

Aprimorando a Classificação de Resistores Utilizando Aprendizado Profundo

Gabriel Montagni Domingues Filho
Nº USP 11800903

Ian Zaniolo Sirbone
Nº USP 4735640

Leonardo Zaniboni Silva
Nº USP 11801049

Lucas Daudt Franck
Nº USP 11845726

William Carrara Orlato
Nº USP 11800991

Resumo: Resistor é um componente eletrônico fundamental e amplamente utilizado em circuitos elétricos. O padrão de marcação dos resistores é baseado em um código de cores que precisa ser identificado para a correta leitura do valor do componente. Nesse escopo, a visão computacional é a área mais utilizada para inferir a resistência a partir das cores. Entretanto, a maioria das soluções existentes precisam que as peças estejam alinhadas e com iluminação uniforme para funcionar. Sendo assim, a lacuna presente na literatura foi abordada utilizando um algoritmo de aprendizado profundo para classificar os resistores. Nossa abordagem usa um *dataset* com diversas imagens, condições de iluminação e ângulos, e compara as arquiteturas *Inception-V3*, *EfficientNetV2-S* e *ResNet50*. Os experimentos apresentaram 91%, 87% e 77% de precisão, respectivamente. Com isso, conclui-se que a arquitetura *Inception-V3* lidou melhor com a tarefa apresentada e obteve os melhores resultados.

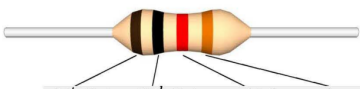
Palavras-chave: visão computacional, aprendizado profundo, classificação de resistores axiais.

I. INTRODUÇÃO

A aplicação da visão computacional no controle de qualidade de processos oferece uma abordagem não-invasiva para a inspeção de peças e componentes. Essa tecnologia acelera a detecção de falhas, reduz a necessidade de intervenção manual por parte dos operadores, e elimina a subjetividade da avaliação. Tais benefícios resultam em maior consistência e confiabilidade na detecção de defeitos, facilitando a rastreabilidade e a documentação, o que contribui para a melhora da qualidade dos produtos [1].

Resistores são componentes passivos utilizados para limitar a corrente elétrica em circuitos eletrônicos. Esses componentes são comercializados nos formatos *surface-mount technology* (SMT) e *through-hole technology* (THT), e estão presentes em praticamente todos circuitos discretos. Os resistores do tipo *THT* são construtivamente maiores que os *SMT* e fabricados com filme de carbono ou filme metálico, apresentando formato cilíndrico axial com faixas coloridas para identificação do valor de resistência [2]. A Figura 1 ilustra um resistor *THT* de 4-faixas juntamente com o código de cores usado na marcação.

O valor nominal da resistência do componente é calculado por $R = (100A + 10B) \times C$, onde A é o valor do 1º dígito, B o valor do 2º dígito, e C o fator multiplicativo (vide Figura 1). A última faixa do resistor indica a tolerância do



	1 st digit	2 nd digit	multiply	tolerance
Black	0	0	1	1% (F)
Brown	1	1	10	2% (G)
Red	2	2	100	
Orange	3	3	1K	
Yellow	4	4	10K	
Green	5	5	100K	0.5% (D)
Blue	6	6	1M	0.25% (C)
Violet	7	7	10M	0.1% (B)
Grey	8	8	100M	0.05% (A)
White	9	9	1G	
Gold			0.1	5% (J)
Silver			0.01	10% (K)
None				20% (M)

Figura 1: Código de Cores para Resistor de 4-faixas [1].

componente em percentual. Também são comuns resistores com 5-faixas, nos quais uma terceira cor é adicionada entre o 2º dígito e o fator multiplicativo, representando o 3º dígito de peso unitário. Os resistores também variam quanto a potência máxima suportada, parâmetro que reflete no tamanho físico do componente, não sendo codificado através das cores.

A classificação de resistores com base no código de cores é uma área de pesquisa pouco explorada. O processamento de imagens é a principal abordagem das publicações na área, sendo exploradas técnicas de filtragem, segmentação, extração das faixas e identificação das cores [3], [4], [5], [6], [7]. Alguns trabalhos utilizam visão computacional em conjunto com uma estrutura física dedicada para aquisição das imagens [2]. Outros autores apresentam uma solução baseada em otimização por colônia de formigas (*ACO*) [8]. Também existem abordagens com processamento de imagem combinado com aprendizado de máquina [1], [9].

No entanto, os trabalhos presentes na literatura focam principalmente em técnicas de processamento de imagem para a resolução do problema, mas ainda não estudaram de modo profundo o uso de redes convolucionais para classificação de resistores. Sendo assim, o presente trabalho tem como proposta a utilização de redes neurais convolucionais para a classificação de resistores, testando as arquiteturas de redes *Inception-V3* [10], *EfficientNetV2-S* [11], e a *ResNet50* [12].

