

Aprendizagem Profunda

Relatório do Trabalho Prático

MEI - 2023/2024

Grupo 4

Hugo Martins
A95125



João Escudeiro
A96075



José Rocha
A97270



Universidade do Minho

1 Introdução

Na medicina, os modelos de classificação de imagens com Deep Learning têm-se demonstrado bastante eficientes para a análise de imagens médicas através de redes neurais convolucionais.

Este projeto visa a criação de modelos de Deep Learning com a capacidade de analisar imagens de ossos obtidas por Raios-x e determinar se há fraturas nos ossos de uma imagem dada como input.

2 Metodologia e Objetivos

A metodologia adotada para a realização do treino e avaliação das redes neurais convolucionais (CNNs) para o nosso problema envolveu várias etapas sistemáticas, que serão descritas a seguir:

- **Análise do Dataset:** Analisar o conjunto de imagens e perceber a distribuição dos dados de treino e teste.
- **Instanciação de Modelos:** Criar modelos de classificação de imagens baseados em redes neurais convolucionais e *visual transformers*. Dois modelos serão baseados nas arquiteturas **AlexNet** e **ResNet**. Depois será feita uma análise com arquiteturas presentes no MONAI e o último modelo será baseado num *visual transformer*.
- **Avaliação dos Modelos:** A avaliação será realizada utilizando métricas de desempenho como accuracy e a matriz de confusão, que permitirão entender a taxa de acertos e erros, bem como a distribuição destes erros entre as diferentes classes. Adicionalmente, a análise das probabilidades da camada de saída será realizada para avaliar a estabilidade e a confiança das previsões dos modelos.
- **Visualização e Análise dos Resultados:** Durante e após o treino, gráficos de accuracy e loss serão gerados para cada época. Estes gráficos serão utilizados para identificar tendências e possíveis problemas, como overfitting ou underfitting.
- **Depuração e Refinamento dos Modelos:** Com base nos resultados obtidos, ajustes nos hiperparâmetros e na arquitetura dos modelos serão realizados iterativamente para melhorar o desempenho.
- **Análise comparativa dos modelos:** Comparar o desempenho dos modelos utilizados de forma a perceber qual é o melhor modelo para o nosso caso de uso.

3 Deep Learning para classificação de imagens

3.1 Dataset

O conjunto de dados é composto por imagens de Raios-x fraturadas e não fraturadas de várias articulações. Estas imagens estão representadas em tons de cinza, possuindo, portanto, apenas um canal de cor.

As imagens foram redimensionadas para um tamanho de 224 por 224 e foi aplicada a operação.

A tarefa é construir um classificador de imagens para detetar fraturas em determinada imagem de Raio-x. Este conjunto de dados é composto por diferentes articulações nas extremidades superiores.

O Dataset conta com 8863 instâncias de treino e 600 instâncias de teste.

Estas imagens foram recolhidas da plataforma Kaggle através do link:

<https://www.kaggle.com/datasets/vuppalaadithyasairam/bone-fracture-detection-using-xrays>.

3.2 Análise do Dataset

Para a análise do Dataset foi feita uma avaliação sobre a distribuição/balanceamento dos dados:

- **Dados de treino:** Os dados de treino são compostos por 4480 imagens com fraturas (50.5%) e 4383 imagens sem fraturas (49.5%).
- **Dados de teste:** Os dados de teste são compostos por 360 imagens com fraturas (40.0%) e 240 imagens sem fraturas (60.0%).

Pela análise (retratada de forma gráfica nas figuras 1 e 2), podemos concluir que há uma proporção aproximadamente igual de imagens com fraturas e de imagens sem fraturas tanto nos dados de treino como nos dados de teste.

Do conjunto dados de treino, 20% será utilizado para validação da rede.

3.3 Modelos de classificação de imagens

3.3.1 AlexNet (PyTorch)

A arquitetura da AlexNet, implementada no PyTorch através da função “torchvision.models.alexnet()”, é composta por cinco camadas convolucionais seguidas por camadas de max pooling para redução de dimensionalidade.

Foram realizados diversos testes de desempenho da arquitetura alterando os hiperparâmetros:

- **EPOCHS:** número de vezes que todo o conjunto de dados de treino é apresentado ao modelo durante o processo de treino. Cada época consiste numa passagem completa por todos os dados de treino durante a qual o modelo ajusta os seus pesos para minimizar a função de perda.

- **BATCH_SIZE:** número de exemplos de treino utilizados numa única iteração do algoritmo de otimização durante o treino do modelo.
- **Learning rate (lr):** controla o tamanho dos ajustes feitos nos pesos do modelo com base no gradiente da função de perda durante o treino.

