UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE - UFRN
CENTRO DE TECNOLOGIA - CT
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO - DCA
DISCIPLINA: INTELIGÊNCIA ARTIFICIAL
PROFESSOR: ADRIAO DUARTE DORIA NETO

CLASSIFICAÇÃO DO COMPORTAMENTO DO MOTORISTA A PARTIR DE IMAGENS UTILIZANDO APRENDIZADO DE MÁQUINA PROFUNDO

MORSINALDO DE AZEVEDO MEDEIROS

NATAL - RN
13 DE DEZEMBRO DE 2022

RESUMO

A inteligência artificial e a aprendizagem de máquina profunda permitiram o desenvolvimento de soluções nas mais diversas áreas do conhecimento. Um desses problemas pode ser a classificação de uma ação do motorista enquanto o mesmo está dirigindo, uma vez que ele pode trazer risco para ele e para outras pessoas caso não esteja dirigindo de forma segura. Com isso em vista, treinou-se dois modelos de aprendizado de máquina, um do zero e outro utilizando a técnica de transferência de aprendizado com o intuito de classificar a ação que o motorista está fazendo a partir de imagens. Assim, obteve-se bons resultados com ambos os modelos, mostrando assim aptos para serem colocados em produção e para realizarem classificações no mundo real.

Palavras-chave: Aprendizagem de máquina. Transferência de conhecimento. Segurança. Trânsito.

1. INTRODUÇÃO

O aprendizado de máquina vem auxiliando no desenvolvimento de soluções para diversos problemas enfrentados pelas mais diversas áreas do conhecimento (TRAN et al. (2021)). Umas das subáreas com bastante enfoque nos últimos anos é a de visão computacional, a qual utiliza o aprendizado de máquina para extração de características em imagens. Além disso, um problema bastante recorrente são os acidentes de trânsito gerados por condutores que se distraem fazendo outra atividade enquanto dirigem. Com isso em vista, treinou-se dois modelos de aprendizagem de máquina utilizando imagens obtidas no site do Kaggle (2022) para a classificação do estado do motorista, isto é, se o mesmo está dirigindo de forma segura, se está falando ao telefone etc.

Assim, foram treinadas duas redes: a primeira delas sem a utilização da transferência de aprendizado e a segunda com esta técnica. O modelo pré-treinado utilizado para esta tarefa foi o ResNet-50, por ser um modelo que tende a performar bem com imagens maiores como as utilizadas no *dataset* de treinamento. Uma vez treinados, os dois modelos apresentaram bons resultados, podendo assim ser posto em produção ou embarcado em um microcontrolador em trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 - REDES NEURAIS ARTIFICIAIS

A aprendizagem de máquina é um subcampo da engenharia que surgiu por volta dos anos 60 e que vem se tornando cada vez mais presente nas mais diversas

áreas do conhecimento. Isso foi possível graças aos avanços computacionais e tecnológicos ocorridos nos últimos anos, os quais possibilitaram o treinamento mais rápido de redes maiores e mais complexas. A estrutura básica de uma rede neural é o neurônio, o qual é inspirado no funcionamento do neurônio biológico, de acordo com O'Shea e Nash (2015). Ou seja, os neurônios podem se interconectar entre si (análogo à sinapse) para formar o que se chama de rede neural, onde cada neurônio possui um peso e uma função de ativação. A ativação do neurônio é análogo ao impulso elétrico emitido pelo neurônio biológico e a estrutura básica de uma rede neural pode ser modelada pela figura 1.