Para tal, um conjunto de imagens de resistores (*dataset*) [13] com 37 classes, e fotos obtidas nos mais variados ângulos, posições e condições de iluminação foi utilizado no treinamento e validação das redes.

O restante deste trabalho está organizado da seguinte forma: Seção II apresenta a definição do problema. Seção III demonstra detalhes acerca da nossa abordagem. Os resultados e análises são apresentados nas seções IV e V, respectivamente. As conclusões estão presentes na seção VI.

II. DEFINIÇÃO DO PROBLEMA

O sistema de identificação de resistores axiais emprega faixas coloridas pintadas no corpo do componente para informar o valor de resistência. São utilizadas 12 cores distintas para essa finalidade, sendo elas: preto, marrom, vermelho, laranja, amarelo, verde, azul, violeta, cinza, branco, ouro e prata. Várias dificuldades são encontradas para a identificação das faixas coloridas como:

a) *Posicionamento*: resistores podem estar em qualquer posição e orientação. A ordem de leitura das faixas é essencial para a identificação do componente [2].

b) *Variação da iluminação*: a forma cilíndrica não uniforme dos resistores faz com que surjam reflexos e halos que distorcem as cores da imagem [4].

c) *Variação da pigmentação*: os pigmentos utilizados para tingir as faixas e o revestimento dos componentes podem variar entre fornecedores [4].

d) *Gama de cores*: pigmentos como amarelo e dourado apresentam matizes semelhantes, variando apenas o brilho. Mesma situação ocorre com as cores cinza e prata [4].

Demir *et al.* [3] propõem um sistema de identificação em tempo real desenvolvido para plataforma *Android*. O algoritmo pré-processa a imagem com filtro de mediana e transforma o resultado para o espaço *HSV*. Na sequência, as faixas coloridas são identificadas e seus respectivos valores encontrados com uma *look-up table* criada a partir de uma análise estatística de um banco de imagens. O sistema proposto tem uma taxa de erro de 8%, mas só classifica resistores que estejam adequadamente alinhados com o gabarito presente na tela do aplicativo, o que limita o escopo de aplicações do método.

Muminovic e Sokic [1] apresentam um sistema de detecção que combina processamento de imagem e aprendizado de máquina para classificar múltiplos resistores em uma foto. O algoritmo realiza a segmentação dos resistores na imagem, aplica operações morfológicas, e alinha os componentes horizontalmente. As faixas são identificadas espacialmente através de análise estatística de posição. Por fim, as cores são classificadas utilizando *Support Vector Machine*, treinada com um *dataset* criado pelos autores. O sistema proposto é limitado a resistores de potência $1/4W$ da série comercial E12. O método apresentou taxa de acerto média de 86%, com os piores resultados para resistores com faixas espaçadas fora dos intervalos definidos na análise estatística.

Li *et al.* [9] desenvolveram um sistema de classificação que trabalha com imagens em preto e branco adquiridas por uma

câmera industrial. O trabalho utiliza uma etapa de processamento de imagem com filtro de ruído e o algoritmo *Retinex* para minimizar efeitos de iluminação não uniforme, e uma rede neural treinada com *backpropagation* para a classificação das cores. Os autores testaram a identificação de cores do sistema utilizando amostras artificiais e atingiram uma taxa de acerto de 87,9%. Nenhum dado de desempenho foi informado em relação ao uso do sistema para a classificação de resistores.

Os estudos citados na literatura apresentam algumas abordagens para classificar resistores com base no código de cores. Entretanto, verifica-se uma falta de soluções que utilizam aprendizado de máquina, com as publicações existentes limitadas quanto ao escopo de aplicação. Nosso trabalho propõe uma estratégia diferente: utilizar um sistema de classificação e aprendizado profundo. Com isso, queremos lidar com os problemas de posicionamento, gama de cores, variação de iluminação e pigmentação, e tornar a tarefa de classificar resistores mais genérica, contribuindo com o desenvolvimento de um trabalho mais adaptável, robusto e que utiliza técnicas pouco exploradas.

III. SOLUÇÃO DO PROBLEMA

Como mencionado, este estudo teve como objetivo avaliar e comparar a eficácia de três modelos na tarefa de classificar resistores axiais pelo código de cores. O primeiro sistema foi baseado em uma arquitetura *Inception-V3*, o segundo em uma *EfficientNetV2-S* e o terceiro em uma *ResNet50*. Sendo assim, as subseções a seguir apresentam informações acerca do *dataset*, modelos das redes, estratégia de treinamento, validação e *setup* experimental utilizado.

