

Teachable Machine

Inteligência Artificial - FIAP - 2024 / 2

Turma: Graduação - 1TIAOR - 2024/2

Alunos:

- Bruno Henrique Conterato - RM561048
- Esdras Santos Almeida - RM560390
- Kety Karoline Prestes - RM559480
- Ludimila Fernanda Vitorino da Conceição - RM559697
- Matheus Francelino dos Santos Silva - RM559762

Objetivos

Os principais objetivos da atividade são:

1. Desenvolver um modelo de machine learning capaz de detectar e classificar diferentes utensílios de cozinha por meio de sua imagem.
2. Familiarização com conceitos básicos de aprendizado de máquina e de visão computacional.
3. Avaliar a eficiência dos modelos treinados.

Scripts

Foram desenvolvidos dois scripts de código para auxiliar o desenvolvimento do projeto. Ambos podem ser encontrados no repositório: <https://github.com/brunoconterato/Kitchen-Utensils> diretamente na pasta src.

1. Script de Inspeção e Separação de Imagens

Objetivos:

- Inspeção de imagens:
 - Verificação de classes;
 - Contagem de elementos das classes;
 - Tradução dos nomes das classes para português brasileiro;
 - Visualização de elementos das classes.
- Separação de imagens em treino e teste;
 - Separação por classe em 80% treino e 20% teste;
 - Cópia das imagens para diretórios de treino e teste separados para facilitar o upload no Teachable Machine;
 - Resumo final sobre as classes selecionadas e suas respectivas quantidades de imagens.

2. Script de Avaliação de Modelos

Objetivos:

- Carregamento de dados:
 - Carregar corretamente as imagens de teste;
 - Obter corretamente os labels de teste a partir dos nomes das pastas dos arquivos.
- Carregamento do modelo treinado no Teachable Machine;
 - Carregar metadados do modelo treinado:
 - número de épocas de treinamento;
 - learning rate utilizado no treinamento;
 - tamanho do batch utilizado no treinamento.
- Avaliação dos modelos carregados
 - Avaliar a acurácia dos modelos carregados nos dados de teste;
 - Selecionar os melhores modelos;
 - Obter gráficos para investigar a relação entre as variáveis de treinamento e a acurácia obtida.

Metodologia e avaliação de resultados experimentais:

1. Coleta de dados

- a. Os dados foram coletados do Dataset Utensil Image Recognition, disponível no Kaggle

<https://www.kaggle.com/datasets/jehanbhathena/utensil-image-recognition>

The screenshot shows the Kaggle interface with the dataset 'Utensil Image Recognition' selected. The left sidebar shows various sections like Home, Competitions, Datasets, Models, etc. The main area displays a grid of 20 categories, each with a folder icon and the number of files it contains. To the right is a detailed file tree showing the structure of the dataset, including sub-folders for binary files and specific utensil types like BOTTLE_OPENER, BREAD_KNIFE, etc. A summary at the bottom indicates there are 1803 files in total.

- b. Esse Dataset contém utensílios de cozinha classificados em 20 classes, cada classe em pastas separadas que levam seu nome. O dataset também contém arquivos binários (preto e branco) dos objetos de cada classe, estes não utilizaremos na nossa análise. São as classes:

"TONGS": "PINÇAS"

"DESSERT_SPOON": "COLHER DE SOBREMESA"

"PEELER": "DESCASCADOR"

"POTATO_PEELER": "DESCASCADOR DE BATATAS"

"KITCHEN_KNIFE": "FACA DE COZINHA"

"PIZZA_CUTTER": "CORTADOR DE PIZZA"

"WOODEN_SPOON": "COLHER DE MADEIRA"

"DINNER_FORK": "GARFO DE JANTAR"

"LADLE": "CONCHA"

"FISH_SLICE": "ESPÁTULA DE PEIXE"

"MASHER": "AMASSADOR"

"SPATULA": "ESPÁTULA"

"TEA_SPOON": "COLHER DE CHÁ"
"DINNER_KNIFE": "FACA DE JANTAR"
"BREAD_KNIFE": "FACA DE PÃO"
"WHISK": "FUÊ"
"BOTTLE_OPENER": "ABRIDOR DE GARRAFAS"
"SOUP_SPOON": "COLHER DE SOPA"
"CAN_OPENER": "ABRIDOR DE LATAS"
"SERVING_SPOON": "COLHER DE SERVIR"

- c. Escolhemos 8 delas para serem nossas classes. Os critérios de escolha foram:
 - i. Escolhemos itens que são mais comuns nas cozinhas, a fim de que a aplicação possa ser útil a um número maior de pessoas.
 - ii. Escolhemos 3 tipos de facas para verificar a capacidade de o modelo diferenciar cada uma delas. São as classes escolhidas:

"COLHER DE SOBREMESA"
"FACA DE COZINHA"
"FACA DE JANTAR"
"GARFO DE JANTAR"
"FACA DE PÃO"
"CONCHA"
"ESPÁTULA DE PEIXE"
"ABRIDOR DE GARRAFAS"

2. Separação dos dados

- a. Uma vez definidas as classes, criamos um script de inspeção e separação de imagens em python que lê as pastas do dataset original (ou seja, as classes) e divide cada classe em dados de treinamento e teste (proporção de 80%/20% aleatoriamente).
- b. Não fazemos aqui a divisão de dataset de validação, uma vez que o próprio Teachable machine já faz essa separação e separa dados para validação durante o treinamento. O dataset de validação, assim como o de teste, não é visto pelo modelo em etapa de treinamento, ele apenas serve para avaliar cada época de treinamento e escolher qual o checkpoint do modelo é o melhor com base nos dados. Essa divisão de dados em treino/validação/teste recebe o nome de validação cruzada (cross validation), é uma das boas práticas em aprendizagem de máquina para garantir que os modelos treinados sejam capazes de aprender com os dados de treinamento, e também manter capacidade de fazer inferência sobre exemplos não vistos durante o treinamento. A essa capacidade nomeamos generalização.

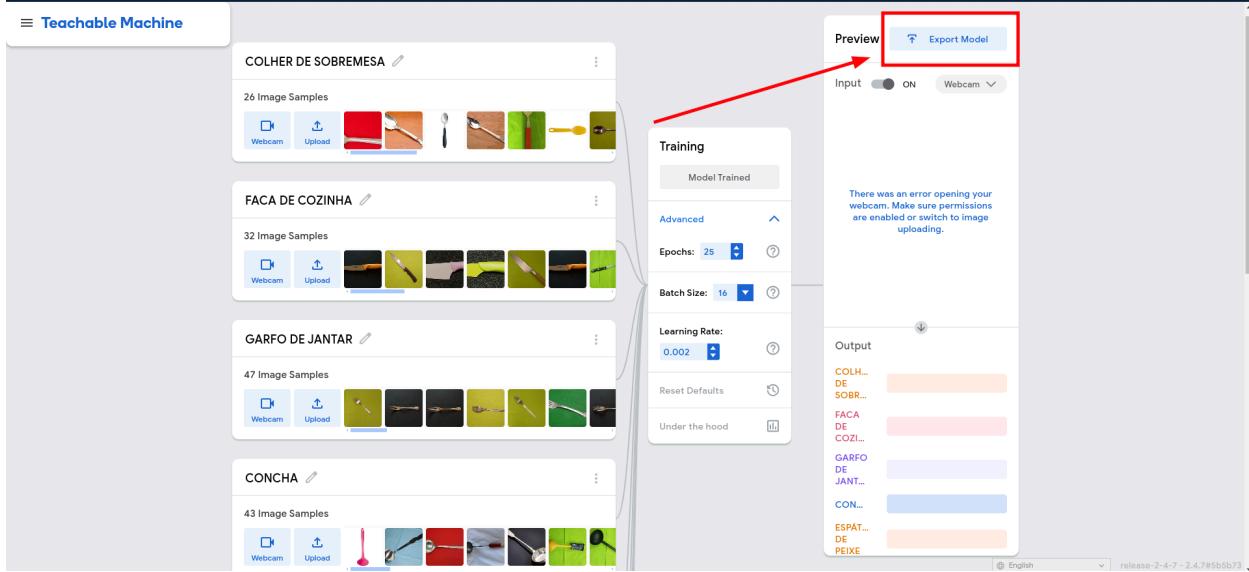
- c. O Script separa as imagens criando as pastas `data_train` e `data_test` e copiando arquivos da localização original dentro das pastas com nome de cada classe, além de imprimir os dados:

Category	Portuguese Category	English Category	Train	Validation	Test
COLHER DE SOBREMESA	COLHER DE SOBREMESA	DESSERT SPOON	26 files (78.79%)	0 files (0.00%)	7 files (21.21%)
KITCHEN KNIFE	FACA DE COZINHA	KITCHEN KNIFE	32 files (78.05%)	0 files (0.00%)	9 files (21.95%)
DINNER FORK	GARFO DE JANTAR	DINNER_FORK	43 files (79.06%)	0 files (0.00%)	12 files (20.94%)
LADLE	CONCHA	LADLE	43 files (79.03%)	0 files (0.00%)	11 files (20.37%)
FISH SLICE	ESPATULA DE PEIXE	FISH_SLICE	65 files (79.27%)	0 files (0.00%)	17 files (20.73%)
DINNER KNIFE	FACA DE JANTAR	DINNER_KNIFE	41 files (78.85%)	0 files (0.00%)	11 files (21.15%)
BREAD KNIFE	FACA DE PAO	BREAD_KNIFE	19 files (79.17%)	0 files (0.00%)	5 files (20.83%)

