Documentação Projeto final de programação

Santiago Vallejo Silva

PUC-Rio, Departamento de Informática

Ferramenta para treinamento e analise de modelos de classificação binaria em imagens

Contexto

Os modelos classificadores binários de imagens permitem reconhecer automaticamente a qual classe pertence uma imagem baseado nas características dela, o modelo vai retornar um valor numérico entre 0 e 1, se o valor está próximo a um dos extremos, pode se dizer que a imagem pertence a uma classe ou a outra.

Esse tipo de modelos são utilizados em diferentes contextos, já que a quantidade de bases de dados que podem se encontrar na rede é grande. Os usos podem ir desde identificar cachorros e gatos, até identificar se por exemplo uma peça de produção é de boa qualidade, ou classificar diagnósticos médicos, podem se utilizar quando a quantidade de dados a classificar é maior a que um usuário humano consegue classificar manualmente.

Escopo

A ideia da ferramenta mostrada nesse trabalho foi inicialmente pensando para trabalhar com apenas um conjunto de dados, mas quando a programação avançou foi possível ver que a ferramenta pode ter um maior alcance, permitindo criar modelos para qualquer conjunto de dados de imagens que tenham um patrão comum, que sejam imagens com a estrutura de pastas certa para criar um modelo de classificação binaria.

Por tanto, entre os usuários visados estão aqueles desenvolvedores ou programadores de sistemas de classificação de *machine learnin*g, ou aqueles estudantes iniciantes no mundo da programação com keras, já que o código da ferramenta é facilmente adaptável para novas estruturas de modelos, além disso,

com um pouco mais de trabalho, a ferramenta pode ser adaptada para criar modelos de classificação multiclasse.

Critérios de aceitação

É esperado que o software permita criar modelos de classificação binaria usando o conjunto de dados fornecido pelo usuário e utilizando os parâmetros definidos para cada experimento.

Além disso, o software deve mostrar os resultados de todos os experimentos de um ou vários conjuntos de dados, de forma organizada e mostrando para cada um deles as métricas obtidas, junto com uma visualização dos gráficos e as imagens individuais, permitindo ao usuário observar em quais imagens o modelo está falhando.

