Hand gesture recognizer

by

Dominik Bober, Szymon Duda, Adam Klekowski, Przemysław Ziaja

AGH University of Science and Technology

Cracow

2020

TABLE OF CONTENTS

CHAPTER 1. OVERVIEW

1.1 Dataset

Dataset which was used to train the model is available at the following link:
`https://www.kaggle.com/gti-upm/leapgestrecog`

Dataset contains in 2 directories photos grouped by class. Some of photos are in different directories but they are instances of them same class (eg fist and moved_fist).

1.2 Technology

Model was crated with Python3. We used Tensorflow, Matplotlib and OpenCV library. The training stage was accelerated using nvidia gpu and software from nvidia like cuDNN.

1.3 Final application

User can obtain predictions using `hand_gesture_recognition/app.py`. There are to patterns which are accepted by script:

- `python3 app.py path_to_photo`
- `python3 app.py path_to_directory_containing_photos`

To change these patterns it is necessary to modify `get()` method in `app.py`

CHAPTER 2. DATA PREPROCESSING

Data which we have used to train model is very clear. There was no need for large preprocessing, however we wanted to test if we can increase performance of model. We have selected a few algorithms for image processing. On photos we can see hand of person which has pale complexion and black background. Naturally we have choosen algorithms equalizing histograms of photographies so we can strenghten differences in area of photo where hand is placed. `hand_gesture_recognition/dataset_preprocessing/Preprocessing.ipynb` notebook contain all results of preprocessing stage.

2.1 Histogram equalization

First we have tried method simple histogram equalization using `cv2.equalizeHist()`. Method equalized histogram using pixels from whole picture. Thanks to this method we can see face of person making hand gesture, but the hand gestures are not sharp. Around the hand is bright glow. It would not be helpful to recognize gesture.

2.2 Clahe algorithm

Using pixels from whole photo is not helpful so we decided to use adaptive algorithm. Clahe algorithm uses pixel from surrounding area of certain pixel to equalize histogram of photo. As we can see histograms of photographies are much more smoother and boarders between fingers are much more visible. At the same time there is not glow around the hand and face of person showing the hand is visible. It is an advantage because in real world it is nearly impossible to get photography of hand with consistent background.

2.3 BBHE algorithm

Next we have tried Bi-histogram equalization. It is a little more advanced algorithm than standard histogram equalization from previous section and much faster than Clahe algorithm. How does it works? Pixels from a photography are divided in 2 sets. The criterion is brightness of pixel. Next on the 2 sets is performed standard histogram equalization and pixels are merging back in to photography. As we can see the hand is visible, boarders between fingers are clear, however it is because the photographies has black background.

2.4 DSIHE algorithm

DISHE algorithm is a variation of BBHE algorithm. The main difference is the criterion of pixels division. We will not describe the algorithm only advantages and disadvantages. The output photographies uses much more channels than output photographies from standard histogram equalization, however there is a bright glow around a hand and algorithm is more expensive than standard histogram equalization.

2.5 Summary

We decided to use Clahe algorithm. The ouput photographies contain clear boarder between fingers and disorder in background. It is computationally expensive but in real world other algorithms will perform poorly.

Also the preprocessing stage has shown that the dataset despite time invested in making photographies is weak. In real world nobody will take a photography of hand in perfect conditions. The main goal of buildin a machine learnig model is to perform well in real world, not on a test set.

CHAPTER 3. RAW DATA MODEL

First we have created model using data from original dataset.

We have decided that there is no replacement for convolutional neural network so we have tried many combinations of convolutional layers.

3.1 1st iteration

First we have set up gpu acceleration and we have written function to get data from hard. Setting up gpu unit is individual thing, but getting data was tricky. We could not read all data at once. In real world case scenario we will not be able to read all data. We made use of `tensorflow.data.Dataset.from_generator()` method and we have created generator that read photographies from direcotry. Photographies are devided into 10 series and each series is divided into classes. Generator at the same time return photo and label, label is extracted using directory name. 1st model was train to recognize 10 classes. Performance was bad. We have obtained less than 40% accuracy. In every model we made use of Adam optimizer.

