Aplicação do Algoritmo NEAT para a Evolução de Redes Neurais em Simulação de Direção Autônoma 2D

Matheus Carara Marczuk[1]
Gabriel Eduardo Lima[1]
Eduardo Jaques Spinosa[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

{gelima,spinosa}@inf.ufpr.br, matheuszuk17@gmail.com

Abstract. *Autonomous vehicles are a core component of Intelligent Transportation Systems. This work explores the application of the NeuroEvolution of Augmenting Topologies (NEAT) algorithm to evolve autonomous vehicle controllers within a 2D simulation environment. The objective was to evaluate decision-making across two scenarios: one where the vehicle has access to instantaneous velocity, and another relying exclusively on distance sensors. Results indicate that the inclusion of velocity facilitates problem-solving, resulting in simpler and more efficient networks. Despite limitations, the study validates the efficacy of NEAT in discovering optimized topologies, demonstrating the algorithm's adaptability without the need for pre-defined architectures.*

Resumo. *Veículos autônomos são um aspecto central em Sistemas Inteligentes de Transporte. Este trabalho explora a aplicação do algoritmo NeuroEvolution of Augmenting Topologies (NEAT) para evoluir controladores de veículos autônomos em um ambiente de simulação 2D. O objetivo foi testar a tomada de decisão em dois cenários: um onde o veículo tem acesso à sua velocidade e outro guiado apenas por sensores de distância. Os resultados indicam que a inclusão da velocidade facilita a resolução do problema, resultando em redes mais simples e eficientes. Apesar das limitações, o estudo valida a eficácia do NEAT em descobrir topologias otimizadas, demonstrando a capacidade de adaptação do algoritmo sem a necessidade de arquiteturas pré-definidas.*

1. Introdução

O desenvolvimento de veículos autônomos é um desafio central na área de Sistemas Inteligentes de Transporte (ITS). Segundo o Escritório das Nações Unidas para a Redução do Risco de Desastres (UNDRR) e o Conselho Internacional de Ciência (2025) [1], o erro humano – englobando fatores como fadiga, distrações, embriaguez e excesso de velocidade – está entre as principais causas de acidentes de trânsito. Nesse cenário, capacitar veículos para evitar colisões apresenta-se como um caminho promissor para um tráfego mais seguro, assim como mais eficiente.

Apesar da idealização remeter ao século XVI, com o veículo autopropelido de Leonardo da Vinci [2], os veículos autônomos tornaram-se viáveis apenas na era moderna. Essa transição tecnológica foi consolidada por marcos importantes na pesquisa acadêmica

e industrial [3–5]. Hoje, a condução autônoma de veículos é uma realidade técnica, restando superar as dificuldades regulatórias e práticas para sua adoção em larga escala [6].

A viabilidade desta tecnologia se dá não apenas pela evolução do hardware, mas também pelo aprimoramento dos sistemas em interpretar o ambiente e tomar decisões complexas. Diferente de ambientes controlados, a direção autônoma exige agentes capazes de adaptação contínua. Foi nesse contexto que o avanço da Inteligência Artificial, especificamente por meio do aprendizado profundo, permitiu que veículos autônomos se tornassem uma realidade [7, 8].

Entretanto, a aplicação de métodos tradicionais de aprendizado profundo impõe alguns desafios técnicos. Essas abordagens demandam alto custo computacional e dependem de grandes volumes de dados rotulados para treinamento, o que nem sempre é acessível. Além disso, existe a complexidade intrínseca ao design dos modelos; a necessidade de projetar arquiteturas manualmente requer expertise técnica e pode resultar em modelos sub-otimizados [9].

Diante dessas limitações, a Computação Evolutiva e a Neuroevolução surgem como alternativas. Diferente do treinamento convencional, a Neuroevolução baseia-se em princípios evolutivos (seleção, mutação e cruzamento) para automatizar a criação de redes neurais [10]. Essa abordagem permite não apenas ajustar os pesos da rede, mas também descobrir a topologia adequada para o problema, evoluindo arquiteturas que são frequentemente mais eficientes [11].

Nesse contexto, o presente trabalho propõe a aplicação do algoritmo *NeuroEvolution of Augmenting Topologies* (NEAT) [12] para evoluir redes neurais voltadas ao controle de veículos autônomos. Os objetivos do estudo concentram-se em três aspectos: validar a capacidade de navegação do agente em ambiente simulado, investigar os padrões comportamentais emergentes e analisar a evolução topológica das redes neurais.

Dada a complexidade do problema, o ambiente de teste foi simplificado para uma simulação 2D. O estudo analisa dois cenários distintos: (1) com velocidade informada, onde a rede recebe dados de distância e da velocidade instantânea; e (2) sem velocidade informada, onde o dado de velocidade é omitido. Essa abordagem visa analisar o desempenho e a capacidade de adaptação do algoritmo frente a problemas similares, porém com complexidades distintas.

O restante deste texto está organizado da seguinte forma. A Seção 2 expõe a fundamentação teórica, abrangendo redes neurais e o algoritmo NEAT. A Seção 3 define o problema estudado e detalha o simulador de direção autônoma desenvolvido. Em seguida, a Seção 4 descreve a modelagem da solução computacional proposta. A Seção 5 reporta o protocolo experimental e discute a análise dos resultados obtidos. Por fim, a Seção 6 apresenta as conclusões e sugere direções para trabalhos futuros.

2. Fundamentação Teórica

Esta seção apresenta os conceitos necessários para a compreensão do trabalho realizado. A discussão inicia-se na Seção 2.1 com uma breve revisão sobre redes neurais artificiais e, em seguida, a Seção 2.2 detalha o algoritmo NEAT. Em ambos os tópicos, destaca-se a inspiração biológica que fundamenta essas abordagens computacionais.

2.1. Redes Neurais Artificiais

As Redes Neurais Artificiais são modelos computacionais inspirados no funcionamento do sistema nervoso biológico. O modelo artificial imita o biológico em dois aspectos: é constituído por unidades de processamento simples interconectadas e sua funcionalidade é baseada na força dessas conexões [13, 14]. A unidade fundamental é o neurônio artificial, uma abstração matemática da célula nervosa. Assim como na biologia, onde dendritos captam sinais para processamento e transmissão via axônio, o neurônio artificial recebe entradas numéricas, processa-as e produz uma saída.

O modelo do neurônio artificial, apresentado na Figura 1, opera realizando uma soma ponderada de seus sinais de entrada x_i pelos pesos sinápticos w_{ki}, acrescida de um viés b_k. O resultado, denominado potencial de ativação (v_k), é submetido a uma função de ativação $\phi(\cdot)$ para gerar a saída y_k.