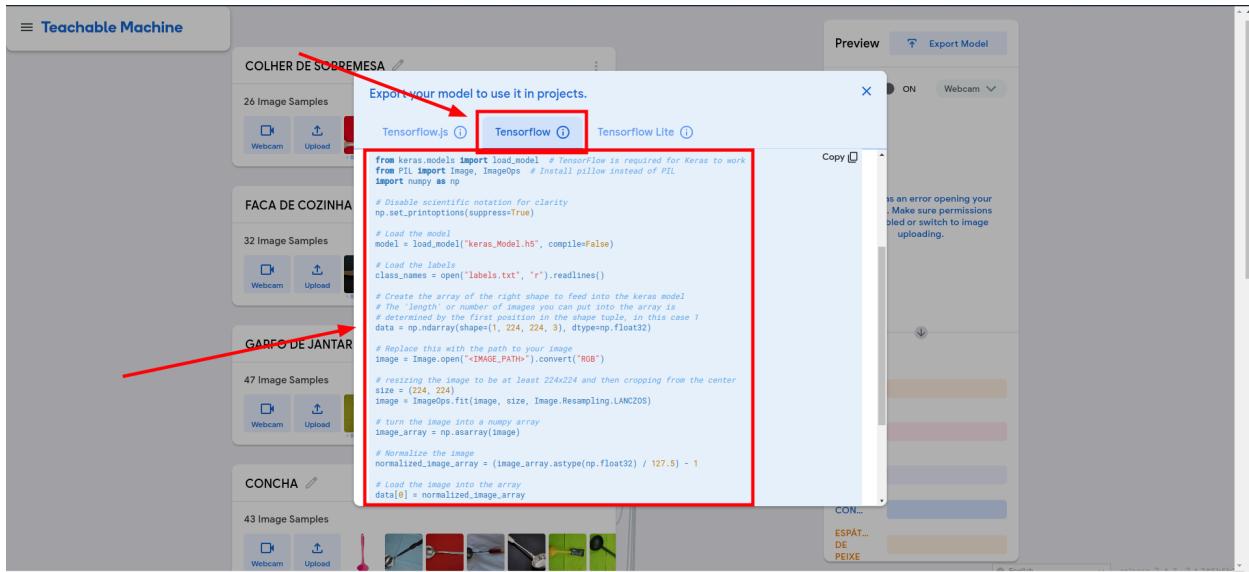
- d. Depois disso, subimos os arquivos das pastas em `data_train` para cada classe do Teachable Machine:

3. Preparação dos dados

- a. Investigamos como o Teachable Machine faz a preparação dos dados, a fim de fazer a mesma preparação no nosso código de avaliação. Para tal, treinamos primeiramente um modelo com os hiper parâmetros padrões (epochs 50, batch size 16 e learning rate 0.001) e clicamos em export model:



- b. Movemos para a aba Tensorflow e verificamos o código de avaliação dos modelos sugeridos pelo Teachable machine:



- c. Percebemos que é necessário reajustar o tamanho da imagem para 224 x 224, além de transformá-las em vetores numpy e normalizar os valores de cada canal (RGB) utilizando a expressão:

```
# Normalize the image
normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1
```

- d. Essa normalização serve para que os valores de cada pixel em cada um dos 3 canais RGB passem a ser entre -1 e 1, centrados em 0, ao invés de estar entre 0 e 255, que é o valor padrão dos pixels em imagens. Valores nessa faixa conferem muito mais estabilidade para o treinamento e inferência da rede neural.

4. Configurações do modelo

- a. Vamos baixar um modelo treinado pelo Teachable Machine e utilizar o summary do Keras para investigar a estrutura do modelo. Como o modelo é muito grande, vamos mostrar apenas as primeiras e as últimas camadas para que possamos fazer inferências a respeito.
- b. Quanto à visualização das primeiras camadas, podemos notar camadas como convolucionais, preenchimento, normalização de batch, ativação relu, e outros tipos de camadas. Esse tipo de camada é um padrão que pode ser considerado pré-transformers, são as arquiteturas que dominavam o Estado-da-Arte em visão computacional anteriormente à aplicação da arquitetura de Transformers (introduzida pelo icônico paper: “Attention is all you need”), que passou a englobar muitos dos modelos estado-da-arte (SOTA) na visão computacional, processamento de linguagem natural e algumas outras categorias de IA dos dias atuais. Vale ressaltar que esse tipo de arquitetura continua muito útil e usual mesmo na era pós-transformers, diversas aplicações práticas o utilizam por questão de eficiência.
- c. Sobre essas camadas, podemos ainda dizer que são responsáveis pela extração de características relevantes para a classificação de cada imagem. Elas são capazes de dizer se cada uma das imagens contém determinadas características ou não, o que é informação importante na classificação.
- d. Analisando as últimas camadas, verificamos duas camadas Dense. As camadas Dense combinam as informações que a rede conseguiu obter das camadas anteriores, ponderando os pesos de cada informação obtida a fim de obter o resultado final baseado nesses pesos. Portanto, as últimas camadas da rede são responsáveis não mais por encontrar se as features estão presentes ou ausentes em cada imagem, mas por combinar essas características para que a saída classifique corretamente a imagem com base nessas características.
- e. A última camada tem saída de 8 valores, que são as probabilidades calculadas pela rede para cada categoria. Portanto, o output da rede para cada imagem é um vetor de probabilidades. Consideramos que a categoria com maior probabilidade é a classificação da imagem pela rede. Podemos utilizar a função argmax do numpy para obter a classificação.
- f. Pesquisas na internet (sem contudo que houvesse comprovação contundente) indicaram que o modelo pré-treinado do Teachable Machine seria o MobileNet

V2. É um modelo otimizado para uso em dispositivos móveis e navegadores (ou seja, em condições de recursos computacionais limitados). Esse modelo tem como características principais menor tamanho (menos pesos treináveis) e complexidade em relação a outros modelos famosos como VGG16, ResNet50 e Inception v3. Embora apresente menor precisão em datasets complexos, mantém precisão considerável em datasets mais simples, além de rapidez para treinamento e inferência.