Diagrama de sequência

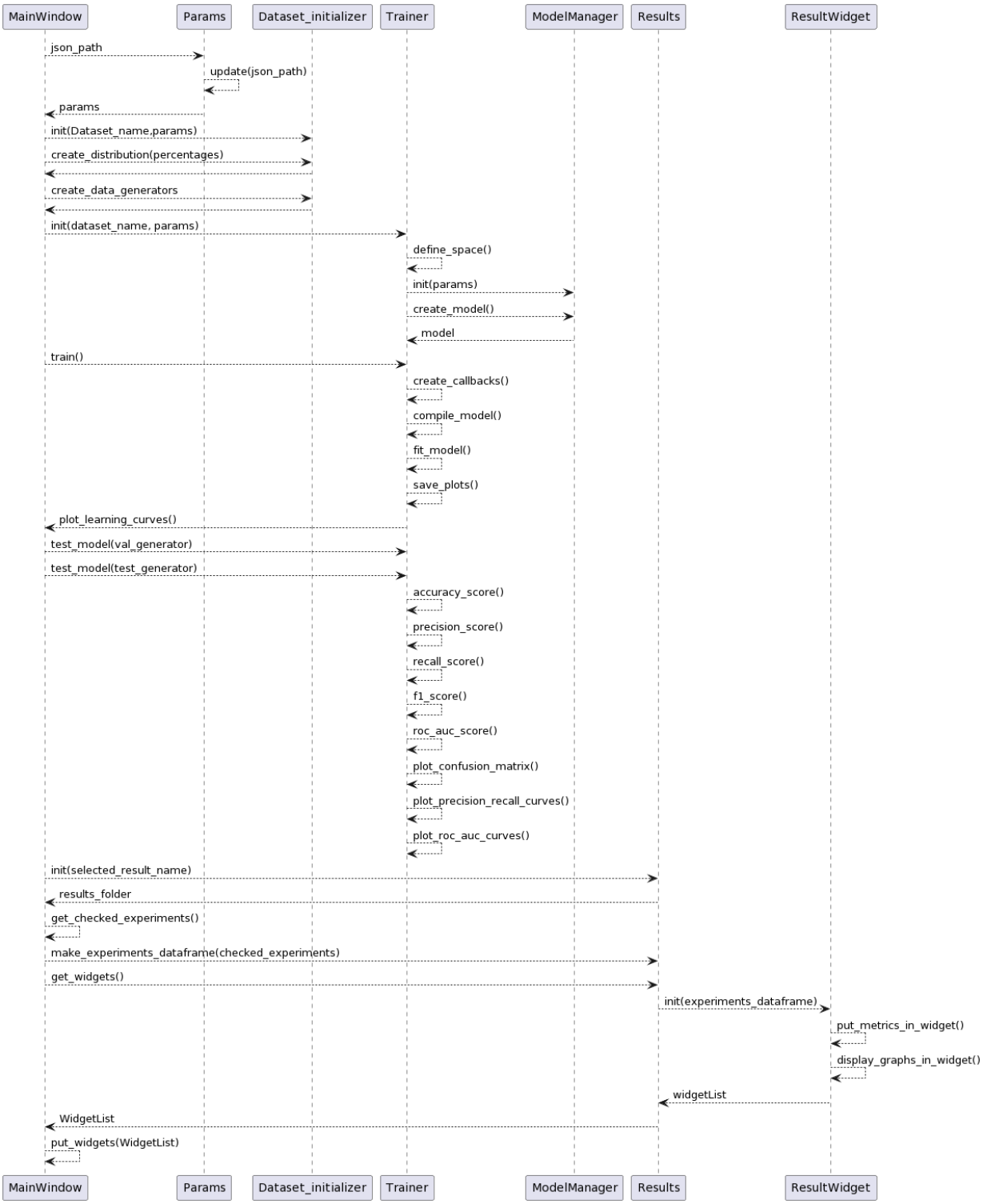