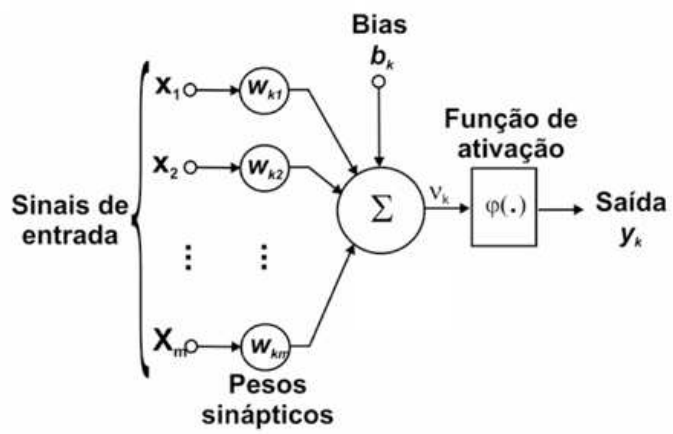


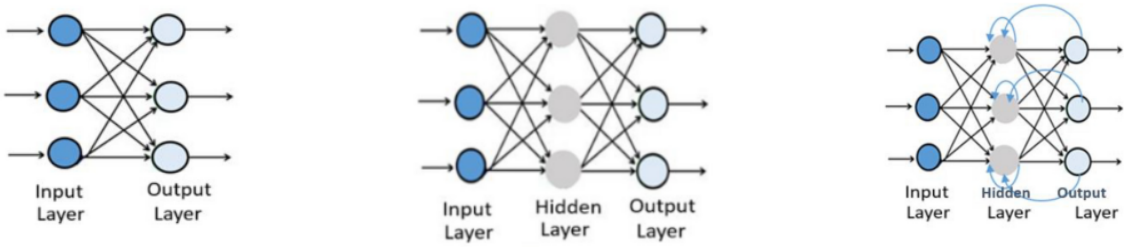
Figura 1. Modelo de Neurônio Artificial. Adaptado de Haykin (2008) [13].

A equação geral é definida por:

$$y_k = \phi(v_k) = \phi\left(\sum_{i=1}^{m} w_{ki} \cdot x_i + b_k\right) \tag{1}$$

Entretanto, o potencial das redes neurais artificiais reside na composição desses neurônios em estruturas complexas. Segundo Haykin (2008) [13] e Hagan *et al.* (2014) [14], distinguem-se três principais arquiteturas (Figura 2), sendo elas:

- Redes *Feedforward* de Camada Única: Neurônios agrupados paralelamente recebem apenas sinais diretos da entrada. Essa organização define uma camada de neurônios.
- Redes *Feedforward* Multicamadas: Consiste na adição de camadas intermediárias para resolver problemas complexos. A arquitetura permite o mapeamento de padrões não-lineares, mantendo um fluxo de informação unidirecional (da entrada para a saída).
- Redes Recorrentes: Permitem conexões cíclicas ou retroalimentação, onde a saída de um neurônio pode influenciar sua própria entrada em instantes futuros. Usada para gerar o efeito de memória temporal.



(a) Rede *Feedforward* de Camada Única

(b) Rede *Feedforward* Multicamadas

(c) Rede Recorrente

Figura 2. Representação das principais arquiteturas de redes neurais artificiais.

Por fim, o processo de aprendizado da rede define-se como a tarefa de otimização dos pesos sinápticos. Diferente dos métodos clássicos baseados em cálculo de gradiente, a neuroevolução – abordagem deste estudo – trata o conjunto de pesos e conexões como o genoma de um indivíduo, otimizando-o por meio de algoritmos genéticos.

2.2. *NeuroEvolution of Augmenting Topologies* (NEAT)

O algoritmo *NeuroEvolution of Augmenting Topologies* (NEAT), proposto por Stanley e Miikkulainen em 2002 [12], representa um método para a evolução de redes neurais artificiais. Diferente de abordagens tradicionais de neuroevolução que otimizam apenas os pesos sinápticos de uma topologia pré-definida, o NEAT evolui simultaneamente os pesos da rede e sua arquitetura. Dessa forma, o algoritmo descobre a topologia mais adequada para uma determinada tarefa de forma autônoma.

A inspiração biológica do algoritmo está no conceito de complexificação incremental. Na natureza, organismos complexos evoluíram a partir de estruturas celulares simples. O algoritmo mimetiza esse processo, iniciando a população com redes de topologia mínima, contendo apenas os neurônios de entrada e saída. A complexidade estrutural é evoluída progressivamente através de mutações, na medida em que a operação garante alguma vantagem à solução.

A Figura 3 ilustra o mapeamento entre o genótipo e seu fenótipo, isto é, a rede neural resultante. O genoma de cada indivíduo consiste em uma lista dinâmica composta por dois tipos de genes: genes de nó, que indicam a existência dos neurônios, e genes de conexão, que descrevem as sinapses da rede. Este último gene armazena informações como o nó de origem e de destino, o peso da conexão, um bit que determina se a conexão está ativa, e um número de inovação.

O número de inovação desempenha um papel crucial no algoritmo, permitindo a identificação histórica de cada gene. Um dos desafios da evolução de topologias está na dificuldade de realizar o cruzamento entre redes com estruturas distintas sem gerar descendentes disfuncionais. O NEAT soluciona esse problema utilizando marcadores para alinhar os genes de pais diferentes, classificando-os em três categorias:

Genes Coincidentes: presentes nos genomas de ambos os pais. Genes Disjuntos: presentes em apenas um dos pais, mas situados dentro do intervalo de inovação do outro. Representam divergências evolutivas mais antigas. Genes em Excesso: presentes em apenas um dos pais e situados fora do intervalo de inovação do outro. Representam as inovações estruturais mais recentes.

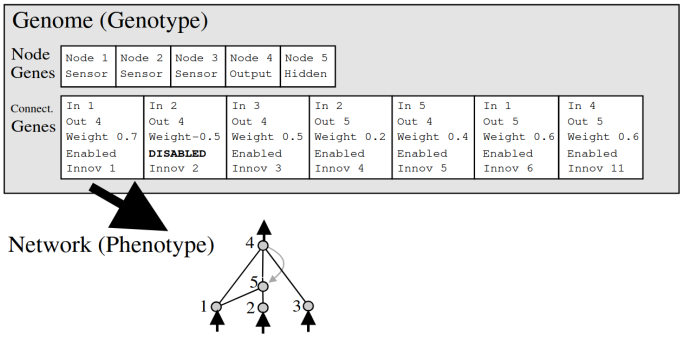


Figura 3. Representação da codificação genética do NEAT, exemplificando o mapeamento entre o genótipo (lista de genes) e o fenótipo (rede neural resultante). Adaptado de Stanley e Miikkulainen (2002) [12].

O processo de cruzamento ocorre através do alinhamento desses genes. Para os genes coincidentes, o descendente herda aleatoriamente o peso de um dos pais. Já para os genes disjuntos e em excesso, eles são copiados do indivíduo com maior aptidão. Em caso de empate, a herança ocorre de forma aleatória. A Figura 4 exemplifica o cruzamento de dois indivíduos, bem como o indivíduo resultante.