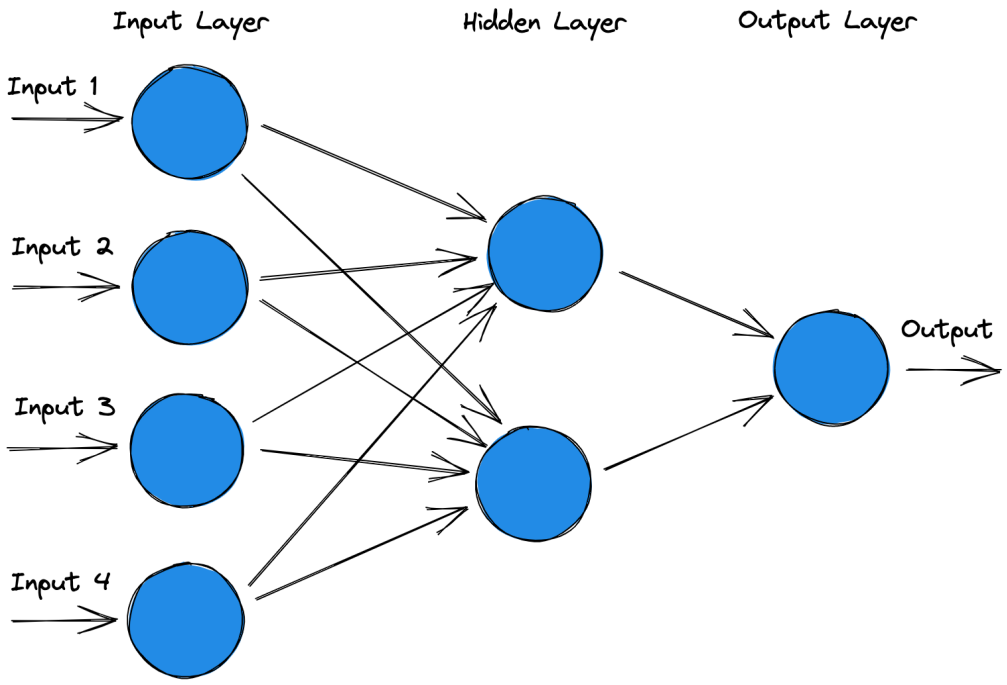

Figura 1 - Rede neural básica. Adaptado de O'Shea e Nash (2015).

A camada de entrada da rede neural recebe um vetor que pode ser multidimensional, o qual é distribuído nas camadas ocultas. Estas últimas, por sua vez, tomam decisões baseadas a partir das camadas anteriores e dos pesos dos neurônios. O que vai determinar se o neurônio vai ser ativado ou não, isto é, se ele contribui com alguma informação relevante para aquela situação, é a função de ativação. A escolha desta função é um hiperparâmetro, ou seja, ela depende do problema e do programador.

Nas camadas ocultas, uma das mais utilizadas é a função Rectified Linear Units (ReLU), proposta Agarap (2019), pois acelera o treinamento e apresenta bons resultados. Já na camada de saída, a escolha vai depender do tipo de problema. Por exemplo, para um problema de classificação multiclasse é utilizada a *softmax*, ao passo que problemas de classificação binária são utilizadas: a função *sigmoid* ou a tangente hiperbólica. Em problemas de regressão, geralmente são utilizadas: a linear ou a ReLU. A figura 2 demonstra os gráficos de algumas funções de ativação.

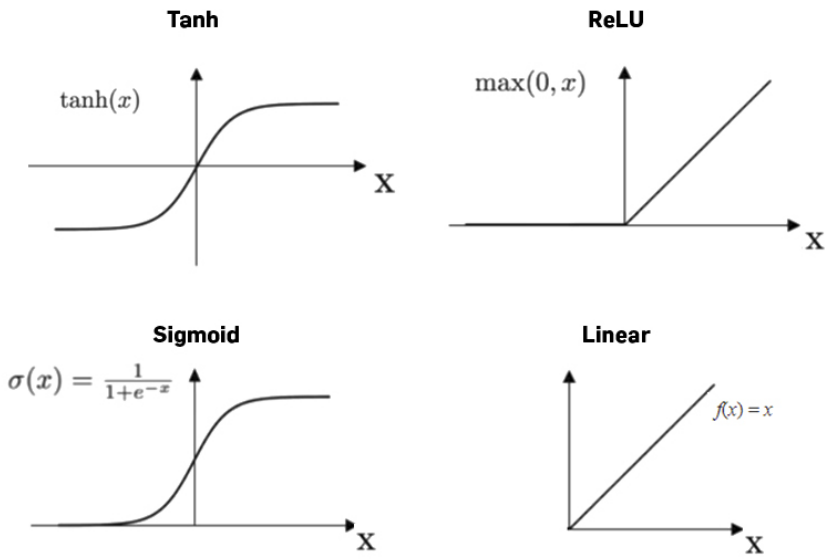

Figura 2 - Funções de ativação. Fonte: AI Wiki (2020)

Uma vez que as contribuições dos neurônios foram calculadas, o erro da rede neural é calculado a partir de uma função custo e, a partir desse erro, o algoritmo *backpropagation* atualiza os pesos dos neurônios. O algoritmo *backpropagation* foi proposto em um artigo lançado originalmente por Rumelhart, Hinton e Williams (1986) e consiste em um processo de ajuste repetitivo dos pesos da rede com objetivo de minimizar a diferença entre o valor previsto pela rede e o valor desejado. Este procedimento foi muito importante para o avanço das redes neurais e é utilizado até os dias de hoje.

