MODELO PARA RECONHECIMENTO DE IMAGENS: Construindo e Treinando uma Rede Neural Convolucional (CNN) com Keras e Tensorflow

IMAGE RECOGNITION CLASSIFIER: Building and Training a Convolutional Neural Network (CNN) with Keras and Tensorflow.

Franco, Gabriela Cristina Moreira
Silva, Renato Moraes
gcristina.franco@gmail.com, renato.silva@facens.br

Centro Universitário Facens – Sorocaba, SP, Brasil

Submetido em: 11/03/2023 . Aceito em:_____.

RESUMO

A linguagem brasileira de sinais é reconhecida legalmente como a segunda língua oficial do Brasil, porém poucas são as pessoas que conhecem essa forma de expressão ou possuem domínio na linguagem de sinais. Nesse artigo, é proposto a criação de uma rede neural convolucional com o auxílio de algumas bibliotecas de aprendizado de máquina para que, assim, seja possível treinar e validar um modelo capaz de categorizar as imagens estáticas de entrada em um dos 20 sinais mapeados. Ao final, discutindo o resultado obtido através da curva de assertividade versus perda e apresentando considerações futuras de implementação.

Palavras-chave: Língua Brasileira de sinais, Reconhecimento de imagens, Redes Neurais Convolucionais, Processamento de imagens, Acessibilidade.

ABSTRACT

Brazilian sign language is legally recognized as the second official language of Brazil, but few people know this form of expression or master sign language. In this article, it is proposed the creation of a convolutional neural network with the help of some machine learning libraries so that, thus, it is possible to train and validate a model capable of categorizing the input static images in one of the 20 mapped signals. At the end, discussing the result obtained through the curve of assertiveness versus loss and presenting future implementation considerations.

Keywords: Brazilian sign language, Image recognition, Convolutional Neural Networks, Image processing, Accessibility.

1. INTRODUÇÃO

Reconhecimento de imagem é uma das tarefas mais populares da área de Aprendizado de Máquina, permitindo que computadores compreendam os elementos visuais que são recebidos por ele de uma forma a solucionar problemas relacionados a entendimento por máquinas (GONZALEZ; WOODS, 2000).

Tarefas que antes só podiam ser realizadas por humanos devido à necessidade de interpretação, agora, podem ser realizadas por máquinas permitindo-as aprender com os erros e aprimorar suas decisões diminuindo erros e ambiguidades (SOUZA; SAMOHYL; MIRANDA, 2008).

Existem diversas utilizações para o reconhecimento de imagens computacionais e muitas delas já fazem parte do dia a dia das pessoas. Algumas dessas utilizações podem ser encontradas no agrupamento de imagens dos álbuns do seu celular (SILVA; SANTOS JUNIOR, 2021), reconhecimento de placas de automóveis (CONCI; MONTEIRO, 2004) e localização de padrões caracterizados por doenças e de anomalias em radiografias (SIQUEIRA, 2010), entre outras infinitas possibilidades.

Enquanto resultados muito satisfatórios já podem ser observados na tradução de textos, o reconhecimento automático e tradução de línguas gestuais está em um estado de desenvolvimento (DIGIAMPIETRI; TEODORO; SANTIAGO; OLIVEIRA; ARAUJO, 2012). Com isso, essa interação gestual acaba se limitando a pessoas que a conhecem fazendo com que, por exemplo, surdos que utilizem a linguagem de sinais tenham sua a inclusão social seriamente afetada, pois ele não é capaz de se fazer entender (SANTOS, 2015).

Através de algoritmos de aprendizagem profundo, como a rede neural convolucional (CNN), é possível, de forma semelhante ao cérebro humano, conduzir a imagem através de neurônios responsáveis por pequenas regiões da imagem, passando por filtros de pesos, relações entre as pequenas regiões da camada atual e preparação de mapa de características condensadas (ACADEMY, 2022).

A motivação desse artigo consiste no reconhecimento de Libras (Linguagem brasileira de sinais) por meio da aplicação de um algoritmo de aprendizagem profundo, CNN, que irá classificar elementos disponibilizadas em um compacto banco de imagens gerado especialmente para esse artigo uma vez que esse tipo de banco de dados não é encontrado facilmente em meios digitais. Sendo assim, essa base de dados é usada neste trabalho tanto para treinar um modelo de reconhecimento de sinais de Libras, quanto para avaliar sua assertividade.

