3ª Avaliação de Inteligência Artificial

Ailesson Nando Souza de Araujo - 22053243
Bruno de Moura Solimões - 22051316
Isabele Martins Nascimento - 22154387
Jakeline Gimaque de Mesquita - 22050618
Luiz Gabriel Favacho de Almeida - 22153921
Mateus de Jesus Santos - 22152028

Agosto de 2024

1 Introdução

O presente trabalho tem como foco o problema conhecido como "O Trem de Michalski". O principal objetivo deste estudo é analisar e relatar os resultados obtidos utilizando dois modelos distintos: o neuro-simbólico e o LTNTorch. Além disso, busca-se identificar e discutir as possíveis razões para as diferenças observadas nos resultados ao aplicar os modelos citados à mesma tarefa.

Visando uma compreensão completa do estudo, será apresentada uma breve explicação dos modelos utilizados, seguida por uma descrição detalhada do problema e dos métodos aplicados. Em seguida, serão expostos os dados coletados e os resultados obtidos. Por fim, serão feitos comentários sobre os resultados, com o intuito de responder às questões propostas para este trabalho, destacando as implicações dos achados.

Como solicitado, as soluções deste trabalho estão disponíveis no GitHub.

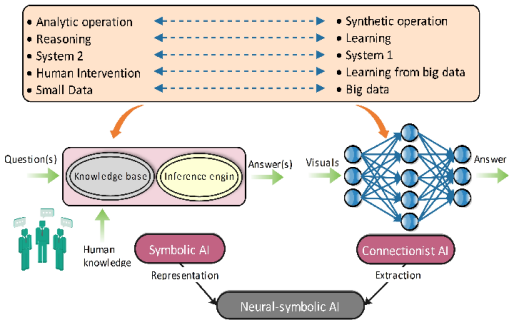
2 Referencial Teórico

2.1 Sistemas Neuro-Simbólicos

Os sistemas neuro-simbólicos representam uma abordagem inovadora na inteligência artificial (IA), que busca integrar o poder de aprendizagem das redes neurais (percepção de baixo nível) com a capacidade de raciocínio simbólico (raciocínio de alto nível), conforme descrito em [1]. Esta combinação visa superar as limitações de ambas as abordagens quando utilizadas isoladamente. Redes neurais são conhecidas por sua habilidade em aprender a partir de grandes quantidades de dados e realizar tarefas complexas, como reconhecimento de padrões e classificação. No entanto, elas são frequentemente vistas como caixas-pretas, com falta de interpretabilidade e dificuldade em incorporar conhecimento explícito. Por outro lado, sistemas simbólicos, baseados em lógica e regras, oferecem alta interpretabilidade e facilidade de incorporação de conhecimento pré-existente, mas têm dificuldade em lidar com incerteza e grandes volumes de dados.

A integração neuro-simbólica busca aproveitar o melhor dos dois mundos. Em essência, sistemas neuro-simbólicos utilizam redes neurais para processamento e aprendizado de dados, enquanto utilizam componentes simbólicos para raciocínio e interpretação dos resultados. Esta combinação permite que os sistemas de IA não apenas aprendam com dados, mas também utilizem e apliquem conhecimento lógico de forma transparente e explicável.

Figure 1: Diagrama de integração entre redes neurais e módulos simbólicos.

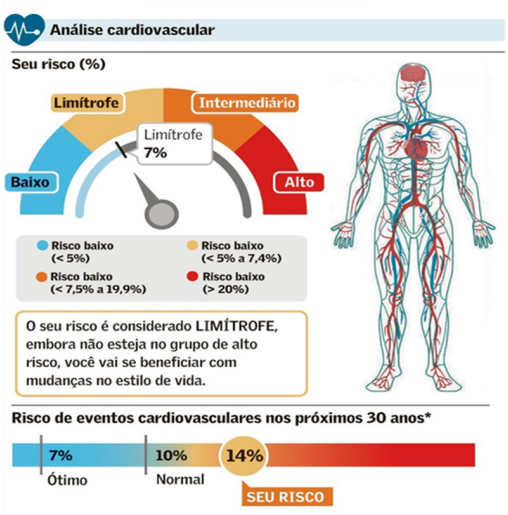


O funcionamento desses sistemas envolve a coordenação entre módulos neurais e simbólicos. As redes neurais são responsáveis por extrair características e padrões a partir dos dados brutos, como imagens, texto ou sinais. Esses padrões são então traduzidos em representações simbólicas que podem ser manipuladas por sistemas de raciocínio lógico. Por exemplo, em um sistema de reconhecimento de imagens, uma rede neural pode identificar objetos em uma imagem, enquanto um módulo simbólico pode deduzir relações entre esses objetos com base em regras lógicas predefinidas.

