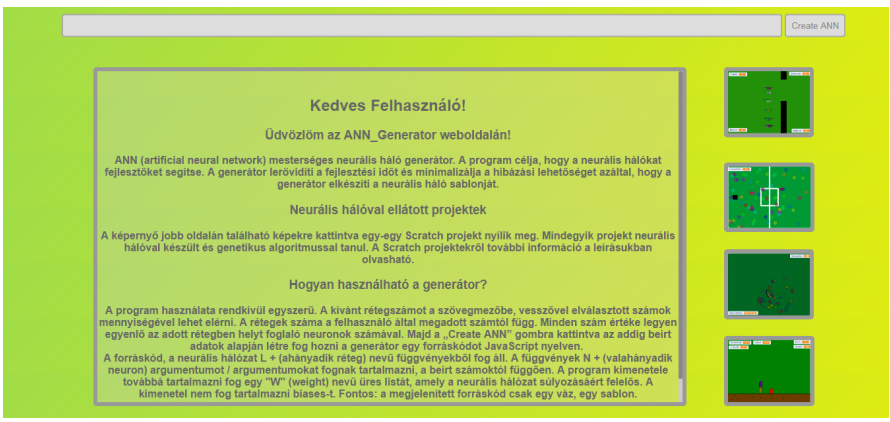
ANN_Generator

Laczkó Örs
Felkészítő tanár: Kondorné Kovács Irén
Kerék Általános Iskola és Gimnázium,
1035 Budapest, Kerék utca 18-20.

1. Bevezetés

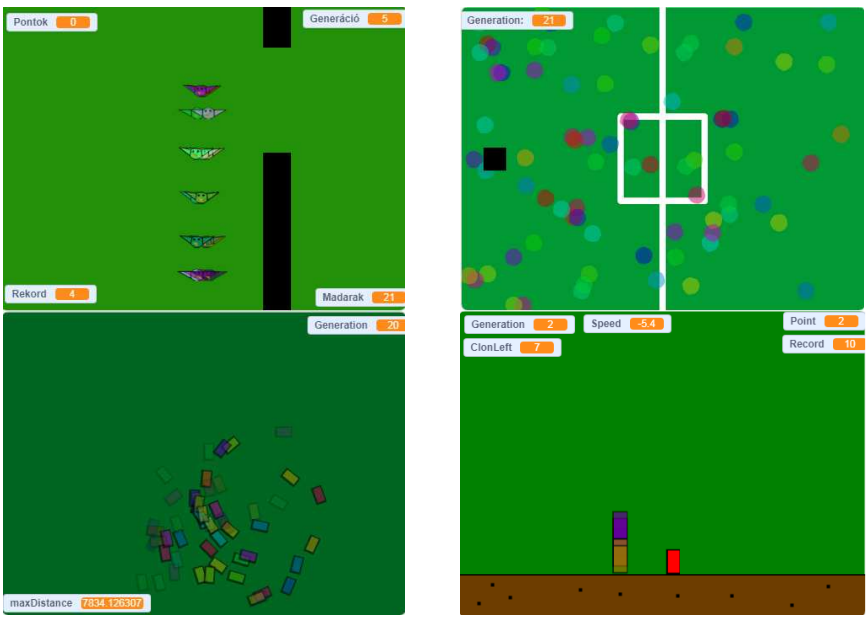
Számos mesterséges intelligenciával és/vagy neurális hálókkal való foglalkozást megkönnyítő szoftver található az interneten, ilyenek például a TensorFlow és a PyTorch. A legtöbb ilyen alkalmazás használata a kezdő programozóknak rendkívül bonyolult, nehezen követhető. Ezen programokat leginkább a Python programnyelv támogatja. Ugyan magam is használom a Pythont, de jelenleg webfejlesztéssel foglalkozom. Számos projektemhez használok neurális hálókat, melyek magas neuron és/vagy réteg számánál a manuális programozás sok időt igényel, továbbá exponenciálisan nő a hibázás lehetősége. Ezen probléma megoldására gondoltam ki, majd fejlesztettem az ANN_Generator nevű webfelületet. Működése – ami az oldalon is olvasható – rendkívül egyszerű. Az oldal tetején lévő párbeszédablakba (vesszővel elválasztva, szóköz nélkül) be kell írni minden kívánt réteg neuron számát, például: [2,3,1]. Majd a „Create ANN" gombra kattintani, s megjelenik az oldalon elhelyezett szürkés alapú paragrafusban a program által generált forráskód (1 ábra).