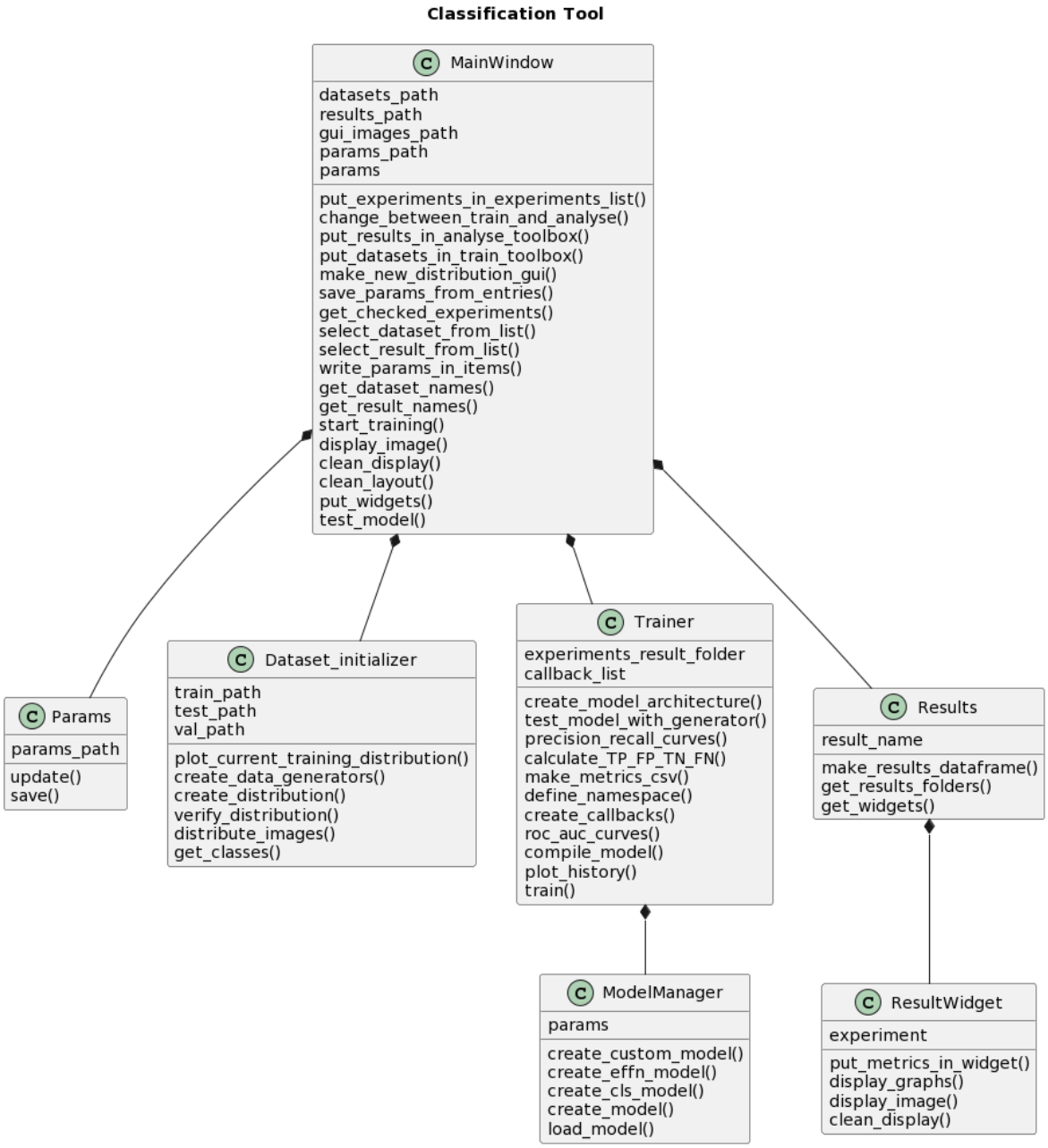


