Project - Report
Naga Maheswara
Deep learning July 19, 2018

Single shot Facial recognition

Definition

1.1 Project overview :

Facial recognition is a classical problem which always attracts a huge interest as it is essential in many scenarios like security unlock, criminal recognition, surveillance etc.

However, this recognition is generally done using a large number of samples for each individual. But this project is intended to recognize a person from a single image available.

This problem is more famously known as one_sample_per_person problem. It is clearly not possible to get a large number of images per each person in every case. It is a hectic process in case of collecting data from a large set of a population or it is very difficult to get at least one picture of a criminal or such person, in such cases, we can never try to get large number of samples. Instead, we need to build a model that could train on a single image per person.

1.2 Problem Statement :

Problem is concerned about obtaining accurate predictions about a person when only one image per person is available for training the model. This model takes an unseen image as an input and predicts the person in the image using the dataset that has only one image per each person.

Workflow of this project is going to be as follows,

1. Data is obtained in such a way that there exists a single sample per each person.
2. Model is trained on the dataset with specific algorithms that work on facial recognition.
3. Unseen images are given as input and the predictions are obtained.
4. Trained model is evaluated using the best metrics that suit for the chosen algorithms.

Thus, we end up with a model that recognizes the person from the input image.

1.3 Metrics :

As the dataset is very sparse and it is being used to retrieve the name of the person (name of the folder in which the train image is located), and we have a large number of such persons, it is preferable to use accuracy

Accuracy is given by,

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_\text{samples}} \sum_{i=0}^{n_\text{samples}-1} 1(\hat{y}_i = y_i)$$

Here y is the actual data and y^ is the predicted person.

Analysis

2.1 Data Exploration :

Data is a large folder containing a subfolder for each person which contains an image of that person. These subfolders are named after that person as we can use these folder names as the labels to the data points.

2.2 Data specification:

Images are to be collected at a specific angle. Crucial muscles that affect the expressions and facial features are *Orbicularis oris* and *Orbicularis oculi*, the intensity of these expressions is directed by *Buccinator(Whistle muscle)*. These muscles are prominently visible in a particular angle for humans which makes it easier to recognize a person from his/her single image. Such an angle is obtained when the face is parallel to the neck and facing the camera exactly.

2.3 Algorithms and Techniques :

This model was approached by trying different algorithms and choose the best algorithm among them. One of the most obvious methods was to use transfer learning where we use a huge and pretrained network on a large dataset. Weights of this pretrained network are of huge value as they've been trained on a huge dataset. Here, the network used is VGG-16 which is pretrained on 2.9 million faces. Such large number of weights is a huge asset as they carry the information related to many faces thus a model human face in general. So, the model is built on these weights, however, this network is not perfectly fit for a user's personal dataset. Thus we retrain a few layers (5 layers from the end) on the user's dataset.

However, this model is not quite suitable in this scenario as the model has learned how a human face looks but it is not trained to identify the specific face in the dataset perfectly as it has just learned a very few weights specific to this dataset.

This situation is best dealt with using Siamese networks which encode a face into 128 measurement vector. Thus we end with each face representing a point in 128-dimensional space. This will immediately be converted into a classification problem which can be accurately solved using KNN Classifier, where each unknown face is taken and k nearest neighbors to the new face are found where we can determine in which class does this face fall in (who's the person in the image).

Identifying a face with a high accuracy using this technique requires an image that covers almost every possible expressions and facial feature distances. Such an image is obtained at an angle discussed previously.

Finally, this model is to be embedded into a web page using *flask* module of python.

2.4 Benchmark :

Setting a benchmark for this model quite a tricky process, as this is a classification problem where the number of classes is the number of images in the training dataset. The worst case benchmark can be set to the accuracy obtained through transfer learning model which used the VGG-16 network.

Here the obtained accuracy was ~1-2% which can be compared to that of a mere guessing. Yet it is the worst possible scenario, thus satisfying the benchmark criteria.

Methodology

3.1 Data Preprocessing :

Data preprocessing generally involves cleaning the data from abnormalities such as outliers, missing values or unbalanced dataset etc. Here the images required to train the data should be collected at a particular angle (as stated in 2.2 Data Specification). All the images that don't meet this criterion are to be discarded and reobtained as they tend to create noise in the model.

Each image should represent its own class and such care is to be taken so that only one image goes into each training class.

Encoding the images to 128-dimensional vector is also analogous to data preprocessing if the classification model is considered to be the actual model.

3.2 Implementation :

Implementing this model involves two major steps namely, training and testing the model.