Outra técnica bastante utilizada é a do *batch normalization*, proposto por Iofee e Szegedy (2015). A vantagem é que ela acelera o treinamento e deixa-o mais estável, uma vez que ela normaliza as entradas das camadas. Ao longo da definição dos modelos, o *batch normalization* foi utilizado com o intuito de incorporar suas vantagens ao processo de treinamento.

2.2 - REDES NEURAIS CONVOLUCIONAIS

Um dos tipos de rede mais importante atualmente é a rede neural convolucional (CNN), a qual possui a característica de extrair características automaticamente através de camadas convolucionais. Esse tipo de rede é análogo às redes neurais convencionais, porém a diferença encontra-se exatamente na presença das camadas convolucionais, as quais funcionam como filtros que vão passando pela imagem e extraindo as características mais marcantes, como bordas, linhas e contornos, por exemplo. A figura 3 mostra a estrutura básica de uma rede neural convolucional.

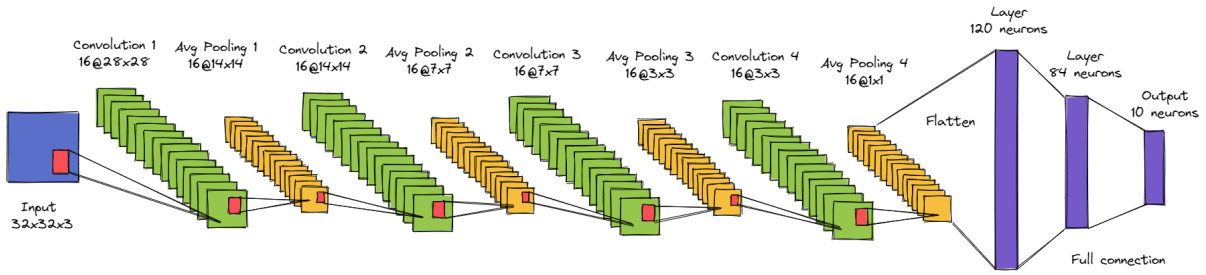

Figura 3 - Estrutura básica de uma CNN. Fonte: O próprio autor (2022).

Além das camadas convolucionais, é possível perceber ainda as camadas de *pooling*, as quais são responsáveis por diminuir o tamanho da imagem com o intuito de diminuir o número de parâmetros. Ao final dos blocos convolucionais e de pooling, há uma camada de *flatten*, a qual é responsável por fazer um achatamento da imagem, isto é, ela transforma a matriz da imagem em um grande vetor. Este vetor é então passado para as camadas *fully-connected*, ou seja, as camadas da rede neural artificial vista na seção 2.1.

A primeira aplicação realizada com sucesso desse tipo de rede foi feita por Yann LeCun et al. (1998) utilizando o dataset MNIST, o qual possui imagens de números escritos à mão. A rede foi chamada de LeNet-5 e possui sete camadas com uma entrada de imagens 32x32 em escala de cinza, chegando uma acurácia de, aproximadamente, 99,2%. Esta técnica também foi muito importante para diversos avanços nesta área e abriu portas para diversas aplicações na área de visão computacional. Um dos avanços que ocorreram ao longo destes últimos anos foi a transferência de aprendizado, a qual será aprofundada na próxima seção.

2.3 - TRANSFERÊNCIA DE APRENDIZADO

A transferência de aprendizado é uma técnica que tem como objetivo pegar um modelo pré-treinado e utilizá-lo para uma segunda tarefa semelhante à primeira, de acordo com Zhuang et al. (2020). Em outras palavras, é possível, por exemplo, pegar um modelo pré-treinado para a classificação de carros, roupas, aviões, navios, entre outros, e treiná-lo para classificar se existe um gato, um cachorro ou um panda em uma determinada imagem.

O processo de treinar a rede para um nova tarefa semelhante à original é conhecido como *fine-tuning*. Para realizá-lo, excluí-se a da cabeça da rede (a camada de classificação original) e adiciona-se uma nova, ao passo que as camadas convolucionais da rede original ficam congeladas, isto é, os pesos dos neurônios não são atualizados. Assim, apenas os pesos dos neurônios da nova cabeça são atualizados para a nova tarefa desejada.

