Aprendizagem de Máquina (2020/Período Especial) - Redes Neurais

Diogo C. T. Batista[1]

[1]Universidade Federal do Paraná (UFPR)
Curitiba – Paraná – Brasil

diogo@diogocezar.com

1. Introdução

Neste laboratório, foi considerada uma base de dados com os meses do ano. As imagens são representações manuscritas das palavras que compreendem os meses: Janeiro, Fevereiro, Março, Abril, Maio, Junho, Julho, Agosto, Setembro, Outubro, Novembro e Dezembro. Portanto, são 12 classes. A Figura 1 demostra alguns exemplos dessas imagens.

Figura 1. Exemplos dos meses manuscritos

As implementações para este laboratório foram relaizadas na plataforma *Google Colab*. Para a obtenção dos resultados explorados neste trabalho, foram realizadas as seguintes etapas: *Preparações*, *Definição dos Modelos*, *Execução do Modelo LeNet5 sem Data Augmentation*, *Execução de um Modelo Personalizado sem Data Augmentation*, *Implementação de Data Augmentation*, *Execução do Modelo LeNet5 com Data Augmentation*, *Execução de um Modelo Personalizado com Data Augmentation*, *Extração de Características* e *Implementação do SVM com as Características Extraídas*.

Cada uma das etapas será detalhada na sequência. Para cada uma das etapas, apresenta-se os resultados obtidos. Ao final do trabalho discute-se os resultados encontrados.

Para a execução dos experimentos, utilizou-se os *frameworks* **Keras** e **Sklearn**.

2. Preparações

Na estapa de preparação, foram realizadas as implementações que possibilitaram:

1. habilitação da GPU;
2. importação dos dados hospedados no GitHub;
3. definição dos arquivos de entrada;
4. definição das funções auxiliares;

Ao executar os experimentos na plataforma Colab do Google, foi possível trabalhar com o processamento via GPU;

Na definição das funções auxiliares, o objetivo foi definir os possíveis componentes reutilizáveis nos experimentos, encapsulando implementações em funções reutilizáveis, além de reduzir a complexidade das implementações, deixando os códigos mais legíveis.

3. Definição dos Modelos

Na etapa de definição dos modelos, criou-se a partir do *frameworks* Keras funções que retornam os modelos. Foram definidos dois modelos. O primeiro modelo definido foi o *LeNet5*. Suas camadas estão representadas na Figura 2.

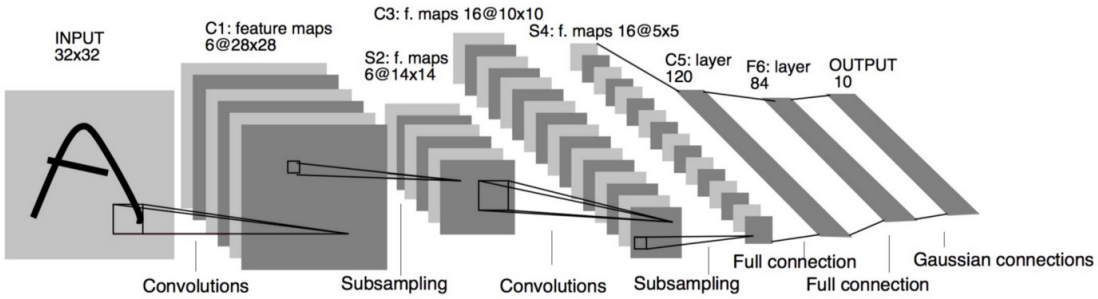

Figura 2. Camadas utilizadas pelo modelo LeNet5

Este modelo foi usado em grande escala para classificar automaticamente dígitos escritos à mão em cheques bancários nos Estados Unidos. Possui apenas 7 camadas, entre as quais existem 3 camadas convolucionais (C1, C3 e C5), 2 camadas de subamostragem (agrupamento) (S2 e S4) e 1 camada totalmente conectada (F6), que são seguidas pela saída camada. Camadas convolucionais usam 5 por 5 convoluções com passo 1. As camadas de subamostragem são 2 por 2 camadas de pooling médias. As ativações sigmóides de Tanh são usadas em toda a rede. Existem várias opções arquitetônicas interessantes feitas no LeNet-5 que não são muito comuns na era moderna de aprendizado profundo.

O segundo modelo, é o mesmo utilizado nas demonstrações em aula e implementa as seguintes as camadas detalhadas no Código 1.

```
1  model = Sequential()
2  model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(img_rows,
       img_cols,3)))
3  model.add(Conv2D(64, (3, 3), activation='relu'))
4  model.add(MaxPooling2D(pool_size=(2, 2)))
5  model.add(Dropout(0.25))
6  model.add(Flatten())
7  model.add(Dense(128, activation='relu'))
8  model.add(Dropout(0.5))
9  model.add(Dense(num_classes, activation='softmax'))
```

Código 1. Modelo Personalizado

Neste modelo, também simplificado, temos 2 convoluções iniciais do tipo *Relu*. Seguida das operações de *MaxPooling2D*, *Dropout*, *Flatten*, *Dense*, *Dropout* e *Dense*.

4. Definindo as Execuções

Quanto ao tamanho da imagem de entrada, para o modelo *LeNet5* utilizou-se o tamanho de 32 por 32 *pixels*, assim como o modelo prevê. Para o modelo *Personalizado* utilizou-se imagens do tamanho 64 por 64 *pixels*.

Com relação ao número de épocas, realizou-se os experimentos com as variações: 32, 64 e 128.

Para todos os experimentos, o tempo de execução em segundos foi registrado.

Para ambos os modelos, realizou-se os seguintes passos:

1. carregamento inicial dos dados (x_train, y_train, x_test, y_test); neste ponto, uma função genérica faz os ajustes necessários nas imagens e retorna os vetores;
2. normalização das imagens;
3. geração dos labels para a matriz de confusão;
4. conversão dos vetores para matrizes binárias;
5. definição e sumarização do modelo;
6. compilação do modelo;
7. treinamento da Rede;
8. obtenção dos resultados.

Os experimentos foram definidos como:

- lenet5_noaug_32
- lenet5_noaug_64
- lenet5_noaug_128
- default_noaug_32
- default_noaug_64
- default_noaug_128
- lenet5_aug_32
- lenet5_aug_64
- lenet5_aug_128
- default_aug_32
- default_aug_64
- default_aug_128

Nos nomes dos experimentos, lenet5 ou default indicam o modelo executado. aug ou noaug indicam no experimento foi utilizada a técnica de data augmentation, ou não. Por fim, os números no final do experimento, indicam a quantidade de épocas testada.

4.1. lenet5_noaug_32

Após a execução deste experimento, obteve-se o gráfico de evolução da acurácia (Figura 3), o gráfico de perda durante o treinamento (Figura 4) e a representação da matriz de confusão (Figura 5). A Tabela 1 sumariza os resultados obtidos.

