BITS Pilani
K K Birla Goa Campus

TENSOR TITANNS

AMAZON ML HACKATHON

Team Lead: **Pranjay Yelkotwar** (f20221209@goa.bits-pilani.ac.in) Team Member1: **Priyansh Jain** (f20221250@goa.bits-pilani.ac.in)
Team Member2: **Nirvan Patil** (f20221250@goa.bits-pilani.ac.in)

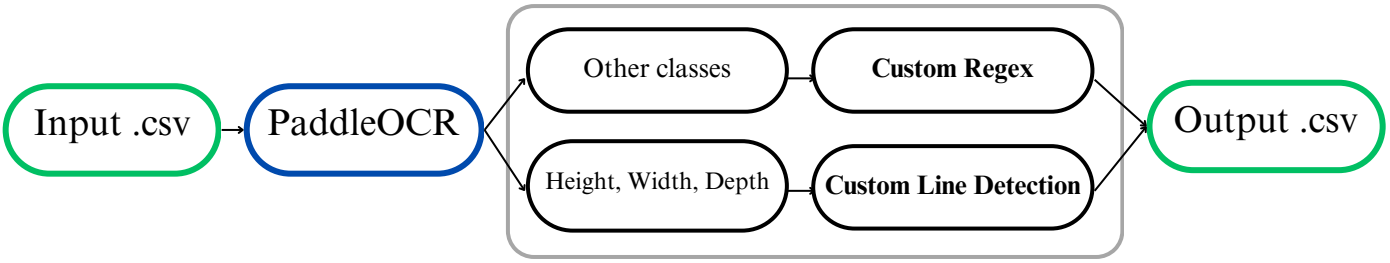
Codebase Repo Link

Problem Statement

In this hackathon, the goal is to create a **machine learning model** that **extracts entity values from images**. This capability is crucial in fields like healthcare, e-commerce, and content moderation, where precise product information is vital. As digital marketplaces expand, **many products lack detailed textual descriptions**, making it essential to obtain key details directly from images. These images provide important information such as weight, volume, voltage, wattage, dimensions, and many more, which are critical for digital stores.

Our Approach *Models Used*

We make use of OCR **(Optical Character Regocnition)** to extract text from the image. The extracted test is passed on to a **Custom Logic Block,** consisting of **Regex** Algorithms and **Line Detection** Algorithms (*explained later*). The output of this Custom Logic Block is then written to .csv in the required format.



Custom Logic Block

Challenges Faced *Experimentation*

- **Limited GPU Resources:**
  - Local deployment of vision **LLMs** resulted in an average processing time of **~1.2 seconds** per image on 2xT4 GPU on Kaggle. Running the entire test set would necessitate approximately **1.56 days of continuous GPU usage**, making this approach impractical.
  - As students with limited access to high-performance GPU infrastructure, fine-tuning smaller pre-trained models or training new ones is unfeasible due to time and resource constraints.

- **Optical Character Recognition (OCR):**
  - Given its lightweight nature, OCR emerged as a viable alternative.
  - After testing various OCR tools, **PaddleOCR** yielded the best performance on a 2xT4 GPU setup on Kaggle.
  - A customized version of PaddleOCR was employed to extract text from the input images.

- **Need for Regex**
  - OCR output is a collection of characters and numbers.
  - Regex used to **filter out relevant numeric values** along with their units.

- **Need for Custom Line Detection Algorithm**
  - For *entity_name: item_weight, maximum_weight_recommendation, voltage, wattage, item_volume*, above methodology can be directly used as extracted text contains keywords *"Volt", "Watt", "Weight",* etc.
  - For *entity_name*: *height, width, depth* extracted text does not contain keywords *"Height", "Width", "Depth"* as the images have arrows marking the dimensions and not text.
  - Thus, there is **no direct pathway to differentiate between the different numeric values** (for height, width and depth) extracted by the OCR.
  - To overcome this hurdle without using large vision models, we developed a lightweight **Custom Line Detection algorithm** based on OpenCV *(explained in next section)*.

Custom Regex

*We develop a **robust** Regex, accounting for decimal points and commas as decimal markers (European). Different Units and multiple of the same units in a sentence. Other functionality mentioned below:*

- **Number Formatting:** The `format_number` function removes trailing zeros and unnecessary decimal points when formatting numbers to two decimal places.
- **Regex for Extraction:**
  - **Weight**: The `extract_weight` function extracts the largest value associated with units like milligrams, grams, kilograms, ounces, pounds, and tons.
  - **Wattage**: The `extract_wattage` function retrieves the largest number related to wattage, either in watts or kilowatts.
  - **Voltage**: The `extract_voltage` function extracts values in millivolts, volts, or kilovolts.
  - **Volume**: The `extract_volume` function handles units like milliliters, liters, gallons, fluid ounces, and cubic feet.
- **OCR Integration**: The `compute_dimension` function performs OCR on an image to extract text and then calls the specific extraction functions to obtain key values, such as weight or wattage.
- **Error Handling**: It includes basic error handling to ensure the process doesn't break if the image is not found or OCR fails to extract text.
- **Center Marking**: The image is processed to mark its center using OpenCV before extracting the text using OCR.
- **Timing and Debugging**: The script calculates the total time taken for the OCR process and displays it depending on the `time_mode` flag.

Line Detection Algorithm

- For images with *entity_name*: *height, width, depth* OCR is used to extract all text and corresponding bounding boxes in the image. Bounding boxes containing numeric values and units are filtered out.
- OpenCV (**cv2.Canny** and **cv2.HoughLinesP**) used to extract all lines in the image.
- **Algorithm Logic:**
  - For each of the filtered bounding boxes, the line closest to their center is obtained.
  - Angle of the line is obtained using standard mathematical equations:
    - If the angle lies between 75° to 105°, the corresponding bounding box's numeric value along with it's corresponding unit is added to a list (of tuples) for potential candidates for height or depth.
    - If the angle lies between 0° to 60°, we have a potential candidate for width.
  - Among all candidates for each category (height, width, depth), the candidate with maximum numeric value of dimension is found and returned as the answer along with the relevant extracted unit.
- Using this process, **each image**, end-to-end, **now takes ~0.2seconds** allowing us to experiment and prepare the test.csv file within a reasonable time frame.

CONCLUSION

In conclusion, by utilizing **PaddleOCR** for text extraction and implementing a **Custom Regex** and **Line Detection algorithms**, we efficiently **address resource limitations** while extracting key dimensions and values. This streamlined approach, with **minimal GPU dependency**, reduced processing time to approximately **~0.2 seconds per image**, enabling **large-scale data generation** in a practical and resource-efficient manner **without** relying on **heavy vision models.**

Codebase Repo Link Kaggle Notebook 1 Kaggle Notebook 2
