RAF-Deep RPG Agent

Ognjen Vinčić, Marko Živković, Teodor Petrović

# Uvod

U ovom izveštaju, detaljno opisujemo proces kreiranja inteligentnog agenta za igru RAF-Deep-RPG, koja se odvija kroz više nivoa sa različitim izazovima. Igrač ima mogućnost sakupljanja raznih resursa, kupovine zlata, i suočavanja sa varvarinima. Ključni zadatak je sakupiti dovoljno zlata i stići do glavne kapije kako bi se prešao nivo. Agent je razvijan koristeći metode učenja potkrepljivanjem, što je bio složen i zahtevan proces. U izveštaju se takođe istražuje kako su različiti aspekti igre uticali na strategiju i pristup u razvoju agenta.

# Analiza problema

Nakon detaljne analize igre, došli smo do nekoliko ključnih zaključaka. Prvo, utvrđeno je da seljak u proseku po potezu pruža najveću vrednost. To ukazuje na to da bi optimalna strategija bila prvo posetiti seljaka (ukoliko je prisutan), sakupiti neophodnu vrednost, prodati je prodavcu sve dok se ne prikupi dovoljno, a zatim nastaviti ka kapiji. Drugo, varvarin bira predmet iz inventara nasumično, a potom uniformno odlučuje o količini predmeta za uzimanje. To znači da ako imam više predmeta sa količinom 1 i samo jedan predmet sa većom količinom, šanse da uzme predmet koji najviše posedujemo su veoma male. Takođe, varvarin nikada ne uzima manje od jednog predmeta i ne uzima zlato, što sugeriše da, ako već imamo dovoljno zlata za prolaz, varvarin ne predstavlja pretnju. Treće, otkrili smo da sve mape zahtevaju 50 zlata za prolazak, iako to nije jasno naznačeno osim u datotekama mapa. Ovo može dovesti do preteranog usredsređivanja na trenirane mape. Četvrto, igrač ne mora posebno paziti na varvarina. Varvarin napada samo ako je igrač unutar njegovog radijusa od tri polja. Čak i tada, ako se igrač kontinuirano kreće, varvarin može u najboljem slučaju ukrasti samo jedan predmet. Stoga, varvarini ne igraju značajnu ulogu u igri i igrači ne bi trebali previše obratiti pažnju na njih. Uzevši u obzir prethodnu analizu, struktura nagrada za agenta u igri trebala bi biti sledeća: za sakupljanje resursa - dodeliti malu pozitivnu nagradu; za prodaju resursa za zlato - srednja pozitivna nagrada; za završetak igre - velika pozitivna nagrada. Ukoliko tokom poteza nema promene u inventaru, treba dodeliti neutralnu ili malu negativnu nagradu. Smanjenje vrednosti inventara treba rezultirati malom do srednjom negativnom nagradom.

# Dizajn agenta

Inicijalno smo primenili osnovnu metodu Q-learninga koristeći tabelu za memorisanje svakog stanja i akcije. Kao što se očekivalo, ovo nije dalo zadovoljavajuće rezultate zbog ogromnog broja stanja koja je trebalo zapamtiti. Nakon toga, eksperimentisali smo sa različitim modelima, uključujući XGBoost, obične neuronske mreže i konvolucijske neuronske mreže (CNN). Jedan od pristupa bio je korišćenje CNN-a gde se mapi daje ulaz sa kodiranim vrednostima za svako polje. Iako ova metoda nije pružila dobre rezultate, inspirisala nas je za sledeći korak. Odlučili smo da koristimo CNN sa stvarnim slikama. Koristeći postojeći frontend, transformisali smo mapu igre u sliku koju smo potom koristili kao ulaz za Proximal Policy Optimization (PPO) iz 'Stable Baselines', koji koristi CNN model. Ovaj pristup je otvorio nove mogućnosti za unapređenje našeg agenta, koristeći složenije vizualne ulaze za bolje razumevanje i interakciju sa igrom.

**PPO (Proximal Policy Optimization)** je algoritam za treniranje politike u dubokom učenju. Glavna ideja PPO-a je izbegavanje velikih promena u politici tokom ažuriranja kako bi se obezbedila stabilnost i konvergencija algoritma. Da bi to postigli, PPO koristi klipovanje prednosti i određivanje odnosa koraka.

