RMAF: ReLU-Memristor-like Activation Function for Deep Learning

Paper

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/43b900af-6e0e-4356-8079-b929d58f3bb1/RMAF_Relu-Memristor-Like_Activation_Function_for_D.pdf

https://ieeexplore.ieee.org/document/9066950

Abstract

- 가장 널리 사용되는 활성화 함수는 ReLU함수이다.
- ReLU의 단점을 보완하기 위해 여러 hand-desinged 대안들이 제안되었으나
- 그들의 inconsistencies(불일치) 문제로 성공한 것은 없다.
- 이 연구에서는 **신경망에서 음수 값의 이점을 활용하기 위해 RMAF(ReLU-Memristor-like Activation Function)라는 활성화 함수를 제안**한다.
- **RMAF는 상수 매개변수(α)와 threshold(임계값)매개변수(p)를 도입**하여 함수를 매끄럽고 단조롭지 않게 만들고 네트워크에 비선형성을 도입합니다.
- 논문에서 말하길 본인들의 RMAF가 ReLU 및 기타 활성화 함수보다 심층 모델과 여러 까다로운 데이터 셋에서 더 잘 작동한다고 한다.

1. 첫 번째로 시험한 데이터 셋은 위스콘신 유방암, MNIST, 아이리스, 자동차 평가 등의 벤치마크 데이터에 대해 다층 퍼셉트론(MLP)으로 학습 및 분류하여 실험을 수행한다.

    - 결과적으로 RMAF는 Sigmoid, Tanh 및 ReLU에 비해 각각 98.74%, 99.67%, 98.81% 및 99.42%의 높은 성능을 달성하였다.
2. 두 번째로 **MNIST, CIFAR-10 및 CIFAR-100 데이터에** 대해 합성곱 신경망**(ResNet)**에서 실험을 수행했으며 제안된 활성화 함수가 **Tanh, ReLU 및 Swish보다 각각 99.73%, 98.77% 및 79.82%의 더 높은 성능 정확도**를 달성했다.
3. 또한 딥 네트워크인 스퀴즈 네트워크(SqueezeNet), DenseNet121, 및 imageNet 데이터 셋에 대한 실험 또한 진행했다.

- **RMAF는 다른 함수보다 빠르게 수렴하며 효율성, 확장성 및 ReLU및 Swish와의 유사성으로 인해 모든 신경망에서 ReLU를 대체할 수 있다고 한다.**

Introduction

- 심층 신경망(DNN)은 input을 feature space로 변환하여 선형적으로 클래스를 분리하기 위해 활성화 함수에 의존한다.
- 활성화 함수는 뉴런의 활성화 여부와 뉴런이 받을 관련 정보를 결정한다.
- 이 논문에서는 **memristive window 함수를 사용한 새로운 활성화 함수 ReLU-Memristor-like activation function (RMAF)을 고안**했다.
- **Memristive Fully Convolutional Network(MFCN)**
    - memristors를 사용하여 이미지 분할을 수행하는 딥 러닝 네트워크
    - Memristor는 저항값이 가변인 비휘발성 메모리 소자
    - FCN (Fully Convolutional Network)의 변형으로, FCN과 마찬가지로 컨볼루션 연산을 사용하여 이미지의 특징을 추출하고, 이 특징을 사용하여 이미지를 분할한다.
    - 장점 : 전력 소모가 적고 memristors의 가변 저항 특성으로 인해 이미지 분할 수행시 높은 수준의 표현력을 제공한다.
    - 단점 : FCN과 비교하여 더 많은 연산을 수행하므로 비효율적이고, 설계 및 구현이 복잡하다
- ReLU-Memristor-like activation function (RMAF)는 이미지 분류, feature recognation 같은 실험에서 기존 ReLU및 기타 활성화 함수보다 성능이 우수함을 보여준다.
- 해당 논문은 또한 RMAF를 평가하기 위해 ResNet50(Residual Neural Network), Alexnet, SqueezeNet 및 DenseNet121고 같은 다양한 신경망을 구현하였다.

