DEEP LEARNING E REDE NEURAL CONVOLUCIONAL PARA O RECONHECIMENTO DE EMOÇÕES

Tauan Oliveira Da Silva[1], Merisandra Côrtes de Mattos[2]

Resumo: Este artigo teve como objetivo realizar a construção, treinamento e análise de diferentes arquiteturas de redes neurais convolucionais no reconhecimento das sete emoções primárias básicas, incluindo raiva, nojo, medo, felicidade, tristeza, surpresa e neutralidade. Foram avaliadas duas arquiteturas amplamente utilizadas, a ResNet e DenseNet. Os resultados obtidos demonstraram que a arquitetura ResNet apresentou o melhor desempenho, chegando a 91% de precisão no reconhecimento das emoções, tendo executado quase metade das épocas, 110 execuções em comparação com a DenseNet que precisou de 210 para alcançar a maior precisão. Através de métricas analisou-se o desempenho dessas redes no reconhecimento de emoções por meio de imagens, mostrando que a escolha adequada da arquitetura da CNN pode ter um impacto significativo na precisão e na eficácia desses sistemas.

Palavras-chave: CNN. Arquiteturas. ResNet. DenseNet.

ABSTRACT: This article aimed to carry out the construction, training and analysis of different architectures of convolutional neural networks in the recognition of the seven basic primary emotions, including anger, disgust, fear, happiness, sadness, surprise and neutrality. Two widely used architectures, ResNet and DenseNet, were evaluated. The results showed that the ResNet architecture presented the best performance, reaching 91% accuracy in emotion recognition, having performed almost half of the epochs, 110 executions compared to DenseNet, which needed 210 to reach the highest accuracy. Through metrics, the performance of these networks in the recognition of emotions through images will be analyzed in practice, showing that the appropriate choice of CNN architecture can have a significant impact on the accuracy and effectiveness of these systems.

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (UNESC), Criciúma – SC - Brasil. tauan-oliv@hotmail.com.
[2]Orientadora, Curso de Ciência da Computação, Grupo de Pesquisa em Inteligência Artificial Aplicada, Universidade do Extremo Sul Catarinense (UNESC), Criciúma - SC - Brasil. mem@unesc.net.

Keywords: CNN. Architectures. ResNet. DenseNet.

1 INTRODUÇÃO

O reconhecimento facial tornou-se um campo de pesquisa que se desenvolveu rapidamente (SRINIVAS *et al*., 2015, tradução nossa), apresentando-se como um dos mais populares quanto ao processamento e a análise de imagens, em função de diversos sistemas que empregam esta tecnologia. A partir desta área, juntamente com o behaviorismo, originaram-se as pesquisas voltadas ao reconhecimento de emoções.

Baseando-se na descoberta desses padrões complexos, pode-se aplicar a inteligência artificial, mais especificamente o *deep learning*, para o reconhecimento das emoções humanas. A inteligência artificial tem por finalidade criar sistemas inteligentes que possam executar tarefas, as quais geralmente são realizadas por seres humanos, sendo influenciadas por diversas áreas do conhecimento, como por exemplo, filosofia, matemática, biologia, economia, neurociências, psicologia, engenharia de computadores e linguística (SABHARWAL; SELMAN, 2011).

Dentre as abordagens da inteligência artificial, tem-se as redes neurais artificiais que se comunicam por meio de ligação (por isso o conceito de “rede”), inspirando-se no funcionamento dos neurônios do cérebro humano (ZANELLATO *et al*, 2017). No que se refere ao aprendizado de máquina, subcampo da inteligência artificial, tem-se o *deep learning*, aprendizado profundo, que permite aos computadores resolverem problemas nos mais diferentes campos, como por exemplo de visão computacional, reconhecimento de imagem e de fala. Nesta abordagem, tem-se as redes neurais profundas (*deep neural networks*) que apresentam várias camadas de processamento para reconhecer padrões mais complexos.

As redes neurais profundas são os modelos de aprendizado de máquina mais populares e amplamente utilizados em visão computacional, não apenas para classificação e segmentação semântica, mas também para tarefas de nível inferior, como aprimoramento de imagem, estimativa de movimento e recuperação de profundidade (BENGIO; LECUN; HINTON, 2021). Dentre as técnicas, tem-se as redes neurais convolucionais que têm se destacado como uma ferramenta promissora para a resolução de problemas de aprendizado de máquina, como por exemplo, para o

reconhecimento de emoções, devido à sua capacidade de extrair características relevantes de dados complexos.

Exploraremos nesta pesquisa a aplicação de redes neurais profundas por meio de modelos de redes neurais convolucionais no reconhecimento das sete emoções básicas, estudadas por Paul Ekman na década de 1960, as quais são raiva, nojo, medo, felicidade, tristeza, surpresa e neutralidade. Sendo aplicado dois modelos de CNN, o DenseNet e ResNet, com o objetivo de avaliar por meio de métricas de desempenho os resultados alcançados por cada um dos modelos gerados.

2 TRABALHOS CORRELATOS

Alguns estudos correlatos a este são abordados nesta seção. Intitulado *"A Novel Facial Expression Intelligent Recognition Method Using Improved Convolutional Neural Network*", escrito por Min Shi, Lijun Xu e Xiang Chen em 2020 para o IEEE, apresenta um método de reconhecimento inteligente de expressões faciais que utiliza uma combinação de algoritmo de agrupamento Fuzzy C-means (FCM) e redes neurais convolucionais (CNN). O algoritmo FCM é aplicado ao núcleo da CNN para extrair características das imagens de expressão no conjunto de treinamento e teste. O trabalho destaca a importância do pré-processamento das imagens de expressão facial, incluindo recorte facial e normalização, para melhorar a qualidade dos recursos extraídos. No entanto, o estudo identifica a necessidade de pesquisas adicionais para aumentar a taxa de reconhecimento de expressões faciais e garantir a robustez do modelo em cenários de fundos complexos e múltiplas faces.

