UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

DEPARTAMENTO DE SISTEMAS DE COMPUTAÇÃO

PROJETO DE VISÃO COMPUTACIONAL

Tradução de Libras por Imagem

DISCIPLINA SSC0910

TÓPICOS AVANÇADO EM CIÊNCIAS DE COMPUTAÇÃO I

Equipe:

Emanuel Percinio Gonçalves de Oliveira - 13676878

Larissa Freire de Jesus Costa - 11207731

Maria Eduarda Kawakami Moreira - 11218751

Professor:

Dr. Fernando Pereira dos Santos

São Carlos, SP

23 de junho de 2023

Resumo

De acordo com o IBGE, cerca de 9 milhões de pessoas no Brasil são portadoras de alguma necessidade especial auditiva e muitas delas usam ferramentas como o Hand Talk [Talk 2016] ou VLibras [VLibras 2016] para traduzir a língua portuguesa para a língua materna dos surdos, a LIBRAS (Língua Brasileira de Sinais). Atualmente, apenas uma pequena parcela da população é dotada do conhecimento da língua de sinais ou até mesmo de ferramentas que auxiliem nessa interação com o portador de necessidades especiais auditivas. Diante desta realidade, este projeto propõe a classificação de imagens com as representações de 20 letras do alfabeto, que por serem estáticas não necessitam de vídeos para sua identificação. Por meio de técnicas de visão computacional e aprendizado de máquina, o objetivo principal do trabalho é desenvolver um classificador para essas letras estáticas do alfabeto, fazendo uma redução de dimensionalidade pelo algoritmo PCA (*Principal Component Analysis*) e colocando as *features* extraídas em uma RNN (*Recurrent neural network*) para a classificação.

Sumário

1 Introdução

1.1 Motivação

É evidente que as relações sociais são pautadas pela comunicação entre as pessoas, pois esta permite o desenvolvimento de laços e constitui um aspecto chave na vida de todo ser humano. De acordo com o IBGE, cerca de 9 milhões de pessoas no Brasil são portadoras de alguma necessidade especial auditiva e muitas delas usam ferramentas como o Hand Talk [Talk 2016] ou VLibras [VLibras 2016] para traduzir a língua portuguesa para a língua materna dos surdos, a LIBRAS (Língua Brasileira de Sinais).

Todavia, uma comunicação mais efetiva acontece quando essa é bilateral, e por isso é preciso também viabilizar a tradução de LIBRAS para o português. Infelizmente, só uma pequena parcela da população é dotada do conhecimento da língua de sinais do país ou até mesmo de ferramentas que auxiliam nessa interação com o portador de necessidades especiais auditivas. Isso se deve, em parte, pelo baixo número de ferramentas já desenvolvidas para a tradução inversa, de LIBRAS para o português e, consequentemente, a baixa disponibilidade de bases de dados para esse fim.

A língua de sinais brasileira é uma linguagem estruturada composta de sinais que representam palavras, verbos, cores, formas e objetos. O alfabeto possui uma complexidade maior em seis letras, que são representadas por gestos dinâmicos.

Apesar de a LIBRAS possuir não apenas as letras do alfabeto, mas também palavras e expressões completas formadas por sinais independentes, é possível formar qualquer palavra e se comunicar, mesmo que de uma forma mais lenta e menos eficiente, apenas com as letras do alfabeto. Percebendo isso, procura-se fazer um bom classificador para estes sinais antes de partir para a maior complexidade do resto da linguagem.

Além de ajudar na comunicação, o trabalho em questão também é motivado pela possibilidade de ser empregado no auxílio da alfabetização da própria Língua Brasileira de Sinais.

1.2 Objetivos

Desenvolver um classificador que identifique a letra sinalizada a partir de uma imagem. Para isso, um *dataset* público será utilizado para treinamento das redes neurais propostas e um próprio *dataset* será construído para complementar o já obtido e contribuir para o desenvolvimento do projeto,

garantindo um melhor resultado.

2 Levantamento Bibliográfico

Dentro do tópico a ser explorado nesse trabalho, que é a tradução de parte do alfabeto de LIBRAS, por meio de visão computacional aplicada a imagens que representam os sinais, existem muitas pesquisas complementares que abordam esse tema ou vão além. Nessa sessão serão elencados alguns trabalhos com intersecção de assunto e as técnicas relevantes por eles abordadas.

