Classificação de Placas de Trânsito com Redes Neurais para Automação de Veículos

Gustavo Bulka Bonafé Bademian e Celso Ap.de França
Departamento de Engenharia Elétrica da Universidade Federal de São Carlos

***Resumo* – O presente artigo tem como objetivo desenvolver um estudo sobre o funcionamento das Redes Neurais Convolucionais na classificação de sinais de trânsito para uso em carros autônomos, podendo assim ser usado como uma ferramenta de auxílio do veículo no processo de decisão durante a condução, seja por meio do controle da velocidade do veículo, identificação de conversões que o veículo deverá fazer ou até mesmo exibição de informações para o passageiro por meio de um computador de bordo. Para a realização desse projeto foi usado o dataset German Traffic Sign [1]. Se trata de um dataset público usado durante a Conferência Conjunta Internacional de Redes Neurais (IJCNN) no ano de 2011. Ele possui 39209 imagens para treinamento do modelo e 12630 imagens para teste, ambas divididas em 43 classes distintas, cada uma indicando um tipo diferente de sinal de trânsito. Ao final do artigo o leitor terá não somente conhecimento sobre a teoria por trás das redes neurais, como também um embasamento teórico sobre o uso prático das Redes Neurais Convolucionais para a classificação de imagens.**

***Palavra*s-*Chave* – Inteligência Artificial, Redes Neurais, Veículos Autônomos, Aprendizado de Máquinas, Automação, Classificação de Imagens.**

1. INTRODUÇÃO

De acordo com Pereira e Botelho, 2018 **[2]**:"*Calculam-se prejuízos econômicos na escala dos milhões de dólares todos os anos com horas produtivas dos trabalhadores presos em engarrafamentos ou afastados devido a acidentes de trânsito. Do total de acidentes de trânsito registrados, 94% destes são causados simplesmente por erro humano."*

Com o avanço da tecnologia, notícias sobre carros capazes de realizar entregas de forma autônoma ou até mesmo conduzir pessoas sem a necessidade de um motorista para guiar, têm surgido cada vez mais. Muitas empresas, como Uber e Tesla, demonstram cada vez mais interesse em realizar estudos e testes com esse tipo de veículo, o que vem tornando esse mercado cada vez mais difundido na sociedade moderna. Além disso, com o auxílio dessas tecnologias é possível tornar os processos de direção mais seguros, tanto para veículos autônomos quanto os guiados por motoristas, uma vez que o tempo tomada de decisão de um algoritmo pode acabar sendo mais rápido que o do ser humano em casos de perigo, assim se tornanando um meio de combate aos engarrafamentos e acidentes de trânsito.

Com base nesse cenário, este trabalho teve como objetivo o desenvolvimento de um sistema no qual o carro autônomo possa identificar e classificar sinais de trânsito. A partir da classificação desse sistema, o veículo poderá tomar decisões durante o processo de condução, seja no controle da velocidade do carro a partir da identificação de um limite de velocidade, na transferência de uma informação aos passageiros por meio de um computador de bordo e qualquer outro meio de manipulação desses dados que possam beneficiar o processo de condução de um veículo, podendo assim auxiliar no cenário atual de constante crescimento dos acidentes de trânsito e engarrafamentos causados por falhas humanas.

[a] E-mail autor: gustavobademian@estudante.ufscar.br

[b] E-mail orientador:celsofr@ufscar.br

2- TRABALHOS RELACIONADOS

O uso de Redes Neurais Convolucionais no ramo da classificação de imagem já vem sido implementado em diversos projetos ao redor do mundo, se provando uma ótima alternativa para casos que precisam de rápido tempo de resposta. STALLKAMP, SCHILIPSING, SALMEN, e IGEL, autores do banco de dados utilizado neste artigo, em **[3] e [4]** demonstram a capacidade das CNN de superar o tempo de resposta dos humanos, sendo aproximadamente 11.4 ms por imagem classificada

Além disso, diferentes algoritmos implementados ao redor do mundo, também conseguem demonstrar ótimos resultados em tempo de resposta na classsificação de sinais de trânsito, como são os casos de CNN combinada com perceptrons de múltiplas **[5], [6]** e Redes Neurais Densas de múltiplas colunas. **[7]**.