Já o artigo *"Exploring Discriminative Representations for Image Emotion Recognition With CNNs"* focado no reconhecimento de emoções a partir de imagens, exploram o uso de CNNs e investigam diferentes abordagens para aprimorar o poder discriminativo das representações aprendidas. O estudo começa destacando a importância do reconhecimento de emoções em diversos domínios, incluindo computação afetiva, interação humano-computador e análise de conteúdo visual. Os autores revisam trabalhos anteriores sobre reconhecimento de emoções em imagens e enfatizam a cultura do uso de CNNs devido à sua capacidade de aprender características relevantes automaticamente.

A proposta de pesquisa neste artigo visa explorar e comparar diferentes técnicas para melhorar o desempenho do reconhecimento de emoções. Os autores investigam o uso de técnicas de pré-processamento de imagens, tais como garantia de características e normalização, além de abordagens para melhorar a representação aprendida pelas CNNs.

Os resultados do estudo demonstram que as representações discriminativas vividas por meio das CNNs podem melhorar significativamente o desempenho do reconhecimento de emoções em imagens. Os autores concluem que explorar essas representações discriminativas é uma estratégia promissora para avançar no campo do reconhecimento de emoções em imagens.

Ambos os estudos enfatizam a importância do pré-processamento das imagens e reconhecem a relevância das CNNs para aprender características relevantes automaticamente. Eles também destacam a necessidade de melhorar o desempenho do reconhecimento de emoções e explorar representações discriminativas para avançar nessa área de pesquisa.

Utilizando-se das técnicas de pré-processamento de imagens e da exploração de diferentes arquiteturas de CNNs busca-se analisar o desempenho dessas arquiteturas em termos de acurácia, eficiência e robustez. Ao investigar e comparar os resultados obtidos, espera-se contribuir para o desenvolvimento de soluções mais eficazes e confiáveis no reconhecimento de emoções.

3 MATERIAIS E MÉTODOS

A caracterização da pesquisa quanto à natureza, é aplicada e de base tecnológica, voltando-se ao uso de algoritmos de redes neurais profundas para o reconhecimento de expressões faciais referente as sete emoções primárias. Em relação aos objetivos é caracterizada como uma pesquisa descritiva, pois avalia o desempenho dos modelos gerados pelos algoritmos de redes neurais convolucionais empregados. No que se refere aos procedimentos tem-se uma pesquisa bibliográfica e experimental, pois envolve a manipulação de diferentes parâmetros e arquiteturas de redes neurais convolucionais.

Na presente pesquisa realizou-se uma avaliação dos modelos gerados por meio dos algoritmos DenseNet e ResNet no reconhecimento de emoções. Para isso, utilizou-se um banco de imagens padronizado, pré-processamento das imagens, e

métricas adotadas para avaliar o desempenho dos modelos, tais como precisão, F1-score, entre outras. Essas métricas serão utilizadas para analisar a eficácia das CNNs no reconhecimento de emoções e permitir uma avaliação criteriosa de seus resultados.

Figura 1 – Fluxo de trabalho

Fonte: Do autor

Na Figura 1 tem-se a estrutura geral desta pesquisa, que compreendeu no seu desenvolvimento as seguintes etapas:

1) Definição da base de imagens:
2) Pré-processamento das imagens;
3) Importação das bibliotecas necessárias e configuração do ambiente.
4) Definição da arquitetura da CNN (DenseNet, ResNet);
5) Definição do tamanho de entrada da imagem e do número de classes;
6) Compilação do modelo;
7) Configuração dos geradores de dados para treinamento e teste;

8) Conexão a um banco de dados SQLite e criação de uma tabela para armazenar os dados das épocas;
9) Treinamento do modelo usando os datasets, com o uso do early stopping e do callback personalizado;
10) Salvamento do modelo treinado.

Neste estudo, a implementação da CNN foi realizada no ambiente do Sistema Operacional (OS) Windows 11 Pro, utilizando um computador NITRO AN515-44 64 Bits equipado com um processador AMD Ryzen 7 4800Hz com Radeon Graphics e 16GB de RAM. A linguagem de programação utilizada foi o Python na versão 3.7.16, com desenvolvimento realizado na IDE Spyder na versão 5.4.2. Essas especificações de hardware e software foram escolhidas para garantir um ambiente adequado e eficiente para a execução do treinamento e avaliação da CNN.

Para o desenvolvimento dessas CNNs, foram utilizadas algumas bibliotecas essenciais, incluindo o Tensorflow 2.6.0 e o keras 2.6.0. O tensorflow é uma biblioteca abrangente de código aberto que é conhecido por sua capacidade de criar modelos de aprendizado de máquina e aprendizado profundo. Um benefício dessas bibliotecas é que o Keras já vem incorporado como uma API dentro do próprio Tensorflow.

3.1 CONJUNTO DE DADOS

A base de dados FER-2013 (FER_2013, 2020), também conhecida como "Facial Expression Recognition 2013", é um conjunto de dados amplamente utilizado no campo do reconhecimento de expressões faciais. Essa base de dados foi compilada por Pierre-Luc Carrier e Aaron Courville em 2013 e contém imagens faciais de baixa resolução em escala de cinza (FER_2013, 2023, tradução nossa).

Para adquirir resultados mais completos, o dimensionamento do dataset foi feito de duas maneiras, um banco completo e um banco reduzido, ambos com uma relação de 80% de treino para 20% de teste, como pode-se observar na Tabela 1.

Tabela 1 – DATASET Completo.