A importância dos sistemas neuro-simbólicos reside em sua capacidade de melhorar a interpretabilidade e a explicabilidade dos modelos de IA. Em muitas aplicações, como medicina, finanças e segurança, é crucial que os sistemas de IA possam justificar suas decisões de maneira compreensível para humanos. Além disso, essa abordagem facilita a incorporação de conhecimento especializado e regras de negócio, permitindo que os modelos aprendam mais eficientemente e de maneira mais alinhada com o conhecimento humano.

Os sistemas neuro-simbólicos têm sido aplicados em diversos problemas de IA. Na medicina, por exemplo, eles são usados para combinar dados clínicos com conhecimento médico para diagnósticos mais precisos e explicáveis. Na robótica, permitem que robôs aprendam a partir de experiências enquanto seguem regras de segurança e comportamento. Na área de processamento de linguagem natural, ajudam a melhorar a compreensão de texto e a resposta a perguntas, combinando aprendizado de linguagem com conhecimento ontológico.

Figure 2: Resultado da aplicação prática de sistemas neuro-simbólicos na medicina.



Logo, os sistemas neuro-simbólicos representam um avanço significativo na IA, combinando o aprendizado poderoso das redes neurais com a clareza e a estrutura dos sistemas simbólicos. Esta abordagem híbrida não só melhora o desempenho dos sistemas de IA, mas também aumenta sua capacidade de explicar e justificar decisões, tornando-os mais confiáveis e aplicáveis em áreas críticas.

2.2 LTNTorch

O LTNTorch é uma biblioteca de aprendizagem de máquina junta em um único framework aprendizagem simbólico e redes neurais. O obejtivo do LTNTorch é combinar a manipulação e inferencia sobre conhecimentos explícitos e lógicos do raciocínio simbólico com a eficacia em tarefaz que tenham dados perceptuais e não estruturados do aprendizado de redes neurais.

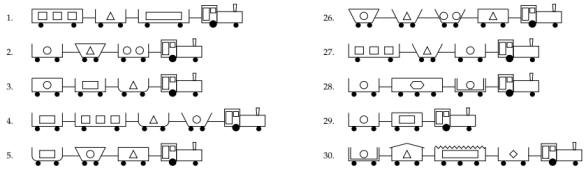
O LTNTorch permite que seja definido e manipulado conhecimentos simbólicos usando lógica de predicados. O framework facilita a integração de redes neurais para aprender representações e inferências a partir de dados perceptuais e não estruturados como imagens e textos.

A biblioteca integra o aprendizado simbólico com as redes neurais por meio de uma abordagem chamada de Aprendizado Lógico-Neuronal (Logical Neural Networks, LNNs). Essa abordagem é composta pelos: predicados Neurais, funções parametrizadas que representam redes neurais que mapeiam entradas para valores que representam a verdade de um predicado; Funções Neurais são mapeamentos parametrizados que podem ser utilizados dentro de expressões lógicas para capturar relações complexas entre variáveis; Lógica Fuzzy que permite que predicados e funções retornem valores contínuos, representando graus de verdade, ao invés de de valroes binários; Regularização por Regras Lógicas que são usadas como uma forma de regularização durante o treinamento das redes neurais, impondo estruturas sobre os dados qeu as redes devem aprender a respeitar, o que pode levar a modelos mais robustos e interpretáveis.

3 Descrição do Problema

O Trem de Michalski é um clássico problema de aprendizado de máquina e raciocínio simbólico, originalmente proposto por Ryszard S. Michalski nos anos 1970. Seu principal objetivo é que um sistema de aprendizado de máquina descubra as regras lógicas que explicam porque certos trens estão indo para o leste enquanto outros vão para o oeste, com base nos atributos dos vagões.

Figure 3: Amostra dos trens do arquivo fornecido pelo professor.



A Figura 3 apresenta uma amostra dos trens do arquivo `100_trains_set`. Nela, é possível observar algumas das características dos trens, que serão detalhadas a seguir. O primeiro desafio encontrado foi a transformação dos dados das 100 imagens em uma tabela, que posteriormente seria convertida para o formato .csv. Esse processo foi realizado manualmente por três pessoas e, em seguida, os dados foram revisados por outras duas. Essa etapa foi realizada com especial cuidado pois é a principal fonte para o treinamento dos modelos. Dessa forma, os trens foram descritos com os seguintes atributos:

1. Quantidade de vagões (valor entre 3 e 5);
2. Quantidade de cargas diferentes que pode levar (valor entre 1 e 4);
3. Para cada vagão de um trem:
   (a) Quantidade de eixos com rodas (valor 2 ou 3);
   (b) Comprimento (valor curto ou longo);
   (c) Formato da carroceria do vagão, que pode ser:
   - Retângulo fechado
   - Retângulo aberto