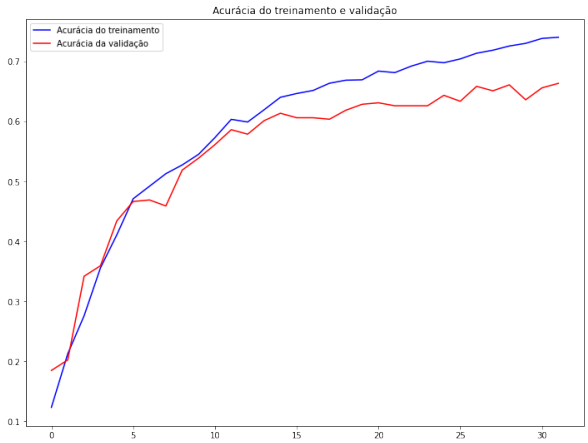

Figura 3. Acurácia

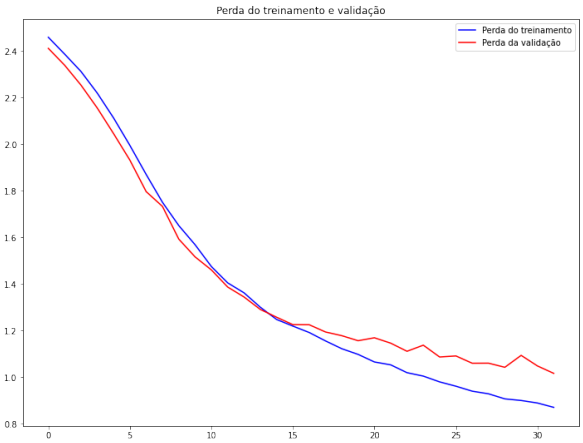

Figura 4. Perda

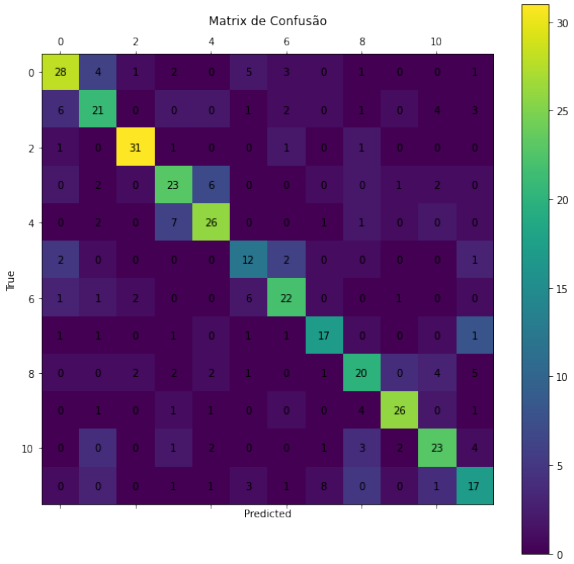

Figura 5. Matrix de Confusão

Experimento	Loss	Acurácia	Tempo de Execução (s)
lenet5_noaug_32	1.00	0.66	5.78

Tabela 1. Resultados Obtidos (lenet5_noaug_32)

4.2. lenet5_noaug_64

Após a execução deste experimento, obteve-se o gráfico de evolução da acurácia (Figura 6), o gráfico de perda durante o treinamento (Figura 7) e a representação da matriz de confusão (Figura 8). A Tabela 2 sumariza os resultados obtidos.

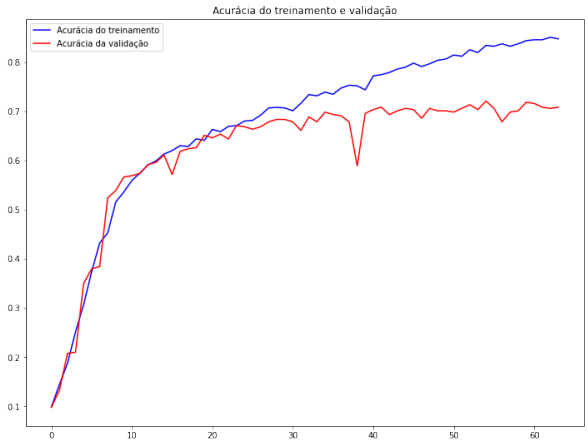

Figura 6. Acurácia

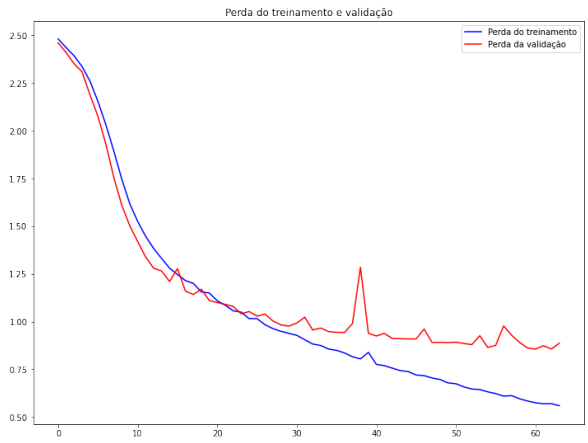

Figura 7. Perda

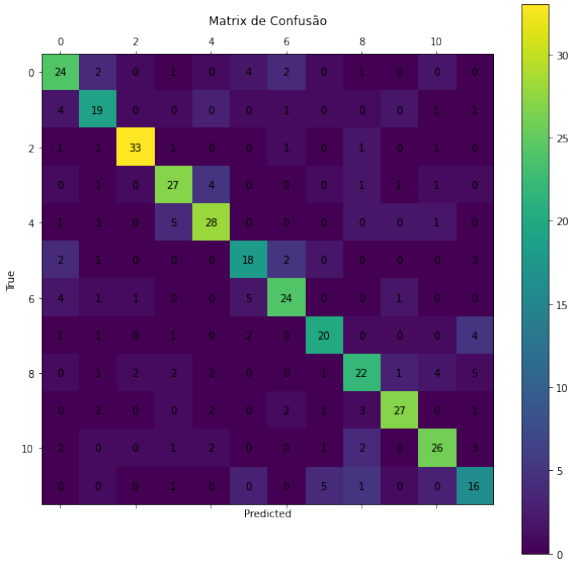

Figura 8. Matrix de Confusão

Experimento	Loss	Acurácia	Tempo de Execução (s)
lenet5_noaug_64	0.88	0.70	9.51

Tabela 2. Resultados Obtidos (lenet5_noaug_64)

4.3. lenet5_noaug_128

Após a execução deste experimento, obteve-se o gráfico de evolução da acurácia (Figura 9), o gráfico de perda durante o treinamento (Figura 10) e a representação da matriz de confusão (Figura 11). A Tabela 3 sumariza os resultados obtidos.

