Relatório: Trabalho Prático - Mineração de Dados
Agente Autônomo para a direção de automóveis em jogos 3d

Curso: Bacharelado em Ciências de Computação

Disciplina: SCC0230 – Inteligência Artificial

Entrega: 13/11/2018

Professora: Solange Oliveira Rezende
Estagiário PAE: Vitor Rodrigues Tonon

Integrantes:
- Antonio Moreira: 9779242
- Leonardo Meireles: 4182085
- Luca Porto: 9778943
- Vitor Brisola: 9791292

Índice Analítico

1. Introdução

Este documento apresenta uma documentação do processo de construção de um agente de direção autônoma no jogo de computador “GTA - San Andreas”.

2. Relatório

2.1. Parte 1 - Descrição da Base de Dados

A base de dados utilizada foi coletada manualmente pelos integrantes deste trabalho e consiste do conjunto de dois elementos principais um “*frame”* e uma entrada do teclado.

A coleta dos dados foi realizada utilizando-se uma aplicação python denominada “*dados.py”* construida pelos integrantes do grupo. A aplicação utiliza os módulos MSS e OpenCV de python para realizar a captura de tela e o processamento das imagens, respectivamente. Com isso, o procedimento de coleta dos dados se torna bem simples, pois basta iniciar a execução da aplicação descrita e muda o foco do computador para o jogo. Por fim, ao pressionar uma das teclas *“W”, “A” ou “D”* do teclado uma imagem do jogo é salva.

O conjunto de dados resultante desse processo de coleta é uma associação do *frame - ou captura de tela -* e a entrada de teclado de resposta a esse *frame.* O *frame* consiste de uma imagem em *grayscale* redimensionada que possui muita informação a respeito da localização do agente (personagem principal) no mundo virtual do jogo “*GTA - San Andreas”*. Assim, essas imagens possuem muito potencial para a mineração de dados, pois apresentam uma descrição do ambiente a volta do personagem - como pode-se observar na figura 1 - e com base no ambiente o personagem deve tomar decisão de qual direção seguir.

Figura 1 - *Apresenta um frame do jogo GTA -San Andreas e a entrada de teclado pressionada no momento da captura do frame.*

Além disso, o conjunto apresenta o componente de entrada do teclado que representa a direção que o agente tomou em relação ao *frame* atual, sendo que as únicas entradas possíveis são *"W*

Com isso, sabendo que o objetivo é que o agente consiga dirigir o veículo na estrada por conta própria, podemos dizer que as informações importantes de ser extraídas de tal dado são variadas e complexas. Porém, podemos observar que é interessante para o sistema extrair padrões da imagem que permitam o identificar que o personagem continua na pista e não está muito próximo à outro objeto.

2.2. Parte 2 - Descrição da Proposta de Mineração

A base de dados descrita na seção anterior sofreu vários processamentos necessários para extração do conhecimento, sendo necessário realizar um pré-processamento a fim de preparar os dados.

Primeiramente foi realizado a conversão de RGB para "grayscale" e redimensionamento da imagem, reduzindo consideravelmente o tamanho dos arquivos de dado, tornando-os viáveis. Outro procedimento realizado foi o balanceamento dos dados, já que a ação de ir para frente era predominante. Depois do balanceamento, a distribuição das direções ficou igual.

Após realizar o pré-processamento descrito foi utilizado uma técnica de Rede Neural Convolucional para a mineração dos dados. Essa técnica foi utilizado pois apresenta uma maneira robusta de extração de dados de imagens.

2.2.1. Convolutional Neural Network (CNN) e AlexNet

Redes Neurais Convolucionais, ou CNN (*Convolutional Neural Network*), são comumente usadas para extrair informações espaciais de um conjunto de dados (*dataset*), neste caso, após o pré-processamento, o conjunto de dados utilizado, que são imagens, pôde ser visto e manipulado como uma matriz. O objetivo deste tipo de rede é, através dos dados fornecidos, extrair características (*features*) comuns aos dados fornecidos e classificá-los de algum modo, assim ao inserir uma nova imagem a rede é capaz de identificar a classe que ela pertence. As redes neurais utilizam o conceito de *neurônio*, que pode estar ativo ou não, dependendo da função de ativação presente. Um neurônio tem uma característica que foi extraída através do aprendizado, portanto, dada uma imagem de entrada, esta característica será ativa, ou não, de acordo com a função de ativação implementada.