- g. Assim sendo, o treinamento do Teachable Machine seria, em verdade, em fine-tuning, que é um ajuste fino dos pesos da rede neural pré-treinada (ou de uma parte dela). Isso significa que partiríamos de um modelo que já tenha sido anteriormente treinado e já com capacidade inicial de reconhecer padrões nas imagens, a partir dele faríamos um pequeno ajuste em seus pesos para que ele aprendesse a focar em padrões mais específicos presentes no dataset de treino. O Fine-tuning costuma envolver treinamento das últimas camadas densas, para que o nosso modelo aprenda a combinar features de acordo com aquelas presentes no dataset de treinamento.

```
[16]: 1 model.summary(expand_nested=True, show_trainable=True)
      ✓ 0.1s
...
Model: "sequential_48"
+-----+-----+-----+-----+
| Layer (type)        | Output Shape | Param # | Trainable |
+=====+=====+=====+=====+
| sequential_45 (Sequential)| (None, 1280) | 410208 | Y         |
+-----+-----+-----+-----+
|| model12 (Functional)    | (None, 7, 7, 1280) | 410208 | Y         |
||-----+-----+-----+-----+
|| input_1 (InputLayer)     | [(None, 224, 224, 3)] | 0       | Y         |
|| Conv1_pad (ZeroPadding2D) | (None, 225, 225, 3) | 0       | Y         |
|| Conv1 (Conv2D)          | (None, 112, 112, 16) | 432    | Y         |
|| bn_Conv1 (BatchNormalizatio | (None, 112, 112, 16) | 64     | Y         |
|| n)                      |
|| Conv1_relu (ReLU)        | (None, 112, 112, 16) | 0       | Y         |
|| expanded_conv_depthwise (De | (None, 112, 112, 16) | 144    | Y         |
|| pthwiseConv2D)           |
|| expanded_conv_depthwise_BN | (None, 112, 112, 16) | 64     | Y         |
|| (BatchNormalization)      |
|| expanded_conv_depthwise_rel | (None, 112, 112, 16) | 0       | Y         |
|| u (ReLU)                 |
|| expanded_conv_project (Conv | (None, 112, 112, 8) | 128    | Y         |
|| 2D)                     |
|| expanded_conv_project_BN | (None, 112, 112, 8) | 32     | Y         |
|| (BatchNormalization)      |
|| block_1_expand (Conv2D)   | (None, 112, 112, 48) | 384    | Y         |
|| block_1_expand_BN (BatchNor | (None, 112, 112, 48) | 192    | Y         |
|| malization)              |
|| block_1_expand_relu (ReLU) | (None, 112, 112, 48) | 0       | Y         |
||-----+-----+-----+-----+
```

```

|| Conv_1_bn (BatchNormalizati (None, 7, 7, 1280) 5120      Y
|| on)

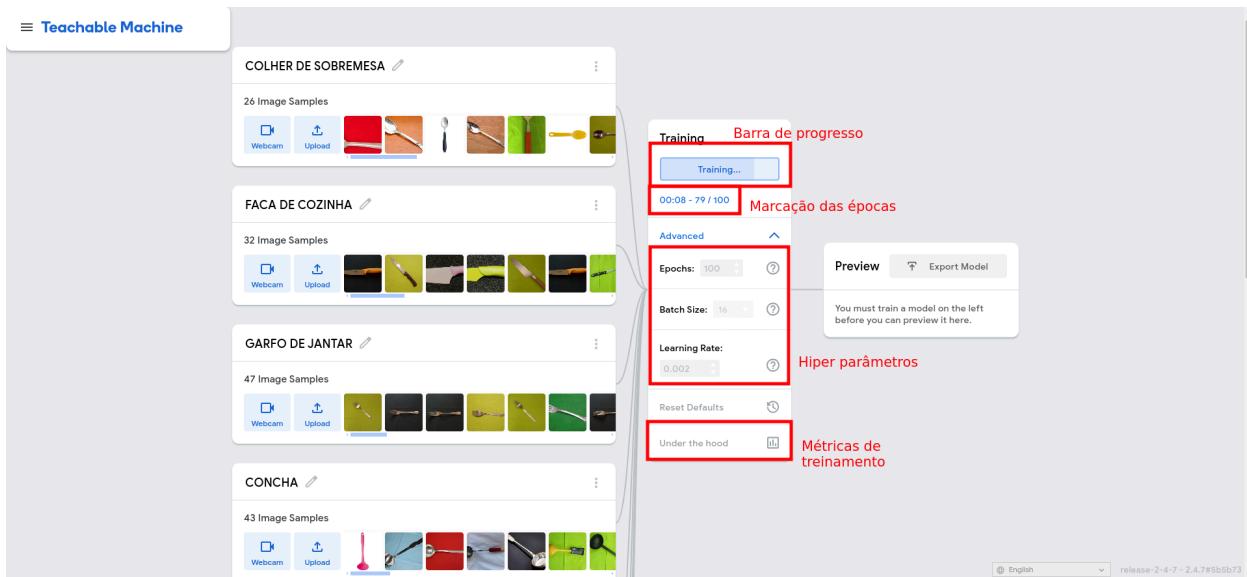
|| out_relu (ReLU)      (None, 7, 7, 1280)      0      Y
-----|-----|-----|-----|-----|-----|-----|-----|
| global_average_pooling2d_Gl (None, 1280)      0      Y
| obalAveragePooling2D12 (Glo
| balAveragePooling2D)
-----|-----|-----|-----|-----|-----|-----|-----|
sequential_47 (Sequential) (None, 8)           128900    Y
-----|-----|-----|-----|-----|-----|-----|-----|
| dense_Dense23 (Dense)   (None, 100)          128100    Y
-----|-----|-----|-----|-----|-----|-----|-----|
| dense_Dense24 (Dense)   (None, 8)            800      Y
-----|-----|-----|-----|-----|-----|-----|-----|
=====|=====|=====|=====|=====|=====|=====|=====|
Total params: 539,108
Trainable params: 525,028
Non-trainable params: 14,080

```

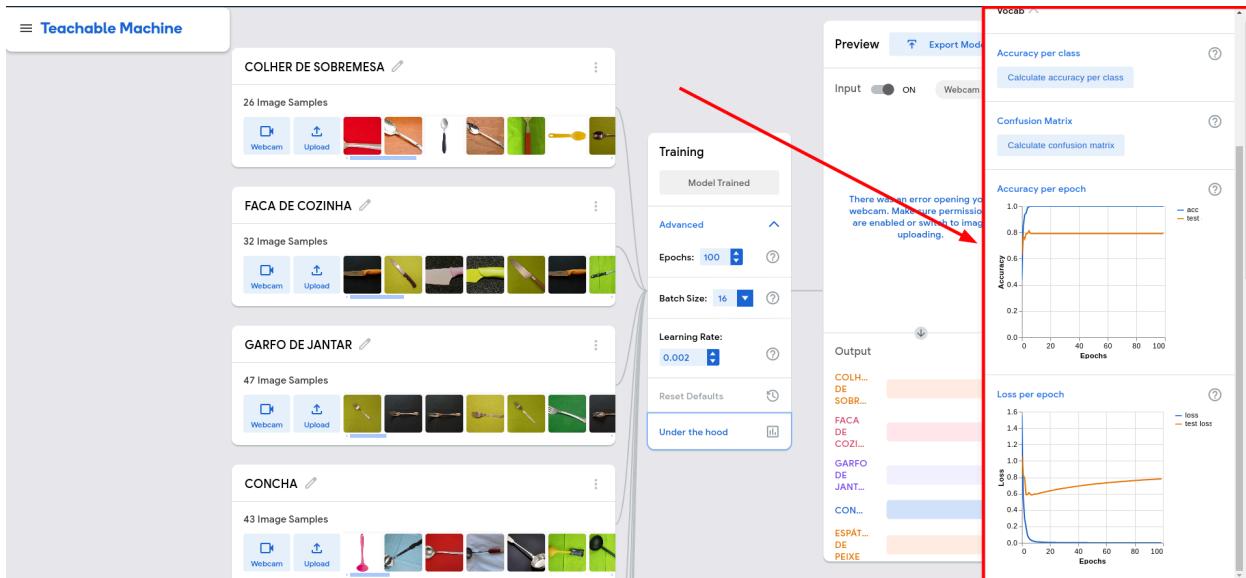
5. Treinamento

- O treinamento do modelo de base do Teachable Machine é feito utilizando-se o próprio navegador, selecionando-se manualmente os arquivos de cada classe (a partir do próprio hard drive ou do Google Drive). Ademais, configuram-se 3 hiperparâmetros: taxa de aprendizado, quantidade de épocas de treinamento e tamanho do batch. Os hiperparâmetros não são parâmetros treináveis, mas controlam as circunstâncias de treinamento do modelo. A escolha correta deles é fundamental para a performance do modelo para inferência futura.
- Para disparar o treinamento, basta-se um clique. Em poucos segundos, o modelo estará treinado. Durante o treinamento, uma barra de progresso mostra a quantidade de épocas já realizadas e o progresso gráfico na barra indica visualmente a proporção de treinamento realizado, dando uma ideia visual ao usuário sobre o progresso do treinamento. Há um marcador do tempo de treinamento.
- A escolha dos hiperparâmetros principais é feita logo abaixo.
- Há o botão “under the hood”, que se traduz por “sob o capô”. É uma seção que mostra detalhes sobre o último treinamento realizado.
- A seção “under the hood” abre uma visão lateral que mostra gráficos de acurácia e loss em cada época. Um pouco sobre cada uma das métricas:

- i. A acurácia informa qual a razão de classes o modelo foi capaz de classificar corretamente nos dados de validação, que são separados automaticamente dos dados de treino pelo próprio Teachable machine
- ii. O loss é a variável que determina o quanto distante a resposta da rede está da resposta esperada. Nosso objetivo é minimizar o loss, minimizando, portanto, a distância entre a resposta esperada e a resposta obtida pelo modelo. O cálculo de loss pode ser propagado pelas camadas da rede neural no sentido do final para o início (algoritmo conhecido como backpropagation) a fim de se atualizar os pesos da rede neural (esse processo se confunde com o próprio treinamento da rede neural, o conhecimento sobre o dataset de treinamento que fica armazenado em seus pesos).



- f. A seção “under the hood” mostra valores de acurácia e de loss tanto para dados de treinamento quanto validação. Nessa seção também é possível solicitar o cálculo da acurácia por classe e da matriz de confusão dos resultados.



