2024/2025

DEVELOPMENT OF A SOFTWARE APPLICATION ON AN EMBEDDED SYSTEM

English handwriting recognition and lexical + Grammatical correction

BERDNYK Mariia - IA/ID
SOP DJONKAM Karl Alwyn - IA/ID
HADDOU Amine - IA/ID

BENNA Bachir - IoT
ESSAM Steven - IoT

POLYTECH NICE-SOPHIA

UNIVERSITÉ CÔTE D'AZUR

WHY THESE CHOICES?

01. **Programm**

02. **Hardware and algorithmes Choices**

03. **Benchmarking**

04. **Limitations**

Description of the application

Goal : Correcting a student dictation

01. **Photographing Textual Content**

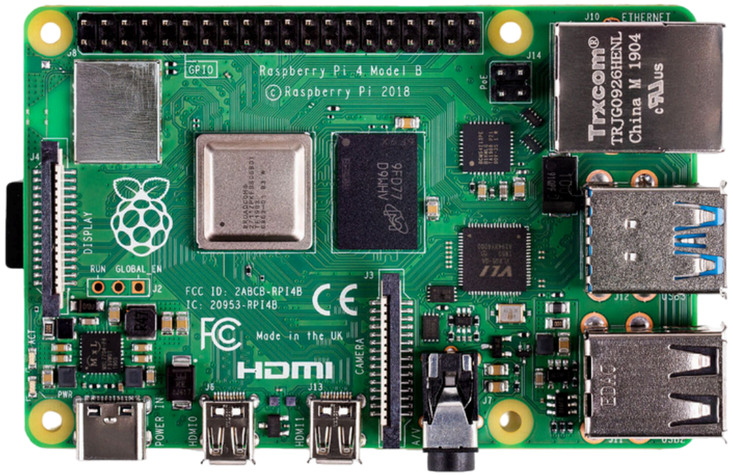
02. **Text Extraction**

03. **Correcting Spelling and Grammar**

HARDWARE CHOICE

Raspberry Pi 4

Main platform to host the application, chosen for its versatility and wide compatibility with AI libraries.

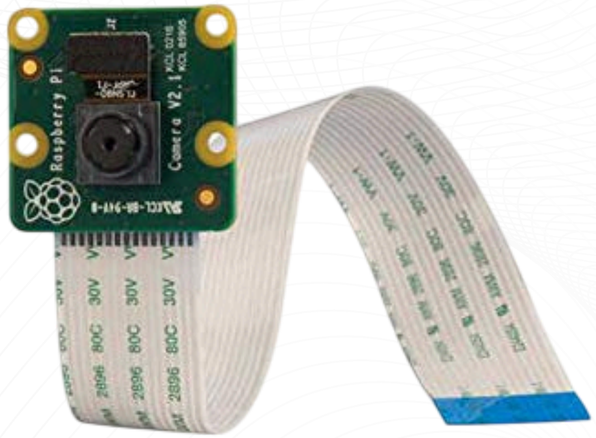


Google Coral TPU

Hardware acceleration module for fast AI inference with low power consumption.

Raspberry Pi Camera

Compact sensor optimized for capturing high-quality images, directly integrable with the Raspberry Pi.



WHY THESE CHOICES?

01.

Raspberry Pi 4

- Strong enough for running tasks in a small, embedded system.

02.

Google Coral TPU

- Makes AI models run faster with less delay.
- Designed for small systems like ours.

03.

Raspberry Pi Camera

- Simple to connect and use with Raspberry Pi.
- Takes clear pictures of handwritten text.

Algorithm choices

01.

Preprocessing

- Background processing to obtain a white background
- Image contrast tweaking
- split multi-line text in multiple image containing each a line

02.

Extract text to image

- Extract text from image (OCR) using an AI model from microsoft
- Our preprocessing help to have better performance than without

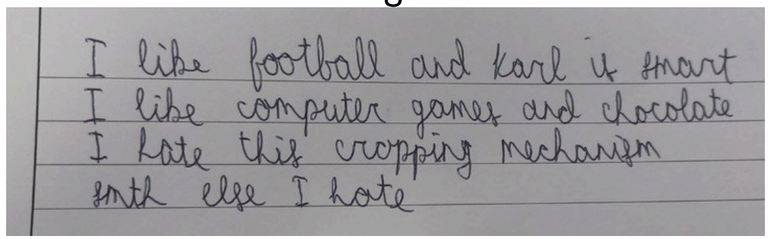
03.