- Training :
    - This phase involves dividing the data into appropriate sets as mentioned in the previous section.
    - Then we train the model using two different algorithms as discussed in earlier section *2.3 Algorithms and Techniques.*
    - We build a deeply learned model using transfer learning over VGG-16 and next we encode the images using Siamese network, train it using KNN Classifier.
    - Once the model is trained on both the algorithms we move on to the testing phase.

- Testing :
    - This phase involves using a test set to evaluate the model efficiency.
    - Here we predict the test results through the previously trained model and store those results to evaluate.
    - We evaluate this model using accuracy as a metric for the reasons discussed in the earlier section *1.3 Metrics.*

Repeat the whole process until the required efficiency is attained. Once the model attains required efficiency, the best algorithm among both transfer learning and siamese network, KNN Classifier is chosen and decided as the final algorithm for the model.

3.3 Refinement :

Initial solution framed to solve this problem is mentioned in *Benchmark* as roughly being equal to guessworks accuracy ~1-2%.

Benchmark classification was clearly not the best technique to be used. So we rely on transfer learning which is almost equalto the benchmark. Further, we proceed with the Siamese Network and KNN Classifiers discussed in the workflow. We end up with an efficiency of 100% until the test image is distorted through 70^0. It gives the further efficiency of 94% when the test image is distorted through an angle of 90^0 and 68% with a distortion of 120^0.

This taking a transfer learning approach was not a good idea as it barely gives a better performance and neither does it take a less computational power and time.

Results

4.1 Model Evaluation and Validation :

The current model uses a 100 image dataset which is highly flexible (accommodating new data point in the existing data set is quite simple). Now the model is trained on the dataset using the first algorithm (Transfer learning approach). This method requires a huge computational power when compared to the next approach. This approach uses a pretrained VGG-16 network to train the model on scenario particular dataset. First 14 layers of the network are left untouched and last 5 layers of the network are reconstructed and retrained. Here the model uses three convolutional layers followed by a MaxPooling layer each and finally ends with three dense layers.

The second approach involves in using *Siamese network* and *KNN Classifier.* Here the images are encoded into a 128-dimensional vector using *Siamese network,* Where each image is analogous to a point in a 128-dimensional space. Finally, we train the model on the encoded dataset using *KNN Classifier.* Every new test image is again encoded into a 128-dimensional vector and the k nearest neighbors to that point are identified to get the class in which this point falls.

The final result is more comforting as it attains the required evaluation mark. Final model is very stable and robust that it can be fit even over a manipulated dataset. This model becomes so robust as the values of input features are not confined to a fixed pattern and the dataset is large enough to avoid overfitting on any small patterns found. Thus the model is free from overfitting and is robust, as the dataset is a well-distributed dataset. Dataset has been evaluated using accuracy on a test set which has many unseen data points. Thus the model reacts very well to unseen data.

The efficiency of the model is above 95% (until the images are ~85^0-90^0) and is not overfitted, can acommodate slight variation in the data. So, our model is a trustworthy model.

4.2 Justification :

The final model built using the *Siamese network* and *KNN Classifier*. The final results are off the chart when compared to the benchmark model. Final results are 100%, 94% and 68% efficient for various levels of distortion, which are very better than benchmark result which is ~1-2% efficient.

Final solution involves two algorithms *Siamese network* and *KNN Classifier.* This model attains a highly appriciable accuracy on the considered dataset. It is also more convenient as the model doesn’t require a large time to train or predict as the model is being dealt as a classification problem which doesn’t require a large number of layers to get trained.

Final model built is around ~95-100% efficient which is a highly efficient model. So the model is significant enough to identify the person in the image from a single training image (taken at the particular angle as mentioned in *2.2 Data Specifications).* This classification can be done with a very less hardware and time utilization when compared to image classification algorithms which are currently used.