COMPLETO	**Raiva**	**Nojo**	**Medo**	**Felicidade**	**Neutro**	**Triste**	**Surpreso**
Treino	3995	436	4097	7215	4965	4830	3171
Teste	959	111	1024	1774	1233	1247	831
REDUZIDO							
Treino	400	400	400	400	400	400	400
Teste	100	100	100	100	100	100	100

Fonte: Do autor.

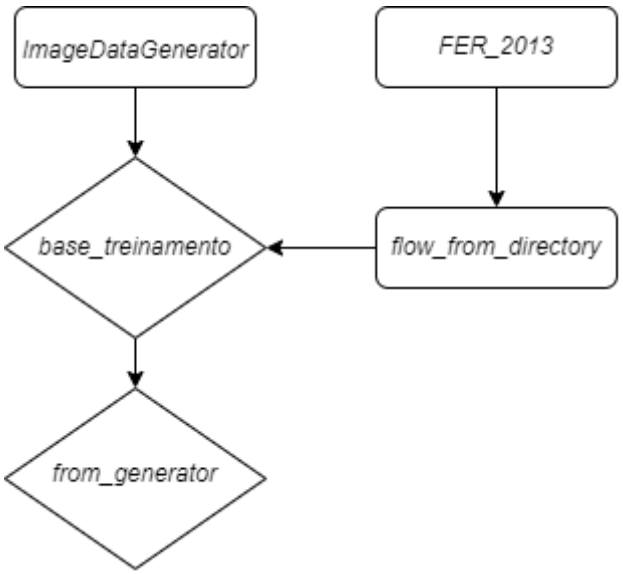
3.2 PRÉ-PROCESSAMENTO

O pré-processamento consistiu na coleta dos dados, normalização, aumento de dados e remoção de ruídos. Essas etapas de pré-processamento das imagens têm o objetivo de padronizar, melhorar a qualidade e aumentar a diversidade do conjunto de dados, contribuindo para o desempenho e a generalização do modelo de CNN. Cabe ressaltar que as técnicas de pré-processamento podem variar de acordo com o contexto e a natureza do problema em questão. Lembre-se que todas as informações de processamento passadas a partir desse ponto foram replicadas igualmente para todas as CNNs.

3.2.1 Consumo do Data-base

Na representação do código da Figura 2, foi utilizado geradores de dados utilizando a classe *ImageDataGenerator* do Keras. Em seguida, é criado o objeto *base_treinamento* através do método *flow_from_directory*, que carrega as imagens de treinamento a partir de um diretório especificado. É definido o tamanho das imagens no parâmetro target_size, quantas imagens vão ser carregadas por execução no batch_size e a classe de modo como *categorical*, indicando que as classes são representadas em formato one-hot encoding.

Figura 2 – Consumo do FER_2013



Fonte: Do autor.

Da mesma forma, o objeto *base_teste* é criado para carregar as imagens de teste a partir de um diretório *archiveMin/test* com as mesmas configurações de tamanho, batch e classe.

Em seguida, são criados os *datasets* usando o método *from_generator* do *tf.data.Dataset*. É fornecida uma função lambda que retorna o gerador de dados correspondente (base_treinamento ou base_teste) juntamente com os tipos de saída esperados (float32 para as imagens e float32 para as classes) e as formas de saída correspondentes *[None, 48, 48, 3]* para as imagens, indicando um tamanho flexível de *batch*, 48x48 *pixels* e 3 canais de cores, e *[None, 7]* para as classes, indicando um tamanho flexível de batch e 7 classes possíveis.

Por fim, os datasets ds_treinamento e ds_teste são configurados para repetição dos dados *repeat()* e pré-busca de elementos em buffer *prefetch()* para otimizar o desempenho. O valor *tf.data.experimental.AUTOTUNE* é utilizado para determinar automaticamente o tamanho do buffer de pré-busca com base na capacidade do sistema.

3.3 DENSENET

DenseNet, uma arquitetura CNN proposta recentemente, possui um padrão de conectividade interessante: cada camada é conectada a todas as outras dentro de um bloco denso. Nesse caso, todas as camadas podem acessar os mapas de

recursos de suas camadas anteriores, o que incentiva a reutilização pesada de recursos. Como consequência direta, o modelo é mais compacto e menos sujeito a overfitting. Além disso, cada camada individual recebe supervisão direta da função de perda por meio de caminhos de atalho, o que fornece supervisão profunda implícita. Todas essas boas propriedades tornam o DenseNet um ajuste natural para problemas de previsão por pixel (Huang *et al*., 2016, Tradução Nossa).

