SimFet nemlinearitás használata CNN-ben

Benkó Barnabás - UOYDOC

May 2020

1 Bevezetés

A celluláris hullámszámítógépek, a neurális hálózatok elődjeiként, mátrixműveletek felhasználásával dolgoznak fel és manipulálnak bemenetként kapott képeket. A megfelelő számolások végén egy úgynevezett aktivációs függvény segítségével határozzák meg a tényleges kimenetet. Ezek a számítógépek legtöbbször egy lineáris függvényt használnak erre a célra, ami egyszerűbb tervezést eredményez, ez azonban a lehetséges kimeneket számát lekorlátozza. Ez a projekt ennek a linearitásnak a kiküszöbölésével és a két módszer különbségeivel foglalkozik.

2 Aktivációs függvény

Egy CNN a következő differenciál egyenletet számítja ki egy kép minden egyes pontjára:

$$\frac{dx}{dt} = -x + Ay + Bu + z$$

ahol $y = f(x)$ és $f(x)$ az aktivációs függény, $-x$ a pont értéke negálva, A a feedback template, B a control template és z a bias. Ez a képletben szereplő Ay fogja meghatározni, hogy milyen irányban kell változtatnunk a képpont értékét.

3 Karakterisztikák

3.1 Lineáris karakterisztika

A legtöbb esetben idáig egy lineáris függvényt, a szigmoid függvényt használtuk.

$$y = 0.5 * |x + 1| - |x - 1|$$

Ezzel a függvénnyel, az egyszerűsége miatt, relatíve könnyen tervezhetünk template-eket, azonban mivel lineáris, ezért bármilyen felhasznált template mátrixok lineáris kombinációja is egy lineáris függvényt eredményez. A kérdés az, hogy mi van, ha egyszer egy olyan műveletet akarunk kitalálni, amelyet lehetetlen megvalósítani szigmoiddal? A válasz az, hogy ez lehetetlen, lásd lineáris függetlenség definíciója. A megoldás egy nem lineáris függvény használatával lehetséges.

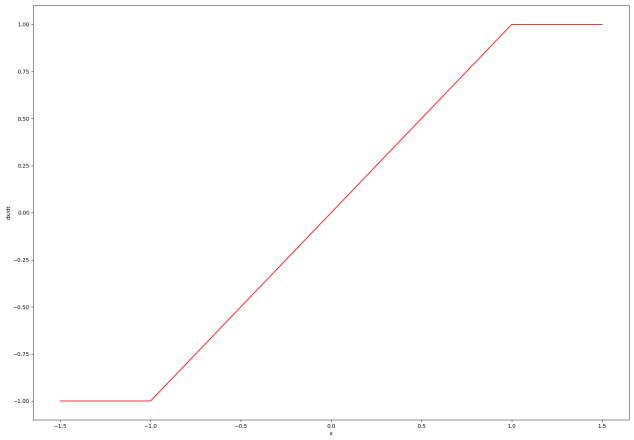


Figure 1: A szigmoid függvény képe

3.2 Nemlineáris karakterisztika

Aktivációs függvénynek gyakorlatilag bármilyen függvény megfelelhet, csupán a megfelelő template-eket kell biztosítanunk hozzá. Ebben a projektben a SimFet tranzisztor átviteli karakterisztikáját kellett használni és bemutatni. A függvény a következő:

$$I_{out} = I_0 + I_1 * exp(-((V_{in} - \frac{0.61 * V_{prg} + 0.47}{0.23})^2)$$

ahol V_{in} a forrás feszültség és V_{prg} a kapufeszültség. I_0 és I_1 konstansokként szerepelnek használatkor.
A függvény ábráján jól látszik a nemlinearitása, a kép leginkább egy keskeny eltolt gauss görbére hasonlít.

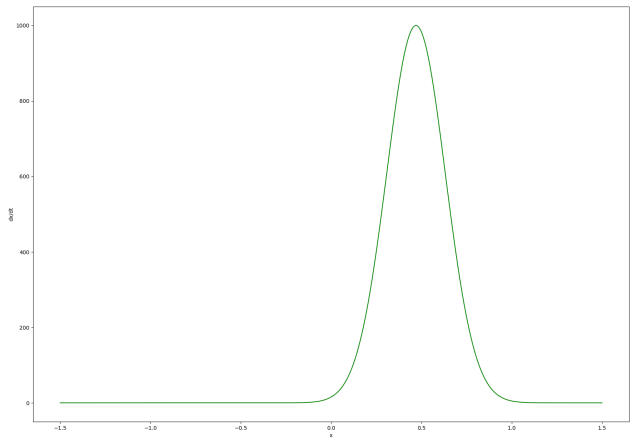


Figure 2: A SimFet karakterisztika képe

Ahhoz, hogy ezt mi megfelelően tudjuk használni, a kapufeszültség konstans -0.75-re állításával nullához igazítjuk a függvényt.

