# Paralleled Facial Image Recognition

# Based On CNN Deep Learning

**Team 09**

**Team member: Yueheng Li**

[1.Introduction 1](#_Toc102249618)

[1.1 Background 1](#_Toc102249619)

[1.2 Motivation 2](#_Toc102249620)

[1.3 Goals 2](#_Toc102249621)

[2. Methodology: 2](#_Toc102249622)

[2.1 Model implementation 2](#_Toc102249623)

[2.2 Parallelism Technology 3](#_Toc102249624)

[2.2.1 Data Parallelism: 3](#_Toc102249625)

[2.3 Experimental steps: 4](#_Toc102249626)

[3.Dataset Description: 5](#_Toc102249627)

[3.1 Data Source: 5](#_Toc102249628)

[4. Experiment and result 6](#_Toc102249629)

[4.1 Experiment Environment： 6](#_Toc102249630)

[4.2 Exploratory Data Analysis of the dataset: 6](#_Toc102249631)

[4.3 Model Construction: CNN model 8](#_Toc102249632)

[4.4 Model Performance analysis 9](#_Toc102249633)

[4.5 Make predictions 9](#_Toc102249634)

[4.6 Data Parallel: Using Dask DataFrame 9](#_Toc102249635)

[4.7 Model Parallel:(Single GPU) 10](#_Toc102249636)

[4.8 Data Parallel:(Multi-GPU) 10](#_Toc102249637)

[5.Conclusion 12](#_Toc102249638)

[6.Reference 13](#_Toc102249639)

## 1.Introduction

### 1.1 Background

Image recognition is one of the many applications of Machine Learning, it can solve problems for security purposes, object detection, face detection, healthcare, entertainment, among others. This application has an enormous potential to help our society, so it is important to find new uses for this tool, improve the current methods and get more accurate and useful insights from it.

卡通人物

描述已自动生成

And Facial Emotion Recognition (FER),as a branch of Image recognition,is the technology that analyses facial expressions from both static images and videos in order to reveal information on one’s emotional state.

### 1.2 Motivation

In most situation, humans are easy to classify human gender or even facial emotions,

but it's hard to describe exactly why we can make such a decision. In the absence of defined features, this distinction is very difficult for traditional machine learning methods. Also, task-related features are not expressed in exactly the same way every time, and everyone looks a little different. Deep learning algorithms provide a way to process information without pre-defined features and make accurate predictions in the presence of changes in how features are expressed.

As the entry-level learners of deep learning, I have a strong interest in facial expression recognition, and try to use CNN model for research and practice to recognize and classify different facial images.

In this project, I will build a Machine Learning Algorithm using CNN to predict from a giving picture if the celebrity is smiling or not. And then, by using the parallelism method, for both data parallelism and model parallelism, I will compare execution runtimes for serial and parallel to show speed up. And then make some conclusions.