Esta técnica tende a ser bastante vantajosa, uma vez que não é necessário treinar uma rede muito complexa do zero. Isso economiza tempo e gasto de energia, gasto este que pode ser convertido em menos gás carbônico expelido na atmosfera. Com isso, empresas com grandes infraestruturas computacionais como a Google, Facebook, entre outras começaram a disponibilizar gratuitamente modelos pré-treinados para a comunidade.

Isso é muito interessante, pois, geralmente, esses modelos já foram treinados em grandes infraestruturas por muitas horas e tendem a performar bem em diversas aplicações semelhantes para as quais foram treinados originalmente. Assim, a comunidade de aprendizagem de máquina testa estes modelos no desenvolvimento de soluções para os mais diversos problemas.

Um problema bastante comum na nossa sociedade são os acidentes de trânsito causados por distrações ao volante. Por possuir um ritmo de vida acelerado, as pessoas tendem a querer fazer diversas coisas ao mesmo tempo, mesmo enquanto dirigem. No entanto, a distração ao volante pode tanto colocar em risco a vida da própria pessoa quanto das outras pessoas. Com tudo isso em vista, treinou-se dois modelos de aprendizagem com o objetivo de classificar o estado do motorista. O primeiro deles foi treinado do zero e o segundo foi realizado o *fine-tuning* do modelo ResNet50.

A rede ResNet-50 (He et al. (2015)) recebe este nome por utilizar a aprendizagem residual profunda (*deep residual learning*, em inglês), ou seja, a entrada de uma determinada camada é passada para uma das camadas à frente, podendo saltar camadas duplas ou triplas camadas que contém não-linearidades. Essa característica é interessante, pois diminui a probabilidade do desaparecimento do gradiente e faz com que a entrada das camadas posteriores receba a entrada de camadas anteriores. Por isso, escolheu-se esta rede para a transferência de aprendizado para o problema em questão.

Por fim, uma coisa muito importante a ser mencionada é a preocupação com o meio ambiente, pois o treinamento de redes neurais, geralmente, é feito utilizando as GPUs. A utilização desse tipo de componente é feita, pois elas aceleram o processo de treinamento, porém elas gastam muita energia para poder realizar todos os cálculos e gasto de energia significa emissão de CO_2 na atmosfera.

Com isso em vista, mensurou-se o consumo de energia para treinar os dois modelos individualmente. Para isso, utilizou-se uma biblioteca chamada de Code Carbon (2019), a qual realiza a mensuração da quantidade de energia total que foi gasto, o gasto de memória RAM, de CPU e de GPU, bem como a emissão de CO_2 na atmosfera. Segundo a documentação do repositório no Github, o cálculo da emissão de CO_2 é feito levando em consideração a fonte de geração da energia consumida. Assim, mesmo que um modelo consuma mais energia do que outro

para ser treinado, a emissão de CO_2 pode ser menor no caso da fonte da energia ser mais limpa.

3. APLICAÇÃO

3.1 CONJUNTO DE DADOS

O conjunto de dados (*dataset*) foi obtido na plataforma Kaggle (2022), a qual possui milhares de dados públicos para o treinamento de modelos de inteligência artificial, bem como possui competições desafiadoras com premiações para as equipes participantes. O nome do dataset é *State Farm Distracted Driver Detection* e possui 22424 imagens, as quais foram divididas numa proporção de 80% para treinamento e 20% para teste. Além disso, o dataset possui dez classes, as quais estão demonstradas na figura 4 e relacionadas na tabela 1.

Classe	**Label**
c0	Safe Driving
c1	Texting - Right
c2	Talking on the phone - Right
c3	Texting - Left
c4	Talking on the phone - Left
c5	Operating the radio
c6	Drinking
c7	Reaching behind
c8	Hair and makeup
c9	Talking to passenger

Tabela 1 - Classes do conjunto de dados. Fonte: O próprio autor (2022)

Ao todo, foram coletadas imagens de 26 motoristas diferentes, cada um realizando as atividades referentes às 10 classes. Assim, a figura 5 mostra a quantidade de imagens de cada classe coletada por cada motorista. Uma vez que os dados estão preparados, segue-se para a etapa de preparação e treinamento dos modelos.

