Controle de Um Veículo Autonomo em Ambiente Simulado CarRacing.v0

Alison de Oliveira Tristão
Otacilio Ribeiro da Silva Netto

18 de outubro de 2024

1 Introdução

Este trabalho tem como objetivo desenvolver uma rede neural para controlar um veículo em um ambiente simulado, utilizando dados gerados por um controlador automatizado. A proposta é aplicar técnicas de aprendizado de máquina para que o sistema aprenda a tomar decisões de direção de forma autônoma. O treinamento será realizado com imagens capturadas do simulador e entradas de teclado, ajustando os parâmetros de um modelo de rede neural convolucional. A avaliação do desempenho focará na evolução da acurácia durante as fases de treinamento e validação, com o intuito de verificar a capacidade do modelo de generalizar o controle do veículo em diversas pistas.

2 Formulação do Problema

Nosso objetivo é desenvolver uma rede neural capaz de controlar o carro no ambiente **CarRacing-v0**, da plataforma OpenAI Gym. Este ambiente foi projetado para testar algoritmos de inteligência artificial, particularmente em cenários que envolvem controle contínuo e visão computacional.

O jogo oferece uma visão 2D do circuito, a partir de uma perspectiva aérea, onde o agente (carro) deve percorrer a pista o mais rápido possível, evitando sair dos limites. O carro conta com três controles principais: **direção**, **aceleração** e **freio**.

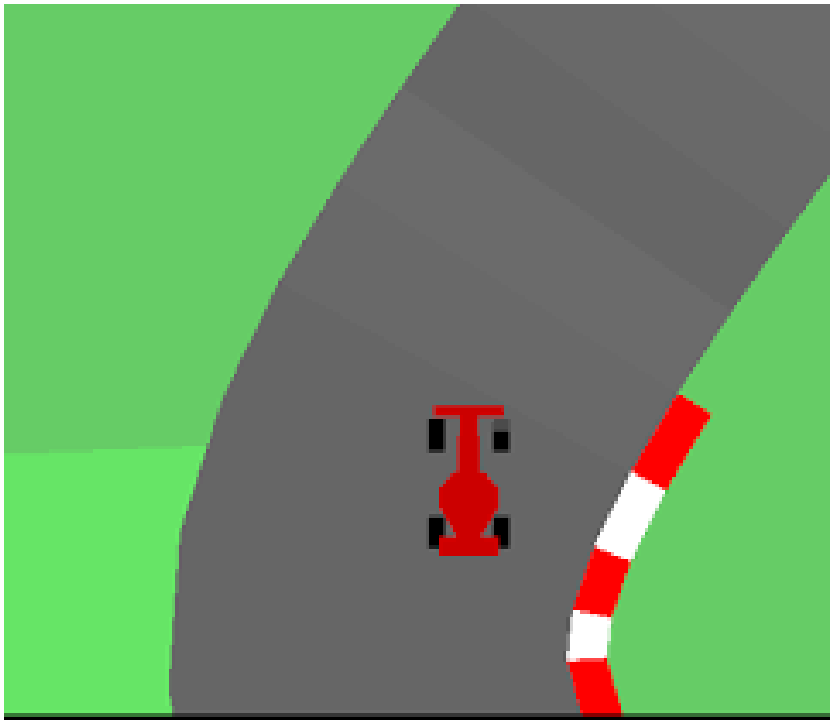

Figura 1: CarRacing-V0

O ambiente disponibiliza uma imagem RGB com resolução de **96x96 pixels**, representando a visão do carro sobre a pista, e o agente precisa interpretar essas imagens para ajustar sua direção corretamente. A pontuação no jogo é baseada na performance do agente, com recompensas positivas atribuídas conforme o carro avança pela pista sem sair dos limites.