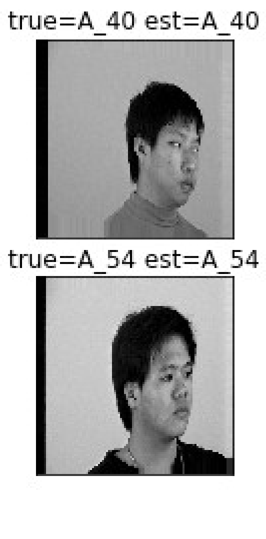
Conclusions

5.1 Free-Form Visualization :

This model is completely involved in image data which is further encoded into numerical format, So a solid pictorial or visual representation is not appropriate in this context. The loss through each epoch of modified VGG-16 network can be given as,

```
Epoch 1/5
17/17 [==============================] - 633s 37s/step - loss: 5.3418 - acc:
0.0074 - val_loss: 5.0793 - val_acc: 0.0114
Epoch 2/5
17/17 [==============================] - 636s 37s/step - loss: 5.1043 - acc:
0.0000e+00 - val_loss: 5.0258 - val_acc: 0.0114
Epoch 3/5
17/17 [==============================] - 957s 56s/step - loss: 5.0150 - acc:
0.0221 - val_loss: 5.0188 - val_acc: 0.0227
Epoch 4/5
17/17 [==============================] - 626s 37s/step - loss: 4.9888 - acc:
0.0221 - val_loss: 5.0554 - val_acc: 0.0227
Epoch 5/5
17/17 [==============================] - 628s 37s/step - loss: 4.8564 - acc:
0.0074 - val_loss: 4.9664 - val_acc: 0.0114
Training loss did not improve more than tol=0.000100 for two consecutive epochs.
Stopping.
```

```
Test output:
```



These visualizations speak about the architecture and complexity of the networks built. However, the architecture of the network is discussed in previous section *4.1 Model Evaluation and Validation.*