[Ribeiro and Rodrigues, 2017]:

A proposta deste trabalho é o mapeamento das letras do alfabeto em libras, para a criação de um dataset de imagens. Utilizam-se da biblioteca Opencv (Open Source Computer Vision Library) e seus classificadores, para mapear os sinais que expressam as letras do alfabeto onde o desenvolvimento de uma aplicação é capaz de receber uma imagem através de uma câmera estacionária devidamente calibrada, identificar o sinal captado e traduzir para a língua portuguesa. Algumas técnicas interessantes abordadas foram a cascata de classificadores, usada para melhorar a performance; as limitações de aplicar um detector de contornos; a metodologia usada para construir o próprio dataset e um método de avaliação do classificador com usuários.

A cascata de classificadores é uma abordagem de aprendizado de máquina na qual uma função em cascata é treinada utilizando várias imagens positivas e negativas. A biblioteca disponível no próprio Opencv analisa as amostras positivas criando padrões entre as imagens negativas de diferenças de intensidade em regiões distintas. Após análise destes padrões ela armazena as características em uma árvore de decisões. O maior problema dessa abordagem é requerer uma vasta gama de exemplos diversificados das imagens.

O detector de contornos usado por eles, o Canny Edge, não reconheceu totalmente as bordas da mão, o que prejudicou a distinção de algumas letras, como o K e o V, que se distinguem pelo posicionamento do dedo polegar sobre o dedo indicador

Para a construção de um dataset próprio, os sinais foram divididos em positivos e negativos, nos quais cada letra foi executada de maneira correta e errada. Os sinais positivos de cada letra foram fotografados 10 vezes em profundidades diferentes, sinais negativos fotografados não tiveram sua profundidade alternada, sendo estas imagens convertidas para cinza.

Para a avaliação usaram três tipos de usuários de acordo com o domínio da Libras por parte deles

— total, médio e nenhum conhecimento.

[NETO, 2019]:

Este trabalho apresenta um protótipo de tradutor de Libras para Português que contempla 20 letras do alfabeto. O mesmo utiliza ferramentas de Visão Computacional, como técnicas de Processamento de Imagem e Aprendizado de Máquina, para desenvolver um classificador capaz de identificar qual a letra sinalizada analisando uma imagem capturada por uma câmera RGB. Os pontos chaves foram a metologia de construção de um dataset manual e a aplicação de segmentação de imagem.

Para a construção do dataset próprio, primeiramente foram tiradas fotos em diferentes posições e com enquadramentos diferentes. Além de padronizar o tamanho das imagens,uma máscara de transparência, criada com o Blender, foi aplicada visando a remoção do plano de fundo. Isto facilitaria posteriormente para a anotação e corte da região de interesse (a mão).

O pré-processamento foi realizado com a ImageDataGenerator disponível na Keras. Essa biblioteca também foi usada para a conversão em escala de cinza e a fim de aumentar a quantidade de exemplos disponíveis aplicando transformações de zoom, rotação, entre outros. A ferramenta flow-from-directory foi usada para carregamento das imagens na rede neural, por otimizar o uso de memória durante o treinamento. Usaram também técnicas de caixa delimitadora e discutiu-se a pauta de como o tom de pele influencia na detecção da mão.

No final além de proporem uma interface gráfica para reconhecimento de libras, acoplaram também à aplicação uma ferramenta de escrita a partir das imagens da câmera.

[Pavan et al., 2010]:

Esta pesquisa demonstrou uma técnica de reconhecimento de gestos que tem como objetivo rastrear sinais de Libras a partir de imagens capturadas de uma webcam.

Neste trabalho foram citadas diferentes linhas de processamento para resolver o mesmo problema do projeto em questão. Citaram a biblioteca OpenCV para rastreamento em tempo real usando segmentação da área de interesse através do algoritmo CamShift (Continuously Adaptive Mean-SHIFT), que captura a imagem original e realiza a distribuição de cores em um modelo de histograma, criando um padrão de cor a ser rastreado. Com a borda da imagem extraída pelo algoritmo de Sobel, realiza-se a comparação com um template pré-definido através de uma técnica conhecida como Shape Matching, que é responsável por encontrar semelhanças entre as imagens capturadas e o template, respeitando certa taxa de erro.

Outra proposta interessante citada foi a utilização de classificadores, através da qual o sistema é capaz de identificar com velocidade os objetos, não dependendo de imagens em sequência ou de um padrão de cores como em outras técnicas. Porém essa técnica exige muitas imagens e capacidade de processamento para desenvolver um classificador preciso e robusto.

A abordagem escolhida portanto foi a utilização de classificadores e detectores de bordas para realizar o rastreamento. Com a utilização de classificadores torna-se desnecessário extrair o fundo com o algoritmo CamShift, o que reduz consideravelmente os ruídos, pois realizando-se a análise diretamente na imagem segmentada a quantidade de decisões do classificador será menor e dessa forma o processamento será mais rápido e preciso. Além disso, a quantidade de imagens necessárias para realizar o treinamento de cada sinal será bem menor.