3 Metodologia

Nossa abordagem consiste em utilizar uma rede neural convolucional (CNN) que, após ser projetada e treinada, terá como entrada os frames capturados do ambiente, junto com a velocidade do veiculo no memento e como saída os valores discretos correspondentes ao controle. A rede será responsável por controlar o carro de forma contínua, mantendo-o na pista.

Para isso, precisamos desenvolver os seguintes passos:

- Implementar um controlador automático para dirigir o carro, que servirá como "especialista". Esse especialista fornecerá as trajetórias desejadas que a rede neural tentará replicar.
- Desenvolver funções para capturar e salvar os frames e a velocidade gerados pelo ambiente, associando cada imagem ao respectivo valor de controle aplicado naquele momento.
- Desenvolver a arquitetura da rede neural e treiná-la com os dados coletados
- usar a rede treinada para prever os comandos em tempo real e controlar o carro

4 Controle

Para coletar dados de treinamento e servir como "especialista"que a rede neural vai imitar, implementamos um **controlador automático** composto por duas partes principais: o reconhecimento da pista e o controle do carro.

Reconhecimento da Pista (Classe `RecognizesRoad`): Esta classe tem como objetivo identificar a posição do carro na pista com base nas imagens fornecidas pelo simulador. O processo é dividido em três partes principais:

- **Conversão da imagem para binário**: Transformamos a imagem RGB do simulador em uma matriz binária, onde a pista é representada por 1 (preto) e as áreas fora dela por 0 (branco). Essa simplificação facilita a identificação da pista e a posição do carro em relação a ela.

- **Recorte da imagem**: Para focar na área mais importante, seleciona-se uma parte específica da imagem, uma linha de pixels, que mostra a pista um pouco à frente do carro.

- **Cálculo da posição ponderada**: A posição do carro na pista é calculada a partir de uma média ponderada das informações da imagem recortada. Cada ponto do vetor binário da pista recebe um peso de acordo com sua posição horizontal, indicando se o carro está mais à esquerda, à direita ou centralizado, permitindo assim ajustar a direção adequadamente.

Controle do Veículo (Classe `Control`): Esta classe implementa um controlador responsável por ajustar a direção do carro com base no erro calculado a partir da posição do veículo na pista. O controle pode ser realizado através de diferentes abordagens, como PI, PID ou On-Off.

Para o treinamento da rede neural, utilizamos o controle On-Off, que funciona da seguinte forma:

- **Controle On-Off**: Com base no valor da posição, que varia de -100 a +100, determinamos o comando de direção. Se a posição for maior ou igual a 35, o carro vira para a esquerda. Se for menor ou igual a -35, o carro vira para a direita. Para valores entre -34 e 34, o carro segue em linha reta.

5 Coleta de dados

Para organizar os dados de treinamento, cada frame capturado foi salvo individualmente como um arquivo de imagem com nome sequencial, por exemplo, `imagem_0.png`, `imagem_1.png`, e assim por diante. Em paralelo, criamos um arquivo `.csv` que associa cada imagem aos comandos de controle correspondentes. Cada linha do arquivo `.csv` contém o nome da imagem seguido pelos valores de controle e a velocidade atual do veiculo. Esses valores estão organizados em 3 colunas, A primeira representa a direção (1 para esquerda, 0 para reto, e 1 para direita), a segunda a aceleracao do veículo (1 para acelerar, -1 para frear e 0 sem aceleração) e a terceira a velocidade. A estrutura do arquivo `.csv` segue o seguinte formato:

```
imagem_0, 0, 1.0, 3.0
imagem_1, -1, 1.0, 4.0
imagem_2, 1, 1.0, 5.0
```

Durante a execução do controle automático para aquisição de dados, utilizamos dois métodos. Inicialmente, deixamos a seed do mapa fixa para simplificação do modelo, descartamos os primeiros 30 frames, que continham uma animação de zoom sobre o veículo, e permitimos que ele percorresse 10 voltas ao longo do percurso. No entanto, também testamos utilizando uma seed aleatória, mantendo as 10 voltas e ignorando os primeiros frames.