6. Avaliação

- Diversos modelos serão treinados, com diferentes hiperparâmetros. A fim de se avaliar cada modelo, baseamos-nos no código sugerido pelo Teachable Machine, e produzimos um script de avaliação que carrega as imagens de teste anteriormente separadas, carrega o modelo e realiza inferência nas imagens de teste. O modelo compara os dados esperados com os dados obtidos e calcula acurácia.
- Adotamos um procedimento padrão: treinamos o modelo utilizando o próprio navegador, em seguida baixamos o modelo (versão Keras h5) com o nome padronizado `converted_keras_<n_epocas>_<learning_rate>_<batch_size>.h5`. Para a learning rate pegamos apenas as casas decimais. Por exemplo, as configurações padrão (learning rate 0.001, 50 épocas e tamanho do batch 16 gerarão o arquivo: `converted_keras_50_001_16.h5`. Dessa forma, ao percorrer os arquivos com extensão h5 na pasta de modelos, conseguimos saber quais foram os hiper parâmetros utilizados.
- Nosso código de avaliação então calcula e imprime informações de avaliação para cada modelo.

```

● evaluating.ipynb - Kitchen-Utensils - Visual Studio Code
File Edit Selection View Go Run Terminal Help
OPEN EDITORS 1 unsaved
evaluating.ipynb
KITCHEN-UTENSILS
model
converted_keras_50_0001_16.h5
converted_keras_50_0001_16.zip
converted_keras_50_001_32.h5
converted_keras_50_001_32.zip
converted_keras_50_001_64.h5
converted_keras_50_001_64.zip
converted_keras_50_002_16.h5
converted_keras_50_002_16.zip
converted_keras_50_005_16.h5
converted_keras_50_005_16.zip
converted_keras_50_005_16.zip
converted_keras_75_0001_16.h5
converted_keras_75_0001_16.zip
converted_keras_75_001_16.h5
converted_keras_75_001_16.zip
converted_keras_75_001_16.zip
converted_keras_75_001_16.zip
converted_keras_75_001_16.zip
converted_keras_75_001_16.zip
converted_keras_75_002_16.h5
converted_keras_75_002_16.zip
converted_keras_75_005_16.h5
converted_keras_75_005_16.zip
converted_keras_75_005_16.zip
converted_keras_100_0001_16.h5
converted_keras_100_0001_16.zip
converted_keras_100_0005_16.h5
converted_keras_100_0005_16.zip
keras_model_50_001_16.h5
keras_model_50_001_16.zip
labels.txt
src
dataset_splitting.ipynb
evaluating.ipynb
running.ipynb
.ignores
README.md
OUTLINE
TIMELINE
src > evaluating.ipynb > model.summary(expand_nested=True, show_trainable=True)
+ Code + Markdown | ▶ Run All ⚡ Restart Clear All Outputs | Variables Outline
68 ) ✓ 30.8s
69 print("Final acc: (best accuracy)")*
... Model: ./model/converted_keras_50_001_20.h5
Epochs: 50, Learning Rate: 0.001, Batch Size: 32
3/3 [=====] - 1s 66ms/step
Accuracy: 0.8846153846153846
Precision: 0.901098135198135
Recall: 0.8846153846153846
F1 Score: 0.8853115695220959
...
Model: ./model/converted_keras_75_002_16.h5
Epochs: 75, Learning Rate: 0.002, Batch Size: 16
3/3 [=====] - 1s 59ms/step
Accuracy: 0.8717948717948718
Precision: 0.880136839798378
Recall: 0.8717948717948718
F1 Score: 0.8727486856680145
...
Model: ./model/converted_keras_75_001_16.h5
Epochs: 75, Learning Rate: 0.001, Batch Size: 16
3/3 [=====] - 1s 60ms/step
Accuracy: 0.8974358974358975
Precision: 0.9014041514041514
Recall: 0.8974358974358975
F1 Score: 0.8979933110367693
...
Model: ./model/converted_keras_25_002_16.h5
Epochs: 25, Learning Rate: 0.002, Batch Size: 16
3/3 [=====] - 1s 60ms/step
Accuracy: 0.8589743589743589
Precision: 0.8697265097265697
Recall: 0.8589743589743589
F1 Score: 0.8553927553927554
...
Model: ./model/converted_keras_50_005_16.h5
Epochs: 50, Learning Rate: 0.005, Batch Size: 16
3/3 [=====] - 1s 61ms/step
Accuracy: 0.8589743589743589
...

```

d. Ao final, o script imprime a configuração com melhor acurácia:

```

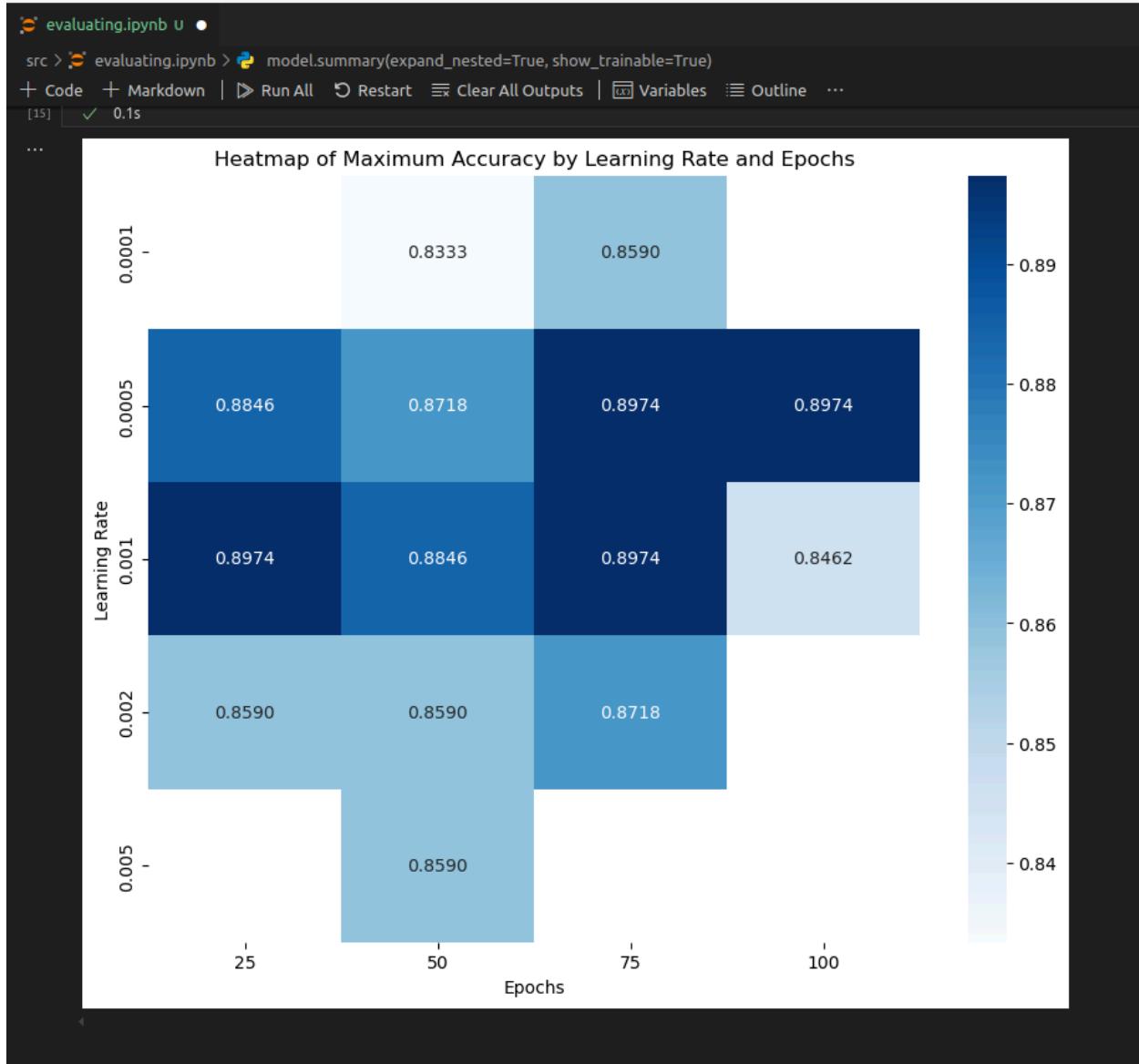
● evaluating.ipynb u ●
src > evaluating.ipynb > model.summary(expand_nested=True, show_trainable=True)
+ Code + Markdown | ▶ Run All ⚡ Restart Clear All Outputs | Variables Outline
3/3 [=====] - 1s 60ms/step
Accuracy: 0.8333333333333334
Precision: 0.8266705966098679
Recall: 0.8333333333333334
F1 Score: 0.8255805009293934

Model: ./model/converted_keras_25_0005_16.h5
Epochs: 25, Learning Rate: 0.0005, Batch Size: 16
3/3 [=====] - 1s 58ms/step
Accuracy: 0.8846153846153846
Precision: 0.8897435897435898
Recall: 0.8846153846153846
F1 Score: 0.8852524287306895

Best model: ./model/converted_keras_75_001_16.h5
Epochs: 75, Learning Rate: 0.001, Batch Size: 16
Final acc: 0.8974358974358975

```

- e. Visando investigar a relação entre os hiperparâmetros taxa de aprendizado e número de épocas, produzimos um mapa de calor (utilizando a biblioteca Seaborn) em que os eixos são as épocas e a taxa de aprendizado, e a cor em escala de azul indica o valor da melhor acurácia obtida para aquela combinação. Alguns valores estão em branco porque não foi realizado o treinamento com aqueles hiper parâmetros.