- Retângulo topo dentado
- Retângulo topo triangular
- Duplo retângulo aberto
- Duplo retângulo fechado
- Duplo retângulo topo triangular
- Trapézio aberto
- Trapézio fechado
- Trapézio topo triangular
- Formato U topo triangular
- Formato U aberto
- Formato U fechado
- Elipse
- Hexágono

(d) Quantidade de cargas no vagão (0 a 3);

(e) Formato da carga (círculo, hexágono, retângulo ou triângulo);

Com o objetivo de aprimorar as conclusões dos modelos, foram propostas 10 proposições (variáveis booleanas) que descrevem se qualquer par de tipos de carga estão ou não em vagões adjacentes no trem (considerando que cada vagão carrega um único tipo de carga). As seguintes relações foram definidas em relação aos vagões de um trem, com valores lógicos variando entre -1 (Falso) e 1 (Verdadeiro):

1. Existe um retângulo próximo de outro retângulo.
2. Existe um retângulo próximo de um triângulo.
3. Existe um retângulo próximo de um hexágono.
4. Existe um retângulo próximo de um círculo.
5. Existe um triângulo próximo de outro triângulo.
6. Existe um triângulo próximo de um hexágono.
7. Existe um triângulo próximo de um círculo.
8. Existe um hexágono próximo de outro hexágono.
9. Existe um hexágono próximo de um círculo.
10. Existe um círculo próximo de outro círculo.

Por fim, há um único atributo de classe que define a direção de um trem, no caso, se é leste ou oeste.

Figure 4: Demostração da meta-rede utilizada.

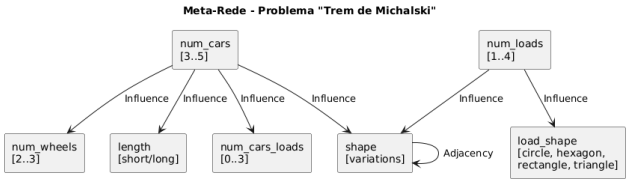
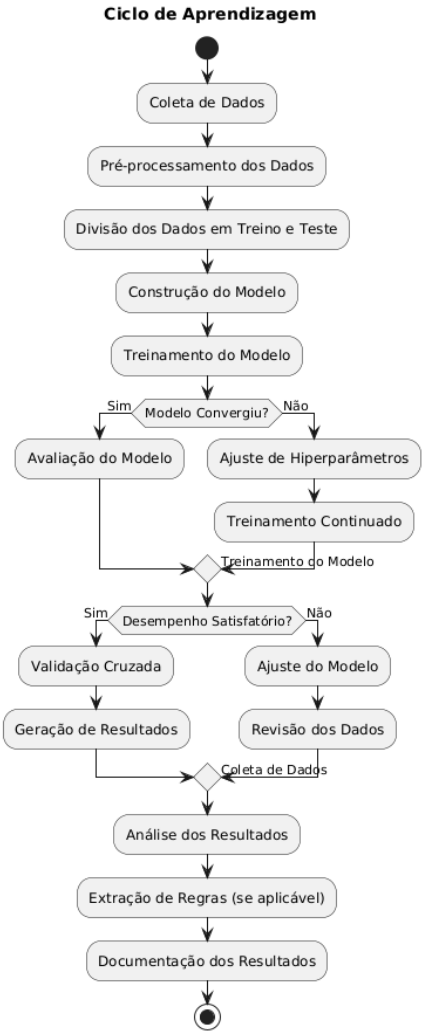


Figure 5: Demostração do ciclo de apredizagem utilizado.



4 Descrição do Modelo

4.1 Implementação 1

A implementação envolve a adaptação e execução de um modelo neuro-simbólico para a classificação de trens. A seguir, apresentamos uma descrição detalhada dos passos realizados:

Preparação dos Dados: Os dados foram lidos de um arquivo CSV e organizados em um DataFrame utilizando a biblioteca pandas. Em seguida, os dados foram separados em subconjuntos conforme os atributos dos trens, como a quantidade de vagões, o número de cargas, e características específicas de cada vagão (número de eixos, comprimento, formato da carroceria, quantidade de cargas, e formato das cargas).

Estruturação das Entradas: As entradas para o modelo foram estruturadas através da transformação dos dados em arrays numpy. Cada atributo relevante foi convertido para um formato numérico adequado

para alimentar a rede neural. Por exemplo, os tipos de carga foram convertidos para valores numéricos sequenciais.

Divisão dos Dados: Os dados foram divididos em conjuntos de treinamento e teste. Inicialmente, 70% dos dados foram usados para treinamento e 30% para teste. Além disso, foi realizada uma validação cruzada para avaliar a robustez do modelo, onde os dados foram divididos em diferentes subconjuntos de treinamento e teste em múltiplas iterações.