6 Rede Neural

Na estrutura da rede neural consiste em um *input* de 84x84x1, uma imagem em escala de cinza. Em seguida, tínhamos uma convolução com *kernel* de 8x8, 32 *feature maps* e *stride* de 4. Seguido de uma convolução com *kernel* de 4x4, *stride* de 2 e 64 *feature maps*. Em seguida, realizamos uma convolução com *kernel* de 3x3 e *stride* de 2, finalizando com uma rede *fully connected* com duas camadas com 512 neurônios, todas utilizando a função de ativação *sigmoid*. No final, utilizamos 2 neurônios com ativação *tanh*, para que a saida esteja com valores entre -1 e 1.

Com o pequeno detalhe que adicionamos a velocidade do veiculo como input junto as layer *fully connected*.

Empregamos a *mean squared error* como função de custo, e o otimizador *Adam*, com *learning rate* de 0.001 e um *batch size* de 32.

Tabela 1: Estrutura da Rede Neural

Camada	**Tipo**	**Kernel**	**Stride**	**Ativação**	**Saída**	**Feat. Map**
1	$Input_1$	84x84x1	-	-	84x84x1	1
2	Conv2D	8x8	4x4	Sigmoid	20x20x32	32
3	Conv2D	4x4	2x2	Sigmoid	9x9x64	64
4	Conv2D	3x3	2x2	Sigmoid	4x4x64	64
5	Flatten	-	-	-	1024	-
6	Concatenate	-	-	-	$1024 + Input_2$	-
7	Dense	-	-	Sigmoid	512	-
8	Dense	-	-	Sigmoid	512	-
9	Dense	-	-	Tanh	2	-

7 Resultados

Executando o treinamento com a seed fixa e aleatória, conseguimos as seguintes acurácias com o treinamento e o conjunto de validação.