Diagrama de classes



Sobre as funcionalidades:

MainWindow: classe encarregada de mostrar a parte visual, identificando os conjuntos de dados e os resultados, permite observar a distribuição do dataset selecionado e o status atual do treinamento, além de mostra todos os resultados gerados.

Dataset_initializer: permite identificar a estrutura de pastas do dataset selecionado, permite fazer uma nova distribuição dos dados e criar os generators (necessários para começar o treinamento)

Trainer: classe encarregada de treinar e testar o modelo, cria as pastas necessárias para alocar todos os resultados, faz plots e calcula e salva as métricas em arquivos csv.

Results classe encarregada de ler os resultados obtidos, organiza todos os resultados selecionados e permite filtra-los de acordo com as métricas.

Testes do software

Teste do treinamento:

Foram realizados vários treinamentos, verificando que o software utilize sempre o *dataset* selecionado e os parâmetros certos, para cada um desses treinamentos foi observado o progresso, para ver o correto funcionamento do progresso, que pode se observar numa barra de progresso e os plots de *accuracy* e *loss* que são atualizados em cada época. Tudo resulto com um bom funcionamento.

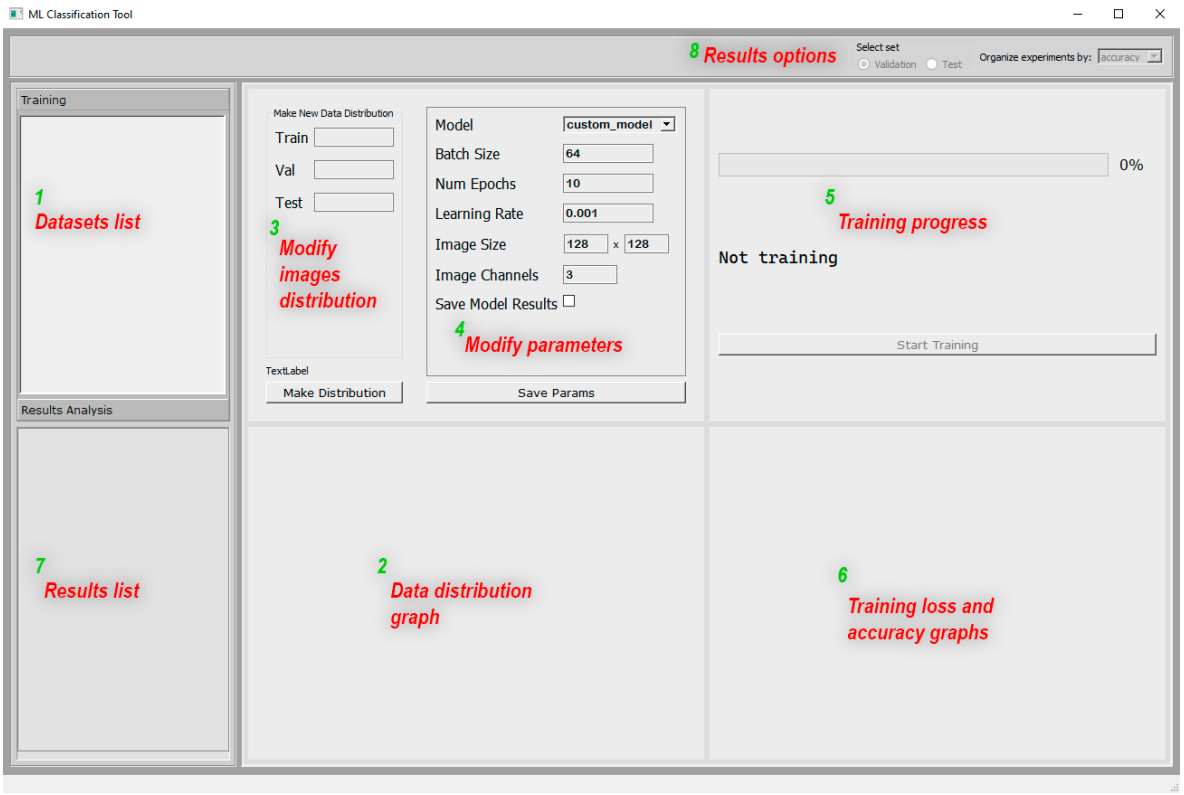
Teste da visualização dos resultados:

Tendo vários treinamentos realizados com três conjuntos de dados diferentes, foi feita uma verificação do software ao momento de mostrar os resultados na tela, observando que todas as métricas correspondem aos resultados selecionados, e verificando que as imagens individuas mostradas correspondem com a quantidade mostrada nas métricas. Tudo resulto com um bom funcionamento.

Documentação para o usuário

1. Interface
2. Datasets
3. Treinamento
4. Analise

1.Interface



[1] Mostra os *datasets* disponíveis

[2] Mostra o gráfico da distribuição dos dados

[3] Permite mudar a distribuição por defeito

[4] Permite modificar os parâmetros de treinamento

[5] Mostra a atividade atual e o progresso do treinamento

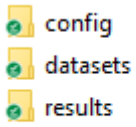
[6] Mostra os gráficos de *loss* e *accuracy* do treinamento

[7] Mostra os resultados disponíveis de cada experimento

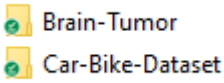
[8] Permite ordenar os resultados de acordo com as métricas e as imagens individuais por FN, FP, TN e TP

2. Datasets

Dentro da pasta *Data* pode se encontrar a pasta *datasets*.