A. Dataset

O *dataset* original é formado por 2778 imagens de resistores separadas em 37 classes segundo valor de resistência e potência. Apesar da disponibilidade dos dados, a informação de potência dos resistores foi desconsiderada na abordagem do trabalho por não ser um parâmetro codificado pelo código de cores. Com isso, as imagens foram agrupadas somente pelo valor comercial de resistência resultando em 31 classes. A Figura 2 apresenta a distribuição de classes ao longo do *dataset*. A Figura 3 ilustra exemplos de fotos de resistores presentes no conjunto de dados.

B. Modelos de IA

O problema foi tratado como uma tarefa de classificação, sendo cada valor comercial de resistor presente no *dataset* considerado como uma classe (vide Figura 2). Para essa abordagem, decidiu-se por avaliar tanto arquiteturas clássicas de redes convolucionais, como *Inception-V3* e *ResNet50-V2*, quanto uma arquitetura mais recente, a *EfficientNetV2-S*. Os modelos foram inicializados com os pesos pré-treinados no *dataset* ImageNet [14], um conjunto de dados amplamente utilizado para pré-treinar modelos de aprendizado profundo. A última camada dos modelos foi removida e substituída por uma camada densa contendo 31 neurônios e função de ativação sigmoide. Os parâmetros de teste foram ajustados no

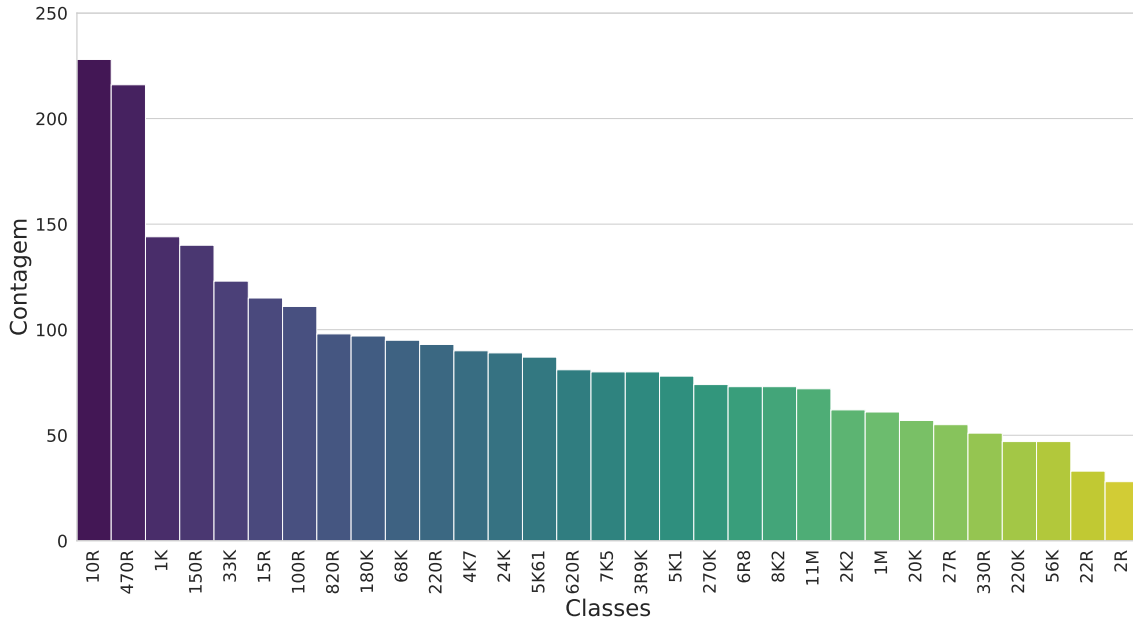


Figura 2: Distribuição de classes no *dataset*.



a) Imagem de um resistor de $2k2\Omega$. Ressalta-se a dificuldade de identificar a quarta faixa do componente devido à pouca iluminação ambiente.



b) Imagem de um resistor de $270k\Omega$. Destaca-se o impacto de reflexos e iluminação não uniforme na leitura das faixas presentes no corpo do componente.

Figura 3: Exemplos de imagens de duas classes de resistores presentes no *dataset*.

treinamento da *Inception-V3* e mantidos constantes para os demais modelos, viabilizando comparações diretas entre eles.

Em termos computacionais, a arquitetura *Inception-V3* tem 27 milhões de parâmetros, 5.71 bilhões de operações de ponto flutuante (*FLOPs*), e um tamanho total de 103 MB. Já o modelo *EfficientNetV2-S* tem 21 milhões de parâmetros, 8.37 bilhões de *FLOPs* e tamanho de 82.7 MB, sendo menor a

menor arquitetura da família *EfficientNetV2*. Por sua vez, a rede *ResNet-V2* conta com 25 milhões de parâmetros, 4.79 bilhões de *FLOPs* e tamanho de 97.8 MB.