1. ábra

A forráskód vágólapra helyezése, majd beillesztése után szabadon felhasználható számtalan projektben. Az ANN_Generátor web felületének jobb oldalán négy Scratch fejlesztőkörnyezetben megtekinthető projekt van elhelyezve. Mindegyik projekt neurális hálóval készült és genetikus algoritmussal tanul. A Scratch projektekről további információ a leírásukban olvasható (2. ábra).

Elérhetőek az alábbi linkeken (AI – artificial intelligence):
AI bird https://scratch.mit.edu/projects/238932969/
AI football https://scratch.mit.edu/projects/246768302/
AI car https://scratch.mit.edu/projects/249160965/
AI rectangles https://scratch.mit.edu/projects/239441108/



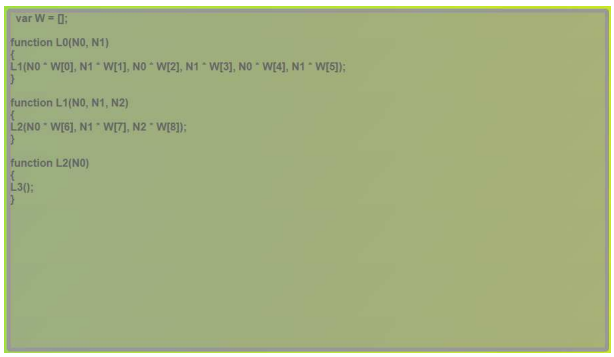
2. ábra

Jelenleg is több projekten dolgozom, melyekhez neurális hálót használok, például egy adattömörítő program, valamint egy olyan alkalmazás, amely egy számsorba megkeresi annak rendezési szabályát, s az alapján fog következtetni a következő elemekre. Ezekben a projektekben óriási segítségemre van az ANN_Generátor.

2. A probléma megoldásának menete, az ANN_Generátor létrehozása

A probléma megoldásának az első lépése paradox módon a végeredmény volt. Először végig gondoltam, hogy milyen kimenetet szeretnék, hogy milyen

output lenne számomra az ideális. Ezt követően a program felhasználói interfészét terveztem meg, hogy milyen inputokra van szükség, illetve, hogy mi az ideális elosztásuk az adatfogadó és megjelenítő elemeknek. Miután minden terv elkészült, először a JavaScript kódot írtam meg, melynek az első része a felhasználói bemenet kezelése, feldolgozása. Működése: a feldolgozott adatlista egy összetett függvényrendszerbe kerül, ahol miden elem (a felhasználó által megadott szám) kap egy metódus fejlécet, argumentumot/argumentumokat, valamint egy metódus törzset. Az így kapott függvény stringek egy listába kerülnek. A függvények legenerálása után egy for ciklus végig lepkéd az elemeken és belesűríti egy változóba, majd megjeleníti az „output" id-vel ellátott paragrafusba a változó tartalmát (3. ábra).

