

In [1]:

```
# Não exibir warnings
import os
import sys
sys.stderr = open(os.devnull, "w") # silence stderr
sys.stderr = sys.__stderr__ # unsilence stderr
```

In [2]:

```
#https://github.com/PacktPublishing/Neural-Network-Projects-with-Python/blob/master/Chapter04/main_vgg16.py

from keras.applications.vgg16 import VGG16
from keras.models import Model

from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix
import tensorflow as tf
import numpy as np
import pandas as pd
import seaborn as sns

from PIL import Image, ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True
```

In [3]:

```
# Load and evaluate a saved model
from numpy import loadtxt
from keras.models import load_model

# Load model
model = load_model('modelo_classificador_B_VGG16.h5')
# summarize model.
model.summary()
```

WARNING:tensorflow:From D:\Usuarios\spi112884\Anaconda3\lib\site-packages\tensorflow_core\python\ops\resource_variable_ops.py:1630: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops) with constraint is deprecated and will be removed in a future version.

Instructions for updating:

If using Keras pass *_constraint arguments to layers.

WARNING:tensorflow:From D:\Usuarios\spi112884\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:4070: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From D:\Usuarios\spi112884\Anaconda3\lib\site-packages\tensorflow_core\python\ops\math_grad.py:1424: where (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.

Instructions for updating:

Use tf.where in 2.0, which has the same broadcast rule as np.where

WARNING:tensorflow:From D:\Usuarios\spi112884\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Model: "model_1"

Layer (type)	Output Shape	Param #
=====		
input_1 (InputLayer)	(None, 128, 128, 3)	0

vgg16 (Model)	(None, 4, 4, 512)	14714688

flatten (Flatten)	(None, 8192)	0

dense_1 (Dense)	(None, 2)	16386
=====		
Total params: 14,731,074		
Trainable params: 16,386		
Non-trainable params: 14,714,688		

In [4]:

```

#Função de geração da matriz de confusão
def print_confusion_matrix(confusion_matrix, class_names, figsize = (10,7), fontsize=11
):
    df_cm = pd.DataFrame(
        confusion_matrix, index=class_names, columns=class_names,
    )
    fig = plt.figure(figsize=figsize)
    try:
        heatmap = sns.heatmap(df_cm, cmap="YlGnBu", annot=True, fmt="d")
    except ValueError:
        raise ValueError("Confusion matrix values must be integers.")
    heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right'
, fontsize=fontsize)
    heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=30, ha='righ
t', fontsize=fontsize)

    b, t = plt.ylim() # discover the values for bottom and top
    b += 0.5 # Add 0.5 to the bottom
    t -= 0.5 # Subtract 0.5 from the top
    plt.ylim(b, t) # update the ylim(bottom, top) values

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    #return fig

```

In [5]:

```

batch = 32

from keras.preprocessing.image import ImageDataGenerator

validation_datagen = ImageDataGenerator(rescale = 1./255)

validation_set = validation_datagen.flow_from_directory('classificador_B/validation/',
                                                        target_size = (128, 128),
                                                        color_mode="rgb",
                                                        batch_size = batch, #alterado para 1
                                                        class_mode = 'categorical',
                                                        shuffle=True)

num_validation = validation_set.samples

```

Found 1200 images belonging to 2 classes.