**Klipovanje Prednosti**: PPO klipuje prednosti (razliku između procenjenih vrednosti akcija i predviđenih vrednosti) kako bi ograničio veličinu ažuriranja politike. Ovaj pristup pomaže u sprečavanju prevelikih promena koje mogu dovesti do nestabilnosti.

**Odnos Koraka:** PPO uvodi odnos koraka koji se koristi kao funkcija gubitka za treniranje politike. Ovaj odnos koraka sadrži klauzulu koja ograničava promene u politici, čime se postiže stabilnost treniranja.

**Višestruki Koraci:** PPO koristi višestruke korake (n-step) kako bi efikasnije koristio podatke iz okruženja. Umesto ažuriranja politike nakon svakog koraka, PPO prikuplja podatke za više koraka pre nego što izvrši ažuriranje.

**Vrednosna Funkcija:** Pored politike, PPO takođe trenira funkciju vrednosti koja procenjuje očekivanu nagraduodređenog stanja. Ovo pomaže u smanjenju varijacije u proceni prednosti.

**Okruženje (Environment):**

Pre nego što definišemo agenta, prvo moramo kreirati okruženje. Ideja je bila da ulazne parameter za model budu nalik slikama (frames) zbog prilagodnosti da model može imati najbolju obzervaciju. Matrica koja se vraća je pretvorena u sliku koja je dimenzije (32, 64, 3).

**Model (Policy):**

Za PPO, "MlpPolicy" se koristi za definisanje politike agenta. "Mlp" označava višeslojni perceptron, što je vrsta neuronske mreže koja se često koristi u dubokom učenju."MlpPolicy" se koristi sa sledećim parametrima:

env: Okruženje u kojem agent deluje.

Verbose = 1: Postavljanje verbose na 1 omogućava ispisivanje dodatnih informacija tokom treniranja.

tensorboard\_log = LOG\_DIR: Opciono, postavljanje direktorijuma za čuvanje podataka za TensorBoard logove.

learning\_rate = 0.000001: Stopa učenja (learning rate) za optimizaciju politike. Ovo je brzina sa kojom model ažurira svoje parametre na osnovu greške.

n\_steps = 256: Broj koraka u svakoj iteraciji treniranja. Ovo može uticati na veličinu uzorka koja se koristi za svaku iteraciju. Definiše veličinu *Batch-a.*

**Agent (Algoritam):**

PPO je algoritam za politiku optimizacije koji se koristi za treniranje agenta. Ovaj algoritam radi iterativno, prikuplja uzorke iz okruženja, ažurira politiku i funkciju vrednosti, i ponavlja ovaj proces kako bi poboljšao performanse agenta tokom vremena.

PPO klasa se koristi za definisanje PPO algoritma, a parametri kao što su stopa učenja (learning\_rate) i broj koraka (n\_steps) utiču na ponašanje algoritma.