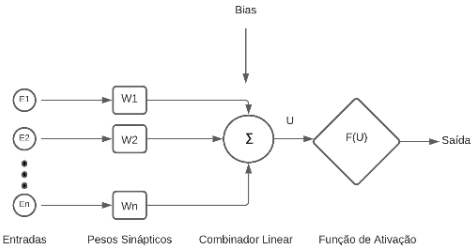
Em conjunto com este artigo, todos os trabalhos citados demonstram a capacidade dos processos de inteligência artificial em tratar o reconhecimento de sinais de trânsito de forma acertiva e rápida, podendo ser usada como uma ótima ferramenta por carros autônomos.

3- FUNDAMENTAÇÃO TEÓRICA

3.1– Neurônio Artificial

Um neurónio artificial, é uma maneira de simular o comportamento de um neurônio real por meio de métodos matemáticos, ou seja, é um meio de simular a transmissão e o nível de atuação dos sinais elétricos transmitidos através das sinapses. A Figura 1 representa o modelo prático de um neurônio artificial utilizado em redes neurais.

Figura 1 – Neurônio artificial



Fonte: Autoria Própria

Em comparação com neurônios reais, as entradas 'E', simulam os impulsos elétricos passados entre os neurônios. Esses sinais podem excitar mais ou menos o neurônio receptor. Esse nível de excitação é denotado pelos pesos sinápticos, assim quanto maior o peso sináptico de uma entrada, maior será sua influência no sinal de saída. O modelo neuronal matemático também conta com a inclusão de uma polarização, conhecida como "Bias", que tem o intuito de aumentar o grau de liberdade do modelo para poder se aproximar ainda mais do comportamento real. O neurônio então, ao possuir os sinais de entrada e os pesos sinápticos, passam por uma combinação linear dada por:

$$\sum_{n=0}^{n} En * Wn$$

A partir dessa combinação o neurônio pode ou não ser ativado dependendo se a combinação dos impulsos que ele recebe ultrapassa, ou não, o seu limiar de excitação, limiar esse também conhecido como "Threshold". Após a combinação linear dos sinais resultando no sinal "U" indicado na Figura 1, o sistema passa por uma função de ativação que tem o objetivo de estabelecer o funcionamento desse Threshold e entregar a saída do modelo, podendo ser caracterizada por diversas funções diferentes como indicado na Tabela 1.

Tabela 1 – Exemplos de funções de ativação

Sigmoide	$\varphi(U) = \frac{1}{1+e^{-U}}$
Gaussiana	$\varphi(U) = e^{-U^2}$
Tangente Hiperbólica	$\varphi(U) = \tanh(U)$
Relu	$\varphi(U) = \max(0, U)$
Softmax	$\varphi(U) = \frac{e^{z*i}}{\sum_{jEgroup} e^{zj}}$

Fonte: Autoria Própria

O estudo nesse documento faz o uso da função de ativação softmax, mostrada acima na Tabela 1. Essa função é comumente utilizada para lidar com problemas de classificação, é caracterizada por ser uma função parecida com a função sigmóide, porém com a capacidade de fazer uma classificação múltipla, enquanto que a função sigmóide apenas é capaz de identificar duas classes apenas, além disso a função softmax faz com que a saída de um neurônio dependa de todos os outros neurônios de saída **[8]**.

A função softmax permite interpretar os valores da camada de saída de uma rede neural, como probabilidades, ou seja, essa função força com que a saída de uma rede neural seja uma probabilidade do grau de pertencimento de uma das classes do problema **[8].**

3.2– Redes Neurais Artificiais (RNA)

A combinação de um ou mais neurônios artificiais é denominada como rede neural artificial. Essas redes podem ser constituídas por uma ou mais camadas ocultas além das camadas de entrada e saída. A principal característica das RNAs, é a capacidade de melhorar o desempenho a partir dos dois diferentes métodos de aprendizagem a seguir:

Aprendizagem supervisionada: O modelo trabalha com uma dupla de dados, caracterizados como dados de entrada e dados de resposta. Uma vez que são apresentados os dados de entrada para a rede neural, após passar por todo o processo matemático dos neurônios o modelo irá obter uma saída. Essa saída é comparada com a resposta desejada, a partir disso é realizado um processo de aprendizagem para ajustar os valores dos pesos sinápticos e do BIAS afim de minimizar a diferença entre o valor obtido e o resultado esperado **[9].**

Aprendizagem não supervisionada: Nesse caso o modelo recebe apenas os dados de entrada e o seu trabalho principal é detectar características em comum entre os dados para agrupá-los **[9]**.