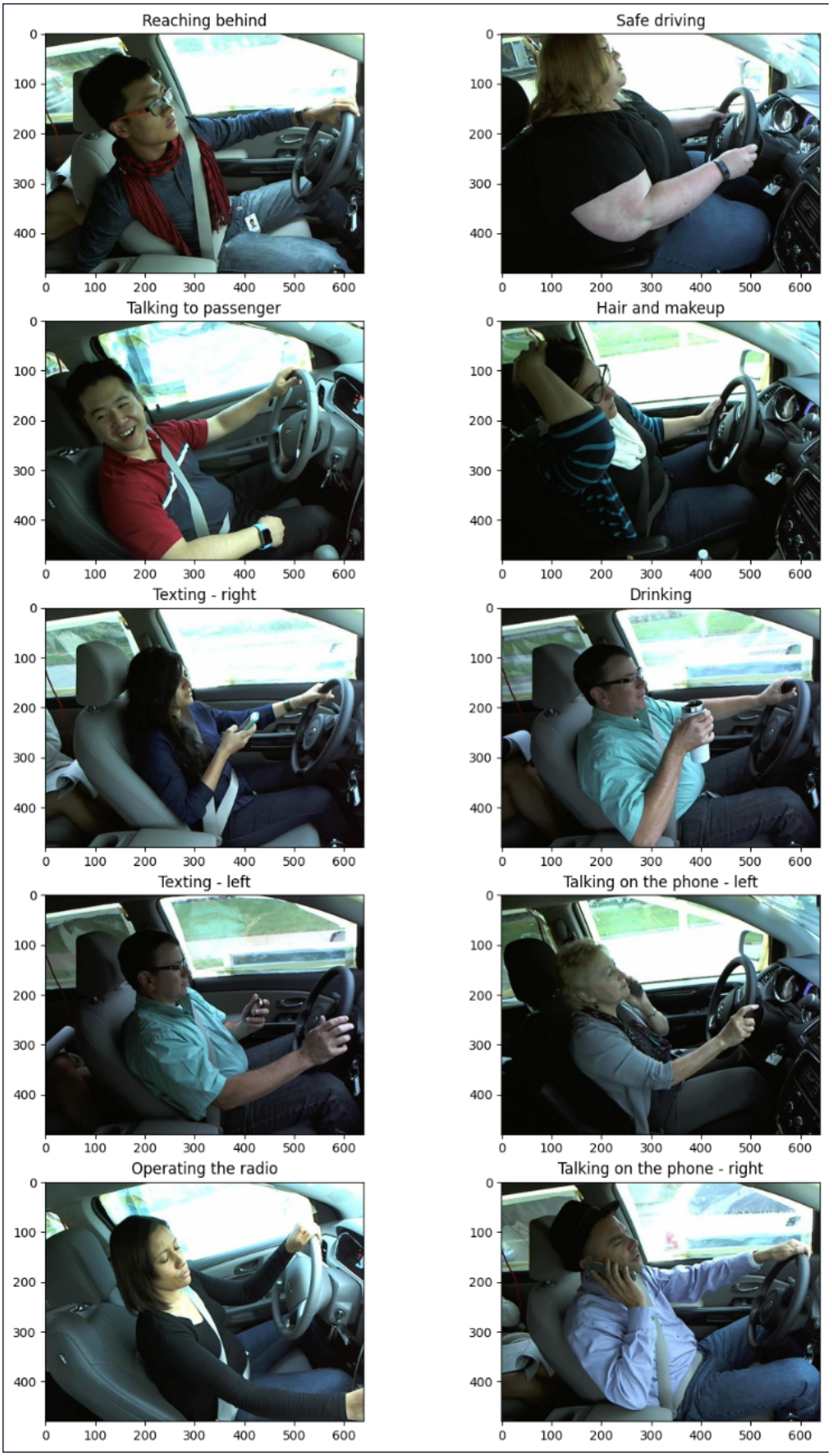

Figura 4 - Exemplos das classes do dataset. Fonte: O próprio autor (2022).