[Bantupalli and Xie, 2018]:

Neste projeto o modelo proposto pega sequências de vídeos e extrai características temporais e espaciais delas. Em seguida, usa a Inception, uma CNN (Convolutional Neural Network) para reconhecer características espaciais. Em seguida, usa uma RNN (rede neural recorrente) para treinar os recursos temporais. Porém o conjunto de dados usado é o conjunto de dados da língua de sinais americana. Ele menciona um projeto que usa PCA para reconhecer silhuetas a partir de uma câmera estática. Comenta também que as redes neurais ainda enfrentam problemas como rastreamento de mãos, segmentação de assuntos do fundo e do ambiente, variação de iluminação, oclusão, movimentos e posição.

[Ardiansyah et al., 2021]:

Neste estudo, são revisados 22 artigos de pesquisa relacionados ao Reconhecimento de Língua de Sinais (SLR) Americana. Das pesquisas mencionadas, o processo SLR pode ser dividido em 5 etapas comuns: aquisição, pré-processamento, segmentação, extração de características e classificação. Nesse projeto é comentado também que o método mais comum na segmentação é Thresholding (limiarização). Geralmente aplicado após a imagem ficar em escala de cinza, limiarização transforma a imagem em forma binária. Como a escala de cinza transforma uma imagem em preto e branco, a limiarização é aplicada para identificar o plano de fundo e o primeiro plano, onde o preto representa o plano de fundo e o branco representa o primeiro plano. Um valor limite será escolhido pelo pesquisador e esse valor determinará qual cor será o plano de fundo e qual será o primeiro plano. Joshi et al. usar Otsu Algorithm, é um algoritmo que determina automaticamente um bom valor limite.

Indica também o filtro morfológico para ajudar reduzir o erro do primeiro plano ou do plano de fundo, aumentando efetivamente o ajuste da região de interesse.

3 Datasets

Para a execução desse projeto, foram utilizados três *datasets* distintos, sendo um composto de 44.104 imagens de 20 letras do alfabeto (letras cuja representação em libras não possui movimento) disponibilizado publicamente[1], outro com 4.426 imagens de 15 letras do alfabeto (letras com representação estática, com exceção das letras 'F', 'G', 'P', 'Q' e 'T') também disponibilizado publicamente[2], por fim, um ultimo dataset foi criado pelo autores como forma de obter imagens que representassem melhor a ideia do projeto implementado.

3.1 Dataset 1

O primeiro dataset,composto de 44.104 imagens divididas entre treino e teste, foi construido a partir de, em média, 7 imagens para cada letra, que passaram por processos de *data augmentation* (brilho, flip horizontal e rotação) para gerar novas imagens. Note que isso implica que cada imagem base gerou cerca de 315 imagens.

A figura 1 (a-c) mostra um exemplo de imagem do *Dataset* 2.

Figura 1: Exemplos de imagens do Dataset 1

a) Letra A

b) Letra B

c) Letra P

Fonte: Dataset 1.

[1] https://github.com/lucaaslb/cnn-libras
[2] https://github.com/biankatpas/Brazilian-Sign-Language-Alphabet-Dataset

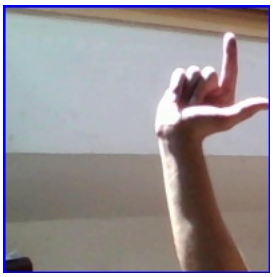
3.2 Dataset 2

O segundo dataset,composto de 4.426 imagens, foi construido a partir de frames de um vídeo para cada letra (movimento a mão e mudando iluminação), que passaram por processos de *data augmentation* (contraste e brilho) para gerar novas imagens.

A figura 2 (a-c) mostra um exemplo de imagem do *Dataset* 1.

Figura 2: Exemplos de imagens do Dataset 2

a) Letra R



b) Letra L

c) Letra V

Fonte: Dataset 2.

3.3 Dataset 3: Criado pela equipe

A partir do baseline inicial, nota-se que chegamos à saturação do resultado com 100% de acurácia no modelo proposto. Para a continuidade do projeto, decidimos criar um dataset próprio, mas isso é um processo custoso e que requer muito tempo. Então optamos por criar uma versão mais simples para conseguirmos usar, pelo menos, como exemplos de teste. A ideia era conseguir imagens totalmente novas para a rede e bem próximas da realidade, sem padronização de fundo e com gestos feitos por pessoas com diferentes níveis de conhecimento sobre a linguagem de libras.