Definição do Modelo: A arquitetura do modelo neuro-simbólico foi definida utilizando a biblioteca Keras. O modelo inicial consistiu em uma rede neural sequencial com várias camadas densas. As camadas incluíram neurônios suficientes para capturar as complexidades dos dados, com funções de ativação ReLU para introduzir não-linearidades.

Hiperparâmetros Utilizados:

- Neurônios por camada: 32 de entrada e 32 de saída
- Função de ativação: ReLU
- Função de perda: binary crossentropy
- Otimizador: Adam
- Épocas de treinamento: 200
- Tamanho do batch: 32

Treinamento e Avaliação do Modelo: O modelo foi treinado utilizando os dados de treinamento, e a performance foi avaliada utilizando os dados de teste. A validação cruzada foi realizada para garantir que o modelo generalizasse bem para novos dados, neste processo, os dados foram divididos em 5 subconjuntos (folds). Os resultados foram comparados com os obtidos através da validação cruzada para analisar a consistência do modelo. Durante o treinamento, foram monitoradas métricas como a precisão e a perda para ajustar hiperparâmetros conforme necessário.

4.2 Implementação 2

A implementação do design e da arquitetura do modelo é dividida em cinco partes: preparação dos dados, estruturação das entradas, divisão dos dados, definição do modelo e, por fim, o treinamento e avaliação do modelo.

Preparação dos Dados: Os dados são lidos do arquivo CSV e organizados em um DataFrame. As colunas são separadas em diferentes subconjuntos, agrupadas por um critério específico.

Estruturação das Entradas: Esta parte organiza as entradas para o modelo, separando-as em arrays numpy com base em suas caracteristicas, facilitando a alimentação dos dados no modelo.

Divisão dos Dados: Os dados foram divididos em conjuntos de treinamento e teste com base em um índice de validação. Além disso, separa as características das etiquetas de classe.

Definição do Modelo: A arquitetura do modelo é composta por várias sub-redes que processam diferentes combinações de entradas. Cada sub-rede é responsável pro uma tarefa específica, como prever o número de vagões ou a forma da carga.

Treinamento e Avaliação do Modelo: O modelo é treinado e avaliado utilizando validação cruzada com iterações. Para cada iteração, os dados são divididos em treinamento e teste, e o modelo é compilado e treinado.

5 Resultados e Análises

5.1 Questão 1

5.1.1 Questão 1.1: Suficiência dos Predicados

Para determinar a suficiência dos predicados fornecidos, foi realizada uma análise preliminar utilizando os 11 predicados iniciais sugeridos. Os resultados indicam que esses predicados são adequados para a tarefa de classificação dos 100 trens. Os predicados foram suficientes para diferenciar entre os trens que se dirigem para o leste e os que se dirigem para o oeste. Portanto, não foi necessário adicionar novos predicados à descrição lógica.

5.1.2 Questão 1.2.a: Treinamento e Teste com Divisão 70/30

Para a primeira parte do treinamento, os dados foram divididos em 70% para treinamento e 30% para teste. O modelo foi treinado utilizando essa divisão, e os resultados mostraram um desempenho consistente tanto no conjunto de treinamento quanto no de teste.

Figure 6: Evolução da perda (loss) durante o treinamento e a validação do modelo.

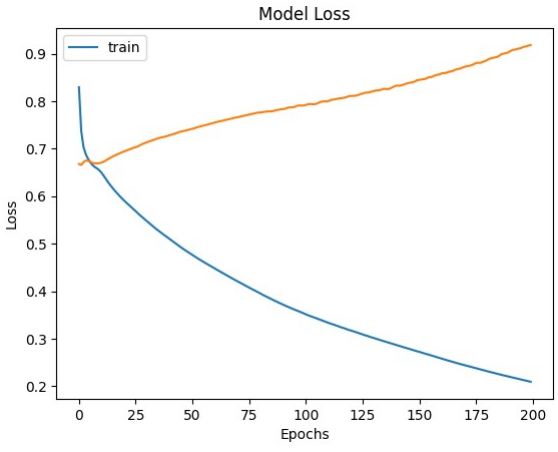


Figure 7: Evolução da acurácia (accuracy) durante o treinamento e a validação do modelo.