### 1.3 Goals

1.Learn how to handle large scale dataset properly

2.Re-implement the process to make image classification based on Deep Learning

3.Do experiments on different platforms by using different parallelism methods

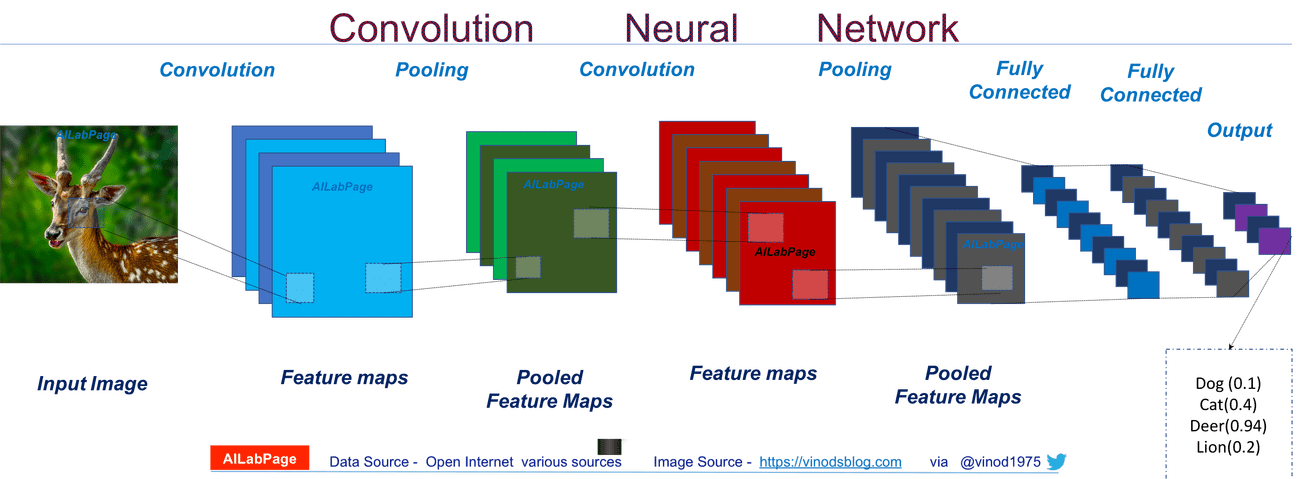
4.Compare different parallel methods and make brief conclusions

## 2. Methodology:

### 2.1 Model implementation

**Convolutional Neural Network, also known as CNN**, is a well-known method in computer vision applications. It is a class of deep neural networks that are used to analyze visual imagery. This type of architecture is dominant to recognize objects from a picture or video. It is used in applications like image or video recognition, neural language processing, etc.

Basic CNN consists of three structures: convolution, activation, pooling. The output of CNN is the specific feature space of each image. When processing the image classification task, we will take the characteristic space output by CNN as the input of fully connected neural network (FCN) and use the fully connected layer to complete the mapping from the input image to the tag set, that is, the classification. The training Error of CNN network needs to be measured by an objective function. At present, the popular objective function is Mean Square Error and K-L divergence. It is easy to judge the Error formula of the output layer. Our model is derived from THE CNN model, eliminating markup errors and changing the CNN model to improve the accuracy of the model.



In this project, I will use keras in python to implement CNN. The labels of the dataset could be set in **“SMILING” and “NOT SMILING”.** Then I will translate the labels and the pictures into the form that can be used to train the model and test it. Finally get the prediction and evaluation of our model.

### **2.2 Parallelism Technology**

### 2.2.1 Data Parallelism:

**1. Dask Parallelism:**

**Dask** is a parallel computation framework that has seamless integration with Jupyter notebook. Originally, it was built to overcome the storage limitations of a single machine and extend the computation capability of Pandas, Numpy, and Scit-kit Learn with DASK equivalents, but soon it found its use as a generic distributed system.



#### 2.2.2 Model Parallelism:

**1. Tensorflow Distributed Computing API**

**tf.distribute.Strategy** is a TensorFlow API to distribute training across multiple GPUs, multiple machines, or TPUs. Using this API, you can distribute your existing models and training code with minimal code changes. And it has been designed with these key goals in mind:

* Easy to use and support multiple user segments, including researchers, machine learning engineers, etc.
* Provide good performance out of the box.
* Easy switching between strategies.

**1.tf.distribute.OneDeviceStrategy(One GPU parallel)**

This is a strategy to place all variables and computation on a single specified device.

**2.tf.distribute.MirroredStrategy(Multi-GPU parallel)**

It supports synchronous distributed training on multiple GPUs on one machine. It creates one replica per GPU device. Each variable in the model is mirrored across all the replicas. Together, these variables form a single conceptual variable called MirroredVariable. These variables are kept in sync with each other by applying identical updates.