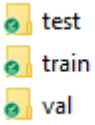
Dentro da pasta *datasets* devem se colocar os *datasets* a ser utilizados, como é mostrado na imagem a seguir:



Dentro de uma pasta de *dataset* podem ser colocar os dados de uma das seguintes maneiras:

A) As pastas com os nomes das classes

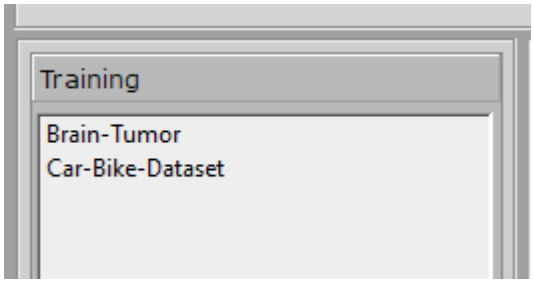
B) As pastas *train*, *val* e *test*



É recomendável usar o *dataset* do jeito A, já que assim é possível realizar uma distribuição das imagens conforme o usuário requeira.

3. Treinamento

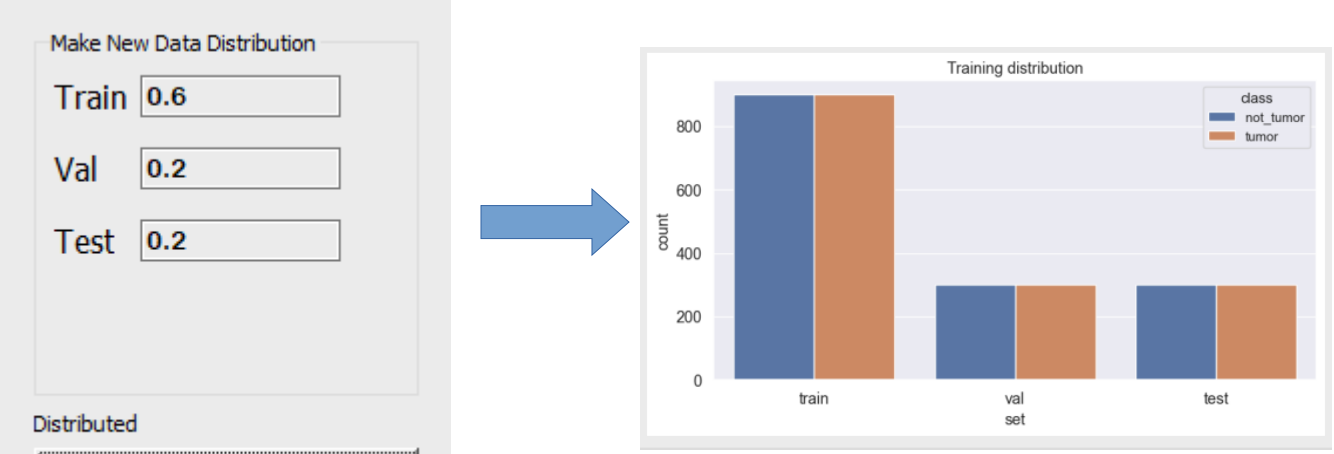
Com o *dataset* na pasta certa já é possível começar com o treinamento.



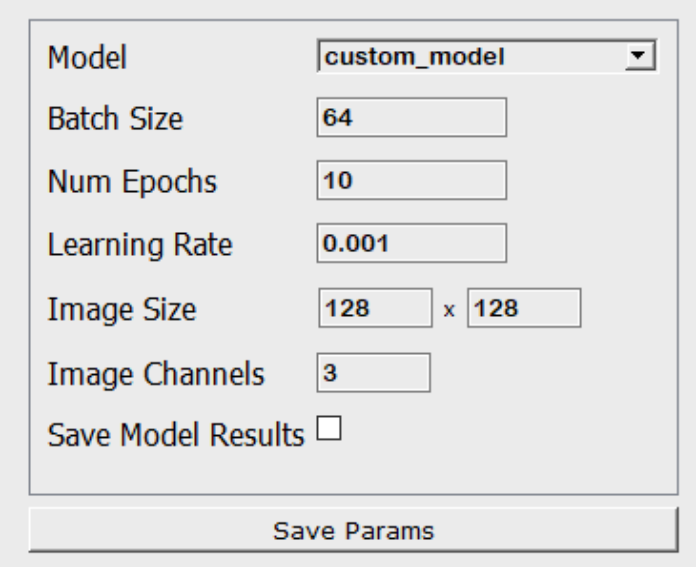
Na aba de treinamento vão aparecer os *datasets* colocados, é necessário clicar duas vezes sobre o *dataset* que quer ser utilizado, ao fazer isso o programa vai mostrar um gráfico com a distribuição do *dataset*, por defeito sendo:

- 70% das imagens para treinamento
- 20% das imagens para validação
- 10% das imagens para teste

O usuário pode mudar a distribuição usando as caixas de texto, colocando a porcentagem como um *float*, os 3 números devem somar 1.

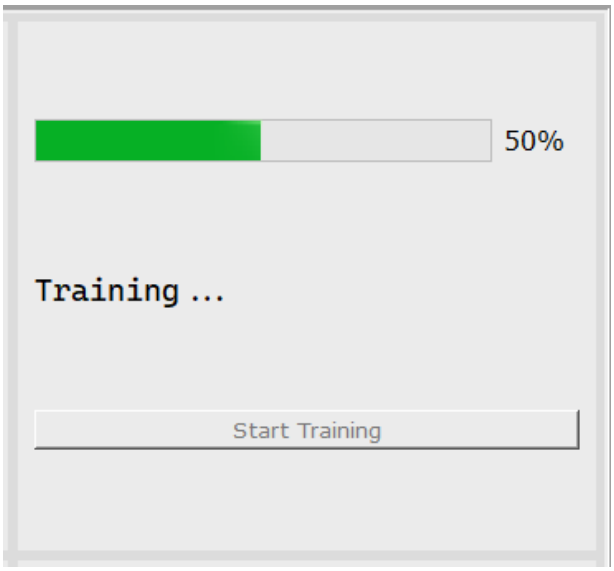


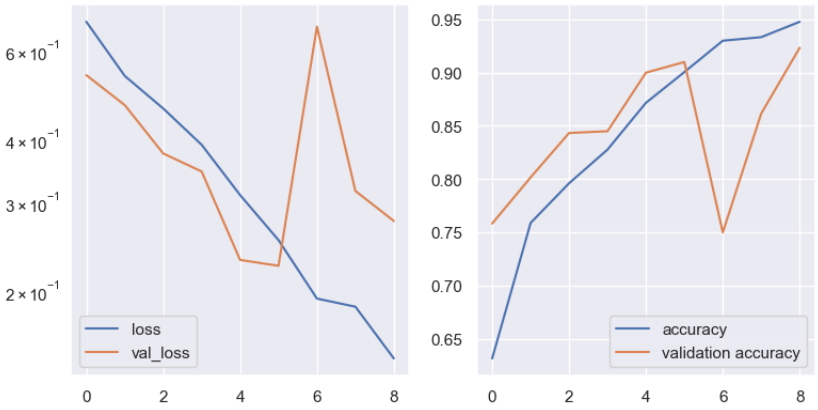
Além disso, o usuário pode mudar os parâmetros do treinamento, como a arquitetura que vai utilizar, o número de épocas ou learning rate.



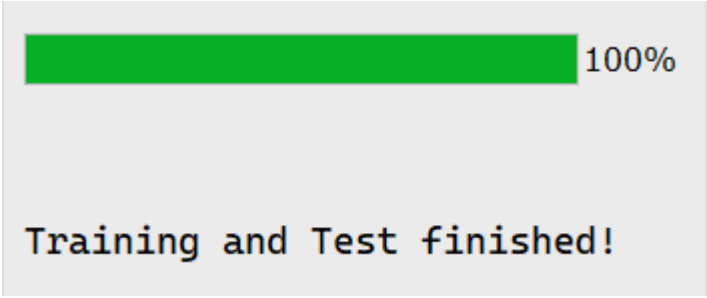
Depois de fazer as mudanças é preciso clicar no botão *Save Params.*

