Estudo Sobre CNN e Reconhecimento de Imagem Utilizando o Dataset FER2013 e Arquitetura Baseada em VGG16 Simplificada

Murillo Henrique Pestana de Carvalho [*] and Alexandre da Silva Simões [§]

[*,§] *São Paulo State University (UNESP)*
Institute of Science and Technology of Sorocaba (ICTS)
Campus of Sorocaba, Brazil

Email: [*] *murillo.carvalho@unesp.br,* [§] *alexandre.simoes@unesp.br*

***Abstract*—In this work, the study of deep neural networks and the application of a simplified model based on VGG16 were carried out, with the objective of obtaining results of accuracy comparable with the average of the results obtained by the original participants of the FER2013 challenge, using this dataset without alterations (such as data augumentation). Due to the high training time, a model was created based on VGG16 in which the layers scheme was changed, removing the last three sets of convolutional layers and adding layers of batch normalization and dropout in order to speed up the training process and reduce the chances of model overfitting. The study included the experiment of 14 variations in the hyperparameters of the developed model and an experiment with the VGG16 for comparison purposes. Considering the time available for the production of this work, good results were obtained, as the achieved precision was above the average of the participants of the FER2013 challenge.**

***Keywords*-Supervised Machine Learning, Deep Neural Networks (CNNs), FER2013, VGG16.**

I. Introdução

O avanço da tecnologia e a crescente disponibilidade de dados têm impulsionado o desenvolvimento de algoritmos de aprendizagem de máquina em diversas áreas. Uma das aplicações mais relevantes é o reconhecimento de padrões em imagens, que tem se beneficiado do uso de técnicas avançadas de machine learning.

O reconhecimento de padrões em imagens é uma tarefa desafiadora devido à sua complexidade e à variabilidade dos dados. Dentre as abordagens utilizadas nesse contexto, destacam-se as redes neurais convolucionais (CNNs), que são capazes de aprender características visuais diretamente dos dados e extrair representações hierárquicas, permitindo o reconhecimento eficiente de objetos e padrões [1].

Um fator crucial no avanço dessa área é a realização de desafios públicos para a comunidade, nos quais uma base de dados é disponibilizada, com foco em um tema específico, onde os pesquisadores competem para obter a melhor performance com seus algoritmos e ganhar as premiações oferecidas. Esses desafios são importantes, pois estimulam o avanço dos algoritmos de aprendizagem de máquina. Além disso, após o término das competições, as bases de dados permanecem disponíveis para a comunidade, permitindo comparações de resultados e discussões subsequentes [2].

Um exemplo de desafio proposto foi o Facial Expression Recognition (FER2013), lançado em 2013 por Pierre-Luc Carrier e Aaron Courville na plataforma Kaggle. Essa competição disponibilizou uma base de dados com 35.887 imagens de rostos em escala de cinza, centralizadas e de tamanho 48x48 pixels, divididas entre sete categorias, contando com 4,002 imagens de expressão surpresa, 5,121 de medo, 4,953 de raiva, 6,198 de neutra, 6,077 de tristeza, 547 de aversão e 8,989 de alegria (como ilustrado na figura 1). O objetivo do desafio era obter a maior acurácia possível no conjunto de dados de teste para categorizá-lo nas sete expressões mencionadas [3].

Figure 1. Exemplos de imagens presentes em cada categoria do conjunto de dados FER2013.

Outro importante desafio é o ImageNet Large Scale Visual Recognition Challenge (ILSVRC), lançado pela Universidade de Stanford na plataforma ImageNet e teve inicio em 2010, com outras edições anualmente até 2017 [4]. Ele tinha por objetivo impulsionar o avanço da tecnologia de reconhecimento de objetos em imagens de larga escala. Na edição de 2014 desse desafio foi criada a arquitetura de CNN conhecida como VGG16, que teve destaque por seus resultados e tornou-se um ponto de partida popular para aplicações de reconhecimento de imagem [1].

Essas competições e desafios têm impulsionado significativamente o avanço da área de reconhecimento de imagens, promovendo o desenvolvimento de algoritmos mais eficientes e precisos para tarefas específicas de classificação de imagens.

