

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Szilágyi Krisztián Gergely

Autonóm jármű fejlesztése

Buggy projekt

Konzulens

Szemenyei Márton

BUDAPEST, 2020

Tartalomjegyzék

[Összefoglaló 3](#_Toc41478180)

[1 Irodalomkutatás 4](#_Toc41478181)

[1.1 Reinforcment Learning 4](#_Toc41478182)

[1.2 PyBullet 4](#_Toc41478183)

[2 Felhasznált technológia 5](#_Toc41478184)

[2.1 Colaboratory 5](#_Toc41478185)

[2.2 PyTorch 6](#_Toc41478186)

[3 Architektúra 7](#_Toc41478187)

[3.1 Felépítése 7](#_Toc41478188)

[3.1.1 Environments 8](#_Toc41478189)

[3.1.2 Multi-head attention 8](#_Toc41478190)

[3.1.3 A2C 8](#_Toc41478191)

[3.1.4 RAdam 8](#_Toc41478192)

[3.2 Tesztelés, eredmények 9](#_Toc41478193)

[Irodalomjegyzék 10](#_Toc41478194)

Összefoglaló

A feladatom egy az Irányítástechnika és Informatika Tanszéken fejlesztett autonóm BB-8 droid beszéd felismerésének és hangvezérlésének megalkotása az Önálló laboratórium tárgy keretein belül. A feladat komplexitása miatt a második pontra, vagyis a parancsok teljesítésére, valamint a megalkotott rendszer beágyazására nem jutott idő ebben a félévben. A beszámolómban időrendi sorrendben mutatom be a feladatom elvégzését, a végén leírom a távlati célokat is, amelyeket a következő félévben kell teljesíteni.

A robot a szenzorjai és feldolgozó egységének köszönhetően rengeteg funkcióval rendelkezik, egy több fős csapat dolgozik ezeknek a funkcióknak az implementálásán. Az én feladatom a hang felismerés és hangvezérlés. Ehhez segítségünkre van négy mikrofon a robot fejegységében, valamint egy NVIDIA Titan X, ami a nagy teljesítményének köszönhetően gyorsabb fejlesztést tesz lehetővé, hiszen a neurális hálók paramétereinek beállítása (a tanítás) rendkívül számításigényes éppen ezért időigényes folyamat. Egy komolyabb struktúra betanítása akár napokban mérhető. Nálunk inkább percekben vagy órákban még a komoly teljesítmény ellenére is.

Alapvetően a feladat egy speech-to-text (STT) működés létrehozása neurális háló segítségével. Felügyelt (supervised) tanítást alkalmazva kéne létrehoznunk egy osztályozót, amely a bemenetére kapott hangból, vagyis amit feldolgoznak a mikrofonok, képez egy karakterláncot és eldönti, hogy ez érvényes utasítás-e és ha igen, melyik utasításnak felel ez meg. Ennek a további feldolgozása, végrehajtása már a csapat más tagjainak a feladata. Mivel nulláról létrehozni egy mély neurális hálót (DNN) túl nagy feladat, ahhoz, hogy egy félév alatt erre sor kerüljön, ezért úgynevezett transfer learning-et fogunk alkalmazni. Ennek lényege, hogy egy nagy adatbázissal előre betanított, kész hálót használunk fel és finom hangoljuk a saját adatainkkal.

# Irodalomkutatás

A projektet megvalósítása egy hosszútávú cél, így ebben a félévben a projekt szkópja főleg az irodalomkutatás és a tervezés volt, nem a megvalósítás és tesztelés. Így a hónapokat leginkább tanulással töltöttem, megismerkedtem a megerősítéses tanulásban használatos fogalmakkal és ezek alapján a cél volt összerakni egy kezdeti architektúrát, melyet a későbbiekben továbbfejlesztve megvalósíthatunk egy teljesen autonóm módon működő jármű szoftverét.