2.3 Experimental steps:

**Step 1: Data Exploration Analysis**

**Step 2: Load image data attributes in Dask format(Parallelism)**

**Step 3: Data Pre-Processing(Local machine)**

**Step 4: Split Dataset into Training, Validation and Test set**

**Step 5: Pre-processing Images: Data Augmentation**

**Step 6: Build up CNN deep learning model**

**Step 7: Explore different parallel method(Parallelism)**

**Step 8: Implement the model to do some predictions**

**Step 9:** **Plot execution run times for serial and parallel parts to show speed up**

**Step 10: Comparison and Conclusion**

**(Most of the procedures are conducted on OOD and Discovery)**

## 3.Dataset Description:

**CelebFaces Attributes Dataset (CelebA)(>1GB)** is a large-scale face attributes dataset with more than **200K** celebrity images, each with **40** attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including:

* **10,177** number of **identities**,
* **202,599** number of **face images**
* **5 landmark locations**, **40 binary attributes** annotations per image.

The dataset can be employed as the training and test sets for the following computer vision tasks: face attribute recognition, face recognition, face detection, landmark (or facial part) localization, and face editing & synthesis.



Figure: Each image has different attribute annotations

A popular component of computer vision and deep learning revolves around identifying faces for various applications from logging into your phone with your face or searching through surveillance images for a particular suspect.

This dataset is great for training and testing models for **face detection**, particularly for recognizing facial attributes such as finding people with brown hair, are smiling, or wearing glasses. Images cover large pose variations, background clutter, diverse people, supported by a large quantity of images and rich annotations. This data was originally collected by researchers at **MMLAB, The Chinese University of Hong Kong(CUHK).**

### 3.1 Data Source:

**Official Website:**

<http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>

**Kaggle Link:**

<https://www.kaggle.com/datasets/jessicali9530/celeba-dataset?select=list_landmarks_align_celeba.csv>

## Experiment and results

### 4.1 Experiment Environment：

1. The local configuration of my PC：

**CPU: Intel i7-10750H GPU:NVIDIA RTX2060 RAM:32GB**

**CUDA 11.4 + Tensorflow 2.5.0**

1. OOD：

**2x Core CPU + K80 GPU +8G RAM Using Tensorflow-GPU**

**CUDA 11.4 + Tensorflow 2.8.0 + anaconda2022/01**

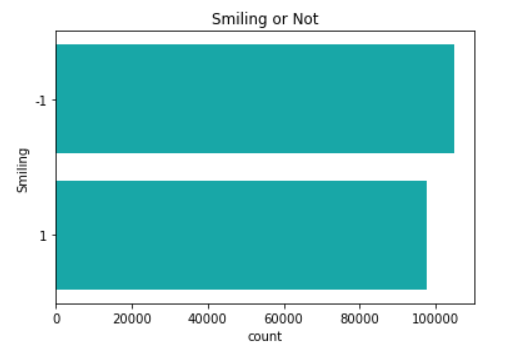
1. Discovery：

Most experiment conducted on **GPU Node 1013**：

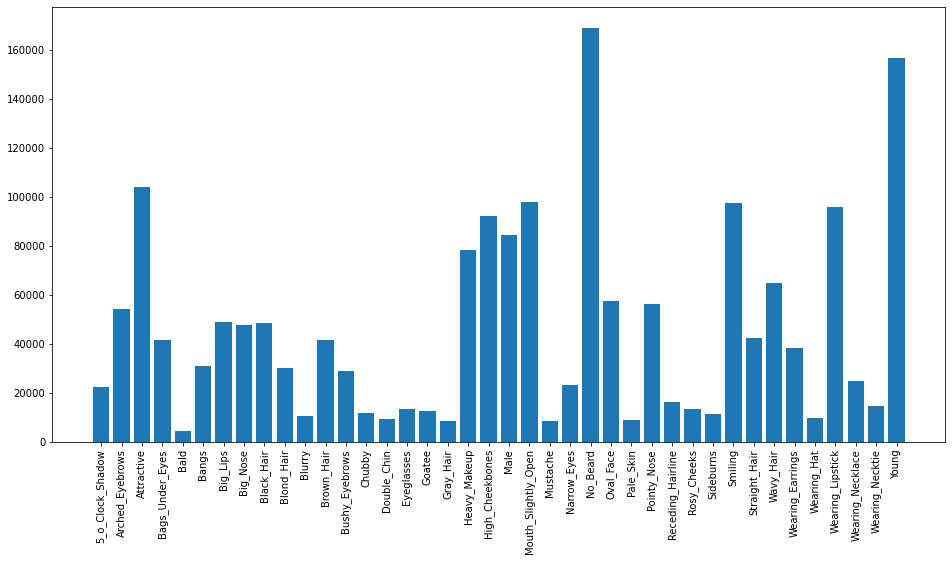
**2x Tesla V100-SXM2-32GB Tensorflow-GPU installed**