II. Revisão Bibliográfica

1) Redes Neurais Artificiais (ANN): É composta por várias camadas de neurônios, incluindo uma camada com os dados de entrada, uma ou mais camadas ocultas (responsáveis pelo processamento da informação) e uma camada de saída. O Multilayer Perceptron (MLP) é uma arquitetura de ANN bastante comum em que os neurônios estão organizados em camadas consecutivas, com cada neurônio conectado a todos os neurônios da camada seguinte [5].

A figura 2 apresenta o esquema de um MLP e também o funcionamento de cada neurônio das camadas escondidas e de saída, que recebem entradas ponderadas (inicializadas com valores randômicos) onde são passadas por uma função de ativação que irá definir a saída do neurônio até chegar na saída do MLP, onde é calculado o custo (diferença entre o valor previsto e o real) para então calcular a correção a ser retropropagada da saída para os neurônios internos, possibilitando a melhora dos resultados à cada iteração [5].

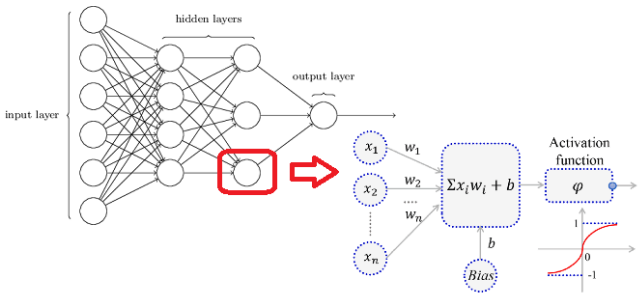

Figure 2. Esquema de uma rede neural artificial e o comportamento de cada neurônio

A função de ativação é responsável por introduzir a não linearidade no modelo, permitindo que a rede neural aprenda a representar relações complexas nos dados de entrada. A ReLU, $f(x) = \max(0, x)$, é uma função de ativação bastante utilizada, em que retorna o valor de entrada caso ele seja igual ou maior que zero e retorna o valor zero se o valor de entrada for negativo. O motivo de sua popularidade é a redução do problema de desaparecimento do gradiente da função custo, no qual os pesos das camadas iniciais são corrigidos cada vez mais lentamente, até um ponto em que a alteração não é significativa e o aprendizado da rede neural fica comprometido [6], [7].

2) Redes Neurais Convolucionais (CNN): As Redes Neurais Convolucionais (CNN), são uma combinação de camadas convolucionais (que serão discorridas a seguir neste trabalho) e camadas densas. Por conta disso as arquiteturas de CNNs costumam ter uma elevada quantidade de parâmetros livres, fazendo com que o tempo de treinamento seja grande. O tempo de treinamento irá depender de diversos fatores, como poder computacional, quantidade de dados de entrada e hiperparâmetros utilizados [5]. Alguns recursos para reduzir o tempo de processamento serão discutidos a seguir.

I. **Early Stopping**: monitora o desempenho do modelo em um conjunto de validação durante o treinamento e o interrompe quando as predições do modelo em relação ao conjunto de validação começarem a piorar. Esse método reduz o tempo de execução por não necessitar que todas as iterações solicitadas sejam cumpridas [5].
II. **Batch**: invés de atualizar os pesos sinápticos para cada dado de entrada, é possível reuni-los em lotes (batch) de forma que a rede neural processe os dados em paralelo e recalcule os pesos apenas após ter processado todos os dados do lote, reduzindo então a quantidade de vezes que o modelo precisará recalcular seus parâmetros [5].
III. **Otimizadores**: os otimizadores são algoritmos que ajustam os pesos e os bias da rede de forma iterativa durante o treinamento para minimizar a função de custo [5]. O otimizador Adam combina o método do gradiente descendente estocástico com o ajuste adaptativo da taxa de aprendizagem, permitindo que o algoritmo convirja mais rapidamente [8].
IV. **Batch Normalization**: adicionar uma camada de normalização dos lotes após camadas convolutivas e camadas densas pode ajudar a estabilizar e acelerar o processo de treinamento [9].
V. **GPU**: é possível utilizar a GPU para realizar os cálculos da DNN juntamente com o processador e acelerar seu treinamento.

As CNNs foram projetadas para extrair características relevantes de uma imagem e permitir identificar essa característica em qualquer local da imagem. Para isso a CNN utiliza de três processos.