A camada de entrada, que recebe o conjunto de imagens, denomina-se *input layer*, esta rede, normalmente, é do tamanho da quantidade de pixels da imagem utilizada. As camadas de aprendizado, que determinarão o funcionamento e classificação dos dados são chamadas de *hidden layers*, o ajuste perfeito neste estágio é o que determinará um bom aprendizado. É importante salientar que os conceitos envolvidos nesta camada não serão explicados neste relatório, para não ficar muito denso, apenas vale ressaltar que existem diversas arquiteturas criadas e testadas atualmente que demonstraram ótimos resultados. Por fim, a camada que determinará a classificação probabilística das imagens é chamada de *output layer.*

Inicialmente, todos os valores dos filtros das camadas convolucionais e os pesos das camadas totalmente conectadas são inicializados de forma aleatória. Em seguida, esses valores são ajustados de forma a otimizar a acurácia da classificação

quando considerando a base de imagens utilizada no processo de treinamento. Este ajuste é realizado atraveś da técnica denominada *Backpropagation* e é assim que a rede *"aprende"*.

Como dito anteriormente, existem diversas arquiteturas de redes convolucionais, neste trabalho optou-se pela mais simples dentre elas, que é a *AlexNet*, criada em 2012 para a *ImageNet*, um competição de visão computacional. Foi utilizado a API desenvolvida pelo *Google*, o *TensorFlow*, que é específica para este tipo de problema, contudo, somente ela é muito complicada de utilizar. Para solucionar este problema utilizamos uma API desenvolvida em cima do TensorFlow, o *TFLearn*, destinada ao público mais inexperiente na área, que, no banco de dados do *GitHub*, contém uma arquitetura pré-implementada de *CNN* utilizando *AlexNet*, com todos os hiperparâmetros pré estabelecidos. A única modificação que fizemos na arquitetura original foi adicionar mais camadas de convoluções associadas a *pooling* e mais *fully-connected layers* no final, essa modificação demonstrou aumentar as taxas de acurácia do modelo.

2.3. Parte 3 - Descrição do Processamento e Discussão de Resultados

O processamento e Mineração dos foi realizado, como explicado anteriormente, utilizando-se em modelo de Rede Neural Convolucional (CNN), mais especificamente a arquitetura AlexNet. Com isso, o processamento da base de dados se tornou muito custoso e não foi possível processar o modelo em nossas próprias máquinas. Assim, foi necessário o instanciamento de uma *cloud do google* para o processamento.

A instanciação retorna um arquivo de resultado do treinamento, o qual ao receber uma imagem retorna a probabilidade de uma direção que deve ser seguida, ou seja, uma tecla a ser pressionada. Com isso, conseguimos aplicar o resultado de cada treinamento na prática e observar o resultado real além das taxas de acerto no treinamento.

A aplicação prática dos resultados de treinamento foi realizada construindo-se uma aplicação em *python* que, basicamente, faz o inverso da aplicação de coleta de dados. Esse código novamente captura a imagem atual da tela - no caso, uma imagem do ambiente virtual do jogo - processa tal imagem pelo resultado do treinamento e recebe uma porcentagem de "certeza" que uma das direções devem ser tomadas. A resposta com maior probabilidade de acerto era então aplicada ao teclado e consequentemente o personagem navegava pelo jogo. Porém, os autores perceberam que o problema requisitava alguns processamentos que iam além do que os dados permitem reconhecer.

O pós-processamento foi então realizado com base nos requisitos mencionados anteriormente. Assim, alguma lógicas foram adicionadas antes de se escolher a resposta da rede de treinamento como certa. Essas lógicas estão relacionados com requisitos mais específicos do problema, como a necessidade de controlar o número sequencial de vezes e a velocidade com que as teclas são pressionadas para controlar a velocidade do personagem no jogo.

O resultado atual da aplicação está bem satisfatório, pois após uma coleta dispendiosa de dados e um grande trabalho sobre o ajuste da mineração desses dados, o sistema já consegue navegar significativamente bem sobre a estrada do jogo. A navegação ocorre totalmente de forma autônoma e o agente já consegue reconhecer quando está saindo da pista e acaba voltando para a mesma quando faz desvios, como , por exemplo, ao sair da pista para a grama ele retorna o veículo para a avenida.

Com isso, a experiência adquirida pelo grupo, foi além do aprendizado técnico, pois o grupo aprendeu, principalmente, a reconhecer e explorar problemas de Inteligência Artificial e aprendeu, não simplesmente a como utilizar as ferramentas necessárias, mas a como buscar por essas ferramentas com base no problema.