Inicialmente divulgamos em nossas redes sociais pedindo colaboração para o trabalho, como mostra as postagens da Figura 3. Pedimos que nossos amigos mandassem uma foto da mão deles fazendo a primeira letra do nome em libras. O único requisito que pedíamos era uma iluminação que deixasse a imagem da mão visível e já filtramos nossas letras de interesse. Após essa primeira tentativa de constituir o dataset conseguimos coletar aproximadamente 60 imagens.

Figura 3: Postagens usados nas redes sociais

Fonte: Elaborada pelo autor.

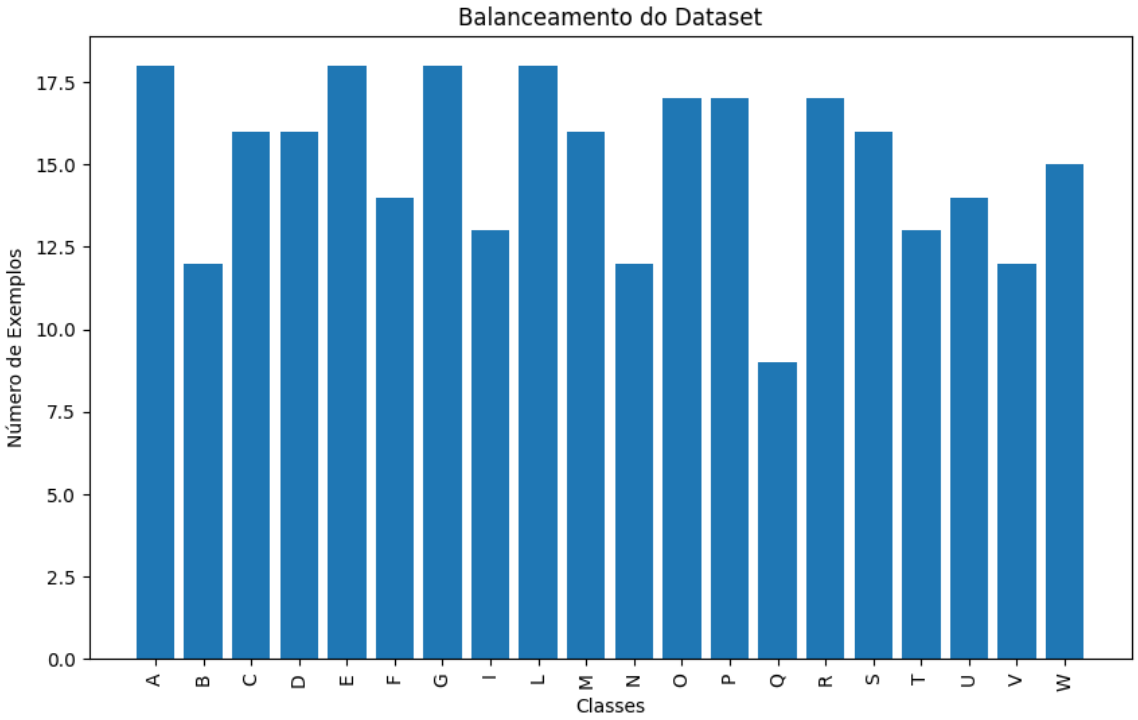
Como nosso objetivo era conseguir pelo menos 200 imagens, resolvemos realizar uma pequena atividade de campo, desenvolvida na própria faculdade. Propomos a alguns alunos voluntários uma abordagem semelhante a utilizada nas redes sociais, que fizessem a inicial do seu nome ou sobrenome em libras, mas ensinávamos caso eles não soubessem. Como para muitos estudantes esse contato foi primário, decidimos por relevar imperfeições que não afetavam a comunicação ou compreenção do sinal. Tentamos colocar um fundo branco para padronizar minimamente essas imagens e ajudar a rede. As fotos obtidas foram acrescentadas ao dataset e ao final totalizamos aproximadamente 240 imagens com as obtidas pelas redes sociais.

Além disso usamos mais 40 imagens retiradas de vídeos em que pessoas faziam os sinais das letras do alfabeto.

Ao final das nossas três abordagens de coleta de dados, padronizamos as imagens (utilizando *resize*) para o mesmo formato das imagens presentes nos demais datasets, nesse caso, $64x64$. Por fim, verificamos o balanceamento entre os exemplos de cada classe, como mostra o gráfico da Figura 4.

Entendemos que tínhamos um bom balancemaento dos dados sem uma discrepância tão significativa entre as classes, um ponto muito importante para não viesar a classificação da rede.

Figura 4: Balanceamento do Dataset



Fonte: Elaborada pelo autor.

