cat and dog classifier

Content: Explanation Convolution operation ReLU Max Pooling Flattening Full-connection Softmax & Cross entropy IMPLEMENTATION Data preprocessing Build the Keras model Compile and fit the model Make predictions and determine accuracy Below are key concepts used in Convolutional Neural Networks.

Convolution Operation The aim of convolution operation is to reduce the size of an image, by using feature detectors that keep only the specific patterns within the image. Stride is the number of pixels with which we slide the detector. If it is one, we are moving it one pixel each time and recording the value (adding up all the multiplied values). Many feature detectors are used, and the algorithm finds out what is the optimal way to filter images. 3 x 3 feature detector is commonly used, but other sizes can be used.

ReLU After feature detectors are applied upon images, ReLU is used to increase non-linearity within images.

Max Pooling Take a 2 x 2 box on the top left corner (starting here), and record the maximum number within the box. Slide it to the right with the stride of 2 (commonly used), and move onto the next row if completed. Repaet this step until all the pixels are evaluated. Aim of max pooling is to keep all the important features even if images have spatial or textual distortions, and also reduce the size which prevents overfitting. So, after applying convolution operation to images, than pooling is applied.

Other pooling techniques are also available such as Mean Pooling, which takes the average of pixels within the box.

Flattening Flatten the matrix into a long vector which will be the input to the artificial neural network

Full Connection Implement full Artificial Neural Network model to optimize weights.

Softmax & Cross entropy Softmax function brings all predicted values to be between 0 and 1, and make them add up to 1. It also comes hand-in-hand with cross-entropy method.

Just seeing how many wrong predictions the classifier made is not enough to evaluate the performance of ANNs. Instead, Cross Entropy should be used to measure how good the model is, as there can be two models that produce same results while one produced better percentages than the other. For classificaion, Cross Entropy should be used, and for regression, Mean Squared Error should be used.

We will create a dog vs cat classifier. To be able to work with keras library, we need proper structure of images. There should be two folders: Test set and Train set. And, in each folder, cat images and dog images should be placed in two separate folders. In this way, keras will understand how to work with them.

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, M
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import KFold
import itertools
import os
import shutil
import random
import glob
import matplotlib.pyplot as plt
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
%matplotlib inline
train_path = 'gdrive/MyDrive/cat_dog/training_set'
test_path =  'gdrive/MyDrive/cat_dog/test_set'
```

```
train_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preproc
    .flow_from_directory(directory=train_path, target_size=(224,224), classes=['cats', 'dogs'
test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preproce
    .flow_from_directory(directory=test_path, target_size=(224,224), classes=['cats', 'dogs']
```

```
Found 8005 images belonging to 2 classes.
Found 2023 images belonging to 2 classes.
```

```
imgs, labels = next(train_batches)
def plotImages(images_arr):
    fig, axes = plt.subplots(1, 10, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()

plotImages(imgs)
print(labels)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
```

```
[[0. 1.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
 [0. 1.]]
```

```
model = Sequential([
    Conv2D(filters=32, kernel_size=(3, 3), activation='relu', padding = 'same', input_shape=(
    MaxPool2D(pool_size=(2, 2), strides=2),
    Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPool2D(pool_size=(2, 2), strides=2),
    Flatten(),
    Dense(units=2, activation='softmax')
])
model.summary()
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
conv2d (Conv2D)             (None, 224, 224, 32)      896

_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 32)     0

_________________________________________________________________
conv2d_1 (Conv2D)           (None, 112, 112, 64)      18496

_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 64)       0

_________________________________________________________________
flatten (Flatten)           (None, 200704)            0

_________________________________________________________________
dense (Dense)               (None, 2)                 401410
```

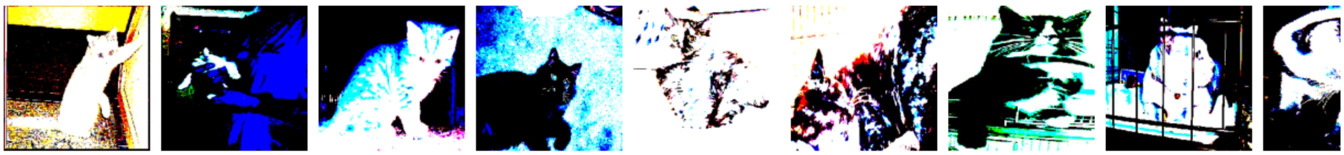
```
=================================================================
Total params: 420,802
Trainable params: 420,802
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(x=train_batches, epochs=4)
```

```
Epoch 1/4
801/801 [==============================] - 1256s 2s/step - loss: 11.1732 - accuracy: 0.5
Epoch 2/4
801/801 [==============================] - 41s 51ms/step - loss: 0.2959 - accuracy: 0.88
Epoch 3/4
801/801 [==============================] - 41s 52ms/step - loss: 0.0863 - accuracy: 0.97
Epoch 4/4
801/801 [==============================] - 41s 51ms/step - loss: 0.0295 - accuracy: 0.99
<tensorflow.python.keras.callbacks.History at 0x7f4387d2e3c8>
```

```
test_imgs, test_labels = next(test_batches)
plotImages(test_imgs)
print(test_labels)
predictions = model.predict(x=test_batches, verbose=0)
np.round(predictions)
cm = confusion_matrix(y_true=test_batches.classes, y_pred=np.argmax(predictions, axis=-1))
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [
```



```
[[1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]]
```