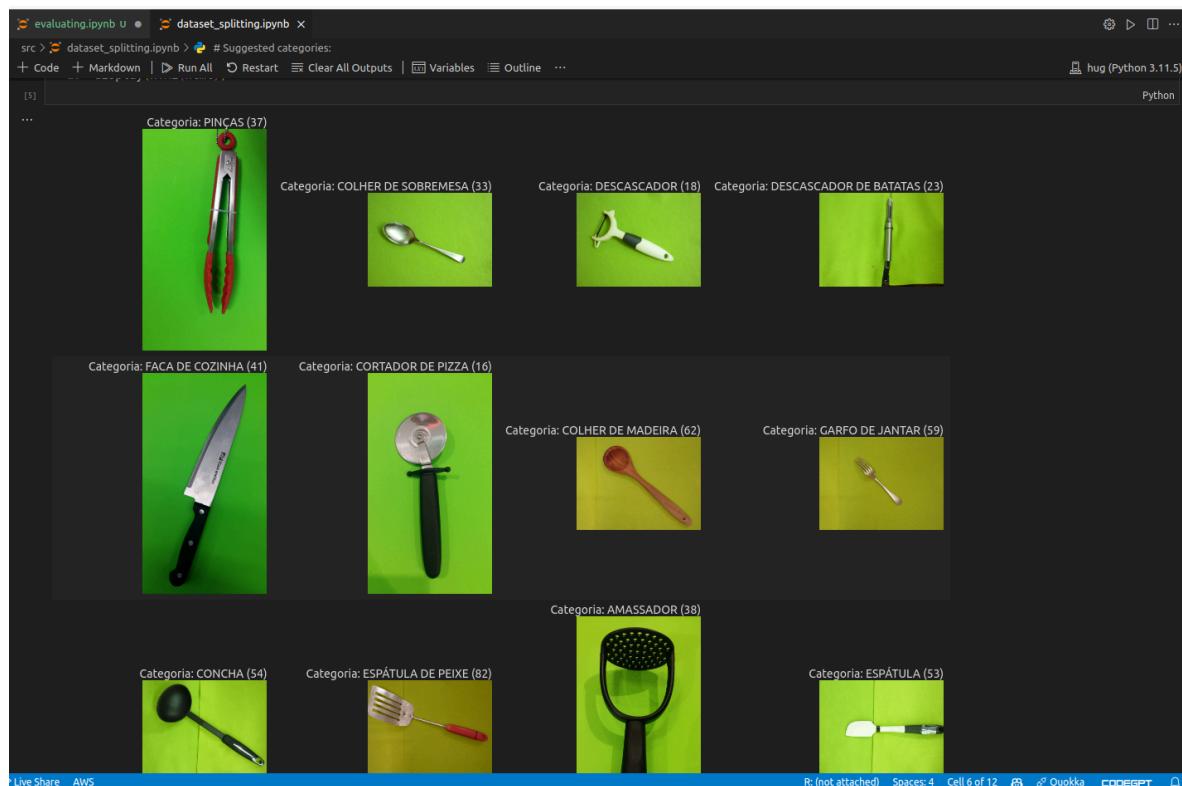
- f. O mapa de calor mostra alguns fatos:
- A região principal encontra-se a uma learning rate de aproximadamente 0.001 a 0.005, mais próxima de 75 épocas de treinamento.
 - Percebe-se uma curva com performance crescente entre mais épocas e menos learning rate. Isso ocorre porque um modelo com learning rate

menor poderá precisar de mais épocas de treinamento para atingir o ponto ótimo. Isso nos dá um indicativo de como variar esses dois parâmetros de maneira ótima na busca por melhores resultados: menores taxas de aprendizado tendem a dar melhores resultados com mais épocas de treino.

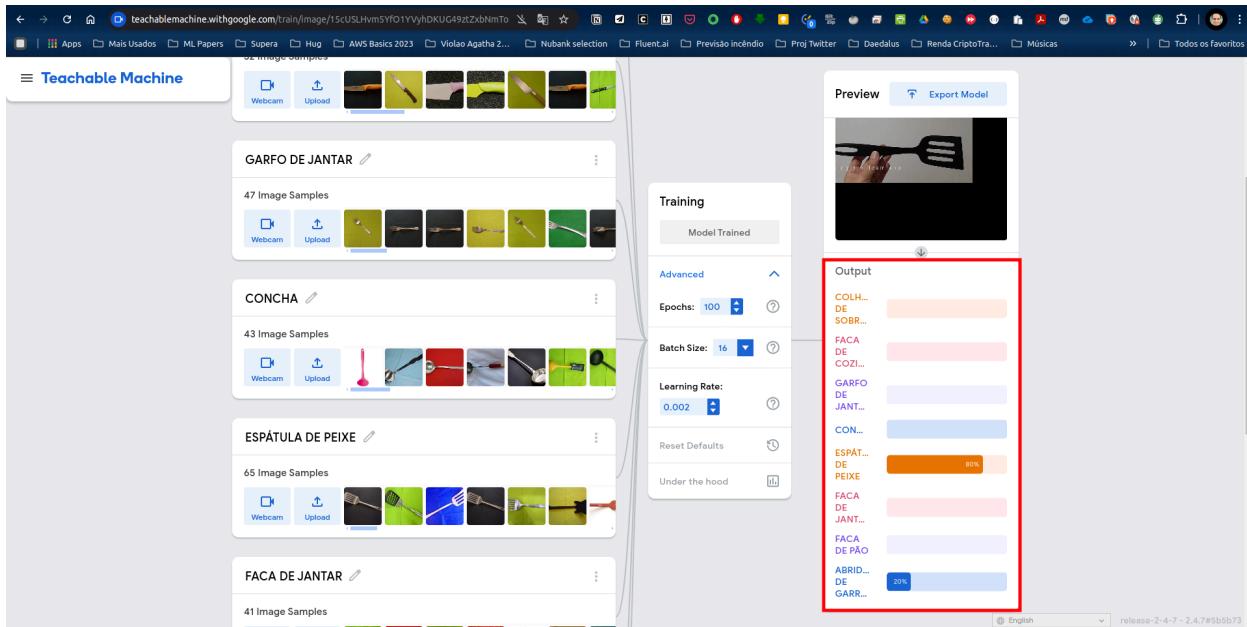
- iii. A melhor acurácia obtida foi de 0.8974 (89,74%) nos dados de teste, essa acurácia pode ser obtida ao menos com 4 configurações de taxa de aprendizado e número de épocas. Essa configuração ótima está representada pela cor azul mais escura do mapa de calor.

7. Pós avaliação

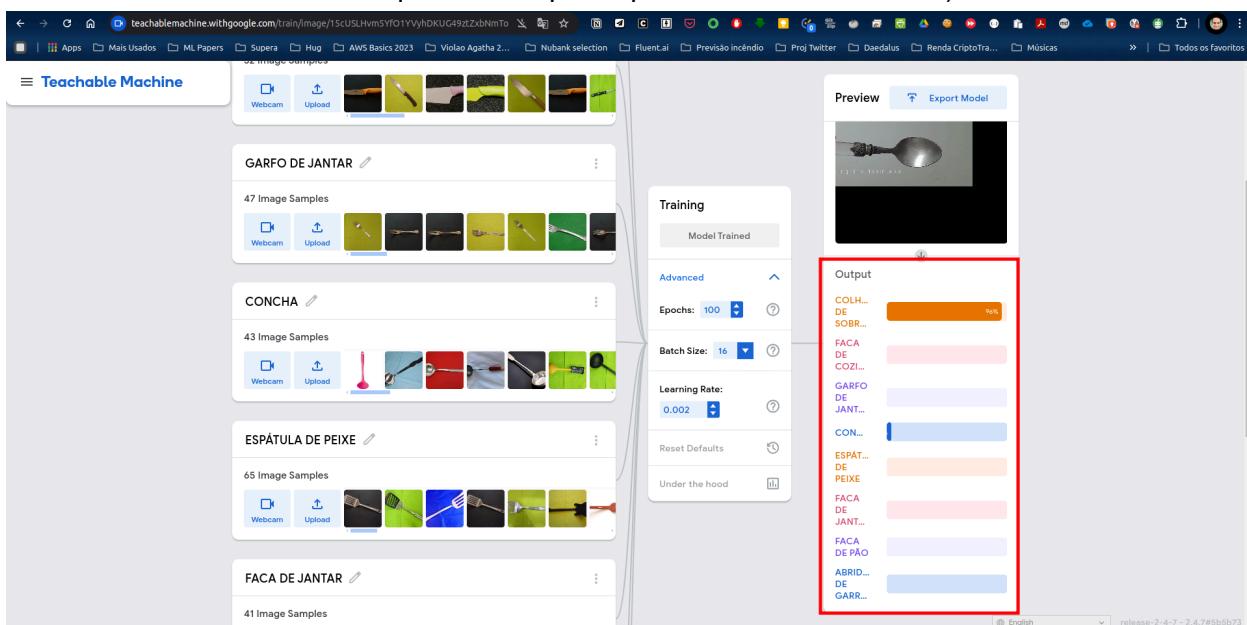
- a. Uma vez que avaliamos o modelo, vamos agora fazer uma etapa suplementar: a avaliação em ambiente real. Isso significa que vamos utilizar a webcam para capturar as imagens. Significa também que haverá diferenças significativas no controle do ambiente de geração das imagens em relação ao dataset. Poderemos avaliar de forma experimental se o modelo continua capaz de gerar boas classificações numa aplicação executando no mundo real.
- b. Vejamos exemplos de imagens de treino (essas imagens estão no script de inspeção e separação das imagens). Repare como os fundos das imagens são bem definidos (geralmente uma única cor), e como a iluminação é aproximadamente constante, sem sombras etc:

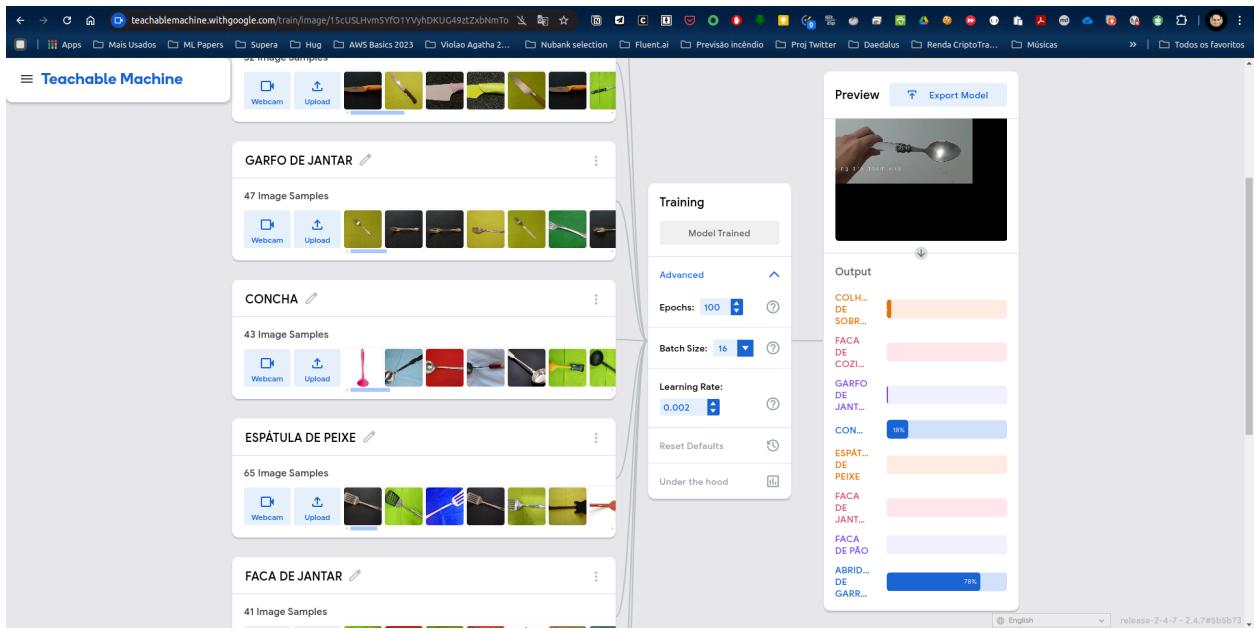


- c. Será que nosso modelo continua capaz de avaliar objetos em ambiente real, com imagem real, iluminação real, possibilidade de fundos diferentes etc? Testamos essa possibilidade. O Teachable Machine permite facilmente utilizar a webcam para apontar objetos e obter a classificação em tempo real:
- d. Vamos testar cada um dos objetos contra a parede branca e ver os resultados:
 - i. Espátula de peixe (previsão nítida com 80% de probabilidade prevista):

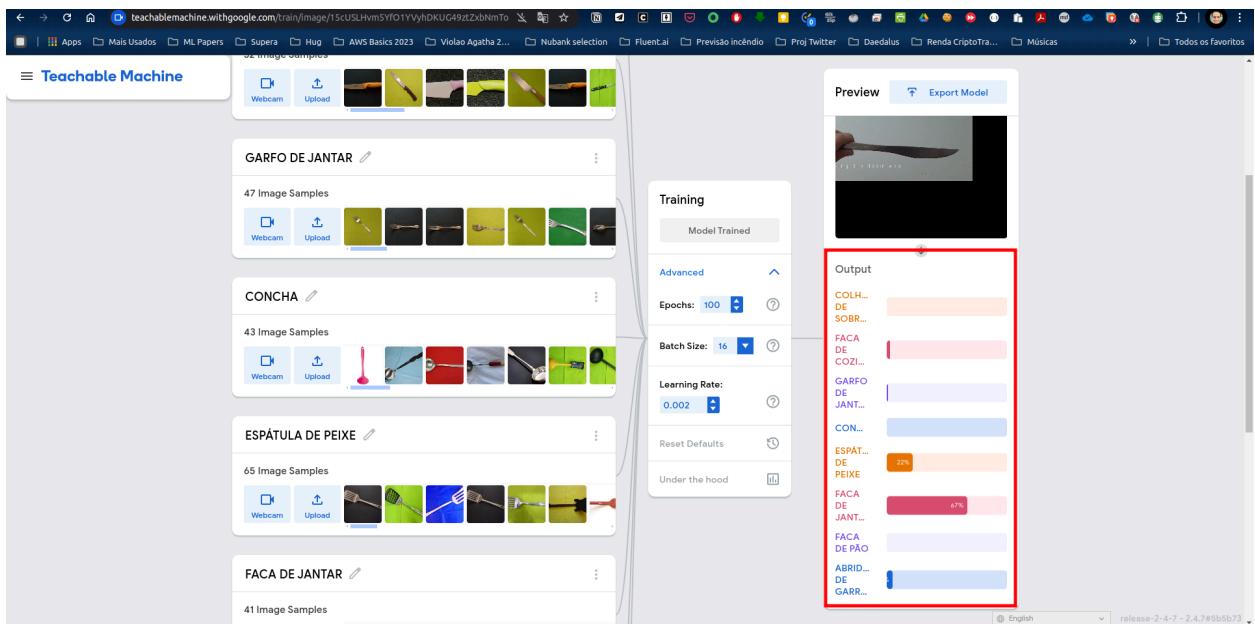


- ii. Colher de sobremesa (confundiu-se com conha e com abridor de garrafa). Presença do cabo e da mão na imagem atrapalharam, a previsão fica melhor quando se aponta para a concha da colher).

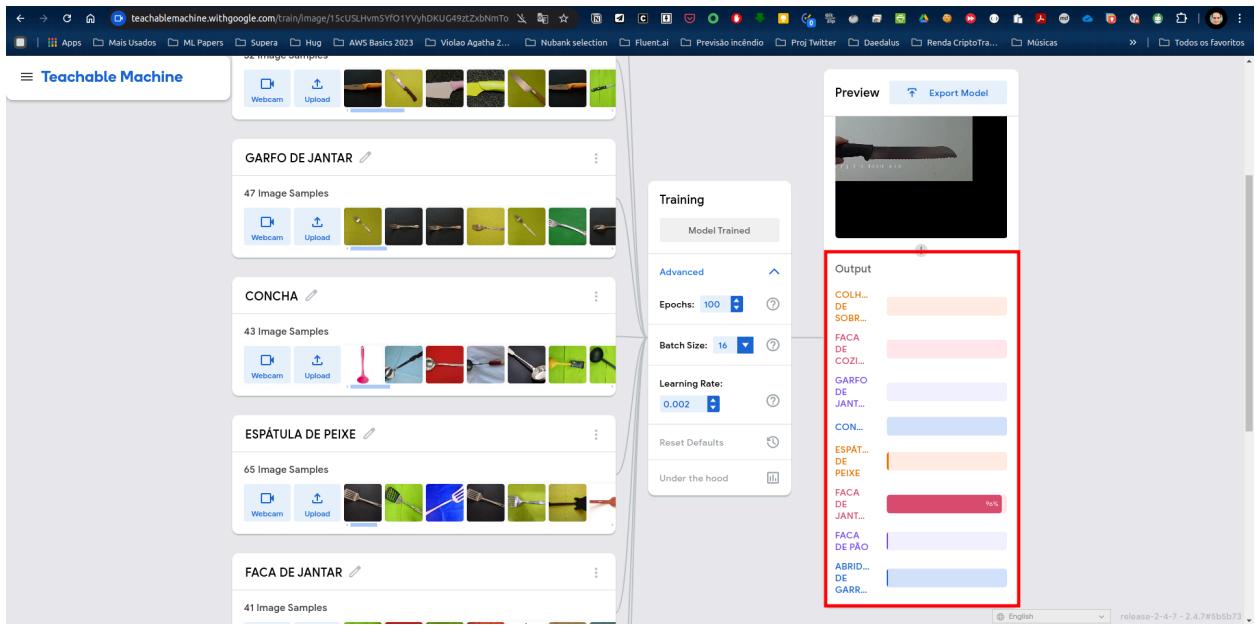




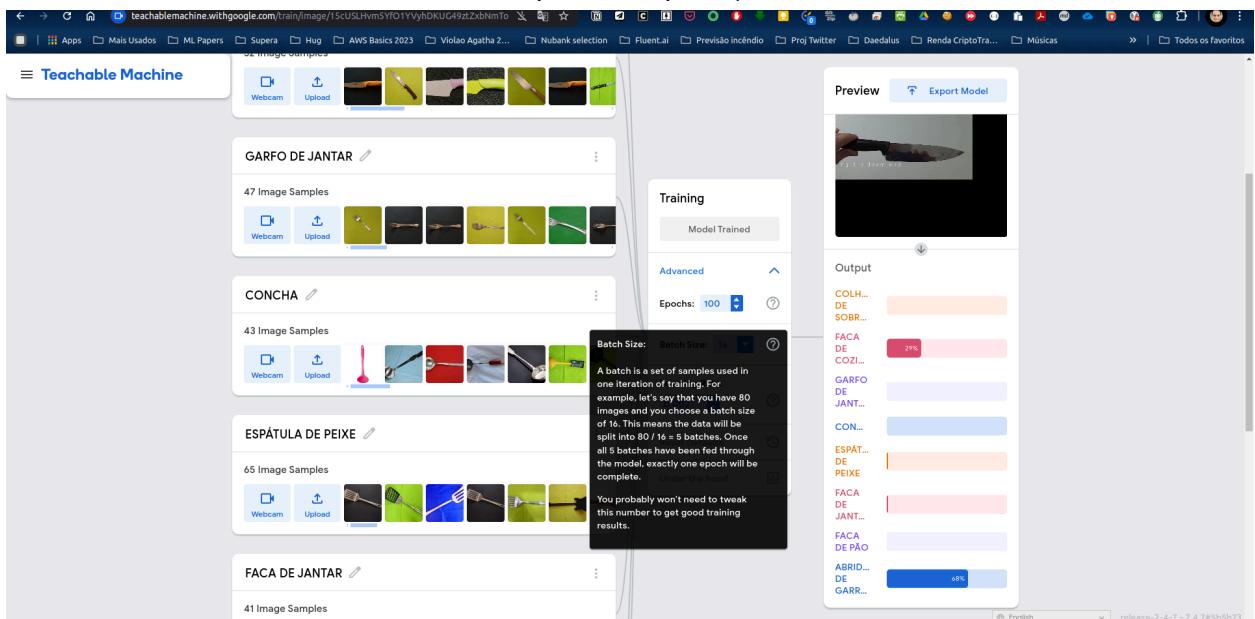
- iii. Faca de jantar (boa previsão de 67%, mas a previsão fica bem instável enquanto se aponta a câmera). Fato interessante: não houve confusão com faca de cozinha nem com faca de pão!