Os modelos foram treinados e validados na plataforma Kaggle com recurso ao acelerador GPU P100 e os valores que constam na tabela 1 referentes ao tempo referem-se ao tempo de treino dos modelos (*wall time*).

batch_size	epoch	lr	elapsed time (min)	accuracy (%)
32	15	0.001	7	61.0
32	15	0.002	6	67.5
32	15	0.005	7	67.7
32	15	0.01	6	59.0
32	30	0.0002	12	45.2
32	30	0.0005	12	67.5
32	30	0.001	12	65.8
32	30	0.002	12	62.0
64	15	0.002	6	62.7
64	15	0.005	6	65.2
64	15	0.01	7	42.7
64	30	0.0002	14	60.0
64	30	0.0005	11	54.8
64	30	0.001	12	66.5
64	30	0.002	12	67.7
64	30	0.005	12	60.0
128	15	0.002	5	53.7
128	15	0.005	5	67.2
128	15	0.01	5	53.8
128	30	0.001	12	55.5
128	30	0.002	11	67.5
128	30	0.005	12	62.5
128	30	0.01	12	61.5

Table 1: Resultados de Treino da AlexNet (PyTorch)

Pela análise da tabela, percebemos que esta arquitetura não atingiu uma accuracy considerada satisfatória e, portanto, não adquiriu “conhecimento” sobre a classificação das imagens em questão.

Além disso, ao voltar a treinar e testar o modelo (usamos aquele com mais accuracy representado na terceira linha da tabela), percebemos grandes variações na accuracy do mesmo, atingindo valores abaixo dos 50%. O grupo considerou a hipótese do problema ser causado por valores de probabilidades na última camada da rede perto de 0,5, o que tornaria a rede mais “aleatória”.

Após imprimir os valores das probabilidades, percebeu-se que a rede atinge valores bastante altos (em geral, perto de 100%) mesmo quando erra, o que poderá

indicar uma confiabilidade na classe errada, uma vez que os padrões visuais das classes são semelhantes e quase indistinguíveis até ao olho humano.

Foi feita a análise da matriz de confusão (Figura 3) obtida treinando a rede 5 vezes para perceber variações e percebeu-se existe um número razoável de verdadeiros positivos (imagens com fraturas que a rede identificou corretamente) porém, para as imagens sem fraturas, a rede não revela possuir a mesma capacidade. Todos os restantes campos da matriz de confusão apresentaram valores muito próximos uns dos outros.

De seguida, foram verificados os gráficos de accuracy e loss obtidos aquando do treino da rede (Figura 4) para verificar a curva de aprendizagem da mesma. Os hiperparâmetros utilizados foram os que constam na terceira linha da tabela 1. Pela análise do gráfico de accuracy, percebemos que os valores de accuracy chegaram a um *learning plateau* pelo que o valor de epochs (15) é suficiente para o treino da rede.

Pela análise do gráfico de loss, não é possível identificar problemas de *underfitting*, *overfitting* ou *unrepresentative data*, pelo que não foi necessária uma análise mais aprofundada dessa questão.

3.3.2 ResNet-50 (PyTorch)

A arquitetura da ResNet-50 no PyTorch é uma rede neuronal convolucional profunda composta por 50 camadas que se destaca pela utilização de blocos residuais para mitigar o desafio do desaparecimento do gradiente em redes profundas.

Foram realizados diversos testes de desempenho da arquitetura alterando os hiperparâmetros:

- **EPOCHS**
- **BATCH_SIZE**
- **Learning rate (lr)**

Os modelos foram treinados e validados na plataforma Kaggle com recurso ao acelerador GPU P100 e os valores que constam na tabela 2 referentes ao tempo referem-se ao tempo de treino dos modelos (*wall time*). Não foram utilizados valores de *batch size* maiores devido às limitações de memória da plataforma.

batch_size	lr	epoch	elapsed time (min)	accuracy (%)
32	0.001	15	12	67.2
64	0.001	15	10	65.8
32	0.002	15	11	62.2
64	0.002	15	10	73.2
32	0.005	15	11	84.7
64	0.005	15	10	67.3
32	0.01	15	10	72.8
64	0.01	15	10	62.3
32	0.001	30	23	63.5
64	0.001	30	20	67.3
32	0.002	30	23	72.5
64	0.002	30	20	68.7
32	0.005	30	22	67.8
64	0.005	30	20	68.3

Table 2: Resultados de Treino da ResNet-50 (PyTorch)

Pela análise da tabela, percebemos que esta arquitetura não atingiu valores satisfatórios, em geral. Apenas o valor da quinta linha tem uma accuracy maior de 84.7%. Porém, após treinar e testar novamente o modelo, percebemos que este apresenta o mesmo problema do que a arquitetura AlexNet, ou seja, obtém um desempenho razoável em termos de verdadeiros positivos, porém tem dificuldade em identificar se existe ou não fraturas em várias imagens das duas classes.

