INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

DATA MODELLING LAB I

MINI PROJECT

GPU-Accelerated Fast Fourier Transform for Blur Detection with Logistic Regression

KOUSHIK LAYEK

SC23M007

December 11, 2023

ABSTRACT

This project delves into the realm of image processing with a focus on efficient blur detection—a critical aspect in various fields such as photography, medical imaging, and remote sensing. Leveraging the power of GPU-accelerated Fast Fourier Transform (FFT) for feature extraction and Logistic Regression for classification, the project aims to achieve accurate and swift blur detection. The objectives include understanding

the dependencies of blur features, integrating GPU-accelerated FFT and Logistic Regression for robust classification, and applying the trained model to new images for real-world applicability. The project seeks to contribute to the image processing field by showcasing the effectiveness of GPU-accelerated techniques, potentially offering a faster and more precise alternative to existing methods. Through a comprehensive

literature review, the project explores existing methods for blur detection, discusses various approaches, and justifies the selection of GPU-accelerated FFT and Logistic Regression. The evaluation involves testing the model on a diverse data set, analyzing performance metrics, and presenting visual results. In conclusion,

the project highlights the significance of efficient blur detection and demonstrates the efficacy of GPU-accelerated techniques in advancing this domain. The findings contribute valuable insights to the image processing community, paving the way for faster and more accurate blur detection in practical applications.

CONTENTS

1. INTRODUCTION
2. PROJECT OBJECTIVES
3. LITERATURE REVIEW
4. SYSTEM ENVIRONMENT
5. DATA ACQUISITION AND PREPROCESSING
6. FFT AND IT'S CODING IMPLEMENTATION
7. LOGISTIC REGRESSION
8. OUTPUT
9. PERFORMANCE MEASURE
10. CONCLUSION
11. REFERENCES

1 INTRODUCTION

1.1 Brief overview of blur detection and its importance:

Blur detection is a crucial aspect of image processing that involves identifying and distinguishing between clear and blurry regions within an image. The importance of blur detection is multi-faceted, extending across various fields and applications.

- **Photography and Aesthetics:** In photography, the quality of captured images is paramount. Detecting and mitigating blur ensures that photos maintain sharpness and clarity, enhancing the overall visual appeal.
- **Medical Imaging:** In medical diagnostics, the accuracy of imaging is vital for proper diagnoses. Detecting and eliminating blur ensures that medical professionals can rely on clear and precise images for accurate assessments.
- **Remote Sensing:** In remote sensing applications, such as satellite imagery, blur detection is crucial for extracting accurate information. Clear images are necessary for tasks like environmental monitoring and mapping.

1.2 Overview of GPU Acceleration,FFT,Logistic Regression

- **Introduction to GPU Acceleration:** GPUs (Graphics Processing Units) are powerful processors designed for parallel processing, making them well-suited for tasks like image processing. They can significantly speed up computations compared to traditional CPUs.
- **Introduction to Fast Fourier Transform (FFT):** FFT is a mathematical algorithm to do the calculation efficiently for Discrete Fourier Transform, widely used for transforming signals, including images. In the context of your project, FFT is likely used for efficient frequency domain analysis, which can be beneficial for blur detection.
- **Introduction to Logistic Regression:** Logistic Regression is commonly used for binary classification tasks, making it suitable for distinguishing between clear and blurry images in your project.

1.3 Advantages of the Combined Approach:

Actually FFT algorithm need Lot of parallel calculation,so it will be computationally efficient to use GPU. Also the ability to handle larger data sets more efficiently compared to traditional methods.

2 PROJECT OBJECTIVES

2.1 Efficient Blur Detection:

First our goal is to find the dependency of blur and collect those are as a features of the data sets for logistic regression.

2.2 Integration of GPU-Accelerated FFT and Logistic Regression:

Combine the power of GPU-accelerated FFT for feature extraction with Logistic Regression for classification to achieve accurate and fast blur detection.

2.3 Application to New Images:

Successfully apply the trained model to new images, providing predictions on whether the input images are blurry or not.

2.4 Contribution to the Field:

Contribute to the field of image processing by demonstrating the effectiveness of GPU-accelerated FFT and Logistic Regression for blur detection, potentially offering a faster and more accurate alternative to existing methods.

3 LITERATURE REVIEW

3.1 Review of Existing Method for Blur Detection in Images:

There are lot of method for detecting blur like -by choosing the optimal threshold ,Convolution Neural network,Gradient-Based Methods,Frequency Domain Analysis also there are some machine learning approach also like Support vector Machine,Random Forests.