C. Treino e Validação

O método de validação cruzada *K-Fold* estratificado foi utilizado para treinar e validar as diferentes arquiteturas testadas.

Nesse método os dados são divididos em K pastas, e cada pasta é usada uma vez para validação, enquanto as demais são utilizadas para treinamento. Esse processo é repetido K vezes, garantindo que cada pasta tenha sido utilizada uma vez na etapa de validação. Essa abordagem garante que cada divisão tenha uma representação proporcional das diferentes classes, o que ajuda a evitar *overfitting* e viés. Como foram testados 3 modelos, a mesma divisão do K -Fold foi utilizada em todos os casos, permitindo comparações diretas.

A Figura 4 ilustra um diagrama do procedimento feito no K -Fold estratificado. O uso dos dados disponíveis para treinamento e validação são maximizados através dessa estratégia, fornecendo uma estimativa mais confiável do desempenho do modelo. Diferentemente da abordagem *hold-out* padrão, no K -Fold estratificado são obtidos K estimativas de cada métrica de desempenho, uma para cada subconjunto de dados. A métrica de desempenho final da rede é dada pela média dessas K estimativas.

D. Setup Experimental

O *setup* experimental foi montado de modo a permitir comparações com outros modelos presentes na literatura e estudos futuros. O *framework* utilizado para realizar os experimentos foi o *Keras* com *backend Tensorflow* [15] utilizando Python como linguagem de programação. O processo de treinamento foi realizado no *Google Colaboratory*, proporcionando um ambiente online conveniente com uma placa de vídeo (*GPU*) gratuita.

Os modelos foram treinados por 10 épocas com lotes de 32 imagens. Para cada arquitetura, as imagens foram redimensionadas para o tamanho padrão de entrada das redes, sendo 299x299 para a *Inception*, 384x384 para a *EfficientNet*, e 224x224 para a *ResNet*. Para otimização, o algoritmo de *Nadam* [16] foi utilizado com taxa de aprendizado de 10^{-4} . A função de perda empregada foi a *categorical crossentropy*. Para o K -Fold estratificado foram feitas 5 divisões e os dados embaralhados. Técnicas de aumento de dados como rotação de até 30°, deslocamento horizontal e vertical, cisalhamento e ampliação, inversão horizontal e vertical foram aplicadas aleatoriamente no conjunto de dados disponíveis.

IV. RESULTADOS

A Figura 5 apresenta os gráficos clássicos de treinamento de redes neurais, permitindo avaliar o comportamento das arquiteturas durante os processos de treino e validação. Para analisar os resultados e permitir comparações, as métricas de precisão, *recall* e *F1-score* dos modelos foram calculadas para cada classe do *dataset* e estão disponíveis nas Tabela I, Tabela II e Tabela III. Essas métricas são as mais utilizadas na literatura para comparação de desempenho entre diferentes arquiteturas de redes neurais como classificadores. A Tabela IV apresenta a média das métricas para cada modelo, compilando os dados e facilitando a comparação.

V. DISCUSSÃO

Conforme a Tabela I, o modelo treinado com a arquitetura *Inception-V3* apresentou precisão mínima foi de 72% para a

Tabela I: Métricas de Precisão, *Recall*, *F1-Score* para cada classe presente no *dataset* para o modelo *Inception-V3*.

Resistência Ω	Precisão	Recall	F1-Score
100	0.91	0.84	0.87
10	0.97	0.99	0.98
11M	0.88	0.97	0.92
150	0.96	0.84	0.90
15	0.90	0.81	0.85
180K	0.97	0.97	0.97
1K	0.96	0.97	0.97
1M	0.72	0.85	0.78
20K	0.96	0.77	0.85
220K	0.85	0.83	0.84
220	0.98	0.92	0.95
22	0.92	0.73	0.81
24K	0.93	0.96	0.94
270K	0.96	0.86	0.91
27	0.96	1.00	0.98
2K2	0.85	0.90	0.88
2	1.00	0.96	0.98
330	0.85	0.76	0.80
33K	1.00	0.96	0.98
3.9K	0.88	0.91	0.90
470	0.89	0.96	0.92
4K7	0.88	0.91	0.90
56K	0.96	0.96	0.96
5K1	0.86	0.83	0.84
5K61	0.94	0.93	0.94
620	0.77	0.95	0.85
68K	0.99	0.98	0.98
6.8	0.90	0.85	0.87
7K5	0.96	0.99	0.98
820	0.90	0.96	0.93
8K2	0.85	0.90	0.87

Tabela II: Métricas de Precisão, *Recall*, *F1-Score* para cada classe presente no *dataset* para o modelo *EfficientNetV2-S*.