## Reinforcment Learning

## PyBullet

# Felhasznált technológia

## Colaboratory

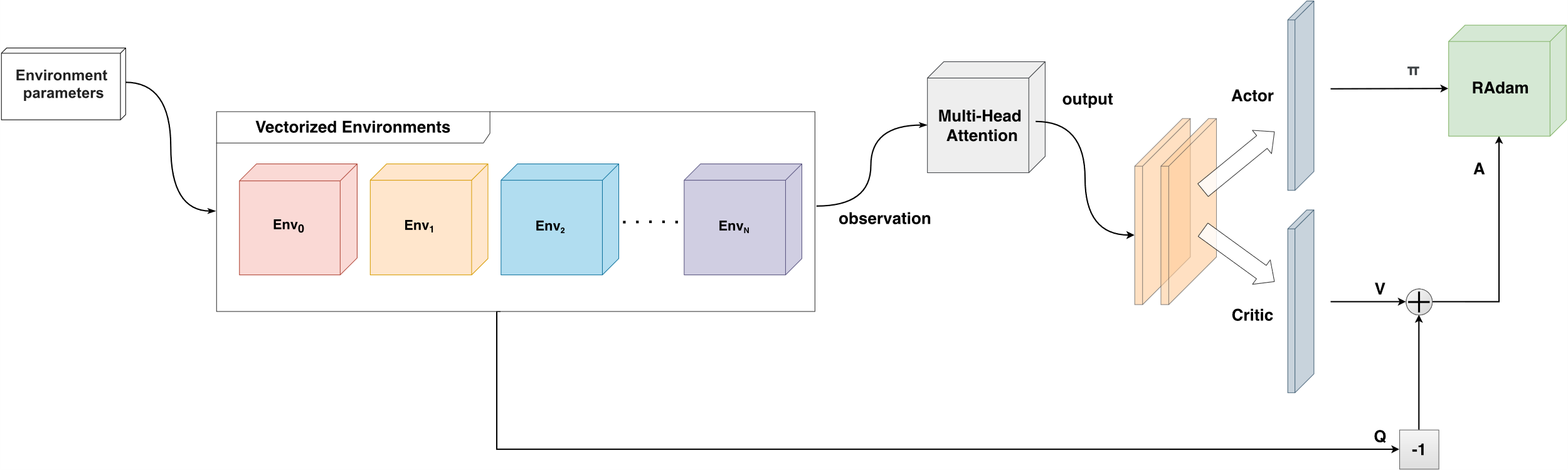
## PyTorch

# Architektúra

A teljes

## Felépítése

Az architektúra több logika részre bontható, a magja az A2C metódust megvalósító kétfejű neurális hálózat. A hálózat bemenetére helyezünk egy Multi-Head attention blokkot, melynek a bemenete a környezetekből érkező megfigyelések tenzora. Mivel A2C-t használunk, így logikus több környezetet futtatnunk egyszerre, hardware erőforrásainktól függően akár 16-ot is tudunk egyszerre. A saját hardware konfigurációmmal 1-4-t futtattam egyszerre. A háló Actor fejének a kimenete a stratégia, melyet a RAdam (Rectified Adam) optimizer követ. A Critic fej kimenete az Érték, melyből a Q-t (diszkontált jutalom) kapjuk meg az előnyt (a baselinet), pontosabban annak inverzét. A Q-t is a környezetben elvégzett akció után kapjuk meg, az előnyt is átadjuk az RAdam-nek.