Por fim, fez-se a análise dos gráficos de accuracy e loss (Figura 6) tal como na arquitetura AlexNet, verificou-se que a rede atinge um *learning plateau*, pelo que 15 epochs é suficiente e que, no gráfico de loss não se identifica nenhum padrão de *underfitting*, *overfitting* ou *unrepresentative data*, pelo que não foi necessária uma análise mais aprofundada dessa questão.

3.3.3 DenseNet (MONAI)

A DenseNet (Redes Neurais Densas) é uma arquitetura de rede neuronal convolucional (CNN) conhecida pela sua eficácia em tarefas de visão computacional, como segmentação de imagem e classificação.

Foram realizados diversos testes de desempenho da arquitetura alterando os hiperparâmetros:

- **block_config:** refere-se a uma configuração específica utilizada na arquitetura DenseNet para determinar a disposição dos blocos de densidade (Dense Blocks) na rede. Cada elemento no block_config representa o número de camadas convolucionais em cada bloco de densidade ao longo da rede.
- **Learning rate (lr)**

Os modelos foram treinados e validados na plataforma Kaggle com recurso ao acelerador GPU P100 e os valores que constam na tabela 3 referentes ao tempo referem-se ao tempo de treino dos modelos (*wall time*).

block_config	lr	elapsed time (min)	accuracy (%)
(6, 12, 24, 16)	0.001	7	60.2
(6, 12, 24, 16)	0.01	7	73.8
(6, 12, 24, 16)	0.1	7	78.3
(6, 12, 32, 24)	0.001	9	60.7
(6, 12, 32, 24)	0.01	9	64.5
(6, 12, 32, 24)	0.1	9	79.8
(8, 16, 40, 32)	0.001	12	68.2
(8, 16, 40, 32)	0.01	12	69.7
(8, 16, 40, 32)	0.1	12	83.5

Table 3: Resultados de Treino da DenseNet (MONAI)

Não foi variado o hiperparâmetro “batch_size” devido a limitações de memória (manteve-se o valor de 32) e não foi variado o hiperparâmetro “epochs” uma vez que a rede chegou ao *learning plateau* como se pode verificar na Figura 7 (manteve-se o valor de 10).

Inicialmente, foram analisadas as probabilidades na camada de saída, sendo constatado que o modelo apresentava elevadas probabilidades (geralmente acima de 99%) para os casos em que acertava e probabilidades mais baixas para os casos em que errava.

Por este motivo, o grupo considerou que a rede não tinha capacidade suficiente para aprender todas as características, pelo que foi aumentado o tamanho da rede. Estas variações e o aumento do *learning rate*, resultaram num modelo com mais capacidade de previsão e mais estável em termos de probabilidades na camada de saída. Com esta arquitetura foi possível resolver o problema dos modelos anteriores que era a identificação correta dos casos em que não existia fratura, atingindo 83.5% de accuracy.

Na Figura 8, está representada a matriz de confusão obtida a partir do teste da rede cujos hiperparâmetros constam na última linha da tabela 3.

3.3.4 EfficientNetBN (MONAI)

A arquitetura EfficientNetBN do MONAI é uma implementação baseada na EfficientNet, sendo um modelo de rede neuronal eficiente em termos computacionais e com um desempenho superior em tarefas de visão computacional como segmentação e classificação de imagens médicas.

Foi decidida a utilização desta arquitetura pela sua eficiência de utilização de recursos computacionais, o que se esperava ser um ponto positivo devido às limitações de recursos da plataforma.