3.2 Justification for Choosing GPU-Accelerated FFT and Logistic Regression:

The justification could revolve around the parallel processing capabilities of GPUs, which significantly enhance the efficiency of FFT operations. Logistic Regression, known for its simplicity and effectiveness in binary classification, is justified as a suitable choice for distinguishing between clear and blurry images.And I was Choosing FFT because FFT converts images to its frequency domain from the time domain . So we can easily read the structure of the image like high frequency component,low frequency component etc ,and for my projects blur is totally depends upon high frequency component.

4 SYSTEM ENVIRONMENT

My code uses GPU acceleration for Fast Fourier Transform (FFT) operations and other GPU-accelerated tasks.So I have to create a Gpu environment ,and I did it in google colab by installing the RAPIDS library.That is utilized for GPU acceleration.The installation script downloads and installs RAPIDS dependencies.

'nvidia-smi' provides information about the GPU(S) available on the system.

```
!nvidia-smi
```

```
Tue Nov 21 18:02:12 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   42C    P8     9W /  70W |      0MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

5 Data Acquisition and Preprocessing

5.1 Data Download:

Start by mounting Google Drive and downloading the dataset using the Google Drive API.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
# Install the necessary library
!pip install gdown

# Define the file ID and destination file path
file_id = '1RObmCDPeQ1Lg-V6u7dT02Pf0qH-QMcTp'

# Set the destination file path
destination_path = '/content/blur_dataset.zip.ext'


# Download the file
import gdown
gdown.download(f'https://drive.google.com/uc?id={file_id}', destination_path,quiet=False)
```

5.2 Data Extraction:

Extract the contents of the downloaded zip file to the desired extraction directory.

```
import zipfile
# Define the path to the ZIP file and the extraction directory
zip_file_path = '/content/blur_dataset.zip.ext'
extraction_dir = '/content/blur_dataset'  # Change this to the desired extraction directory

# Unzip the file
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extraction_dir)
```

5.3 Data Exploration and Data Representation:

Explore the structure of the dataset, including the directories containing blurred and sharp images and Represent the data using Pandas DataFrames, creating a column for image paths and a target column indicating blurred or sharp.

```
import cv2
import pandas as pd
import os
import cupy as cp
import matplotlib.pyplot as plt

# Define the directory containing the images
image_directory_blurred = '/content/blur_dataset/blur_dataset/defocused_blurred/'
image_directory_sharp = '/content/blur_dataset/blur_dataset/sharp/'

# Get a list of all files in the directory
image_files_blurred = os.listdir(image_directory_blurred)
image_files_sharp = os.listdir(image_directory_sharp)

# Create a DataFrame with image paths and add directory path to each file name
data0 = pd.DataFrame({'Image_Path': [image_directory_blurred + file for file in image_files_blurred]})
data1 = pd.DataFrame({'Image_Path': [image_directory_sharp + file for file in image_files_sharp]})

# Add a 'Target' column with a value of 0 for each image
data0['Target'] = 0
data1['Target'] = 1

# Display the DataFrame
print(data0.head())
print(data1.head())

# Concatenate the two DataFrames into a single DataFrame
data = pd.concat([data0, data1], ignore_index=True)

# Ground truth labels
ground_truth = data['Target'].tolist()
```

5.4 Data Loading and Shuffling

Shuffle the data and select a subset for faster processing during development.Load the images into NumPy arrays, converting them to Cupy arrays for GPU acceleration.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import cupyx.scipy.fftpack as cufft
import numpy as np

# Shuffle the data
data_shuffled = data.sample(frac=1, random_state=42)  # Shuffle the data

# Select a subset of the shuffled data
subset_size = 350
subset_data = data_shuffled.head(subset_size)
# Convert to 'data' is in DataFrame
image_path_list =subset_data['Image_Path'].tolist()
Ground_truth = subset_data['Target'].tolist()