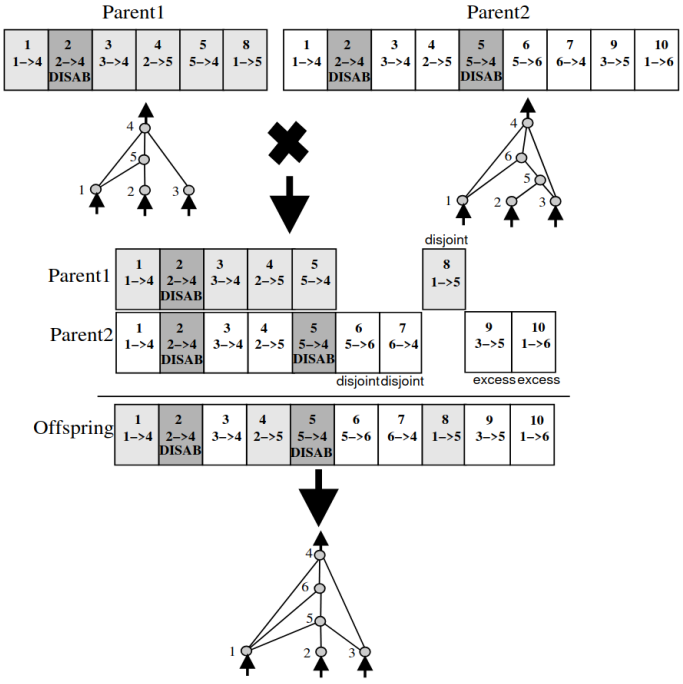


Figura 4. Ilustração do cruzamento genético do NEAT, onde a combinação das estruturas de dois pais para formar a topologia do indivíduo filho. Adaptado de Stanley e Miikkulainen (2002) [12].

Além disso, o NEAT introduz o conceito de especiação. Novas características evolutivas podem não conferir vantagem imediata. Por exemplo, uma nova topologia pode apresentar um desempenho menor enquanto seus pesos não são ajustados. Para evitar que inovações topológicas sejam eliminadas precocemente, o algoritmo divide a população em espécies. Assim, os indivíduos competem dentro de seu próprio grupo, protegendo as topologias recentes da competição com redes já otimizadas.

A definição das espécies baseia-se na distância de compatibilidade entre os genomas. Essa métrica quantifica a divergência entre dois indivíduos considerando tanto a topologia quanto os pesos, sendo computada conforme a equação:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \bar{W} \tag{2}$$

Onde E e D representam, respectivamente, a quantidade de genes em excesso e disjuntos; $\bar{W}$ é a diferença média dos pesos nas conexões coincidentes; e N é o número total de genes no maior genoma. Os coeficientes c_1, c_2 e c_3 são parâmetros ajustáveis que ponderam a importância relativa da estrutura versus pesos.

Concluindo, além do ajuste dos pesos sinápticos, o algoritmo define dois operadores de mutação da arquitetura da rede neural (Figura 5): a adição de conexão, que estabelece novas sinapses com pesos aleatórios entre nós desconectados, e a adição de nó, que introduz um neurônio intermediário dividindo uma conexão existente. Cada uma dessas mutações gera um novo número de inovação, atribuído ao gene resultante. Adicionalmente, implementações do algoritmo preveem a remoção de conexões conforme uma taxa probabilística.

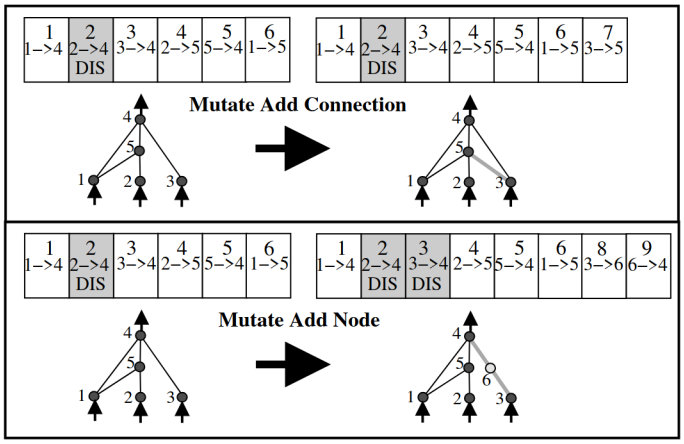


Figura 5. Exemplificação dos operadores de mutação estrutural do algoritmo NEAT. Adaptado de Stanley e Miikkulainen (2002) [12].

3. Definição do Problema: Simulador de Direção 2D

O problema central abordado neste estudo consiste no controle autônomo de um veículo em ambiente simulado. Dada a complexidade de simular o tráfego real, o escopo do problema foi delimitado a uma simulação bidimensional, onde espera-se obter um agente capaz de controlar um veículo com física simplificada, utilizando sensores de distância e velocidade, aliados a atuadores básicos.

A escolha de desenvolver um simulador próprio, em vez de utilizar implementações existentes ou benchmarks públicos[1], deve-se a requisitos de desempenho e delimitação do foco do trabalho. Dado o alto volume de simulações iterativas exigido

[1]Exemplos de simuladores populares incluem: Car Learning to Act (CARLA) [15]; AirSim [16]; The Open Racing Car Simulator (TORCS) [17].

pela computação evolutiva, um ambiente simplificado oferece menor custo computacional e maior agilidade. Essa abordagem abstrai complexidades desnecessárias, sendo suficiente para isolar e avaliar a eficácia do aprendizado da rede neural.

A apresentação do simulador segue uma estrutura em duas etapas. A Seção 3.1 descreve a modelagem do problema, incluindo a física do veículo e seus sensores. Por fim, a Seção 3.2 expõe a infraestrutura técnica, resumindo as linguagens e bibliotecas que compõem a implementação.

3.1. Ambiente de Simulação

O ambiente de simulação consiste em uma pista bidimensional renderizada a uma taxa fixa de 60 quadros por segundo. A definição da pista baseia-se em uma imagem estática, na qual a área navegável e a área intransponível são distinguidas pelas cores preta e cinza, respectivamente. A Figura 6 apresenta o traçado adotado, que foi selecionado por sua complexidade. Ao combinar retas e curvas de diferentes angulações, este circuito oferece maior dificuldade do que traçados lineares e elípticos.

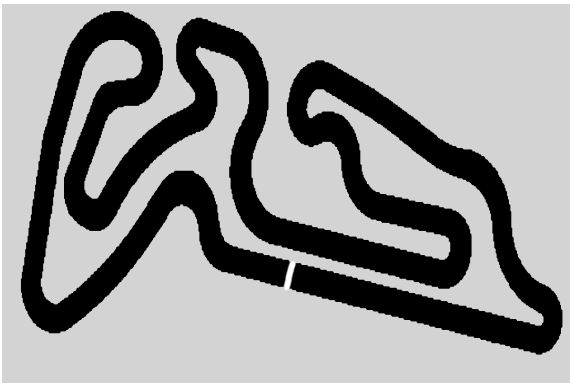


Figura 6. Traçado adotado para a simulação. A área navegável é representada em preto.

O veículo é representado por um objeto de 6×12 pixels, e seu comportamento é definido pelas variáveis de posição cartesiana (x, y), ângulo de orientação (θ), e velocidade escalar (v). A física da simulação utiliza valores pré-definidos para movimentação: aceleração de $0.06\,\text{px/frame}^2$ e frenagem ativa de $0.1\,\text{px/frame}^2$. Adicionalmente, forças resistivas desaceleram o veículo: um atrito constante de $0.02\,\text{px/frame}^2$ e uma força de arrasto calculada por um coeficiente de amortecimento de 0.01. A velocidade escalar é limitada a 5.0 e o veículo só pode realizar curvas se estiver em movimento $(v > 0.5)$.