Foram realizados diversos testes de desempenho da arquitetura alterando os

hiperparâmetros:

- **model_name:** variantes do EfficientNet, especificadas por um parâmetro chamado b (como EfficientNetB0, B1, B2, etc.). Cada variante possui uma estrutura de rede neuronal convolucional com diferentes níveis de profundidade e complexidade, sendo B0 o menor e mais simples e B7 o maior e mais complexo.
- **Learning rate (lr)**

Os modelos foram treinados e validados na plataforma Kaggle com recurso ao acelerador GPU P100 e os valores que constam na tabela 4 referentes ao tempo referem-se ao tempo de treino dos modelos (*wall time*).

model_name	lr	elapsed time (min)	accuracy (%)
efficientnet-b0	0.001	7	71.2
efficientnet-b0	0.0001	7	79.3
efficientnet-b1	0.001	7	72.0
efficientnet-b1	0.0001	7	74.8
efficientnet-b2	0.001	8	65.3
efficientnet-b2	0.0001	8	65.2

Table 4: Resultados de Treino da EfficientNetBN (MONAI)

Pela análise das probabilidades obtidas aquando do treino do primeiro modelo obtido, foi possível detetar que o modelo apresentava probabilidades altas mesmo quando falhava, o que poderia indicar problemas de overfitting. Por esta razão, decidimos testar o modelo com o learning rate menor, o que mostrou algumas melhorias.

Ao contrário do que aconteceu com a arquitetura anterior (DenseNet), o aumento do tamanho da rede não levou a um aumento da accuracy do modelo, o que foi de encontro às expectativas do grupo uma vez que a arquitetura anterior apresentava mais incerteza nas previsões erradas, pelo que beneficiaria de um tamanho de rede maior com capacidade para mais informação, ao contrário da arquitetura EfficientNetBN.

No entanto, apesar desta arquitetura demonstrar um desempenho menor do que a DenseNet, comparativamente às arquiteturas AlexNet e ResNet do Pytorch, teve um melhor desempenho sobretudo na classificação correta das imagens em que não constam fraturas o que pode ser justificado pelo facto de ser uma arquitetura da biblioteca MONAI que são arquiteturas mais apropriados para classificação de imagens médicas.

Não foi variado o hiperparâmetro “epochs” dado que o valor de 10 foi suficiente para os modelos convergirem e não foi variado o hiperparâmetro “batch_size” devido a limitações de memória, pelo que se manteve com o valor de 32.

Nas Figuras 9 e 10 estão representados os gráficos de accuracy e loss e a matriz de correlação, respetivamente, para o modelo cujos hiperparâmetros constam na segunda linha da tabela 4 (modelo com a maior accuracy).

3.3.5 ViT (MONAI)

O ViT (Vision Transformer) é uma arquitetura de rede neuronal desenvolvida inicialmente para processamento de imagens, especialmente para tarefas de visão computacional. O MONAI adota e adapta o ViT para aplicações em imagens médicas.

O motivo de usar este tipo de arquitetura foi o potencial que poderia ter para classificação de um conjunto de imagens com características mais complexas que é o caso em questão, bem como explorar alternativas de arquiteturas de Deep Learning.

Foram realizados diversos testes de desempenho da arquitetura alterando os hiperparâmetros:

- **num_layers:** determina quantas camadas de transformadores são empilhados na arquitetura do ViT.
- **patch_size:** determina o tamanho dos patches em que a imagem de entrada é dividida antes de ser alimentada na rede.
- **Learning rate (lr)**
- **EPOCHS**

Os modelos foram treinados e validados na plataforma Kaggle com recurso ao acelerador GPU P100 e os valores que constam na tabela 5 referentes ao tempo referem-se ao tempo de treino dos modelos (*wall time*).

num_layers	patch_size	lr	epochs	elapsed time (min)	accuracy (%)
12	32	0.001	100	72	60.3
12	32	0.0001	70	47	52.7
6	16	0.0001	70	81	70.7
6	16	0.0001	50	58	65.3
24	64	0.0001	50	34	61.3
24	64	0.0001	30	20	68.8

Table 5: Resultados de Treino do ViT (MONAI)

No primeiro conjunto de hiperparâmetros foram usados 100 epochs para perceber em que ponto a rede chegava ao *learning plateau*. O ponto era atingido pouco antes dos 70 epochs pelo que decidimos diminuir o valor para 70. Foi feita também uma análise das probabilidades na camada de saída, sendo constatado que a rede errava com probabilidades a rondar os 85% pelo que poderia indicar overfitting, e então foi decrementado também o learning rate.

Na segunda tentativa o problema persistiu, pelo que decidimos diminuir o tamanho da rede bem como o patch_size e, a seguir, aumentar estes valores para perceber de que forma poderiam alterar os resultados. Apesar do problema persistir em todas as tentativas, esta arquitetura demonstrou um melhor desempenho a detetar corretamente imagens sem fraturas ao contrário das duas redes iniciais

que não pertenciam à biblioteca MONAI, como se pode verificar na Figura 11, onde está retratada a matriz de correlação para os hiperparâmetros da última linha da tabela 5.