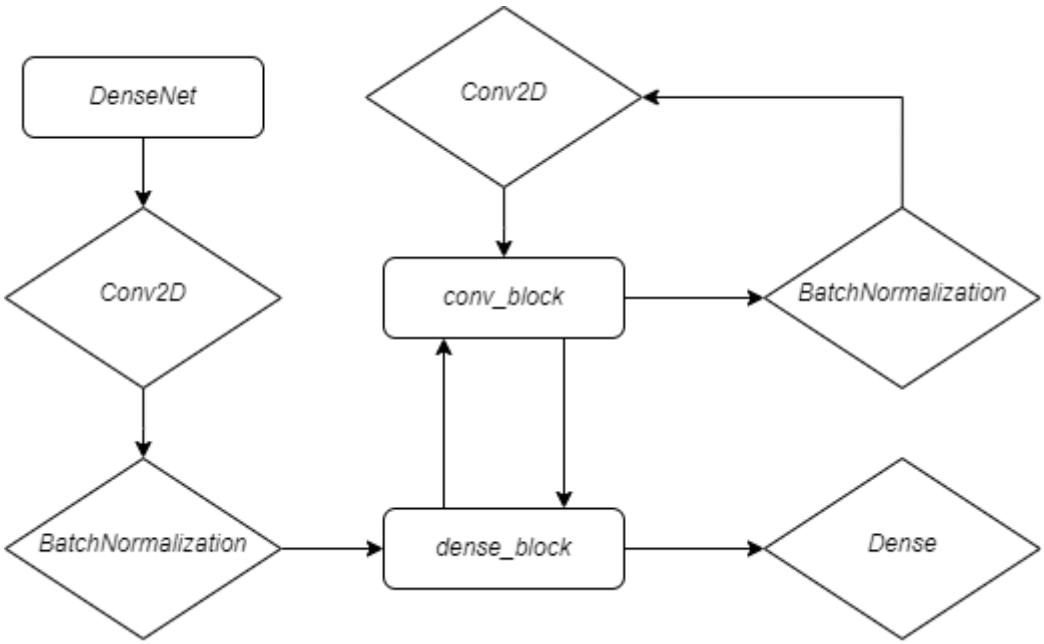
3.3.1 Arquitetura

Como pode ser visto na Figura 3, foram criadas três funções para a montagem da arquitetura, a função conv_block é responsável por definir um bloco de convolução. Ela recebe como entrada um tensor input e um número de filtros *filters*. Nessa função, é aplicada uma camada de normalização *BatchNormalization()* ao tensor de entrada. Ela normaliza os valores de ativação ao ajustar a média e a variância dos dados em mini-batches durante o treinamento, sendo seguida por uma camada convolucional *Conv2D* com os filtros especificados. A função retorna o tensor resultante.

A função *dense_block* define um bloco de conexão, que é composto por múltiplos blocos de convolução *conv_block*. Ela recebe como entrada um tensor input, o número de filtros *filters* e o número de camadas “num_layers”. Dentro dessa função, um loop é executado para criar várias camadas de convolução em sequência. A cada iteração do loop, um bloco de convolução é criado usando a função conv_block, e o resultado é concatenado com o tensor de entrada x. Ao final do loop, a função retorna o tensor resultante.

A função DenseNet é responsável por definir o modelo completo da DenseNet. Ela recebe o formato de entrada da imagem *input_shape* e o número de classes *num_classes* como parâmetros. Em seguida, é aplicada uma camada convolucional inicial com 64 filtros e ativação ReLU.

Figura 3 – DenseNet



Fonte: Do autor.

Após essa camada, é adicionado um bloco de conexão *dense_block* com três camadas de convolução. Em seguida, são adicionadas camadas de normalização *BatchNormalization* e uma camada de *pooling* global médio *GlobalAveragePooling2D* para reduzir as dimensões espaciais.

Após isso, são adicionadas camadas densas *Dense* com ativação ReLU e normalização. A camada de saída possui o número de neurônios correspondente ao número de classes, com ativação softmax para classificação multiclasse. Por fim, o modelo é compilado.

3.4 RESIDUAL NEURAL NETWORK

ResNet é atualmente a rede CNN melhorada mais amplamente utilizada e foi proposta no ILSVRC-2015. Os pesquisadores deste projeto aumentaram a profundidade da rede para melhorar a expressão e a capacidade de aprendizado do modelo no estágio inicial, desde a clássica estrutura de rede CNN AlexNet (sete camadas) até VGGNet-16 e VGGNet-19. No entanto, à medida que a rede se aprofunda, um gradiente desaparece e esse problema diminui a velocidade de convergência e o ACC da rede. O proponente do ResNet combinou o conceito de

representação residual com um modelo CNN para formar uma estrutura de bloco com aprendizado residual básico (Liu *et al*., 2021).

3.4.1 Arquitetura

Na Figura 4 tem-se a arquitetura da Resnet composta por duas funções, a principal *create_resnet* e a *resnet_block* que é encarregada de criar as camadas de convolução com seus atalhos que são uma das características da Resnet. A função *create_resnet* que tem como objetivo criar um modelo ResNet recebe dois parâmetros: *input_shape* e *num_classes*. Utiliza-se o *input_shape* para criação da camada de entrada, a partir daí, são adicionadas camadas convolucionais para realizar a extração de características. A primeira camada convolucional tem 64 filtros, um de kernel de (3, 3), um passo de (1, 1) e preenchimento (padding) *same*.