Os controles do veículo abstraem os sistemas mecânicos em um sistema discreto de quatro atuadores binários: aceleração, frenagem, esterçamento à esquerda e esterçamento à direita. O esterçamento é definido por uma taxa fixa de variação angular de $3.0°/\,\text{frame}$. Como não há controle analógico, a navegação pela trajetória depende da capacidade do controlador em alternar esses comandos na frequência adequada.

Quanto às colisões, o sistema utiliza uma verificação simplificada no centro do veículo. Se o pixel central sobrepor uma área intransponível, a simulação é encerrada. Essa abordagem trata o veículo logicamente como um ponto, ignorando suas bordas para fins de cálculo, o que é suficiente para o escopo do problema. Além disso, não há interação com outros veículos.

Para perceber o ambiente, o agente dispõe de um sistema sensorial baseado em *Ray Casting*. A percepção é realizada pela projeção de cinco vetores de detecção a partir do centro do veículo, dispostos nos ângulos relativos de $-90°$, $-45°$, $0°$, $+45°$, e $+90°$ (Figura 7). O algoritmo verifica a cor da pista ao longo de cada raio até uma distância máxima de 100 pixels, retornando valores normalizados de proximidade com a área intransponível.

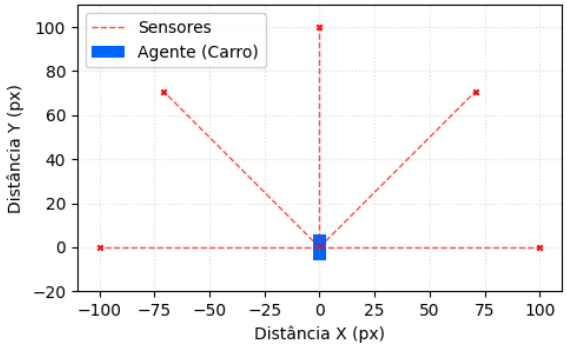


Figura 7. Representação do sistema sensorial do veículo. Em vermelho, são projetados sinais para os sensores de distância.

Embora o simulador apresente limitações – como a ausência de falhas simuladas nos sensores e um modelo físico que ignora mecânicas mais complexas — ele provê um ambiente suficientemente desafiador, adequado para a prova de conceito e análise do algoritmo NEAT.

3.2. Detalhes sobre a Implementação

O simulador foi desenvolvido usando a linguagem Python 3, escolhida por sua eficiência na prototipagem e facilidade de integração com bibliotecas de neuroevolução. A biblioteca Pygame foi utilizada para renderização gráfica e captura de eventos. A arquitetura do sistema está centrada na classe *Car*, que encapsula os atributos e métodos responsáveis pela simulação do veículo.

Adicionalmente, uma interface gráfica (ilustrada na Figura 8) foi projetada para permitir o monitoramento em tempo real do comportamento do veículo. Além da pista, o sistema exibe o veículo (retângulo azul) e a sua percepção sensorial (linhas vermelhas). Ademais, ela apresenta métricas como a velocidade escalar, o estado de ativação dos atuadores (acelerador, freio e esterçamento) e o tempo decorrido de execução.

O simulador está acessível em duas versões: a primeira (*manual-control.py*) permite que um usuário controle o veículo por meio de comandos de teclado. Este código foi utilizado para analisar a viabilidade de condução na pista, bem como para ajustar os parâmetros da simulação. Já a segunda versão (*evolve-control.py*) foi desenvolvida com base na versão anterior, integrando o algoritmo NEAT para a evolução de redes neurais para o controle do veículo.

Para obter maiores detalhes sobre a implementação, dependências e instruções para execução, recomenda-se a consulta ao repositório online do projeto[2], onde todo o material desenvolvido está hospedado.

[2]https://bit.ly/Bio-inspired-T3

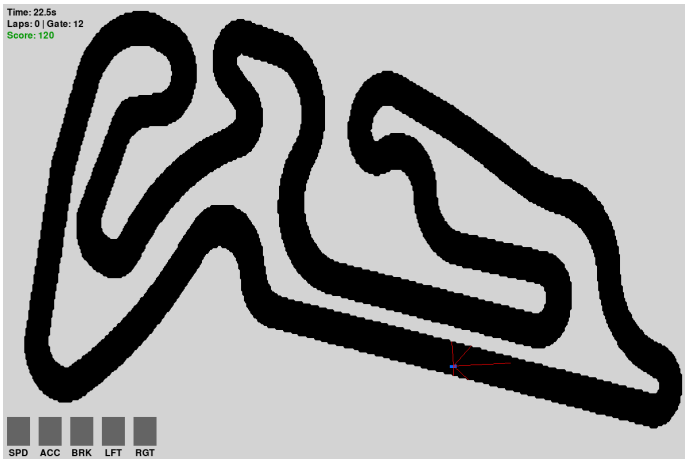


Figura 8. Ilustração da interface gráfica do simulador. Além do traçado, são exibidos o veículo, informações de seus atuadores, e informações gerais da simulação.

4. Desenvolvimento do Controlador Neuroevolutivo

Esta seção apresenta a abordagem computacional adotada para a resolução do problema de navegação autônoma. A solução baseia-se na evolução de controladores baseados em redes neurais artificiais usando NEAT para operar no ambiente simulado descrito na Seção 3.1. O texto em sequência detalha a modelagem da solução (Seção 4.1) e a descrição dos principais aspectos técnicos da sua implementação (Seção 4.2).

4.1. Modelagem da Solução

Diferente das abordagens tradicionais de aprendizado profundo, que exigem a predefinição de uma topologia fixa, o algoritmo NEAT inicia o processo evolutivo com uma topologia mínima, sem a presença de camadas ocultas. A complexidade da rede cresce incrementalmente, guiada pela necessidade de solucionar o problema. Portanto, a modelagem inicial concentra-se na definição dos vetores de entrada e saída e sua relação com o ambiente simulado.

A camada de entrada atua como o sistema perceptivo do agente. Visando uma modelagem naturalística, optou-se por restringir a percepção a sensores embarcados, similares aos disponíveis em um veículo real. Alternativas como o uso de coordenadas globais foram descartadas, pois induziriam o agente a memorizar sua posição na pista ao invés de aprender a reagir ao ambiente de simulação.

Sendo assim, o trabalho investiga duas configurações experimentais para o vetor de entrada: Cenário 1 (velocidade informada): a rede recebe seis sinais; o sinal dos cinco sensores de distância e a velocidade escalar instantânea do veículo. Cenário 2 (velocidade não informada): a rede recebe apenas os sinais dos cinco sensores de distância. A proposição desses cenários objetiva investigar a capacidade de evolução da rede neural em relação a problemas de complexidades distintas.

Todos os valores de entrada são normalizados para o intervalo $[0, 1]$. Esse pré-processamento é adotado para garantir a estabilidade numérica e evitar viés relacionado à magnitude dos sinais, impedindo que variáveis com escalas maiores (distância em pi-

xels) tenham um peso desproporcional na decisão da rede, apenas devido à sua ordem de grandeza, em detrimento de variáveis menores (velocidade).

