Steel Corrosion DL

October 5, 2018

0.0.1 Create a model, read data and train, validate, test. Save all results, from each run to files, including the trained models

In [6]:
```
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, l
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.utils import to_categorical

import os
import glob #to count jpg files
import h5py
from keras.callbacks import ModelCheckpoint
import numpy as np
import matplotlib.pyplot as plt
from __future__ import print_function
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Dropout, Flatten
from keras import regularizers, optimizers

from keras.applications import VGG19, ResNet50, xception, inception_v3

import time
import winsound #to play a sound when the program finishes

image_size = 197

def runCNN(application = 'ResNet50', noOfRuns = 1, epochs = 100, batch_size = 32, image

    trainPath = 'TVT/TVTAll/train'
    validPath = 'TVT/TVTAll/validation'
    testPath = 'TVT/TVTAll/test'

    jpgCounterValid = len(glob.glob(validPath + "/**/*.jpg"))
    jpgCounterTest = len(glob.glob(testPath + "/**/*.jpg"))

    for i in range (0, noOfRuns):
```

```
print("Starting run ", i)
start_time = time.time()
keras.backend.clear_session()
if application == 'ResNet50':
    res_conv = ResNet50(weights='imagenet', include_top=False, input_shape=(ima
    print('ResNet50 application')
elif application == 'VGG19':
    res_conv = VGG19(weights='imagenet', include_top=False, input_shape=(image_
    print('VGG19 application')
elif application == 'Xception':
    res_conv = xception.Xception(weights='imagenet', include_top=False, input_s
    print('Xception application')
elif application == 'InceptionV3':
    res_conv = inception_v3.InceptionV3(weights='imagenet', include_top=False,
    print('InceptionV3 application')

# Create the model
# Freeze the layers except the last 4 layers
for layer in res_conv.layers[:-4]:
    layer.trainable = False
model1 = Sequential()

# Add the res convolutional base model
model1.add(res_conv)

# Add new layers
model1.add(Flatten())
model1.add(Dense(1024, activation='relu'))
model1.add(Dropout(0.5))

model1.add(Dense(7, activation='softmax'))


lossValue = 'categorical_crossentropy'
classModeValue = 'categorical'

model1.compile(loss=lossValue, optimizer=optimizers.RMSprop(lr=1e-4), metrics=
# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(rescale=1./255,
                        rotation_range=20,
                        width_shift_range=0.2,
                        height_shift_range=0.2,
                        horizontal_flip=True,
                        fill_mode='nearest')

# this is the augmentation configuration we will use for testing:
# only rescaling
valid_datagen = ImageDataGenerator(rescale=1./255)
```

```
test_datagen = ImageDataGenerator(rescale=1./255)
seed = 5

# this is a generator that will read pictures found in
# subfolDers of 'data/train', and indefinitely generate
# batches of augmented image data
train_generator = train_datagen.flow_from_directory(
        trainPath,  # this is the target directory
        target_size=(image_size, image_size),
        batch_size=batch_size,
        class_mode=classModeValue, seed=seed, shuffle = True)  # since we use

# this is a similar generator, for validation data
validation_generator = valid_datagen.flow_from_directory(
        validPath,
        target_size=(image_size, image_size),
        batch_size=batch_size,
        class_mode=classModeValue, shuffle = False)

test_generator = test_datagen.flow_from_directory(
        testPath,
        target_size=(image_size, image_size),
        batch_size=jpgCounterTest,