I. **Convolução**: a convolução (representado na figura 3) é o conceito de aplicar um filtro aos pixels de uma região da imagem, multiplicando e somando os resultados. Essa operação é repetida em toda a imagem, resultando em uma nova imagem de tamanho reduzido chamado de mapa de caracteristicas. As CNNs podem ter várias camadas de convolução, cada uma com centenas de filtros, em que cada um extrai diferentes características, uma mais complexa que a anterior [5].
II. **Função de ativação**: nas CNNs a função de ativação é aplicada aos mapas de características gerados após a convolução, ativando ou desativando os neurônios com

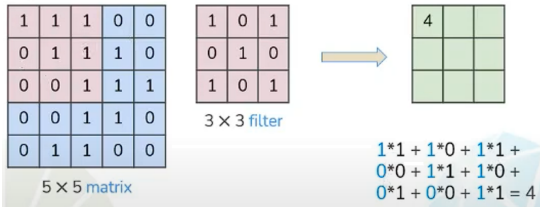

Figure 3. Demonstração do processo de convolução.

base em um determinado limiar. Isso ajuda a destacar as regiões de interesse nas imagens e permite que a rede aprenda padrões mais discriminativos [5].

III. **Polling**: essa operação tem como objetivo reduzir a dimensão dos mapas de características gerados pelas camadas convolucionais, buscando preservar as informações mais relevantes. A figura 4 retirada da aula sobre Deep Neural Networks (DNNs) do Prof. Dr. Alexandre da Silva Simões demonstra como o pooling é realizado, dividindo a imagem em regiões não sobrepostas e selecionando um valor representativo para cada região (no caso da figura foi o valor de máximo). O valor selecionado também pode ser a média (average pooling) ou o valor mais frequente (mode pooling) dentro da região [5].

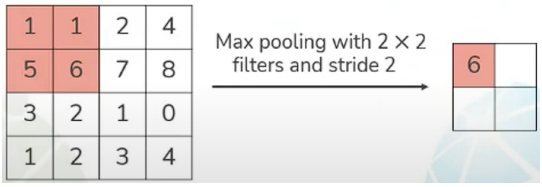

Figure 4. Demonstração do processo de Polling.

A. *VGG16*

A VGG16 é uma arquitetura de CNN conhecida por sua profundidade, com um total de 16 camadas, incluindo 13 camadas convolucionais e 3 camadas totalmente conectadas. Ela se destaca por utilizar um grande número de filtros convolucionais 3x3, o que permite uma representação mais rica de características dos dados de entrada. Além disso, ela utiliza o max pooling [10].

Uma das principais contribuições da VGG16 foi demonstrar que o aumento do número de camadas convolucionais leva a um melhor desempenho no reconhecimento de objetos. Ela alcançou resultados impressionantes na competição ImageNet, alcançando uma precisão de classificação de mais de 90% [4].

A VGG16 também tem sido amplamente utilizada como base para transfer learning [11], [12], onde os pesos pré-treinados dessa rede são aproveitados para outras tarefas de reconhecimento de imagens. Isso ocorre porque a VGG16 aprende características genéricas que são úteis para uma ampla variedade de tarefas, permitindo uma inicialização mais eficiente do treinamento em novos conjuntos de dados.

Apesar de seu sucesso, a VGG16 tem algumas limitações, como o alto consumo de recursos computacionais (como mostra a figura 5) e o aumento do risco de overfitting em conjuntos de dados menores devido à sua profundidade.

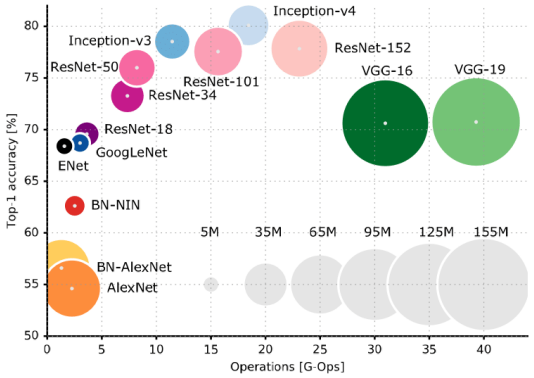

Figure 5. Comparação das principais arquiteturas de CNN em relação à acurácia atingida utilizando o conjunto de dados ImageNet

B. *FER2013*

Foram encontrados dois trabalhos que se encaixaram na proposta deste artigo, a utilização exclusiva do conjunto de dados FER2013 e utilização da arquitetura VGG16. Foi então estudado os parâmetros e técnicas utilizadas afim de identificar possibilidades de implementações que melhorassem os resultados obtidos.