In [6]:

```

### Conjunto de Validação ###

print ("### Matriz de confusão para o conjunto de validação ###")

#Conjunto de validação
validation_datagen = ImageDataGenerator(rescale = 1./255)

validation_set = validation_datagen.flow_from_directory('classificador_B/validation/',
                                                    target_size = (128, 128),
                                                    color_mode="rgb",
                                                    batch_size = batch, #alterado para 1
                                                    class_mode = 'categorical',
                                                    shuffle= False)

#Confution Matrix
Y_pred = model.predict_generator(validation_set, num_validation//batch, verbose=1)

test_preds = np.argmax(Y_pred, axis=-1)
l=test_preds.shape[0]
test_trues = validation_set.classes
cm =confusion_matrix(test_trues[:l], test_preds)

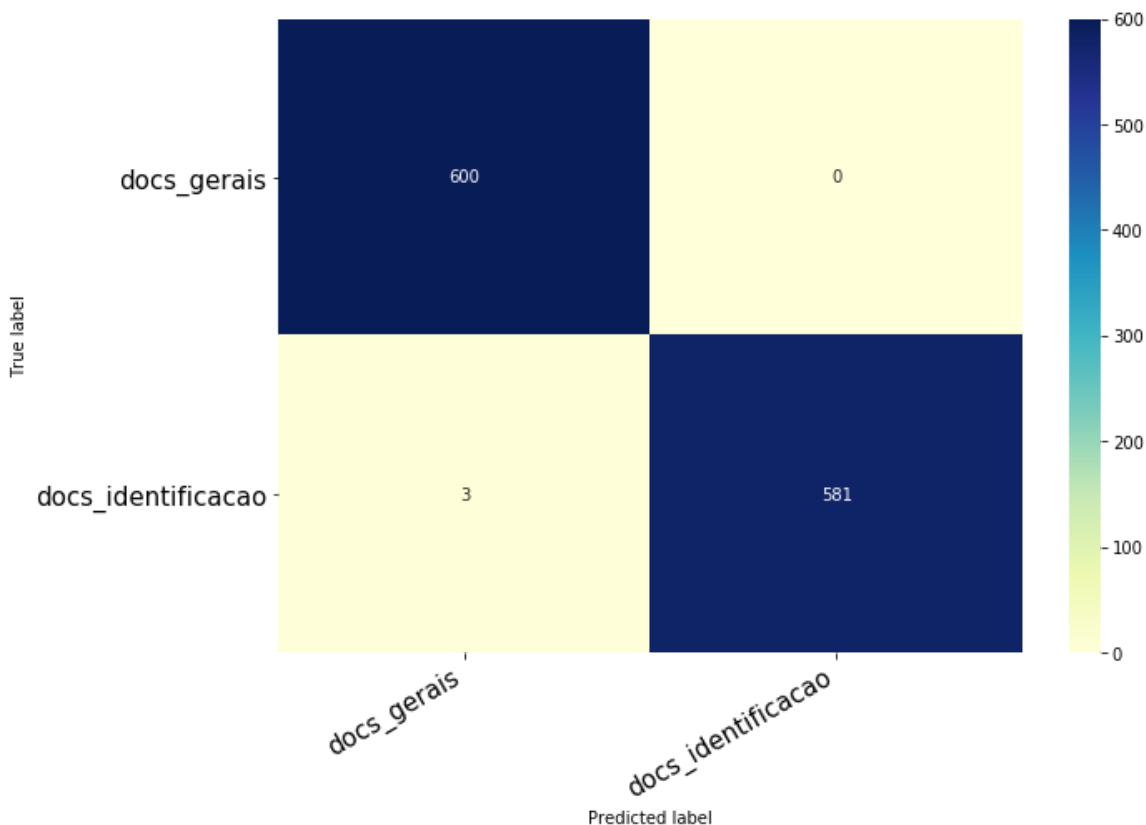
print_confusion_matrix(cm, ["docs_gerais", "docs_identificacao"], figsize = (10,7), fontsize=15)

```

```

### Matriz de confusão para o conjunto de validação ###
Found 1200 images belonging to 2 classes.
37/37 [=====] - 25s 666ms/step