Pela análise dos gráficos de accuracy e loss obtidos aquando do treino do último modelo, pode haver indícios de *unrepresentative data* uma vez que a perda na validação é menor do que a perda no treino.

4 Análise de resultados

4.1 Resultados obtidos e Análise crítica

Da análise dos resultados de treino das cinco arquiteturas que utilizamos, podemos concluir que a DenseNet (MONAI) é aquela que mais se adequa ao nosso dataset pela sua accuracy e análise das probabilidades da camada de saída.

Dos restantes modelos, os modelos do PyTorch (AlexNet e ResNet-50), apesar de contarem com pré-treino, apresentaram problemas na classificação correta dos casos em que não existia fraturas.

Os modelos da plataforma MONAI apresentaram melhores resultados sobretudo na classificação correta dos casos em que não existiam fraturas o que pode ser justificado pelo pré-treino das redes com imagens médicas. Apesar disso, as arquiteturas EfficientNet e ViT não conseguiram alcançar um desempenho e estabilidade (em termos de probabilidades na camada de saída) tão alto como a DenseNet.

Do estudo do dataset, concluímos que o mesmo apresenta imagens bastante difíceis de classificar mesmo para o olho humano como, por exemplo, o caso de lesões nos pulsos, dificultando a rede de aprender a distinção entre o normal e o fraturado. No entanto, com o aumento do tamanho da rede DenseNet foi possível estender a capacidade de previsão dos modelos e alcançar uma accuracy bastante aceitável para o problema em questão (83.5%).

A tabela 6 mostra o desempenho do melhor modelo obtido com cada arquitetura bem como o tempo utilizado para o treino da mesma na plataforma Kaggle com o acelerador GPU P100. É importante referir que embora a ResNet-50 tenha o modelo com a maior accuracy, o modelo não é tão estável em termos de probabilidades na camada de saída como os modelos da biblioteca MONAI. Os modelos guardados encontram-se na Google Drive do projeto. A pasta está organizada em 5 subpastas cada uma com os modelos de uma arquitetura. As subpastas referentes às arquiteturas do PyTorch contêm apenas o modelos com o melhor desempenho enquanto que as do MONAI, contêm todos os modelos ordenados pela linha da coluna nas tabelas referentes.

Arquitetura	Biblioteca	Accuracy do melhor modelo (%)	Tempo de treino (min)
AlexNet	PyTorch	67.7	7
ResNet-50	PyTorch	84.7	11
DenseNet	MONAI	83.5	12
EfficientNetBN	MONAI	79.3	7
ViT	MONAI	70.7	81

Table 6: Comparação de desempenhos das diferentes arquiteturas

4.2 Trabalho futuro

Futuramente, poder-se-ia alargar este estudo sobre Deep Learning com outras arquiteturas do MONAI para permitir explorar novas alternativas e até mesmo aprender mais sobre o dataset em questão com a análise do desempenho e possíveis desafios de outros modelos.

Além disso, para aprofundar o estudo já feito, poder-se-ia testar e otimizar modelos baseados na DenseNet noutra ambiente de execução com mais recursos computacionais. Isto permitiria ampliar ainda mais o tamanho da rede e avaliar o desempenho da mesma segundo essas alterações de hiperparâmetros. Como o desempenho tendeu a melhorar com o aumento tamanho da rede (e aumento do learning rate), valores ainda maiores poderiam resultar numa accuracy maior dos modelos.

5 Conclusão e considerações finais

Em resumo, consideramos que conseguimos explorar o dataset e realizar um estudo bastante abrangente sobre a aplicação de redes neurais convolucionais sobre o mesmo.

Além disso, conseguimos alcançar resultados bastante satisfatórios, nomeadamente com o uso da arquitetura DenseNet da biblioteca MONAI que se revelou a melhor arquitetura dentre todas as utilizadas para o nosso problema dado poder cobrir todos os casos de fraturas.

6 Referências

- [1] “bone fracture detection using x-rays, Kaggle”, <https://www.kaggle.com/datasets/vuppalaadithyasairam/bone-fracture-detection-using-xrays>.
- [2] “Models and pretrained weights, PyTorch”, <https://pytorch.org/vision/stable/models.html>.
- [3] “Project MONAI, MONAI”, <https://docs.monai.io/en/stable/index.html>.