O processo de aprendizagem das RNAs pode ser dado a partir de diferentes métodos, a Tabela 2 tem como finalidade indicar alguns desses principais métodos de aprendizagem.

Tabela 2 – Processo de Aprendizagem

Ajuste de Erro (Regra Delta)	Muito usada em algoritmos de treinamento supervisionado, tem como objetivo, através de vários ciclos, ajustar os pesos sinápticos por meio do erro obtido através da diferença entre o valor de saída da rede e o valor desejado **[10]**.
Hebbiana	Se trata de um ajuste dos pesos sinápticos baseado no postulado de Hebb, em que o peso das conexões é baseado nas atividades conjuntas entre neurônios **[10]**.
Boltzmann	É um método estocástico derivado de conceitos da estatística em que os neurônios podem assumir dois estados (0 ou 1), podendo estar agrupados ou não, sendo responsáveis pela interação com o ambiente **[10]**.
Competitiva	É um método em que os neurônios são expostos aos sinais de entrada e apenas os que obtiveram saída próxima ao valor esperado são ativados enquanto que os outros tem seus pesos sinápticos alterados **[10]**.

Fonte: Autoria Própria

O modelo de rede neural criado neste documento, usa como base o processo de aprendizagem baseado no algoritmo de Adam **[11]**. Esse algoritmo é baseado na descida estocástica do grádiente, porém nesse caso a taxa de aprendizagem ao invés de ser mantida constante por todo o processo ajuste dos pesos sinápticos da rede neural, é mantida uma taxa para cada peso da rede, possibilitando assim a mudança e ajuste ao longo de todo o treinamento do modelo.

Processo de Overfitting

O processo de overfitting é algo que sempre se deve ficar atento assim como o processo de underfitting de um modelo. O Overfitting pode ser definido como o modelo aprende de mais com os dados indicados para o modelo, ou seja, ele se comporta como se tivesse "decorado" as respostas dos dados disponibilizados de tanto treinar, assim ao disponibilizar um novo dado ao modelo ele pode não conseguir identificar devidamente pois esse não foi um dos dados que o modelo "decorou" **[12]**.

Processo de Underfitting

Ao contrário do processo de overfitting o processo de underfitting se dá pelo mau treinamento do modelo, assim ele não consegue ajustar os pesos de forma satisfatória para fazer novas previsões **[12]**.

3.3– Tipos de Redes Neurais Artificiais

3.3.1– Redes Neurais Profundas (Deep Neural Networks)

Esse primeiro tipo de rede neural trata de uma rede neural convencional com múltiplas camadas ocultas podendo ter milhares de neurônios.

O uso de múltiplas camadas ocultas permite com que a rede neural seja possível se adequar a problemas mais complexos. Por exemplo, em um algoritmo de identificação de palavras uma ou algumas camadas poderiam ficar encarregadas de receber os pixels de uma imagem e identificar letras, enquanto que outras camadas subsequentes poderiam ser encarregadas de a partir dessas letras identificar palavras **[13]**.

3.3.2– Redes Neurais Recorrentes (Recurrent Neural Networks)

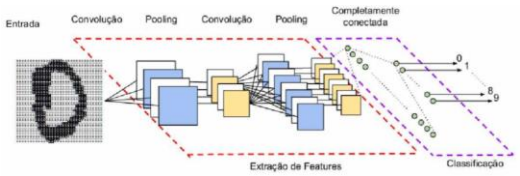
Esse segundo tipo de rede neural tem como ideia principal o uso de informações sequenciais, ou seja, se trata de um modelo de rede neural profunda que leva em consideração o resultado anterior para obter o atual. Esse tipo de rede é comumente utilizado no processo de reconhecimento de linguagem natural, pois o algoritmo ao reconhecer frases é interessante que ele leve em consideração não só as palavras atuais, mas também as anteriores para identificar padrões de formação de frases **[14]**.

3.3.3– Redes Neurais Convolucionais (Convolutional Neural Networks - CNN)

Esse terceiro tipo de rede neural é o tipo de rede utilizado para o propósito desse trabalho, sendo geralmente utilizado nem processos que envolvam visão computacional.