Em relação à camada de saída, ela é composta por quatro neurônios que são mapeados diretamente para os atuadores do simulador: aceleração, frenagem e esterçamento (esquerda/direita). A função de ativação utilizada é a sigmoide logística, restringindo as saídas ao intervalo $[0, 1]$. A decisão de controle segue uma lógica baseada em um limiar de ativação: se a saída do neurônio exceder o valor de 0.5, o atuador correspondente é acionado. Dessa forma, o controlador pode ativar múltiplos atuadores simultaneamente, cabendo à evolução selecionar as combinações mais eficientes.

Além da definição topológica da rede, a aplicação do NEAT exige a formulação de uma função de aptidão (*fitness*). Esta métrica é responsável por quantificar a qualidade das soluções e guiar o processo evolutivo. Embora a literatura apresente abordagens complexas para modelar essa função [18–20], este trabalho adotou uma estratégia simples baseada em *checkpoints*.

A pista foi segmentada manualmente por 379 linhas transversais, conforme ilustrado na Figura 9. O cálculo da aptidão ocorre da seguinte forma: somam-se 10 pontos a cada transposição de *checkpoint* e 500 pontos adicionais ao completar uma volta. Ressalta-se que a pontuação depende do progresso sequencial de *checkpoints*, de modo que a condução na contramão ou em *loops* não gera incremento na aptidão.

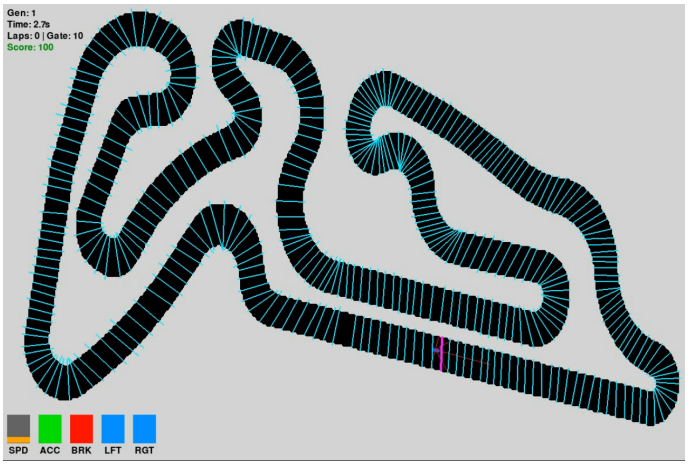


Figura 9. Exemplificação da distribuição de *checkpoints* – segmentos de retas destacados em azul claro. O *checkpoint* em roxo demarca o próximo objetivo a ser transapassado pelo veículo, guiando sua condução.

A configuração dos *checkpoints* segue duas escolhas de modelagem intencional. Primeiro, a alta densidade de distribuição visa fornecer recompensas frequentes, evitando lacunas de pontuação que prejudicariam a evolução. Segundo, a definição geométrica como segmento de retas (e não pontos) permite que o agente seja recompensado em toda a largura da pista. Isso impede que uma trajetória pré-definida seja forçada, dando liberdade para que o algoritmo descubra a condução adequada.

Existem ainda três critérios para a interrupção da simulação: a ocorrência de colisão, a estagnação (não passar por um *checkpoint* em 3 segundos) ou o esgotamento do tempo total de 1 minuto. Em todos os casos, a aptidão acumulada até o instante da

interrupção é preservada para avaliação pelo algoritmo evolutivo. Destaca-se ainda que essa abordagem estimula indiretamente a condução rápida e eficiente, já que agentes velozes conseguem passar por mais *checkpoints* dentro do tempo limite.

Por fim, é apresentada a configuração dos hiperparâmetros do algoritmo NEAT. A parametrização foi padronizada para todos os experimentos, visando garantir a consistência comparativa do estudo e devido ao escopo proposto. A arquitetura das redes restringe-se ao modelo Feedforward, proibindo conexões recorrentes para simplificar a topologia resultante. A função de ativação adotada para todos os nós é a Sigmoide, comumente utilizada nas primeiras redes neurais artificiais.

O Quadro 1 resume os hiperparâmetros adotados para o algoritmo NEAT, separados por categoria de funcionalidade. Destaca-se o tamanho da população fixado em 150 indivíduos e as taxas de mutação, calibradas para favorecer a diversidade genética e a inovação topológica. O limite de 100 gerações foi estabelecido para balancear o custo computacional e avaliar a eficácia do algoritmo em cenários de recursos limitados, investigando sua capacidade de convergência em curto prazo.

Quadro 1. Configuração dos Hiperparâmetros do Algoritmo NEAT.

Categoria	**Parâmetro**	**Valor**	**Descrição**
Geral	Tamanho da População	150	Quantidade total de agentes (genomas) avaliados simultaneamente em cada geração.
	Critério de Aptidão	Máx.	A aptidão de uma espécie é definida pelo score do seu melhor indivíduo.
	Gerações	100	Critério de parada do experimento (limite de iterações).
Ativação	Função de Ativação	Sigmoide	Função não-linear aplicada a todos os nós, mapeando a saída para o intervalo $[0, 1]$.
Mutação de Pesos	Taxa de Mutação	80%	Probabilidade de um peso de conexão existente ser alterado.
	Poder da Mutação	0.5	Desvio padrão da distribuição normal usada para ajustar o peso (intensidade da mudança).
	Taxa de Substituição	10%	Probabilidade de um peso ser totalmente substituído por um novo valor aleatório.
Mutação Estrutural	Adicionar Nó	50%	Probabilidade de criar um novo neurônio oculto dividindo uma conexão existente.
	Adicionar Conexão	40%	Probabilidade de criar uma nova sinapse entre dois nós desconectados.
	Remover Conexão	30%	Probabilidade de excluir uma conexão existente.
Especiação	*Disjoint Coeff.* (c_1)	1.0	Peso dado aos genes disjuntos no cálculo da distância genômica.
	Weight Coeff. (c_3)	0.5	Peso dado à diferença de pesos no cálculo da distância genômica.
	Limiar de Compatibilidade	3.0	Distância genética mínima para que dois indivíduos sejam considerados de espécies diferentes.
Reprodução	Elitismo	3	Número de melhores indivíduos de cada espécie preservados inalterados para a próxima geração.
	Limiar de Sobrevivência	20%	Fração da espécie permitida a se reproduzir via cruzamento.
	Estagnação Máxima	20	Limite de gerações sem melhoria na aptidão antes que uma espécie seja extinta.

4.2. Detalhes sobre a Implementação

A integração do algoritmo NEAT ao simulador foi realizada por meio da biblioteca neat-python. Além disso, optou-se pelo desacoplamento entre a evolução genética e a renderização visual, permitindo dois modos de operação. Para otimizar o treinamento, o sistema opera predominantemente em modo *headless* (sem interface gráfica). Essa abordagem elimina o custo computacional de desenho na tela, permitindo que a simulação dos 150 indivíduos ocorra na velocidade máxima de processamento.

A visualização, quando habilitada, prioriza a exibição apenas do melhor indivíduo. Após a avaliação acelerada de toda a população, o algoritmo identifica o genoma com maior aptidão e executa uma rotina de *replay*. Somente nesta etapa a interface gráfica é ativada para re-simular o comportamento do melhor indivíduo, permitindo a inspeção visual de sua performance.