3.1. ábra

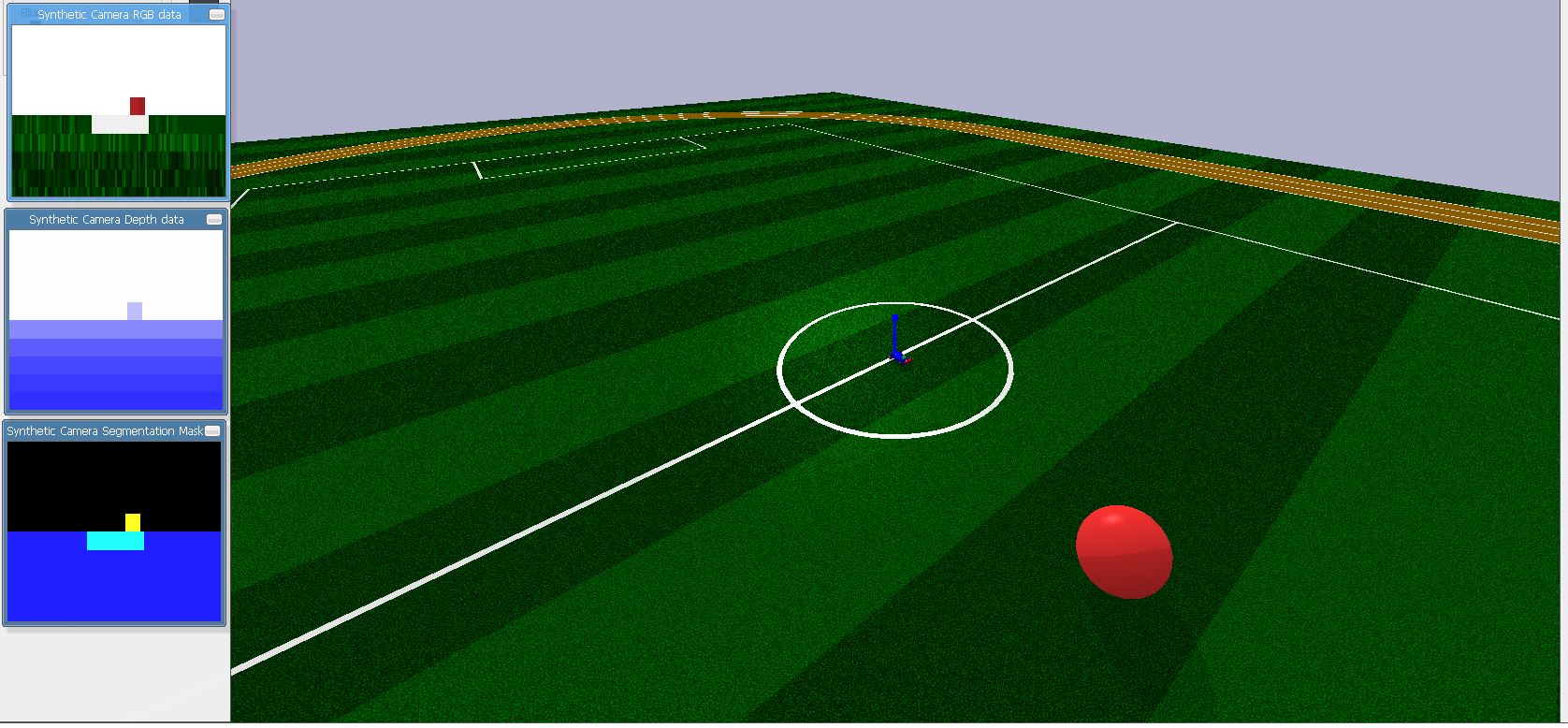
### Environments

A félév során több környezettel is próbálkoztunk. Először a SafetyGym-mel, melyről kiderült, hogy a MuJoCo fizikai motort használja, melyre ingyen csak diákként lehet éves licenszt szerezni egy adott gépre. Ez kizárja annak a lehetőségét, hogy Colab-on tudjuk használni, valamint a jővőbeli fejleszthetőséget kockáztatjuk meg azzal, ha nem kapunk licenszt vagy csak az eredti árán, mely 500€. Így másik motor után kellett nézni, végül a PyBullet-re esett a választás. Ez teljesen ingyenes így Colabon is lehet futtatni. Azonban itt sok problémába ütköztem. Mivel a „debug”-hoz, hogy követni tudjuk az ágens akcióit a környezetben igen hasznos lenne, ha meg tudnánk jeleniteni a környezetet, legalább a teszteléshez, még ha csak a tanításhoz nem is. Így az egy fontos szempont volt, hogy a Colabon meg tudjuk ezt valósítani. Sajnos mint kiderült ez egy nagyon nehezen megvalósítható feature, egyszerűbb kkörnyezetekkel, mint például az Atari ezt sikerült elérni oly módon, hogy videót készit a környezetből és a futás végén ezt visszanézhetjük. De a PyBullet környezet a Colabbal nehéz megnyitni és sajnos nem lehet videóra rögziteni sem. Így a Colabról egyelőre le kell mondanunk, amíg nem lesz tökéletes a szoftver működése és nem lesz kész hozzá a környezet. Így végül ki kellett javítanunk a TF hibákat és a saját gépemen megoldani a futást.