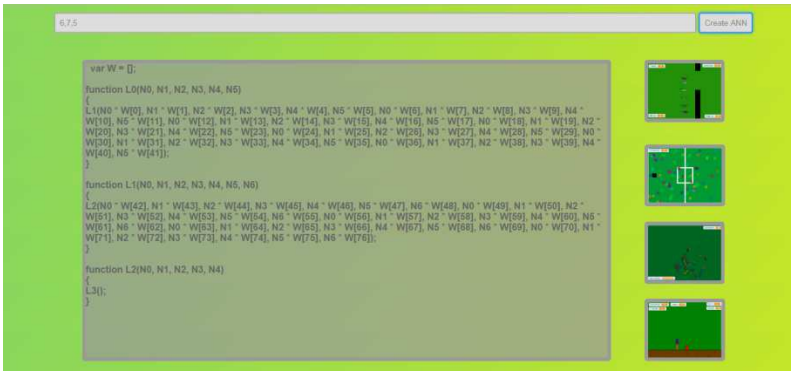


3. ábra

Második lépés volt a weboldal formázásának megoldása, az oldal struktúrájának a kialakítása. Idetartoznak az elemek elhelyezése és méretezése, mértékük természetesen százalékban van megadva, így minden képernyőn ugyanolyan megjelenésű. Idetartoznak továbbá a stilisztikai elemek is, mint például a színek, betűtípusok, kiemelések, illetve a görgetősáv megformázása. A programablak tartalmaz még CSS animációkat is, ilyen például a színváltós háttér és az érintésre felvillanó elemek.

3. Elért eredmények

A program jelenlegi állapotában a felhasználótól kapott/beolvasott érvényes adatokat feldolgozza és eredményül hibátlan JavaScript kódot generál a kapott adatok alapján. A program hatékonysága a neurális háló méretével nő. Például egy 6 input 7 hidden-layer 5 output neuron méretű neurális háló legenerálásához a következőt kell beírni a szövegablakba: 6,7,5 (4. ábra) . Eredményül az 1. táblázatban található forráskód keletkezik.



4. ábra

1. Generált forráskód

```
var W = [];

function L0(N0, N1, N2, N3, N4, N5)
{
L1(N0 * W[0], N1 * W[1], N2 * W[2], N3 * W[3], N4 * W[4], N5 * W[5], N0 * W[6], N1 * W[7], N2 * W[8], N3 * W[9], N4 * W[10], N5 * W[11], N0 * W[12], N1 * W[13], N2 * W[14], N3 * W[15], N4 * W[16], N5 * W[17], N0 * W[18], N1 * W[19], N2 * W[20], N3 * W[21], N4 * W[22], N5 * W[23], N0 * W[24], N1 * W[25], N2 * W[26], N3 * W[27], N4 * W[28], N5 * W[29], N0 * W[30], N1 * W[31], N2 * W[32], N3 * W[33], N4 * W[34], N5 * W[35], N0 * W[36], N1 * W[37], N2 * W[38], N3 * W[39], N4 * W[40], N5 * W[41]);
}

function L1(N0, N1, N2, N3, N4, N5, N6)
{
L2(N0 * W[42], N1 * W[43], N2 * W[44], N3 * W[45], N4 * W[46], N5 * W[47], N6 * W[48], N0 * W[49], N1 * W[50], N2 * W[51], N3 * W[52], N4 * W[53], N5 * W[54], N6 * W[55], N0 * W[56], N1 * W[57], N2 * W[58], N3 * W[59], N4 * W[60], N5 * W[61], N6 * W[62], N0 * W[63], N1 * W[64], N2 * W[65], N3 * W[66], N4 * W[67], N5 * W[68], N6 * W[69], N0 * W[70], N1 * W[71], N2 * W[72], N3 * W[73], N4 * W[74], N5 * W[75], N6 * W[76]);
}

function L2(N0, N1, N2, N3, N4)
{
L3();
}
```

1. táblázat

A projektekben kedvenc témámmal foglalkoztam, a mesterséges intelligenciával. Az MI fejlesztésében látom mind magam, mind az emberiség jövőjét. Hiszek abban, hogy ennek a projektnek köszönhetően mások is könnyebben, kevesebb idő ráfordítással készítenek neurális hálókat tartalmazó programokat.