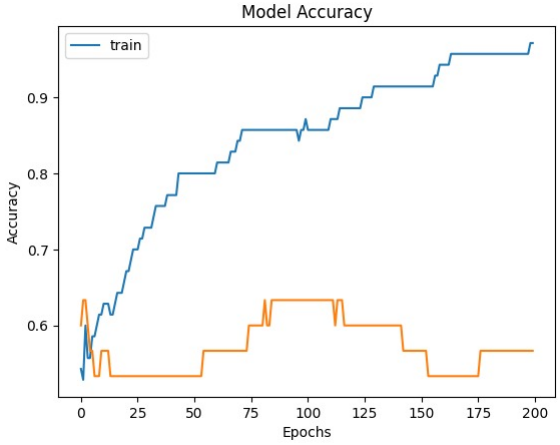
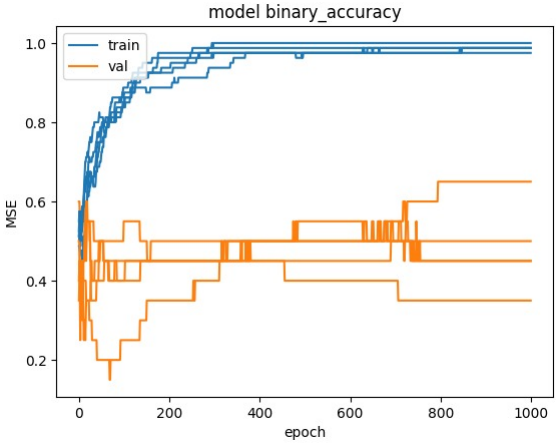


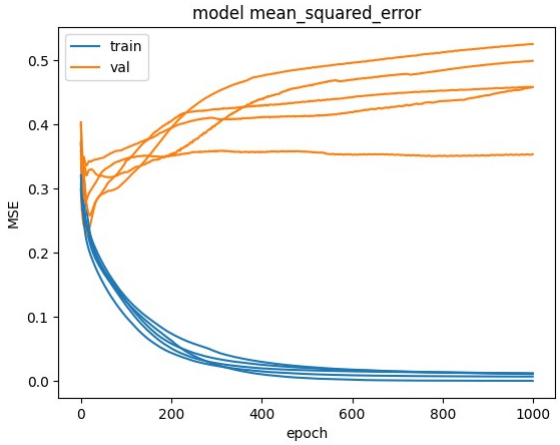
Figure 9: Gráfico de Acurácia Binária (Model Binary Accuracy) na validação cruzada.



5.1.3 Questão 1.2.b: Validação Cruzada

A validação cruzada foi realizada com 5 folds, onde cada fold utilizou uma parte dos dados para teste e o restante para treinamento. Isso permitiu uma avaliação mais robusta e confiável do desempenho do modelo.

Figure 8: Erro quadrático médio (mean squared error) para cada fold durante a validação cruzada.



Comparando os resultados da divisão 70/30 com os da validação cruzada, podemos observar o seguinte:

- **Consistência dos Resultados**: A validação cruzada apresenta uma variação menor nos erros e na acurácia entre os folds, indicando um desempenho mais consistente em diferentes subconjuntos de dados. A divisão 70/30 também mostra uma boa performance, mas a validação cruzada proporciona uma avaliação mais robusta do modelo.
- **Desempenho Geral**: Ambas as abordagens indicam que o modelo é eficaz na classificação dos trens, com perdas decrescentes e acurácias crescentes ao longo do treinamento. A acurácia média obtida na validação cruzada confirma que o modelo generaliza bem, com uma ligeira vantagem em termos de confiabilidade em comparação à divisão 70/30.
- **Recomendações**: A utilização de validação cruzada é recomendada para futuras avaliações do modelo, pois oferece uma medida mais confiável do desempenho real do modelo. A análise das perdas e acurácias sugere que o modelo atual está bem ajustado, sem sinais claros de overfitting ou underfitting.

Essa comparação demonstra a eficácia do modelo neuro-simbólico na tarefa de classificação de trens, reforçando os resultados obtidos e a robustez do método utilizado.

5.2 Questão 2

Na questão 2, ao testar uma solução em LTNTorch para o problema dos trens, com base em 11 predicados aplicados aos 10 trens do livro, foram obtidos resultados significativos. A figura 10 mostra a tabela gerada o resultado da questão 2.1. A coluna 'Accuracy' indica a precisão da classificação de cada trem. A coluna 'Output of flat network' apresenta os valores de saída da rede neural, variando entre 0 e 1, e representando a probabilidade de o trem pertencer à classe 'east'. A coluna 'Desired output' indica o valor esperado para cada trem (1 para 'east' e 0 para 'west'). Por fim, a coluna 'Class' mostra a classificação final do trem, ou seja, se ele pertence à região leste ou oeste."