A partir de uma boa quantidade de exemplos para os nossos obejtivos e diante de um bom balanceamento de classes, buscamos aumentar o dataset para termos mais exemplos e uma maior variabilidade de imagens. Para isso aplicamos algumas técnicas de data augmentation. Decidimos por implementar seis diferentes técnicas, como mostra a Figura 4, mas selecionávamos quatro tipos randomicamente para cada imagem. Assim não saturávamos nosso dataset com exemplos altamente semelhantes e ainda garantíamos a variabilidade entre elas através da permutação das técnicas.

Técnicas de data augmentation implementadas:

- Flip - No contexto de libras, onde os gestos das mãos têm uma simetria espelhada, o flip pode ajudar a melhorar a robustez do modelo em reconhecer gestos independentemente da orientação da mão

- Contraste - Ao ajustar o contraste, podemos ampliar ou reduzir a diferença entre os níveis de intensidade dos pixels, tornando o modelo mais resistente a variações de iluminação que podem ocorrer em ambientes reais.

- Saturação - No contexto de libras, onde os gestos das mãos podem ter diferentes tons de pele e

cores de roupas, ajustar a saturação ajuda o modelo a reconhecer gestos independentemente da variação de cor.

- Ruído Salt and Pepper - A adição de ruído salt and pepper em uma imagem ajuda a simular imperfeições e ruídos que podem estar presentes em imagens reais.

- Rotações - A rotação das imagens é uma técnica valiosa para simular variações na orientação dos gestos das mãos. Ao girar as imagens em 20 graus no sentido horário e antihorário, estamos criando variações de orientação que podem ser encontradas na captura de gestos reais.

Aplicamos também a escala de cinza em todas as imagens visando simplificar a representação visual, eliminando a informação da cor, que consideramos não relevante para o nosso problema.

Figura 5: Exemplos das técnicas de data augmentation aplicadas para uma imagem do dataset

Fonte: Elaborada pelo autor.

---

4 Baseline Inicial Implementado

Na primeira etapa do projeto, propõe-se um estudo genérico para mensurar a complexidade do problema. Para isso, utiliza-se o *Dataset* 1.

Após a extração das imagens presentes no banco de dados, é feita a extração de características e redução de dimensionalidade. A extração de característica é realizada por cinco modelos de CNNs distintos (Tabela 1) e a redução de dimensionalidade é realizada pelo algoritmo PCA (*Principal Component Analysis*). Por fim, as *features* extraídas são colocadas em uma RNN para classificação com duas camadas densas, a primeira composta de 40 neurônios e uma ativação “ReLU”, e a segunda com 20 neurônios e uma ativação “softmax”. A função de perda utilizada para treinar a rede foi a crossentropia categorical e a métrica usada para avaliar o modelo foi a acurácia categórica.

Tabela 1: Arquiteturas Utilizadas para Extração de *features*

Modelo	Tamanho (MB)	Parâmetros	Profundidade
VGG16	528	138.4M	16
ResNet50	98	25.6M	107
EfficientNetB3	48	12.3M	210
MobileNetV2	14	3.5M	105
DenseNet169	57	14.3M	338

Fonte: Keras API

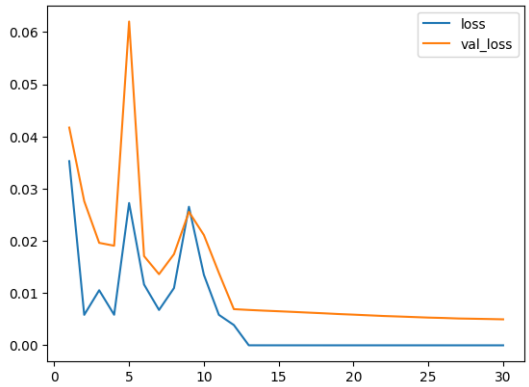
Ponderando-se o desempenho dos modelos com suas características, escolhe-se o modelo que será utilizado para a continuidade do projeto.

O desempenho obtido com cada arquitetura testada é analisado por meio de três valores, o valor da função de perda, da métrica em um *dataset* de validação e a métrica (*score*) no *dataset* de teste. O código com os resultados obtidos estão disponibilizados num repositório do GitHub[3].