I. **Emotion Recognition on FER-2013 Face Images Using Fine-Tuned VGG-16 [13]**: nesta pesquisa, empregou-se a arquitetura VGG16, utilizando os pesos pré-treinados do modelo treinado no conjunto de dados ImageNet, para a rede neural convolucional (CNN), enquanto que para a rede neural densamente conectada (ANN) foram utilizados valores aleatórios. Além disso, foram aplicadas técnicas de aumento de dados (data augmentation) para ampliar o tamanho do conjunto de treinamento. Foram explorados diferentes otimizadores, incluindo SGD, Adam e Swats, com taxas de aprendizado variando entre 0,01 e 0,001. Para agregar informações espaciais, utilizou-se a técnica de Global Average Pooling (GAP). Cada experimento foi treinado por um total de 30 épocas (iterações) para cada rede.

II. **Facial Emotion Recognition: State of the Art Performance on FER2013 [14]**: neste estudo, empregou-se uma variação da arquitetura VGGNet composta por quatro grupos de camadas convolucionais, cada grupo contendo duas camadas convolucionais, seguidas ao

final por três camadas densas. Para melhorar o desempenho e a generalização do modelo, foram aplicadas técnicas de aumento de dados (data augmentation). Cada experimento foi treinado por um total de 300 épocas, utilizando uma taxa de aprendizado de 0,0001 e um momento de 0,9. A busca automática por melhores hiperparâmetros foi realizada para otimizar o desempenho do modelo. Os otimizadores utilizados foram o SGD e o Adams. Além disso, foram incluídas camadas de batch normalization e Dropout para regularizar o modelo e evitar overfitting.

Ambos os trabalhos conseguiram atingir uma acurácia superior aos 69% no conjunto de testes, sendo que [14] atingiu 73%, um valor ainda superior ao atingido pelo vencedor do desafio na época, provando ser boas fontes de informação para basear este artigo. Apesar disso, por conta do tempo disponível para a elaboração deste estudo ter sido pequeno, não foi possível aplicar técnicas de auto tuning, o GAP, as técnicas de aumento de dados, nem executar centenas de épocas por teste.

III. Abordagem proposta

Como o autor não tem definido um tema de manuscrito do programa de pós-graduação, não foi possível aproveitar este trabalho para avançar com a entrega do trabalho final. Tendo isso em mente, buscou-se por um conjunto de dados de médio porte e que apresentasse um tema que despertasse o interesse do autor. A escolha por um conjunto de dados de médio porte se deu por conta do equilíbrio entre o elevado tempo de treinamento para conjuntos de dados de grande porte e a dificuldade de se obter bons resultados a partir de conjunto de dados pequenos. Outro fator importante na escolha da proposta foi o reduzido tempo disponível para a produção deste estudo.

Uma vez validado que o tempo de execução de um modelo com duas camadas convolutivas e três camadas densas no Google Colab foi de 50 minutos para o conjunto de dados MNIST, composto por 60.000 imagens de tamanho 28x28 pixels, acreditou-se que o FER2013 com pouco mais de 35.000 imagens de tamanho 48x48 seria aceitável, já que possui aproximadamente três vezes o número de pixels total que o MNIST tem.

Tendo escolhido o conjunto de dados, iniciou-se a busca por arquiteturas sequenciais de CNNs (por conta da baixa complexidade para compreendê-las) que apresentassem bons resultados. Após identificar a VGGNet e verificar que com a quantidade de camadas que o menor de seus modelos possui, concluiu-se que não seria possível utilizar todas as camadas de nenhum modelo da VGGNet, por conta da quantidade de parametros livres (onde a VGG16 original possui 138 milhões de parâmetros livres e a VGG16 utilizando apenas um canal de cor possui aproximadamente 34 milhões) e pelo FER2013 ser um conjunto de dados menor que o ImageNet e para utilizá-lo sem aumentar seus dados pode causar overffiting com modelos da VGGNet completos.