Grammar & spelling checker

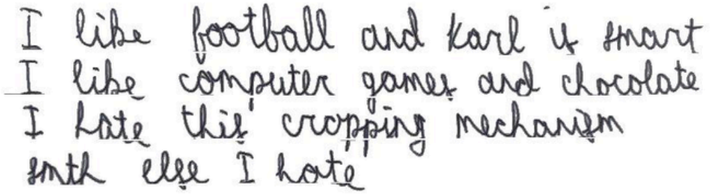
- Usage of an AI model for grammar checking
- Usage of python library for spelling checking
- Input: the text from the previous step

Preprocessing:
Background removal

Original

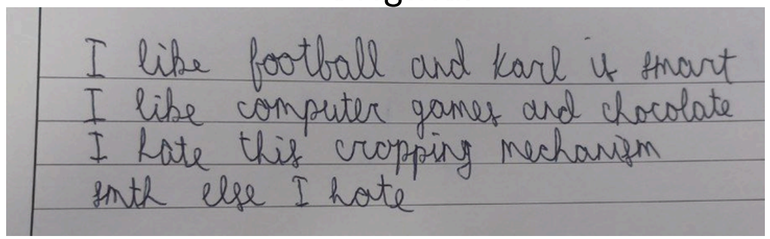


Final

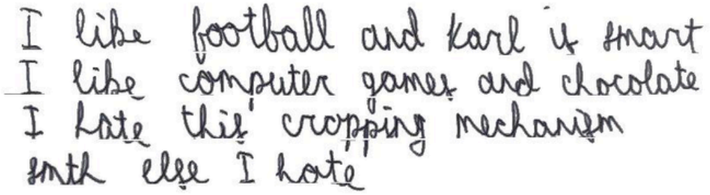


Preprocessing:
Background removal

Original



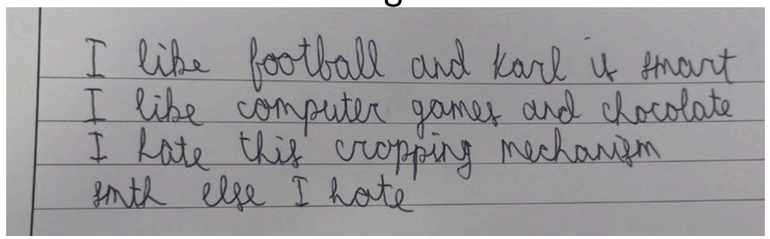
Final



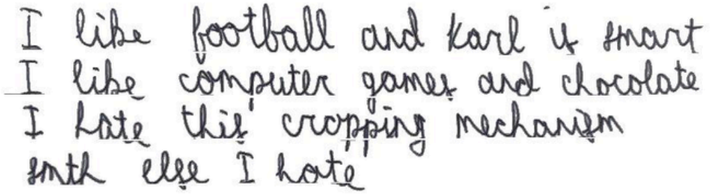
OpenCV, rembg or others background removal libraries do not work -> they remove the handwriting itself too.

Preprocessing:
Background removal

Original



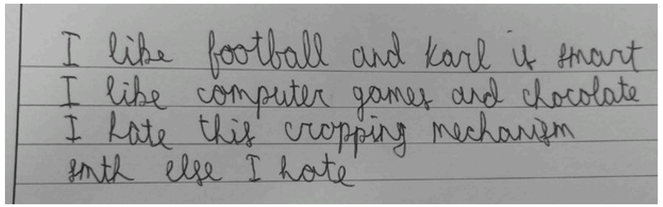
Final



OpenCV, rembg or others background removal libraries do not work -> they remove the handwriting itself too.