        class_mode=classModeValue, shuffle = False)

outputFolder = 'TVT\SavedResults\\All\\' + application
#create the folder
if not os.path.exists(outputFolder):
    os.makedirs(outputFolder)

filepath = outputFolder + "\WeightsBestRun" + str(i) +".hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, save_best
callbacks_list = [checkpoint]

history2 = model1.fit_generator(
        train_generator,
        #steps_per_epoch=100,#2000 // batch_size,
        epochs=epochs,
        validation_data=validation_generator,
        callbacks=callbacks_list,
        validation_steps=jpgCounterValid // batch_size)
elapsedTrainingTime = time.time() - start_time
model1.load_weights(filepath)

xTest, yTest = test_generator.next()
evaluation = model1.evaluate(xTest, yTest)
print("Loss and evaluation on the test set in run ", i, ":", evaluation)
```

```
                ########################################

                #save results to file
                #import os.path
                outputFileName = outputFolder + '\ResultsCNN.txt'
                if os.path.isfile(outputFileName):#if the file exists, add to it, otherwise cr
                    f = open(outputFileName,'a')

                    f.write(str(i) + '\t' + str(epochs) + '\t' + str(batch_size) + '\t' + str(
                            + '\t' + str(noOfRuns) + '\t' + str(jpgCounterValid) + '\t' + str(
                            + '\t' + str(evaluation[0]) + '\t' + str(evaluation[1])
                            + '\t' + str(min(history2.history['loss'])) + '\t' + str(max(histo
                            + '\t' + str(min(history2.history['val_loss'])) + '\t' + str(max(h
                            + '\t' + str(elapsedTrainingTime) + '\n')
                    f.close() #I did not test if these close() are necessary. It works without
                else:
                    f = open(outputFileName,'w')
                    f.write('currentRun\tepochs\tbatch_size\timage_size\tnoOfRuns\tvalidationI
                    f.close()
                    f = open(outputFileName,'a')
                    f.write(str(i) + '\t' + str(epochs) + '\t' + str(batch_size) + '\t' + str(
                            + '\t' + str(noOfRuns) + '\t' + str(jpgCounterValid) + '\t' + str(
                            + '\t' + str(evaluation[0]) + '\t' + str(evaluation[1])
                            + '\t' + str(min(history2.history['loss'])) + '\t' + str(max(histo
                            + '\t' + str(min(history2.history['val_loss'])) + '\t' + str(max(h
                            + '\t' + str(elapsedTrainingTime) + '\n')
                    f.close()
                #save acc and loos for train and validation during evolution
                outputFileRuntime = outputFolder + '\ResultsRuntimeCNN' + str(i) + '.txt'
                print('The following file is saved: ', outputFileRuntime)
                np.savetxt(outputFileRuntime, np.c_[history2.history['loss'], history2.history

            print(noOfRuns, ' runs are over and the result files are saved.')

        # 'OlAPVCVLP', 'OlEtalonCV' = over
        # 'OlEtalonLP' gata cu 'VGG19', 'ResNet50'

        for apps in ['VGG19', 'ResNet50', 'Xception', 'InceptionV3']:
            runCNN(application = apps, noOfRuns = 10, epochs = 100, batch_size = 16, image_size
        frequency = 2500  # Set Frequency To 2500 Hertz
        duration = 1000  # Set Duration To 1000 ms == 1 second
        #notify that the run is over
        winsound.Beep(frequency, duration)

Starting run  0
VGG19 application
Found 339 images belonging to 7 classes.
Found 113 images belonging to 7 classes.
```

```
Found 111 images belonging to 7 classes.