# Load images into NumPy arrays
images = np.asarray([cp.asnumpy(cp.array(cv2.imread(path, cv2.IMREAD_GRAYSCALE), dtype=cp.uint8)) for path in image_path_list])
```

6 FFT AND IT'S CODING IMPLEMENTATION

The Fast Fourier Transform (FFT) is an algorithm used to efficiently compute the Discrete Fourier Transform (DFT) and its inverse.The twiddle factors help simplify the computation of complex exponentails in the FFT algorithm, leading to a more efficient implementation. They are a key component in reducing the time complexity of the DFT from O(N2) to O(NlogN) in FFT algorithms like the Cooley-Tukey algorithm.

- Recursively re-expresses discrete Fourier transform of size $N = N_1 N_2$

From DFT we have

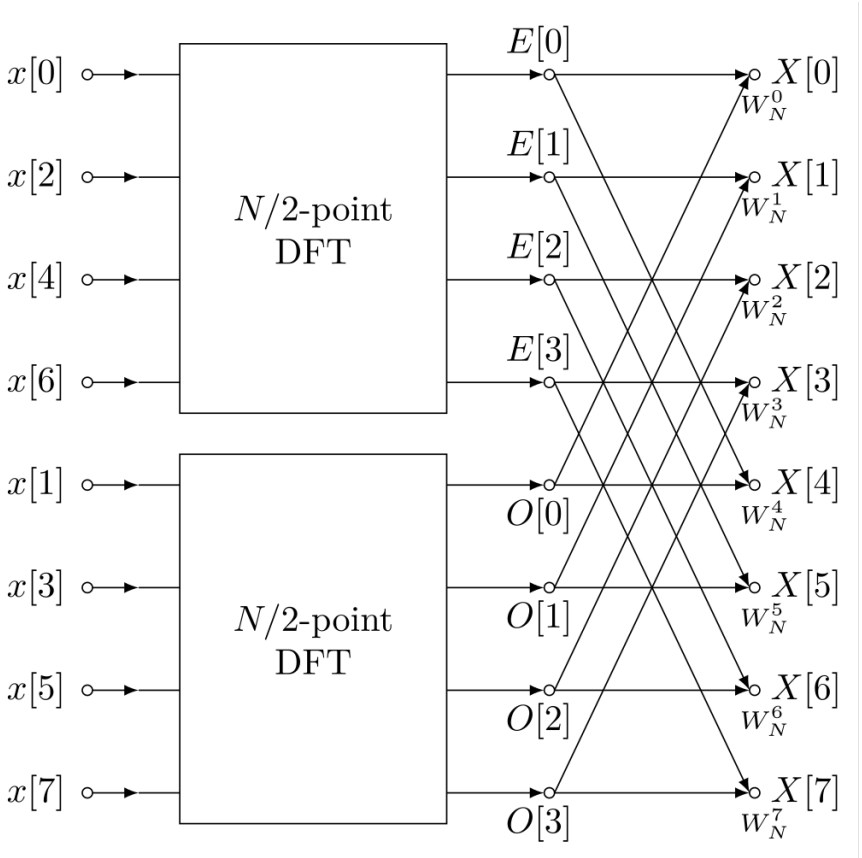
$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-i\frac{2\pi}{N}kn}, where k \in Z$$

It can be written as

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot \omega_N^{nk}, where k \in Z$$

i.e,$X[k] = \sum_{r=0}^{(N/2)-1} x[2r] \cdot \omega_N^{2rk} + \sum_{r=0}^{(N/2)-1} x[2r+1] \cdot \omega_N^{2rk} \cdot \omega_N^k$
i.e,

$$X[k] = G[k] + \omega_N^K H[k]$$



6.1 Coding Implementation

```python
def detect_blur_fft_gpu(image_gpu, index, size=60, vis=False):
    (h, w) = image_gpu.shape
    (cX, cY) = (int(w / 2.0), int(h / 2.0))

    fft = cufft.fft2(cp.asarray(image_gpu))
    fftShift = cp.fft.fftshift(fft)

    if vis and (index == 0 or index == 1):  # Visualize for indices 0 and 1
        magnitude = 20 * cp.log(cp.abs(fftShift))
        (fig, ax) = plt.subplots(1, 2)
        ax[0].imshow(cp.asnumpy(image_gpu), cmap="gray")
        ax[0].set_title("Input")
        ax[0].set_xticks([])
        ax[0].set_yticks([])

        ax[1].imshow(cp.asnumpy(magnitude), cmap="gray")
        ax[1].set_title("Magnitude Spectrum")
        ax[1].set_xticks([])
        ax[1].set_yticks([])

        plt.savefig(f'magnitude_spectrum_{index}.png')  # Save the figure
        plt.show()

    fftShift[h // 2 - size:h // 2 + size, w // 2 - size:w // 2 + size] = 0
    fftShift = cp.fft.ifftshift(fftShift)
    recon = cp.fft.ifft2(fftShift)
    mean = cp.mean(20 * cp.log(cp.abs(recon)))
    return mean