- 사용한 이미지 데이터 셋은 다음과 같다.
  - 1차원 데이터로 Wisconsin 유방암, Car, Iris, MNIST 데이터셋,
  - 2차원 이미지 데이터로 MNIST, CIFAR-10, CIFAR-100, ImageNet 데이터셋

Problem Statement

A. MEMRISTIVE WINDOW

- 선형 이온 drift 모델이라 불리는 멤리스터 모델은 memristive device의 선형 drift를 고려하여 제안되었다
- 이 멤리스터 모델은 해결해야 할 몇 가지 경계 효과 문제가 있는데
- 이를 바탕으로 scalability, boundary lock, 그리고 비선형 효과를 동시에 달성가능한 새로운 멤리스터 모델을 만드는 것이 중요했다.
- 멤리스터 모델링 하기 위해 비선형함수인 window 함수를 사용 한다.
  - window 함수는 선형 모델에서 인식되지 않는 경계 조건에 대한 설명을 위한 접근 방식이다.
- 여러 window 함수가 멤리스터의 모델링에 사용되었다.
- 이 논문은 boundary lock 현상을 해결하기 위한 새로운 window function을 찾는 논문에서 영감 받았다.
- *(B.1)-(B.4)*를 만족하는 윈도우 함수$f(x)$를 갖는 memristive 시스템으로 [36]의 (4)-(5)가 주어지면 내부 상태 변수는 S자형 곡선을 따른다.

$$x = S(q) = F^{-}1(\alpha(q - q_0))$$

- 여기서 sigmoid $x = S(q)$가 얻어지면, 다음을 적용해 $f(x)$를 계산한다.

$$\frac{1}{f(x)} = \alpha \frac{d}{dx}[S^{-1}(x)]$$

- transformation은 물리적, 생물학적 프로세스를 모델링하는데 사용되는 광범위한 sigmoidal 함수에서 windows를 achieve한다.
- sigmoidal은 분석적으로 window함수에서 부터 얻어진다.

$$f(x) = 1 - [(x - 0.5)^2 + 0.75]^p$$

- boundary lock 문제$(2x - 1)$를 해결하기 위해 $(x - stp(-i))$로 바꾼다.
- 따라서 다음과 같은 새 함수가 정의된다.

$$f(x) = 1 - [0.25(x - stp(-i))^2 + 0.75]^p \qquad (4)$$

- stp(short term plasticity)는 멤리스터 설계시 boundary lock 효과를 해결하기 위해 적용된다.
- i는 memristive 장치를 통해 흐르는 전류를 나타낸다.
- **stp**는 다음과 같이 정의된다.

$$stp(i) = \begin{cases} 1, if\ i \geq 0\ (v \geq 0) \\ 0, if\ i < 0\ (v < 0) \end{cases}$$

- 함수의 확장성을 개선하고자 window함수를 얻기 위해 (4)를 수정한다.
- 따라서 다양한 용도에 맞게 기능을 조정할 수 있다.

$$f(x) = j(1 - [0.25(x - stp(-i))^2 + 0.75]^p)$$

  - j : scale parameter
  - p : 적절한 j로 상향 or 하향 조정됨
  - 다른 j 선택시 window 기능이 다양한 응용프로그램에 맞게 조정될 수 있다.

B. The Proposed Method : RMAF

- 본 논문은 **스칼라 값을 입력 받아 스칼라를 출력하는 스칼라 활성화 함수를 찾는 것에 초점을 맞춘다.**
  - 스칼라 활성화 함수는 네트워크의 구조를 변경하지 않고 ReLU함수를 대체할 수 있기 때문이다.
- **RMAF에는 ReLU 및 Swish와 같은 속성이 포함**되어 있고,
- 임계값 p 및 j 하이퍼파라미터가 첨부되어 심층 신경망의 분류 정확도를 향상시킬 수 있다.
- **스칼라 생성과 DNN모델의 비선형성에 대한 초점**을 기반으로
- RMAF 활성화 함수를 구성하기 위해 수식을 수정한다.

$$f(x) = j(1 - [0.25(x - stp(-i))^2 + 0.75]^p)$$

  - 첫 번째 빼기 연산자를 (/)로, $x + stp(-1)^2$를 함수 $(1 + e^{-x})$로 대체하여 (6)을 수정한다.

$$RMAF(x) = (j(1/[0.25t(1 + e^{-x}) + 0.75]^p))$$

- 수정을 통해 위의 수식은 상수 or 훈련 가능한 파라미터 αx와 곱해졌고 $\alpha = 1$로 초기화 된다.