Resistência	Precisão	Recall	F1-Score
100	0.92	0.60	0.73
10	0.84	0.90	0.87
11M	0.91	0.89	0.90
150	0.92	0.83	0.87
15	0.91	0.71	0.80
180K	0.96	0.98	0.97
1K	0.81	0.97	0.88
1M	0.65	0.59	0.62
20K	0.81	0.74	0.77
220K	0.97	0.68	0.80
220	0.66	0.95	0.78
22	0.83	0.73	0.77
24K	0.94	0.98	0.96
270K	0.85	0.91	0.88
27	0.82	0.93	0.87
2K2	0.90	0.84	0.87
2	0.89	0.89	0.89
330	0.89	0.63	0.74
33K	0.97	0.93	0.95
3.9K	0.82	0.85	0.83
470	0.97	0.92	0.95
4K7	0.93	0.86	0.89
56K	1.00	0.96	0.98
5K1	0.84	0.88	0.86
5K61	0.96	0.90	0.93
620	0.75	0.88	0.81
68K	0.99	0.97	0.98
6.8	0.90	0.85	0.87
7K5	0.90	0.75	0.82
820	0.57	0.97	0.71
8K2	0.98	0.77	0.86

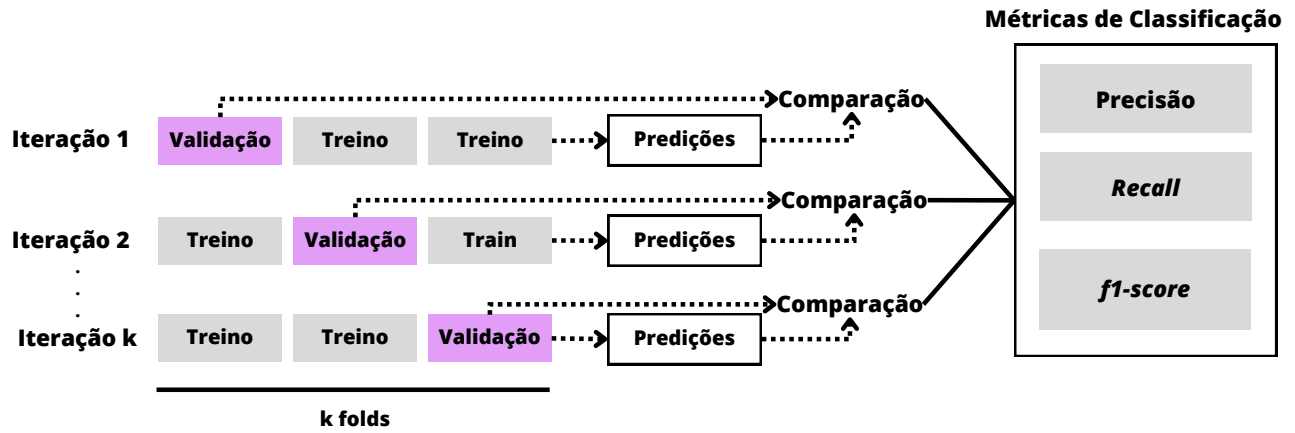


Figura 4: Paradigma do K -Fold estratificado. Os rótulos verdadeiros das pastas de validação são comparadas com as saídas preditas pelo modelo treinado em cada iteração, permitindo computar as métricas de classificação.

Tabela III: Métricas de Precisão, *Recall*, *F1-Score* para cada classe presente no *dataset* para o modelo *ResNet50*.

Resistência Ω	Precisão	<i>Recall</i>	<i>F1-Score</i>
100	0.60	0.86	0.71
10	0.78	0.95	0.86
11M	0.88	0.79	0.83
150	0.72	0.74	0.73
15	0.63	0.77	0.69
180K	0.98	0.99	0.98
1K	0.88	0.78	0.83
1M	0.47	0.61	0.53
20K	0.62	0.40	0.49
220K	0.70	0.68	0.69
220	0.89	0.70	0.78
22	0.48	0.39	0.43
24K	0.88	0.74	0.80
270K	0.82	0.86	0.84
27	0.88	0.78	0.83
2K2	0.77	0.65	0.72
2	0.96	0.93	0.95
330	0.85	0.76	0.80
33K	0.91	0.83	0.87
3.9K	0.81	0.65	0.72
470	0.75	0.80	0.77
4K7	0.66	0.66	0.66
56K	0.98	0.84	0.91
5K1	0.62	0.62	0.62
5K61	0.78	0.83	0.80
620	0.57	0.69	0.63
68K	0.97	0.93	0.95
6.8	0.66	0.60	0.63
7K5	0.75	0.74	0.74
820	0.92	0.73	0.82
8K2	0.66	0.53	0.59

Tabela IV: Média das métricas para os modelos testados.

