INTELIGÊNCIA COMPUTACIONAL

Projeto Fase - 3

3ª Fase

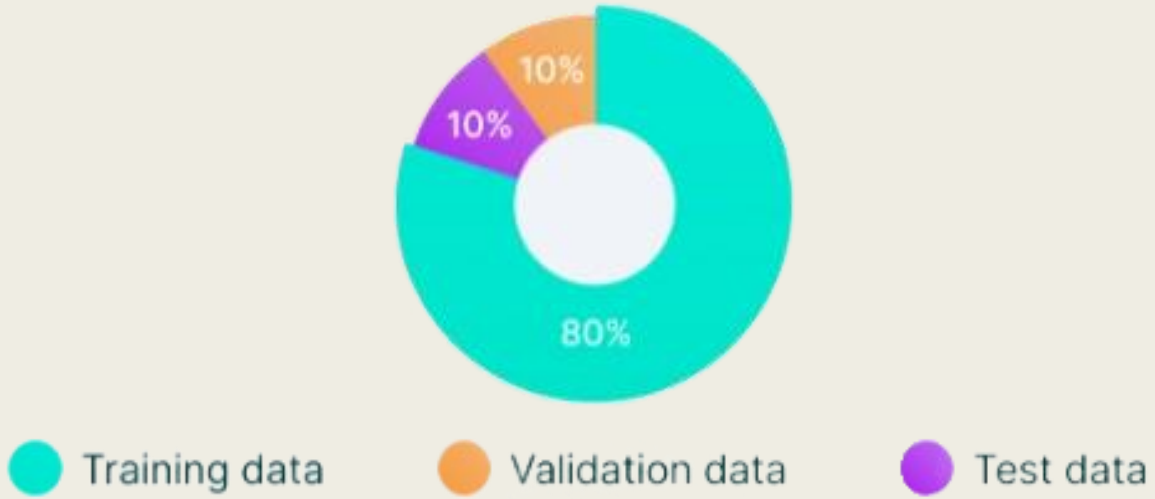
- Parte 1 - CNN;
- Parte 2 - Grid Search;
- Parte 3 - CNN + Bee.

Parte 1 - CNN

Construção da Rede Neuronal

```
# mobilenetv2 é um modelo (deep learning) do tipo CNN que é pré-treinado, ou seja,
# já foi treinada para classificar imagens, sendo esta capaz de classificar
# 1000 categorias das mesmas
mobilenetv2_layer = mobilenetv2.MobileNetV2(include_top = False, input_shape = (IMAGE_WIDTH, IMAGE_HEIGHT,IMAGE_CHA
                    weights = 'imagenet')

# O modelo mobilenetv2 já está pré-treinado, logo, não queremos que estas camadas que ele tras
# sejam treinadas novamente pelo tensorflow
mobilenetv2_layer.trainable = False

model = Sequential()

model.add(keras.Input(shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS)))

# Cria uma camada para aplicar o preprocessamento na imagem, para ter as caracteristicas pretendidas
def mobilenetv2_preprocessing(img):
  return mobilenetv2.preprocess_input(img)

model.add(Lambda(mobilenetv2_preprocessing))

#model.add(tf.keras.layers.BatchNormalization()) | Demora mt tempo a treinar com esta camada +/- 15 por epoch

# Camadas pre treinadas
model.add(mobilenetv2_layer)


model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(Flatten(name="featuresCamadaFlatten"))
model.add(tf.keras.layers.Dropout(0.3))
model.add(Dense(len(categories), activation='softmax'))


model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['categorical_accuracy'])
model.summary()


feature_extractor = keras.Model(
    inputs=model.inputs,
    outputs=model.get_layer(name="featuresCamadaFlatten").output,
)

x = tf.ones((1, 224, 224, 3))
features = feature_extractor(x)
print("Número de Features da Camada featuresCamadaFlatten:")
print(features)
```

Divisão em Treino, Validação e Teste




```
Num Imagens de Treino =  12412   Num Imagens de Validação =  1551   Num Imagens de Teste =  1552
```