Epoch 1/100
22/22 [==============================] - 19s 864ms/step - loss: 2.1683 - acc: 0.1512 - val_loss

Epoch 00001: val_acc improved from -inf to 0.26549, saving model to TVT\SavedResults\All\VGG19\
Epoch 2/100
22/22 [==============================] - 12s 554ms/step - loss: 1.8245 - acc: 0.2882 - val_loss

Epoch 00002: val_acc improved from 0.26549 to 0.38938, saving model to TVT\SavedResults\All\VGG
Epoch 3/100
22/22 [==============================] - 12s 554ms/step - loss: 1.7896 - acc: 0.3358 - val_loss

Epoch 00003: val_acc did not improve from 0.38938
Epoch 4/100
22/22 [==============================] - 12s 556ms/step - loss: 1.5777 - acc: 0.3679 - val_loss

Epoch 00004: val_acc did not improve from 0.38938
Epoch 5/100
22/22 [==============================] - 12s 556ms/step - loss: 1.5377 - acc: 0.3817 - val_loss

Epoch 00005: val_acc improved from 0.38938 to 0.42478, saving model to TVT\SavedResults\All\VGG
Epoch 6/100
22/22 [==============================] - 12s 553ms/step - loss: 1.4824 - acc: 0.4102 - val_loss

Epoch 00006: val_acc improved from 0.42478 to 0.46903, saving model to TVT\SavedResults\All\VGG
Epoch 7/100
22/22 [==============================] - 12s 554ms/step - loss: 1.3262 - acc: 0.4760 - val_loss

Epoch 00007: val_acc did not improve from 0.46903
Epoch 8/100
22/22 [==============================] - 12s 555ms/step - loss: 1.3441 - acc: 0.4386 - val_loss

Epoch 00008: val_acc did not improve from 0.46903
Epoch 9/100
22/22 [==============================] - 12s 554ms/step - loss: 1.2752 - acc: 0.4842 - val_loss

Epoch 00009: val_acc did not improve from 0.46903
Epoch 10/100
22/22 [==============================] - 12s 554ms/step - loss: 1.1983 - acc: 0.5732 - val_loss

Epoch 00010: val_acc improved from 0.46903 to 0.51327, saving model to TVT\SavedResults\All\VGG
Epoch 11/100
22/22 [==============================] - 12s 554ms/step - loss: 1.2223 - acc: 0.5354 - val_loss

Epoch 00011: val_acc did not improve from 0.51327
Epoch 12/100
22/22 [==============================] - 12s 555ms/step - loss: 1.2559 - acc: 0.5329 - val_loss
```

0.0.2 Load a model and compute confusion matrices from it

In [3]:
```
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.models import Sequential
from keras.applications import ResNet50
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array,
from keras.applications import VGG19, ResNet50, xception, inception_v3
import os

import time #to measure runtime
import keras
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt
import glob #to count jpg files
import numpy as np



def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues,
                          fileName = 'CM.pdf'):

    f = plt.figure()
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        cm = np.around(100*cm, decimals = 1)
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
```

```
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

    #the matrix could be saved as pdf from the method
    f.savefig(fileName, bbox_inches='tight')

image_size = 197 # should be the same as in the call of runCNN

#just change application to the desired architecture
application = 'InceptionV3'#'ResNet50'
#application = 'ResNet50'
runNo = 0

#10 is the total no of runs
#in cmTest all cm will be gathered to compute the average afterwards
cmTest = np.ndarray(shape=(10, 7, 7), dtype=float)

#compute an average over 10 runs for the confusion matrices
for runNo in range(0, 10):
    keras.backend.clear_session()
    modelFilePath = 'TVT\SavedResults\\All\\' + application + "\WeightsBestRun" + str(


    lossValue = 'categorical_crossentropy'
    classModeValue = 'categorical'

    if application == 'ResNet50':
        res_conv = ResNet50(weights='imagenet', include_top=False, input_shape=(image_s
        print('ResNet50 application ', runNo)
    elif application == 'VGG19':
        res_conv = VGG19(weights='imagenet', include_top=False, input_shape=(image_size
        print('VGG19 application ', runNo)
    elif application == 'Xception':
        res_conv = xception.Xception(weights='imagenet', include_top=False, input_shape