Modelo	Precisão	<i>Recall</i>	<i>F1-Score</i>
<i>EfficientNetV2-S</i>	0.87	0.85	0.85
<i>Inception-V3</i>	0.91	0.90	0.91
<i>ResNet50</i>	0.77	0.74	0.75

classe de resistor de $1M\Omega$, e valor máximo de 100%, para as classes 2Ω e $33k\Omega$. Já o *F1-Score* mostra um equilíbrio entre as métricas de precisão e *recall* para todas as classes, tendo

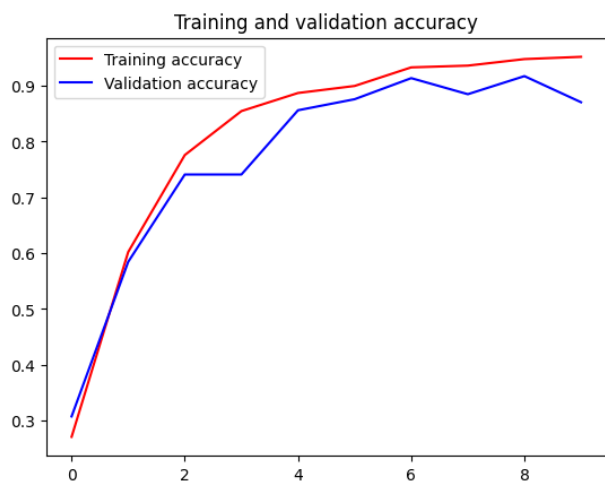
valor mínimo de 78%. O valor de *recall* alto na maioria das classes demonstra que o modelo é eficiente em identificar os verdadeiros positivos. Ressalta-se que as classes com a menor precisão não foram necessariamente as com menos amostras, demonstrando que o modelo não focou em aprender mais as classes com mais exemplos.

A Tabela II mostra que o modelo baseado na *EfficientNetV2-S* teve mais oscilação nas precisões ao longo das classes, em alguns casos atingindo apenas 57% de precisão. Entretanto, os dados mostram que na maioria dos casos bons resultados foram alcançados. O *F1-score* manteve-se acima de 62% para todos os casos e o *recall* teve uma baixa de 59%. O caso em que as três métricas foram menores acontece para o resistor de $1M\Omega$.

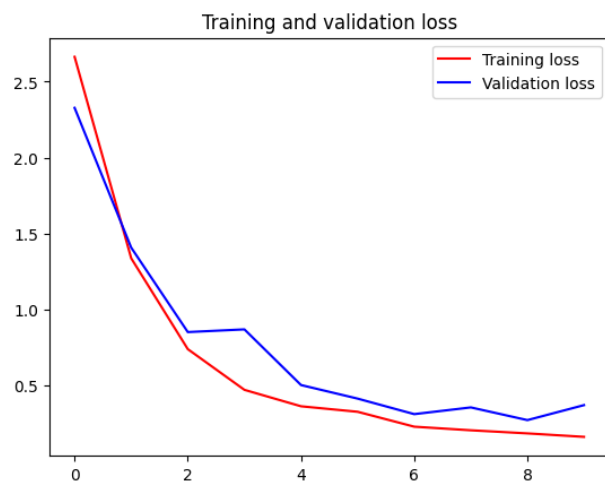
Já a Tabela III apresenta os resultados para a arquitetura baseada na *ResNet50*. Em geral, esse modelo apresentou os piores resultados e teve as maiores oscilações de métricas entre as classes, sendo o maior valor de precisão 98% e o menor 47%. O *recall* e o *F1-Score* apresentaram comportamentos similares aos da precisão. Esse modelo, assim como o *EfficientNetV2-S*, teve os piores resultados para a classe do resistor de $1M\Omega$.

A Tabela IV mostra que os dois primeiros modelos tiveram eficiência próxima, sendo o modelo com a *Inception-V3* ligeiramente melhor com 91% de precisão, enquanto da *EfficientNetV2-S* ficou com 87%. Já o modelo baseado na *ResNet50* foi o pior em todas as métricas, e não conseguiu se adaptar ao problema nas 10 épocas de treinamento. Foram definidas apenas 10 épocas de treinamento para limitar o tempo de execução, o qual aumentaria em 25 vezes para cada modelo em uma situação com 50 épocas e K -Fold estratificado com $k = 5$.