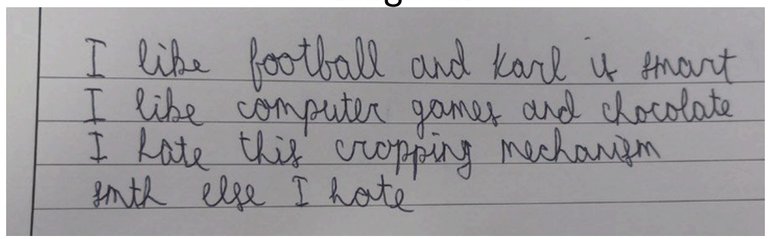
Custom background removal:

1. **Convert the image to grayscale**

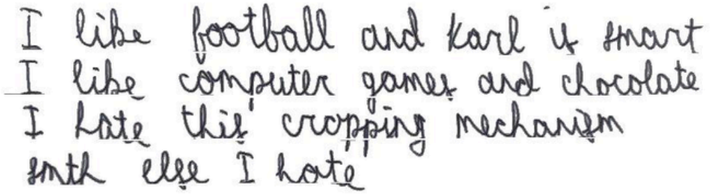


Preprocessing:
Background removal

Original



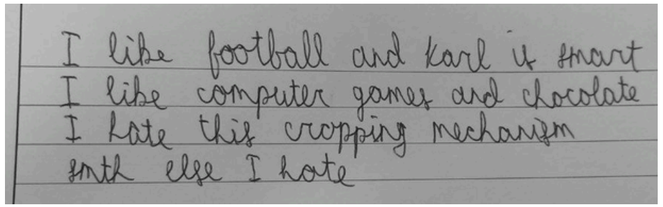
Final



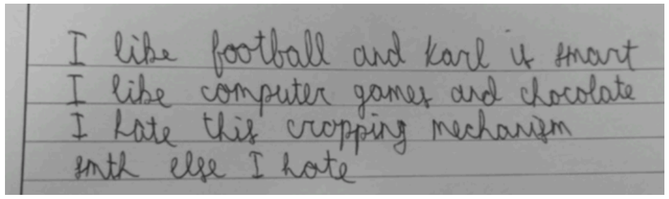
OpenCV, rembg or others background removal libraries do not work -> they remove the handwriting itself too.

Custom background removal:

1. Convert the image to grayscale

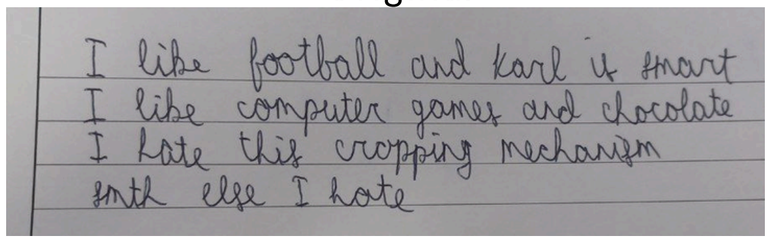


2. Apply Gaussian Blur to get read of the noise (5x5 kernel)

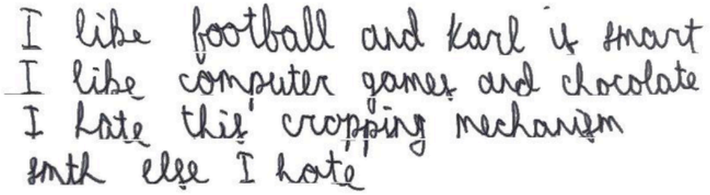


Preprocessing: Background removal

Original



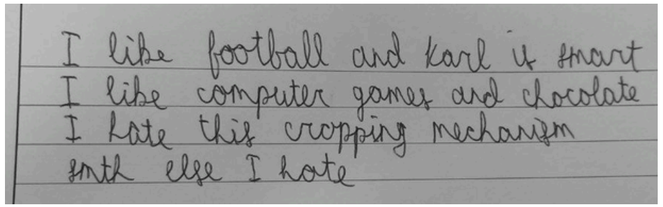
Final



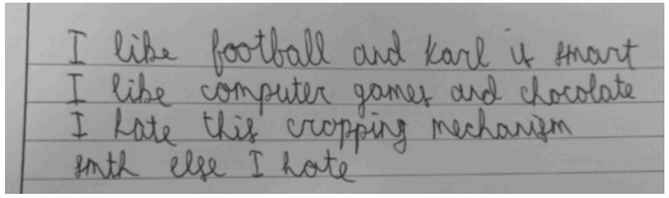
OpenCV, rembg or others background removal libraries do not work -> they remove the handwriting itself too.

Custom background removal:

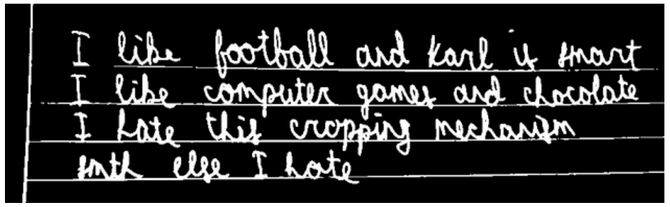
1. Convert the image to grayscale



2. Apply Gaussian Blur to get read of the noise (5x5 kernel)

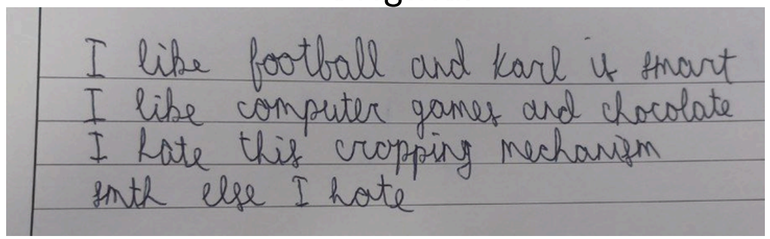


3. Apply adaptive thresholding to binarize the image

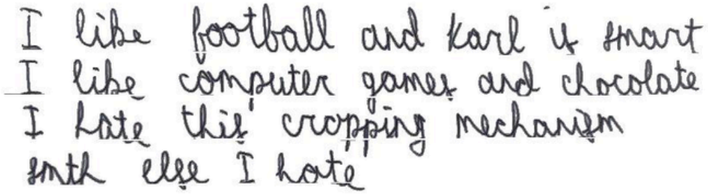


Preprocessing: Background removal

Original



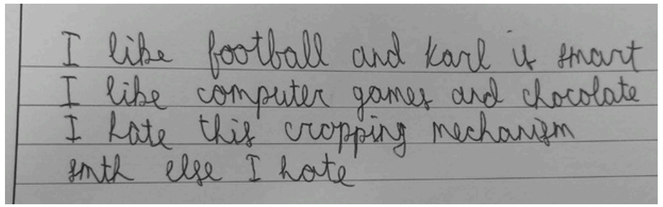
Final



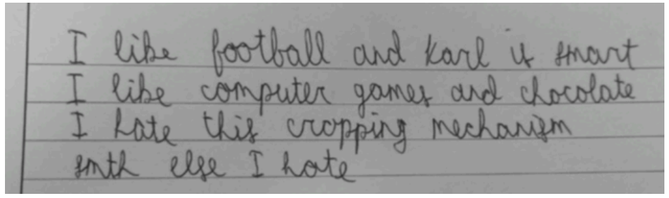
OpenCV, rembg or others background removal libraries do not work -> they remove the handwriting itself too.

Custom background removal:

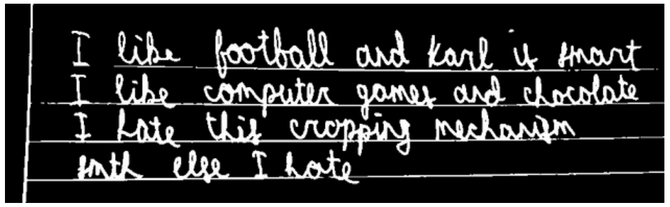
1. Convert the image to grayscale



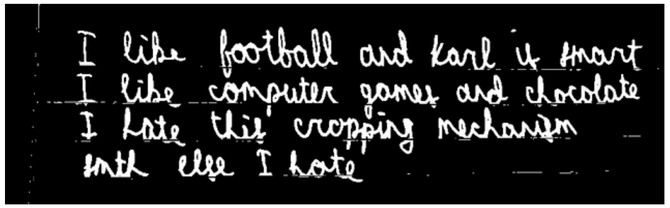
2. Apply Gaussian Blur to get read of the noise (5x5 kernel)



3. Apply adaptive thresholding to binarize the image

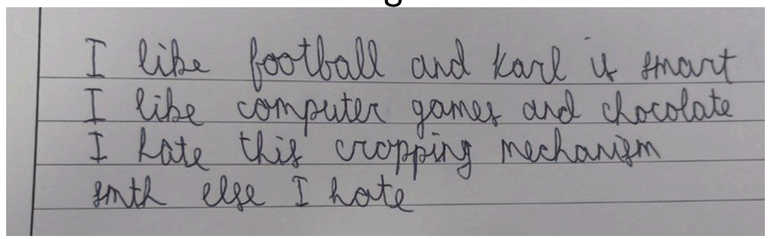


4. **Detect vertical & Horizontal lines** by getting the structured vertical and horizontal elements. Combine horizontal and vertical lines into one grid mask and invert it from the original image

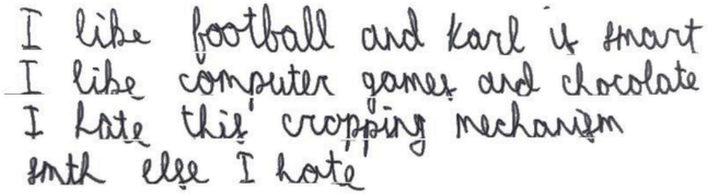


Preprocessing: Background removal

Original



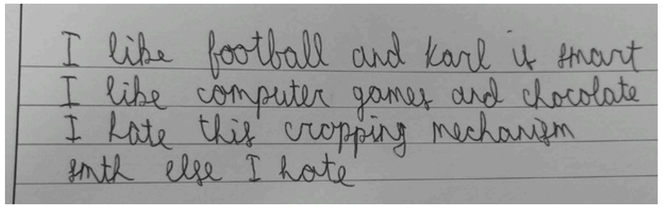
Final



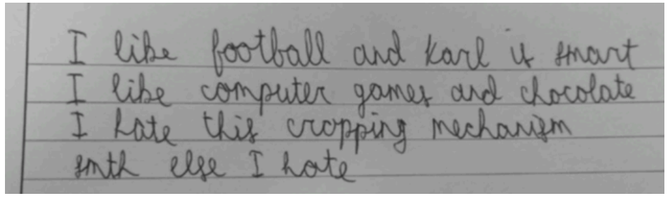
OpenCV, rembg or others background removal libraries do not work -> they remove the handwriting itself too.

Custom background removal:

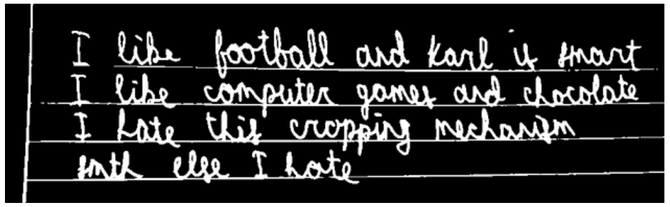
1. Convert the image to grayscale



2. Apply Gaussian Blur to get read of the noise (5x5 kernel)

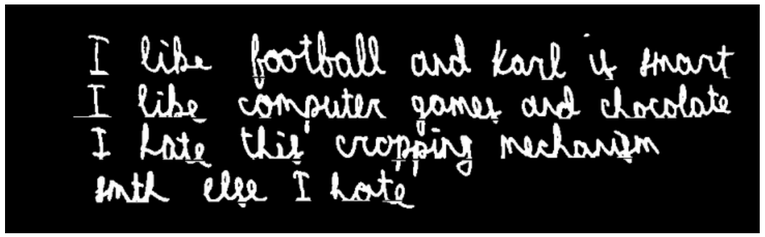


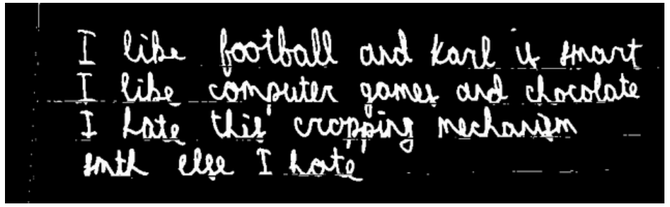
3. Apply adaptive thresholding to binarize the image