3.2 2nd iteration

We realise that it would be tough to create good architecture by ourself. We decided to use VGG-16 architecture. There are learned models in the Internet but VGG-16 has different input shape than our photos. We decided to implement it by ourselves and modify it a little bit. Figure 3.1 presents original VGG-16 architecture. Despite fact that we decreased number of convolutional layers and number of learning parameters was 20 times smaller accuracy jump to 75%.

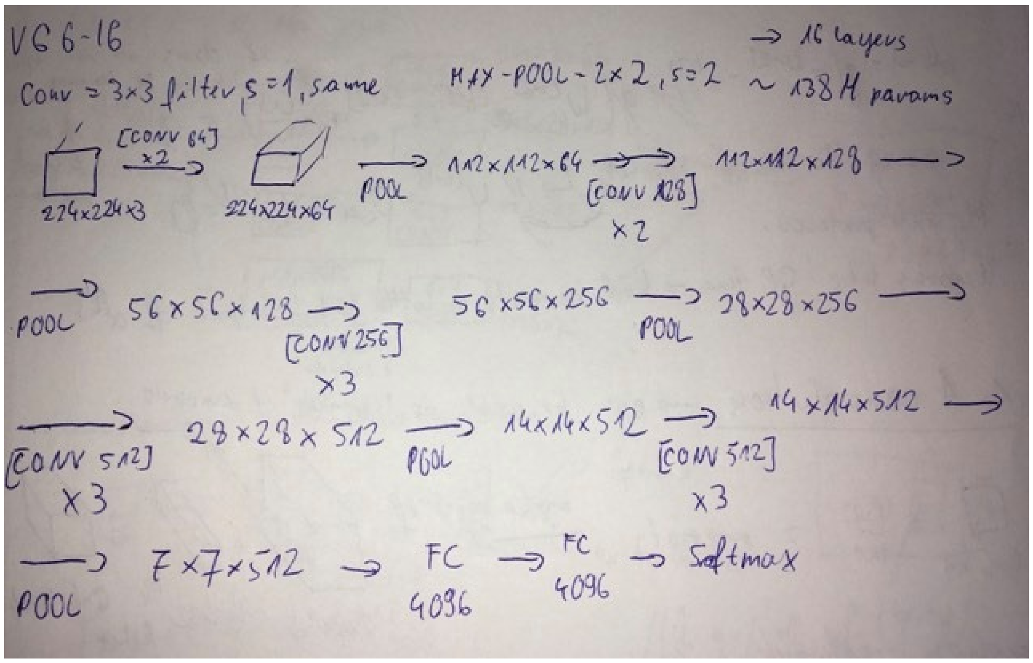


Figure 3.1 VGG-16 architecture

It is worth to notice that we have no information about learning process. After learning with fit method it is possible to get information about parameters like loss, accuracy in each epoch, however our pipeline does not support setting up epoch number in `fit()` method. Number of epochs is set up in method `tf.data.Dataset.from_generator()`. To prevent overfitting we stop learning in random moment and evaluate it to obtain maximum accuracy at test set.

CHAPTER 4. CLAHE MODEL

4.1 Assumptions and preparation

At the begining we made an assumption that instance of class fist and fist_moved is the same class and there is no point to separate moved hand gestures from not moved.