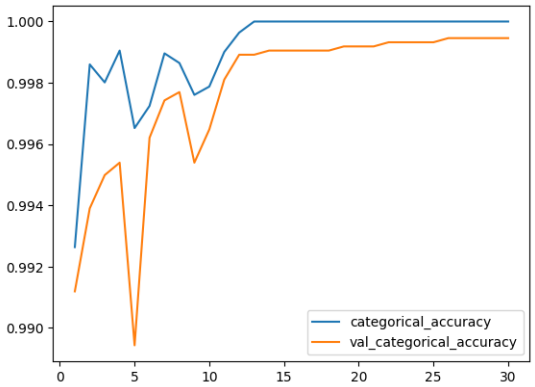
4.1 VGG16

A figura 6 (a e b) mostra o desempenho da rede com as features extraídas pela arquitetura VGG16.

Figura 6: Treinamento da RNN com features extraídas pela VGG16.



a) Função de perda



b) Acurácia

Fonte: Elaborada pelo autor.

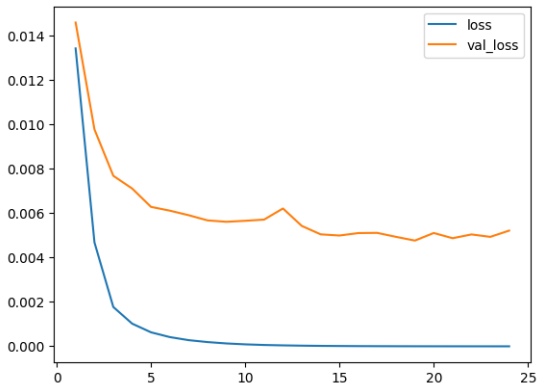
$$\boxed{VGG16_{score} = 99.92\%}$$

[3]https://github.com/emanuelpg/Traducao-de-Libras

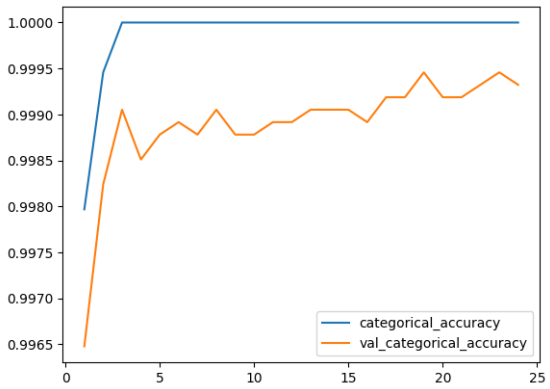
4.2 ResNet50

A figura 7 (a e b) mostra o desempenho da rede com as features extraídas pela arquitetura ResNet50.

Figura 7: Treinamento da RNN com features extraídas pela ResNet50.



a) Função de perda



b) Acurácia

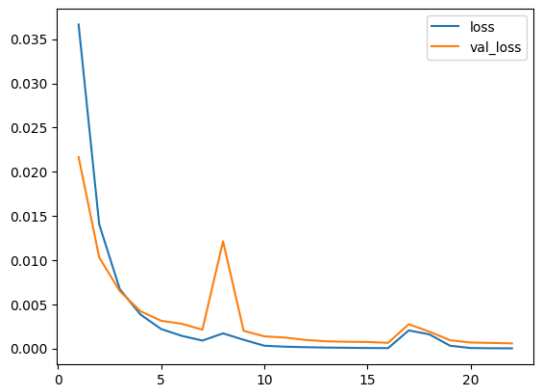
Fonte: Elaborada pelo autor.

$$\boxed{ResNet50_{score} = 99.97\%}$$

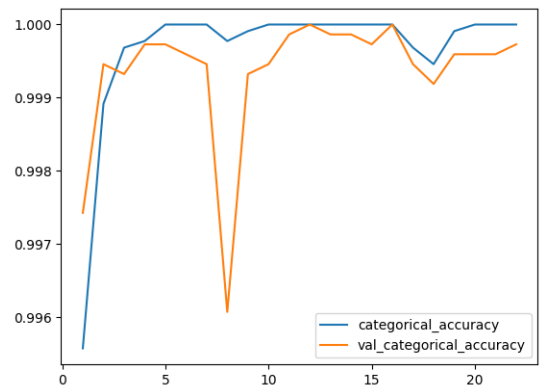
4.3 EfficientNetB3

A figura 8 (a e b) mostra o desempenho da rede com as features extraídas pela arquitetura EfficientNetB3.