4. Detect vertical & Horizontal lines by getting the structured vertical and horizontal elements. Combine horizontal and vertical lines into one grid mask and invert it from the original image

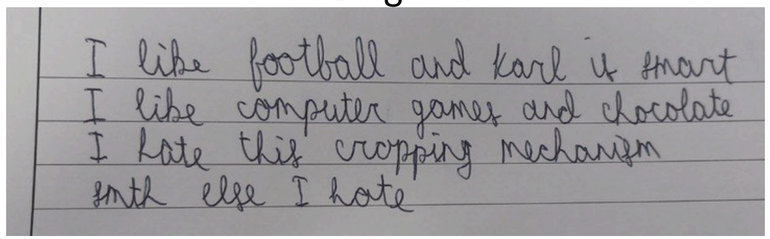
5. Remove the small noise:



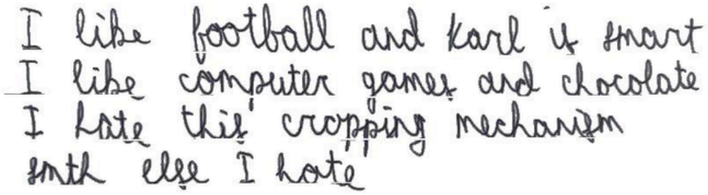


Preprocessing: Background removal

Original



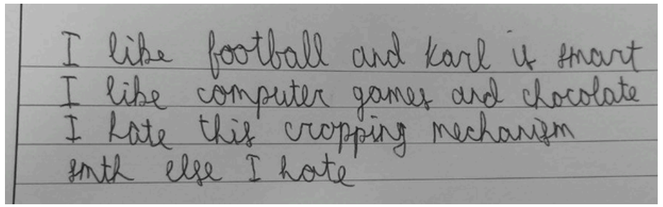
Final



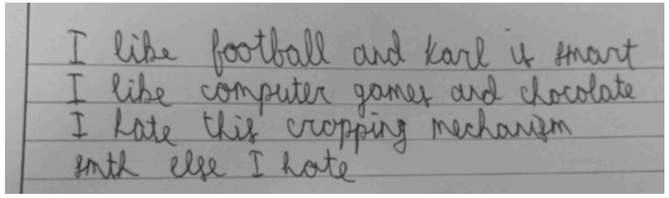
OpenCV, rembg or others background removal libraries do not work -> they remove the handwriting itself too.

Custom background removal:

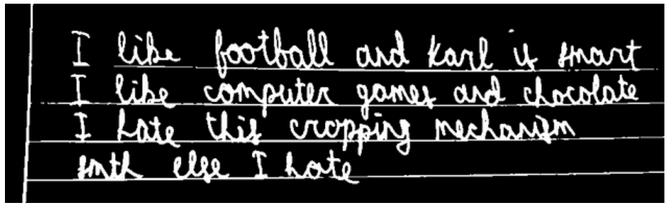
1. Convert the image to grayscale



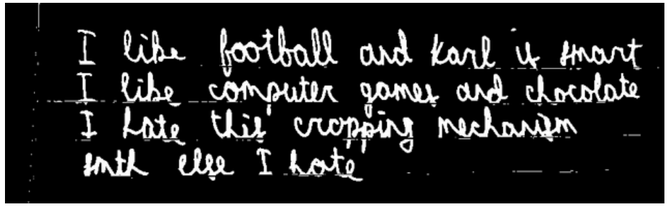
2. Apply Gaussian Blur to get read of the noise (5x5 kernel)