**CUDA 11.4 + Tensorflow 2.8.0 +** **anaconda2022/01**

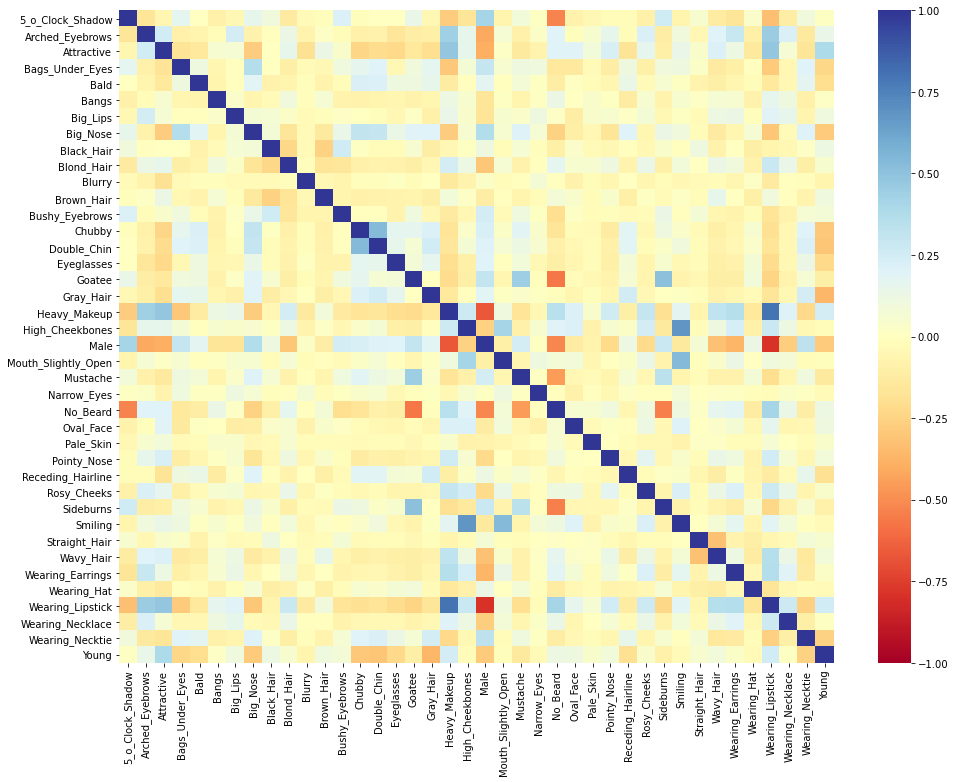
### 4.2 Exploratory Data Analysis of the dataset:



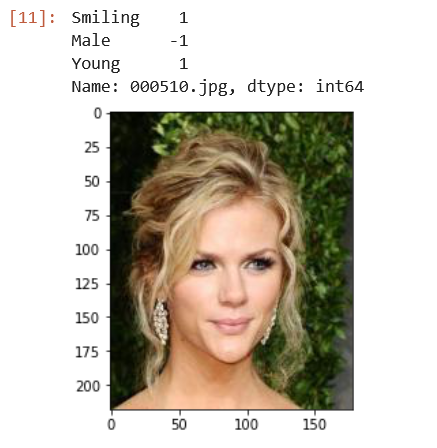
*Figure: The number of smiling images*



*Figure: The number of each attribute(feature)*

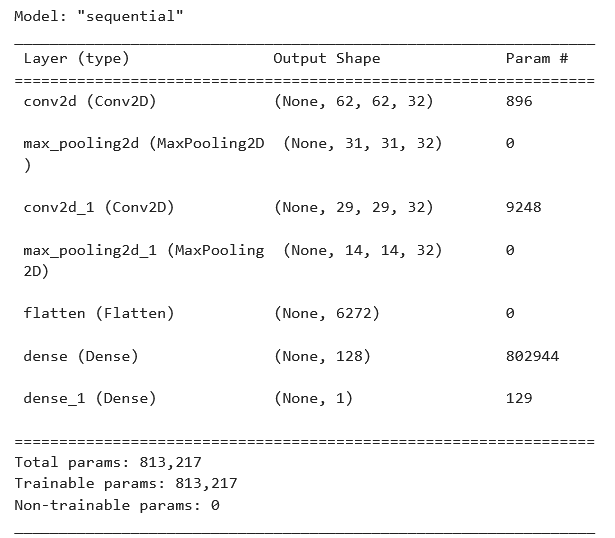


*Figure: The correlation between each attribute(feature)*



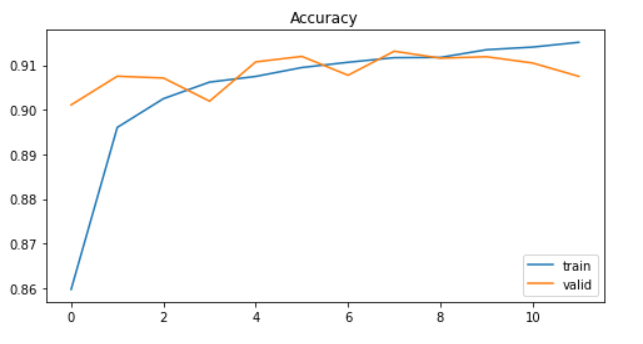
*Figure: Sample images*

### 4.3 Model Construction: CNN model



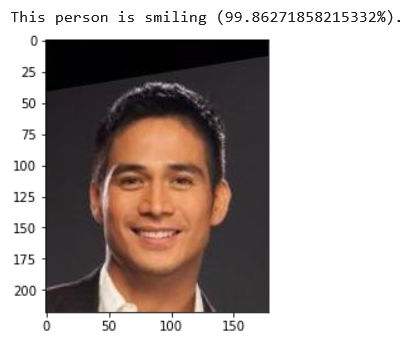
*Figure: CNN model*

### 4.4 Model Performance analysis



*Figure: The train and valid accuracy*

### 4.5 Make predictions



*Figure: The model can figure if a person is smiling or not*

### 4.6 Data Parallel: Using Dask DataFrame



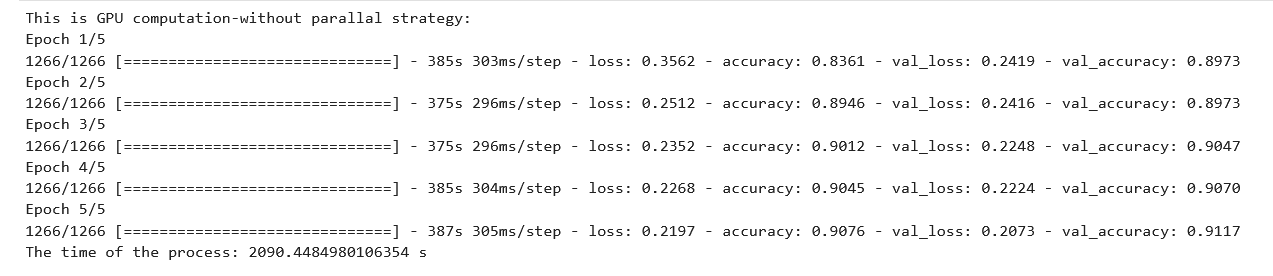


The normal pandas Dataframe takes 0.53s to load the attribute data, while the process in dask array takes only 0.034s ,which is much faster!