A meta traçada para os resultados foi a de obter uma acurácia suficientemente grande para me classificar entre os 30 melhores participantes do antigo desafio FER2013 num total de 56 grupos participantes (sendo que o 30º pontuou 0.49289 de acurácia para o conjunto de teste), por estar próximo à média da pontuação dos participantes.

IV. Materiais e métodos

O conjunto de dados FER2013 foi obtido através da plataforma Kaggle [15] no formato de imagens JPG. Para o desenvolvimento do código utilizou-se o PyCharm e o desenvolvimento foi realizado em Python em ambiente local.

Primeiramente o conjunto de dados foi repartido entre conjunto de treino, validação e teste, sendo que seus tamanhos foram variados ao longo das execuções (menos o conjunto de teste que foi fixado em 20% do conjunto total). A partir das imagens foi extraído o valor de cinza dos pixels (de 0 a 255) e o label da saída através da biblioteca do Keras (tf.keras.utils.image_dataset_from_directory), em seguida os valores dos pixels de todos os conjuntos de dados foram normalizados (divididos por 255).

Para a construção do modelo deste artigo utilizou-se a sequência de camadas demonstradas pelo esquema da figura 6, onde todas as camadas convolutivas possuem as caractísticas listadas abaixo, sendo que apenas o número de filtros é alterado utilizando os valores 32, 64, 128 e 256 (do filtro mais próximo da camada de entrada ao mais próximo das camadas densas).

I. kernel_size = 3x3;
II. strides = 1;
III. activation = ”relu”;
IV. padding = ”same”

Foi adotado o método de pooling máximo, com um tamanho de janela de 2x2 e um passo de 2 pixels, mantendo o preenchimento (”same”). Após cada saída da camada convolucional e camada densa, foi aplicada uma camada de normalização de batch e inserida uma camada de dropout após cada pooling máximo e camada densa. A função de ativação utilizada em todas as camadas com exceção da camada densa de saída foi a ReLU. A camada densa de saída utilizou a função softmax para apresentar a porcentagem de combinação com as sete saídas esperadas pelo modelo, correspondentes às diferentes expressões faciais. Por fim, foram inseridas duas camadas densas escondidas com 512 neurônios cada, utilizando o otimizador Adam e o custo de validação foi calculado utilizando a *sparse categorical crossentropy*. Para fins de comparação, a figura 7 apresenta o esquema do modelo original da VGG16.

Foram conduzidos 14 experimentos de treinamento utilizando o modelo proposto (VGG*), bem como um experimento de treinamento utilizando o modelo VGG16. Cada treinamento foi realizado com diferentes configurações de

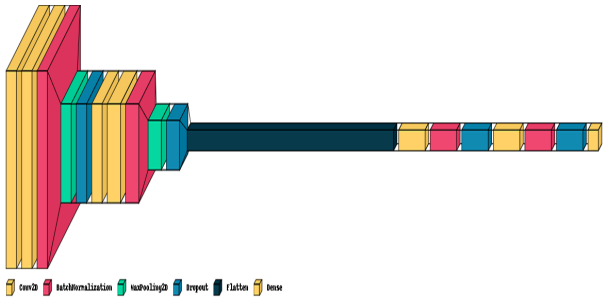

Figure 6. Esquema de camadas do modelo criado a partir da VGG16

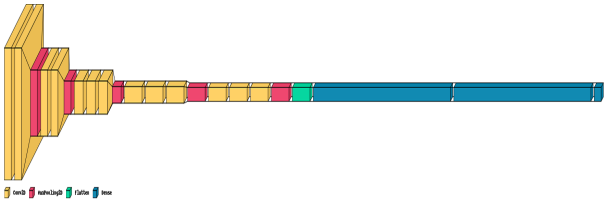

Figure 7. Esquema de camadas do modelo original da VGG16

hiperparâmetros, buscando manter ao menos dois treinamentos que diferem entre sí por apenas um parâmetro (dessa forma permitindo a comparação dos resultados de teste), nos quais as seguintes variáveis foram alteradas: o número máximo de épocas, a utilização ou não de Early Stopping (ES), a tolerância do Early Stopping (Tol), a taxa de aprendizagem (LR), a proporção dos dados de validação em relação ao conjunto total (Val Part) e a taxa de dropout. A tabela I compila os valores de todos os hiperparâmetros utilizados em cada treinamento.