Esse tipo de rede neural é uma variação das redes neurais de múltiplas camadas, tendo sido inspirada no processo biológico de processamento de dados visuais. Esse tipo de rede neural é capaz de aplicar filtros em dados visuais mantendo a relação de vizinhança entre os pixels de uma imagem ao longo de todo o processamento da rede **[15]**. A Figura 2 a seguir exemplifica a arquitetura de uma rede neural convolucional:

Figura 2 – CNN



Fonte: VARGAS,A.C.G; CARVALHO,A.M.P.C; VASCONCELOS,C.N. 2019 **[16]**

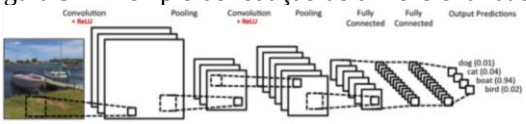
Uma rede neural convolucional é constituída por múltiplas camadas com funções distintas. Geralmente é comum aplicar sobre a camada de entrada dos dados de uma imagem as chamadas camadas de convolução, camada essa constituída por inúmeros neurônios que têm como tarefa aplicar filtros em conjuntos de pixels, assim cada neurônio é conectado diretamente a um conjunto de pixels da camada anterior, para cada uma dessas conexões é atribuído um peso que geralmente é interpretado em forma de matriz, comumente chamado de Kernel ou Máscara **[17]**.

No caso das CNNs, nem todos os neurônios de uma camada então interligados a todos os neurônios da camada anterior como é normalmente feito nas redes neurais mais comuns, nesse caso apenas um subconjunto de entradas é de fato conectado a cada um dos neurônios, a partir disso esse tipo de rede neural passa a realizar análise de campos receptivos locais, ou seja, os neurônios de uma mesma camada passam a ser agrupados em mapas que nada mais são que agrupamento de saídas de neurônios que varrem uma determinada parte da imagem que foram processadas com filtros em comum.

Uma das formas de avaliar o aprendizado desse tipo de rede é verificar as ativações dos filtros obtidos ao fim do treinamento. As primeiras camadas desse tipo de rede costumam utilizar filtros de arestas, bordas e cores que pode e serão utilizadas em conjunto para identificação de parâmetros mais complexos ao longo da imagem **[17]**.

É completamente comum aplicar as funções de ativação como o caso da “Relu”, após cada uma das camadas de convolução com o intuito de tentar gerar algum tipo de linearidade nos dados de saída dos neurônios, uma vez que geralmente os dados de uma imagem não são linearmente separáveis. Outra camada importante geralmente utilizada após as camadas de convolução e ativação, é a chamada camada de Pooling, também conhecida como camada de agrupamento, sua função é reduzir a dimensionalidade dos dados da rede permitindo ao final chegar na saída de classificação desejada [4]. A Figura 3 a seguir mostra um exemplo de classificação de imagem em que é possível observar a redução de dimensionalidade a cada camada de Pooling permitindo chegar em apenas 4 saídas, a partir de inúmeros pixels, ao final do processamento da rede.

Figura 3 – Exemplo de redução de dimensionalidade



Fonte: AMIDI, A.; AMIDI, S, 2018 **[18]**

Além disso é de grande importância a implementação de camadas de dropout, camadas em que serão descartados aleatoriamente uma quantidade definida de neurônios, esse tipo de método visa impedir que ocorra o processo de overfitting em modelos de Redes neurais Convolucionais.

4 METODOLOGIA

4.1 – Aquisição dos dados

Todos os dados foram adquiridos através do dataset German Traffic Sign [2], que se trata de um dataset publico usado durante a Conferência Conjunta Internacional de Redes Neurais (IJCNN) no ano de 2011. O dataset possui 39209 imagens para treinamento do modelo e 12630 imagens para teste, ambas divididas em 43 classes distintas, cada uma indicando um tipo diferente de sinal de trânsito como mostrado na Figura 4.