A análise dos resultados da classificação dos trens revela que o modelo, em sua maioria, conseguiu classificar os trens com alta precisão, demonstrando um bom desempenho. Entretanto, o trem 1 se destacou como um caso em que a precisão da classificação foi ligeiramente inferior, indicando que o modelo pode ter encontrado características nesse trem que dificultaram a classificação correta. A saída da rede neural, que representa a probabilidade de um trem pertencer à classe "east", mostrou-se um bom indicador da classe final atribuída. Ou seja, quanto mais próximo de 1 estiver o valor da saída da rede neural, maior a probabilidade do

Figure 10: Resultado por treinamento com LTNTorch.

	Train	Cars Accuracy	Output of flat network	Desired output	Class
0	0	1.000000	1.00	1.0	east
1	1	0.888889	1.00	1.0	east
2	2	1.000000	1.00	1.0	east
3	3	1.000000	0.79	1.0	east
4	4	1.000000	0.16	0.0	west
5	5	1.000000	0.05	0.0	west
6	6	1.000000	0.00	0.0	west
7	7	1.000000	0.00	0.0	west
8	8	1.000000	0.00	1.0	east
9	9	1.000000	0.19	0.0	west

trem ser classificado como "east". No entanto, em alguns casos, como no trem 8, a saída da rede neural não correspondeu à classe final atribuída, sugerindo que o modelo pode ter tido dificuldades em classificar esses trens com alta confiança, possivelmente devido a características ambíguas ou a falta de dados suficientes para esses casos específicos.

Figure 11: Resultado do treinamento com o código fornecido.

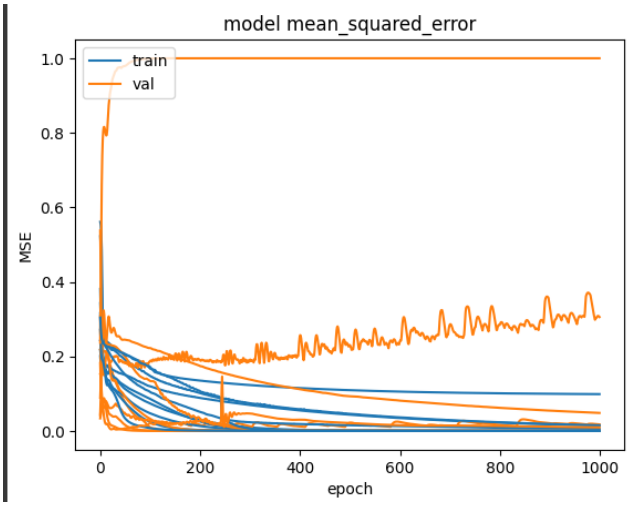
trem	saída contínua	saída discreta	saída esperada
0	1.00	1	1.0
1	1.00	1	1.0
2	1.00	1	1.0
3	1.00	1	1.0
4	0.83	1	1.0
5	0.37	0	0.0
6	0.00	0	0.0
7	0.54	1	0.0
8	0.00	0	0.0
9	0.00	0	0.0

Ao analisarmos as tabelas 10 e 11 é possível notar que LTNTorch é a melhor escolha devido à sua capacidade de integrar diretamente lógica simbólica com aprendizado profundo, sua eficiência em comparação com o código fornecido, e sua escalabilidade para lidar com conjuntos de dados maiores. Ele proporciona uma solução mais direta e adequada para implementar, testar e comparar os resultados do problema dos trens usando predicados lógicos.

Na segunda parte da questão 2 foi a plicado o código em LTNTorch para 100 trens, para a análise do desempenho do código foi gerado dois gráficos, o gráfico exibindo as métricas de erro quadrático médio (MSE) e acurácia binária, fornece uma visão clara do desempenho do modelo durante o treinamento. O MSE, que mede a diferença entre as previsões do modelo e os valores reais, diminui à medida que o modelo aprende, indicando um ajuste cada vez melhor aos dados. A acurácia binária, por sua vez, aumenta com o treinamento, mostrando que o modelo está se tornando mais capaz de classificar corretamente os trens em suas respectivas classes.

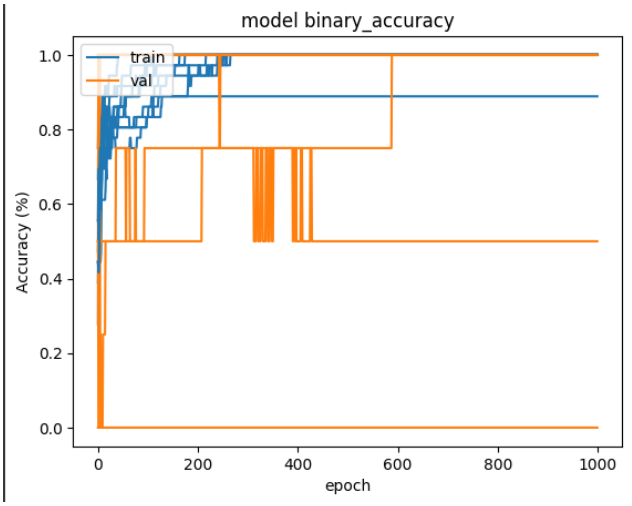
O gráfico apresentado ilustra o processo de aprendizado de um modelo, evidenciando o comportamento