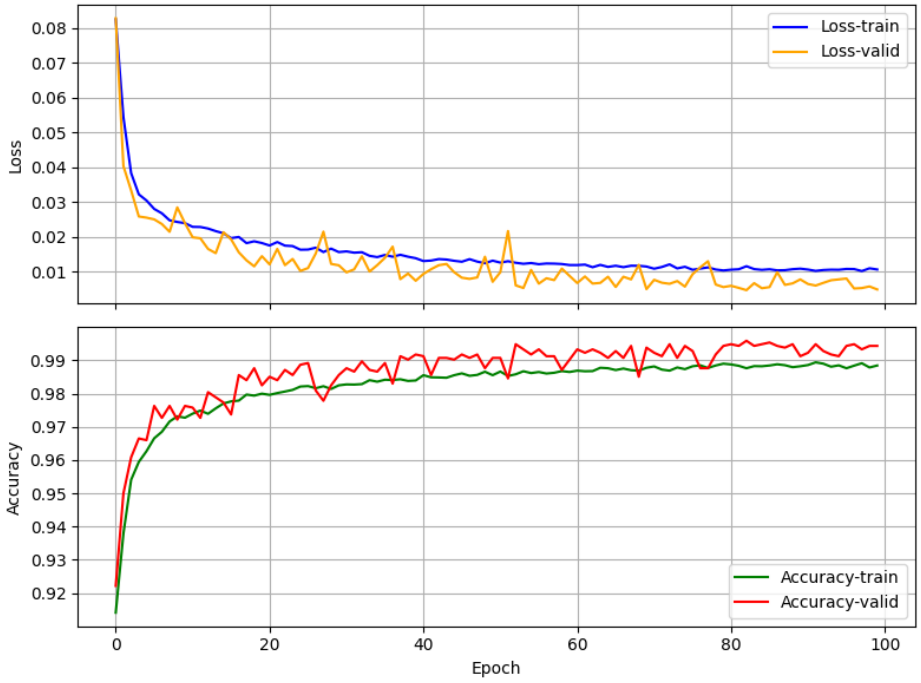

Figura 2: Acurácia com a seed fixa

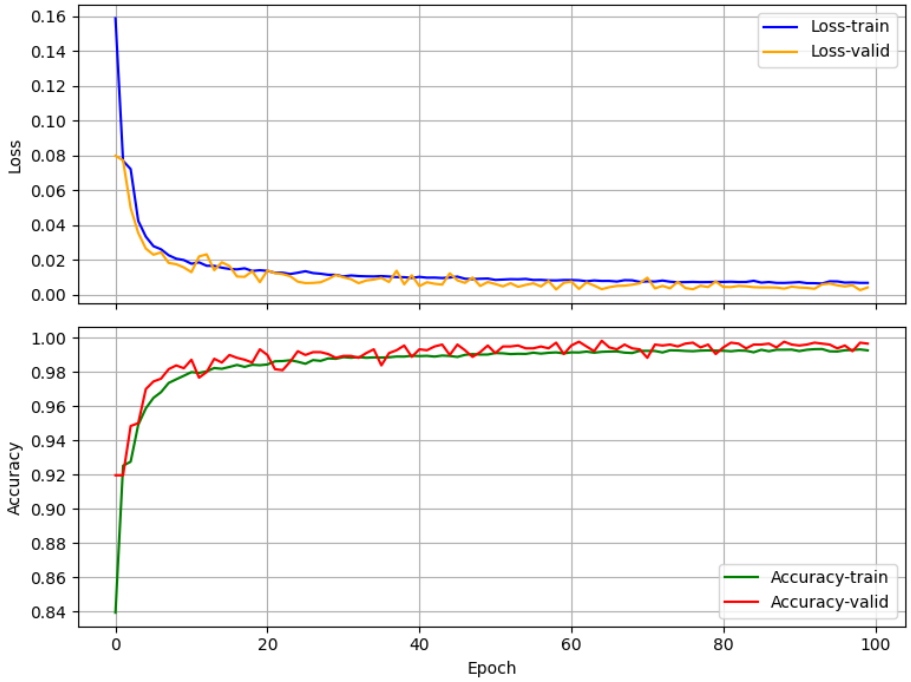

Figura 3: Acurácia com a seed aleatória

Assim, a pontuação média de ambos os modelos, controlando a simulação, é de 900, na qual o modelo treinado com a seed fixa não consegue controlar com precisão um mapa com seed aleatória, mas o treinado com a seed aleatória funciona na seed fixa do outro modelo. Batendo de frente com o controle automatizado, que faz a mesma média de pontuação, o modelo também ganha de um humano comum jogando, que pontuou cerca de 850 pontos.

8 Conclusão

Neste trabalho, analisamos o desempenho de uma rede neural convolucional para controlar um veículo em um ambiente simulado. Os resultados mostraram que o modelo treinado com uma seed fixa teve dificuldades em se adaptar a novos mapas gerados aleatoriamente, enquanto o modelo com seed aleatória foi capaz de generalizar bem. Além disso, ambos os modelos conseguiram pontuações médias de 900, superando o desempenho de um jogador humano, que fez cerca de 850 pontos. Esses resultados indicam que as redes neurais profundas podem ser eficazes no controle automático em ambientes complexos, mostrando potencial para aplicações futuras em sistemas autônomos.

Referências

KERAS. Applications. Disponível em: https://keras.io/api/applications/. Acesso em: 04 out. 2024.

OPENCV. OpenCV Documentation. Disponível em: https://opencv.org/. Acesso em: 04 out. 2024.

VERGOTTEN. Optimizers: A comprehensive study of optimizers in deep learning. Medium, 07 nov. 2019. Disponível em: https://medium.com/@vergotten/optimizers-a-comprehensive-study-of-optimizers-in-deep-learning-80a54490181f. Acesso em: 04 out. 2024.