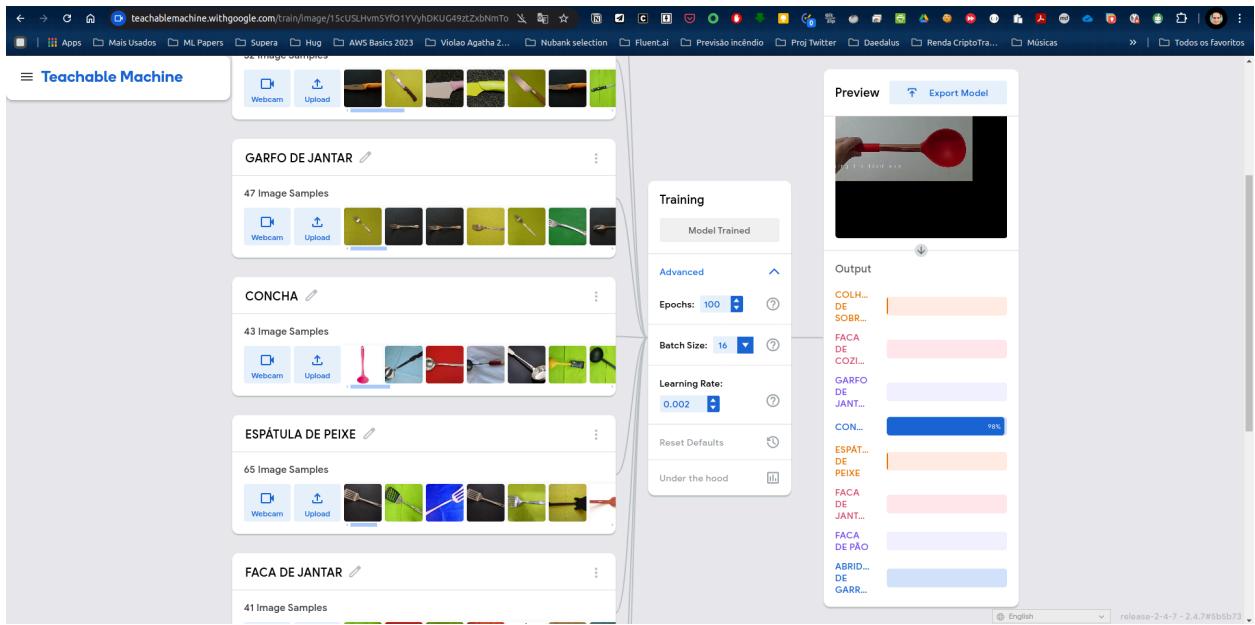
- iv. Faca de pão (falha nítida do modelo: classificação como faca de cozinha. Explica-se pela menor quantidade de facas de pão nos dados de treino: 32 de cozinha contra 19 de pão).



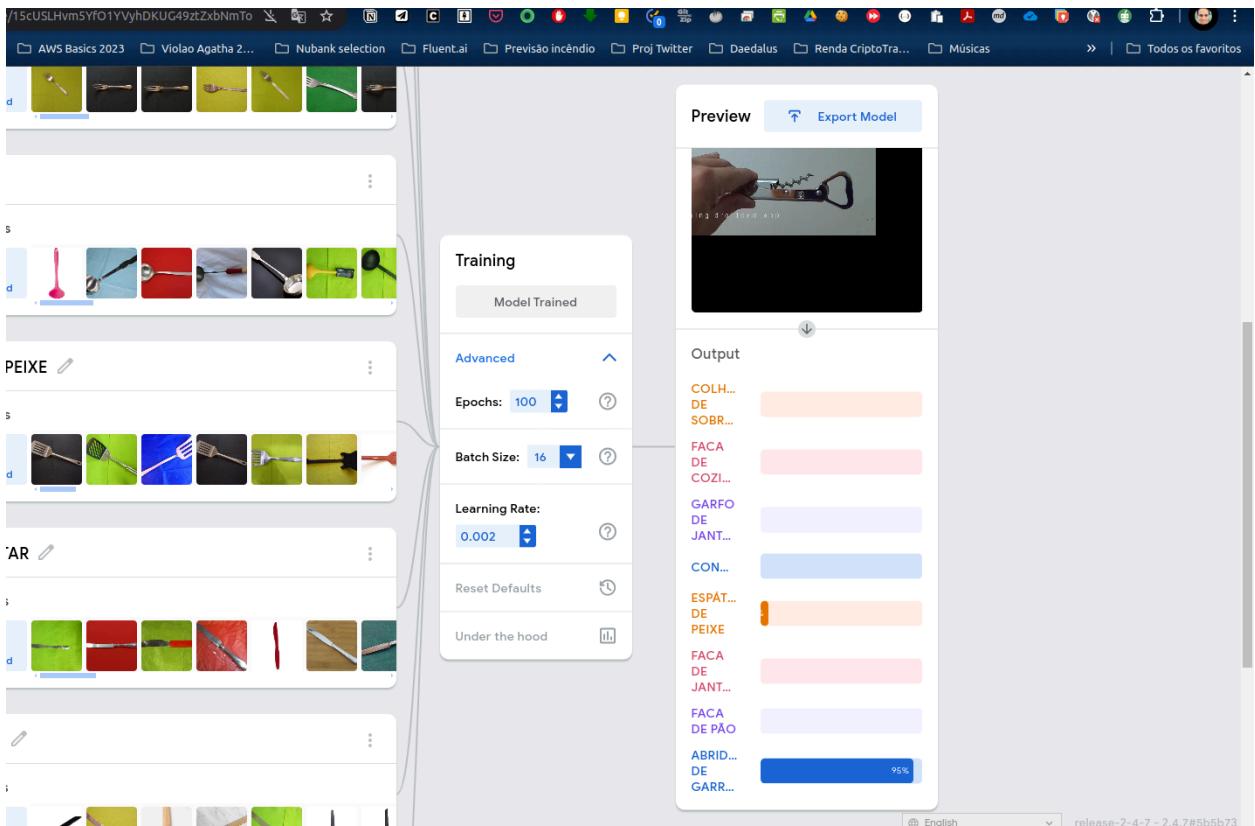
- v. Faca de cozinha (erro do modelo: nota-se bastante confusão com a faca de cozinha durante o apontamento da câmera. Às vezes, também foi confundida com espátula de peixe).