```



In [7]:

```
### Conjunto de Teste ###

print ("### Matriz de confusão para o conjunto de teste ###")

test_datagen = ImageDataGenerator(rescale = 1./255)

test_set = test_datagen.flow_from_directory('classificador_B/test/',
                                            target_size = (128, 128),
                                            color_mode="rgb",
                                            batch_size = 1,
                                            class_mode = 'categorical',
                                            shuffle=False)

num_test = test_set.samples

#Confution Matrix
Y_pred = model.predict_generator(test_set, num_test, verbose=1)

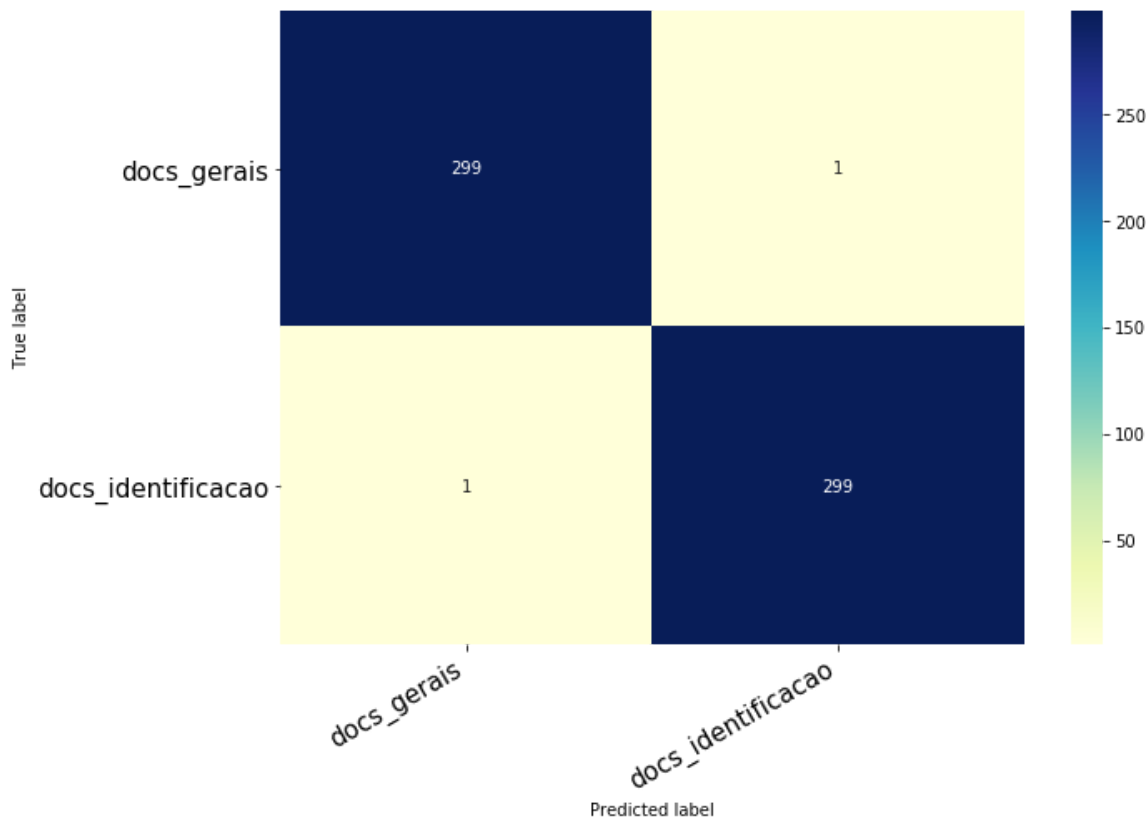
test_preds = np.argmax(Y_pred, axis=-1)
l=test_preds.shape[0]
test_trues = test_set.classes
cm =confusion_matrix(test_trues[:l], test_preds)

print_confusion_matrix(cm, ["docs_gerais", "docs_identificacao"], figsize = (10,7), fontsize=15)

# Accuracy and Loss for the Test set
loss, acc = model.evaluate_generator(test_set, num_test, verbose=1)

# Final accuracy and loss
print ("Test accuracy: %.3f" % acc)
print ("Test loss: %.3f" % loss)
```

```
### Matriz de confusão para o conjunto de teste ###
Found 600 images belonging to 2 classes.
600/600 [=====] - 17s 28ms/step
600/600 [=====] - 13s 21ms/step
Test accuracy: 0.997
Test loss: 0.000
```



In []: