Hvordan "tænker" et Deep Neural Network (DNN) ?

Vi har tidligere arbejdet med XOR, og fandt frem til, at den **ikke** kan løses med ét lag af neuroner. Det var nødvendigt at tilføje et hidden layer. Da kunne modellen lære at predicte en XOR gate korrekt. Nedenfor ser du to XOR modeller fra Simulatoren med et hidden layer med to neuroner. Model 1 og Model 2 ligner hinanden, bortset fra vægtene, som bliver tildelt tilfældigt ved start og derefter trænet indtil den predicter rigtigt ved alle fire input.

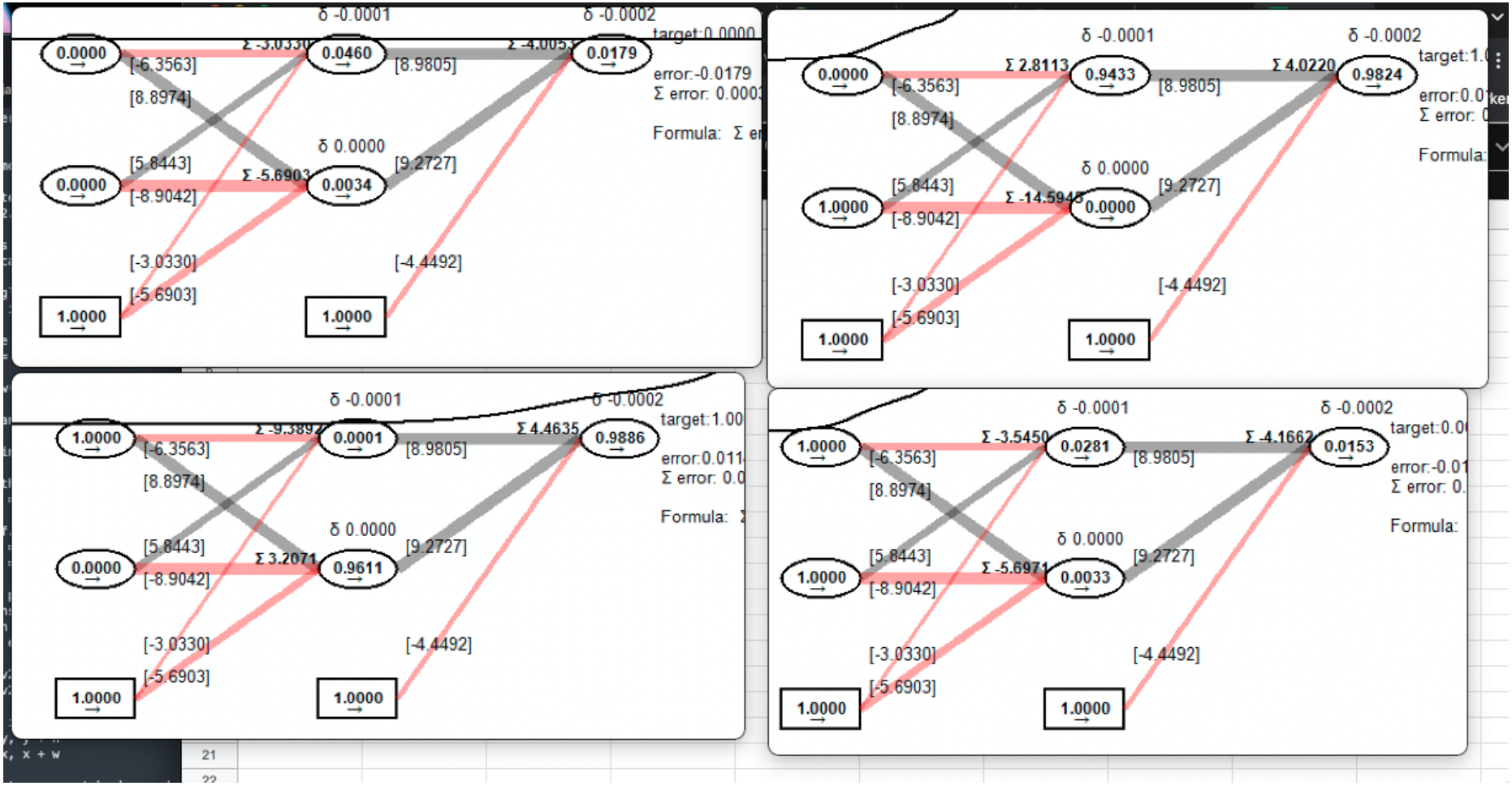
Opgave 1:

Du skal analysere de to modeller for at finde frem til en beskrivelse af, hvordan det neurale netværk (DNN) har "tænkt" for at løse XOR:

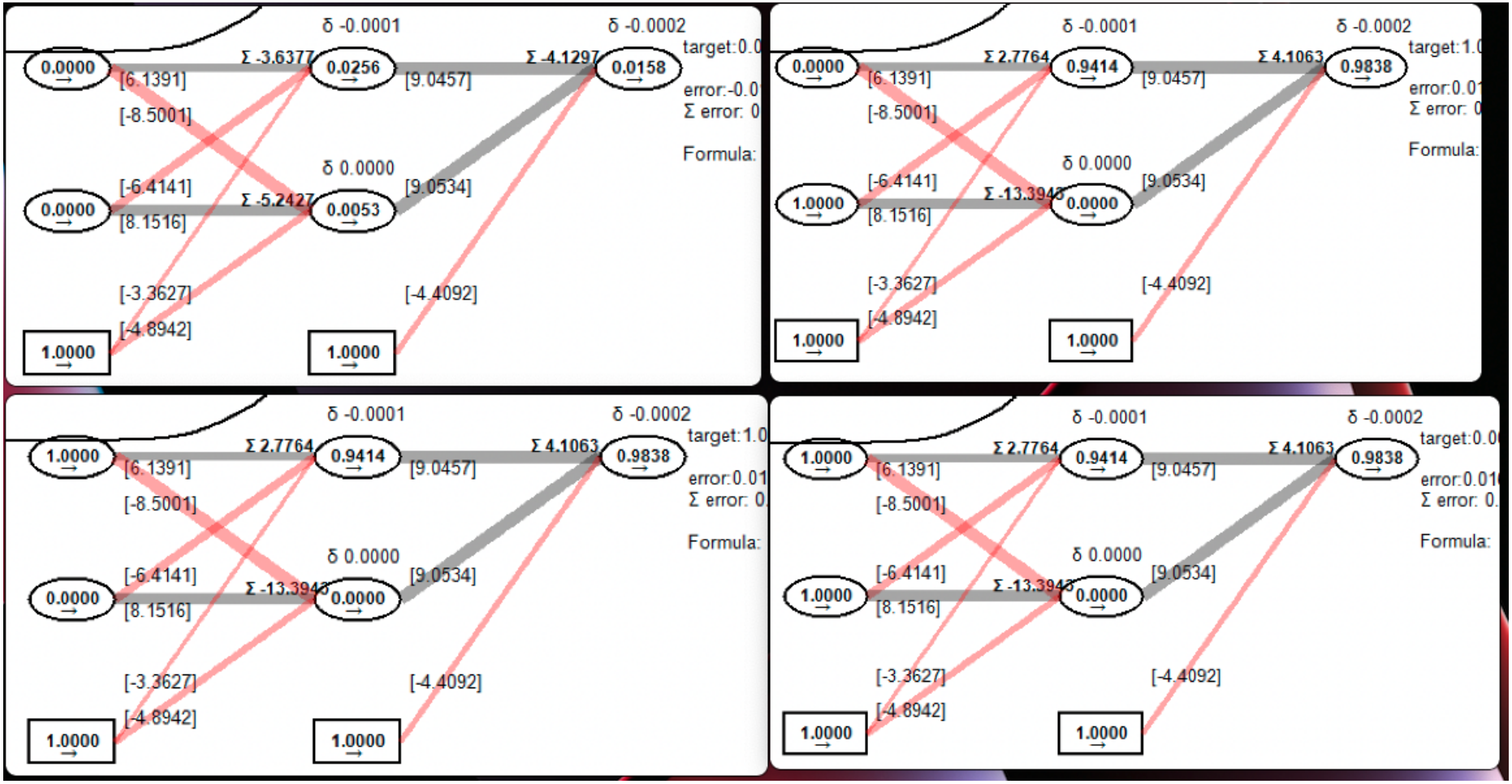
1. Hvordan vil du beskrive Model 1's løsning?
2. Hvordan vil du beskrive Model 2's løsning?
3. Kan du finde en fælles beskrivelse af Model 1 og Model 2?

Du er velkommen at køre Simulatoren selv, for at undersøge om din beskrivelse passer, eller om du får en helt anden type løsning end Model 1 og Model 2. **Er tilbage kl 13.05**

Model 1:


δ -0.0001
δ -0.0002
target:0.0000
0.0000
[-6.3563]
Σ -3.0330
0.0460
[8.9805]
Σ -4.005
0.0179
error:-0.0179
Σ error: 0.000
[8.8974]
Formula: Σ er
δ 0.0000
[5.8443]
0.0000
Σ -5.6903
0.0034
[9.2727]
[-8.9042]
[-3.0330]
[-4.4492]
[-5.6903]
1.0000
1.0000
δ -0.0001
δ -0.0002
target:1.0
0.0000
[-6.3563]
Σ 2.8113
0.9433
[8.9805]
Σ 4.0220
0.9824
error:0.0
Σ error: 0
[8.8974]
Formula:
δ 0.0000
[5.8443]
1.0000
Σ -14.5945
0.0000
[9.2727]
[-8.9042]
[-3.0330]
[-4.4492]
[-5.6903]
1.0000
1.0000
δ -0.0001
δ -0.0002
target:1.00
1.0000
[-6.3563]
Σ -9.3892
0.0001
[8.9805]
Σ 4.4635
0.9886
error:0.011
Σ error: 0.0
[8.8974]
Formula: Σ
δ 0.0000
[5.8443]
0.0000
Σ 3.2071
0.9611
[9.2727]
[-8.9042]
[-3.0330]
[-4.4492]
[-5.6903]
1.0000
1.0000
δ -0.0001
δ -0.0002
target:0.0
1.0000
[-6.3563]
Σ -3.5450
0.0281
[8.9805]
Σ -4.1662
0.0153
error:-0.01
Σ error: 0.
[8.8974]
Formula:
δ 0.0000
[5.8443]
1.0000
Σ -5.6971
0.0033
[9.2727]
[-8.9042]
[-3.0330]
[-4.4492]
[-5.6903]
1.0000
1.0000

Model 2:



Fin-tuning af modellen.

Simulatoren giver flere muligheder for at fin-tune modellen.

I modellerne ovenfor har jeg brugt Bias på begge lag. Hvis man ikke har brug for Bias, kan den nemt **fjernes** ved at sætte dens input til 0.
Dertil har jeg brug sigmoid aktiveringsfunktion. Men du kan vælge imellem 4 aktiveringsfunktioner:

1. sigmoid
2. tanH
3. ReLU
4. HardLimiter

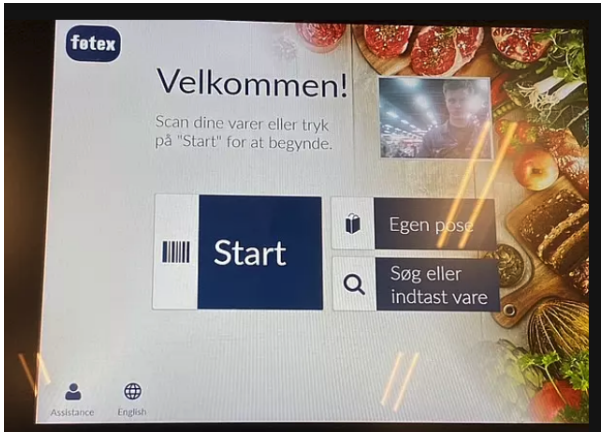
Opgave 2:

Kan du få modellen at lære XOR ved at bruge færre neuroner, end de 7 som bruges i modellerne ovenfor?

Vi gennemgår analysen på tavlen bagefter.

Grøntsagscanner til Supermarked

I flere supermarkeder findes selv-scan kasser, som vi kunder er glade for. Man tager en vare, scanner barcoden og lægger varen på hylden til højre.
Indtil videre kan scanneren dog ikke håndtere grøntsager og frugter. Her skal kunden selv finde varen fra en menu på skærmen.



Opgave 3: er tilbage kl 14.20

Følgende datasæt har 15 forskellige grøntsager. Din opgave er at lave og træne et Convolutional Neural Network (CNN), som kan forudsige hvilken grøntsag kunden kommer med. Der forventes en accuracy på min. 82%. Brug evt. denne kode som start og tilpas til denne opgave.

Kamera

Vi er nu kommet til hardware-delen. Vi har brug for et kamera, som står klar til at filme en grøntsag, og umiddelbart efter oplyse, hvilken grøntsag det er.
Colab kan ikke så nemt tilgå kamera via Python kode. En løsning er at installere et Python IDE og så køre projektet lokalt.

Opgave 4 (ekstra):

a) Installér et Python IDE på din PC. F.eks. Pycharm, som er gratis for KEA studerende.
b) Installér cv2 (henter video fra kamera)
c) Installér tensorflow
    i) Mac brugere kan bruge denne guide.
d) Brug denne Python kode, til at få kamera til at vise video.
e) Gem den trænede model fra Opgave 3, og download filen til din PC.
f) Tilføj kode til while(True) loopet i filen fra punkt c), som:
    i) importerer tensorflow
    ii) loader den trænede model fra Opgave 3
    iii) bearbejder billedet fra kamera, så det kan bruges som input til modellen (på samme måde som punkt 10 i guiden.)
    iv) bruger modellen til at forudsige hvilken grøntsag, der findes på billedet
    v) udskriver svaret på skærmen.
g) Tip: Eftersom kamera tager 20-50 billeder pr. sekund, kan du vælge at tage et gennemsnit af predictions af f.eks. 20 billeder. Så bliver teksten mere "rolig".

Aflevering:

Lav en max 2 min. video af din løsning, og send LINK til Fronter. Demonstrér hvordan det virker, og fortæl én ting, du har lært.

jone@kea.dk
Okt. 2022