A Parallel and Distributed Facial Recognition System

Kamel Gerado
Department of Computer Science
Umm Al-Qura University
Makkah, Saudi Arabia
s437036583@st.uqu.edu.sa

I. Introduction and motivation

In recent years, we have witnessed the success of deep learning across multiple domains including facial recognition application. But we have also seen that due to the large size and computational complexities of the models and data, the performance of the deep learning procedures is reduced.

To overcome these challenges, many research has been conducted to improve the performance of these models, parallel and distributed deep learning approaches have been introduced and they are a staple of modern applications. We need to leverage multiple cores or multiple machines to speed up applications or to run them at a large scale.

In this project, we adopt the Ray's architecture, a general-purpose cluster-computing framework that enables simulation, training, and serving for AI applications. The requirements of these workloads range from lightweight and stateless computations, such as for simulation, to long running and stateful computations, such as for training and inference.

II. Related Work

The parallel and distributed processing approach has been used by the computer vision community for fast and accurate face detection applications. Dalia and Salma [2] used MPI for Face Recognition, which used principal component analysis (PCA) algorithm and propose two different parallel architectures to accelerate training and testing phases, by exploiting the benefits of distributed memory architecture.

Another advanced system designed by Abhishek and Shahzad [4] describes how to accelerate a real-world face detection and tracking system by taking advantage of the multiple processing cores that are present in most modern CPUs.

Several deep learning models for facial recognition have been proposed which have tools like OpenMPI, Python multiprocessing, and ZeroMQ, which provide low-level primitives for sending and receiving messages. These tools are very powerful, but they provide a different abstraction and so single-threaded applications must be rewritten from scratch to use them.

On the other end of the spectrum, we have domain-specific tools like TensorFlow for model training, Spark for data processing and SQL, and Flink for stream processing. These tools provide higher-level abstractions like neural networks, datasets, and streams. However, because they differ from the abstractions used for serial programming, applications again must be rewritten from scratch to leverage them.

III. Project Description

In order to develop the system, we use Python 3.9 with OpenCV 4.5 [1] and Ray 1.13 [3]. We will be using OpenCV's highly improved "convolution neural networks" (cnn) module to extract faces from the images and dlib library [4] to construct our face embeddings. These embeddings will be saved to create the database of faces from which the face recognition system will later try to match and predict the identity of the person. In real time, for any faces which are recoginzed from the video feed, facial embeddings will be calculated and it will be matched with all the facial embeddings in the dataset. This is a searching problem where the elements don't have an order. So, we have to search linearly. This will take huge amount of time. We take this opportunity to parallelize the searching process using Ray remote functions (tasks).

The face detection model and the facial embedding used in this approach are inherently very fast and accurate but can slow down considerably when face matching has to be done on a large database. This is because it is a searching problem where distance of a facial embedding has to calculated with respect to all of the entries in the previously created database. Here we have an opportunity to speed up the process. Instead of comparing the distance with the threshold distance linearly (or sequentially), we can distribute the search area within the database among number of nodes. When we employ multiple nodes to search in different parts of the database in parallel, we can find the required result faster.

IV. Reflection

Parallel programming is much more complex and difficult than sequential programming, and it is therefore more challenging to develop a parallel applications with the same software quality in a sequential context, but after reading about the cluster-computing framework Ray and experimenting some of its features, I was able to develop a parallel application without major modifications and complexity.

I was amazed by the capabilities of the Ray framework in terms of production, with a rich set of libraries and integrations

I can easily deploy my machine learning models and turn them into products.

V. Conclusion

A robust solution was developed for the large scale facial recognition problem. We were able to model a solution that could process, retrieve and compare facial embeddings. Due to our usage of the cluster computing, the algorithm works more efficiently, without sacrificing any accuracy. It is able to do so, even when using a database containing thousands of facial embeddings, by using Ray remote task. This makes the system quite fast and allows it to be highly scalable.

References

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] Dalia Ibrahim and Salma Hamdy Elsayed. Parallel architecture for face recognition using mpi. *International Journal of Advanced Computer Science and Applications*, 8, 01 2017.

[3] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 561–577, 2018.

[4] Abhishek Ranjan and Shahzad Malik. Parallelizing a face detection and tracking system for multi-core processors. In *2012 Ninth Conference on Computer and Robot Vision*, pages 290–297, 2012.