UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria
de Manresa

TREBALL FINAL DE GRAU

IMPLEMENTACIÓ D'UN SISTEMA INTEL·LIGENT DE VISIÓ ARTIFICIAL PER UN PROCÉS DE CONTROL DE QUALITAT INDUSTRIAL MITJANÇANT XARXES NEURONALS

(Annexos)

Grau en Enginyeria Electrònica Industrial i Automàtica

Curs 20/21

Autora: Anna Marbà Mas

Director: Víctor Barcons Xixons

6 de Juliol del 2021, Manresa

ÍNDEX DE CONTINGUT

1. ANNEX A: Codi de programa *main.py*

```
#Llibreries:
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam

#Dataset path:
my_data_dir = 'C:/Users/Anna Marbà/Documents/TFG/Programació/dataset/'
train_path = my_data_dir + 'train_set/'  #carpeta on es guarden les imatges d'entrenament
val_path = my_data_dir + 'val_set/'      #carpeta on es guarden les imatges de validació
image_shape = (300,300,1)            #mida de la imatge. 1 canal ja que és en escala de grisos

#Hiperparàmetres:
epochs = 20
lr = 0.001
batch_size = 4

#Tibar el dataset del directori i preprocessament de les dades:
image_gen = ImageDataGenerator(rescale=1/255)
train_set = image_gen.flow_from_directory(train_path,
                                target_size=image_shape[:2],  #adapta la mida de la imatge inicial al
que necessita la CNN
                                color_mode="grayscale",
                                batch_size=batch_size,
                                class_mode='binary',shuffle=True)
val_set = image_gen.flow_from_directory(val_path,
                                target_size=image_shape[:2],
                                color_mode="grayscale",
                                batch_size=batch_size,
                                class_mode='binary',shuffle=False)
```

```
#Creació del model:
model = Sequential()

model.add(Conv2D(filters=8, kernel_size=(3,3),padding='same', input_shape=image_shape,
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=16, kernel_size=(3,3),padding='same', input_shape=image_shape,
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(224, activation='relu'))

model.add(Dense(1, activation='sigmoid'))


#Compilar el model:
model.compile(loss='binary_crossentropy',
        optimizer=Adam(lr=lr),
        metrics=['accuracy'])
early_stop = EarlyStopping(monitor='loss',patience=5)

#Entrenar el model:
results = model.fit_generator(train_set,epochs=epochs,
                    validation_data=val_set,
                    callbacks=[early_stop])

#Guardar el model i els pesos:
dir='C:/Users/Anna Marbà/Documents/TFG/Programació/model'
model.save('C:/Users/Anna Marbà/Documents/TFG/Programació/model/model.h5')
model.save_weights('C:/Users/Anna Marbà/Documents/TFG/Programació/model/weights.h5')

#Resum de l'arquitectura del model:
model.summary()
```

```
#Metrics history:
losses = pd.DataFrame(model.history.history)  #La llibreria Pandas permet preparar les dades emmagatzemades al history per representar-les en un gràfic

#Mètriques (loss i accuracy) gràfics:
plt.subplots(figsize=(7,7))
sns.lineplot(data=losses, palette='Set1')
plt.xlabel('Epochs')
plt.legend(labels=[
   'Training Loss',
   'Training Accuracy',
   'Validation Loss',
   'Validation Accuracy'])
plt.ylim(0,1)
plt.title('Model Evaluation', size=17, weight='bold')
plt.show()
```

2. ANNEX B: Codi de programa *predict.py*

```
#Llibreries:
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.models import load_model

#Mida de la imatge a predir:
image_shape=(300,300,1)

#Path dels fitxers model.h5 i weights.h5 :
model_architecture = 'C:/Users/Anna Marbà/Documents/TFG/Programació/model/model.h5'
model_weights = 'C:/Users/Anna Marbà/Documents/TFG/Programació/model/weights.h5'

#Carregar l'arquitectura del model i pesos per construir el model amb els paràmetres que ha
après durant l'entrenament:
model = load_model(model_architecture)
model.load_weights(model_weights)

#Triar la imatge per predir i carregar-la:
img = load_img('C:/Users/Anna Marbà/Documents/TFG/Programació/prova/365.bmp',
target_size=image_shape[:2], color_mode="grayscale")
img = img_to_array(img)
img=img/255  #és necessari per carregar la imatge preprocessada
img = np.expand_dims(img, axis=0) #afegeix un nou eix per insertar la predicció: (prediction, 300,
300, 1)

#Predir la classe de la imatge d'entrada:
prediction = model.predict(img)
print(prediction)  #probabilitat que la imatge pertanyi a la classe OK
if prediction < 0.5:
   print("classification: NOK")
else:
   print("classification: OK")  #programar una classificació binària a partir del valor entre 0 i 1
corresponent a la probabilitat
```