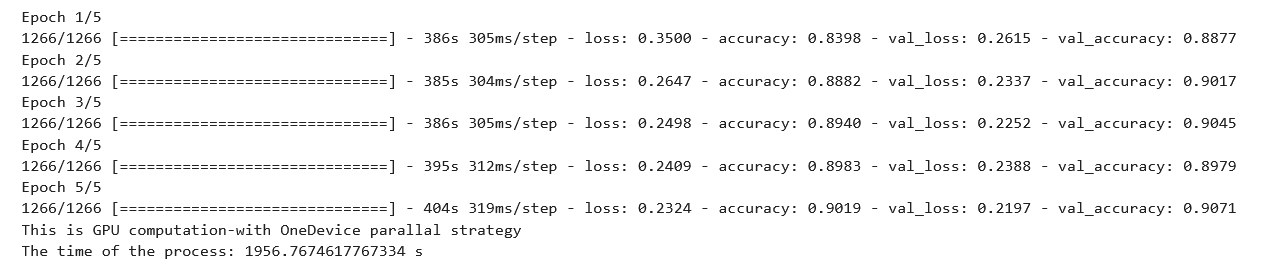
### 4.7 Model Parallel:(Single GPU)

Using different Tensorflow distribute computing strategies on one GPU. Here we can see a rapid speed-up by using different model parallel strategies

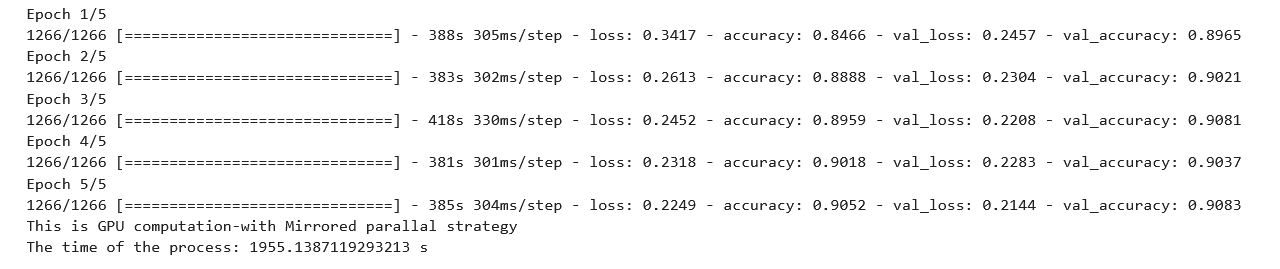
1. This is time of GPU computation without parallel strategy



1. This is time of GPU computation with parallel strategy (One device Strategy)



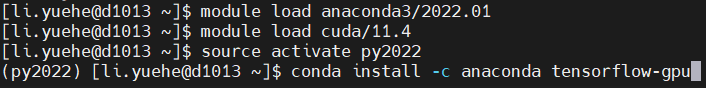
1. This is time of GPU computation with parallel strategy (Mirrored Strategy)



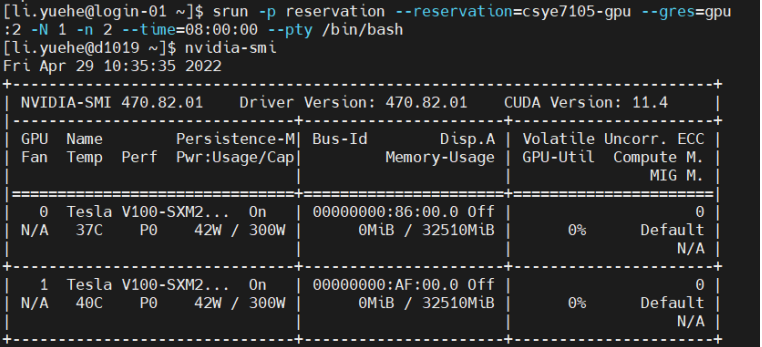
*Figure: Elapsed time for different parallel strategies*