Esse artigo é organizado em cinco seções. A 2 dois apresenta trabalhos similares a esse artigo apresentado as diferenças e pluralidade em que o tema pode ser abordado. A Seção 3 descreve todo os passos realizados para a geração do banco de imagens (conjunto de dados), construção da rede neural, treinamento e validação do modelo. A Seção 4 descreve os resultados obtidos. A seção 5 descreve as possíveis tarefas futuras que podem ser desenvolvidas a partir das contribuições fornecidas por este projeto.

2. TRABALHOS CORRELATOS

Três trabalhos correlatos foram analisados. O primeiro deles usa o sensor Kinect® para capturar informações como posição, velocidade e orientação da mão para classificação. Em outro o reconhecimento é realizado através da posição da mão em vídeos curtos. Por fim o terceiro trabalho utiliza-se de imagens estáticas coletadas do Laboratório de Pesquisas Inteligentes e Cognitivos da Universidade Estadual de Feira de Santana.

A maioria dos trabalhos correlatos focam em reconhecer os gestos através sensores que identificam os eixos das mãos ou reconhecimento, como foi o caso de Santos (2015) em que o método empregado por ele foi o sensor de profundidade Kinect da empresa Microsoft e Dias, Souza e Pistori (2006) em que houve a extração de informações dos gestos em gravações de vídeo.

Ainda assim, as redes neurais são extensamente utilizadas como foi o caso avaliado por Caiafa, Fonseca, Lima, Araujo e Silva (2012) em que analisou o resultado obtido pelo treinamento do reconhecimento entre cinco diferentes redes sendo elas: ResNet50, InceptionV3, VGG16, VGG19 e LeNet concluindo que, independentemente da rede neural aplicada, as arquiteturas de aprendizagem profundo obtém resultado satisfatório para a classificação de sinais estáticos em Libras.

De forma semelhante a outros trabalhos, esse artigo utiliza-se de redes neurais para classificar as imagens e armazenar os pesos com melhor avaliação para que possam ser utilizados, posteriormente, em diversas aplicações para o reconhecimento dos gestos em Libras.

3. METODOLOGIA

Esse estudo se concentra na construção e treinamento de um modelo utilizando-se de CNN implementada por meio de bibliotecas como Keras e Tensorflow. Considerando um conjunto de dados formado por imagens que representam letras do alfabeto da linguagem oficial de sinais brasileira, também conhecida como Libras.

Os resultados são avaliados através de métricas que servirão de indicativos do quão qualidade efetiva da aprendizagem do modelo ao longo de seu treinamento até a sua possível utilização em outras finalidades tecnológicas. Detalhes do conjunto de dados, métodos empregados e dos resultados obtidos são apresentados nos subtópicos a seguir.

3.1. Base de Dados

Libras, assim como a língua falada portuguesa, é representada por uma diversidade de palavras e simbologias. Porém, ao invés de serem expressas sonoramente, são expressas através dos gestos e sinais executados pelas mãos do locutor, podendo conter movimentações ou não com as mãos além de expressões faciais.

O conjunto de dados utilizado nesse estudo foi inteiramente reunido e gerado por autoria própria contendo um total de 1.600 imagens separadas em vinte grupos diferentes possuindo natureza categórica. Para maior diversidade, as imagens foram

coletadas em três dias diferentes para que houvesse variedade de fundos e luminosidade nas imagens. Também foram coletadas imagens em dois dispositivos diferentes garantindo assim qualidade de imagens diferentes, respeitando sempre a dimensão de 640px de largura e 480px de altura. Esse tamanho de imagem foi usado pois é o tamanho gerado pelo dispositivo de menor resolução na orientação horizontal.

A quantidade de imagens separadas foi de 80% para treino e 20% para teste. Assim, do total de 80 imagens por categoria, 64 são designadas para treino e 16 para teste do modelo.

Esse valor foi definido devido ao tamanho enxuto do conjunto de dados, mas vale lembrar que não existe uma regra definida para a quantidade exata das separações ideal, assim cada caso deve ser analisando segundo a sua finalidade de estudo.

