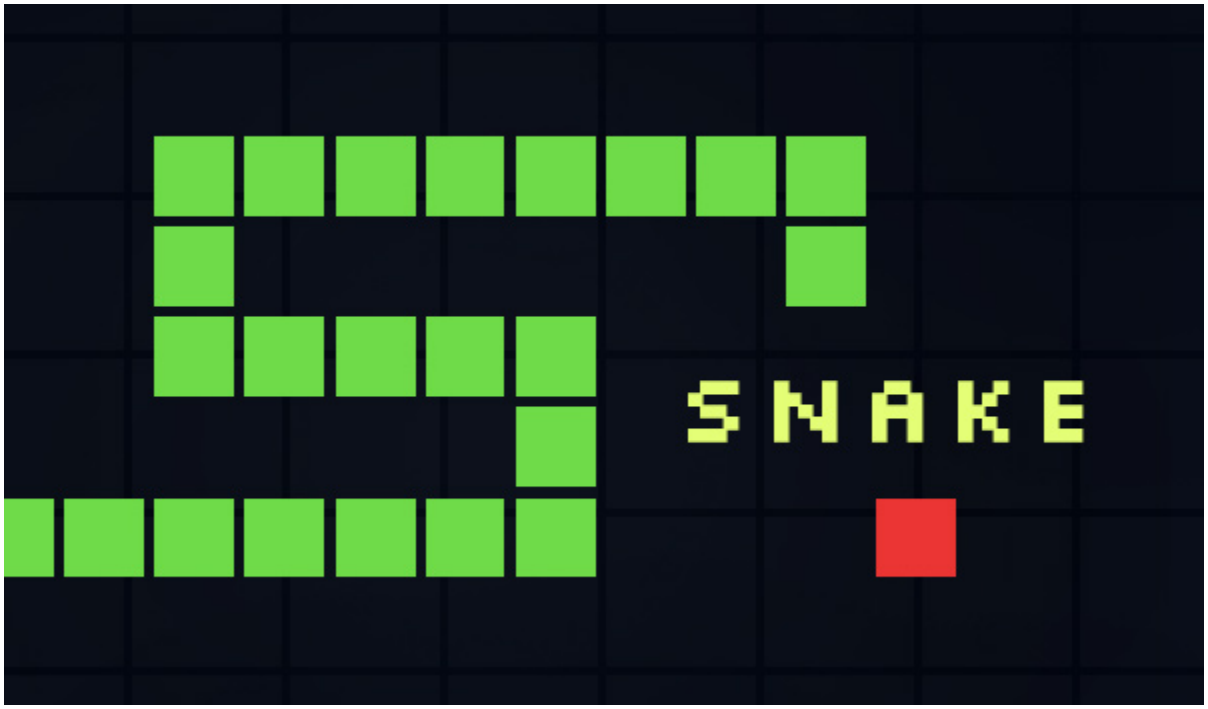

مقدمه:

یادگیری تقویتی یکی از شاخه‌های مهم یادگیری ماشین است که در آن یک عامل چگونگی انجام دادن کارها در یک محیط را با آزمایش و خطا یاد می‌گیرد. در این رویکرد، عامل با دریافت پاداش‌ها و تنبیه‌ها از محیط، رفتار خود را برای رسیدن به هدف نهایی تنظیم می‌کند. Q-Learning، یک تکنیک محبوب در یادگیری تقویتی است. این روش به عامل اجازه می‌دهد تا به طور موثرتری بهینه‌سازی کند و تصمیمات بهتری بگیرد. اکنون، با پیشرفت‌های اخیر در عمیق‌تر کردن این تکنیک‌ها، Deep Q-Learning وارد میدان شده است که یکپارچگی Q-Learning با شبکه‌های عصبی عمیق را ارائه می‌دهد و امکان پردازش موثرتر و یادگیری از داده‌های پیچیده‌تر را فراهم می‌آورد.

برای دسترسی به فایل‌های پروژه به این لینک بروید.

