Age and Gender Prediction

Reporting

This project aims to build a model that predicts the age and gender of an individual based on their facial features. We achieve this using transfer learning from two advanced face recognition models, FaceNet and ArcFace. In this report, I detail how I collected and processed the data, our model architecture and training, and the results of our experiments. I also conclude with potential future directions for this project.

Composed by Tran Le Minh
25 April 2023

Table

Python Library Choices

Python Library

- Keras/Tensorflow: I aimed for code simplicity when building the model, and thus chose Keras over Pytorch.
- Matplotlib, Numpy, Pandas, Scipy: These are libraries that are commonly used in data preprocessing or analysis.
- OpenCV: I mainly working with OpenCV.
- Deepface: This is a crucial library as it provides essential tools for basic image processing and also offers pre-trained models for transfer learning

Dataset and Preprocessing Data

When I started working on the project, I had a few options for datasets that included age and gender details such as UTK, IMDBcrop, MegaAge-Asian,... . However, when it came to actually cleaning and preprocessing the data, I ended up only using the UTK and IMDBcrop datasets. The final model that was trained was mostly built using the UTK dataset.

IMDbcrop:

IMDbcrop dataset contain over 460000 images of celebrities, which were colected from IMDb website. Each piece of information for each image is contained in a .mat file in Matlab format and can be read and processed using Scipy to obtain the information. The format is as follows:

- **dob:** date of birth (Matlab serial date number)
- **photo_taken:** year when the photo was taken
- **full_path:** path to file
- **gender:** 0 for female and 1 for male, *NaN* if unknown
- **name:** name of the celebrity
- **face_location:** location of the face. To crop the face in Matlab run
- **face_score:** detector score (the higher the better). *Inf* implies that no face was found in the image and the *face_location* then just returns the entire image
- **second_face_score:** detector score of the face with the second highest score. This is useful to ignore images with more than one face. *second_face_score* is *NaN* if no second face was detected.
- **celeb_names (IMDB only):** list of all celebrity names
- **celeb_id (IMDB only):** index of celebrity name

When it comes to age and gender prediction, I only need to use the relevant information regarding age and gender, along with the cropped face images from the dataset. Therefore, I remove most of the other information and only keep the variables full_path, dob, gender, and photo_taken to calculate the age when the photo was taken. Once the age and gender details are obtained, data processing begins by removing any outliers of the age variable that fall outside the range of 0-100. Subsequently, the relevant data is loaded. It may seem peculiar that the variable 'full_path' is retained despite being redundant for model training. However, this variable serves the purpose of correctly loading the corresponding image. The path of removed image is removed during the removal of outlier data points, and therefore, 'full_path' remains necessary for correct image retrieval.

Due to limited computer resources, loading all 460,000 images at once is not feasible. To overcome this challenge, I utilized the k-fold cross-validation technique to train the model. This approach was appropriate given that the dataset is comprehensive and each partitioned subset is similar to one another. This consistency ensured that the model was stable across all the folds. That how load_image_data function work

UTKFace:

UTKFace dataset contains over 20000 face images in the wild. I only used the Aligned&Cropped Faces version because it much faster for preprocessing image. Information of each image is embedded in the file name, formated like `[age]_[gender]_[race]_[date&time].jpg`

- **[age]** is an integer from 0 to 116, indicating the age
- **[gender]** is either 0 (male) or 1 (female)
- **[race]** is an integer from 0 to 4, denoting White, Black, Asian, Indian, and Others (like Hispanic, Latino, Middle Eastern).
- **[date&time]** is in the format of yyyymmddHHMMSSFFF, showing the date and time an image was collected to UTKFace

Like the IMDbcrop dataset, I only require age and gender information from our dataset. Hence, I discard any extraneous data when loading the file and exclude age values exceeding 100 years. After loading the data, I noticed that the age distribution was uneven, particularly at age 0 and 25. To rectify this, I randomly excluded data points at age 0 and age 25. Therefore, the code for removing data points is executed twice to ensure that the data is not heavily skewed at any particular age.

Conclusion

As I progressed through the project, I realized that to use the model with a webcam or any arbitrary input image, I would need to use Mediapipe or Harrcascade XML to extract the facial region, making the UTK dataset more suitable as the images were already pre-processed. Extracting faces using Mediapipe with the IMDbcrop dataset proved challenging due to complex backgrounds, and the age distribution of celebrities may not be representative of the general population. However, the dataset still holds potential for training by adjusting the model parameters and finding an optimal crop. Ultimately, such considerations are necessary to ensure that the model performs well in real-world testing.

Approaches to Model Building and Other Methods

This section covers two models that I used for transfer learning, namely ArcFace and FaceNet. I will also discuss my approaches towards these models, as well as other approaches that I tried but were unsuccessful for my particular case.