Figura 4 – Classificação dos sinais

Fonte: STALLKAMP, J; SCHILIPSING, M; SALMEN, J; IGEL, C.2011 **[1]**

4.2 – Especificações do computador usado no desenvolvimento do projeto

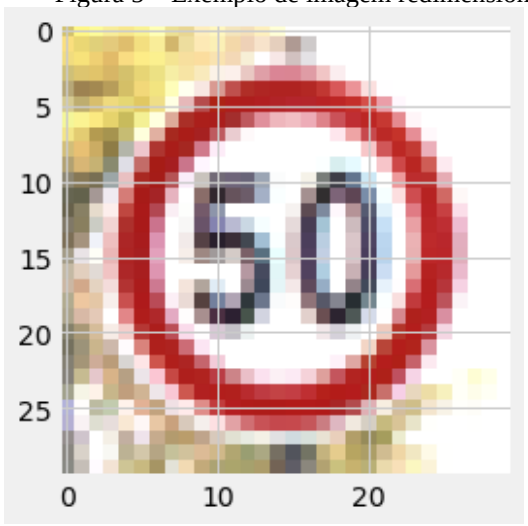
O projeto em questão foi realizado com o uso de linguagem de programação Python 3.7, em seu desenvolvimento foram utilizadas bibliotecas de inteligência artificial como Tensorflow e Scikit Learn. Os experimentos foram realizados com o uso de um processador Intel(R) Core (TM) i5-7300HQ CPU @ 2.50GHz 2.50 GHz, 16 GB de RAM, através do sistema operacional Linux 20.04 LTS – Focal Fossa e com o uso de uma GPU dedicada Nvidia Geforce GTX-1050.

4.3 – Manipulação dos dados

Como as imagens utilizadas estão separadas devidamente em pastas respectivas a cada tipo de placa, foi implementada uma função em python que pudesse mapear e ler todas as imagens através de um arquivo csv com os caminhos.

Como cada imagem é formada por um número diferente de pixels, essa leitura de imagens foi feita em formato matricial com o uso da biblioteca skimage, permitindo que todas as imagens que possuíam quantidade de pixel diferente, fossem redimensionadas e transformadas em arrays, por meio da biblioteca PIL, para um tamanho fixo de 30x30 pixels em padrão RGB, como mostrado na Figura 5 a seguir:

Figura 5 – Exemplo de imagem redimensionada



Fonte: Autoria Própria

4.4 – Parâmetros definidos para a implementação do modelo da CNN

- Input de imagens com altura e largura de 30px.
- Saída: 43 classes equivalentes a cada uma das classes de placas.
- 70% dos dados serão utilizados com o propósito do treinamento da rede neural, ou seja, 31367 imagens.
- 30% dos dados serão utilizados com o propósito de teste do modelo criado, ou seja, 7842 imagens.
- No processo de divisão de treino e teste será usada a função train_test_split do pacote model_selection da biblioteca sklearn, com random state de 42 para que seja possível recriar o mesmo padrão.
- Para uso na rede neural, todos os dados são normalizados .
- Após cada camada de Pooling será implementado um dropout de 25% dos dados para evitar o processo de overfitting no modelo criado e aumentar a acurácia.
- Em cada camada de convolução foi utilizada a função de ativação RELU, uma vez que ela não ativa todos os neurônios das camadas ocultas ao mesmo tempo.
- A compilação do modelo foi feita com o otimizador de adam e perda "categorical_crossentropy", e uso da métrica de acurácia.
- O aprendizado do modelo será baseado no algoritmo de ADAM, com base na função de ativação softmax.
- As camadas de convolução serão definidas por filtros de tamanhos 32 e 64 respectivamente nas primeiras camadas com kernel size de 5 por 5 e 3 por 3 respectivamente.

A seguir na Tabela 3 estão listadas todas as especificações da rede neural criada. A primeira coluna mostra a especificação do tipo de camada da rede sendo camada de convolução (conv2d), maxpooling, dropout, flatten ou dense. A segunda indica a formato de saída de cada camada. A terceira representa o número de parâmetros da camada, ou seja, o número de pesos. Por fim, ao final da tabela é mostrado o número total de parâmetros aprendíveis e não aprendíveis do modelo criado.

Tabela 3 – Especificação da CNN

Model: "sequential"

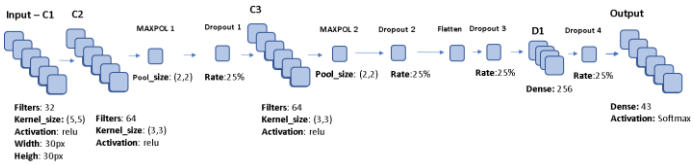
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	2432
conv2d_1 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 12, 12, 64)	0
dropout (Dropout)	(None, 12, 12, 64)	0
conv2d_2 (Conv2D)	(None, 10, 10, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 5, 5, 64)	0
dropout_1 (Dropout)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dense (Dense)	(None, 256)	409856
dropout_2 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 43)	11051

Total params: 478,763
Trainable params: 478,763
Non-trainable params: 0

Fonte: Autoria Própria

A Figura 6 mostra um modelo visual da rede neural criada, sendo possível observar mais informações sobre cada camada contruída a partir do modelo criado na Tabela 3.