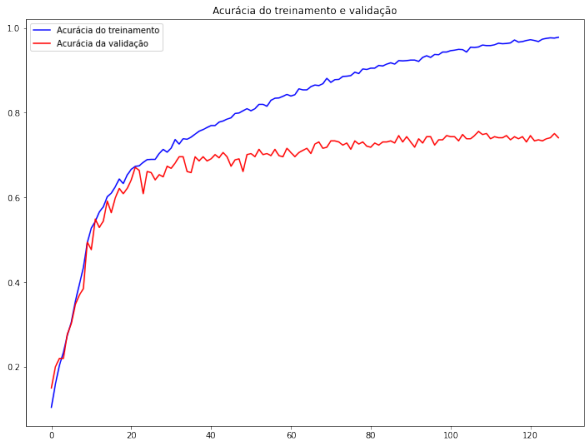

Figura 9. Acurácia

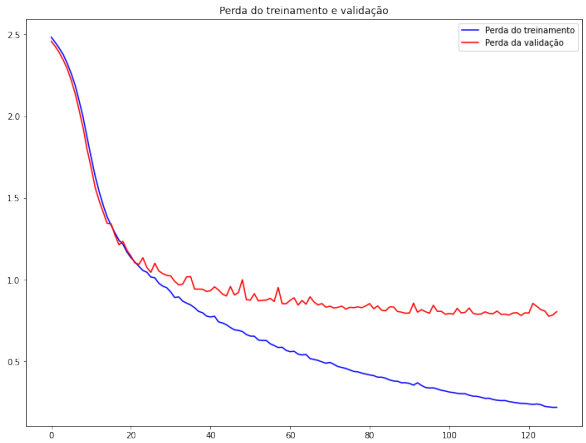

Figura 10. Perda

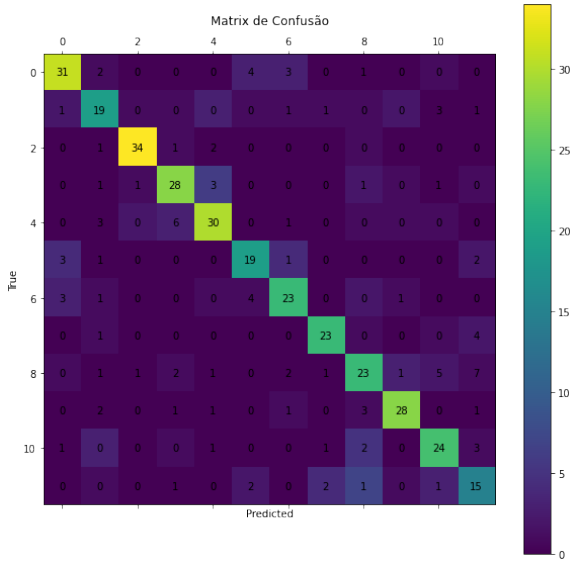

Figura 11. Matrix de Confusão

Experimento	Loss	Acurácia	Tempo de Execução (s)
lenet5_noaug_128	0.80	0.74	16.44

Tabela 3. Resultados Obtidos (lenet5_noaug_128)

4.4. default_noaug_32

Após a execução deste experimento, obteve-se o gráfico de evolução da acurácia (Figura 12), o gráfico de perda durante o treinamento (Figura 13) e a representação da matriz de confusão (Figura 14). A Tabela 4 sumariza os resultados obtidos.

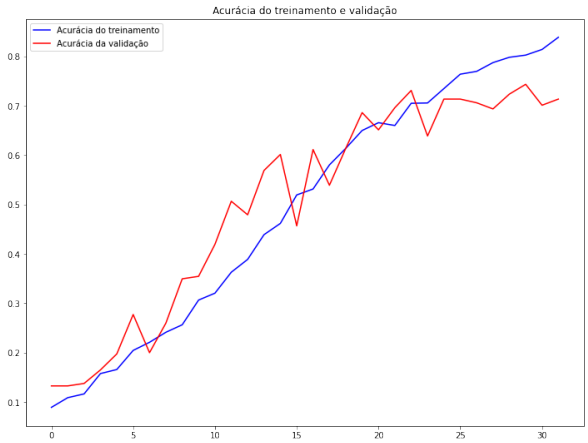

Figura 12. Acurácia

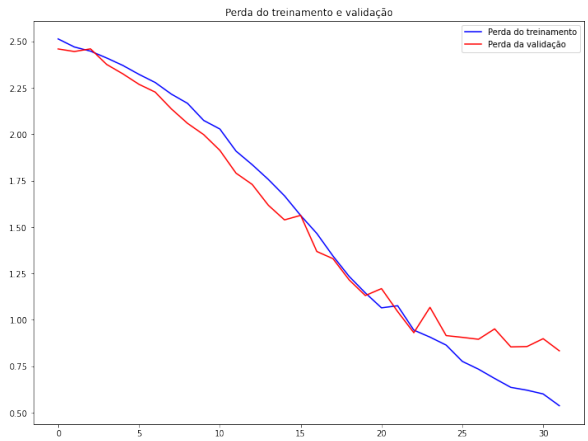

Figura 13. Perda

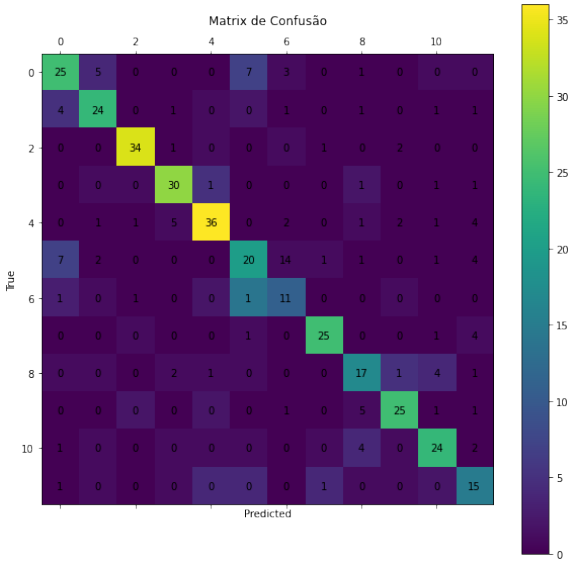

Figura 14. Matrix de Confusão

Experimento	Loss	Acurácia	Tempo de Execução (s)
default_noaug_32	0.83	0.71	12.37

Tabela 4. Resultados Obtidos (default_noaug_32)

4.5. default_noaug_64

Após a execução deste experimento, obteve-se o gráfico de evolução da acurácia (Figura 15), o gráfico de perda durante o treinamento (Figura 16) e a representação da matrix de confusão (Figura 17). A Tabela 5 sumariza os resultados obtidos.