A feladatunkhoz kettő hasonló környezet van már kiépítve, egy autónak kell eljutnia egy nagy üres pályán (jelenleg focipályán) a pályaközepétől egy random elhelyezett labdához. A két környezet abban különbözik leginkább, hogy míg az egyiknél a megfigyelés csak a labda poziciója a kamera képén (x,y), addig a másiknál a teljes rgbd kamera kimenete. Az utóbbi környzetet fogjuk a jővőben tovább fejleszteni, a saját feladatunkra szabni.

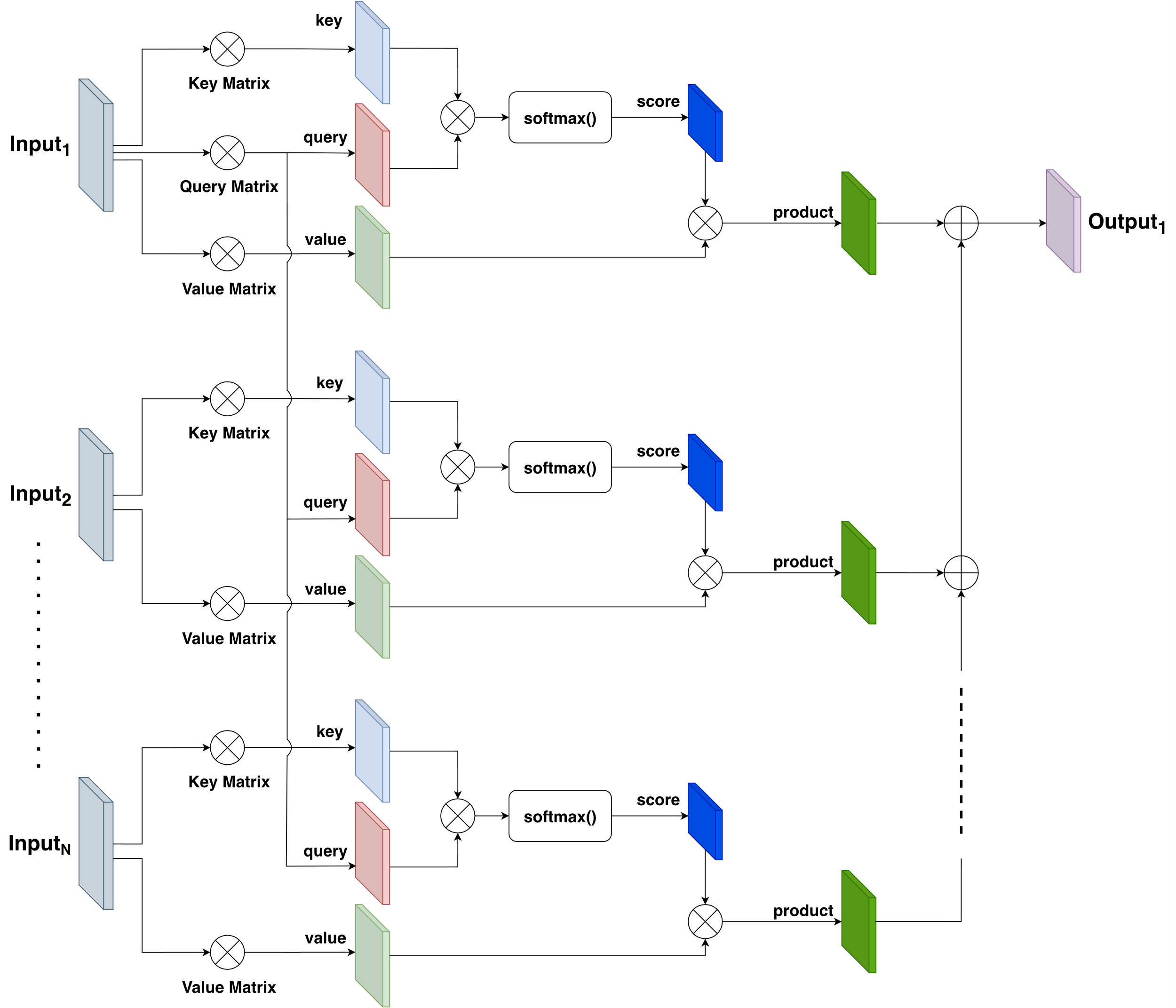
A sub proc vecenv-vel oldottuk meg a vectroizálást. 5-6 rollout.