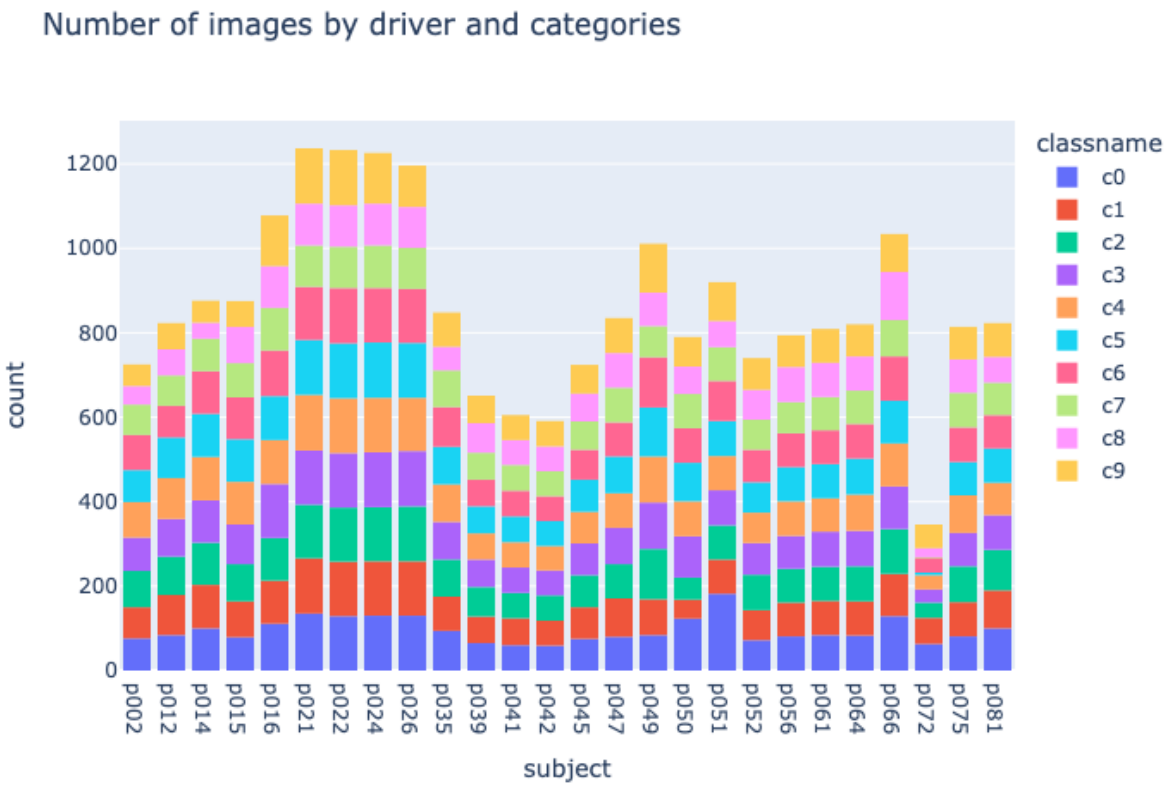

Figura 5 - Quantidade de imagens por cada motorista. Fonte: O próprio autor (2022).

3.2 MODELOS

O primeiro modelo, treinado do zero, possui cinco blocos de camadas convolucionais sendo composta por: a camada convolucional com função de ativação ReLU e um *batch normalization*. O primeiro bloco possui um filtro na camada convolucional de tamanho igual a 32, ao passo que o segundo e o terceiro bloco possuem 64 filtros e o quarto e o quinto bloco possuem 128 filtros. Além disso, foi colocada ainda uma camada de *Max Pooling* entre o primeiro e o segundo bloco, entre o terceiro e o quarto e após o quinto, antes da camada de *Flatten*.

Após a camada de *Flatten*, foram colocadas uma camada Densa com 1024 neurônios, função de ativação ReLU e *batch normalization*, seguida pela camada de saída com dez neurônios e função de ativação *softmax*, por ser um problema de classificação multiclasse. Ao ser feita uma sumarização do modelo, observou-se que o mesmo possui cerca de 58 milhões de parâmetros para serem treinados e aprendidos. A arquitetura utilizada para este modelo foi obtida a partir de várias experiências visando obter um modelo que apresentasse uma boa performance e ela pode ser encontrada no repositório do Github [REF].

Como mencionado anteriormente, utilizou-se a técnica de transferência de aprendizado para o treinamento do segundo modelo, o qual foi baseado no

ResNet50. Assim, foi importado o modelo excluindo-se a cabeça da rede e inicializando-se os pesos treinandos a partir do ImageNet. Para a nova cabeça, foi utilizado uma camada de *Global Average Pooling* após a saída da ResNet, seguida por três camadas densas, sendo a primeira delas com 512 neurônios, a segunda com 256 e a última com 10 neurônios e função de ativação *softmax*. A função de ativação utilizada nas duas primeiras camadas densas foi a ReLU.