Table I
PARÂMETROS UTILIZADOS EM CADA TREINAMENTO EXECUTADO

ID	Modelo	Épocas	Tol	ES	BS	LR	Val Part	Dropout
1	VGG*	38	-	False	64	0.0001	0%	0.5
2	VGG*	50	10	True	64	0.00001	8%	0.25
3	VGG*	50	10	True	64	0.00005	20%	0.25
4	VGG*	50	15	True	64	0.0001	8%	0.3
5	VGG*	50	15	True	64	0.0001	8%	0.5
6	VGG*	50	15	True	64	0.0001	20%	0.3
7	VGG*	50	15	True	64	0.00001	20%	0.3
8	VGG*	50	20	True	64	0.00002	20%	0.3
9	VGG*	50	-	False	64	0.0001	20%	0.3
10	VGG*	60	-	False	64	0.00001	20%	0.3
11	VGG*	100	-	False	64	0.00001	20%	0.3
12	VGG*	100	15	True	64	0.00001	20%	0.3
13	VGG*	50	15	True	64	0.00001	20%	0.5
14	VGG*	50	-	False	64	0.0001	20%	-
15	VGG16	50	-	False	64	0.0001	20%	-

Após cada treinamento, o modelo aprendido foi utilizado para realizar predições sobre as 7.178 imagens do conjunto de testes. A partir dos resultados foi registrado a acurácia, as médias macro de precisão, sensibilidade e F1 score, o número de épocas necessárias para concluir o treinamento e o tempo total em minutos, bem como duas matrizes confusão, sendo que a segunda é uma normalização da primeira.

V. RESULTADOS E DISCUSSÕES

Os dados provenientes dos treinamentos executados foram disponibilizados na tabela II, onde o ID identifica o número do treinamento realizado, e as demais colunas identificam as métricas de acurácia, precisão, sensibilidade e F1 score (medidas em porcentagem), o número de épocas e o tempo de treinamento (medido em minutos).

Table II
MÉTRICAS DAS PREDIÇÕES DO MODELO SOBRE O CONJUNTO DE TESTE

ID	Acurácia	Precisão	Sensibilidade	F1 Score	Épocas	Tempo
1	61.35	59.73	59.05	58.05	38	139
2	55.82	52.26	52.06	50.70	42	148
3	53.26	48.71	48.76	46.78	22	65
4	56.52	52.13	51.95	49.92	24	80
5	59.19	56.31	56.17	54.58	38	125
6	54.67	52.97	51.11	50.46	26	75
7	53.44	49.66	48.74	47.77	50	144
8	56.81	53.41	53.72	52.30	50	152
9	57.73	54.71	54.50	53.04	50	143
10	56.52	54.48	52.92	51.98	60	184
11	57.82	55.27	54.49	53.29	100	304
12	54.69	51.16	50.36	49.23	70	209
13	33.20	23.61	26.03	21.81	19	58
14	52.79	50.45	49.93	48.78	50	124
15	56.83	54.25	52.45	51.92	50	353

Ao excluir o treinamento 13, a variação da acurácia foi pequena (aproximadamente 13.20%), onde um motivo para isso poderia ser que as alterações nos hiperparametros não foram agressivas. O treinamento 13 teve um resultado muito abaixo dos demais por conta da baixa taxa de aprendizagem em conjunto com o alto dropout, de maneira que a tolerância de 15 épocas não foi suficiente para alcançar a convergência.

Quanto ao tempo de execução, assim como o previsto, o maior de todos foi o treinamento utilizando o VGG16, sendo que no treinamento 11 foi executado o dobro de épocas, mas mesmo assim o tempo foi menor.

Quanto ao early stopping, ao comparar os treinos 6 e 9, podemos ver uma vantagem nas métricas do modelo que não utilizou early stopping, o que indica que a tolerancia de 15 épocas não foi suficiente e fez com que o modelo sofra com underfitting. Nesse caso seria necessário realizar novos experimentos com um número de tolerância maior para identificar o ponto ótimo para o early stopping.

Em relação ao tamanho do conjunto de validação, comparando os treinos 1 e 5, em que apenas esse parâmetro foi alterado (inclusive foram inserido o número de 38 épocas no treino 1 para possibilitar essa comparação), constata-se que para o modelo e conjunto de dados utilizados, quanto menos dados dedicada-se ao conjunto de validação, maior será as métricas de teste (tendo em mente que o número ótimo de épocas à serem treinadas é conhecido). Isso indica que o modelo está sofrendo com a falta de dados de entrada para

aprender melhor as características que melhor classificam os dados.