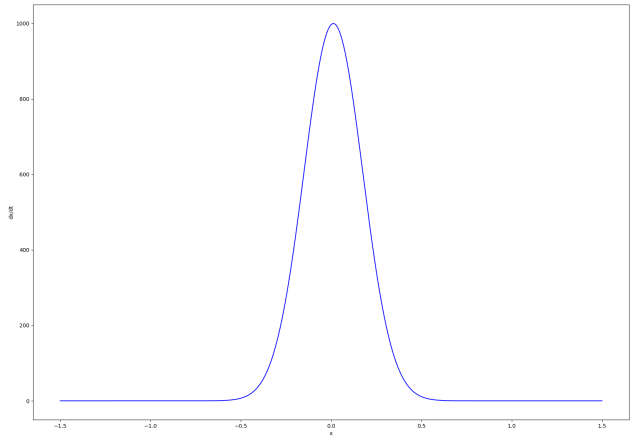


Figure 3: A SimFet karakterisztika eltolva

Meg kell figyelnünk, hogy a függvény nem nulla értékei egy nagyon kis tartományon vannak csak jelen, nagyjából a $[-0.5, 0.5]$ tartományon. Ez azt eredményezi, hogy $f(x) = \epsilon$ ha x nem eleme $[-0.5, 0.5]$, és $f(x) >> 1$ egyébként. A korábban említett ODE szerint ekkor, ha $B = 0$, $z = 0$ és $A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ akkor

$$\frac{dx}{dt} = -x + f(x)$$

Innen lépésenként nézve:
ha $x = 0$:

$$\frac{dx}{dt} \approx 0 + nagyon\ nagy\ szám$$

x új értéke $= 1$
ha $x = 1$:

$$\frac{dx}{dt} = -1 + \epsilon \approx -0.99$$

x új értéke ≈ 0
ha $x = -1$:

$$\frac{dx}{dt} = 1 + \epsilon \approx 0$$

x új értéke ≈ 0

Ebből a levezetésből látható, hogy nagyon kis lépésekkel kell dolgoznunk, hiszen nagyon könnyen egy végtelen ciklusba kerülhetünk a függvény határain.

4 Megvalósítás

A kódban az új karakterisztikák a $CNN.non\_linearity()$ függvény valósítja meg.

```
@staticmethod
def non_linearity(v_in, v_prg=-0.75):
    """Piece-wise non-linear simfet function.

    Args:
        v_in : Input to the non-linearity.
        v_prg: Voltage for the Gate.
    """

    # RF
    voltage_conversion = 1e4

    # I_0
    i_0 = 10e-10

    # I_1
    i_1 = 10e-2

    return voltage_conversion * (i_0 + i_1 * np.exp(-((v_in - (0.61 *
        v_prg + 0.47)) / 0.23) ** 2))
```

Látható a fixált kapuáram és a kis i_0 és i_1. Az eredményt még módosítanunk kell, hogy áramokból feszültség értékeke kapjunk.

5 Eredmények

A tapasztalatok alapján az új függvény néhány esetben elvégezte a feladatát az eredeti template-eket használva, de ezekben az esetekben a kimeneti képen általában felcserélődött a fekete és a fehér pixelek színe. Eredeti *EdgeDetection* eljárást használva például a kép teljesen fekete lett, kivéve az éleket, amik fehérek lettek.

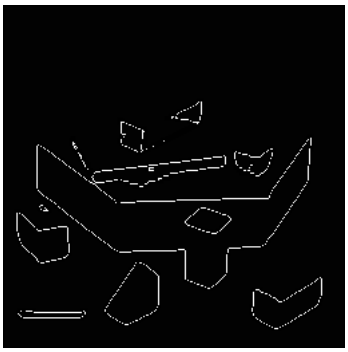
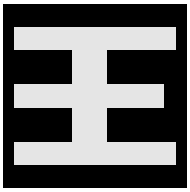
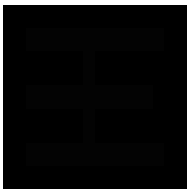


Figure 4: Invertálódott output

Az egyszerűbb logikai operátorok esetében, szükséges újragondolni a template-eket.



(a) Új template



(b) Régi template

Figure 5: Nemlineáris függvény különböző template-ekkel

Az eredeti logikai invertálás:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \; B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \; z = 0$$

Az új logikai invertálás:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \; B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \; z = -1$$

További példák megtalálhatóak a kódban, illetve a csatol képek között.