شبکه‌های عصبی عمیق:

شبکه‌های عصبی عمیق، که از اساسی‌ترین مؤلفه‌های یادگیری عمیق به شمار می‌روند، امروزه در بسیاری از کاربردهای پیچیده یادگیری ماشین نقش مهمی ایفا می‌کنند. این شبکه‌ها الهام گرفته از ساختار و کارکرد مغز انسان هستند و توانایی یادگیری ویژگی‌ها و الگوها از داده‌های بزرگ و پیچیده را دارند. با استفاده از چندین لایه پردازشی، شبکه‌های عصبی عمیق می‌توانند سطوح بالایی از انتزاع و نمایش داده‌ها را فراهم آورند، که این امر به آن‌ها اجازه می‌دهد تا الگوهای پیچیده‌تری را در داده‌ها شناسایی کنند. این تکنولوژی در بسیاری از زمینه‌ها از جمله تشخیص گفتار، ترجمه زبان‌ها، تشخیص تصویر و خودروهای خودران به کار رفته است و انقلابی در زمینه‌ی یادگیری ماشین به وجود آورده است. شبکه‌های عصبی عمیق، با توانایی خود در پردازش و یادگیری از داده‌های بزرگ و پیچیده، نقش کلیدی در پیشرفت‌های اخیر در هوش مصنوعی ایفا می‌کنند و زمینه را برای نوآوری‌های بیشتر فراهم می‌آورند.

در اینجا چند منبع برای درک بهتر شبکه‌های عصبی قرار دادیم.

3Blue1Brown

ویدئوهای شبکه عصبی سروش مهربان

چرا یادگیری تقویتی عمیق:

- مقیاس‌پذیری: در یادگیری تقویتی سنتی، الگوریتم‌ها معمولاً برای محیط‌های نسبتاً ساده و با تعداد محدودی از حالت‌ها و اقدامات طراحی شده بودند. اما وقتی با محیط‌های پیچیده‌تر و با تعداد زیادی از حالت‌ها و اقدامات مواجه می‌شویم، این الگوریتم‌ها کارآمدی خود را از دست می‌دهند. Deep Learning به RL کمک می‌کند تا در محیط‌های بزرگ‌تر و پیچیده‌تر به شکل کارآمدتری عمل کند.
- تعمیم‌پذیری: یکی از محدودیت‌های اصلی RL سنتی این است که معمولاً به سختی می‌تواند آموخته‌های خود را به موقعیت‌های جدید تعمیم دهد. Deep RL

از قدرت شبکه‌های عصبی عمیق برای فراگیری ویژگی‌ها و الگوهایی که می‌توانند در محیط‌های متفاوت نیز کاربرد داشته باشند، استفاده می‌کند. این امر به الگوریتم‌ها امکان می‌دهد تا در موقعیت‌های جدید و ناشناخته نیز کارآمد باشند.

- قابلیت پردازش اطلاعات پیچیده: Deep RL قادر است از داده‌های خام و پیچیده، مانند تصاویر و صداها، به طور مستقیم استفاده کند، که این امر در RL سنتی بسیار دشوار یا غیرممکن است.

در این پروژه، ما به بررسی و پیاده‌سازی بازی کلاسیک 'مار' (Snake) با استفاده از تکنیک Deep Q-Learning می‌پردازیم. بازی مار، که در آن بازیکن کنترل یک مار را بر عهده دارد و هدف او این است که مار را بدون برخورد با موانع یا خودش هدایت کند تا غذا بخورد و رشد کند، یک محیط ساده اما چالش‌برانگیز برای آزمایش و ارزیابی مدل‌های یادگیری تقویتی فراهم می‌کند. با استفاده از Deep Q-Learning، ما یک مدل را آموزش می‌دهیم تا تصمیم‌های بهینه‌ای در بازی بگیرد. این رویکرد شامل ترکیب Q-Learning، که یک روش یادگیری تقویتی است، با شبکه‌های عصبی عمیق برای پردازش و یادگیری از وضعیت‌های مختلف بازی است. نتیجه، یک عامل هوش مصنوعی است که می‌تواند به طور مستقل یاد بگیرد که چگونه بازی مار را با مهارت و کارایی بالا انجام دهد، و در نهایت نشان‌دهنده‌ی قابلیت‌های پیشرفته‌ی یادگیری ماشین و یادگیری تقویتی است.

کتابخانه PyTorch:

در این پروژه، ما از PyTorch، یک کتابخانه قدرتمند و انعطاف‌پذیر برای یادگیری عمیق، استفاده کرده‌ایم. PyTorch به خاطر API ساده و کاربر پسند، انتخاب محبوبی برای توسعه‌دهندگان و محققان در زمینه‌ی هوش مصنوعی است. این کتابخانه امکانات گسترده‌ای برای ساخت و آموزش شبکه‌های عصبی عمیق فراهم می‌کند و به خصوص برای پروژه‌هایی که نیاز به سفارشی‌سازی بالا دارند، بسیار مناسب است. PyTorch

همچنین با پشتیبانی از ابزارهای محاسباتی پیشرفته و تسریع‌کننده‌های سخت‌افزاری مانند GPUها، بهبود قابل توجهی در سرعت و کارایی پردازش‌های یادگیری عمیق ارائه می‌دهد. استفاده از PyTorch در این پروژه به ما این امکان را می‌دهد تا یک مدل Deep Q-Learning قدرتمند و بهینه را برای بازی Snake توسعه دهیم، که قادر است پیچیدگی‌های بازی را درک کرده و عملکرد خود را به طور مداوم بهبود بخشد.
در این لینک می‌توانید درک بهتری از کتابخانه PyTorch پیدا کنید.

ساختار پروژه:

Game.py	منطق بازی و روند اجرای آن در این فایل قرار دارد.
model.py	در این فایل اطلاعات مربوط به مدل Torch قرار دارد.
replay_buffer.py	در این فایل اطلاعات مربوط به ذخیره روند بازی قرار دارد.
Replay.ipynb	با استفاده از این notebook می‌توانید با استفاده از مدل آموزش داده شده خود بازی را اجرا کنید و روند اجرای بازی را مشاهده کنید.
Snake.ipynb	در این notebook مراحل اصلی مربوط به آموزش مدل قرار دارد.

فایل‌هایی که نیاز به تغییر دارند:

شما نیاز دارید که فایل‌های **model.py** و همینطور **Snake.ipynb** را تغییر دهید.
تمام بخش‌هایی که نیاز به تغییر دارند با **TODO** مشخص شده‌اند.
نکته: نیازی به تغییر در فایل **Game.py** نیست ولی تسلط به این بخش از پروژه ضروری است و در تحویل از آن سوال پرسیده می‌شود.

در این پروژه شما باید تمام بخش‌های ناقص فایل‌های مورد نظر را تکمیل کنید و در انتها با اجرای کامل فایل **Snake.ipynb** مدل مورد نظر را ایجاد کنید.
توجه کنید که این مدل در پوشه **model** ذخیره می‌شود. در صورتیکه که این پوشه در root پروژه شما وجود ندارد خودتان آن را ایجاد کنید.
در فایل **Snake.ipynb** بعد از مراحل مربوط به آموزش مدل دو سلول برای بررسی عملکرد مدل قرار داده شده است. در این بخش از کتابخانه **Matplotlib** استفاده شده است.

مواردی که باید آپلود کنید:

- تکمیل شده دو فایل **Snake.ipynb** و **model.py**
- محتویات پوشه **model**
- دو تصویر ایجاد شده توسط **Matplotlib**
- فایل گزارش موارد خواسته شده

مواردی که باید در گزارش خود بیاورید:

گزارش ۱: در هنگام تعریف مدل از دو نوع لایه در شبکه عصبی استفاده کردیم **Linear** و همینطور **ReLU**. درباره هر کدام از این لایه‌ها تحقیق کنید و توضیح دهید که چرا باید خروجی هر لایه Linear به یک لایه ReLU داده شود. همچنین نقش هر کدام از این لایه‌ها را بیان کنید.

گزارش ۲: در فرایند آموزش مدل از **Adam Optimizer** استفاده کردیم. ابتدا توضیح دهید که چرا نیاز به استفاده از Optimizer داریم و سپس درباره Adam تحقیق کنید و نحوه عملکرد آن را توضیح دهید.

گزارش ۳: در فرایند آموزش مدل از **MSE Loss** استفاده کردیم. ابتدا توضیح دهید که چرا نیاز به Loss داریم سپس درباره MSE تحقیق کنید و نحوه عملکرد آن را توضیح دهید.

گزارش ۴: در فرایند آموزش مدل عددی تصادفی تولید می‌کنیم و با مقایسه این عدد با **Epsilon** تصمیم می‌گیریم که Action بعدی چه باشد. توضیح دهید که چرا به چنین فرایندی نیاز داریم و انجام این کار چه تاثیری روی آموزش ما خواهد داشت. همچین بیان کنید که با کم یا زیاد کردن Epsilon چه اتفاقی خواهد افتاد.

گزارش ۵: در فرایند آموزش مدل توضیح دهید که **Q-Targets** چه مفهومی دارد. سپس روش محاسبه آن را بیان کنید و شرح دهید که چرا از این فرمول استفاده می‌کنیم.

تدریسیاران: با ارتباط