Os treinos 6 e 7 reafirmam o ponto colocado sobre o early stopping, pois como foi identificado que a tolerância estava muito baixa, ao aumentar a taxa de aprendizagem o modelo dará pulos maiores nas correções de seus pesos, possibilitando alcançar resultados que só seriam possíveis se a tolerancia do early stopping fosse maior.

Se compararmos o treino 6 com o 9, ou o 10 com o 11, podemos ver que em ambas as situações os treinos que não utilizaram early stopping e estão se especializando em seus dados de treino e perdendo a capacidade de generalizar, provando a utilidade do early stopping.

Finalmente, quando comparamos os treinos 15 e 16, onde o treino 15 foi especialmente executado sem as camadas de normalização de batch e nem dropout afim de permitir uma comparação fiel ao modelo VGG16, podemos ver que o resultado da VGG16 é melhor, assim como o esperado pela literatura. Porém assim como já foi dito anteriormente, o tempo de execução foi o fator limitante para que fosse acrescentado recursos à VGG16 ao invés de removê-los.

A figura 8 apresenta a matriz confusão do treinamento 1, onde podemos verificar os principais falsos positivos e falsos negativos previstos pelo modelo.

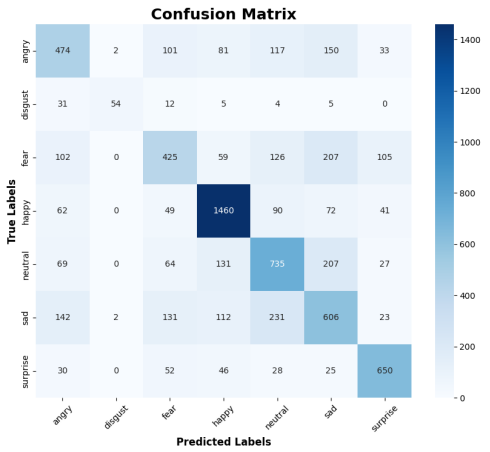

Figure 8. Matriz confusão resultante da predição dos dados de teste do treinamento 4

Com o objetivo de facilitar o compreendimento dos números apresentados na matriz confusão da figura 8, foi feita a normalização dos dados e apresentados em formato de porcentagem (sendo que os 100% se dá pela soma das porcentagens do eixo dos labels verdadeiros) pela figura 9.

As duas situações com maior índice de erro foi a previsão ”raiva” quando deveria ser ”aversão” e a previsão ”tristeza” quando deveria ser ”medo”. A figura 10 mostra exemplos das duas classes de cada uma dessas situações em que é possível validar as semelhanças entre as duas classes envolvidas em cada situação.

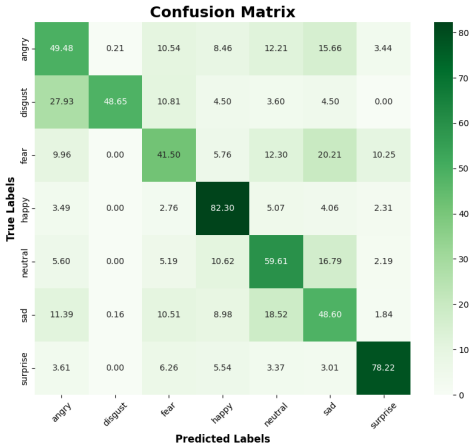

Figure 9. Matriz confusão resultante da predição dos dados de teste do treinamento 4 de forma a mostrar seus resultados em porcentagens.

Figure 10. Exemplos de proximidade nas expressões faciais em que o algoritmo apresentou maior erro.

Apesar da classe de aversão possuir bem menos exemplares que as demais, obteve 48,65% de acerto das entradas e com isso não foi a pior classificada, como era de se esperar quando há desbalanceamento na quantidade de dados entre classes, invés disso, foi a classe ”medo” com 41,50%. É possível que o motivo seja que a classe de aversão seja relativamente mais fácil de classificar corretamente por conta de características marcantes nas imagens, que facilitam a aprendizagem do modelo. Esse ponto é reforçado ao observar a classe ”surpresa”, que após a ”aversão” é a classe com menor número de dados, porém apresentou o segundo melhor resultado (acertando 77,62% das entradas).