Quadro 1 – Categorias consideradas em conjunto de dados.

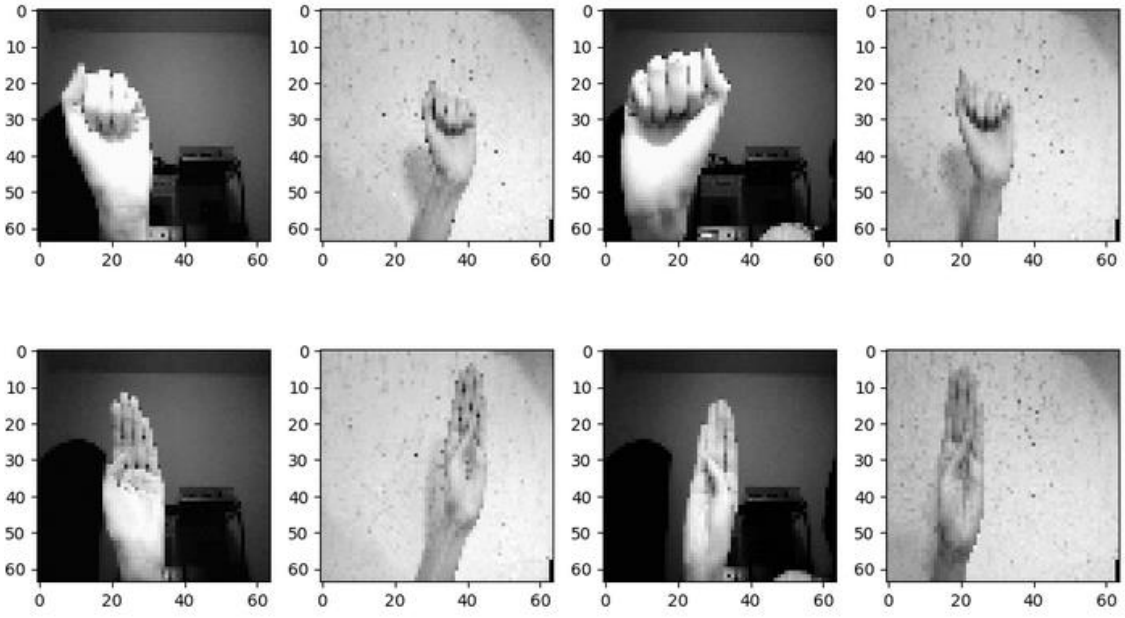
Grupo	LETRA DO ALFABETO	QTD. IMAGENS TOTAL	QTD. IMAGENS TREINO	QTD. IMAGENS TESTE
1	A	80	64	16
2	B	80	64	16
3	C	80	64	16
4	D	80	64	16
5	E	80	64	16
6	F	80	64	16
7	G	80	64	16
8	I	80	64	16
9	L	80	64	16
10	M	80	64	16

Grupo	LETRA DO ALFABETO	QTD. IMAGENS TOTAL	QTD. IMAGENS TREINO	QTD. IMAGENS TESTE
11	N	80	64	16
12	O	80	64	16
13	P	80	64	16
14	Q	80	64	16
15	R	80	64	16
16	S	80	64	16
17	T	80	64	16
18	U	80	64	16
19	V	80	64	16
20	W	80	64	16

Fonte: Elaborado pelo autor.

Com o conjunto de dados finalizado, foi possível gerar uma amostragem de dados já considerando um pré-processamento básico, como: a coloração em escala de cinza e o dimensionamento das imagens que serão trabalhadas.

Figura 1 - Amostra de conjunto de dados.



Fonte: Elaborado pelo autor.

3.2. Gerando Lotes de Treinamento e Validação

Há inúmeras combinações de parâmetros que podem ser utilizadas para gerar e padronizar o lote de treinamento. Nessa etapa, deve ser escolhido os parâmetros com base no conjunto de dados, porém nada impede que mudanças e experimentações sejam necessárias. Ao contrário do que se pode pensar, não há uma fórmula fixa e mágica para essa etapa. Assim, a experimentação e testes serão bem frequentes até que se atinja a geração de um modelo treinado com qualidade.