📌 **RMAF 활성화 함수**

$$RMAF(x) = \left[j \left(\frac{1}{(0.25 \cdot (1 + e^{-x}) + 0.75)^p} \right) \right] \cdot \alpha x$$

- **최종적으로 RMAF 활성화 함수**는 위와 같다
  - 파라미터 $p > 0$은 영역의 flatness를 제어하고

- 파라미터 j, p를 조정하여 ReLU와 같은 함수를 만들고 주어진 네트워크로 확장할 수 있다.

- **그림 1은 RMAF와 미분된 함수의 flot이다.**

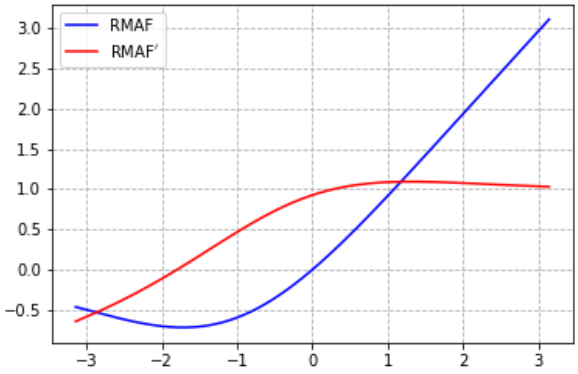
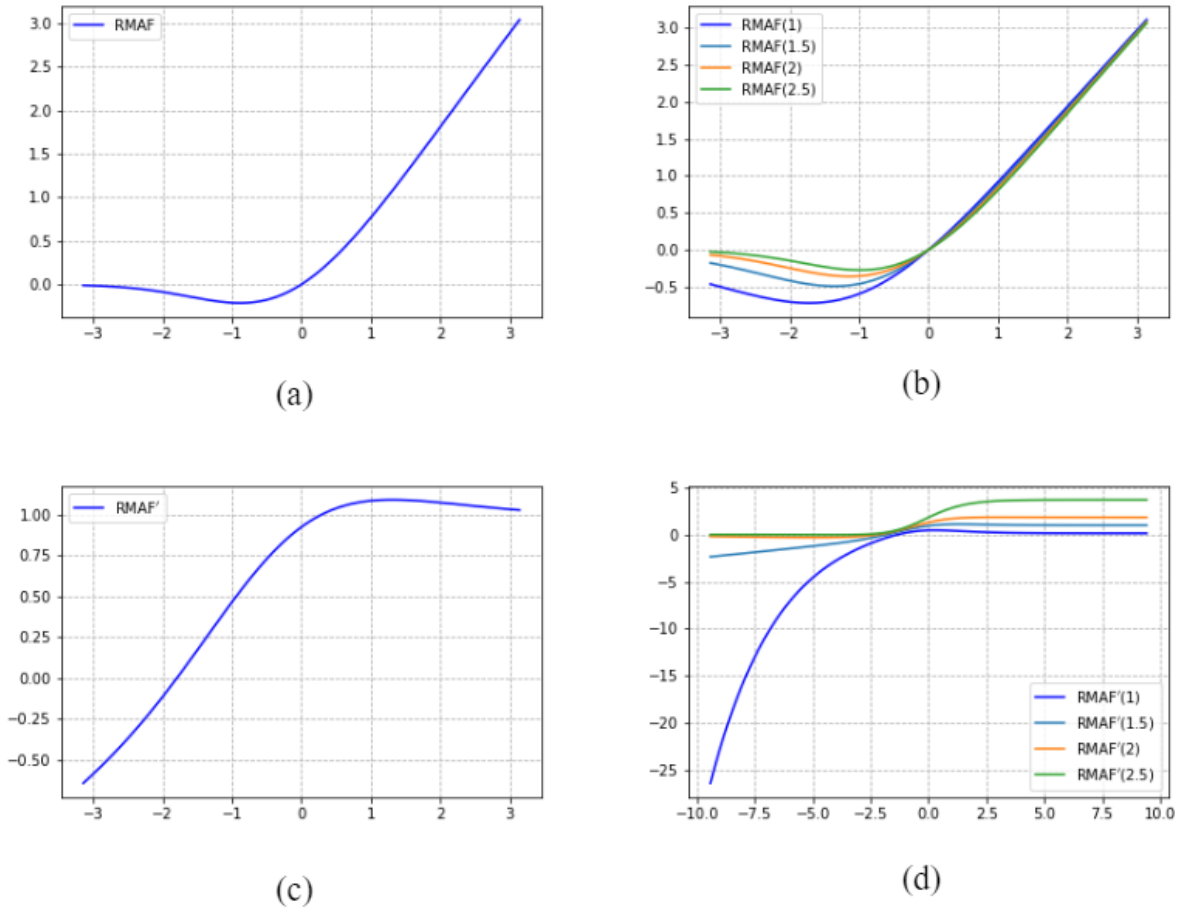
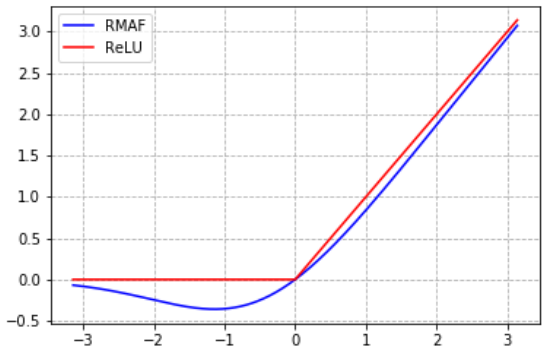


