Gustav Nilsson(guni1700), Elias Berglin(elbe1703) 2021-02-25

# Laboration 3 - Artificial Neural Network

## What do you have to install on the local computer? Document the code, and the results:

IDE: Pycharm
Python Packages:

- TensorFlow
- SciPy
- Pillow

Cuda Drivers & CUNN to enable GPU processing

## Key concepts:

- Multi-layer perceptron:

It is when you have a ANN network with feedforwarding, and multiple layers.

- Sigmoid:

Sigmoid is an activation function often used for neural networks, it transforms the input to the function into a value [0 → 1.0].

- Weight:

Weight is the parameter within a neural network that exists in the hidden layers of neural networks that changes the value of the input depending on the path it takes through the neural network. So different paths have different weight amounts and by doing that it can find the right paths.

- Feed forward:

It does not form cycles between nodes but instead has an input layer → hidden layer → output layer for example.

- Back propagation:

It is a fine tuning method of the neural network weights based on errors in the epoch before, reducing error rates and making the model more reliable.

- Gradient:

Gradient measure changes in weights done in regards to errors calculated from a loss function and is used in back propagation.

- Deep learning:

Deep learning is a class of machine which uses multiple layers to get higher level features from input.

## How many attributes does your dataset have?

We have the input of a picture that is 200x200 pixels.

## What is the size and accuracy of your ANN model?

Accuracy: 99.9%

## How long GPU time did the training require?

about 38 minutes or: 2289.245589017868s

## How can the training time be reduced?

You can use GPU, which we did to speed up the training of the model.

## Code

### Dataset split:

```
import os
import random
from shutil import copyfile

def split_data(source, training, testing, split_size):
    files = []
    for filename in os.listdir(source):
        file = source + filename
        if os.path.getsize(file) > 0:
            files.append(filename)
    training_length = int(len(files) * split_size)
    testing_length = int(len(files) - training_length)
    shuffled_set = random.sample(files, len(files))
    training_set = shuffled_set[0: training_length]
    testing_set = shuffled_set[:testing_length]

    for filename in training_set:
        this_file = source + filename
        destination = training + filename
        # print(this_file)
        # print(destination)
        copyfile(this_file, destination)

    for filename in testing_set:
        this_file = source + filename
        destination = testing + filename
        copyfile(this_file, destination)

cat_source = 'PetImages/Cat/'
training_cats_dir = 'cats-vs-dogs/training/cats/'
testing_cats_dir = 'cats-vs-dogs/testing/cats/'
dog_source = 'PetImages/Dog/'
training_dog_dir = 'cats-vs-dogs/training/dogs/'
testing_dog_dir = 'cats-vs-dogs/testing/dogs/'

split = 0.7

split_data(cat_source, training_cats_dir, testing_cats_dir, split)
split_data(dog_source, training_dog_dir, testing_dog_dir, split)
```

## Creation and Training of Model:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import time

traingin_dir = 'cats-vs-dogs/training'
train_datagen = ImageDataGenerator(rescale=1.0 / 255.)
train_generator = train_datagen.flow_from_directory(traingin_dir, batch_size=64,
class_mode='binary', target_size=(200, 200))

model = keras.models.Sequential([
    keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(200, 200, 3)),
    keras.layers.MaxPool2D(2, 2),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Conv2D(128, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(2, 2),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(1, activation='sigmoid')
])

with tf.device('/gpu:0'):
    model.compile(optimizer=keras.optimizers.SGD(lr=0.001, momentum=0.9),
loss='binary_crossentropy', metrics=['accuracy'])
    start = time.time()
    model.fit(train_generator, epochs=50, verbose=1)
    end = time.time()
    elapsed_time = end - start
    print(elapsed_time)

model.save('cat_vs_dog.h5')
```

## Testing the Model:

```
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator

model = keras.models.load_model('cat_vs_dog.h5')

model.summary()

datagen = ImageDataGenerator(rescale=1.0 / 255.)

testing_dir = 'cats-vs-dogs/testing'

test_data = datagen.flow_from_directory(testing_dir, batch_size=64, class_mode='binary',
target_size=(200, 200))

step_size = test_data.n // test_data.batch_size

test = model.evaluate(test_data, return_dict=True, batch_size=64)

print(test)
```