Para esse estudo foi utilizada uma função disponível na biblioteca Keras, chamada "*ImageDataGenerator*", com o argumento de inverter ocasionalmente imagens no eixo horizontal, garantindo que imagens "espelhadas" também façam parte do lote de treinamento, ou seja, realizando o aumento de dados (*data argumentation*).

As técnicas de aumento de dados são importantes para casos em que o conjunto de dados inicial é muito pequeno, deseja-se aprimorar a acuracidade do modelo ou até mesmo prevenir que seu modelo sobre ajuste (*overfitting*). Há diversas técnicas para o aumento de dados. Para o reconhecimento de imagens a mais populares são: junção e embaralhamento de imagens criando uma imagem a partir de outras imagens, removendo partes da imagem, adicionando efeitos de borrão, troca de cores, entre outra (AWAN, 2022). Nesse caso foi escolhida o espelhamento das imagens para que o treinamento não fique muito moroso.

Assim, com o gerador de amostra de dados criado, é realizada a chamada da função "*flow_from_directory*" que irá carregar amostras de imagens de cada categoria do conjunto de dados. Nesse momento, aproveita-se para padronizar as imagens coletadas em caso de existir alguma inconsistência entre elas. Algumas das padronizações aplicadas foram: padronização das dimensões das imagens em 150px por 150px mantendo a mesma dimensão de altura e largura e para que seu processamento seja mais ágil; coloração em escala de cinza; definição do número de imagens a serem utilizadas para o treino; definição do tipo de lista de rótulos que devem ser retornados (nesse caso, categóricos); e embaralhamento das imagens que serão utilizadas no treinamento.

Para o gerador de validação, as mesmas configurações foram utilizadas sendo a única diferença o diretório em que as imagens são coletadas. Para o caso de treinamento, as imagens são coletadas da separação *train* enquanto para a validação, as imagens são coletadas da separação *test*. Abaixo quadro com a configurações utilizadas para o treinamento e validação do modelo.

Figura 2 – Treinamento e validação utilizados.

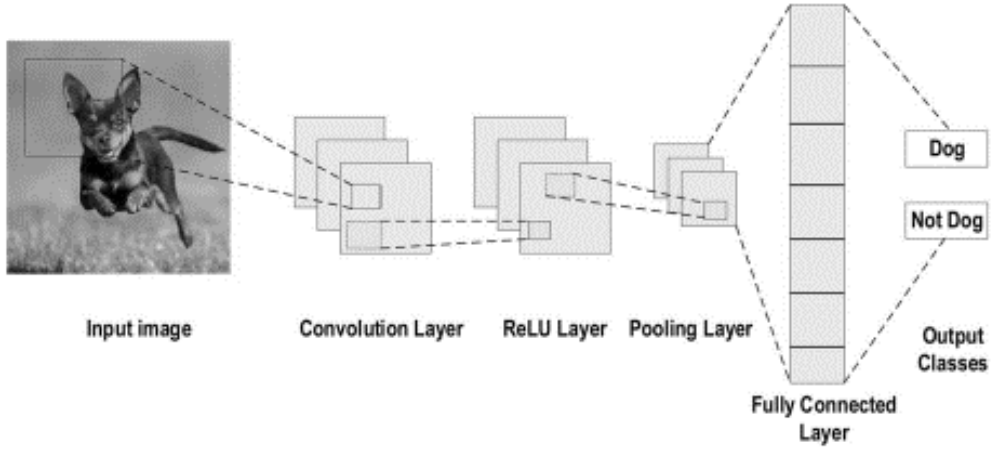
```
img_size = 150
batch_size_train = 64
batch_size_validation = 16

datagen_train = ImageDataGenerator(horizontal_flip=True)

train_generator = datagen_train.flow_from_directory("train/",
                                                    target_size=(img_size,img_size),
                                                    color_mode="grayscale",
                                                    batch_size=batch_size_train,
                                                    class_mode='categorical',
                                                    shuffle=True)

datagen_validation = ImageDataGenerator(horizontal_flip=True)
validation_generator = datagen_validation.flow_from_directory("test/",
                                                    target_size=(img_size,img_size),
                                                    color_mode="grayscale",
                                                    batch_size=batch_size_validation,
                                                    class_mode='categorical',
                                                    shuffle=False)
```

Fonte: Elaborado pelo autor.