FIGURE 1: The proposed RMAF function and its derivative.

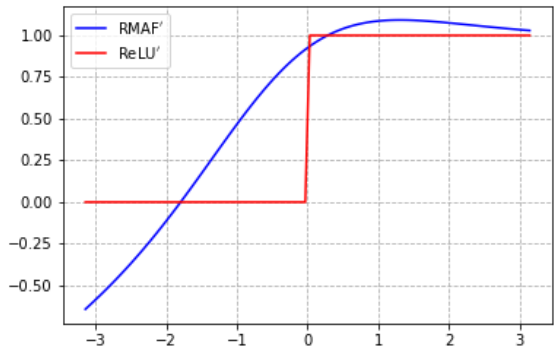
- 이를 통해 RMAF가 Google Brain에서 제안한 **Swish와 유사한 속(x≥0)**을 가졌음을 확인할 수 있었다.
- **그림 2**는 다르게 초기화된 p의 RMAF 함수 및 도함수를 보여준다.



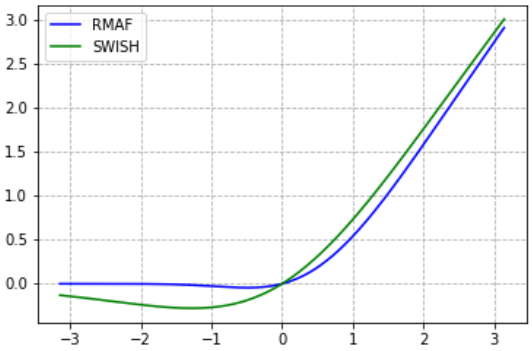
- **그림 3**에서 **ReLU 및 Swish와 RMAF의 비교**를 시각화한다.(함수도 같이)