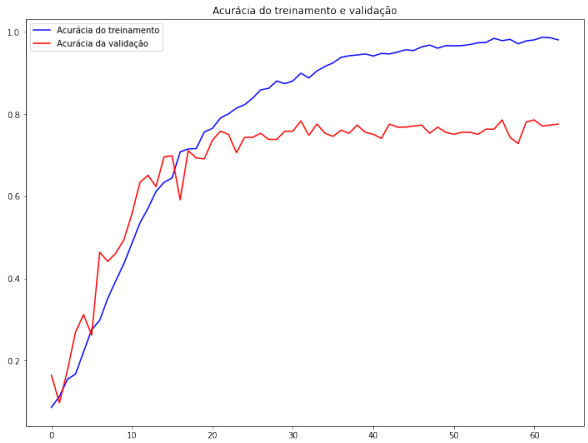

Figura 15. Acurácia

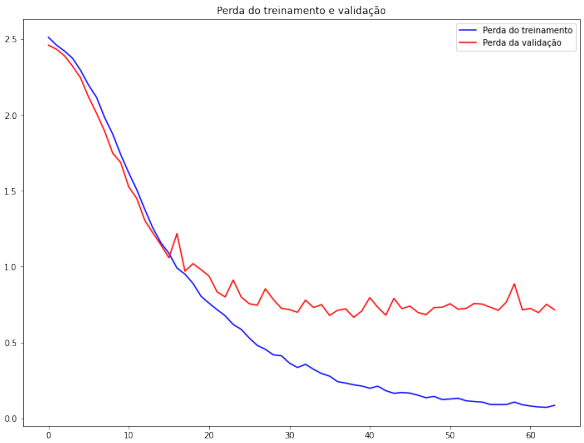

Figura 16. Perda

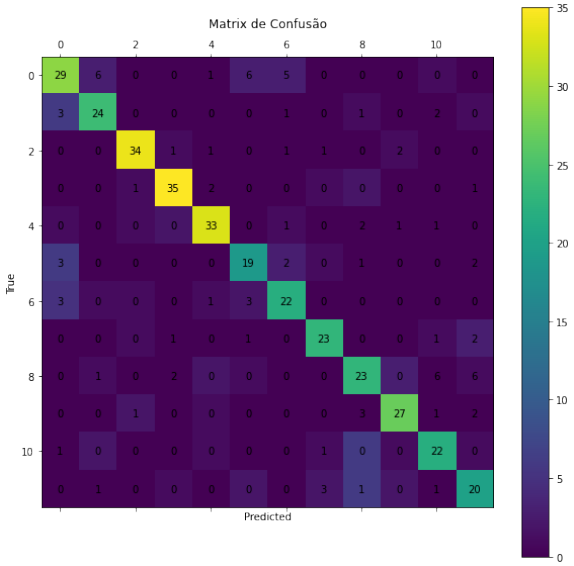

Figura 17. Matrix de Confusão

Experimento	Loss	Acurácia	Tempo de Execução (s)
default_noaug_64	0.71	0.77	21.38

Tabela 5. Resultados Obtidos (default_noaug_64)

4.6. default_noaug_128

Após a execução deste experimento, obteve-se o gráfico de evolução da acurácia (Figura 18), o gráfico de perda durante o treinamento (Figura 19) e a representação da matriz de confusão (Figura 20). A Tabela 6 sumariza os resultados obtidos.

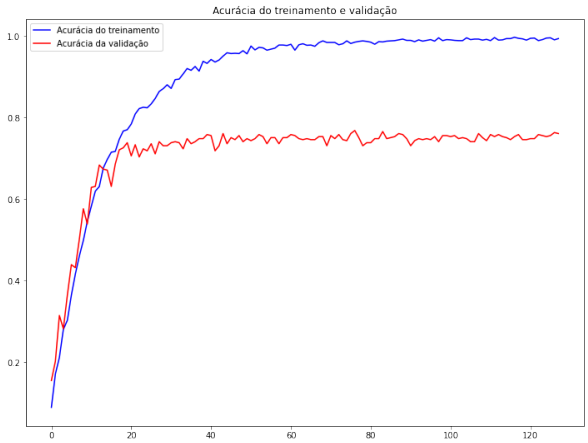

Figura 18. Acurácia

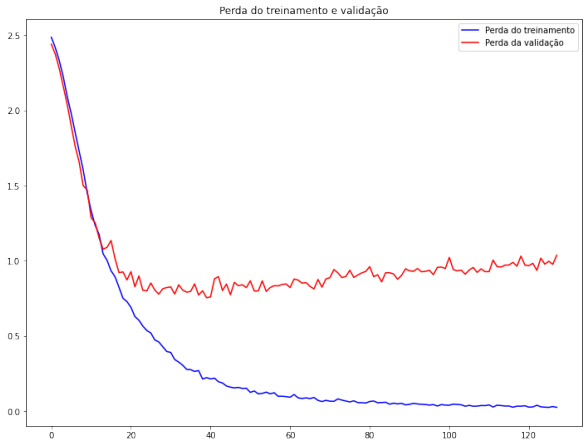

Figura 19. Perda

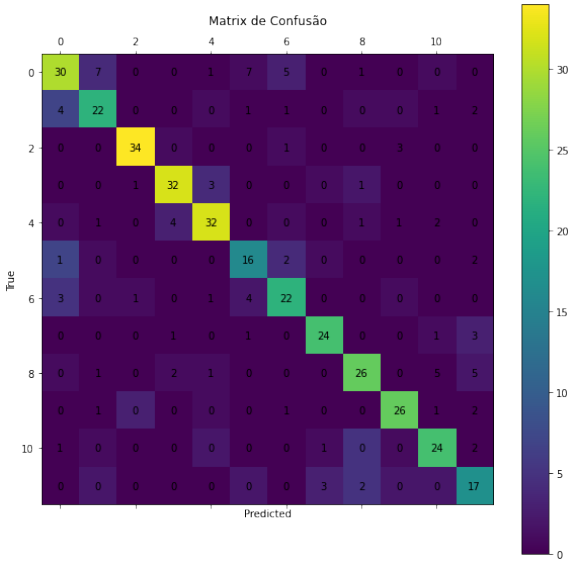

Figura 20. Matrix de Confusão

Experimento	Loss	Acurácia	Tempo de Execução (s)
default_noaug_128	1.03	0.76	40.14

Tabela 6. Resultados Obtidos (default_noaug_128)

5. Implementação de Data Augmentation

Neste ponto do experimento, foram implementadas técnicas para a realização de *data augmentation*. Para isso, foram implementadas funções específicas para a realização de pequenas variações nas imagens da base de treinamento original. O Código 2 mostra quais foram as funções utilizadas para a realização destas transformações.

```
1  flip_rotation_brightness_zoom(path, zoom=[0.5, 1.0], brightness=[0.2, 1.0], rotation
       =90, flip_horizontal=False,
2  flip_vertical=False, subdir="zoom")
3
4  random_zoom(path, zoom=[0.5, 1.0], subdir="zoom")
5
6  random_brightness(path, brightness=[0.2, 1.0], subdir="brightness")
7
8  random_rotation(path, rotation=90, subdir="rotation")
9
10 horizontal_vertical_flip(path, flip_horizontal=False, flip_vertical=False, subdir="
       flip")
11
12 orizontal_vertical_shift(path, size=0.5, bool_width=True, subdir="shift")
```

Código 2. Funções de Transformação das Imagens

A função **flip_rotation_brightness_zoom** realiza transformações de rotação, brilho e zoom na imagem. A função **random_zoom** realiza a transformação de zoom da imagem. A função **random_brightness** realiza a transformação de brilho da imagem. A função **random_rotation** realiza a transformação de rotacão da imagem. A função **horizontal_vertical_flip** realiza a transformação de flip vertical. A função **horizontal_vertical_shift** realiza a transformação de shift. Todas as funções possuem parâmetros.