Data Augmentation - Teste

Softmax

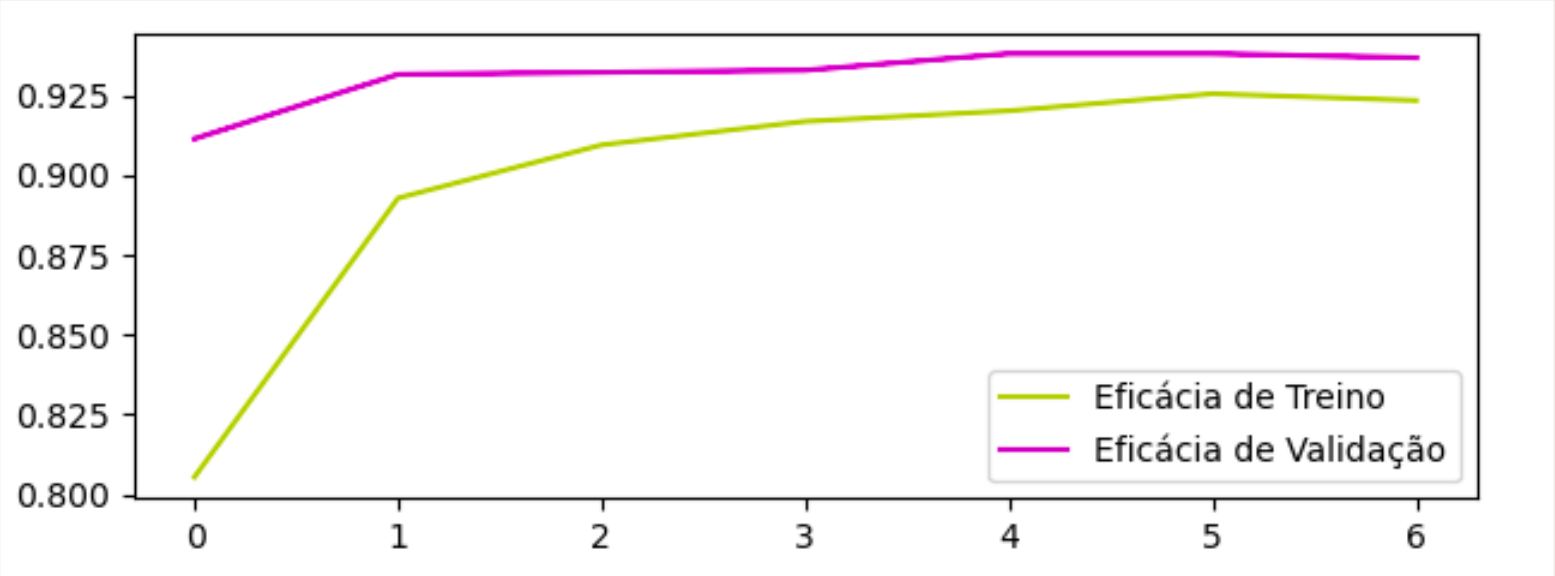
GlobalAveragePooling2D

```
Epoch 1/20
193/193 [==============================] - 331s 2s/step - loss: 0.6619 - categorical_accuracy: 0.7992 - val_loss:
0.3179 - val_categorical_accuracy: 0.9115
Epoch 2/20
193/193 [==============================] - 341s 2s/step - loss: 0.3374 - categorical_accuracy: 0.8907 - val_loss:
0.2730 - val_categorical_accuracy: 0.9219
Epoch 3/20
193/193 [==============================] - 348s 2s/step - loss: 0.2827 - categorical_accuracy: 0.9060 - val_loss:
0.2535 - val_categorical_accuracy: 0.9219
Epoch 4/20
193/193 [==============================] - 350s 2s/step - loss: 0.2559 - categorical_accuracy: 0.9149 - val_loss:
0.2452 - val_categorical_accuracy: 0.9277
Epoch 5/20
193/193 [==============================] - 342s 2s/step - loss: 0.2430 - categorical_accuracy: 0.9176 - val_loss:
0.2477 - val_categorical_accuracy: 0.9290
Epoch 6/20
193/193 [==============================] - 345s 2s/step - loss: 0.2278 - categorical_accuracy: 0.9266 - val_loss:
0.2529 - val_categorical_accuracy: 0.9277
Epoch 7/20
193/193 [==============================] - 347s 2s/step - loss: 0.2223 - categorical_accuracy: 0.9263 - val_loss:
0.2194 - val_categorical_accuracy: 0.9362
Epoch 8/20
193/193 [==============================] - 348s 2s/step - loss: 0.2137 - categorical_accuracy: 0.9289 - val_loss:
0.2238 - val_categorical_accuracy: 0.9336
Epoch 9/20
193/193 [==============================] - ETA: 0s - loss: 0.2187 - categorical_accuracy: 0.9240Restoring model we
ights from the end of the best epoch: 7.
193/193 [==============================] - 343s 2s/step - loss: 0.2187 - categorical_accuracy: 0.9240 - val_loss:
0.2227 - val_categorical_accuracy: 0.9323
Epoch 9: early stopping
```