3.3. Criando Modelo de Rede Neural Convolucional (CNN)

Segundo (Alzubaidi, 2021), comumente a CNN utiliza-se de várias camadas de convolução, seguidas por camadas de subamostragem (*pooling*), enquanto as camadas finais são chamadas de camadas totalmente conectadas ou densas (*fully connected* ou *dense layers*). As pontuações de classificação são geradas através da camada final de saída. Assim, para cada determinada entrada existe uma pontuação que representa a probabilidade de uma classe especifica do conjunto de dados.

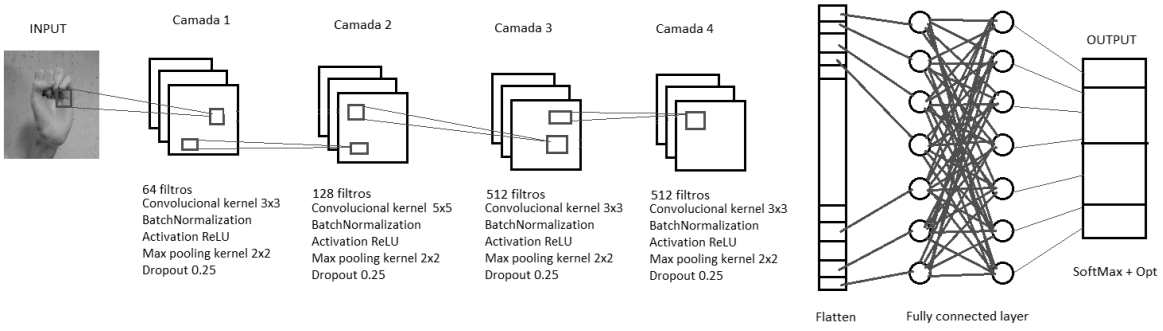
Figura 3 - Arquitetura da rede neural convolucional.



Fonte: ALZUBAIDI (2021, p. 14).

Seguindo esses conceitos, foi criada uma rede neural contendo 4 blocos de filtros convolucionais, a utilização da função de ativação não linear “ReLu” e 2 camadas densas finais. Também foi definido o optimizador “Adam” e uma a taxa de aprendizado (*learning rate*) de 0.0005 para agilizar os treinamentos. O resumo das configurações da rede neural criada pode ser observado na Figura 4 apresentada a seguir.

Figura 4 - Resumo da rede neural criada.



Fonte: Elaborado pelo autor.

Com isso, o total de parâmetros treináveis é de 4.481.428, representando a soma total dos valores de pesos existentes após um *input* passar por todas as camadas da rede neural construída durante o treinamento. Ainda existem os parâmetros não treináveis que são parâmetros fixos que determinam a arquitetura da rede e pesos que não são atualizados ao longo do treinamento.

3.4. Treinando e Avaliando Modelo

Para a etapa de treinamento e validação do modelo se faz necessário a criação de um objeto histórico através da função *fit* no gerador de amostra de dados criado no Tópico 3.1 desse artigo. Também será necessário a criação de retornos (*call-backs*) para seja possível armazenar o retorno dos treinamentos, ou seja, os resultados obtidos em um modelo.

Para dar início a criação desse objeto histórico, foi definido o número de épocas (*epochs*), ou seja, a quantidade de vezes que um conjunto de dados é passado pela rede. Nesse caso foi definido um total de 15 épocas baseando-se na taxa de aprendizado e quantidade de imagens na amostra do conjunto de dados.

Com o número de épocas definido, é preciso definir o número de interações que ocorrerá em cada época, que é calculada pela divisão inteira do número total de imagens separados para treino pelo número total da amostra de imagens para treinamento.

Seguindo para a definição dos *call-backs*, foi criado um controle dos melhores pesos obtidos através do parâmetro de “valor de acurácia”. Assim, é possível salvar um modelo cada vez mais preciso a cada treinamento realizado na fase de testes, treinamento e avaliações do modelo.