A sumarização do modelo mostrou que, ao final, esta nova rede possui cerca de 24 milhões de parâmetros, ou seja, ela têm menos parâmetros do que a primeira rede mostrada. Isso foi feito com o intuito de verificar o quão impactante isso é no tempo de treinamento e na acurácia no modelo.

3.3 TREINAMENTO

Uma vez que os modelos foram definidos, seguiu-se para a etapa de treinamento, a qual foi realizada utilizando o Google Collaboratory. Ambos os modelos foram treinados por sete épocas utilizando o otimizador Adam e a função custo sendo a entropia cruzada categórica esparsa. Além disso, foi utilizada a técnica de *data augmentation* com o objetivo de inserir ruído durante o treinamento e melhorar a generalização do modelo na base de teste. As figuras 6 e 7 mostram os gráficos de acurácia e de erro para o primeiro modelo e para o segundo, respectivamente.

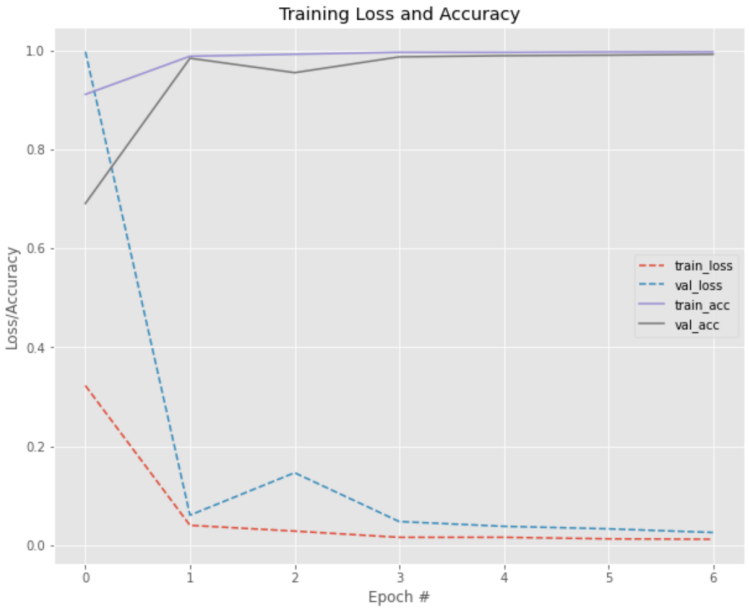

Figura 6 - Acurácia e erro do modelo baseline

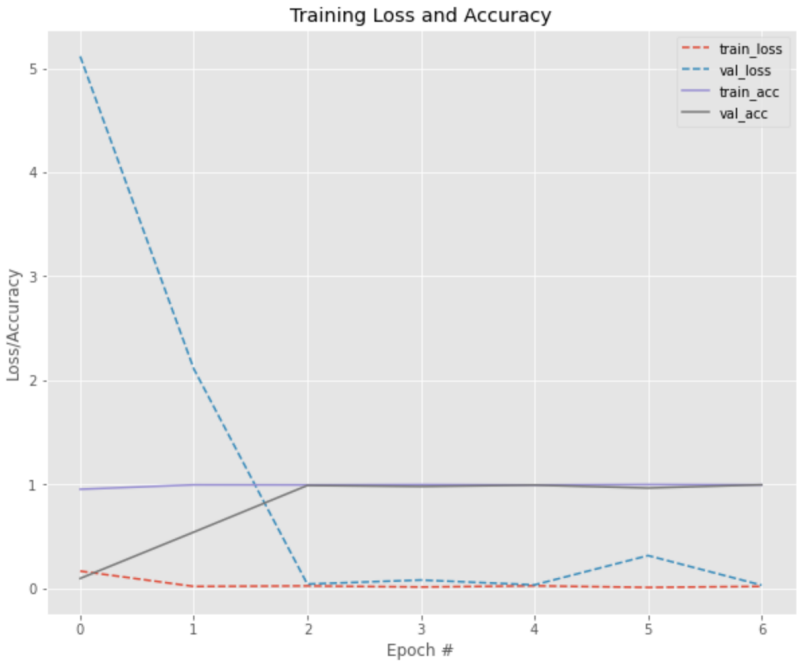

Figura 7 - Acurácia e erro do modelo RenNet-50

É possível perceber que tanto o custo quanto a acurácia de ambos os modelos melhoraram rápido ao longo de poucas épocas. Além disso, com a biblioteca Code Carbon, foi possível perceber que, apesar do modelo baseline ter consumido mais energia para ser treinado, o treinamento dele resultou em menos emissão de CO_2, como mostra a tabela 2.