Figure 12: Gráfico das métricas de erro quadrático médio



do erro quadrático médio (MSE) durante o treinamento e a validação. A fase inicial mostra uma rápida redução do MSE tanto para o treinamento quanto para a validação, indicando que o modelo está aprendendo as características dos dados. No entanto, a partir de certo ponto, o MSE de validação tende a se estabilizar ou aumentar, enquanto o MSE de treinamento continua a diminuir. Esse comportamento típico sugere um caso de sobreajuste, onde o modelo memoriza os dados de treinamento em vez de generalizar para novos dados. A linha divisória entre o aprendizado eficaz e o sobreajuste é crucial para determinar o número ideal de épocas de treinamento.

Figure 13: Gráfico das métricas de erro quadrático médio



5.3 Questão 3

Para responder às sub-questões da questão 3, foi crucial a comparação dos dados obtidos na questão 1 e na questão 2.

5.3.1 Questão 3.a

Primeiramente, a arquitetura do modelo Neuro-Simbólico integra a lógica simbólica com redes neurais tradicionais. Essa combinação permite que o modelo utilize a inferência lógica, que é interpretável e baseada em regras explícitas, ao mesmo tempo em que aproveita a capacidade das redes neurais de aprender a partir dos dados fornecidos. No entanto, o LTNTorch apresenta uma abordagem mais avançada na integração entre aprendizado simbólico e redes neurais. Ele utiliza predicados neurais e lógica fuzzy, permitindo que predicados e regras sejam aprendidos diretamente a partir dos dados, com graus de verdade que vão além do simples "verdadeiro" ou "falso".

Enquanto o modelo Neuro-Simbólico tem como principal vantagem a capacidade de incorporar conhecimento pré-existente, guiando o processo de aprendizado e tornando o modelo mais robusto e explicável, o LTNTorch oferece maior flexibilidade. Isso ocorre porque o LTNTorch não depende de regras simbólicas rígidas, podendo aprender representações contínuas e adaptativas que se ajustam melhor aos dados perceptuais. Dessa forma, o LTNTorch se destaca em comparação ao modelo Neuro-Simbólico, que, devido à rigidez das regras simbólicas, pode ter sua flexibilidade limitada para capturar nuances e padrões complexos nos dados.

Em relação aos resultados numéricos, a maior flexibilidade do LTNTorch permitiu uma melhor adaptação aos dados dos trens. Embora haja diferenças entre os modelos em termos de interpretação e acurácia, ambos apresentaram desempenho semelhante nesse aspecto específico. No entanto, de forma geral, o LTNTorch se destacou como o modelo superior. Sua capacidade de aprender representações mais adaptáveis e flexíveis refletiu-se nos resultados, que demonstraram uma menor diferença de desempenho entre os conjuntos de treino e teste.

5.3.2 Questão 3.b

Para o modelo Neuro-Simbólico, a extração seria feita de forma decomposicional, onde o foco é examinar a estrutura interna da rede neural. Aqui, os pesos e as ativações dos neurônios relevantes para a regra em questão seriam inspecionados diretamente. O passo a passo consiste em identificar os neurônios responsáveis pela ativação das características *car(T,C)*, *short(C)* e *closed_top(C)*, em seguida, analisar como esses neurônios se conectam ao neurônio de saída que representa a classificação *east(T)*. Com base nessa análise das ativações e conexões, constrói-se a regra lógica que relaciona essas características à saída desejada.

No caso do LTNTorch, a extração seria realizada de forma pedagógica, tratando a rede como uma "caixa preta". Neste método, a rede seria testada com diferentes entradas para observar os padrões de saída, e assim, extrair as regras a partir das respostas da rede. Novamente, sobre os passos, primeiro seriam geradas amostras de entrada que representassem as características *car(T,C)*, *short(C)*, e *closed_top(C)*. Em seguida, as amostras seriam passadas pelo modelo LTNTorch e observar-se-iam as classificações produzidas. Com base nos resultados obtidos, seria sintetizada uma regra lógica que generalizasse a relação entre essas características e a classificação *east(T)*.

Dessa forma, as regras poderiam variar de acordo com cada modelo. O primeiro modelo apresentado poderia oferecer uma extração mais direta, mas limitada pela rigidez das regras simbólicas, o que levaria a uma regra mais rígida. Já para o segundo modelo, a regra seria mais flexível e adaptável, possivelmente incluindo graus de verdade que permitissem uma interpretação mais suave e menos binária, capturando nuances que o modelo Neuro-Simbólico pode perder. De qualquer forma, a regra extraída deveria ser aplicada a cada um dos trens e comparada com as previsões e classificações reais para avaliar a precisão e a cobertura da regra.