Para imagem da base de treinamento foram realizadas as operações descritas no Código 3.

```
horizontal_vertical_flip(image_path, flip_horizontal=True, flip_vertical=False)
horizontal_vertical_flip(image_path, flip_horizontal=False, flip_vertical=True)
horizontal_vertical_flip(image_path, flip_horizontal=True, flip_vertical=True)
horizontal_vertical_flip(image_path, flip_horizontal=False, flip_vertical=False)
horizontal_vertical_shift(image_path, bool_width=True)
horizontal_vertical_shift(image_path, bool_width=False)
random_rotation(image_path, rotation=10)
random_rotation(image_path, rotation=20)
random_rotation(image_path, rotation=30)
random_rotation(image_path, rotation=45)
random_brightness(image_path)
random_brightness(image_path, brightness=[0.1, 0.2])
random_brightness(image_path, brightness=[0.3, 0.4])
random_brightness(image_path, brightness=[0.4, 0.5])
random_zoom(image_path)
random_zoom(image_path, zoom=[0.1, 0.5])
random_zoom(image_path, zoom=[0.1, 0.2])
random_zoom(image_path, zoom=[0.1, 0.3])
flip_rotation_brightness_zoom(image_path, rotation=30)
flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.5], brightness=[0.1, 0.5])
flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.5], brightness=[0.1, 0.5])
flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.5], brightness=[0.1, 0.5],
    rotation=30)
flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.5], brightness=[0.1, 0.5],
    rotation=30)
flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.5], brightness=[0.1, 0.5],
    rotation=30)
flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.8], brightness=[0.1, 0.8],
    rotation=45)
flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.8], brightness=[0.1, 0.8],
    rotation=45)
flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.2], brightness=[0.1, 0.2],
    rotation=30)
flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.2], brightness=[0.1, 0.2],
    rotation=45)
flip_rotation_brightness_zoom(image_path, zoom=[0.9, 1], brightness=[0.9, 1],
    rotation=30)
flip_rotation_brightness_zoom(image_path, zoom=[0.9, 1], brightness=[0.9, 1],
    rotation=45)
```

Código 3. Funções de Augmentation Executadas

A base inicial de treinamento, é composta por **1578** imagens. Após execução do *script*, obteve-se o total de **27558** imagens.

5.1. lenet5_aug_32

Após a execução deste experimento, obteve-se o gráfico de evolução da acurácia (Figura 21), o gráfico de perda durante o treinamento (Figura 22) e a representação da matriz de confusão (Figura 23). A Tabela 7 sumariza os resultados obtidos.

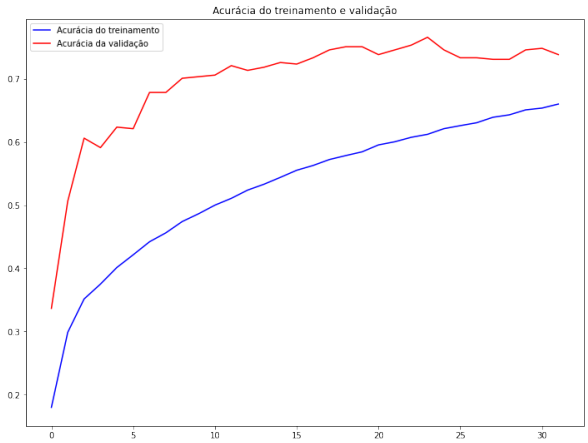

Figura 21. Acurácia

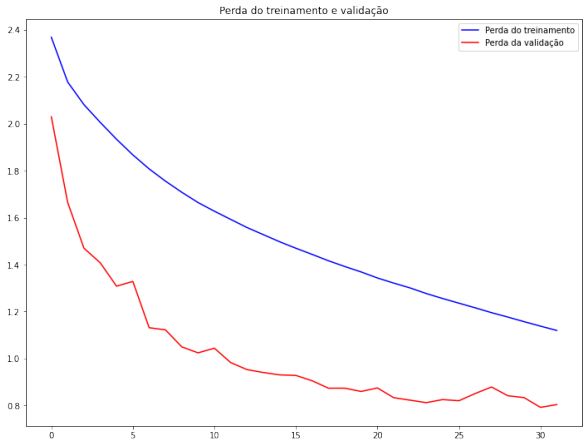

Figura 22. Perda

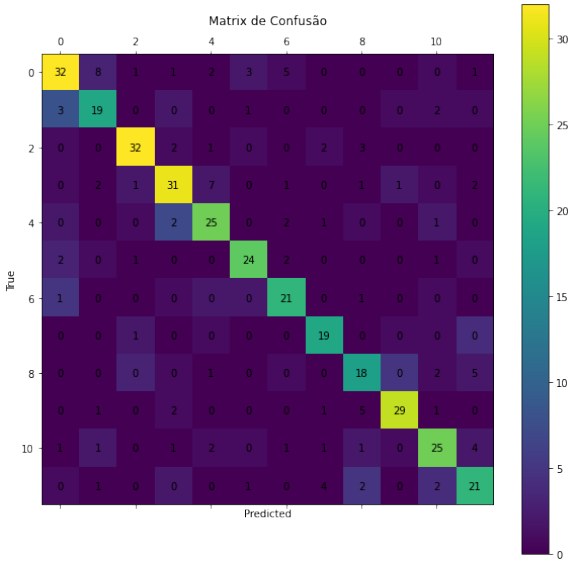

Figura 23. Matrix de Confusão

Experimento	Loss	Acurácia	Tempo de Execução (s)
lenet5_aug_32	0.80	0.73	63.91

Tabela 7. Resultados Obtidos (lenet5_aug_32)

5.2. lenet5_aug_64

Após a execução deste experimento, obteve-se o gráfico de evolução da acurácia (Figura 24), o gráfico de perda durante o treinamento (Figura 25) e a representação da matriz de confusão (Figura 26). A Tabela 8 sumariza os resultados obtidos.

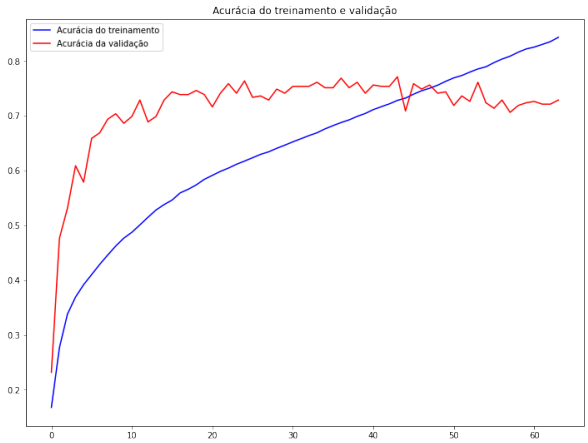

Figura 24. Acurácia

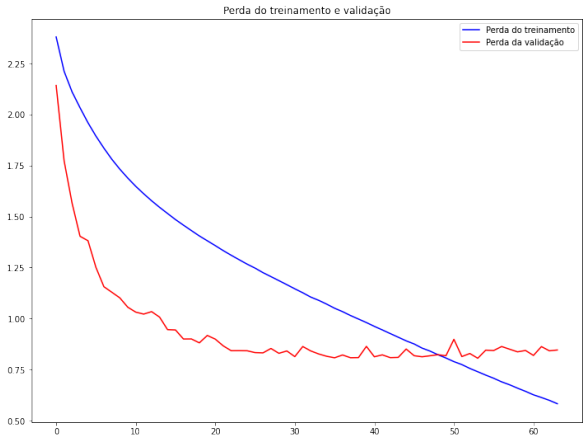

Figura 25. Perda

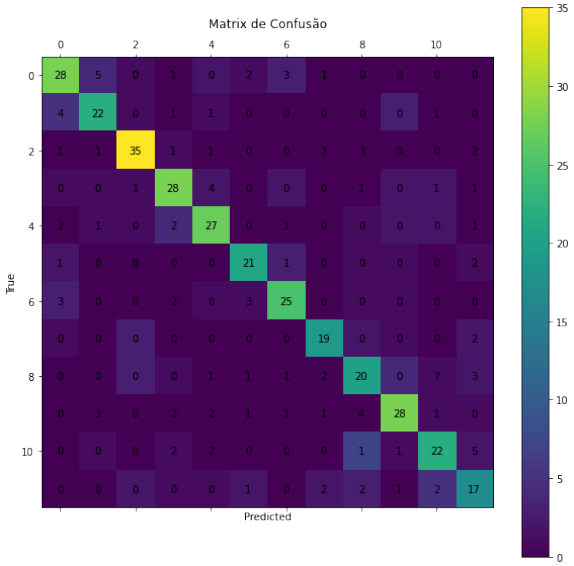

Figura 26. Matrix de Confusão

Experimento	Loss	Acurácia	Tempo de Execução (s)
lenet5_aug_64	0.84	0.72	105.89

Tabela 8. Resultados Obtidos (lenet5_aug_64)

5.3. lenet5_aug_128

Após a execução deste experimento, obteve-se o gráfico de evolução da acurácia (Figura 36), o gráfico de perda durante o treinamento (Figura 37) e a representação da matriz de confusão (Figura 38). A Tabela 9 sumariza os resultados obtidos.