Figura 6 – Especificação da CNN



Fonte: Autoria Própria

5 RESULTADOS E DISCUSSÕES

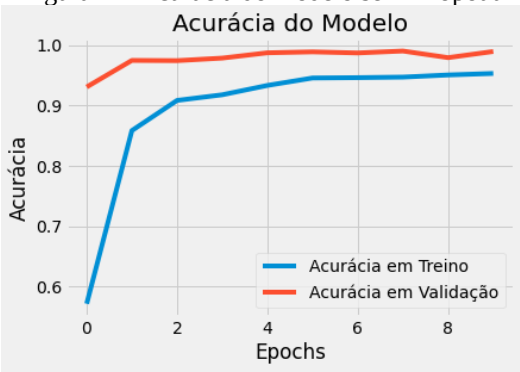
Em um primeiro momento foi realizada a implementação do modelo da rede neural sem utilizar as camadas de dropout, os resultados obtidos podem ser verificados a partir da Tabela 4 e das Figuras 7 e 8.

Tabela 4 – Resultado sem dropout

Epoch	Acurácia
1	0.528262197971344
2	0.8316383361816406
3	0.8931041955947876
4	0.9155800938606262
5	0.9277265667915344
6	0.9400643706321716
7	0.9480664134025574
8	0.9429655075073242
9	0.9523703455924988
10	0.955143928527832

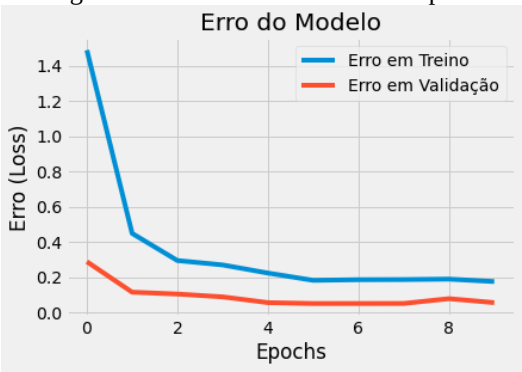
Fonte: Autoria Própria

Figura 7 – Acurácia do modelo sem Dropout



Fonte: Autoria Própria

Figura 8 – Erro do modelo sem Dropout



Fonte: Autoria Própria

Foi inferido nesse primeiro momento um ótimo comportamento da rede neural convolucional no processo de identificação de imagens. Pela análise dos resultados foi possível observar que o modelo teve em média 88% de acurácia, não apresentando overfitting nem underfitting aparente.

Em um segundo momento foram incluídas as camadas de dropout, os resultados desse novo cenário são mostrados a seguir na Tabela 5 e Figuras 9 e 10.

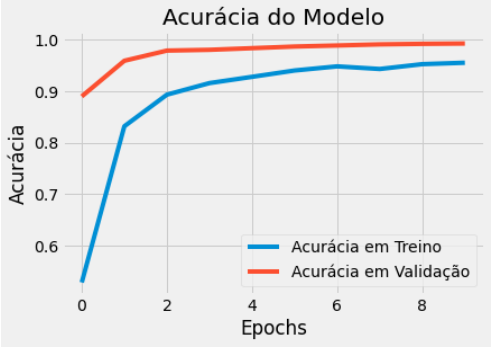
Tabela 5 – Resultado com dropout

Epoch	Acurácia
1	0.7856345772743225
2	0.9716581106185913
3	0.9805209040641785
4	0.9847291707992554
5	0.9846653938293457
6	0.9851754903793335
7	0.9888736605644226
8	0.9900532364845276
9	0.9920936226844788
10	0.9902126193046578

Fonte: Autoria Própria

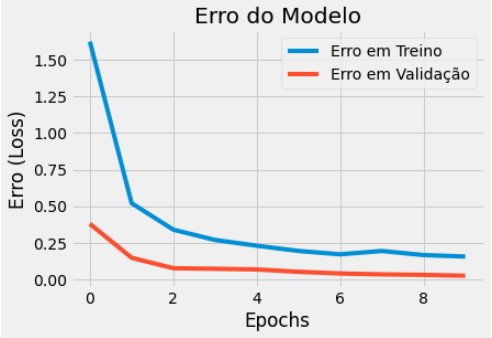
Foi inferido nesse segundo momento um grande crescimento médio dos resultados de acurácia da rede neural convolucional. no processo de identificação de imagens. Pela análise dos resultados foi possível observar que o modelo teve sua acurácia média aumentada para 96,53% aproximadamente, não apresentando overfitting nem underfitting aparente.