7 Apêndices

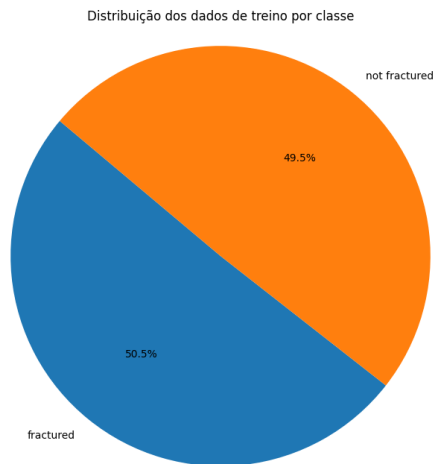


Figure 1: Distribuição dos dados de treino por classe.

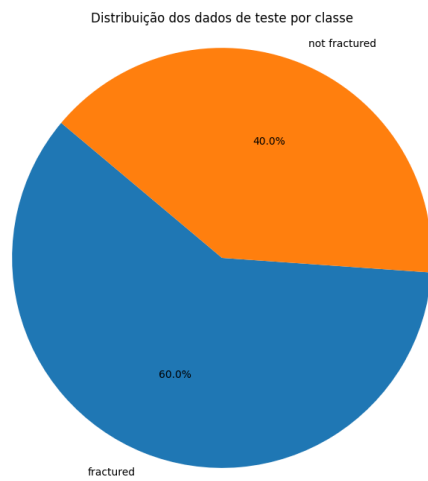


Figure 2: Distribuição dos dados de teste por classe

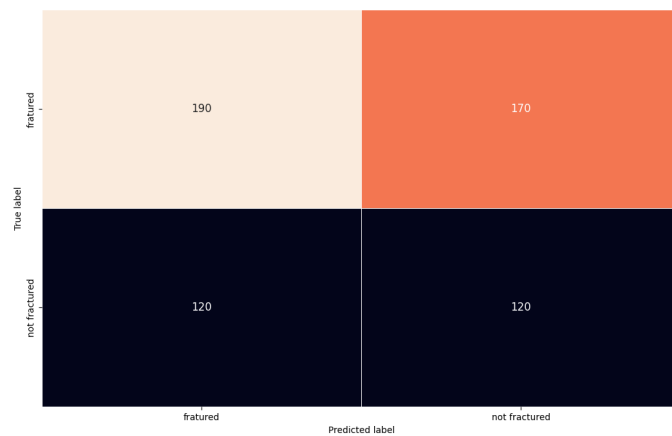


Figure 3: Matriz de confusão da AlexNet (PyTorch)

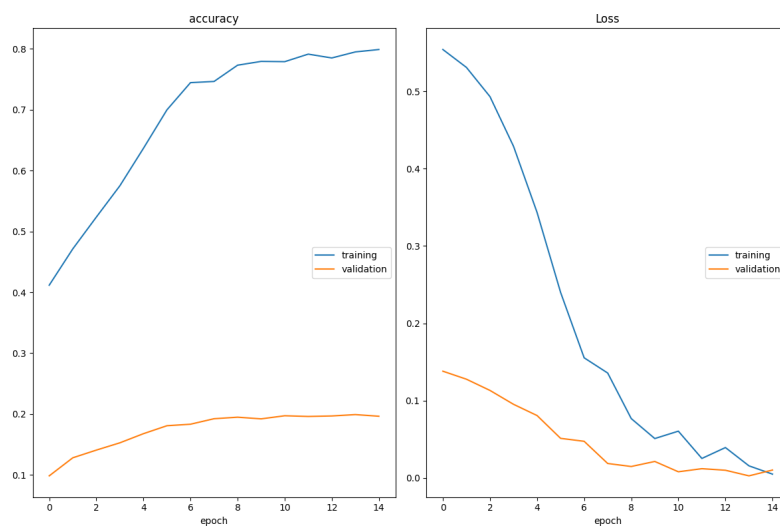


Figure 4: Gráficos de accuracy e loss da AlexNet (PyTorch)

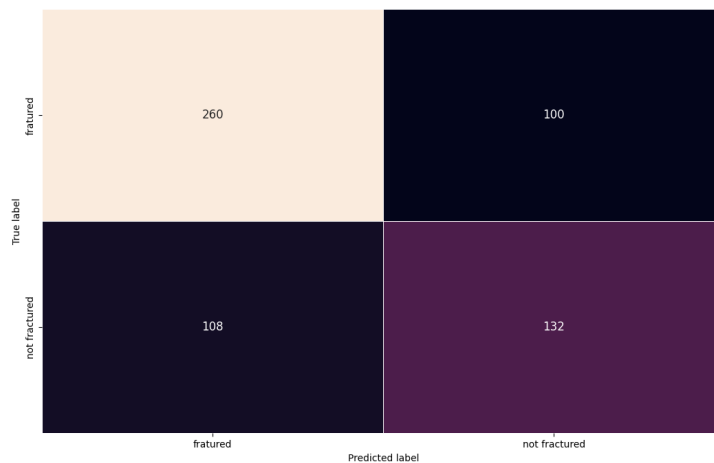


Figure 5: Matriz de confusão da ResNet (PyTorch)