        print('Xception application ', runNo)
    elif application == 'InceptionV3':
        res_conv = inception_v3.InceptionV3(weights='imagenet', include_top=False, inpu
        print('InceptionV3 application ', runNo)


    # Freeze the layers except the last 4 layers
    for layer in res_conv.layers[:-4]:
        layer.trainable = False

    # Create the model
    model1 = Sequential()
```

```
    # Add the res convolutional base model
    model1.add(res_conv)

    # Add new layers
    model1.add(Flatten())
    model1.add(Dense(1024, activation='relu'))
    model1.add(Dropout(0.5))


    model1.add(Dense(7, activation='softmax'))

    model1.load_weights(modelFilePath)

    testPath = 'TVT/TVTAll/test'
    jpgCounterTest = len(glob.glob(testPath + "/**/*.jpg"))

    test_datagen = ImageDataGenerator(rescale=1./255)
    test_generator = test_datagen.flow_from_directory(
                    testPath,
                    target_size=(image_size, image_size),
                    batch_size=jpgCounterTest,
                    class_mode=classModeValue, shuffle = False)

    testBatch, trueLabelsTest = next(test_generator)

    startTime = time.time()
    predLabelsTest = model1.predict(testBatch)
    stopTime = time.time()

    print('It took ', (stopTime - startTime), ' seconds to classify ', jpgCounterTest,


    cmTest[runNo] = confusion_matrix(trueLabelsTest.argmax(1), predLabelsTest.argmax(a

averageCM = np.mean( np.array(cmTest), axis=0 )

print(averageCM)

cm_plot_labels = ['$CV_{Noinhib}$', '$CV_{PVA}$', '$CV_{PVA/nAg}$', '$LP_{Noinhib}$',


outputFolder = 'TVT\SavedResults\\All\\' + application
#create the folder
if not os.path.exists(outputFolder):
    os.makedirs(outputFolder)

cmFileName = outputFolder + "\cmAll" + application + ".pdf"
```

```
        plot_confusion_matrix(averageCM, cm_plot_labels, normalize = False, title='', fileName
InceptionV3 application  0
Found 111 images belonging to 7 classes.
It took  6.0706117153167725  seconds to classify  111  files, that is  0.054690195633484436  se
InceptionV3 application  1
Found 111 images belonging to 7 classes.
It took  3.928145408630371  seconds to classify  111  files, that is  0.035388697375048385  sec
InceptionV3 application  2
Found 111 images belonging to 7 classes.
It took  3.945848226547241  seconds to classify  111  files, that is  0.035548182221146314  sec
InceptionV3 application  3
Found 111 images belonging to 7 classes.
It took  4.02634072303772  seconds to classify  111  files, that is  0.03627333984718666  secon
InceptionV3 application  4
Found 111 images belonging to 7 classes.
It took  3.9039859771728516  seconds to classify  111  files, that is  0.03517104483939506  sec
InceptionV3 application  5
Found 111 images belonging to 7 classes.
It took  3.90279483795166  seconds to classify  111  files, that is  0.035160313855420365  seco
InceptionV3 application  6
Found 111 images belonging to 7 classes.
It took  3.864328622817993  seconds to classify  111  files, that is  0.034813771376738675  sec
InceptionV3 application  7
Found 111 images belonging to 7 classes.
It took  3.9351577758789062  seconds to classify  111  files, that is  0.03545187185476492  sec
InceptionV3 application  8
Found 111 images belonging to 7 classes.
It took  3.8992459774017334  seconds to classify  111  files, that is  0.035128342138754355  se
InceptionV3 application  9
Found 111 images belonging to 7 classes.
It took  4.022607803344727  seconds to classify  111  files, that is  0.03623970994004258  seco
[[ 7.3  2.4  3.3  1.5  0.5  0.6  0.4]
 [ 2.7  6.4  1.5  2.   0.   0.9  3.5]
 [ 1.8  4.8  3.7  1.   0.3  0.8  0.6]
 [ 0.4  0.9  0.6 11.   0.   0.5  3.6]
 [ 0.1  0.2  0.5  0.1 11.   2.7  2.4]
 [ 0.8  0.5  1.2  1.4  5.4  5.2  0.5]
 [ 0.5  0.4  0.   7.   0.4  0.   7.7]]
Confusion matrix, without normalization
[[ 7.3  2.4  3.3  1.5  0.5  0.6  0.4]
 [ 2.7  6.4  1.5  2.   0.   0.9  3.5]
 [ 1.8  4.8  3.7  1.   0.3  0.8  0.6]
 [ 0.4  0.9  0.6 11.   0.   0.5  3.6]
 [ 0.1  0.2  0.5  0.1 11.   2.7  2.4]
 [ 0.8  0.5  1.2  1.4  5.4  5.2  0.5]
 [ 0.5  0.4  0.   7.   0.4  0.   7.7]]
```