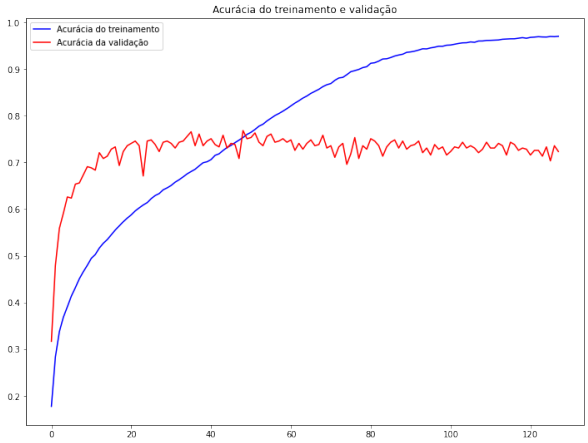

Figura 27. Acurácia

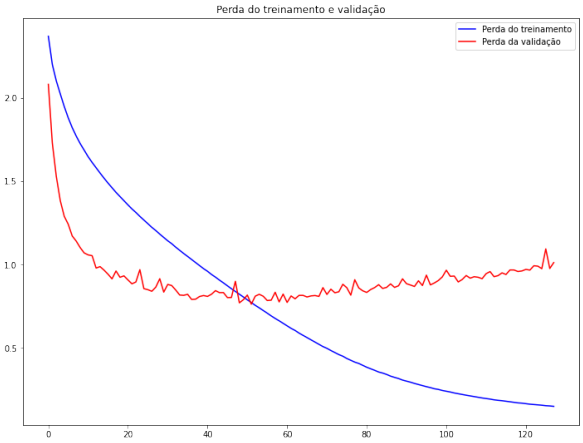

Figura 28. Perda

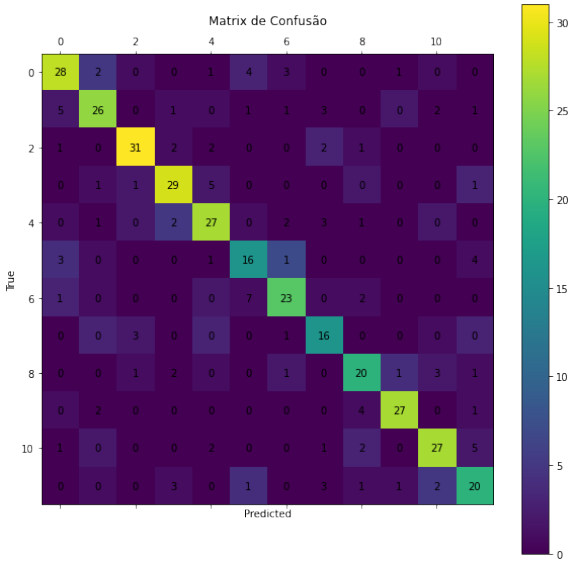

Figura 29. Matrix de Confusão

Experimento	Loss	Acurácia	Tempo de Execução (s)
lenet5_aug_128	1.01	0.72	193.93

Tabela 9. Resultados Obtidos (lenet5_aug_128)

5.4. custom_aug_32

Após a execução deste experimento, obteve-se o gráfico de evolução da acurácia (Figura 30), o gráfico de perda durante o treinamento (Figura 31) e a representação da matriz de confusão (Figura 32). A Tabela 10 sumariza os resultados obtidos.

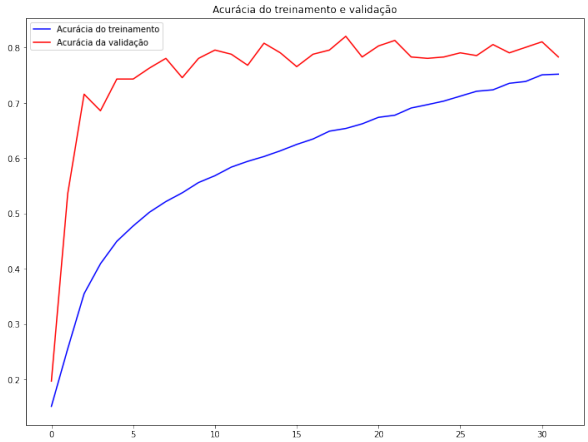

Figura 30. Acurácia

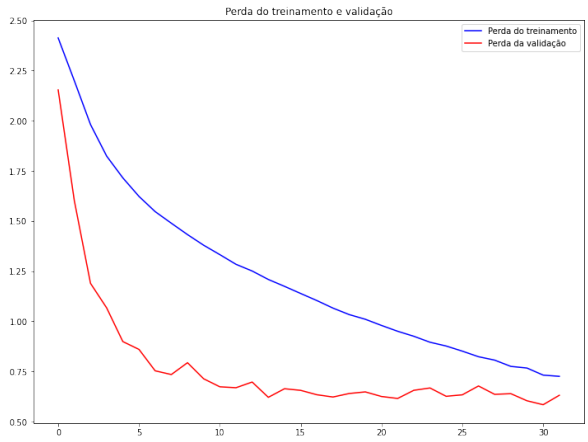

Figura 31. Perda

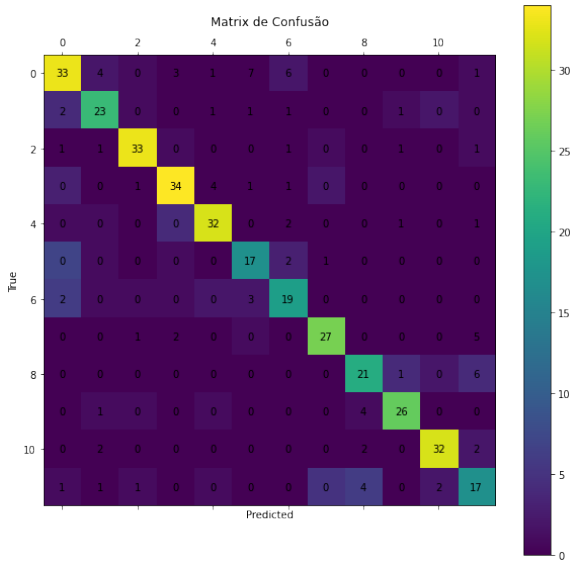

Figura 32. Matrix de Confusão

Experimento	Loss	Acurácia	Tempo de Execução (s)
default_aug_32	0.63	0.78	159.49

Tabela 10. Resultados Obtidos (custom_aug_32)

5.5. custom_aug_64

Após a execução deste experimento, obteve-se o gráfico de evolução da acurácia (Figura 33), o gráfico de perda durante o treinamento (Figura 34) e a representação da matriz de confusão (Figura 35). A Tabela 11 sumariza os resultados obtidos.