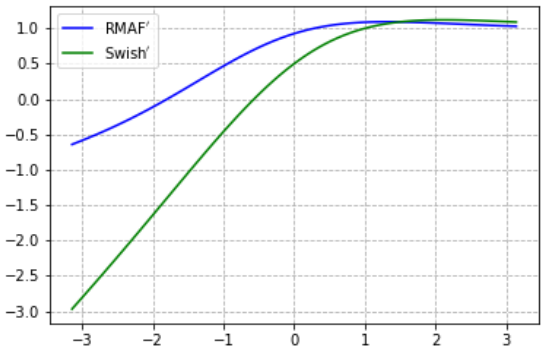
(a)



(b)



(c)



(d)

- Swish는 이미지 분류 문제의 여러 모델에서 ReLU보다 좋은 성능을 보여주지만
- Swish의 도함수는 nonsparse의 비율이 높아 네트워크의 계산 복잡도를 증가시킨다.

📌 0이 아닌 값이 많아 그 값의 도함수를 다 계산하기 때문에

- 하지만, RMAF는 순전파, 역전파 중에 대부분의 뉴런을 종종 비활성화하는 다른 쪽 ReLU의 hard zero속성을 유지할 수 있어
- RMAF함수는 데이터를 fitting하고 동적 특성에 따라 기본 형태에 대한 지식을 제공할 때 사용되는 모든 point다
  - 작은 매개변수 값은 RMAF함수를 ReLU함수와 매우 유사하게 만들고, 큰 매개변수 값은 RMAF함수를 더 평활하게 만든다.
  - 따라서 RMAF 함수는 ReLU 함수에서는 볼 수 없는 패턴을 식별할 수 있습니다. 이것은 RMAF 함수가 더 나은 예측을 하거나 더 현실적인 새로운 데이터를 생성할 수 있게

해준다.

- DNN은 여러 미분 가능한 함수로 구성되어 역방향 전파 동안, 도함수를 계산하여 매개변수를 업데이트한다.
- RMAF함수의 도함수는 다음과 같다.

$$f(x) = \frac{\alpha x}{[0.25(1+e^{-x})+0.75]^p}$$

j = 1일 때

- 편의상 $f_1(x) = \alpha x$, 그리고 $f_2(x) = [0.25(1+e^{-x})+0.75]^p$라고 하자
  그럼 아래와 같이 도출할 수 있다.

$$\frac{d[f_1(x)]}{dx} = \alpha$$

$$\begin{aligned}\frac{d[f_2(x)]}{dx} &= p[0.25(1+e^{-x})+0.75]^{p-1} \cdot \frac{d[0.25(1+e^{-x})+0.75]}{dx} \\ &= -0.25pe^{-x}[0.25(1+e^{-x})+0.75]^{p-1}\end{aligned}$$

- 아래의 compound 함수 유도 규칙에 따라

$$\frac{df(x)}{dx} = \frac{d[\frac{f_1(x)}{f_2(x)}]}{dx} = \frac{\frac{d[f_1(x)]}{dx} f_2(x) - \frac{d[f_2(x)]}{dx} f_1(x)}{f_2^2(x)}$$

- 다음을 얻을 수 있다

$$\begin{aligned}\frac{df(x)}{dx} &= \frac{\alpha f_2(x) + 0.25p\alpha x e^{-x}[0.25(1+e^{-x})+0.75]^{p-1}}{[f_2(x)]^2} \\ &= \frac{\alpha}{[0.25(1+e^{-x})+0.75]^p} \\ &\quad + \frac{0.25p\alpha x e^{-x}[0.25(1+e^{-x})+0.75]^{p-1}}{[0.25(1+e^{-x})+0.75]^{2p}}\end{aligned}$$

$$\frac{d}{dx}RMAF(x) = \frac{\alpha}{[0.25(1+e^{-x})+0.75]^{p}} + \frac{0.25p\alpha x e^{-x}}{[0.25(1+e^{-x})+0.75]^{p+1}}$$

RMAF class

```
class RMAF(nn.Module):
    def __init__(self):
        super(RMAF, self).__init__()
    # p=0.5, j=1, alpha=1
    def forward(self, x):
        rmaf=(1/(0.25*(1+torch.exp(-x))+0.75)**0.5)*x
        return rmaf
```