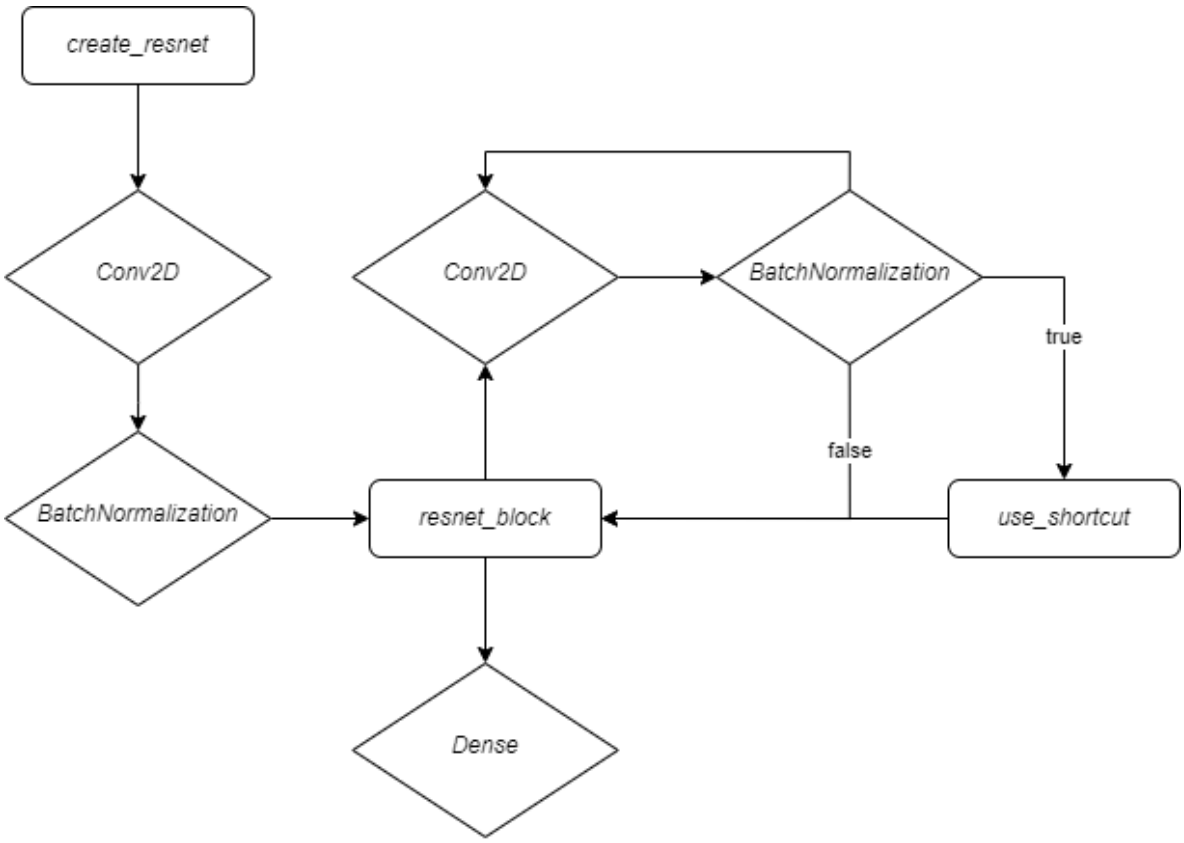
Em seguida, é aplicada *BatchNormalization()* e a ativação ReLU. O código continua com a definição do bloco ResNet. São adicionados 13 blocos representados por *resnet_Block()*, com diferentes configurações de filtros e estratégias de atalho *(shortcut)*. Cada bloco consiste em camadas convolucionais, normalização por lote e ativação ReLU.

Já a função *resnet_block* começa recebendo *inputs*, depois são aplicadas duas camadas convolucionais *Conv2D* em sequência. O parâmetro *strides* controla o movimento dos filtros. Ele define o número de *pixels* a serem pulados em cada direção (horizontal e vertical) ao mover o filtro durante a convolução.

A parte principal do bloco é a conexão de atalho *shortcut*. Se use_shortcut for verdadeiro, é criada uma conexão de atalho entre a entrada inputs e a saída da segunda camada convolucional. Isso é feito por meio de uma convolução com um kernel de tamanho (1, 1) e passo definido por *strides*, em seguida, é aplicada a normalização por lote à conexão de atalho. A saída da segunda camada convolucional e a conexão de atalho são somadas usando a função Add() do Keras.

Caso use_shortcut seja falso, a conexão de atalho é simplesmente a própria entrada inputs, que é somada à saída da segunda camada convolucional, por fim, é aplicada a ativação ReLU à saída final do bloco e o resultado é retornado pela função.

Figura 4 – Arquitetura ResNet



Fonte: Do autor.

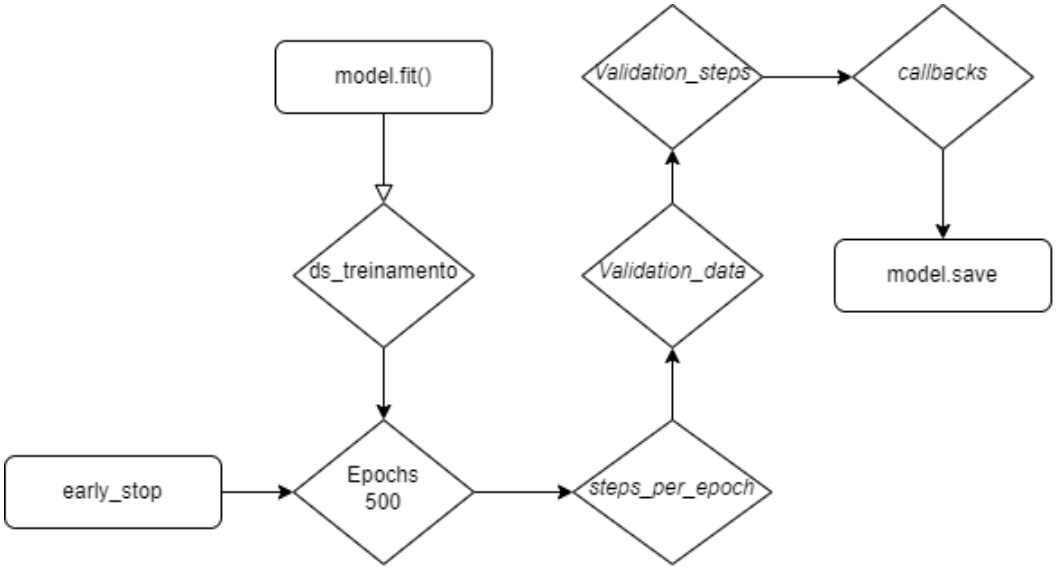
Após a sequência de blocos ResNet, é aplicada uma camada de pooling global médio *globalAveragePooling*() que em vez de usar camadas densas (fully connected) tradicionais para realizar a classificação, o Global Average Pooling calcula a média dos valores de ativação em cada mapa de características e retorna um único valor para cada canal. Finalmente, uma camada densa é adicionada com o número de neurônios igual a num_classes, seguida de uma função de ativação softmax para gerar as probabilidades de cada classe.