Além do monitoramento, o módulo de *replay* atua como um registrador. Durante a reexecução, utiliza-se a biblioteca OpenCV para capturar e compilar os quadros em um arquivo de vídeo. Simultaneamente, a definição da rede neural (genótipo) é serializada e salva em disco, podendo ser usada para obter a representação topológica da rede via Graphviz. A persistência desses artefatos permite análises sem a necessidade de reprocessar a simulação e está disponível no repositório oficial do projeto.

5. Experimentos e Resultados

Com base na modelagem computacional estabelecida na Seção 4.1, o experimento consistiu na execução do algoritmo NEAT nos cenários de velocidade informada e velocidade não informada. Foram realizadas dez execuções independentes para cada cenário, e os resultados são apresentados por execução, permitindo tanto a comparação intra-cenários quanto inter-cenários. A discussão dos resultados é dividida em análise quantitativa (Seção 5.1) e análise qualitativa (Seção 5.2).

5.1. Análise Quantitativa

A análise quantitativa fundamenta-se em dados coletados nos recortes geracionais 1, 10, 25, 50, 75 e 100, avaliando a eficácia e a complexidade do melhor indivíduo. A eficácia avalia a capacidade de navegação e considera métricas como a evolução da aptidão, a geração de convergência (momento da primeira volta completa) e o total de voltas acumuladas ao final do experimento. Já a complexidade investiga a evolução estrutural da rede neural, contabilizando o número de neurônios e conexões sinápticas.

A Tabela 2 apresenta os resultados de eficácia de cada execução experimental, evidenciando uma disparidade de desempenho entre os cenários. No cenário sem informação de velocidade (velocidade não informada), os agentes demonstraram dificuldade para completar o percurso. A maioria das execuções resultou em aptidão inferior a 700 pontos, indicando falha nas curvas iniciais. Em contrapartida, as execuções 3, 5, 7 e 8 alcançaram pontuações entre 1280 e 1730, avançando na pista, mas sem computar voltas completas.

Em contraste, o cenário com a velocidade informada apresentou alta taxa de convergência. Das 10 execuções, 9 superaram 4000 pontos, o que corresponde à conclusão de pelo menos uma volta. A Execução 5 obteve a melhor performance global (9320 pontos), seguida pelas Execuções 8 (9060) e 6 (8790). Estes resultados indicam que os agentes não

Tabela 2. Resultados de Eficácia: Evolução da Aptidão e Métricas de Convergência (10 Execuções)

Cenário	Execução	Voltas Completas	Geração de Convergência	Aptidão vs. Gerações					
				1	10	25	50	75	100
Velocidade não Informada	1	0	–	390	450	450	450	450	450
	2	0	–	410	420	430	630	630	630
	3	0	–	410	1280	1280	1280	1280	1280
	4	0	–	410	440	440	440	440	440
	5	0	–	400	430	1730	1730	1730	1730
	6	0	–	400	420	440	440	450	450
	7	0	–	400	430	1340	1340	1340	1340
	8	0	–	400	420	470	470	1580	1710
	9	0	–	400	420	420	420	430	430
	10	0	–	410	620	620	620	620	980
Velocidade Informada	1	1	7	400	4550	4550	4550	4710	4710
	2	1	80	140	3380	3380	3380	3410	7210
	3	0	–	220	1380	1380	1500	2240	3270
	4	1	53	130	1410	2420	3340	6780	7220
	5	2	33	270	1440	2790	7690	7810	9320
	6	2	6	390	8790	8790	8790	8790	8790
	7	1	53	1400	2690	2690	3340	5880	6350
	8	2	26	610	3270	5070	6110	7500	9060
	9	1	98	410	1340	1340	1340	3210	5730
	10	1	41	360	1390	3130	5320	6890	6890

apenas completaram o trajeto, mas mantiveram velocidade suficiente para iniciar voltas subsequentes dentro do tempo limite.

Outro aspecto relevante é a dispersão dos resultados intra-cenário. Como ilustrado na Tabela 2, a natureza estocástica do algoritmo resultou em pontuações distintas para cada execução, validando a importância de realizar múltiplos testes para mitigar vieses. Contudo, os dados confirmam que a inclusão da velocidade facilita a convergência do modelo. Sem essa informação, a rede *feedforward* enfrenta dificuldades para conduzir o veículo, visto que não possui memória para estabelecer correlações temporais apenas com informações de distância.

Dando sequência, a Tabela 3 detalha a evolução da complexidade topológica (número de neurônios e conexões) ao longo das gerações. Inicialmente, a diversidade de topologias finais reforça a natureza estocástica do algoritmo NEAT, que explorou diferentes espaços de busca para cada execução. Isso também é evidenciado pelo fato de que diferentes topologias em gerações distintas atingiram a melhor aptidão, especialmente para o cenário de velocidade informada.

Ao comparar os cenários, observa-se uma distinção nas estratégias evolutivas. No cenário com velocidade não informada, houve uma tendência à tornar a topologia mais complexa: as redes mantiveram um número elevado de conexões, e frequentemente adicionaram neurônios ocultos. Esse comportamento sugere que o algoritmo tentou compensar a falta de informação e a dificuldade de convergência aumentando a capacidade computacional da rede. Em contrapartida, no cenário com a velocidade informada, a evolução

Tabela 3. Resultados de Complexidade: Evolução Topológica (Nós/Conexões) por Geração (10 Execuções)

Cenário	Execução	Topologia do Melhor Indivíduo (Neurônios/Conexões)					
		Gen 1	Gen 10	Gen 25	Gen 50	Gen 75	Gen 100
Velocidade não Informada	1	4/20	6/20	6/20	6/20	6/20	6/20
	2	4/20	4/13	5/12	4/15	4/15	4/15
	3	4/20	4/19	4/19	4/19	4/19	4/19
	4	4/20	4/20	4/20	4/20	4/20	4/11
	5	4/20	4/16	5/8	5/8	5/8	5/8
	6	4/20	5/13	6/10	6/10	7/11	7/11
	7	4/20	4/13	6/11	6/11	6/11	6/11
	8	4/20	5/15	9/15	9/15	9/15	9/12
	9	4/20	4/15	4/15	4/15	4/14	4/14
	10	4/20	5/9	5/9	5/9	5/9	6/11
Velocidade Informada	1	4/24	5/16	5/16	5/16	4/4	4/4
	2	4/24	5/12	5/12	5/12	5/12	8/12
	3	4/24	4/14	4/7	4/6	4/5	4/3
	4	4/24	4/13	7/12	11/12	12/11	8/8
	5	4/24	6/26	4/4	6/7	4/4	4/5
	6	4/24	4/20	4/20	4/20	4/20	4/20
	7	4/24	4/23	4/23	4/7	5/8	6/8
	8	4/24	5/16	5/12	5/7	6/8	7/8
	9	4/24	6/19	6/19	6/19	4/8	4/4
	10	4/24	4/20	4/21	5/10	5/5	5/5

favoreceu a simplificação, demonstrando que estruturas mínimas foram suficientes para solucionar o problema.