Feito isso, já está tudo pronto para começar o treinamento, o botão Start Training já deveria estar habilitado, clicando nele o treinamento vai começar, será mostrada uma barra de progresso e os gráficos de *loss* e *accuracy.*





Quando o treinamento estiver pronto, o programa vai indica-lo



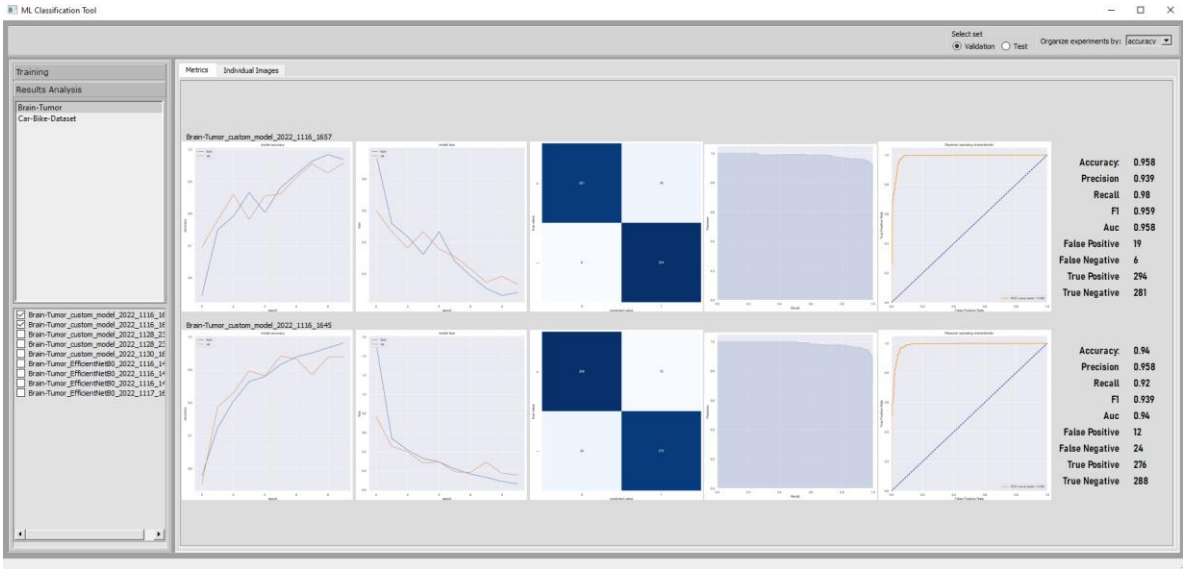
3. Análise dos resultados

Quando um treinamento termina é gerado um novo elemento na lista de experimentos, um experimento corresponde a um ou vários treinamentos realizados com um mesmo *dataset*, basta clicar duas vezes num experimento para ver a lista dos resultados dele.

Os resultados são elementos selecionáveis, quando um resultado é selecionado aparecem por defeito os resultados do modelo sobre o conjunto de validação, podem ser vistos também os resultados sobre o conjunto de teste.

Na esquina superior direita o usuário pode ordenar os resultados em ordem descendente de acordo com as distintas métricas, as quais são:

- Accuracy
- Precision
- Recall
- F1
- Quantidade de falsos positivos
- Quantidade de falsos negativos
- Quantidade de verdadeiros positivos
- Quantidade de verdadeiros negativos



Na aba *Individual Images* aparecem aquelas imagens que o usuário precise olhar, as quais podem ser:

- Falsos positivos
- Falsos negativos
- Verdadeiros positivos
- Verdadeiros negativos

Clicando no nome da imagem o texto vai ser copiado no *clipboard* do computador.

Clicando numa imagem vai se abrir o explorador de arquivos, tendo copiado o nome da imagem é possível realizar uma busca rápida dela.