3.5 EXECUÇÃO

Aqui são realizadas as etapas finais do treinamento do modelo visto na Figura 5. Nele foi usando o Early Stopping e a gravação do modelo treinado em um arquivo HDF5.

Foi utilizado um “early_stop = EarlyStopping(monitor='val_accuracy', patience=30, verbose=1)”, técnica que interrompe o treinamento do modelo quando não ocorre melhoria na métrica de validação, fazendo com que ele pare a execução caso não haja uma melhoria com relação a maior acurácia das últimas épocas.

Figura 5 – Fluxo de treinamento.



Fonte: Do autor.

O comando “model.fit()” inicia o treinamento do modelo, aqui estão os parâmetros passados no mesmo:

1) Inicialmente é passado o datasets ds_treinamento;
2) *Epochs* define o número máximo de épocas;
3) O parâmetro *steps_per_epoch=steps_per_epoch* especifica o número de passos por época durante o treinamento;
4) *Validation_data=ds_teste* indica o *dataset* de validação usado para avaliar o desempenho do modelo;
5) *Validation_steps=len(base_teste)* especifica o número de passos a serem executados na validação;
6) O parâmetro *callbacks=[early_stop, save_epoch_params]* recebe uma lista de *callbacks* a serem usados durante o treinamento.
7) Por último o trecho “model.save('modelo/ denseNetMAX.h5')” salva o modelo treinado no formato HDF5.

4 RESULTADOS E DISCUSSÃO

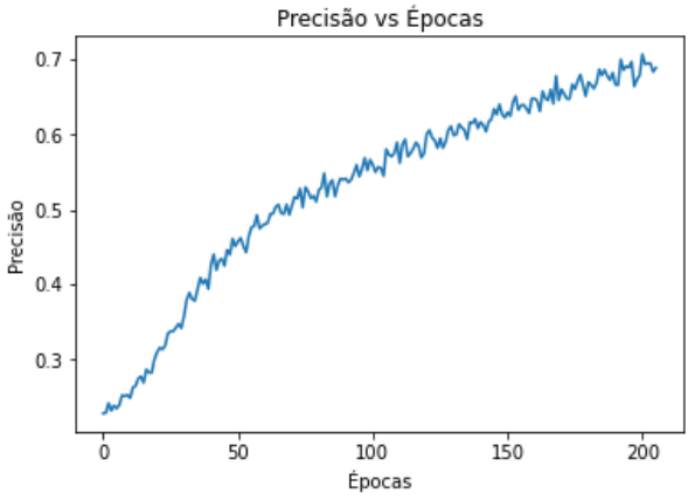
Na seção de Resultados e Discussões, serão apresentados os principais resultados obtidos a partir da análise das arquiteturas CNNs no reconhecimento de emoções.

4.1 ANÁLISE DOS RESULTADOS DENSENET

Ao analisar o desempenho da arquitetura DenseNet no treinamento com o conjunto de dados completo, pode-se observar no Gráfico 1 que a curva de aprendizado apresentou um desenvolvimento consistente logo após a primeira época, indicando que a DenseNet possui potencial para o reconhecimento de padrões em imagens.

Destaca-se que a DenseNet alcançou seu melhor resultado após um longo período de treinamento, totalizando 206 épocas, antes de ser interrompida pelo critério de parada definido pela técnica "EarlyStopping". Essa arquitetura demonstrou uma capacidade notável de aprender e melhorar ao longo do tempo, evidenciando sua eficácia no contexto do reconhecimento de emoções.

Gráfico 1 – DenseNet dataset completo.



Fonte: Do autor.

Tendo um resultado satisfatório, com uma precisão de 78% como pode-se observar na tabela 2 e a pior precisão em 22%.

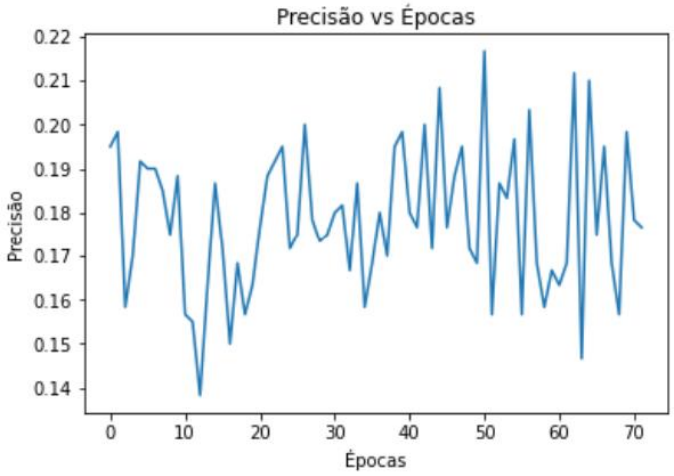
Tabela 2 – DenseNet resultados dataset completo.

Época	F1_score	Recall	Precisão	Loss	Time_Exec
0	0.1644	0.2021	0.2287	19525	332678387
200	0.1660	0.1737	0.7064	0.7893	43014864466

Fonte: Do autor.

Ao analisar o desempenho da arquitetura DenseNet no treinamento com um banco de dados reduzido, representado no Gráfico 2, pode-se observar uma estagnação que pode ter ocorrido devido à baixa carga de dados no treinamento ou a resolução das imagens ser muito baixa.

Gráfico 2 – Densenet dataset reduzido.



Fonte: Do autor.

Não se obteve melhorias significativas ao longo do processamento da CNN, e o desempenho estagnou desde a sua primeira rodada, rodando um total de 72 épocas, apresentando sua maior precisão nesse ponto. Neste caso, o melhor resultado obteve somente 21% de precisão, tendo-se uma queda acentuada em comparação com o banco completo.

Tabela 3 – DenseNet resultados dataset reduzido.