A felbontáson már változtattam egyelőre csak ennyit nyúltam bele az environment-be, át kellett írni a szerzett A2C-t hogy fogadni tudja helyesen az observationt.



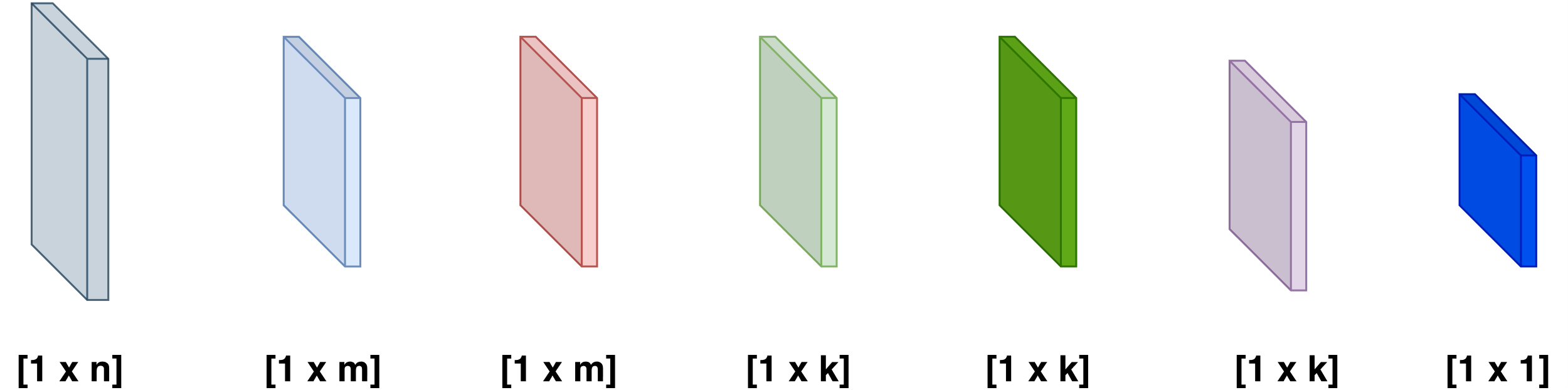
### Multi-head attention

Ezt a függvényt nem olyan régen bevezették a PyTorch-ban, így nem kell az implementálásával foglalkozni. A működése:



3.2. ábra

Ahol a mátrixok méretei az alábbiak:



3.3. ábra

### A2C

### RAdam

A Rectified Adam egy módosított Adam (Adaptive Moment Estimation), ez a state-of-the-art optimalizáló eljárás. Azonabn mielőtt rátérnék, hogy miért jó a Rectified Adam, előbb nézzük meg, hogy működik az Adam. Az Adam két ismert algoritmus, az RMSProp és az AdaGrad jó tulajdonságait ötvözi, célja a nevéből is adódóan az adaptív tanulási sebesség, akárcsak az RMSPropnál (viszont itt a gradiens négyzetek összegzésén kívül a gradienseket is összegezzük), de kijavítja az AdaGrad nagy hátrányát, az időben csökkenő tanulási sebességet. A pramatérei, az alfa, mely igazából a tanulási ráta vagy lépéshossz (éta), a béta1 és béta2, melyek a gradiensek első (átlag) és második momemntumának (középnélküli varianciájának) exponenciális felejtési rátája, ezek 1 körüli értékek. Valamint szükség van még az epszilonra, mely a numerikus stabilitást biztosítja, azaz, hogy a nevező érétke sose lehessen nulla. A publikáció író ajánlási alapján ezeket a következő módon szokás beállítani: alfa = 0.001, béta1 = 0.9, béta2= 0.999 és epszilon = 10^-8. Ezek segítségével számljuk ki ezeket, ahol gt a gradiensek, Theta a háló paraméterei.