This shows us that the transfer learning approach was a great misinterpreted approach in this scenario. Immediate next approach is to use *Siamese network* and *KNN Classifier.* This approach produced a very great result showing an accuracy of ~95%. Whose output can be visualized as follows,

```
Looking for faces in A_05.jpg
(480, 640, 3)
- Found A_05 at (315, 118)
Looking for faces in A_88.jpg
(480, 640, 3)
- Found A_88 at (366, 180)
Looking for faces in A_41.jpg
(480, 640, 3)
- Found A_41 at (407, 201)
Looking for faces in A_25.jpg
(480, 640, 3)
- Found A_25 at (315, 167)
Looking for faces in A_90.jpg
(480, 640, 3)
- Found A_90 at (340, 118)
Looking for faces in A_50.jpg
(480, 640, 3)
- Found A_50 at (390, 219)
Looking for faces in A_29.jpg
(480, 640, 3)
- Found A_29 at (340, 167)
Looking for faces in A_12.jpg
(480, 640, 3)
- Found A_12 at (319, 142)
Looking for faces in A_66.jpg
(480, 640, 3)
- Found A_66 at (345, 180)
Looking for faces in A_89.jpg
(480, 640, 3)
- Found A_89 at (345, 98)
Looking for faces in A_51.jpg
(480, 640, 3)
- Found A_51 at (315, 192)
Looking for faces in A_10.jpg
(480, 640, 3)
- Found A_10 at (260, 112)
Looking for faces in A_04.jpg
(480, 640, 3)
- Found A_06 at (365, 167)
Looking for faces in A_69.jpg
(480, 640, 3)
- Found A_69 at (345, 139)
Looking for faces in A_30.jpg
(480, 640, 3)
- Found A_30 at (319, 142)
Looking for faces in A_38.jpg
```

```
(480, 640, 3)
- Found A_38 at (324, 180)
Looking for faces in A_68.jpg
(480, 640, 3)
- Found A_68 at (315, 192)
Looking for faces in A_82.jpg
(480, 640, 3)
- Found A_82 at (340, 118)
Looking for faces in A_37.jpg
(480, 640, 3)
- Found A_37 at (345, 180)
Looking for faces in A_55.jpg
(480, 640, 3)
- Found A_55 at (340, 167)
Looking for faces in A_48.jpg
(480, 640, 3)
- Found A_48 at (340, 167)
Looking for faces in A_76.jpg
(480, 640, 3)
- Found A_76 at (345, 180)
Looking for faces in A_54.jpg
(480, 640, 3)
- Found A_53 at (386, 180)
Looking for faces in A_58.jpg
(480, 640, 3)
- Found A_58 at (324, 201)
Looking for faces in A_15.jpg
(480, 640, 3)
- Found A_15 at (319, 142)
Looking for faces in A_14.jpg
(480, 640, 3)
- Found A_14 at (390, 167)
Looking for faces in A_20.jpg
(480, 640, 3)
- Found A_20 at (390, 167)
Looking for faces in A_28.jpg
(480, 640, 3)
- Found A_28 at (414, 142)
Looking for faces in A_86.jpg
(480, 640, 3)
- Found A_86 at (365, 93)
Looking for faces in A_47.jpg
(480, 640, 3)
- Found A_47 at (340, 167)
Looking for faces in A_24.jpg
(480, 640, 3)
- Found A_24 at (319, 142)
Looking for faces in A_62.jpg
(480, 640, 3)
- Found A_62 at (340, 118)
Looking for faces in A_21.jpg
(480, 640, 3)
```

```
- Found A_21 at (319, 171)
Looking for faces in A_03.jpg
(480, 640, 3)
- Found A_03 at (216, 167)
Looking for faces in A_02.jpg
(480, 640, 3)
Looking for faces in A_84.jpg
(480, 640, 3)
- Found A_84 at (266, 142)
Looking for faces in A_09.jpg
(480, 640, 3)
- Found A_09 at (290, 112)
Looking for faces in A_23.jpg
(480, 640, 3)
- Found A_23 at (319, 142)
Looking for faces in A_11.jpg
(480, 640, 3)
- Found A_11 at (315, 142)
Looking for faces in A_77.jpg
(480, 640, 3)
- Found A_77 at (345, 160)
Looking for faces in A_32.jpg
(480, 640, 3)
- Found A_32 at (324, 139)
Looking for faces in A_74.jpg
(480, 640, 3)
- Found A_74 at (340, 167)
Looking for faces in A_59.jpg
(480, 640, 3)
- Found A_59 at (390, 142)
Looking for faces in A_19.jpg
(480, 640, 3)
- Found A_19 at (365, 142)
Looking for faces in A_42.jpg
(480, 640, 3)
- Found A_42 at (266, 118)
Looking for faces in A_78.jpg
(480, 640, 3)
- Found A_78 at (366, 118)
Looking for faces in A_57.jpg
(480, 640, 3)
- Found A_57 at (315, 93)
Looking for faces in A_83.jpg
(480, 640, 3)
- Found A_83 at (340, 142)
Looking for faces in A_56.jpg
(480, 640, 3)
- Found A_56 at (365, 167)
Looking for faces in A_06.jpg
(480, 640, 3)
- Found A_06 at (291, 167)
Looking for faces in A_39.jpg
```

```
(480, 640, 3)
- Found A_39 at (345, 180)
Looking for faces in A_75.jpg
(480, 640, 3)
- Found A_75 at (345, 180)
Looking for faces in A_87.jpg
(480, 640, 3)
- Found A_87 at (345, 160)
Looking for faces in A_64.jpg
(480, 640, 3)
- Found A_64 at (386, 222)
Looking for faces in A_65.jpg
(480, 640, 3)
- Found A_65 at (340, 192)
Looking for faces in A_40.jpg
(480, 640, 3)
- Found A_40 at (428, 180)
Looking for faces in A_85.jpg
(480, 640, 3)
- Found A_85 at (365, 142)
Looking for faces in A_17.jpg
(480, 640, 3)
- Found A_17 at (290, 142)
Looking for faces in A_72.jpg
(480, 640, 3)
- Found A_72 at (345, 201)
Looking for faces in A_79.jpg
(480, 640, 3)
- Found A_79 at (345, 118)
Looking for faces in A_46.jpg
(480, 640, 3)
Looking for faces in A_53.jpg
(480, 640, 3)
- Found A_53 at (340, 167)
Looking for faces in A_71.jpg
(480, 640, 3)
- Found A_71 at (390, 167)
Looking for faces in A_70.jpg
(480, 640, 3)
- Found A_70 at (315, 93)
Looking for faces in A_43.jpg
(480, 640, 3)
- Found A_43 at (366, 160)
Looking for faces in A_44.jpg
(480, 640, 3)
- Found A_44 at (340, 118)
Looking for faces in A_67.jpg
(480, 640, 3)
- Found A_67 at (291, 142)
Looking for faces in A_08.jpg
(480, 640, 3)
- Found A_08 at (315, 192)
```

```
Looking for faces in A_27.jpg
(480, 640, 3)
- Found A_27 at (266, 118)
Looking for faces in A_52.jpg
(480, 640, 3)
- Found A_52 at (315, 142)
Looking for faces in A_35.jpg
(480, 640, 3)
- Found A_35 at (266, 142)
Looking for faces in A_49.jpg
(480, 640, 3)
- Found A_49 at (304, 201)
Looking for faces in A_34.jpg
(480, 640, 3)
- Found A_34 at (340, 167)
Looking for faces in A_45.jpg
(480, 640, 3)
- Found A_45 at (390, 167)
Looking for faces in A_26.jpg
(480, 640, 3)
- Found A_26 at (315, 167)
Looking for faces in A_18.jpg
(480, 640, 3)
- Found A_18 at (315, 167)
Looking for faces in A_61.jpg
(480, 640, 3)
- Found A_61 at (386, 118)
Looking for faces in A_60.jpg
(480, 640, 3)
- Found A_60 at (315, 93)
Looking for faces in A_31.jpg
(480, 640, 3)
- Found A_31 at (315, 118)
Looking for faces in A_80.jpg
(480, 640, 3)
- Found A_80 at (345, 201)
Looking for faces in A_81.jpg
(480, 640, 3)
- Found A_81 at (366, 201)
Looking for faces in A_22.jpg
(480, 640, 3)
- Found A_22 at (266, 167)
Looking for faces in A_63.jpg
(480, 640, 3)
- Found A_63 at (340, 142)
Looking for faces in A_07.jpg
(480, 640, 3)
- Found A_07 at (390, 118)
Looking for faces in A_73.jpg
(480, 640, 3)
Looking for faces in A_13.jpg
(480, 640, 3)
```

```
- Found A_13 at (319, 142)
Looking for faces in A_16.jpg
(480, 640, 3)
- Found A_16 at (200, 112)
Looking for faces in A_36.jpg
(480, 640, 3)
- Found A_36 at (304, 118)
Looking for faces in A_01.jpg
(480, 640, 3)
- Found A_01 at (365, 167)

accuracy : 0.9438202247191011
```