# Assuming 'images' is a list of images
indices_to_visualize = [0, 1]  # Change this to the indices of the images you want to visualize
features = cp.array([detect_blur_fft_gpu(image, index=i, vis=True) if i in indices_to_visualize else detect_blur_fft_gpu(image, index=i) for i, image in e
```

7 FEATURE EXTRACTION FROM IMAGES AND LOGISTIC REGRESSION

From the above code it is clear that we are collecting the feature for each data set and then we apply this feature(consider as a data set) in logistic regression as a binary class.

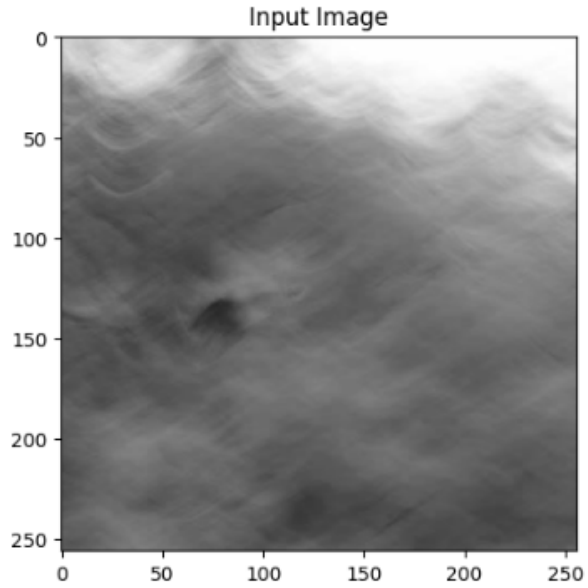
```
#using Logistic Regression from scikit-learn
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
P=model.fit(features_train.reshape(-1,1), Ground_truth_train)

# Assuming your data is stored in a DataFrame or Series named '
Ground_truth_test_list=Ground_truth_test.tolist()

# Evaluate the model on the testing set
predictions = P.predict(features_test.reshape(-1,1))
```

8 OUTPUT

The output evaluation simulates a real-world scenario where the trained model is confronted with an image.This mimics the model's performance when deployed in practical applications, such as analyzing images from different sources or captured under various conditions.

8.1 Coding implementation:

```
# Load the input image
input_image_path = '/content/atmosph_blur.jpeg'
input_image = cv2.imread(input_image_path, cv2.IMREAD_GRAYSCALE)
resized_image = cv2.resize(input_image, (256, 256))
input_image_gpu = cp.asarray(resized_image)

# Detect blur using the GPU-accelerated FFT function
blur_score = detect_blur_fft_gpu(input_image_gpu)

# Reshape the feature for prediction
input_feature = blur_score.reshape(1, -1)

# Convert Cupy array to NumPy array
input_feature_numpy = input_feature.get()

# Make the prediction using the trained model
prediction = P.predict(input_feature_numpy)

# Display the image
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))  # Assuming the image is in BGR format
plt.title("Input Image")
plt.show()

# Print the result
if prediction == 0:
    print("The input image is blurry.")
else:
    print("The input image is not blurry.")
```




```
The input image is blurry.
```

9 PERFORMANCE MEASURE

9.1 Introduction to Performance Measure:

For logistic regression binary class classification problem we checked our model performance by calculating **True Positive,True Negative,False Positive,False Negative** .

9.2 Metrics Section:

- Accuracy:The overall correctness of the model's predictions.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}}$$

- Precision: The ratio of true positives to the total predicted positives.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

- Recall (Sensitivity): The ratio of true positives to the total actual positives

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

- F1-score: The harmonic mean of precision and recall.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

9.3 For My model:

True_Positive = 27, True_Negative = 37,False_Negative = 2,False_Positive = 4

- Accuracy = 0.914285
- Precision = 0.87096
- Recall = 0.93103
- F1 score = 0.8999

10 CONCLUSION

In conclusion, this project aimed to develop an efficient blur detection system using GPU-accelerated Fast Fourier Transform (FFT) in conjunction with Logistic Regression. The key findings and outcomes of the project can be summarized as follows:

- **Efficient Blur Detection:**The project successfully established a correlation between blur and specific features extracted through GPU-accelerated FFT. This laid the foundation for effective logistic regression-based classification.
- **Comparison In GPU and CPU:**In **GPU** it took **15.07** min. And **CPU** it took **41.03** min
- **Limitation Of Model:** Model is working for more or less all image .It is not effective for Noise-Induced Blur,Low Resolution Blur and Digital Artifacts.
- **Difficulties During the Project:** I faced to collect the data sets for all kind of blur image.

11 REFERENCES

- https://pyimagesearch.com/2020/06/15/open
- https://www.kaggle.com/datasets/kwentar/blur-dataset
- Chen, Ming, et al. "Image deblurring method based on fourier transform." 2019 6th International Conference on Systems and Informatics (ICSAI). IEEE, 2019.
- Digital Image Processing ,Third Edition,Rafael C. Gonzalez,Richard E. Woods.

————THE END————