FaceNet:

Based on the official articles of FaceNet, it’s a deep convolutional neural network (CNN) model based on InceptionResNetV1 used for face recognition, which was introduced by Google researchers in 2015. Its architecture consists of convolutional and pooling layers followed by fully connected layers that map the input image to a 128-feature embedding.

During training, FaceNet learns to optimize the embedding by minimizing the distance between embeddings of images of the same person, and the opposite for different person. This is done through a triplet loss function that takes three images as input: an anchor image, a positive image of the same person, and a negative image of a different person. FaceNet outputs a vector of 128-feature embedding. This embedding represents a compact and discriminative representation of the facial features of the person in the image. The distance between two embeddings can be used as a measure of similarity between the two faces - smaller distances indicate that the faces are more similar, while larger distances indicate that the faces are more dissimilar. FaceNet's output can be used for various tasks such as face verification, face identification, and face clustering.

Due to the challenges in obtaining FaceNet, developed by Google researchers, so I used a substitute version from the DeepFace library. After loading the model and freezing specific layer as well as remove output layer of pretrained model, I had two approaches:

- Using the Conv2D layer followed by a Flatten and Activation layer as the output layer.
- Using the Dense layer as the output layer.

The main difference between the first and the second method is that the Conv layer will learn local patterns in the input feature maps whereas the Dense layer will learn global patterns. In other words, the Dense layer will be able to recognize more abstract correlations between features, and the Conv layer would be able to detect features in the input images that are spatially close to each other.

ArcFace:

As ArcFace's official documentation, it’s also a CNN model based on ResNet-50 or ResNet-100 use for face recognition task, developed by Jiankang Deng, Jia Guo, and Stefanos Zafeiriou in 2018.

While training, ArcFace also learns to optimize the embedding by minimizing the distance between embeddings of images of the same person, and maximizing the distance for different persons. The difference lies in its method, as the model is optimized using an angular margin-based loss function that encourages the learning of discriminative features that are well-separated for different classes. The output of the model is a 512-feature embedding vector, which operates similarly to the FaceNet model but has significantly more features than FaceNet.

The remaining part of the model building process is similar to FaceNet in terms of method approaches and model processing techniques, such as freezing and adding or removing layers.

Model Compiling and Conclusion:

The model has two outputs: age and gender. For age prediction, we used the weighted average pooling method which can avoid age bias in skewed data and handle uncertainty in cases where the model has multiple possible answers. For gender, since it only has two possible values, we used the argmax method. Therefore, I created a function that defines a metric for the model to use during training for age prediction, using the weighted average pooling method and for gender, I used the accuracy metric. Both I chose to use categorical_crossentropy as the loss function for simplicity.

For both pretrained models, the input is normalized. Therefore, I added a new normalization input layer on top of each model. For FaceNet, the image is normalized by dividing it by 127.5 and then subtracting 1, while for ArcFace, the image is normalized by subtracting 127.5 and then dividing it by

128. Although they appear to have the same normalization, after testing, I discovered that they must be kept as is, as using the wrong normalization can lead to suboptimal results.

Before training, I determined that both ArcFace and FaceNet models would perform well for the age and gender prediction task due to their suitability as face models. Both pretrained models have learned features on human faces that are well-suited for transfer learning. While both models are primarily used for face recognition and identification tasks, ArcFace may be more suitable for the current task because ArcFace's learning method can result in a more compact and well-separated distribution of the embeddings in the feature space. While FaceNet's triplet loss function can result in a more spread-out distribution of the embeddings in the feature space. However, in Age and Gender prediction task, a well-separated distribution of the embeddings in the feature space is highly demanded, thus ArcFace would perform significantly better.

Other Methods:

Before exploring FaceNet and ArcFace, I experimented with several pretrained models that were readily available online, including the EfficientNetV2S from InsightFace_Keras and the Keras Applications library (I modified the stride of the first convolutional layer to preserve as much information as possible from the input image, which was relatively small), OpenFace from DeepFace library and MobileFaceNet.

Although they were originally designed for face-related tasks, those models and their pre-trained weights did not perform as well as ArcFace and FaceNet. Therefore, in order to fully utilize them, I would need to retrain the entire model after loading their weights. However, due to limited computer resources, I am unable to fully train them in this way.

Tranning:

I trained the model for 200 epochs without using a batch size due to limited computer resources. It may sound illogical, but without using batch_size, I was able to train the model than when I used it (the latter resulted in errors after only a few epochs). To control the training process, I also used early stopping, which monitored the age_output_mae rather than using validation data. Although this approach may lead to overfitting, but due to the inability to use batch_size, I have implemented my own solution to address this issue. I reloaded the data and split it multiple times after training once. I didn't use random_state in train_test_split, so the data would be shuffled without following any specific pattern. This ensured an even distribution of data during training, and the model won't be overfitted.