|  |  |  |
| --- | --- | --- |
|  |  | (3.x) |
|  |  | (3.x) |
|  |  | (3.x) |

Egy további korrekciót kell még alkalmazni, ha netán a gradiensek átlaga és a gradiens négyzetek átlaga kezdetben nagyon kis értékűek lennének, nehogy beragadjon ilyen kis értéken ezért korrigálunk a bétákkal (bias-corrected mozgó átlag és mozgó második momentum), így a kezdeti értékek a gradiensek és gradiens négyzetek lesznek (Hadamard/elemenkénti szorzatuk), így az egyenletek a következőképpen alakulnak:

|  |  |  |
| --- | --- | --- |
|  |  | (3.x) |
|  |  | (3.x) |
|  |  | (3.x) |

A probléma az Adammal, hogy kezdetben nagy a variancia, melyet jó lenne csökkenteni. Erre az egyik módszer a warmup (AdamW), azaz, hogy a tanulási ráta nem egy konstans, vagy csökkenő érték (decay), hanem egy bizonyos T ideig kezdetben növeljük az alfát, ezzel csökkentve a varianciát. A rectified ezzel szemben úgy oldja ezt meg, hogy először kiszámljuk az egyszeri mozgó átlag közelítésének (SMA) a maximum hosszát, mely … Majd ezt felhasználva minden iterációban kiszámoljuk a pt-t és ha ez átlép egy küszöböt akkor változtatunk a tanulási rátán, azaz az alfán, mely egyébként jelen esetben egy konstans. Beszorzunk egy un. variance rectification term-mel. Egyéb esetben csak alfával szorozzuk az első momentumot.

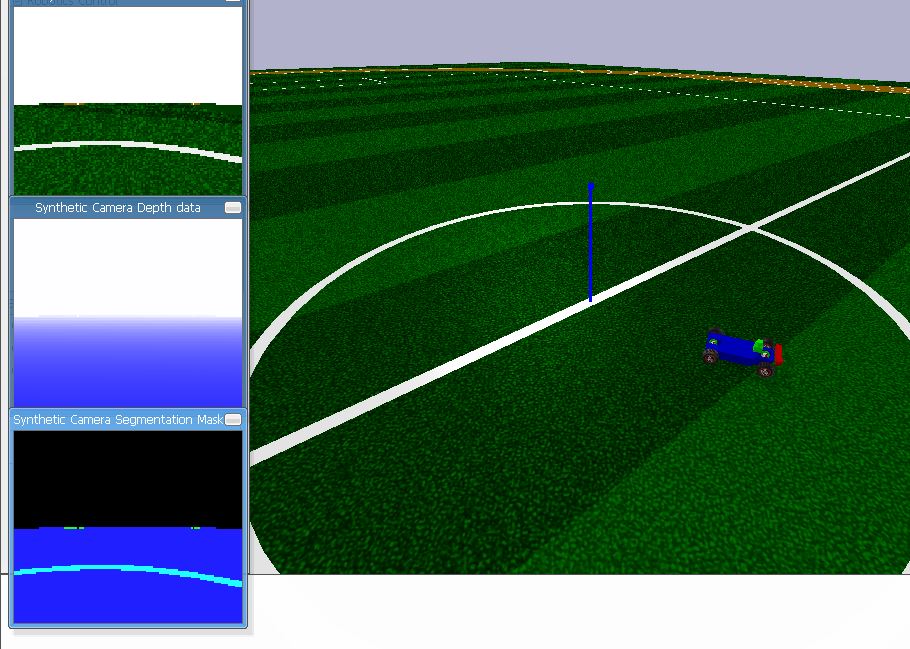
|  |  |  |
| --- | --- | --- |
|  |  | (3.x) |
|  |  | (3.x) |
|  |  | (3.x) |
|  |  | (3.x) |
|  |  | (3.x) |
|  |  | (3.x) |

## Tesztelés, eredmények

|  |  |  |  |
| --- | --- | --- | --- |
| RNN típus/Dekóder | | WER | CER |
| LSTM | Greedy | 8.021 | 3.314 |
| Beam Search | 8.150 | 3.353 |
| GRU | Greedy | 8.668 | 4.444 |
| Beam Search | 8.279 | 4.211 |

3.1 táblázat: AN4 teszthalmazának eredményei, WER/CER eredményei százalékban értendők

A kocsi egyelőre ugye inkább csak random mozgást végez, néhány esetben sikerült megközelítenie a labdát. A környezetet majd nagy mértékben át kell írni, rendes pályát kell kialakítani, vonalvezetés, kanyarok, falak, stb. Jól be kell állítani a pontozást.



3.4. ábra

Irodalomjegyzék