Figura 9 – Acurácia do modelo com Dropout



Fonte: Autoria Própria

Figura 10 – Erro do modelo com Dropout



Fonte:Autoria Própria

É notável que a inclusão do processo de dropout após cada uma das camadas de Pooling do modelo fez com que a acurácia do processo aumentasse, isso se dá pois o processo de dropout previne alguns casos de overfitting do modelo, assim permitindo atingir acurácias melhores. A seguir na Tabela 6 existe uma comparação mais detalhada sobre os resultados com e sem as camadas de dropout.

Tabela 6 – Taxa de acerto por classe

Classe	Taxa de Acerto com Dropout	Taxa de Acerto sem Dropout
0	97%	100%
1	98%	97%
2	99%	99%
3	97%	94%
4	96%	93%
5	89%	93%
6	81%	79%
7	98%	91%
8	98%	94%
9	100%	92%
10	99%	98%
11	95%	85%
12	98%	98%
13	100%	99%
14	100%	100%
15	94%	99%
16	99%	98%
17	100%	99%
18	91%	87%
19	100%	95%
20	98%	100%
21	78%	66%
22	96%	92%
23	98%	97%
24	92%	92%
25	94%	94%
26	83%	79%
27	50%	45%
28	98%	97%
29	98%	83%
30	77%	74%
31	98%	97%
32	100%	100%
33	100%	100%
34	99%	99%
35	99%	94%
36	98%	98%
37	100%	92%
38	100%	99%
39	97%	87%
40	98%	71%
41	63%	97%
42	99%	93%

Fonte: Autoria Própria

A partir da Tabela 6, foi possível observar de fato que a acurácia do modelo com dropout foi melhor para a maior parte das classes, caracterizando assim as acurácias maiores mostradas anteriormente na Tabela 5. Porém também é possível notar que esse comportamento não é geral, classes como “20” e “41” tiveram acurácia maior sem as camadas de dropout. Com isso é possível inferir que o processo de dropout de fato pode trazer benefícios gerais ao modelo, porém se for necessário uma visão mais detalhada em uma classe específica, é necessário se atentar sobre a necessidade dessa aplicaçãona classe de estudo.

É possível comparar os resultados deste artigo com os de ZHILONG, ZHONGJUN e ZHIGUO **[19]**, como mostrado a seguir nas Tabelas 7 e 8, sendo identificada uma acurácia próxima em ambos os projetos além de uma breve comparação com outros modelos, tanto em relação com a acurácia, como também em relação ao tamanho do modelo e velocidade do modelo.

Tabela 7 – Comparação de diferentes métodos

Autor	**Método**	**Precisão**
Zheng et al **[19]**	CNN+PCA+LVQ	94,62%
Alex Net **[19]**	CNN	95,90%
GoogLeNet **[19]**	CNN+Inception	96,50%
Zhou et al **[19]**	ReduceV2Net	98,20%
Zhilong et al **[19]**	CNN	99,30%
Modelo proposto neste artigo	CNN	99,2%

Fonte: Alterado de ZHILONG, H; ZHONGJUN, X. ZHIGUO, Y. **[19]**

Tabela 8 – Comparação de diferentes métodos

Autor	Método	Velocidade(ms/imagem)
Ciregan et al **[19]**	MCDNN	11,4
Tang et al **[19]**	SVM+HOG+ Gabor+ LBP	39,8
Li et al **[19]**	ELM+HOG+LBP	2,97
Zhilong et al **[19]**	CNN	0,22
Modelo proposto neste artigo	CNN	0,2090

Fonte: Alterado de ZHILONG, H; ZHONGJUN, X. ZHIGUO, Y. **[19]**

O modelo criado nesse artigo teve velocidade (ms/imagem) de 0,2090, velocidade bem próxima aos resultados de ZHILONG, ZHONGJUN, ZHIGUO **[19]**. Com isso é possível inferir que os modelos criados em ambos os artigos provaram ser uma ótima alternativa no trabalho de identificação de sinais de trânsito, uma vez que um dos fatores mais importantes nesse tipo de trabalho é o tempo de resposta do veículo.