Data Augmentation - Teste

Softmax

GlobalAveragePooling2D



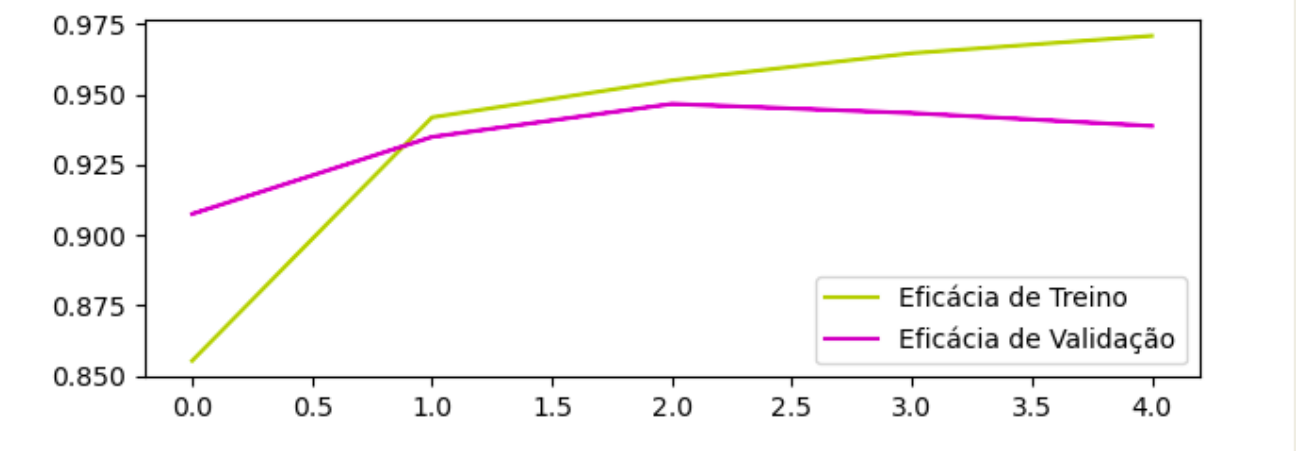

```
Eficácia de teste =  93.69 %
```

Data Augmentation - Teste

Softmax

GlobalAveragePooling2D

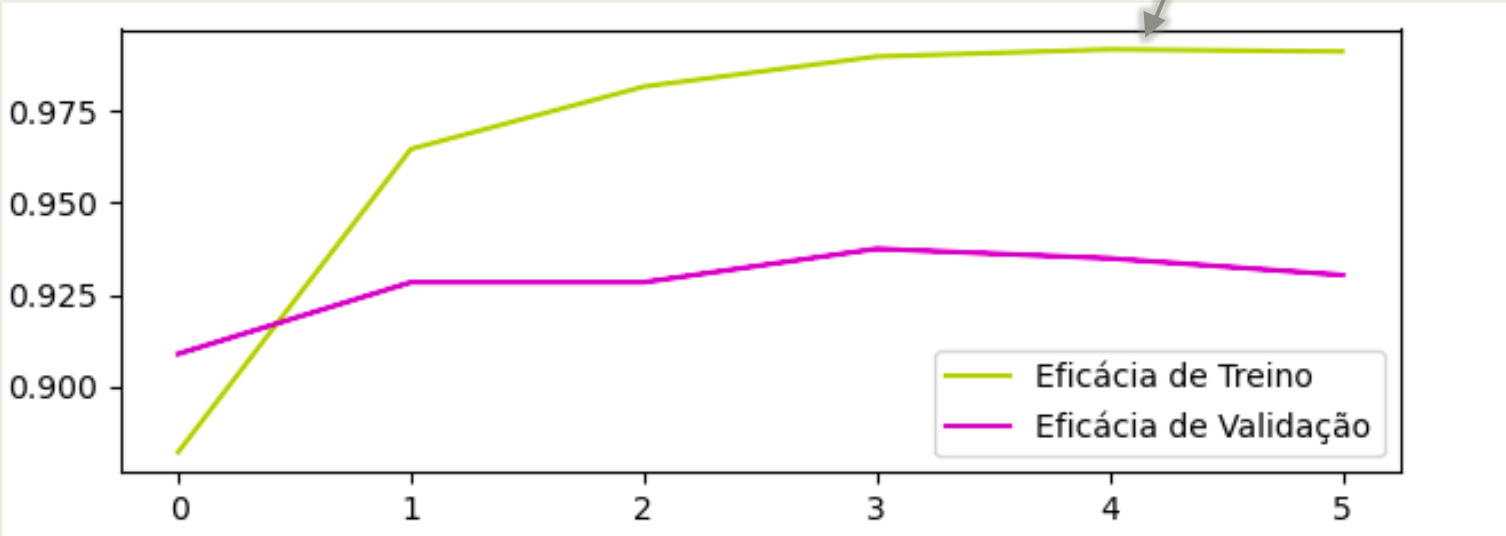
```
Epoch 1/20
193/193 [==============================] - 284s 1s/step - loss: 0.4882 - categorical_accuracy: 0.8554 - val_loss:
0.2689 - val_categorical_accuracy: 0.9076
Epoch 2/20
193/193 [==============================] - 282s 1s/step - loss: 0.1962 - categorical_accuracy: 0.9419 - val_loss:
0.2049 - val_categorical_accuracy: 0.9349
Epoch 3/20
193/193 [==============================] - 283s 1s/step - loss: 0.1498 - categorical_accuracy: 0.9550 - val_loss:
0.1789 - val_categorical_accuracy: 0.9466
Epoch 4/20
193/193 [==============================] - 279s 1s/step - loss: 0.1230 - categorical_accuracy: 0.9646 - val_loss:
0.1821 - val_categorical_accuracy: 0.9434
Epoch 5/20
193/193 [==============================] - ETA: 0s - loss: 0.1036 - categorical_accuracy: 0.9708Restoring model we
ights from the end of the best epoch: 3.
193/193 [==============================] - 279s 1s/step - loss: 0.1036 - categorical_accuracy: 0.9708 - val_loss:
0.1680 - val_categorical_accuracy: 0.9388
Epoch 5: early stopping
```



Eficácia de teste = 94.33 %

Flatten - Teste

```
model.add(Flatten(name="featuresCamadaFlatten"))
#model.add(tf.keras.layers.GlobalAveragePooling2D(name="featuresCamadaGlobal"))

model.add(Dense(len(categories), activation='softmax'))
```




```
Eficácia de teste =  93.69 %
```