This was the accuracy obtained when the level of distortion was $+45^0$ to -45^0 (90^0 in total).

5.2 Reflection :

The actual aim behind the project is to provide a model that can be trained and tested instantly using a single image for each subject and embedding that model into a web page so that the data collection and computation risks are reduced to a great extent.

Summary of the project can be given as,

- ➢ Field of application where machine learning has a huge role is chosen (image processing).
- ➢ A potential problem which has a practical application is chosen (Single Shot Facial Recognition).
- ➢ Dataset is analyzed and notable observations are noted such as balance among the classes etc.
- ➢ Algorithms that suit the dataset and problem statement are analyzed.
- ➢ A benchmark is set for the model to rely on.
- ➢ Dataset is preprocessed, the model was built to train on the preprocessed dataset using selected algorithms (Transfer Learning, Siamese networks, and KNN Classifier).
- ➢ The trained model is evaluated for its efficiency over an unseen test dataset.
- ➢ Several parameter tunings and architectures were tried to attain the best efficiency possible.
- ➢ A satisfiable solution is obtained by Siamese network and KNN Classifier with an accuracy of ~95%.

Most interesting step through this whole process was researching and analyzing several algorithms to find the best algorithm that suits the situation. I was exposed to many different and interesting networks such as “VGG-16”, “AlexNet” e.t.c which were very complex and trained on large datasets. But the whole idea of using transfer learning to get the required result was a huge disaster as it always ended up at an accuracy level of ~1-2% which is comparable to that of mere guessing.

Most difficult part of this project was finding the best approach that solves the problem left by the transfer learning approach. The immediate approach that struck my mind was to convert the image into a numerical format and look for the patterns in the images manually so that I can feed the pattern to the model manually. This was suicidal as this forces one to look at thousands of images and get the patterns which virtually impossible.

So the next immediate thought was to convert an image to a small dimensional vector but through some image processing model this time. This led me to search for algorithms which convert an image to a small dimensional vector where every image can be treated as a point and the whole problem reduces to find the class to which the new image belongs to. This finally comes down to be a graphical classification problem which can be solved efficiently using KNN Classifier.

After a long search, the algorithm that converts an image to vector is found as Siamese network which uses two sister networks that work on an image simultaneously to generate a 128-dimensional vector. Facial recognition module of python also has face_encodings and face_locations methods to achieve this functionality.

Final solution attained is a very good model and this solution fits the expectations for the problem I've chosen. This model can be used in a general setting to solve similar problems such as recognizing vehicles, animals e.t.c This can further extend to other fields such as recognizing handwriting or art style of a specific person or recognizing something that is virtually unique to any particular thing.

5.3 Improvement:

This model has attained a decent efficiency of ~95%. However, this can further be enhanced using more powerful hardware which can accommodate more complex architectures that can generate a higher dimensional vector, yet not too complex that they face a vanishing gradient problem, pushing the efficiency further.

Dataset can also be improved by adding more data points which populate the space constructed by encoded vectors. New algorithms are not necessarily required for this problem, as these algorithms do a decent classification in a considerably comfortable time when operated on a powerful device.

When this model is considered as a benchmark model, we end with a very little room to improve this further because the first benchmark was set with an efficiency of ~1-2% and has a lot of room to operate on, so that the model reaches a higher efficiency, but second benchmark has already reached an efficiency of 90% leaving us with a very little space to operate on. So, even this space can be utilized to push the model to further efficiency.