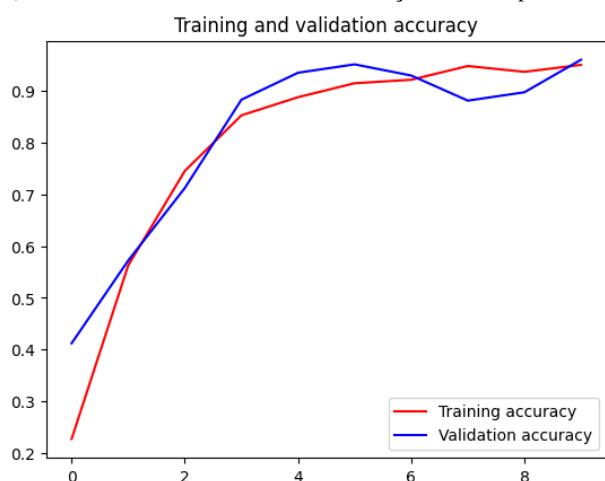
Ao analisar a Figura 5 fica evidente que os modelos baseados na *Inception-V3* e na *EfficientNetV2-S* apresentaram curvas de validação bem próximas das de treino, indicando bom aprendizado do modelo. Já no caso da *ResNet50*, as curvas de validação ficaram mais distantes das de treino, corroborando



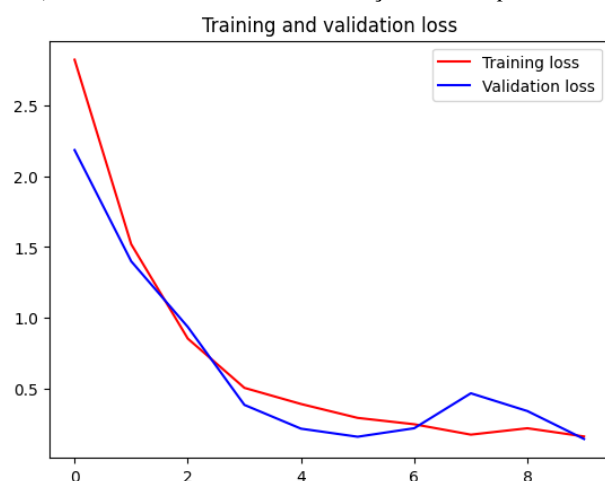
a) Acurácia de treinamento e validação da *Inception-V3*



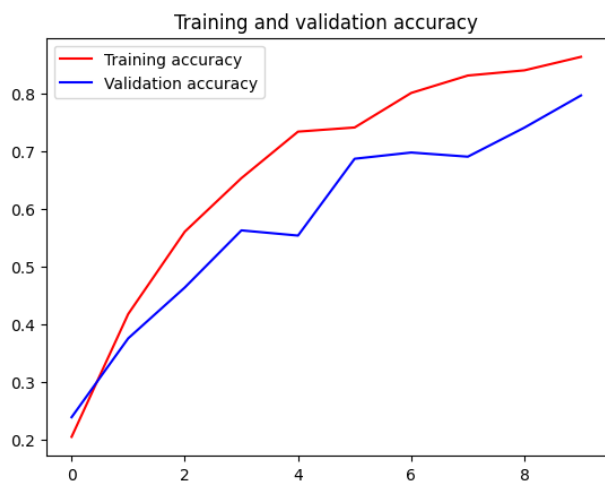
b) Erro de treinamento e validação da *Inception-V3*



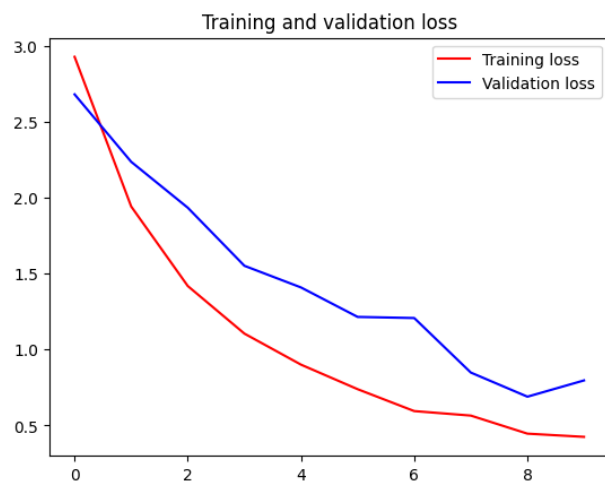
c) Acurácia de treinamento e validação da *EfficientNetV2-S*



Erro de treinamento e validação da *EfficientNetV2-S*



e) Acurácia de treinamento e validação da *ResNet50*



Erro de treinamento e validação da *ResNet50*

Figura 5: Gráficos de acurácia e erro durante o processo de treinamento e validação das arquiteturas propostas. São apresentados como exemplo apenas os gráficos de uma iteração K aleatória do *K-Fold* estratificado de cada modelo.

para os piores resultados obtidos nas métricas de desempenho. Também fica evidente que todos os modelos não atingiram valor mínimo de erro durante o treinamento, indicando que esses modelos ainda poderiam ser treinados por mais épocas para apresentar resultados um pouco melhores.