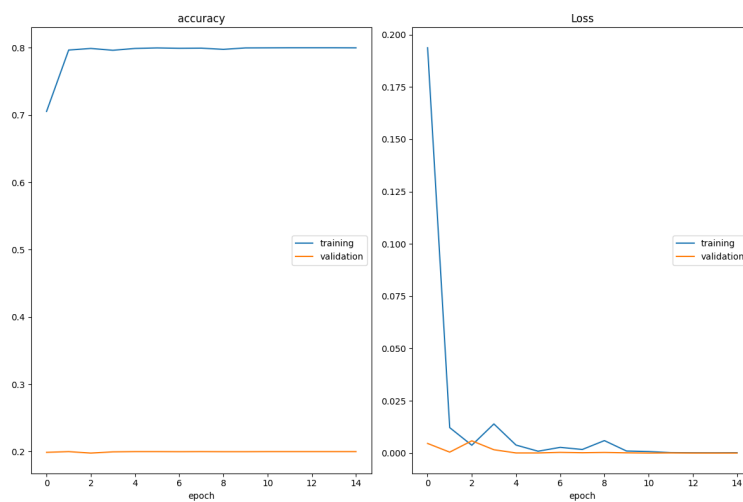


Figure 6: Gráficos de accuracy e loss da ResNet (PyTorch)

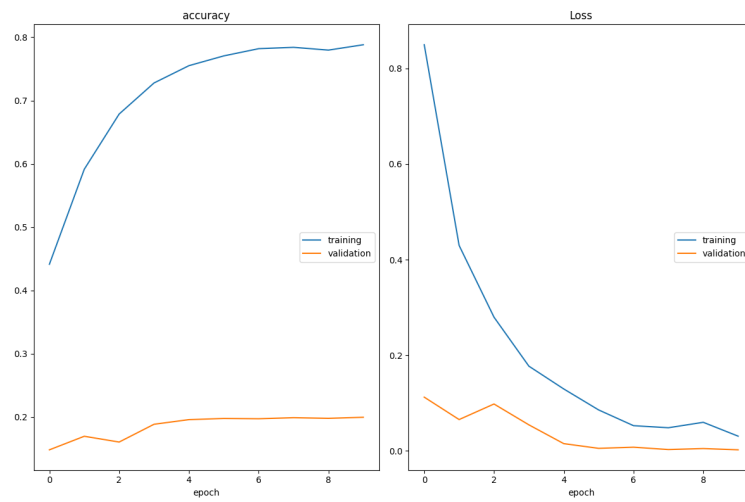


Figure 7: Gráficos de accuracy e loss da DenseNet (MONAI)

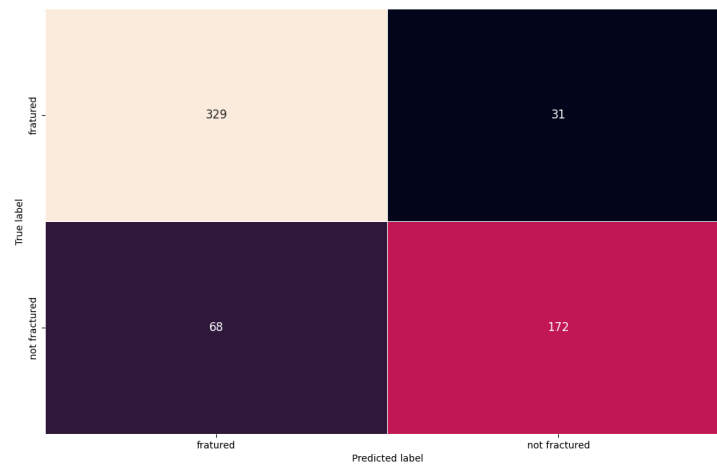


Figure 8: Matriz de confusão da DenseNet (MONAI)