Modelo	Energia (KWh)	Emissão de CO_2 (Kg)
Baseline	0.075479	0.047626
ResNet-50	0.098735	0.012736

Tabela 2 - Gasto de energia e emissões de carbono do treinamento

Isso pode ter acontecido pelo fato da biblioteca conseguir identificar o tipo de geração de energia utilizado no treinamento. Assim, é possível que o segundo modelo tenha sido treinado com uma fonte de energia mais limpa e/ou renovável, por isso, o treinamento deste emitiu menos CO_2 na atmosfera. É importante destacar que o caderno com todas as informações e códigos utilizados encontram-se no repositório do Github.

3.4 - VALIDAÇÃO

Devido ao formato de codificação das imagens não conseguiu-se gerar a matriz de confusão para os modelos. Assim, decidiu-se iterar sobre o dataset de

treinamento, realizar a predição de uma imagem por vez e verificar se o modelo acertou ou não. Com isso obteve-se os resultados mostrados na tabela 3:

Modelo	Acertos	Erros	Acurácia
Baseline	4442	42	99.06%
ResNet-50	4457	27	99.40%

Tabela 3 - Resultado da validação dos modelos

Com isso, é possível perceber que o modelo ResNet-50 performou um pouco melhor do que o modelo baseline, mostrando assim o poder desta técnica. Além disso, foi o modelo que menos contribuiu para a emissão de CO_2 na atmosfera e isso é muito importante, pois é fundamental que haja essa preocupação com o meio ambiente.

CONCLUSÕES

Com isso, foi possível perceber que a transferência de aprendizado é uma técnica bastante poderosa no contexto de aprendizagem de máquina. Assim, os objetivos deste trabalho foram alcançados, uma vez que os modelos treinados obtiveram excelentes resultados e pode-se comparar a diferença de performance e a diferença nos custos de treinamento para os dois modelos.

Com este estudo, foi possível ainda fixar vários conceitos básicos e importantes que compõem o mundo da inteligência artificial, além de poder aplicar este conhecimento no desenvolvimento de uma solução para um problema real. Este processo de aprendizagem é muito importante para a formação dos discentes enquanto futuros engenheiros, uma vez que o objetivo da formação é ensinar as ferramentas para que se utilize na solução de problemas.

Uma vez que os modelos estão treinados, os próximos passos envolvem a colocação destes modelos em produção para que eles possam ser utilizados na vida real. Além disso, é possível ainda tentar embarcar em um microcontrolador com o intuito de utilizá-lo em um hardware com limitações de processamento, memória e energia para se observar a performance dele.

REFERÊNCIAS

AGARAP, Abien F. M. Deep learning using Rectified Linear Units (ReLU). arXiv:1803.08375v2, p. 1-7, 7 Fev. 2019.

AI WIKI. Activation Function, 2020. Disponível em: https://machine-learning.paperspace.com/wiki/activation-function. Acesso em: 3 dez. 2022.

Code Carbon, 2019. Disponível em: https://github.com/mlco2/codecarbon. Acesso em: 3 dez. 2022.

Github. Artificial Intelligence, 2022. Disponível em: . Acesso em:

HE, Kaiming et al. Deep Residual Learning for Image Recognition. arXiv:1512.03385v1. p. 1-12, 10 Dez. 2015.

IOFFE, Sergey; SZEGEDY, Christian. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv: 1502.03167v3, p. 1-11, 2 Mar. 2015.

Kaggle. State Farm Distracted Driver Detection, 2022. Disponível em: https://www.kaggle.com/c/state-farm-distracted-driver-detection/data. Acesso em: 3 dez. 2022.

LECUN, Yann et al. Gradient-Based Learning Applied to Document Recognition. IEEE. v. 1, p. 1-46, Nov. 1998.

O'SHEA, Keiron; NASH, Ryan. An Introduction to Convolutional Neural Networks. arXiv:1511.08458v2, p. 1-11, 2 Dez. 2015.

RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning representations by back-propagation errors. Letters to Nature. v. 323, p. 533-536, 9 out. 1986.

TRAN, Sunny et al. Solving Machine Learning Problems. arXiv:2107.01238v1, p. 1-38, 2 Jul. 2021.

ZHUANG, Fuzhen. A Comprehensive Survey on Transfer Learning. arXiv:1911.02685v3. p. 1-31, 23 Jun. 2020.