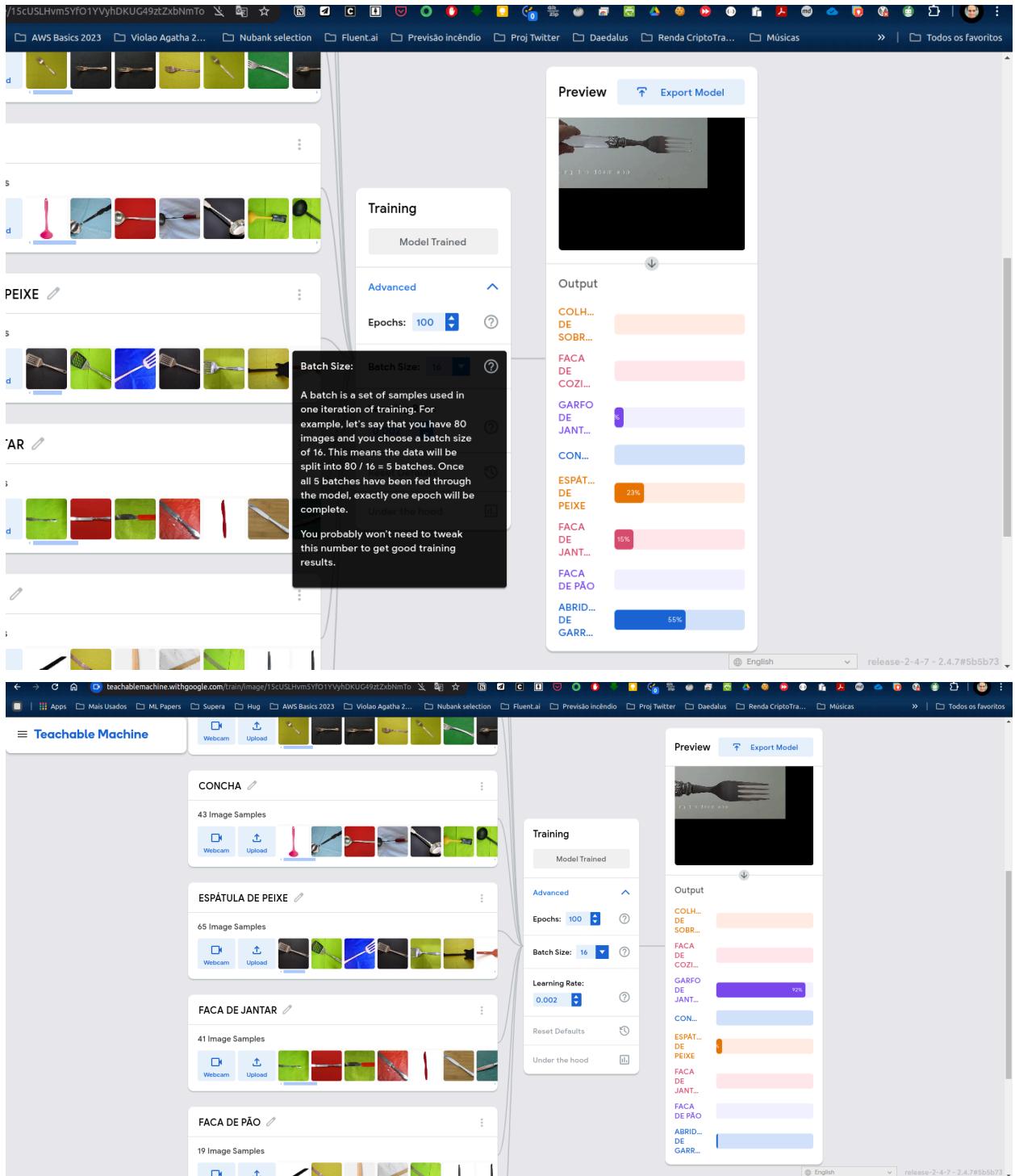
- vi. Concha (Foi a previsão mais correta do modelo: 98% de probabilidade)



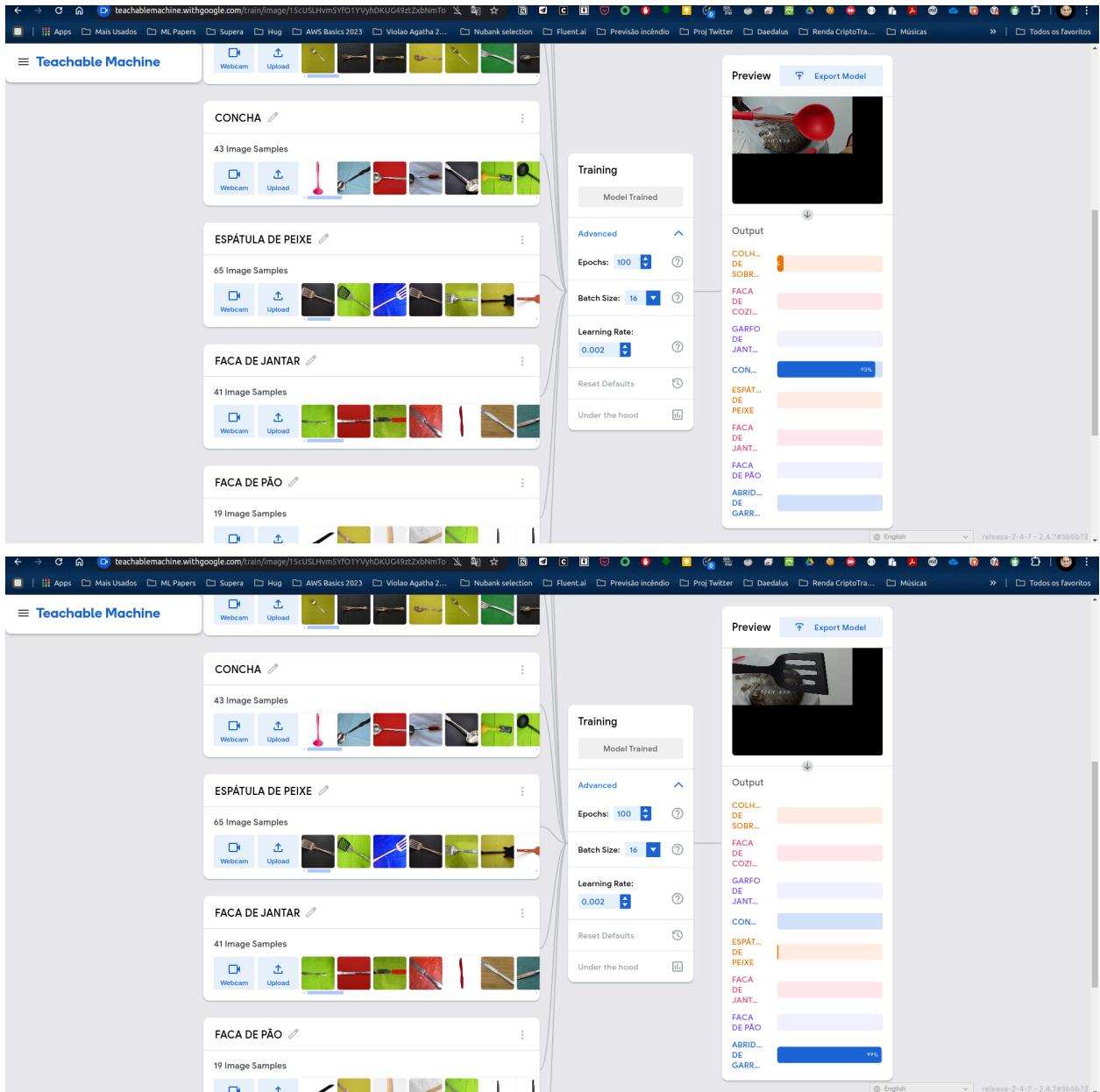
vii. Abridor de garrafas (bastante assertivo: 98% de probabilidade)



viii. Garfo de jantar (previsão bem errada quando o cabo é mostrado, assertividade maior quando focamos apenas a ponta, mesmo assim ainda bem estável).



- Vamos testar fundo não neutro (previsão da concha continua boa, a espátula perde bastante com fundo neutro):



- f. Podemos perceber que aspectos relevantes nas medições em tempo real, como ângulos diferenciados, iluminação, reflexos nas imagens, características do fundo etc são relevantes para a performance dos modelos em momento de inferência. Visto que os datasets foram obtidos em condições essencialmente mais controladas, houve certa perda de performance quando as imagens foram obtidas em ambiente real a partir de imagens de câmera.

Conclusões do Experimento de Classificação de Utensílios de Cozinha

1. Eficácia do modelo: O modelo alcançou uma acurácia máxima de 89,74% nos dados de teste, demonstrando boa capacidade de classificação para as 8 classes de utensílios de cozinha selecionadas.
2. Hiperparâmetros ótimos: Os melhores resultados foram obtidos com taxas de aprendizado entre 0,001 e 0,005, e cerca de 75 épocas de treinamento. Isso sugere um equilíbrio entre aprendizado gradual e tempo suficiente para convergência.
3. Relação entre taxa de aprendizado e épocas: observou-se uma relação inversa entre taxa de aprendizado e número de épocas necessárias para atingir bons resultados. Taxas de aprendizado menores tendem a requerer mais épocas de treinamento.
4. Generalização para ambiente real: Houve uma perda significativa de performance quando o modelo foi testado em condições reais usando uma webcam. Isso destaca a importância da diversidade nos dados de treinamento para melhorar a robustez do modelo.
5. Influência do ambiente: Fatores como ângulo, iluminação, reflexos e características do fundo afetaram significativamente a performance do modelo em tempo real, ressaltando a necessidade de treinar com dados mais variados e representativos de condições reais.
6. Confusão entre classes similares: O modelo apresentou dificuldades em distinguir entre tipos similares de utensílios, como diferentes tipos de facas, indicando a necessidade de mais dados ou features mais discriminativas para essas classes.
7. Impacto do desequilíbrio de classes: A menor quantidade de amostras para algumas classes (como facas de pão) resultou em classificações erradas mais frequentes, destacando a importância de um conjunto de dados balanceado.
8. Eficácia do fine-tuning: O uso de um modelo pré-treinado (possivelmente MobileNet V2) com fine-tuning mostrou-se eficaz para esta tarefa, permitindo bons resultados com recursos computacionais limitados.
9. Importância da validação em ambiente real: O experimento demonstrou claramente a diferença entre performance em dados de teste controlados e em aplicações do mundo real, enfatizando a necessidade de validação em condições reais de uso.
10. Potencial para melhorias: Os resultados sugerem que há espaço para melhorias, possivelmente através de aumento de dados, uso de técnicas de regularização, ou exploração de arquiteturas de modelo alternativas para lidar melhor com variações em condições reais.