Figura 8: Treinamento da RNN com features extraídas pela EfficientNetB3.



a) Função de perda



b) Acurácia

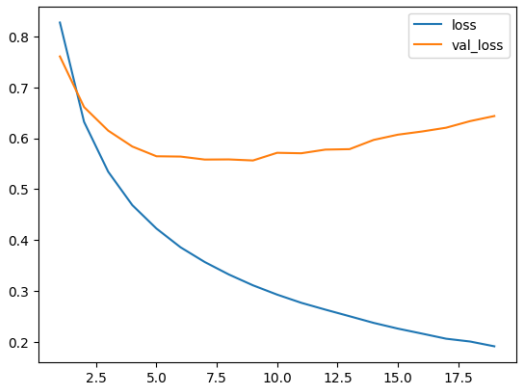
Fonte: Elaborada pelo autor.

$$\boxed{\textit{EfficientNetB3}_{score} = 99.96\%}$$

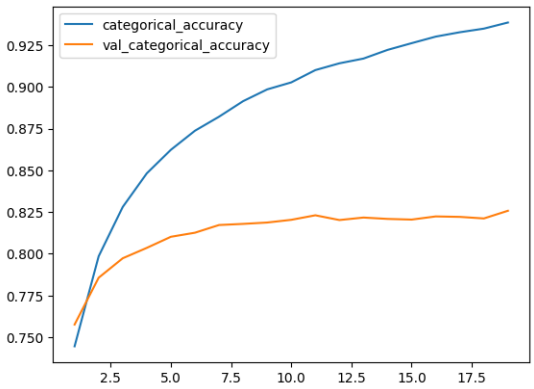
4.4 MobileNetV2

A figura 9 (a e b) mostra o desempenho da rede com as features extraídas pela arquitetura MobileNetV2.

Figura 9: Treinamento da RNN com features extraídas pela MobileNetV2.



a) Função de perda



b) Acurácia

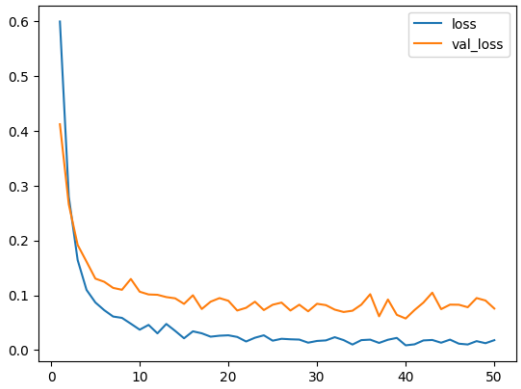
Fonte: Elaborada pelo autor.

$$\boxed{\textit{MobileNetV2}_{score} = 82.29\%}$$

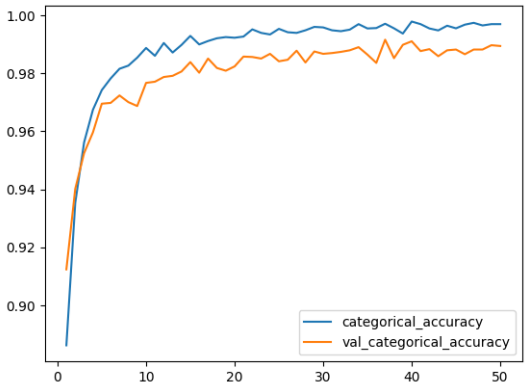
4.5 DenseNet169

A figura 10 (a e b) mostra o desempenho da rede com as features extraídas pela arquitetura DenseNet169.

Figura 10: Treinamento da RNN com features extraídas pela DenseNet169.



a) Função de perda



b) Acurácia

Fonte: Elaborada pelo autor.

$$\boxed{DenseNet169_{score} = 99.24\%}$$

4.6 Conclusões

A tabela 2 apresenta um resumo dos resultados obtidos pela rede para cada uma das arquiteturas analisadas. A partir dela decide-se utilizar a arquitetura EfficientNetB3 para extração de característica, por seu bom custo-benefício em termos de desempenho e tempo de execução.

Tabela 2: Resultados obtidos pela RNN classificadora

Extração de features	Melhor "Val_loss"	Melhor "Val_accuracy"	Test_Score
VGG16	0.0050	99.95%	99.92%
ResNet50	0.0048	99.95%	99.97%
EfficientNetB3	0.0045	100%	99.96%
MobileNetV2	0.4464	82.58%	82.29%
DenseNet169	0.0577	99.16%	99.24%

Fonte: Keras API

5 Baseline Final Implementado

Durante uma análise mais cuidadosa do dataset utilizado no baseline inicial para o reconhecimento de sinais de Libras em imagens estáticas, identificamos algumas limitações que podem ter contribuído negativamente para os resultados obtidos. Uma das principais preocupações observadas foi a falta de diversidade e variação nas imagens.

O dataset original consistia em imagens com poucas modificações aplicadas através do data augmentation, o que resultou em exemplos altamente semelhantes entre si. Essa falta de diversidade pode ter levado a um possível "overfitting", em que o modelo decorou as imagens ao invés de aprender e por isso atingimos a saturação.