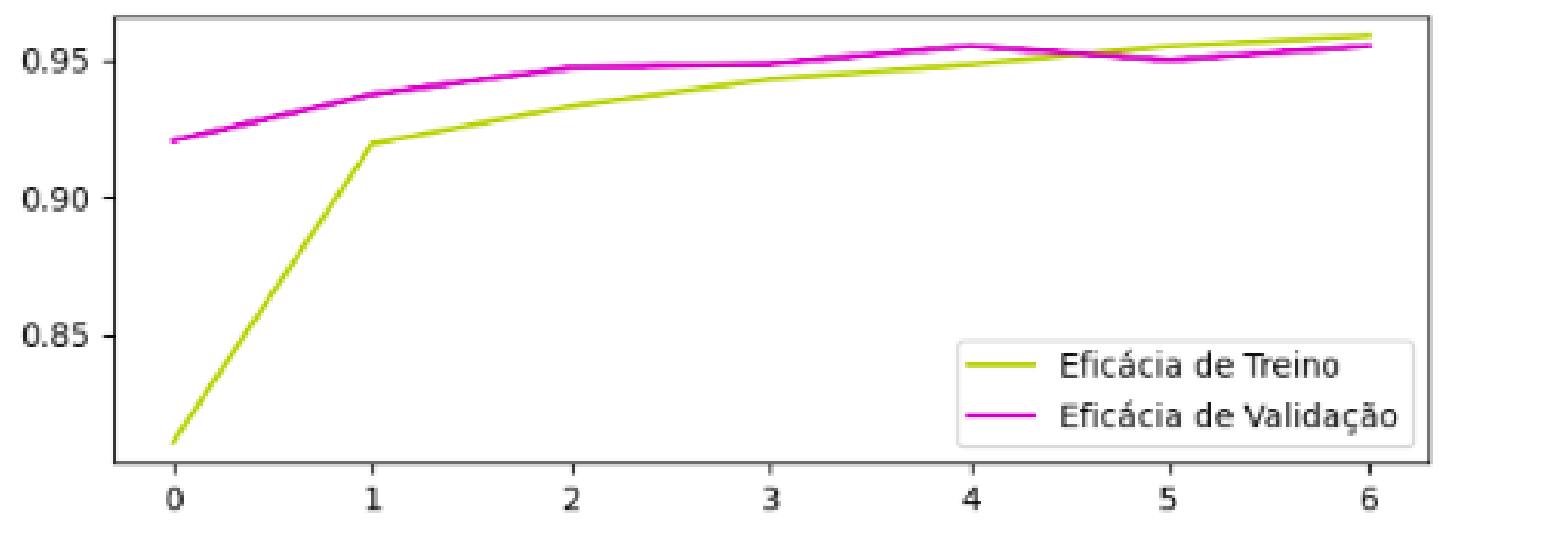
GlobalAveragePooling2D + Relu
Flatten + Relu

```
Epoch 1/20
193/193 [==============================] - 280s 1s/step - loss: nan - categorical_accuracy: 0.2017 - val_loss: nan
- val_categorical_accuracy: 0.0618
Epoch 2/20
193/193 [==============================] - 280s 1s/step - loss: nan - categorical_accuracy: 0.0611 - val_loss: nan
- val_categorical_accuracy: 0.0625
Epoch 3/20
193/193 [==============================] - ETA: 0s - loss: nan - categorical_accuracy: 0.0613Restoring model weigh
ts from the end of the best epoch: 1.
193/193 [==============================] - 281s 1s/step - loss: nan - categorical_accuracy: 0.0613 - val_loss: nan
- val_categorical_accuracy: 0.0625
Epoch 3: early stopping
```

Eficácia de teste = 5.8 %

Eficácia de teste = 6.51 %

GlobalAveragePooling2D + Flatten + Dropout + Softmax




```
Eficácia de teste =  94.85 %
```

Previsões

	precision	recall	f1-score	support
battery	0.98	0.93	0.95	85
biological	0.94	0.98	0.96	104
cardboard	0.93	0.98	0.96	87
clothes	0.99	0.98	0.99	539
glass	0.93	0.91	0.92	200
metal	0.87	0.84	0.85	69
paper	0.95	0.93	0.94	114
plastic	0.87	0.87	0.87	99
shoes	0.93	0.97	0.95	191
trash	0.92	0.94	0.93	64
accuracy			0.95	1552
macro avg	0.93	0.93	0.93	1552
weighted avg	0.95	0.95	0.95	1552

Parte 2 - Grid Search

Manual Search

Escolhe-se os hiperparametros com base no nosso julgamento, na nossa experiência. Depois treinamos o modelo, avaliamos a sua *accuracy* e repetimos o processo. Paramos, quando o valor da *accuracy* nos é satisfatório.

Random Search

É onde as diversas combinações de hiperparametros são feitas de forma aleatória e são usadas para encontrar a melhor solução.

Bayesian Optimization

É uma estratégia de projeto sequencial para funções do tipo “black-box” (é um dispositivo, sistema, ou objeto que produz informações úteis sem revelar nenhuma informação sobre o seu funcionamento interno. Ou seja, as explicações para as suas conclusões permanecem opacas ou “negras”, o que é muito frequente nesta área de Inteligência computacional), a qual reduz também o número de iterações de pesquisa ao escolher os valores de entrada tendo em mente os valores anteriores.

Evolutionary Algorithm

Cria uma população de N modelos de *machine learning* com alguns hiperparametros predefinidos. Depois gera alguns descendentes com hiperparametros semelhantes aos dos melhores modelos para obter novamente uma população de N modelos. Assim, no final do processo, apenas alguns modelos é que vão sobreviver fazendo combinações e variações de parâmetros que são semelhantes à evolução biológica. Este algoritmo simula o processo de seleção natural, o que significa que as espécies que se podem adaptar às mudanças no seu ambiente, podem sobreviver, reproduzir-se e passar para a próxima geração.

Grid Search

É o método tradicional de otimização de hiperparametros e funciona como uma grelha de hiperparametros e dados de treino e teste em que constrói um modelo para cada combinação de hiperparametros especificados e avalia cada modelo.