6 Conclusões

Nesse documento foi dada uma visão sobre as redes neurais artificiais, com foco nas Redes Neurais Convolucionais. Também foi demonstrada a incrível eficácia desse tipo de modelo para tarefas de reconhecimento de imagens, uma vez que o modelo chegou a ser 99,2% de acurácia no processo de identificação e classificação da base de dados.

Foi possível observar o processo de overfitting ocorrendo durante a parte prática do projeto e demonstrada uma possível solução/prevenção desse problema. Além disso foi possível identificar que o número de épocas usadas (10 épocas), não se faz totalmente necessário uma vez que a acurária se estabiliza a partir da época de número 5, podendo assim ser diminuído o número de épocas para poupar recursos computacionais .

Em comparação com os outros autores, o modelo criado neste artigo teve um ótimo desempenho, ficando abaixo apenas em precisão em relação ao trabalho de Zhilong et al **[19]**, uma vez que, em velocidade, o modelo proposto conseguiu superar os demais modelos.

Por fim, o projeto por se tratar de uma grande rede neural com milhares de dados faz com que o processo de treinamento do modelo proposto seja lento, assim em um caso comercial em que os recursos computacionais são importantíssimos seria interessante fazer a implementação desse mesmo projeto usando o Apache Spark, permitindo assim que todo o processamento do modelo pudesse ser feito em ambiente distribuído, com clusters de computadores e em memória RAM diminuindo drasticamente o tempo e o uso de recursos computacionais do projeto.

7 Referências

[1] STALLKAMP, J; SCHILIPSING, M; SALMEN, J; IGEL, C. The German Traffic Sign Recognition Benchmark – GTSRB. INSTITUT FÜR NEUROINFORMATIK, 2011.

[2] PEREIRA,S.B.;BOTELHO, R. Design de Interações: fatores humanos e os carros autônomos, PGDESIGN UFRGS. Design & Tecnologia 16,p. 1,2018.

[3] STALLKAMP, J; SCHILIPSING, M; SALMEN, J; IGEL, C. The German traffic sign recognition benchmark: a multi-class classification competition. Proc. IEEE IJCNN, p. 1453-1460, 2011.

[4] STALLKAMP, J; SCHILIPSING, M; SALMEN, J; IGEL, C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. Neural Netw, vol32, no. 0, p 323-332, 2012.

[5] Ciresan,D; Meier, U; Masci, J; Schmidhuber, J. A committee of neural networks for traffic sign classification, Proc. IEEE IJCNN, p. 1918–1921, 2011.

[6] Sermanet, P; LeCun, Y. Traffic sign recognition with multi-scale convolutional networks, Proc. IEEE IJCNN, p. 2809–2813, 2011.

[7] Ciresan,D; Meier, U; Masci, J; Schmidhuber, J., Multi-column deep neural network for traffic sign classification, Neural Netw., vol. 32, p. 333–338, Aug. 2012.

[8] CECCON, D. Funções de ativação: definição,características, e quando usar cada uma, IA EXPERT ACADEMY, Conceitos sobre IA,2020.A

[9] HURWITZ,J; Kirsch, D. Machine Learning for dummies, IBM Limited Edition, p. 15, 2018 B

[10] ROSA, J.L.G. Redes Neurais – SCC-5809. ICMC USP São Carlos, 2011.C

[11] KINGMA,D.P; BA, J.L. Adam: A Method for Stochastic Optimization. University of Amsterdam, University of Toronto, 2015.D

[12] Koehrsen, W. Overfitting vs. Underfitting: A Complete Example. Towards data science, 2018.E

[13] KYRYKOVYCH, A. Deep Neural Networks, KD nuggets, 2020.F

[14] BISWAL, A. Recurrent Neural Network (RNN) Tutorial: Types, Examples, LSTM and More, SimpliLearn, 2021.G

[15] BISWAL, Convolutional Neural Network Tutorial, SimpliLearn, 2021.H

[16] VARGAS,A.C.G; CARVALHO,A.M.P.C; VASCONCELOS,C.N. Um estudo sobre Redes Neurais Convolucionais e sua aplicação em detecção de pedestres, Universidade Federal Fluminense, p1, 2019.i

[17] WOOD, T. Convolutional Neural Network. DeepAI, 2021.J

[18] AMIDI, A.; AMIDI, S. Pense-bête de reseaux 8.K

[19] ZHILONG, H; ZHONGJUN, X. ZHIGUO, Y. Traffic Sign Recognition Based on Convolution Neural Network Model, Qilu University of Technology, 2020.