Também foi adicionado um *call-back* com a finalidade de ser um redutor de taxa de aprendizado. Essa técnica é implementada no caso de não houver melhoria no treinamento após um número de épocas percorrido, permitindo que o modelo considere etapas de aprendizado menores para a solução levando em conta a função de custo. Assim, é monitorado o valor de perda em cada e época. Caso, o valor de perda não ocorra para mais de duas épocas seguidas a taxa de aprendizado é reduzida em 0.1 podendo, no máximo, ser reduzida até o valor final de 0.00001. Após isso, nenhuma redução a mais será permitida.

Por fim, para acompanhamento em tempo real do treinamento foi utilizado a função “*PlotLossesKerasTF*” e, finalmente, a criação do objeto históricos com todos as configurações e call-backs criados para o treinamento e validação do conjunto de dados. A configuração final do objeto histórico criado pode ser visualizada na Figura 5 a seguir.

Figura 5 - Objeto histórico criado para treinamento.

```
%%time

epochs = 20
steps_per_epoch = train_generator.n//train_generator.batch_size
validation_steps = validation_generator.n//validation_generator.batch_size

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1,
                              patience=2, min_lr=0.00001, mode='auto')
checkpoint = ModelCheckpoint("model_weights.h5", monitor='val_accuracy',
                             save_weights_only=True, mode='max', verbose=1)
callbacks = [PlotLossesKerasTF(), checkpoint, reduce_lr]

history = model.fit(
    x=train_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=epochs,
    validation_data = validation_generator,
    validation_steps = validation_steps,
    callbacks=callbacks
)
```

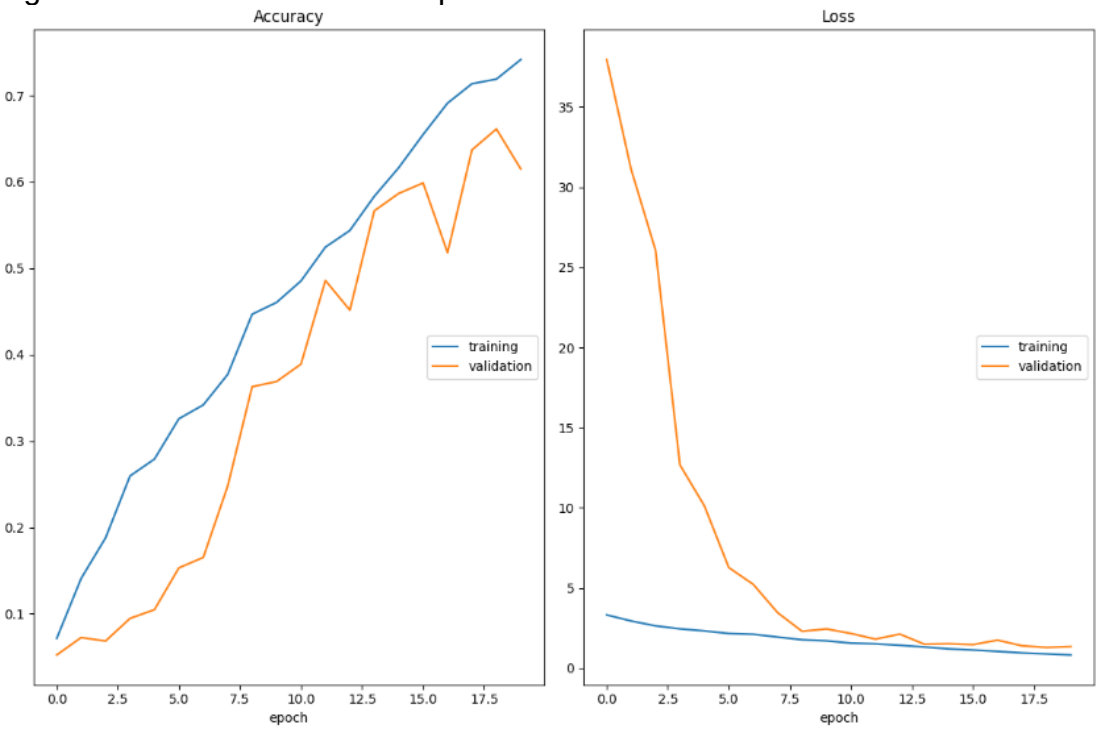
Fonte: Elaborado pelo autor.