**Treniranje:**

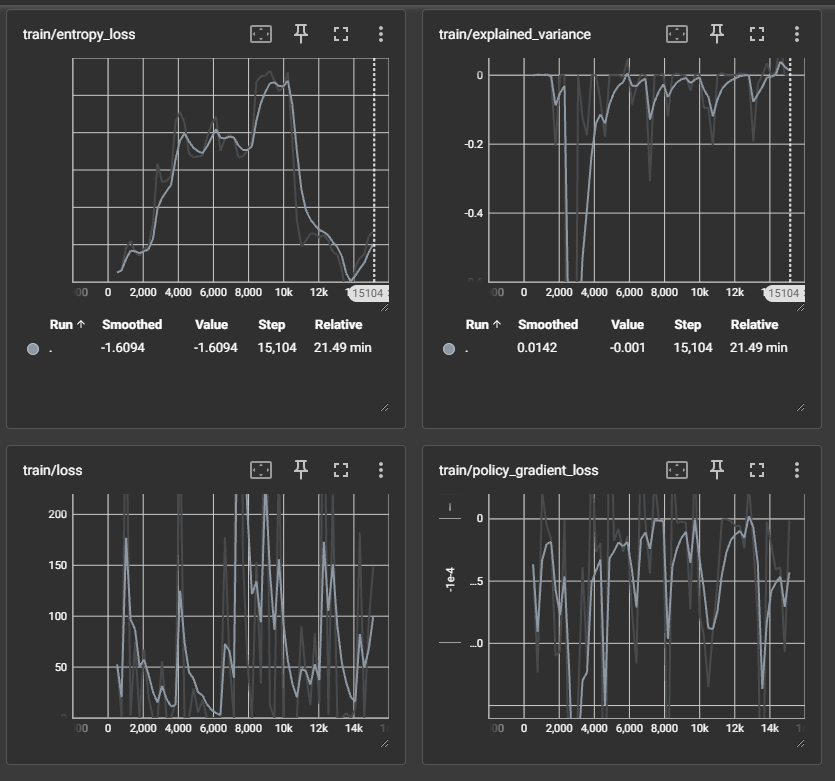
Nakon definisanja okruženja, politike i algoritma, treniranje se vrši pozivom learn metode na instanci PPO modela. Ovo pokreće iterativni proces treniranja gde model koristi prikupljene uzorke iz okruženja kako bi poboljšao svoje parametre. Takođe tokom treniranja koristimo klasu “**callback**” koji automatski čuva model agenta u memoriju. Callback može da sačuva model nakon što prođe određen broj koraka.

# Rezultati

Tokom treniranja došli smo do dosta zapažanja u ponašanju našeg agenta koje agent konstano primenjuje.

1. Agent povremeno pokušava da otvari kapiju misleći da ima dovoljno zlata.
2. Kada je bandit (lopov) u blizini agent menja ponašanje i pravi distancu. Ako se desi da bandit kruži oko kapije naš agent neće prilaziti nego će se uporno kretati okolo.
3. Agent zna da ako je pokraden i ako nema dovoljno zatvorenih polja sa materijalima, on uporno traži seljaka.
4. Nakon što sakupi neku vrednost materijala on kreće ka prodavcu i uvek pokuša I kapiju da otvori nakon dobijanja zlata.
5. Agent je uvek u pokretu, retko kad čeka.

Na narednoj slici se vide glavne informacije naše uspešnosti agenta koji je treniran kroz 15000 koraka.



Najvažniji parametri koje moramo pratiti su loss i explained\_variance.

Na grafu “train-loss” vidimo da loss baš varira u zavistnosti od broja koraka koje agent pravi. U početku je loss visok što je i očekivano sve dok ne napravi 4000 koraka posle koga kreće da raste sa malim padom sve dok ne predje 6000 koraka. U tom momentu agent mnogo greši i ne snalazise najbolje u prostoru. To zavisi od situacije i problema koji agent rešava, nije svaka epizoda idealan. U suštini loss se čas poveća čas smanji.

Graf train- explained\_variance predstavlja koliko dobro model objašnjava varijabilnost podataka koji su mu dostupni. Vidimo da ima nagli pad posle 2000 koraka što govori da model loše generalizuje na dostupnim podacima. Nakon samo 2000 koraka agent kreće bolje da generalizuje podatke i kreće da raste. Posle 14000 koraka agentova vrednost explained\_variance postaje pozitivan što daje znak da dužim treniranjem možemo da imamo bolju uspešnost.

Agent u početku gleda sa sam uči preko nagrada i same situacije mogu da bude lake i teške. Agent kreće da se stabilizuje tek nakon 10000 koraka nakon čega je samo bitno duže trenirati.

# Zaključak

Dizajniranje i treniranje modela za igru pokazalo se kao izuzetno složen zadatak. Proces izrade specifičnog modela i razvoj sistema nagrada zahtevali su mnogo eksperimentisanja i prilagođavanja. Pored toga, bilo je potrebno kontinuirano balansiranje između različitih elemenata igre kako bi se osigurao optimalan performans. Na kraju, iako je model pokazao značajan napredak, jasno je da je neophodno dodatno treniranje i fina kalibracija kako bi se postigla maksimalna efikasnost. Ovo iskustvo ukazuje na kompleksnost razvoja veštačke inteligencije u igrama koristeći metode učenja potkrepljivanjem, gde svaki element mora biti pažljivo usklađen kako bi se postigao harmoničan i efikasan sistem.