Ressalta-se que a tarefa de classificação das 31 classes de resistores não era trivial em função das variações das imagens do *dataset*. No conjunto de dados utilizado existiam fotos de uma mesma classe com fundos diferentes, luminosidade não uniforme, baixa resolução, e componentes com pigmentações e tamanhos variados. Algumas imagens apresentavam mais de um resistor simultaneamente, o que nos leva a afirmar que as precisões alcançadas foram boas para a tarefa proposta.

VI. CONCLUSÃO

O trabalho apresenta uma abordagem para classificar resistores com base no código de cores utilizando redes neurais convolucionais e aprendizado profundo. O conjunto de dados utilizados no treinamento e validação foi baseado em imagens reais de resistores em diferentes orientações e condições de iluminação. O *dataset* foi alterado para tornar a tarefa mais real, excluindo a classificação dos resistores por potência e mantendo apenas as classes separadas por valores comerciais de resistência. Para tal tarefa de classificação, foram testados os modelos clássicos de aprendizado profundo *Inception-V3*, *EfficientNetV2-S* e *ResNet50*.

Os resultados de treinamento, validação e teste mostram melhor desempenho da arquitetura *Inception-V3*, sendo a mais indicada para a tarefa. Entretanto, esses resultados foram ligeiramente superiores aos do modelo *EfficientNetV2-S*, que é uma arquitetura mais nova, mais rápida e de menor tamanho, sendo menos custosa de treinar e implementar computacionalmente. A *ResNet50* trouxe os piores resultados e não conseguiu adaptar-se rapidamente a tarefa. Em suma, conclui-se que redes neurais de aprendizado profundo conseguem lidar com a tarefa proposta e classificar corretamente os resistores.

Em trabalhos futuros, testes com os modelos treinados por mais épocas serão realizados para avaliar o desempenho máximo de cada rede. Pretende-se também testar e avaliar novas arquiteturas de redes convolucionais.

REFERÊNCIAS

- [1] M. Muminovic and E. Sokic, "Automatic segmentation and classification of resistors in digital images," in *2019 XXVII International Conference on Information, Communication and Automation Technologies (ICAT)*. IEEE, 2019, pp. 1–6.
- [2] Y.-S. Chen and J.-Y. Wang, "Computer vision on color-band resistor and its cost-effective diffuse light source design," *Journal of Electronic Imaging*, vol. 25, no. 6, pp. 061 409–061 409, 2016.
- [3] M. F. Demir, A. Cankirli, B. Karabatak, A. Yavariabdi, E. Mendi, and H. Kusetogullari, "Real-time resistor color code recognition using image processing in mobile devices," in *2018 International Conference on Intelligent Systems (IS)*. IEEE, 2018, pp. 26–30.
- [4] K. L. Chan and H. Wang, "Reading resistor values by color image processing," in *Automatic Inspection and Novel Instrumentation*, vol. 3185. SPIE, 1997, pp. 157–168.
- [5] A. Jadon, A. Varshney, N. G. Varshney, and M. S. Ansari, "Simple and efficient non-contact technique for resistor value estimation," in *2018 International Conference on Communication and Signal Processing (ICCS)*. IEEE, 2018, pp. 0938–0941.
- [6] R. Molina, P. Federigi, V. Gil-Costa, and M. Printista, "Hybrid classification of resistors through image processing," in *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2014, pp. 103–106.
- [7] P. Niklaus and G. Ulli, "Automated resistor classification," *Group Thesis, Swiss Federal Institute of Technology, Computer Engineering and Networks Laboratory*, 2015.
- [8] S. Wibawanto, K. C. Kirana, and H. Ramadhan, "Ant colony optimization for resistor color code detection," *Knowledge Engineering and Data Science*, vol. 6, no. 1, pp. 15–23, 2023.
- [9] X. Li, Z.-h. Zeng, M. Chen, and S.-y. Che, "A new method of resistor's color rings detection based on machine vision," in *2017 Chinese Automation Congress (CAC)*. IEEE, 2017, pp. 241–245.
- [10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
- [11] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*. PMLR, 2021, pp. 10 096–10 106.
- [12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [13] İlker Önal, E. Özcan, and R. Bayir, "Resistornet-dataset," GitHub, 2018. [Online]. Available: <https://github.com/Eralpozcan/ResistorNet-Dataset>
- [14] 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.
- [15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: <https://www.tensorflow.org/>
- [16] T. Dozat, "Incorporating nesterov momentum into adam," *International Conference on Learning Representations (ICLR)*, 2016.