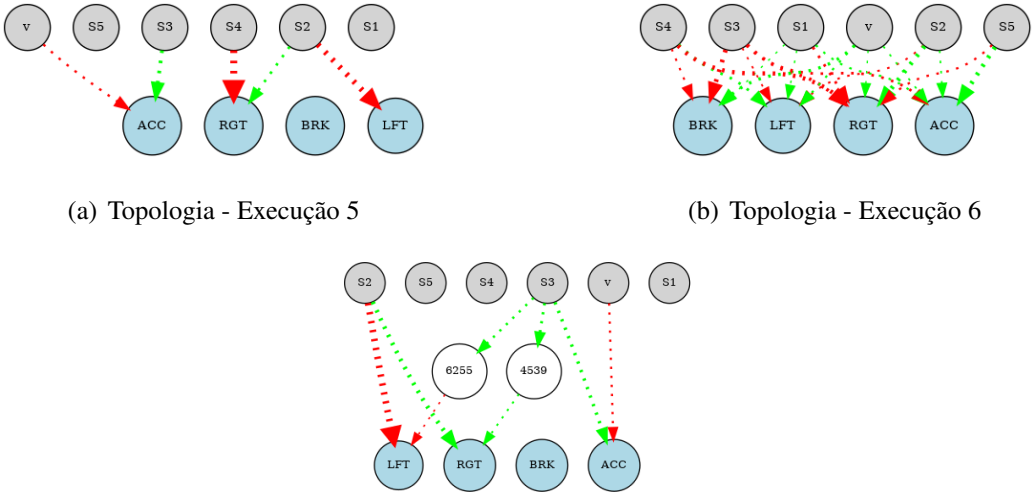
Por fim, a Figura 10 ilustra as topologias das melhores execuções (5, 6 e 8) para o cenário com velocidade informada. A Execução 8 apresentou uma topologia mais complexa, indicando o uso de camadas ocultas para processar os dados. A Execução 6 manteve uma estrutura densa (4 neurônios e 20 conexões), utilizando praticamente todos os dados de entrada disponíveis. Já a Execução 5 convergiu para uma abordagem minimalista (4 neurônios e apenas 5 conexões), descartando a maioria dos inputs e operando sem camadas ocultas. Essa variabilidade confirma que, para este problema, múltiplas arquiteturas de rede podem atingir a eficácia.

5.2. Análise Qualitativa

Após a avaliação quantitativa e topológica, esta seção dedica-se à análise qualitativa do comportamento dos agentes nos ambientes simulados. O objetivo é compreender como as redes neurais evoluídas processam os dados sensoriais para executar ações de controle. A inspeção visual das trajetórias permitiu identificar que a disponibilidade da informação de velocidade foi determinante na definição das estratégias de navegação[3].

No cenário com velocidade não informada, a ausência de *feedback* sobre o próprio movimento induziu comportamentos pouco adaptativos. Nas execuções de menor aptidão

[3]Os vídeos das execuções analisadas nesta seção encontram-se disponíveis no repositório do projeto.



(a) Topologia - Execução 5

(b) Topologia - Execução 6

(c) Topologia - Execução 8

Figura 10. Topologias finais do melhor individuo após 100 gerações para as execuções: (a) Execução 5, (b) Execução 6 e (c) Execução 8. Os nós cinzas representam as entradas (sensores e velocidade); os azuis, as saídas (atuadores); e os brancos, a camada oculta. As conexões são indicadas por linhas pontilhadas: verde para pesos positivos e vermelho para negativos, onde a espessura da linha é proporcional à magnitude da conexão.

(1, 4, 6 e 9), observou-se uma tendência à colisão por excesso de velocidade: os agentes priorizavam a recompensa imediata por meio da aceleração máxima, falhando consistentemente na primeira curva.

Já nas execuções que conseguiram progredir (como a 5 e a 8), emergiu uma estratégia baseada no aproveitamento da inércia. Nesses casos, os agentes alternavam entre aceleração e "ponto morto", sem utilizar o freio de forma efetiva. Embora tenha permitido avanços parciais, essa técnica mostrou-se insuficiente para transpor curvas complexas que exigiam redução ativa da velocidade.

Em contrapartida, no cenário com velocidade informada, a inclusão da velocidade instantânea como entrada viabilizou técnicas mais refinadas, como a modulação por largura de pulso (PWM) nos atuadores. Entre os casos de sucesso, identificaram-se três perfis distintos:

- O primeiro (Execuções 5 e 8) caracterizou-se pela eficiência via controle do acelerador: o agente da Execução 5, por exemplo, aboliu o uso do freio, controlando a velocidade exclusivamente por pulsos no acelerador e utilizando a inércia apenas nos vértices das curvas.
- O segundo perfil (Execução 6) adotou uma estratégia oposta: manteve a aceleração constante no máximo e controlou a velocidade via modulação do freio, resultando em uma condução agressiva no limite da pista.
- Por fim, um perfil híbrido foi notado nas demais execuções, onde os agentes demonstraram uma evolução no aprendizado: inicialmente cautelosos e dependentes do freio, progressivamente minimizam seu uso para priorizar o momento linear.

6. Conclusão

Este trabalho apresentou o desenvolvimento e a aplicação de controladores baseados em redes neurais artificiais, evoluídos através do algoritmo NEAT, para a condução de veículos autônomos em um ambiente de simulação 2D. O objetivo principal foi analisar a capacidade de adaptação do algoritmo sob duas configurações de entrada: agentes com acesso à velocidade instantânea versus agentes dependentes exclusivamente de sensores de distância.

A análise experimental comprova que a eficácia da navegação depende da qualidade dos dados de entrada. Enquanto o cenário com velocidade informada induziu a convergência para topologias minimalistas e controle eficiente via modulação, o cenário oposto resultou em complexidade estrutural excessiva e ineficaz. Esse contraste demonstra que a modelagem do problema é de suma importância para explorar adequadamente o algoritmo NEAT e evoluir topologias adequadas ao problema.

Este estudo apresenta algumas limitações relacionadas à simplificação da simulação, que restringiu aspectos físicos e focou em um cenário de agente único, desconsiderando as complexidades do tráfego real. Outra restrição refere-se ao design experimental. Devido ao custo computacional, a evolução foi limitada a 100 gerações. Contudo, o Apêndice A explora testes estendidos com $10{,}000$ gerações para amenizar essa questão. Por fim, os hiperparâmetros do algoritmo foram mantidos fixos, impedindo a análise de diferentes configurações para o algoritmo evolutivo. Em especial, destaca-se a possível relevância em explorar redes recorrentes para o cenário sem velocidade informada.

Para trabalhos futuros, sugere-se investigar diferentes combinações de hiperparâmetros e diversificar a modelagem inicial, introduzindo ruído nos sensores. Além disso, é de suma importância validar a capacidade de generalização dos indivíduos obtidos, testando os melhores controladores em pistas inéditas para garantir que não houve apenas memorização do traçado atual. Recomenda-se, ainda, a exploração de algoritmos de *novelty* [21] e *surprise* [22] *search*, que podem incentivar o surgimento de comportamentos distintos e inovadores.

Referências

[1] United Nations Office for Disaster Risk Reduction and International Science Council, "UNDRR–ISC Hazard Information Profiles – 2025 Update: TL0405 Road Traffic Accident," 2025. Disponível em: https://www.undrr.org/terms/hips/TL0405. Acesso em: 06/12/25.

