 함수를 근사하는 방법 사용.

=> GELU 함수를 효율적계산 -> 모델의 학습과 예측 성능 향상

04 Discussion

GELU를 사용할 때 Tip 2

1)	2)
0.5x(1 + tanh[√ 2/π(x + 0.044715x^3)])	$x\sigma(1.702x),$

두 방법 모두 GELU보다 성능 부족, ReLU와 ELU보다 더 좋은 결과를 보임

(둘 다 충분히 빠르고 구현하기 쉬운 근사 함수이기 때문)

-> 합리적인 비선형성 선택임.

05 Conclusion

이 논문에서 다양한 데이터셋으로 평가한 결과 :

GELU 정확도 > ELU, ReLU의 정확도

-> GELU 함수는 기존의 비선형 활성화 함수에 대한 대안으로 가능성이 높아짐.

감사합니다

3조 20 유하정 21 신혜빈 21 우진주