3. ANNEX C: Taula dels valors de les mètriques

En aquest *Annex* es mostren els valors de les mètriques per cada configuració corresponents als millors resultats obtinguts de totes les execucions realitzades de l'algoritme.

- 20 èpoques:

			lr						
			0.00001	0.0001	0.001	0.004	0.006	0.008	0.01
batch_size	4	Loss	0.6607	0.3102	0.0022	0.0148	0.0956	0.2191	0.6934
		Accuracy	0.6498	0.9536	1.0000	0.9921	0.9758	0.9068	0.5311
		Val_loss	0.6695	0.3641	0.0429	0.0514	0.1235	0.5148	0.6933
		Val_accuracy	0.5250	0.8625	0.9750	0.9750	0.9675	0.8000	0.5000
	8	Loss	0.6413	0.5807	0.0254	0.0025	0.1239	0.1333	0.693
		Accuracy	0.6390	0.6523	0.9972	1.0000	0.9515	0.9404	0.5229
		Val_loss	0.6494	0.5757	0.1040	0.0598	0.3118	0.4036	0.6931
		Val_accuracy	0.7500	0.6625	0.9750	0.9875	0.8750	0.8125	0.5000
	16	Loss	0.6669	0.6864	0.0707	0.0030	0.0149	0.2062	0.6928
		Accuracy	0.8715	0.4889	0.9862	1.0000	0.9972	0.9184	0.5306
		Val_loss	0.6740	0.6866	0.1607	0.0702	0.1338	0.2435	0.6932
		Val_accuracy	0.7375	0.5000	0.9250	0.9875	0.9750	0.9375	0.5000
	32	Loss	0.6778	0.6079	0.1264	0.0473	0.2417	0.1704	0.6932
		Accuracy	0.6163	0.6613	0.9821	0.9840	0.9270	0.9494	0.4882
		Val_loss	0.6791	0.6305	0.2430	0.1524	0.2226	0.1637	0.6931
		Val_accuracy	0.5250	0.7625	0.9375	0.9375	0.9500	0.9375	0.5000

- Convergència:

			lr						
			0.00001	0.0001	0.001	0.004	0.006	0.008	0.01
batch_size	4	Loss	0.4682	0.0477	0.0022	0.0148	0.0956	0.2191	0.6934
		Accuracy	0.9089	0.9951	1.0000	0.9921	0.9758	0.9068	0.5311
		Val_loss	0.5165	0.1135	0.0429	0.0514	0.1235	0.5148	0.6933
		Val_accuracy	0.7750	0.9625	0.9750	0.9750	0.9675	0.8000	0.5000
	8	Loss	0.4995	0.1968	0.0254	0.0025	0.1239	0.1333	0.6930
		Accuracy	0.8959	0.9719	0.9972	1.0000	0.9515	0.9404	0.5229
		Val_loss	0.5596	0.2756	0.1040	0.0598	0.3118	0.4036	0.6931
		Val_accuracy	0.7500	0.9000	0.9750	0.9875	0.8750	0.8125	0.5000
	16	Loss	0.5430	0.1497	0.0707	0.0030	0.0149	0.2062	0.6928
		Accuracy	0.8965	0.9614	0.9862	1.0000	0.9972	0.9184	0.5306
		Val_loss	0.5909	0.2178	0.1607	0.0702	0.1338	0.2435	0.6932
		Val_accuracy	0.7875	0.9375	0.9250	0.9875	0.9750	0.9375	0.5000
	32	Loss	0.5946	0.3243	0.1264	0.0473	0.2417	0.1704	0.6932
		Accuracy	0.8523	0.9500	0.9821	0.9840	0.9270	0.9494	0.4882
		Val_loss	0.6217	0.3939	0.2430	0.1524	0.2226	0.1637	0.6931
		Val_accuracy	0.7500	0.8875	0.9375	0.9375	0.9500	0.9375	0.5000