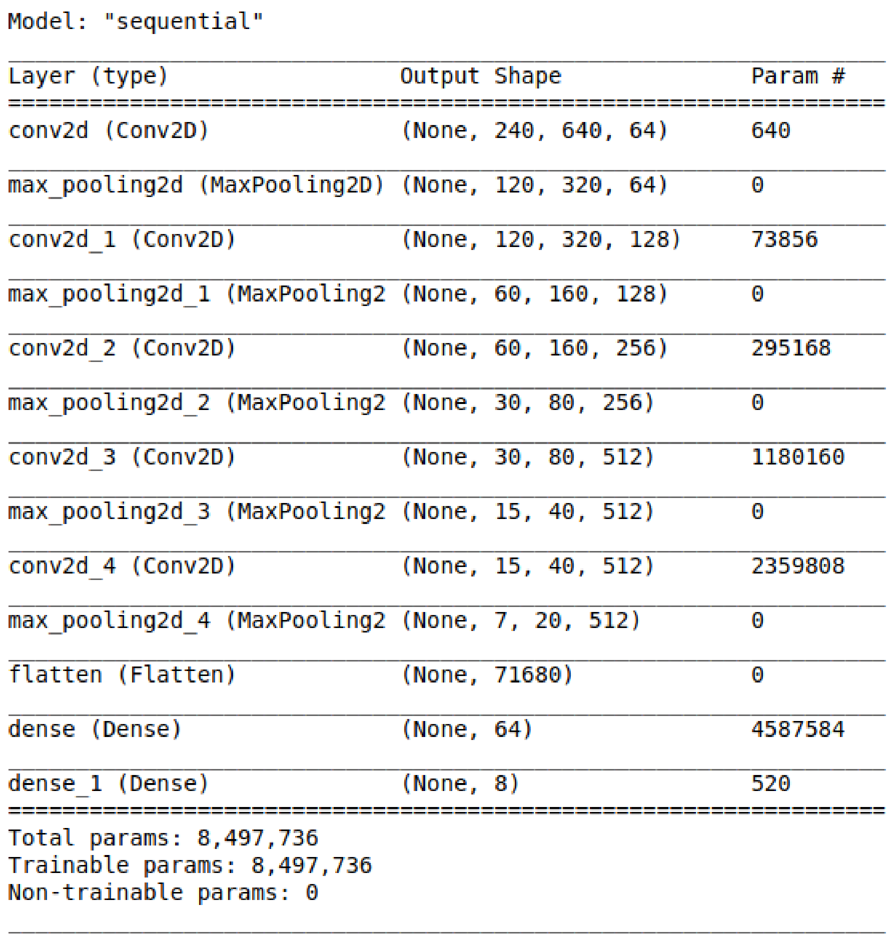
To speed up computations we have prepared script that applies Clahe algorithm on photographies, moves moved hand gustures to not move class and saves it on hard drive. The script is stored in `hand_gesture_recognition/archive/leapgestrecog/to_clahe.py`.

4.2 Model

The greatest change comparing to raw data model is input data. The structure of directories is the same as in original dataset, however the number of categories has decreased by 2. In previous model number of class was encoded in directory name. We have made a simple work around. Depending on directory name data generator scales class number so classes are encoded from 0 to 7.

To prevent overfitting we considered use of method `tf.keras.preprocessing.image.ImageDataGenerator()`. The method takes a picture and can transform it by applying rotation, zoom etc. The transformation retain original size of picture. During training it turned out that regularization L2 on dense layer and breaking training is enough.

Thanks to these changes we obtain 91% accuracy on test set. Below we present summury of our greatest model.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 240, 640, 64)      640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 120, 320, 64)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 120, 320, 128)     73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 60, 160, 128)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 60, 160, 256)      295168
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 80, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 30, 80, 512)       1180160
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 15, 40, 512)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 15, 40, 512)       2359808
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 20, 512)        0
_________________________________________________________________
flatten (Flatten)            (None, 71680)             0
_________________________________________________________________
dense (Dense)                (None, 64)                4587584
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 520
=================================================================
Total params: 8,497,736
Trainable params: 8,497,736
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.1 Model summary

CHAPTER 5. SUMMARY AND DISCUSSION

As we describe model performs very well. Model have similar accuracy on training set and test set so there is no overfitting. Thanks to gpu speed up we could create and train many models and choose the best one.

We have to mention that model work well on data that we got from kaggle, however in real life we are not certain about performance. Photgraphies was taken in laboratory conditions and to improve we should take photos using webcam. The main disadvantage of the model is that we actually do not know to what stimuli the model is sensitive. One possible method to solve the problem is to blur random parts of image and see how does it affect on the prediction. Better solution is to rebuild model and use YOLO algorithm so we could obtain category and frame around the hand.