4. RESULTADOS E DISCUSSÃO

A medida utilizada para avaliação do modelo após o período de treinamento e teste foi a variável valor da acurácia. Ao longo das 15 épocas foi possível obter um valor final de 61,5% de efetividade obtido no tempo total de 1 hora e 28 minutos de execução.

A Figura 6 ilustra a variação de taxa média de acertos ao longo das épocas, onde na esquerda é possível visualizar a curva de acurácia e a direita a curva de perda.

Figura 6 - Curva de acurácia e perda.



Fonte: Elaborado pelo autor.

Nessa figura é interessante notar que tanto a curva de acurácia quanto a curva de perda seguem desenho considerado como um bom *fit*, ou seja, um ajuste harmonioso em relação a um modelo que sofreu *underfitting* ou *overfitting* apresentando uma pequena lacuna entre as linhas de treino e validação, também conhecida como lacuna de generalização, e apresentando certa estabilidade ao final da curva (BROWNLEE, 2019).

5. CONCLUSAO

A linguagem de sinais brasileira é um meio natural para facilitar a interação entre qualquer espectro de deficiência auditiva e aqueles que não conhecem ou não possuem proficiência na linguagem de sinais, servindo de um importante mecanismo de inclusão que deve ser conhecido e estudado para que ideias e assuntos sejam entendidos em quaisquer situações.

Ao longo desse estudo foi apresentada as condições necessárias para realizar o treinamento e validação de um modelo com nível razoável de assertividade.

Esse estudo mostrou ser capaz de efetuar o aprendizado e o reconhecimento de estáticos através da aplicação da abordagem CNN que, através da medida de desempenho obtida, mostrou-se ser uma abordagem viável e conceituada no reconhecimento de imagens uma vez que a acurácia e perda entre treinamento e validação apresentam uma curva de aprendizagem satisfatória.

Até o momento apenas 20 gestos foram considerados para classificação, entretanto com estudos futuros espera-se aumentar a quantidade de gestos para melhor experiencia.

Futuramente, também considera-se aumentar o resultado final obtido, ou seja, a acurácia do modelo bem como a aplicação em meios tecnológicos como vídeos chamadas para que o reconhecimento possa ser presenciado em tempo real.

REFERÊNCIAS

ACADEMY, Data Science. Deep Learning Book. 2022. Disponível em: https://www.deeplearningbook.com.br. Acesso em: 21 fev. 2023.

ASSUNÇÃO, Renato. Deep Learning. 2004. Disponível em: https://homepages.dcc.ufmg.br/~assuncao/AAP/Sem%2002%20-%20Aula%2004.pdf. Acesso em: 06 fev. 2023

ALZUBAIDI, Laith. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. concepts, CNN architectures, challenges, applications, future directions. 2021. Publicado em The Jornal of Big Data. Disponível em: https://d-nb.info/1233589695/34. Acesso em: 25 jan. 2023.

AWAN, Abid Ali. A Complete Guide to Data Augmentation. 2022. Disponível em: https://www.datacamp.com/tutorial/complete-guide-data-augmentation. Acesso em: 05 fev. 2023.

AWS - AMAZON. O que é uma rede neural? [2022]. Disponível em: https://aws.amazon.com/pt/what-is/neural-network/. Acesso em: 23 jan. 2023.

BROWNLEE, Jason. How to use Learning Curves to Diagnose Machine Learning Model Performance. 2019. Disponível em: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/. Acesso em: 22 fev. 2023.

CAIAFA, Eros G. A.; FONSECA, Fabiana F.; LIMA, Amaro A.; ARAUJO, Gabriel M.; SILVA, Eduardo A. B. da. Aprendizado profundo no reconhecimento de sinais estáticos de Libras**.** 2020. 5 f. Dissertação (Mestrado) - Curso de Tecnologia, E Telecomunicações e Processamento de Sinais, Simpósio Brasileiro de Telecomunicações e Processamento de Sinais, Florianópolis, 2020. Disponível em: https://www.sbrt.org.br/sbrt2020/papers/1570660513.pdf. Acesso em: 06 fev. 2023

CONCI, Aura; MONTEIRO, Leonardo Hiss. RECONHECIMENTO DE PLACAS DE VEÍCULOS POR IMAGEM. 2004. 12 f. Tese (Doutorado) - Curso de Computação,

Universidade Federal Fluminense, Niteroi, 2004. Disponível em: http://www2.ic.uff.br/~aconci/CONENPLACAS.pdf. Acesso em: 30 jan. 2023.