Um fator que atrapalha o modelo a atingir resultados melhores são os erros de rotulagem nas categorias de expressão. A figura 11 apresenta duas imagens de cada classe extraídas do FER2013, onde uma está bem rotulada e a outra não. Por mais que as expressões faciais possam variar de pessoa para pessoa, os exemplos mal rotulados atrapalham o modelo a aprender corretamente quais características dos mapas de caracteristicas aprendidos realmente definem aquela classe, piorando sua performance.

VI. Conclusões

Classificar expressões faciais é uma tarefa que em alguns casos pode ser difícil por conta da proximidade dos gestos

Figure 11. Exemplos de outliers nas classes do FER2013

do corpo humano ao passar por experiências que motivam essas expressões. Esse é um dos fatores pelos quais os melhores resultados de estudos com o FER2013 atingem a casa dos 70% enquanto que com o MNIST, por exemplo, aproxima-se dos 100%. Para aumentar a dificuldade dessa tarefa, o FER2013 apresentou diversos casos de imagens mal rotuladas, gerando outliers na aprendizagem e afetando os resultados dos testes.

Com apenas 14 experiências de ajuste dos hiperparâmetros do modelo criado neste trabalho já foi possível alcançar uma acurácia de 61,35%, superior à acurácia de 56,83% obtida através da VGG16, e necessitando de apenas pouco mais de 1/3 do tempo de treinamento. Isso demonstra que o modelo sugerido é promissor, principalmente quando comparado com os 73% da bibliografia estudada, que utilizou de aumento de dados, GLP, busca automática por melhores hiperparâmetros, pré-treino, diferentes otimizadores, entre outras técnicas que neste trabalho não foi possível abordar, mas que poderia melhorar os resultados.

A partir da comparação entre os treinamentos 1 e 5, é possível verificar a necessidade da aplicação de aumento de dados no FER2013, uma vez que foi alterado apenas a parcela de dados separada para a validação, de forma que o treinamento 1 tinha mais dados de entrada e obteve melhores resultados.

Por fim, a simplificação do modelo VGG16 para um modelo com dois grupos de camadas convolucionais foi uma necessidade por conta do tempo de execução do modelo original que impossibilitaria este estudo, porém a bibliografia demonstrou que os melhores resultados foram obtidos ao utilizar um modelo com quatro conjuntos de camadas ou até mesmo a VGG16 completa. Isso indica que para ser capaz de extrair características suficientemente complexas de forma a distinguir emoções que a primeira vista são bastante semelhantes, é necessário utilizar mais do que dois conjuntos de camadas convolucionais.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon, "Video and image based emotion recognition challenges in the wild: Emotiw 2015," in *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, 2015, pp. 423–426.

[3] W. C. Y. B. Dumitru, Ian Goodfellow, "Challenges in representation learning: Facial expression recognition challenge," 2013. [Online]. Available: https://kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challeng

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[6] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[7] Z. Hu, J. Zhang, and Y. Ge, "Handling vanishing gradient problem using artificial derivative," *IEEE Access*, vol. 9, pp. 22 371–22 377, 2021.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 448–456.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2014, pp. 2402–2410.

[11] S. Zekić, O. Mađar, M. Ristanović, and D. Sejdinović, "Transfer learning with vgg16 neural network for melanoma classification," in *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2019, pp. 221–225.

[12] M. Asim, S. I. A. Shah, S. U. Khalid, M. Sharif, and H. Yasmin, "Transfer learning with deep convolutional neural network for liver fibrosis staging on histopathological images," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2018, pp. 469–472.

[13] I. G. P. Kusuma Negara, J. Jonathan, and A. Lim, "Emotion recognition on fer-2013 face images using fine-tuned vgg-16," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, pp. 315–322, 2020.

[14] Y. Khaireddin and Z. Chen, "Facial emotion recognition: State of the art performance on FER2013," *CoRR*, vol. abs/2105.03588, 2021. [Online]. Available: https://arxiv.org/abs/2105.03588

[15] M. Sambare, "Fer-2013 learn facial expressions from an image," https://www.kaggle.com/datasets/msambare/fer2013, 2020, acessado em: 01/07/2023.