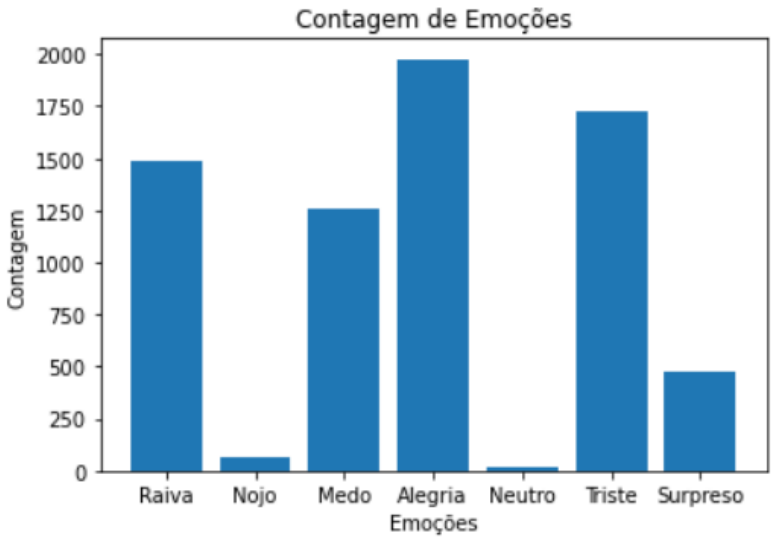
Época	F1_score	Recall	Precisão	Loss	Time_Exec
12	0.0958	0.1374	0.1383	18597	390127074
50	0.0720	0.1374	0.2166	18379	1492435638

Fonte: Do autor.

Para uma análise mais completa, foi criado um gráfico que demonstra a quantidade de acertos por emoção. Como pode ser observado no Gráfico 3, a alegria

alcançou a maior precisão entre todas, acompanhada pela tristeza e raiva, já a precisão mais baixa foi nojo e neutro.

Gráfico 3 – Precisão do modelo por classe.

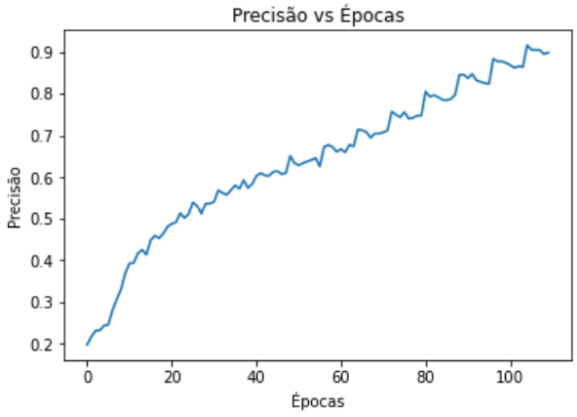


Fonte: Do autor.

4.2 Análise dos Resultados ResNet

Após a execução de um total de 110 epochs no banco completo, foi constatado que ResNet obteve um grande desempenho na curva de aprendizado. Como podemos ver no Gráfico 3, a curva de aprendizado da ResNet apresenta uma semelhança com a curva da DenseNet, porém com uma diferença no início do treinamento.

Gráfico 3 – ResNet dataset completo.



Fonte: Do autor.

Esses resultados reforçam a eficácia da arquitetura ResNet em lidar com problemas de reconhecimento de padrões em conjuntos de dados completos, destacando sua capacidade de aprender e generalizar informações relevantes. O melhor resultado obtido foi de 91% e o pior 19%, como pode-se observar na tabela 4.

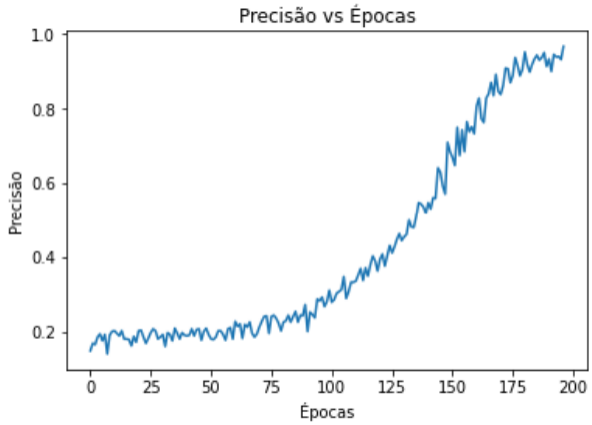
Tabela 4 – ResNet resultados dataset completo.

Época	F1_score	Recall	Precisão	Loss	Time_Exec
0	0.0877	0.1779	0.1976	20568	658748252
104	0.1685	0.1682	0.9159	0.2338	52595414972

Fonte: Do autor.

Já no banco reduzido, a arquitetura Resnet alcançou seu pico depois de executar 197 épocas, tendo um comportamento diferente, se manteve estável antes de iniciar sua curva de aprendizado que estabiliza próximo a 90%.

Gráfico 4 – ResNet dataset reduzido.



Fonte: Do autor.

Tendo seu melhor resultado com 96% de precisão, como visto na Tabela 5, bem semelhante ao banco de dados completo.

Tabela 5 – DenseNet resultados dataset reduzido.