5.3.3 Questão 3.c

Para essa questão, serão analisados os resultados obtidos para as seguintes teorias:

- **Teoria A**: Se um trem tem um vagão curto e fechado, ele vai para o leste; caso contrário, vai para o oeste.
- **Teoria B**: Se um trem tem dois vagões ou um vagão com teto irregular, ele vai para o oeste; caso contrário, vai para o leste.
- **Teoria C**: Se um trem tiver mais de dois tipos diferentes de carga, ele vai para o leste; caso contrário, vai para o oeste.

Com o modelo utilizado na questão "1.2.a", foram obtidos os seguintes resultados para cada regra:

Regra	Resultado
A	[False, True, True]
B	[]
C	[False, True, False, True, False]

Table 1: Resultados obtidos na questão 1.2.a.

Para a Regra A, quando o valor é 'False', isso indica que o modelo não fez a classificação correta para o primeiro trem que deveria ser classificado como "leste" (quando o vagão é curto e fechado). Quando o valor é 'True', isso indica que o modelo fez a classificação correta para os trens subsequentes que deveriam ser classificados como "leste".

A Regra B está vazia, o que significa que não houve trens que atenderam às condições dessa regra (ou seja, não havia trens com dois vagões ou com teto irregular nos dados testados).

E, para a Regra C, quando o valor é 'False', isso indica que o modelo não fez a classificação correta para os trens que deveriam ser classificados como "leste"

(porque tinham mais de dois tipos diferentes de carga). Quando o valor é 'True', isso indica que o modelo fez a classificação correta para os trens subsequentes que deveriam ser classificados como "leste".

Já para o modelo da questão "1.2.b" (Validação Cruzada), os resultados foram os seguintes:

Regra	**Resultado**
A	[True, False, False, False, True, True, False, True]
B	[False]
C	[True, True, True, False, False, False, True, True, False, False, False, False, True, True, True, True, False, False, False, True, False]

Table 2: Resultados obtidos na questão 1.2.b (Validação Cruzada).

Para a Regra A, foram analisados 8 valores de predições, dos quais 5 são verdadeiros, indicando que, em 5 casos, o modelo previu "leste" corretamente quando o trem tinha um vagão curto e fechado.

Na Regra B, apenas 1 valor de predição foi analisado, e este está incorreto, indicando que o modelo não previu "oeste" corretamente para trens com dois vagões ou com teto irregular.

E finalmente na Regra C, foram analisados 21 valores de predições, dos quais 12 são verdadeiros, indicando que, em 12 casos, o modelo previu "leste" corretamente quando o trem tinha mais de dois tipos diferentes de carga.

References

[1] L. De Raedt, R. Manhaeve, S. Dumančić, T. Demeester, and A. Kimmig, "Neuro-symbolic= neural+ logical+ probabilistic," *NeSy'19@ IJCAI, the 14th International Workshop on Neural-Symbolic Learning and Reasoning*, vol. 1, no. 1, pp. 1-10, 2019.

[2] CARRARO, Tommaso. *LTNtorch: PyTorch implementation of Logic Tensor Networks*. Março 2022. DOI:

[3] SUN, Ron. *Connectionist Symbol Processing: Theories and Applications*. Lawrence Erlbaum Associates, 1995.

[4] GARCEZ, Artur d'Avila, LAMB, Luis C., GABBAY, Dov M. *Neural-Symbolic Learning Systems: Foundations and Applications*. Springer, 2002.

[5] NILSSON, Nils J. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann, 1998.

[6] POOLE, David, MACKWORTH, Alan, GOEBEL, Randy. *Computational Intelligence: A Logical Approach*. Oxford University Press, 1998.

[7] OPPENHEIM, Alan V. *Symbolic and Knowledge-Based Signal Processing*. Prentice Hall, 2004.

[8] GARCEZ, Artur d'Avila, LAMB, Luis C., GABBAY, Dov M. *Neural-Symbolic Cognitive Reasoning*. Springer, 2009.

[9] RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning representations by back-propagating errors. *Nature*, 323(6088), 533-536, 1986.

[10] LECUN, Yann; BOTTOU, Léon; BENGIO, Yoshua; HAFFNER, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324, 1998.

[11] KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, pp. 1097-1105, 2012.

[12] SERAFINI, Luciano; GARCEZ, Artur d'Avila. Logic Tensor Networks (LTN).

[13] Zenodo Release of LTNtorch. Disponível em: `https://zenodo.org/record/6394282`.