### 4.8 Data Parallel:(Multi-GPU)

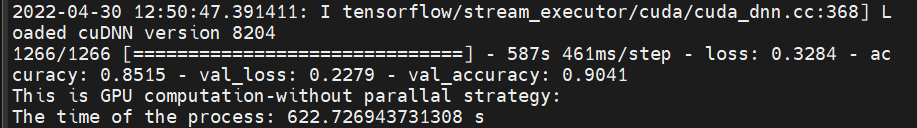
**Environment configuration**



*Figure:* ***Anaconda3/2022.01+cuda/11.4+tensorflow-gpu***



*Figure:* ***2x Tesla V100-SXM2-32GB***

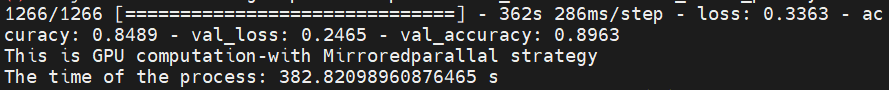


*Figure: Elapsed time of* ***none-parallel GPU******computation(K80)***

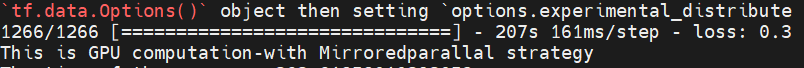
文本

描述已自动生成

*Figure: Elapsed time of* ***One device-parallel GPU******computation(K80)***



*Figure: Elapsed time of* ***Mirrored-parallel GPU******computation(K80)***



*Figure: Elapsed time of* ***Mirrored-parallel GPU******computation(V100)***

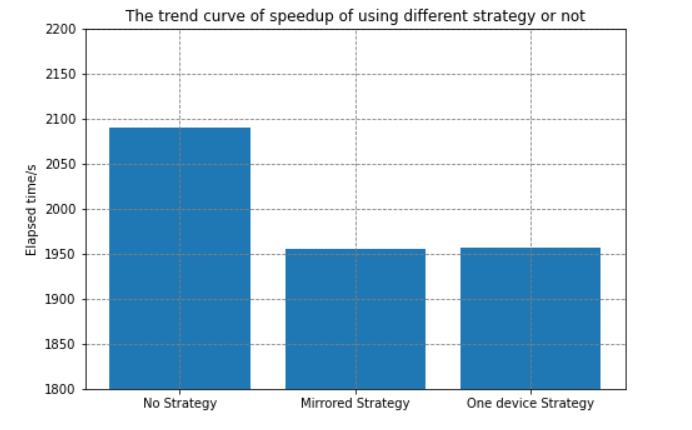


*Figure: Elapsed time of* ***Mirrored-parallel GPU******computation on two GPU(V100)***