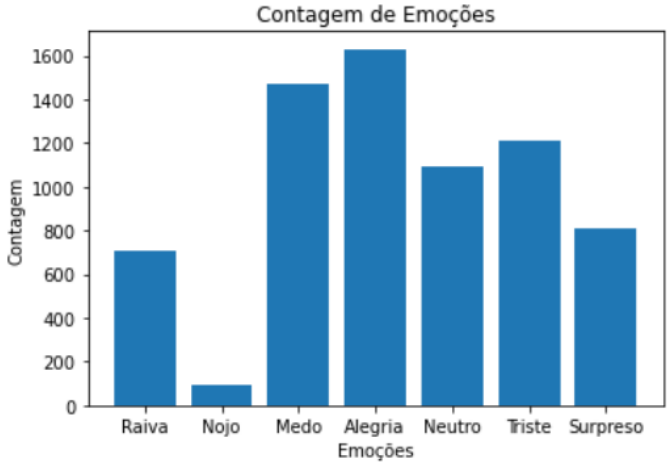
Época	F1_score	Recall	Precisão	Loss	Time_Exec
7	0.0716	0.1713	0.1416	19123	667472026
196	0.1381	0.1563	0.9666	0.1234	14602531899

Fonte: Do autor.

Com uma variação totalmente diferente da DenseNet a emoção com a maior precisão foi a felicidade, porém o nível de precisão nas demais é muito superior, além

de conseguir um maior alinhamento nas colunas com a exceção do nojo que não conseguiu bons resultados, de maneira semelhante a DenseNet.

Gráfico 5 – Precisão do modelo por classe.



Fonte: Do autor.

4.3 DISCUSSÃO

Após análise, foi observado que a arquitetura ResNet apresenta uma precisão de 91%, maior que os 83,86% demonstrada no artigo de Min Shi, Lijun Xu e Xiang Chen, mencionado na revisão da literatura. Essa diferença pode ser atribuída a vários fatores, incluindo a utilização de arquiteturas diferentes e a incorporação do algoritmo FCM (Fuzzy C-Means) na camada de convolução no estudo anterior. Portanto, é necessário conduzir novas pesquisas para validar essas métricas e investigar as razões subjacentes para a discrepância nos resultados.

5 CONCLUSÃO

Após analisar os resultados das arquiteturas de CNNs DenseNet e ResNet, em termos de desempenho e curvas de aprendizado, pode-se chegar a algumas conclusões.

A DenseNet demonstrou uma curva de aprendizado constante e um desempenho satisfatório no banco de dados completo, enquanto o reduzido não evoluindo ao longo das épocas, se mantendo com uma precisão baixa. A arquitetura DenseNet possui conexões densas entre as camadas, o que permite troca de

informações e o aproveitamento dos recursos durante o treinamento. Isso resulta em um aprendizado mais eficiente e uma capacidade maior de reconhecer padrões nas imagens.

No entanto, a arquitetura que obteve o melhor desempenho geral foi a ResNet. A curva de aprendizado da ResNet mostrou um crescimento mais acentuado no início, indicando uma rápida melhoria na precisão do modelo. Isso é resultado das conexões residuais utilizadas pela ResNet, que permitem um fluxo de informações mais suave e eficiente durante o treinamento. A ResNet demonstrou capacidade de aprender representações complexas desde o início, resultando em um melhor desempenho final.

Além disso foi observado durante os testes uma maior dificuldade na identificação de emoções como nojo e neutro. Onde ambas as arquiteturas não conseguiram se desenvolver nesse quesito. Já a alegria se destacou em ambas, sendo a emoção com o maior nível de acerto, dentre todas as sete emoções.

A ResNet mostrou-se mais eficaz entre as arquiteturas empregadas, seguida pela DenseNet, onde também se pode observar uma variação drástica no reconhecimento de cada uma das emoções. Essas conclusões destacam a importância da escolha adequada da arquitetura de rede neural convolucional, levando em consideração a complexidade do problema e o conjunto de dados disponível. Cada arquitetura possui suas características distintas que afetam diretamente seu desempenho e capacidade de aprendizado.

No que se refere a trabalhos futuros, pode-se explorar abordagens de ensemble, combinando várias arquiteturas de CNNs para obter um desempenho superior. Essa abordagem pode envolver a combinação de DenseNet e ResNet, bem como a inclusão de outras arquiteturas relevantes.

REFERÊNCIAS

Bengio, Y., LeCun, Y., and Hinton, G. (2021). Deep learning for AI. Communications of the ACM, 64(7):58–65.

Breuer, R., & Kimmel, R. (2017). A Deep Learning Perspective on the Origin of Facial Expressions. 1–16. http://arxiv.org/abs/1705.01842

FER_2013 Dataset. 2020. Disponível em: <https://www.kaggle.com/datasets/msambare/fer2013> Acesso em: 25 março 2023.

FER_2013 Reference Guide. 2023. Disponível em: <https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data>Acesso em: 17 maio 2023.

GÉRON, A.: Hands-On Machine Learning with Scikit-Learn & TensorFlow - Concepts, Tools, and Techniques To Build Intelligent Systems. 2017.

Huang Gao, Liu Zhuang, Kilian Q. Weinberger and Laurens Van Der Maaten, Densely Connected Convolutional Networks, 2016.

KERAS Reference Guide. 2023. Disponível em: <https://keras.io/about/>Acesso em: 17 maio 2023.

SABHARWAL, Ashish; SELMAN, Bart. Book review. Artificial Intelligence, [S.L.], v. 175, n. 5-6, p. 935-937, abr. 2011. Elsevier BV. http://dx.doi.org/10.1016/j.artint.2011.01.005.

T. Liu, T. Chen, R. Niu and A. Plaza, "Landslide Detection Mapping Employing CNN, ResNet, and DenseNet in the Three Gorges Reservoir, China," in *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 11417-11428, 2021, doi: 10.1109/JSTARS.2021.3117975.

TENSORFLOW Reference Guide. 2023. Disponível em: < https://www.tensorflow.org/about/bib?hl=pt-br> Acesso em: 17 maio 2023.

U. Srinivas, Y. Suo, M. Dao, V. Monga, and T. D. Tran, “Structured sparse priors for image classification,” IEEE Transactions on Image Processing, vol. 24, no. 6, pp. 1763–1776, 2015.