Experimento	Loss	Acurácia	Tempo de Execução (s)
default_aug_64	0.96	0.76	297.53

Tabela 11. Resultados Obtidos (custom_aug_64)

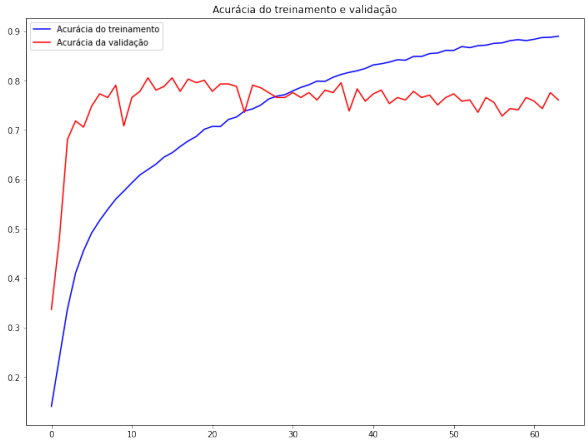

Figura 33. Acurácia

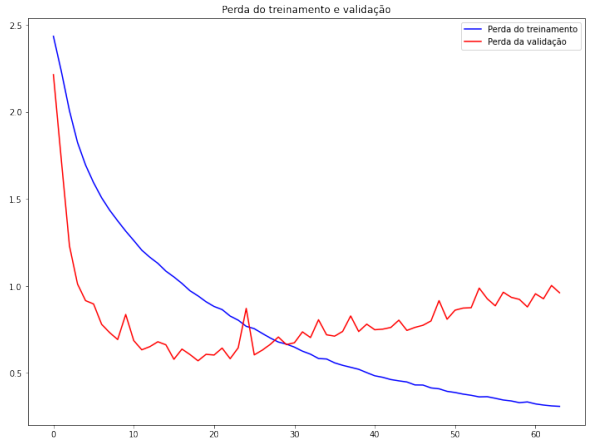

Figura 34. Perda

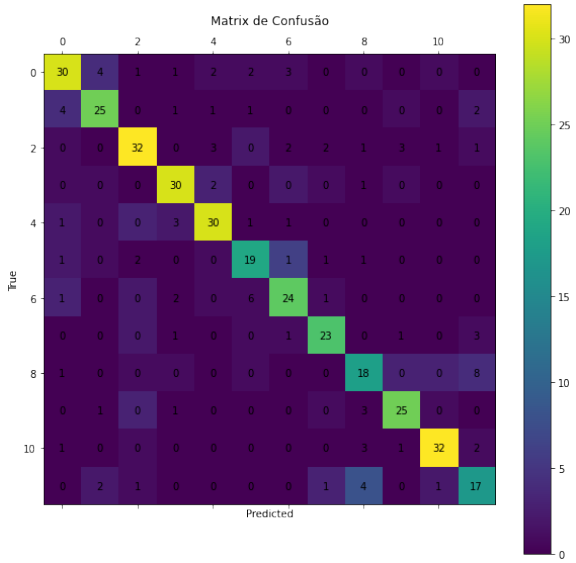

Figura 35. Matrix de Confusão

5.6. custom_aug_128

Após a execução deste experimento, obteve-se o gráfico de evolução da acurácia (Figura 36), o gráfico de perda durante o treinamento (Figura 37) e a representação da matriz de confusão (Figura 38). A Tabela 9 sumariza os resultados obtidos.

Experimento	Loss	Acurácia	Tempo de Execução (s)
default_aug_128	1.26	0.76	557.75

Tabela 12. Resultados Obtidos

6. Extração de Características

Na sequência dos experimentos, utilizou-se um modelo pré-treinado da **ImageNet** para a extração das características das imagens, como demonstrado no Código 4.

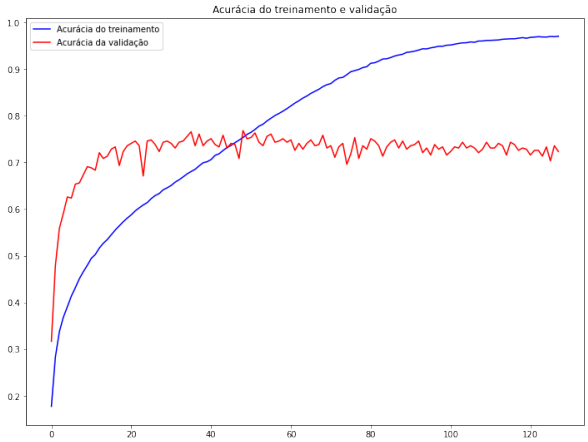

Figura 36. Acurácia

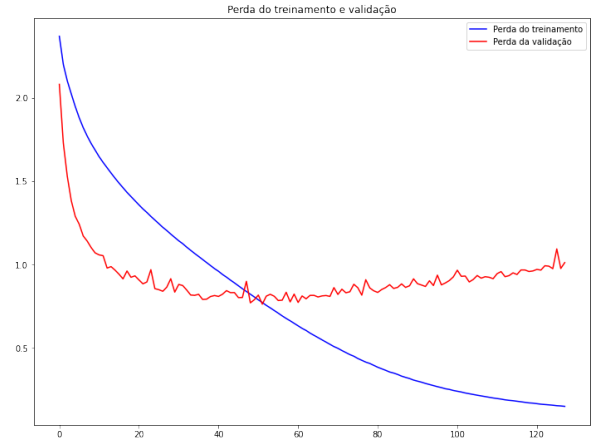

Figura 37. Perda

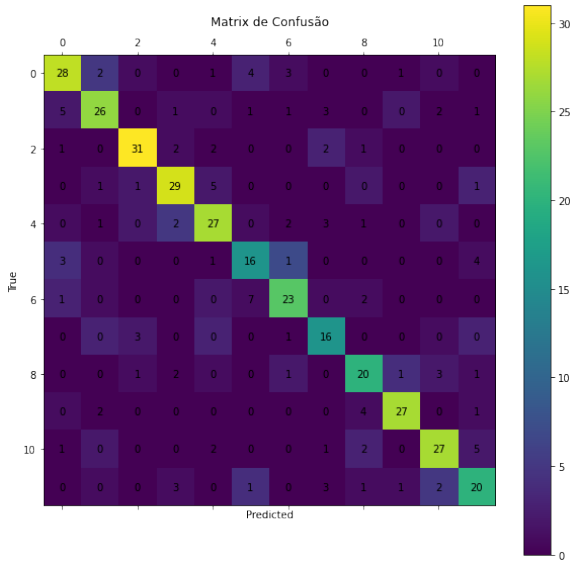

Figura 38. Matrix de Confusão (custom_aug_64)

```
1  model = InceptionV3(weights='imagenet', include_top=False)
```

Código 4. ImageNet - InceptionV3

Como base de dados, utilizou-se inicialmente a base original sem data augmentation e na sequência, utilizou-se a base transformada pela técnica.