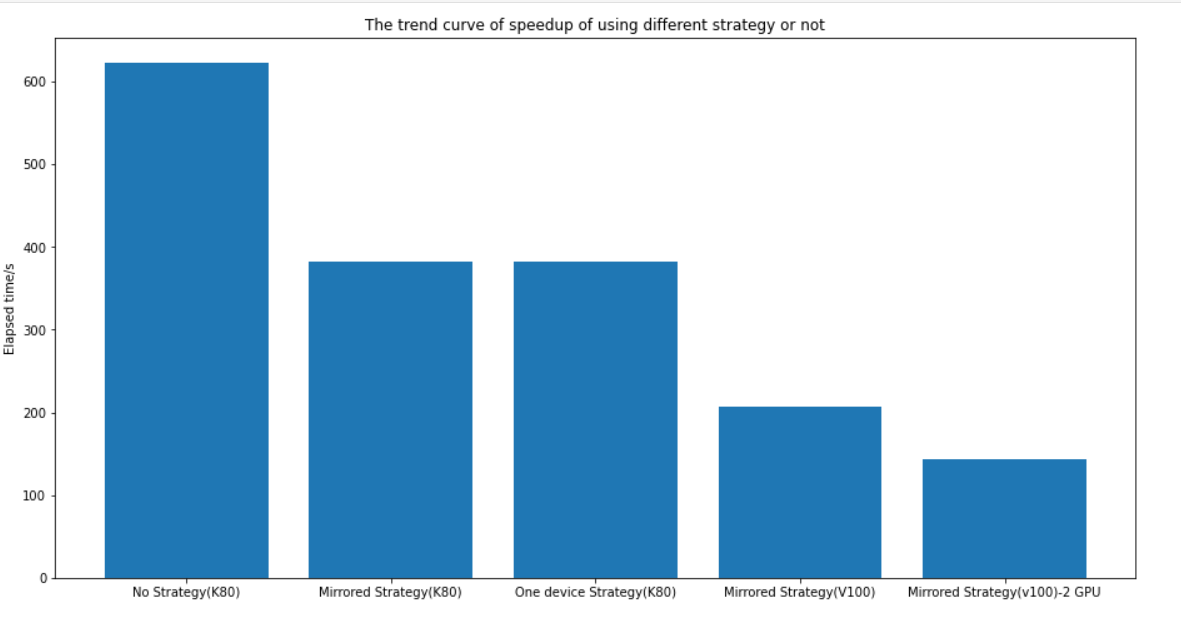
## 5.Conclusion

According to the experiments that I have conducted. We have the following results:

For the Model Parallelism, One device distributed strategy(One device parallel) and Mirrored distributed strategy(Multiple-device parallel) have significant speed up comparing with that don’t have any parallelism structure.



*Figure: Elapsed time for different parallel strategies for 5 epochs*



*Figure: Elapsed time for all experiments in 1 epoch*

After comparing all the experiment results, we come out the conclusion that:

* **Both Data Parallelism and Model Parallelism can help gain speed up in parallel computing**
* **Device computation power can hugely influence the computation speed.**
* **The device(CPU or GPU) number can affect the computation speed.**

## 6.Reference

1.Shashvat, G. (2020, July 18). Parallel processing in python using DASK - the Startup medium. TheStartup.https://medium.com/swlh/parallel-processing-in-python-using-dask

2.Working with GPUs — northeastern university research computing 2.0.0 documentation. (n.d.). Northeastern.Edu. Retrieved April 30, 2022, from <https://rc-docs.northeastern.edu/en/latest/using-discovery/workingwithgpu.html>

3.CelebA Dataset. (n.d.). Edu.Hk. Retrieved May 1, 2022, from <http://mmlab.ie.cuhk.edu.hk/>

projects/CelebA.html

4.Mandal, M. (2021, May 1). CNN for deep learning. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/>

5.Karagiannakos, S. (2020, October 22). Distributed deep learning training: Model and data parallelism in tensorflow. AI Summer. <https://theaisummer.com/distributed-training/>

6.Keras Team. (n.d.). Multi-GPU and distributed training. Keras.Io. Retrieved May 1, 2022, from <https://keras.io/guides/distributed_training/>

7.TensorFlow 1.0. (n.d.). Effectivemachinelearning.Com. Retrieved May 1, 2022, from <https://effectivemachinelearning.com/TensorFlow_1.0/10._Multi-GPU_processing_with_data_parallelism>

8.CNRS. (n.d.). IDRIS - TensorFlow: Multi-GPU and multi-node data parallelism. Idris.fr. Retrieved May 1, 2022, from <http://www.idris.fr/eng/jean-zay/gpu/jean-zay-gpu-tf-multi-eng.html>