Outra questão foi a ausência de uma distinção adequada entre as imagens usadas nos conjuntos de treinamento e teste. Se as mesmas imagens foram utilizadas tanto para treinamento quanto para teste, isso pode ter resultado em uma avaliação enviesada do desempenho do modelo.

Tendo isso em mente, decidimos criar um novo dataset, implementando técnicas de data augmentation como explicado na sessão 3.

Afim de superar as limitações do baseline inicial, realizamos quatro novas abordagens de código:

5.1 Estretégia 1: Treino com o dataset inicial e teste com o nosso dataset

Utilizamos o mesmo dataset do baseline inicial (*Dataset* 1) para treinamento da rede, mas dessa vez, a validação e testes foram realizados com o nosso dataset (*Dataset* 3).

O resultado obtido para os dados de teste foi de apenas 19.16%, demonstrando o quanto a rede foi incapaz de aprender com o *dataset* 1 e generalizar as classificações.

5.2 Estretégia 2: Treinamento e teste com o nosso dataset criado, utilizando a EfficientNetB3

Treinamento e teste realizados com exemplos do nosso dataset criado, aplicando um fine-tunning com a rede selecionada, a EfficientNetB3.

O resultado obtido nos dados de teste foi com uma acurácia de 61.59%. Acreditamos que o resultado não tenha sido tão bom pelo nosso dataset ser menor do que o dataset original. Nesse caso, a EfficientNetB3 pode não aprender direito por ser complexa demais para o nosso conjunto de dados.

5.3 Estretégia 3: Treinamento e teste usando nosso dataset, mas criando uma rede rasa própria

Treinamento e teste com o exemplos do nosso dataset utilizando uma rede rasa criada do zero. (*Dataset* 3). Na configuração da rede rasa usamos cinco camadas de convolução para a extração de características com a função de ativação Relu, muito empregada na literatura. Adicionamos uma camada de MaxPooling com filtro 2x2 e uma de Dropout para desligar de forma aleatória 40% dos neurônios, afim de evitar o overffiting. Essa proporção foi testada empiricamente e concluímos que adicionando a camada de dropout conseguíamos um resultado melhor na métrica de acurácia do modelo em comparaçõ a sua ausência. Também foi importante ajustar a taxa de aprendizado para um valor baixo, pois a rede estava ocilando muito durante o treinamento. Realizamos a fase de teste com a rede rasa, mas obtivemos um resultado parecido com o item anterior e a melhor acurácia que conseguimos obter foi de 56.29%.

5.4 Estratégia 4: Treinamento e teste utilizando uma adaptação da EfficientNetB3 com o nosso dataset

Como o nosso dataset era mais simplificado, usar um modelo muito complexo não ia ajudar no treinamento e podia até ocasionar overfitting. Decidimos assim, usar camadas mais internas da EfficientNetB3 para realizar a predição, pois as camadas iniciais da rede extraem características mais gerais. A rede possui 240 camadas no total e empiricamente optamos por selecionar as primeiras 70, 1/3 das camadas. Graças a essa abordagem conseguimos aumentar a acurácia da rede significativamente, obtendo uma taxa de 82.12% de acerto.

O código com os resultados obtidos estão disponibilizados num repositório do GitHub[4].

6 Possibilidades de melhorias futuras

Para melhorar o resultado que obtivemos, algumas técnicas poderiam ser implementadas e testadas futuramente. Acreditamos que a segmentação apresentaria bons resultados, pois permitiria, a aprtir da identificação de objetos, separar as mãos que estão realizando os sinais dos demais elementos da

[4]https://github.com/emanuelpg/Traducao-de-Libras

cena. Isso possibilitaria uma análise mais precisa e focada na extração de características das mãos, o que poderia contribuir para um melhor reconhecimento dos sinais e desempenho de classificação.

Referências

[Ardiansyah et al., 2021] Ardiansyah, A., Hitoyoshi, B., Halim, M., Hanafiah, N., and Wibisurya, A. (2021). Systematic literature review: American sign language translator.

[Bantupalli and Xie, 2018] Bantupalli, K. and Xie, Y. (2018). American sign language recognition using deep learning and computer vision.

[NETO, 2019] NETO, V. C. L. (2019). Reconhecimento de sinais do alfabeto da libras usando visão computacional.

[Pavan et al., 2010] Pavan, A. R., Cazhurriro, P. J., and Modesto, P. F. (2010). Reconhecimento de gestos com segmentação de imagens dinâmicas aplicadas a libras.

[Ribeiro and Rodrigues, 2017] Ribeiro, D. R. S. and Rodrigues, M. T. A. N. (2017). Tradução de libras por imagem.