Foram realizados dois experimentos utilizando o classificador SVM, implementado pela biblioteca **Sklearn**. Criou-se arquivos de treinamento e teste para para a coleção de dados original e para a coleção com data augmentation.

As características foram extraídas das imagens e salvas como no exemplo descrito no Código 5.

```
1  0 1:0.000000 2:0.000000 3:0.001412 4:0.000000 5:0.014124 6:0.000000 ...
```

Código 5. Exemplo do Formato de Entrada

7. Implementação do SVM

Após a execução deste experimento, obteve-se as matrizes de confusão. A Figura 39 mostra a matriz de confusão obtida para o experimento sem data augmentation, enquanto que a Figura 40 mostra a matriz de confusão obtida com os dados obtidos após a aplicação da técnica de data augmentation.

A Tabela 13 mostra os resultados de F1Score, Acurário e o tempo de execução obtidos nos experimentos.

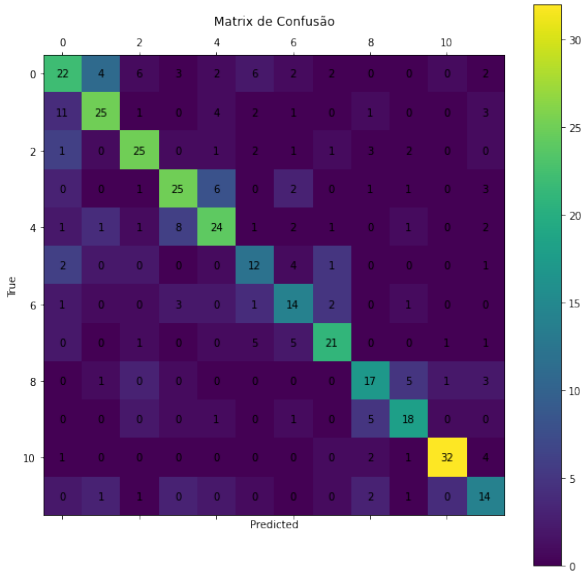

Figura 39. Sem Augmentation

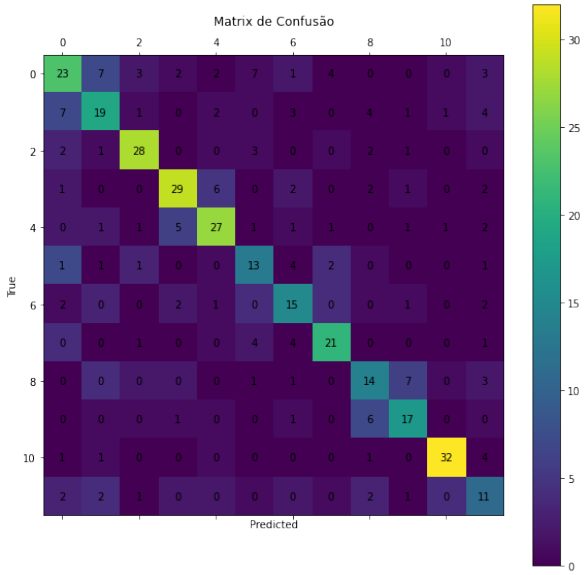

Figura 40. Com Augmentation

Experimento	F1Score	Acurácia	Tempo de Execução (s)
svm_aug	0.616	0.621	1992.622
svm_noaug	0.615	0.621	12.658

Tabela 13. Resultados Obtidos (SVM)

8. Discussão dos Resultados

A Tabela 14 mostra a sumarização de todos os resultados obtidos nos experimentos.

Experimento	Loss	Acurácia	Tempo de Execução (s)
lenet5_noaug_32	1.00	0.66	5.78
lenet5_noaug_64	0.88	0.70	9.51
lenet5_noaug_128	0.80	0.74	16.44
default_noaug_32	0.83	0.71	12.37
default_noaug_64	0.71	0.77	21.38
default_noaug_128	1.03	0.76	40.14
lenet5_aug_32	0.80	0.73	63.91
lenet5_aug_64	0.84	0.72	105.89
lenet5_aug_128	1.01	0.72	193.93
default_aug_32	0.63	0.78	159.49
default_aug_64	0.96	0.76	297.53
default_aug_128	1.26	0.76	557.75

Tabela 14. Sumários dos Obtidos

Nos experimentos realizados o foco não foi a otimização dos resultados de Acurácia, por este motivo, não foram exploradas alternativas ou ajustes de parâmetros.

Na implementação de Data Augmentation, é possível notar em algumas imagens, que o resultado gerado podem não corresponder exatamente a uma informação que ajude com características verdadeiras de uma determinada classo. Isso pode ter influenciado diretamente na melhoria dos resultados.

Por exemplo, é possível observar algumas anomalidas destacadas na Figura 41.

Figura 41. Exemplos sem características verdadeiras

É possível notar que o melhor resultado foi obtido pelo experimento *default_aug_32* com o valor de Acurácia = 0.78 e Loss = 0.63.

O pior resultado foi obtido pelo experimento *lenet5_noaug_32* com o valor de Acurácia = 0.66 e Loss = 1.

Para os experimentos da modelo LeNet5 sem data augmentation, foi possível notar que os gráficos de acurácia e perda, se mantiveram bastante equilibrados, com a exeção da execução com 128 épocas na qual,após a época 20 o acurácia e perda se estabilizam. Quanto as matrizes de confusão, os erros se concentram praticamente nos mesmos lugares, pois os dados fonte são os mesmos. Os resultados melhoraram com os experimentos com mais épocas.

Para os experimentos da modelo LeNet5 com data augmentation, foi possível notar que os gráficos de acurácia e perda, tiveram uma distância considerável para 32 épocas;

Para 64 épocas temos uma estabilidade após a época 50, que se mantém para o experimento de 128 épocas. Quanto as matrizes de confusão, os erros se concentram praticamente nos mesmos lugares, pois os dados fonte são os mesmos. Os resultados pioram com os experimentos com mais épocas.

Já para os experimentos do modelo personalizado sem data augmentation, foi possível notar que os gráficos de acurácia e perda, se mantiveram bastante equilibrados, e com uma estabilidade após época 20. Quanto as matrizes de confusão, os erros se concentram praticamente nos mesmos lugares, pois os dados fonte são os mesmos. Os resultados melhoraram com os experimentos com mais épocas.

E por fim, para os experimentos do modelo personalizado com data augmentation, foi possível notar que os gráficos de acurácia e perda, se mantiveram bastante equilibrados, e com uma estabilidade após época 20. Quanto as matrizes de confusão, os erros se concentram praticamente nos mesmos lugares, pois os dados fonte são os mesmos. Os resultados pioram com os experimentos com mais épocas.

A Tabela 15 mostra os resultados de F1Score, Acurário e o tempo de execução obtidos nos experimentos relacionados ao SVM. É possível notar que não houve melhora nos resultados, provavelmente por conta da qualidade das imagens geradas pela técnica de data augmentation.

Experimento	**F1Score**	**Acurácia**	**Tempo de Execução (s)**
svm_aug	0.616	0.621	1992.622
svm_noaug	0.615	0.621	12.658

Tabela 15. Resultados Obtidos (SVM)

9. Código Fonte

Os códigos fonte podem ser encontrados em:

- GitHub: https://github.com/diogocezar/phd-machine-learning-lab3
- Google Colab: https://bit.ly/3gR739z