[2] Mobileye, "A brief history of autonomous vehicles – from renaissance to reality," 2025. Disponível em: https://www.mobileye.com/blog/history-autonomous-vehicles-renaissance-to-reality/. Acesso em: 06/12/25.

[3] E. D. Dickmanns, "The development of machine vision for road vehicles in the last decade," in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 1, pp. 268–281, IEEE, 2002.

[4] E. D. Dickmanns, *Dynamic vision for perception and control of motion*. Springer, 2007.

[5] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: autonomous vehicles in city traffic*, vol. 56. Springer Science & Business Media, 2009.

[6] T. Litman, "Autonomous vehicle implementation predictions," 2017. Disponível em: https://www.vtpi.org/avip.pdf. Acesso em: 06/12/25.

[7] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 712–733, 2020.

[8] J. Ren, H. Gaber, and S. S. Al Jabar, "Applying deep learning to autonomous vehicles: A survey," in *2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pp. 247–252, IEEE, 2021.

[9] C. White, M. Safari, R. Sukthanker, B. Ru, T. Elsken, A. Zela, D. Dey, and F. Hutter, "Neural architecture search: Insights from 1000 papers," *arXiv preprint arXiv:2301.08727*, 2023.

[10] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 24–35, 2019.

[11] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.

[12] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[13] S. O. Haykin, *Neural Networks and Learning Machines*. Upper Saddle River, NJ: Pearson, 3 ed., 2008.

[14] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural Network Design (2nd Edition)*. Martin Hagan, 2 ed., 2014.

[15] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*, pp. 1–16, PMLR, 2017.

[16] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics: Results of the 11th international conference*, pp. 621–635, Springer, 2017.

[17] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "Torcs, the open racing car simulator," 2000. Disponível em: http://torcs.sourceforge.net. Acesso em: 06/12/25.

[18] L. Cardamone, D. Loiacono, and P. L. Lanzi, "On-line neuroevolution applied to the open racing car simulator," in *2009 IEEE Congress on Evolutionary Computation*, pp. 2622–2629, IEEE, 2009.

[19] D. Loiacono, L. Cardamone, and P. L. Lanzi, "Learning to drive in the open racing car simulator using online neuroevolution," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 3, pp. 176–190, 2010.

[20] M. Salem, A. Mora, J. Merelo, and P. García-Sánchez, "Evolving a torcs modular fuzzy driver using genetic algorithms," in *International Conference on the Applications of Evolutionary Computation*, pp. 340–353, Springer, 2018.

[21] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary computation*, vol. 19, no. 2, pp. 189–223, 2011.

[22] D. Gravina, A. Liapis, and G. Yannakakis, "Surprise search: Beyond objectives and novelty," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 677–684, 2016.

[23] W. B. Langdon and R. Poli, "Fitness causes bloat," in *Soft Computing in Engineering Design and Manufacturing*, pp. 13–22, Springer, 1997.

[24] L. Trujillo, L. Muñoz, E. Galván-López, and S. Silva, "neat genetic programming: Controlling bloat naturally," *Information Sciences*, vol. 333, pp. 21–43, 2016.

Apêndice A: Análise para 10,000 Gerações

Os experimentos reportados na seção principal deste trabalho foram limitados a 100 gerações devido ao alto custo computacional associado à simulação de cada indivíduo[4]. No entanto, essa restrição levanta questionamentos em relação a convergência dos indivíduos: o cenário sem informação de velocidade (Cenário 2) seria capaz de completar uma volta se submetido a um processo evolutivo mais longo? De forma análoga, qual seria o desempenho para o cenário com velocidade informada (Cenário 1)?

Para endereçar essas questões, este apêndice apresenta os resultados de experimentos estendidos para 10,000 gerações. Devido à inviabilidade de replicar a metodologia completa[5], optou-se pela realização de uma única execução para cada cenário. A análise concentra-se nos resultados finais de aptidão, comportamento de navegação e topologia das redes resultantes.

Em termos quantitativos, foram observados ganhos de aptidão em ambos os cenários. O indivíduo do Cenário 2 obteve uma aptidão final de 2,910 pontos. Embora este valor seja superior aos valores observados nas execuções de 100 gerações, ele ainda é insuficiente para caracterizar uma volta completa. Já o melhor indivíduo do Cenário 1 alcançou a aptidão de 10,040 pontos, também superando as métricas obtidas nos experimentos originais.

A análise comportamental revela padrões distintos de adaptação. O agente do cenário sem velocidade demonstrou um aprendizado parcial. Inicialmente, o veículo adota uma estratégia cautelosa, aplicando aceleração mínima suficiente apenas para superar a inércia e transpor os *checkpoints* iniciais. Contudo, ao aproximar-se da antepenúltima curva do circuito, o agente aciona a aceleração máxima e acaba colidindo.

Já no cenário com velocidade informada, o comportamento manteve-se consistente com o observado nos experimentos originais. A principal diferença ao longo das gerações foi o aumento incremental da velocidade média. Todavia, o limite de velocidade não foi atingido; o indivíduo possivelmente estabilizou em uma solução de ótimo local.

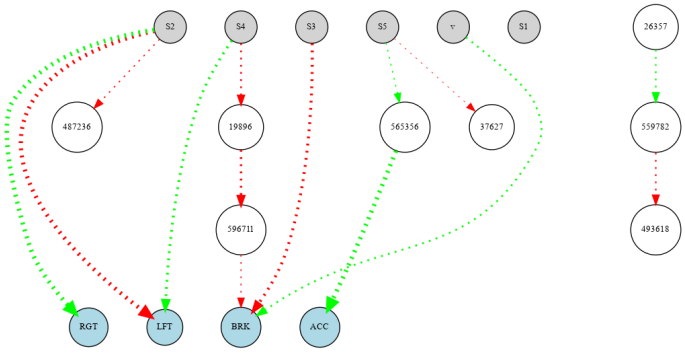


Figura 11. Topologia final do melhor agente do cenário com velocidade informada após 10,000 gerações. Destaca-se a presença de estruturas vestigiais não conectadas à camada de saída.

[4]Ressalta-se que, à medida que os agentes evoluem e tornam-se capazes de navegar por períodos mais longos na pista, o tempo de processamento por geração aumenta.

[5]Uma única execução do Cenário 1 com 10,000 gerações demandou $\approx$ 7 dias de processamento.

Adicionalmente, a análise da topologia evoluída no Cenário 1 revela uma estrutura mais complexa do que a observada no trabalho original. Além disso, observa-se a presença de neurônios e conexões que não possuem ligação com a camada de entrada nem de saída (Figura 11). A persistência desses artefatos é um fenômeno relatado na literatura, onde material genético redundante é preservado ao longo das gerações por não impactar negativamente a aptidão do indivíduo [23, 24].

Concluindo, os experimentos adicionais evidenciam que o aumento isolado do número de gerações não é suficiente para sanar as deficiências do Cenário 2. Para torná-lo viável, são necessárias alterações na modelagem do agente ou nos hiperparâmetros. Além disso, observa-se que, embora a execução com $10{,}000$ gerações produza agentes mais aptos, o ganho de desempenho não é proporcional ao custo computacional investido. Portanto, aprimorar a modelagem do problema mostra-se uma estratégia mais eficiente do que apenas estender o tempo de simulação.