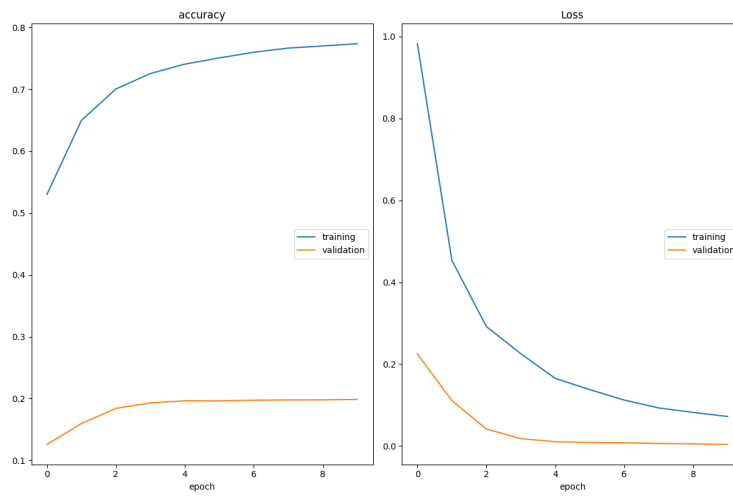


Figure 9: Gráficos de accuracy e loss da EfficientNetBN (MONAI)

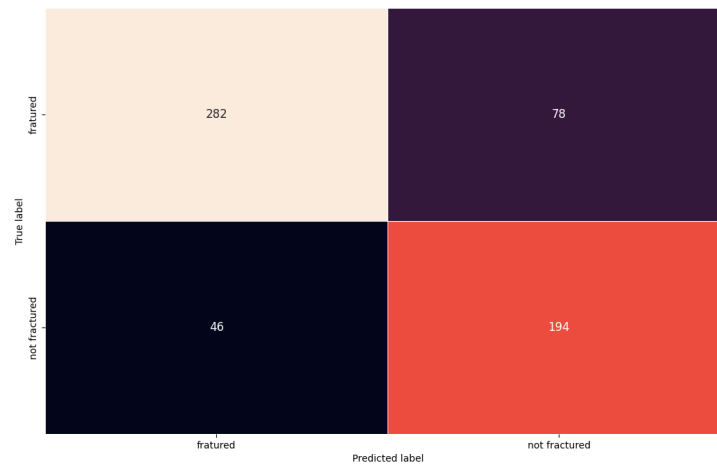


Figure 10: Matriz de confusão da EfficientNetBN (MONAI)

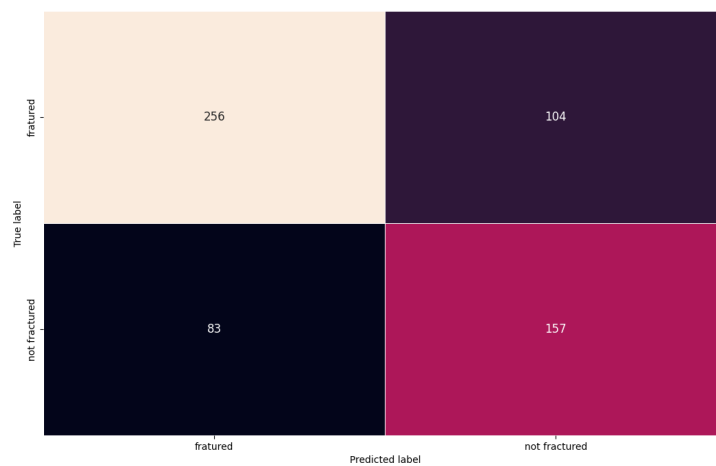


Figure 11: Matriz de confusão do ViT (MONAI)

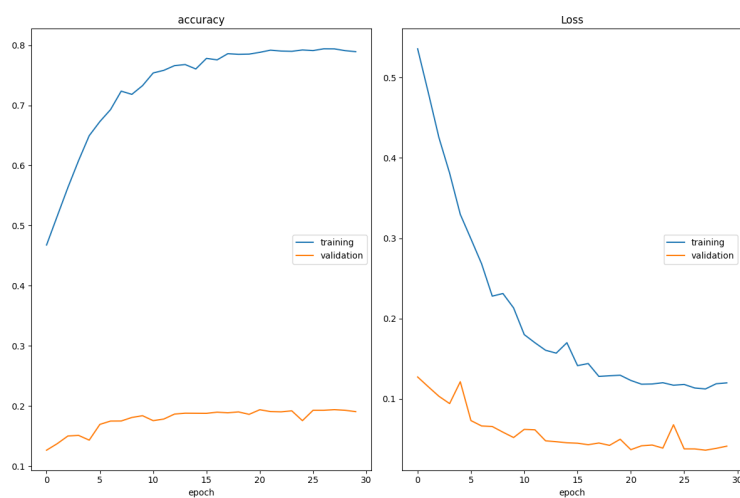


Figure 12: Gráficos de accuracy e loss do ViT (MONAI)