

# PROIECT INVATARE AUTOMATA

Cats vs Dogs Classification using CNN

Student: Bucă-Ghebaură Elizabetha Alexandrina

Specializare: Calculatoare Romana

Grupa: 4.S1 A

Din foarte multe imagini cu pisici si caini, am incercat sa clasific ce animal este prezent intr-o anumita imagine, folosind *CNN* -> *Convolutional Neural Networks*.

Pentru inceput, am importat librariile necesare pentru functionarea programului:

Getting Started

+ Code + Markdown

```
[21]: import numpy as np
import pandas as pd
from pathlib import Path
import os.path

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split

import tensorflow as tf

from sklearn.metrics import confusion_matrix, classification_report
```

Ne vor trebui librariile numpy, pandas, os, matplotlib, seaborn, sklearn, tensorflow.

Am specificat apoi Path-ul dataset-ului:

```
image_dir = Path('../input/cat-and-dog-images-dataset/Dog and Cat .png')
```

Mai departe, am initializat *filepaths* cu o lista de imagini in format *.png* si apoi m-am folosit de libraria *os.path.split*, utilizata pentru a imparti numele path-ului intr-o pereche head si tail. Tail este ultima componenta a numele path-ului, iar head este inceputul path-ului. Adica vom avea drept head -> */input/cat-and-dog-images-dataset/*, iar tail -> *Dog and Cat.png*.

## Create File DataFrame

```
filepaths = list(image_dir.glob(r'**/*.png'))
labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1], filepaths))

filepaths = pd.Series(filepaths, name='Filepath').astype(str)
labels = pd.Series(labels, name='Label')

image_df = pd.concat([filepaths, labels], axis=1)
```

Am afisat apoi efectiv dataset-ul in mod tabelar, in randuri si coloane. Avem 999 linii si 2 coloane (pisici si caini).

image_df		
	Filepath	Label
0	../input/cat-and-dog-images-dataset/Dog and Ca...	Dog
1	../input/cat-and-dog-images-dataset/Dog and Ca...	Dog
2	../input/cat-and-dog-images-dataset/Dog and Ca...	Dog
3	../input/cat-and-dog-images-dataset/Dog and Ca...	Dog
4	../input/cat-and-dog-images-dataset/Dog and Ca...	Dog
...	...	...
994	../input/cat-and-dog-images-dataset/Dog and Ca...	Cat
995	../input/cat-and-dog-images-dataset/Dog and Ca...	Cat
996	../input/cat-and-dog-images-dataset/Dog and Ca...	Cat
997	../input/cat-and-dog-images-dataset/Dog and Ca...	Cat
998	../input/cat-and-dog-images-dataset/Dog and Ca...	Cat

999 rows × 2 columns

Am facut split pentru train si test:

```
train_df, test_df = train_test_split(image_df, train_size=0.7, shuffle=True, random_state=1)
```

Mai departe, am incarcat practic datele dataset-ului si am gasit 560, 139 si 300 de imagini valide apartinand a 2 clase.

## Load Image Data

```
train_generator = tf.keras.preprocessing.image.ImageDataGenerator(  
    rescale=1./255,  
    horizontal_flip=True,  
    width_shift_range=0.2,  
    height_shift_range=0.2,  
    validation_split=0.2  
)  
  
test_generator = tf.keras.preprocessing.image.ImageDataGenerator(  
    rescale=1./255  
)
```

```
train_images = train_generator.flow_from_dataframe(  
    dataframe=train_df,  
    x_col='Filepath',  
    y_col='Label',  
    target_size=(224, 224),  
    color_mode='rgb',  
    class_mode='binary',  
    batch_size=32,  
    shuffle=True,  
    seed=42,  
    subset='training'  
)  
  
val_images = train_generator.flow_from_dataframe(  
    dataframe=train_df,  
    x_col='Filepath',  
    y_col='Label',  
    target_size=(224, 224),  
    color_mode='rgb',  
    class_mode='binary',  
    batch_size=32,  
    shuffle=True,  
    seed=42,  
    subset='validation'  
)  
  
test_images = test_generator.flow_from_dataframe(  
    dataframe=test_df,  
    x_col='Filepath',  
    y_col='Label',  
    target_size=(224, 224),  
    color_mode='rgb',  
    class_mode='binary',  
    batch_size=32,  
    shuffle=False  
)
```

Found 560 validated image filenames belonging to 2 classes.  
Found 139 validated image filenames belonging to 2 classes.  
Found 300 validated image filenames belonging to 2 classes.

Mai departe, am antrenat modelele.

## Training

+ Code

+ Markdown

```
inputs = tf.keras.Input(shape=(224, 224, 3))
x = tf.keras.layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu')(inputs)
x = tf.keras.layers.MaxPool2D()(x)
x = tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu')(x)
x = tf.keras.layers.MaxPool2D()(x)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dense(128, activation='relu')(x)
x = tf.keras.layers.Dense(128, activation='relu')(x)
outputs = tf.keras.layers.Dense(1, activation='sigmoid')(x)

model = tf.keras.Model(inputs=inputs, outputs=outputs)

model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

history = model.fit(
    train_images,
    validation_data=val_images,
    epochs=100,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(
            monitor='val_loss',
            patience=5,
            restore_best_weights=True
        ),
        tf.keras.callbacks.ReduceLROnPlateau(
            monitor='val_loss',
            patience=3
        )
    ]
)
```

Un *Epoch* este atunci cand toate datele de training sunt utilizate simultan si sunt definite ca numarul total de iteratii ale tuturor datelor de training sau sa zicem ca s-ar putea defini ca numarul de treceri pe care le face un set de date de training in jurul unui algoritim.

*EarlyStopping* ne permite sa „masuram” performanta programului, si odata ce va fi apelat, va opri procesul de training. Este practic o oprire timpurie a training-ului, printr-un apel invers.

*ReduceLROnPlateau* se va apela atunci cand o valoare nu se mai imbunatateste. Acest apel monitorizeaza o cantitate, si daca nu se observa nicio imbunatatire pentru un numar de „rabdare” de epochs, atunci rata de invatare este redusa. Astfel, am observat, ca nu mereu vom avea acelasi numar de epochs.

Output pentru epochs:

```
Epoch 1/100
18/18 [=====] - 10s 553ms/step - loss: 0.6959 - accuracy: 0.4889 - val_loss: 0.6920 - val_accuracy: 0.4820
Epoch 2/100
18/18 [=====] - 10s 525ms/step - loss: 0.6920 - accuracy: 0.5279 - val_loss: 0.6895 - val_accuracy: 0.4892
Epoch 3/100
18/18 [=====] - 10s 573ms/step - loss: 0.6867 - accuracy: 0.5443 - val_loss: 0.6914 - val_accuracy: 0.5180
Epoch 4/100
18/18 [=====] - 10s 555ms/step - loss: 0.6820 - accuracy: 0.5782 - val_loss: 0.6845 - val_accuracy: 0.5612
Epoch 5/100
18/18 [=====] - 10s 564ms/step - loss: 0.6804 - accuracy: 0.5708 - val_loss: 0.6716 - val_accuracy: 0.6043
Epoch 6/100
18/18 [=====] - 10s 558ms/step - loss: 0.6618 - accuracy: 0.6318 - val_loss: 0.6778 - val_accuracy: 0.5396
Epoch 7/100
18/18 [=====] - 10s 533ms/step - loss: 0.6620 - accuracy: 0.6021 - val_loss: 0.6573 - val_accuracy: 0.6331
Epoch 8/100
18/18 [=====] - 10s 536ms/step - loss: 0.6381 - accuracy: 0.6325 - val_loss: 0.6521 - val_accuracy: 0.6403
Epoch 9/100
18/18 [=====] - 10s 562ms/step - loss: 0.6685 - accuracy: 0.5863 - val_loss: 0.6532 - val_accuracy: 0.5971
Epoch 10/100
18/18 [=====] - 10s 540ms/step - loss: 0.6402 - accuracy: 0.6263 - val_loss: 0.6568 - val_accuracy: 0.6187
Epoch 11/100
18/18 [=====] - 9s 525ms/step - loss: 0.6375 - accuracy: 0.6389 - val_loss: 0.6916 - val_accuracy: 0.5468
Epoch 12/100
18/18 [=====] - 10s 545ms/step - loss: 0.6735 - accuracy: 0.6040 - val_loss: 0.6854 - val_accuracy: 0.5612
Epoch 13/100
18/18 [=====] - 10s 555ms/step - loss: 0.6824 - accuracy: 0.5550 - val_loss: 0.6645 - val_accuracy: 0.6115
```

Aici ni se afiseaza in cate secunde se calculeaza, ce loss avem, ce acuratete, valoarea pentru loss si valoarea pentru acuratete.

Rezultatele in urma testarii imaginilor: **Test Loss: 0.65543; Test Accuracy: 62.67%**

## Results

+ Code

+ Markdown

```
results = model.evaluate(test_images, verbose=0)

print("Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

Test Loss: 0.65543  
Test Accuracy: 62.67%

Urmeaza predictia:

```
predictions = (model.predict(test_images) >= 0.5).astype(np.int)

cm = confusion_matrix(test_images.labels, predictions, labels=[0, 1])
clr = classification_report(test_images.labels, predictions, labels=[0, 1], target_names=["CAT", "DOG"])

plt.figure(figsize=(6, 6))
sns.heatmap(cm, annot=True, fmt='g', vmin=0, cmap='Blues', cbar=False)
plt.xticks(ticks=[0.5, 1.5], labels=["CAT", "DOG"])
plt.yticks(ticks=[0.5, 1.5], labels=["CAT", "DOG"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

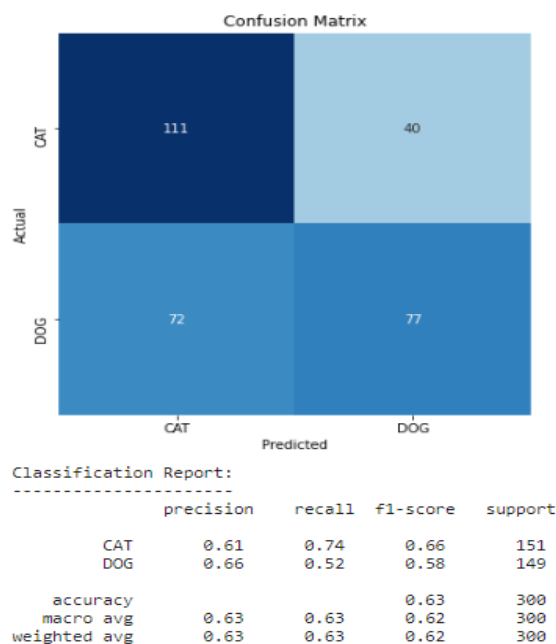
print("Classification Report:\n-----\n", clr)
```

Aici avem o matrice cu ce s-a prezis, si care sunt de fapt datele actuale corecte:

Actual: 111 cat, 72 dog

Predicted: 40 cat, 77 dog

Adica, in final am avea o precizie de 0.61 pentru pisici si 0.66 pentru caini.



## REFERINTE:

1. <https://www.datacamp.com/tutorial/convolutional-neural-networks-python>
2. <https://www.geeksforgeeks.org/cat-dog-classification-using-convolutional-neural-network-in-python/>
3. <https://www.kaggle.com/code/kashit/cat-and-dog-classification-with-cnn/notebook>
4. <https://www.kaggle.com/code/gpreda/cats-or-dogs-using-cnn-with-transfer-learning/notebook>
5. <https://www.kaggle.com/code/mihailsavushkin/homework-2>
6. <https://www.kaggle.com/datasets/arpitjain007/dog-vs-cat-fastai/code>
7. <https://www.kaggle.com/datasets/shaunthesheep/microsoft-catsvsdogs-dataset/code>
8. <https://www.kaggle.com/competitions/petfinder-pawpularity-score/discussion/274259>
9. <https://www.kaggle.com/code/bulentsiyah/dogs-vs-cats-classification-vgg16-fine-tuning>
10. <https://www.youtube.com/watch?v=FLf5qmS0kwU>

## Concluzii

Facand research pentru acest proiect pe internet, am observat ca exista foarte multe metode de rezolvare a acestei teme. Initial incercasem sa citesc toate datele, facand cate un label pentru fiecare poza si incercand sa fac o identificare pentru ambele animale. Astfel, as fi parcurs intr-un fel aceiasi pasi pe care i-am parcurs si in aceasta varianta finala prezentata mai sus, doar ca in alte moduri. Exemplu de varianta pe care incercasem sa o fac: <https://www.kaggle.com/code/rajansahu/dog-vs-cat>

In final, consider ca am invatat multe lucruri interesante facand research pentru acest proiect (chiar daca toate cunostintele pe care le-am acumulat nu se afla in totalitate in varianta finala prezentata mai sus) si tema pe care mi-am ales-o m-a facut sa fiu cat mai curioasa - sa aflu mai multe moduri de rezolvare a problemei si sa observ cum functioneaza diferite metode folosite pentru solutionarea temei.

In continuare, pe viitor, voi incerca sa rezolv in modul/varianta despre care vorbeam mai sus (cel cu citirea efectiva a pozelor, afisare a anumitor poze si incercarea de a „ghici” ce fel de animal se afla acolo).