DIAS, Jessica Barbosa; SOUZA, Kleber Padovani de; PISTORI, Hemerson. Conjunto de Treinamento para Algoritmos de Reconhecimento de LIBRAS. 2006. 7 f. Tese (Doutorado) - Curso de Engenharia e Computação, Universidade Católica Dom Bosco, Campo Grande, 2006. Disponível em: http://www.gpec.ucdb.br/pistori/publicacoes/dias_wvc2006.pdf. Acesso em: 06 fev. 2023.

DIGIAMPIETRI, Luciano A.; TEODORO, Beatriz; SANTIAGO, Caio R. N.; OLIVEIRA, Guilherme A.; ARAUJO, Jonatas C. Um sistema de informação extensível para o reconhecimento automático de LIBRAS. 2012. 12 f. Tese (Doutorado) - Curso de Escola de Artes, Ciências e Humanidades, Universidade de s˜Ao Paulo, São Paulo, 2012. Disponível em: https://sol.sbc.org.br/index.php/sbsi/article/view/14410/14256. Acesso em: 30 jan. 2023.

FACURE, Matheus. Redes Neurais Feedforward Densas: implemente modelos básicos de deep learning usando tensorflow. Implemente modelos básicos de Deep Learning usando TensorFlow. 2017. Disponível em: https://matheusfacure.github.io/2017/05/15/deep-ff-ann/. Acesso em: 25 jan. 2023.

GONZALEZ, Rafael C.; WOODS, Richard E..Processamento de Imagens Digitais. São Paulo: Blucher, 2000.

GOODFELLOW, Ian J. Challenges in Representation Learning: a report on three machine learning contests. 2013. 8 f. Dissertação (Mestrado) - Curso de Ciências da Informação, Universidade de Montreal, Canada, 2013. Cap. 3. Disponível em: https://arxiv.org/pdf/1307.0414.pdf. Acesso em: 25 jan. 2023.

OXFORD INTERNET INSTITUTE IN PARTNERSHIP WITH GOOGLE. Reconhecimento de imagem: como os sistemas de ia aprendem a "ver" o mundo ao redor. [2022]. Disponível em: https://atozofai.withgoogle.com/intl/pt-BR/. Acesso em: 23 jan. 2023.

SANTOS, Jonilson Roque dos. Reconhecimento das configurações de mão de libras baseado na análise de discriminante de fisher bidimensional utilizando imagens de profundidade. 2015. 96 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Federal do Amazonas, Manaus, 2015. Disponível em: https://tede.ufam.edu.br/handle/tede/5011. Acesso em: 30 jan. 2023.

SILVA, David Vinicius da; SANTOS JUNIOR, Carlos Roberto dos. Agrupamento de guras utilizando a Rede ART Fuzzy. 2021. 2 f. Tese (Doutorado) - Curso de Computação e Matemática Aplicada, Universidade do Mato Grosso do Sul, Hortolândia, 2021. Disponível em: https://proceedings.sbmac.org.br/sbmac/article/view/125433/3494. Acesso em: 30 jan. 023.

SIQUEIRA, Mozart Lemos de. Reconhecimento automático de padrões em imagens ecocardiográficas. 2010. 107 f. Tese (Doutorado) - Curso de Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2010. Disponível em: https://www.lume.ufrgs.br/handle/10183/19040. Acesso em: 30 jan. 2023.

SOUZA, Gueibi Peres; SAMOHYL, Robert Wayne; MIRANDA, Rodrigo Gabriel de. Métodos Simplificados de Previsão Empresarial. Rio de Janeiro: Ciência Moderna, 2008.

SOURCEX. Why is image recognition a key function of AI? 2021. Disponível em: https://www.sourcex.ai/why-is-image-recognition-a-key-function-of-ai/. Acesso em: 19 jan. 2023.