Face Detection

Computational intelligence project
Faculty of Mathematics
University of Belgrade

Darko Mladenovski 278/2017

September 2023

Contents

1 Description

Face detection is a computer vision problem in computer science that involves identifying and locating human faces within digital images or video frames. The primary objective of face detection is to determine whether a given image or frame contains one or more human faces and, if so, to precisely locate the position and possibly additional information about these faces within the visual data. Face detection has a wide range of applications, including facial recognition, emotion analysis, people counting, video surveillance, and even in consumer applications like automatic tagging in photo management software. It's a fundamental building block in many computer vision systems that deal with human interaction and analysis of visual data.

2 Definition

The input for the face detection problem, as you would assume, is a set of user-chosen images, which will later be used as training data for our model. The goal will be to train up a model that will be able to detect a face in real time, using our device's camera. To this end, we will be using a powerful neural network, combined with a custom classificator and regressor that will help us determine whether a face is present in the frame or not. The quality of the solution will be checked via many runtime metrics, and of course the end result - using our camera to see whether or not it captures the faces correctly.

3 Implementation

The implementation itself will consist of several steps:

1. Data preprocessing (collecting, annotating and partitioning of our data)
2. Image augmentation
3. Building our model (neural network)
4. Training and testing

3.1 Data preprocessing

3.1.1 Collecting data

The data we will be using in this project is fairly easy to deduce - it will consist of images. There are many ways to go about gathering images, but the most efficient way to go about it is via the *opencv* library. We will simply use the predefined functions to capture a certain amount of images.

3.1.2 Data annotation

What does annotation entail in this specific problem? We simply want some indicator whether there is a face on the given image or not. To that end we will use the *LabelMe* annotation tool. What we will do is create bounding boxes (basically just rectangles) around a face, if there is one on the image, and give then suitable labels (in this case, "face"). What we end up with will be a .json file, that we will join to it's corresponding image via their shared identifier.

3.1.3 Data partitioning

As in most problems from this field of study, we will need to separate our data into 3 different partitions:

1. Train - used to train our model
2. Test - used to test how our model performs on new, unseen data
3. Validation - used for hyperparameter tunning and model selection

We will be moving roughly 70% of our data to the training partition, and 15% of the data to both test and validation partitions.

3.2 Image augmentation

Image augmentation serves as a great "quick and dirty" way to increase the volume of our data manyfold. We will be applying simple image manipulation to our data (horizontal flip, vertical flip, negative, etc.) in order to gain a much larger quantity of data, which will in turn help our model to perform better down the line. The solution used in this project uses the *albumentations* library, which makes this process quite easy and straightforward. So with a couple of lines of code, we went from 30 images all the way up to 5400. This will make a big impact on the performance of our model.

3.3 Building the model

Our model will consist of 3 components:

- Base VGG16 neural network
- Classification layer - in this instance binary classification, to determine whether a face is present or not
- Regression layer - needed in order to return the exact coordinates of a bounding box containing a face

VGG16, short for "Visual Geometry Group 16," is a popular convolutional neural network (CNN) architecture for deep learning in computer vision. It was developed by the Visual Geometry Group at the University of Oxford and is known for its simplicity and effectiveness. The "16" in VGG16 refers to the total number of weight layers in the network. This neural network will be used as a base, which we will pass through to our classification and regression layers. The output of this model will contain both classification and regression predictions.

3.4 Training and testing

3.4.1 Training

We talked about how our model looks like, but what about our loss functions? We will need 2 of them, one each for regression and classification. As this is a fairly simple classification we will be using *BinaryCrossentropy()*. For regression we defined our custom loss function which calculates the differences between actual and predicted coordinates, and the width and height of the bounding box, and returns their sum. Next in line is the actual training of the model. My own model was trained over 10 epochs, using the data we partitioned earlier. Each step and epochs tracks the loss functions we defined earlier, and prints them out.

Metrics			
Epoch	Regress loss	Class loss	Total loss
1	0.3335	0.1892	0.4281
2	0.0542	0.0278	0.0681
3	0.0278	0.0180	0.0368
4	0.0307	0.0250	0.0432
5	0.0157	0.0081	0.0197
6	0.0057	0.0008	0.0061
7	0.0032	0.0002	0.0032
8	0.0024	0.0001	0.0024
9	0.0019	0.00007	0.0019
10	0.0014	0.00006	0.0015

3.4.2 Testing

We saw some metrics of our model, now onto the testing phase. Firstly the model was tested on randomized batches of images from our test folder and it showed barely satisfactory results (this is most likely due to the low number of epochs i trained it on). When tested in real time, using my PC camera, model showed mediocre results with tracking faces, but excellent results on edge cases (faces covered with hands, rapidly leaving the frame, etc.). This further solidifies my theory that it needs many more epochs of training, but sadly i am limited by my current hardware.

4 Conclussion

One of the key takeaways from this project is the importance of data. The quality and diversity of the training dataset significantly impact the model's performance. Data preprocessing techniques, including data augmentation and normalization, can make a substantial difference in the model's ability to generalize to real-world scenarios.We also grappled with the trade-off between accuracy and efficiency. While deep learning models offer impressive accuracy, they often come at the cost of increased computational demands. This project allowed us to appreciate the balance that must be struck to achieve real-time or resource-efficient face detection. Face detection remains a crucial component of many intelligent systems, and our exploration has been a valuable step towards mastering this foundational technology.

5 References

- https://github.com/wkentaro/labelme
- https://albumentations.ai/
- https://opencv.org/
- https://machinelearningmastery.com/how-to-perform-face-detection-with-classical-and-deep-learning-methods-in-python-with-keras/
- https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/