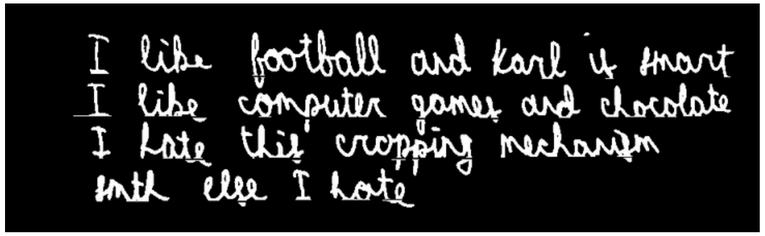
3. Apply adaptive thresholding to binarize the image



4. Detect vertical & Horizontal lines by getting the structured vertical and horizontal elements. Combine horizontal and vertical lines into one grid mask and invert it from the original image



5. Remove the small noise:



6. **Put the image into the black content, white background** -> Final

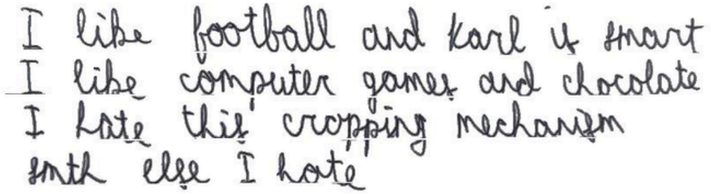
Preprocessing:
Cutting mechanism

OpenCV or others cutting libraries do not work -> as they cannot cut the image by lines if the caps or tails are present.

Custom lines cut mechanism:

1. Apply horizontal projection to detect text lines
2. Threshold to detect significant text regions (e.g. <10% of the maximum)
3. Create the first extracted draft lines
4. Calculate the average vertical size of lines and Filter out lines that are less than half the average height
5. Combine those filtered lines with the previous line and the next one. (These lines are caps and tails of the characters)
6. **Clean the image from small noise (*not really working*)**

Original



Final

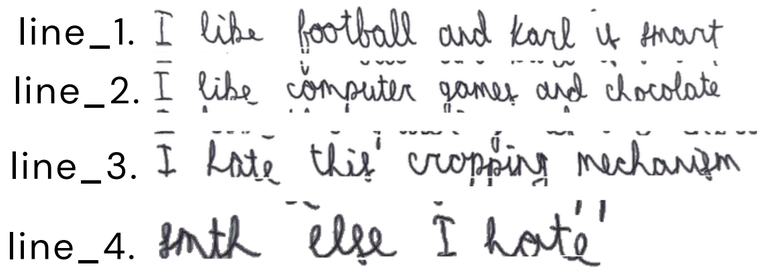


Image2Text:

TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models

The TrOCR model is an **encoder-decoder** model, consisting of an **image Transformer** as encoder, and a **text Transformer** as decoder. The image encoder was initialized from the weights of BEiT, while the text decoder was initialized from the weights of RoBERTa.

Model size | 333M params

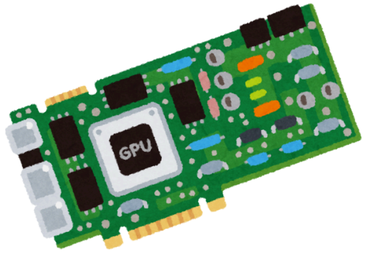
~

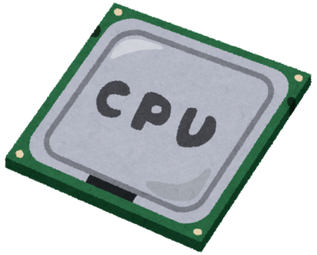
1.24 GB before quantization

~

0.65 GB with FP16 quantization || 0.31 GB with INT8 quantization

TrOCR benchmarking: 1000 images





NVIDIA GeForce RTX 4060 8188MiB	
execution time	max/min taken memory
6 images at a time:	
113.59 s	**1861/231 MiB**
4 images at a time:	
116.82 s	**1827/231 MiB**
2 images at a time:	
132.85 s	**1759/231 MiB**

13th Gen Intel(R) Core(TM) i7-13700H
execution time
6 images at a time:
1158.93 s
4 images at a time:
1232.53 s
2 images at a time:
1127.03 s

Quantized TrOCR benchmarking: 1000 images

GPU: NVIDIA GEFORCE RTX 4060 8188MIB

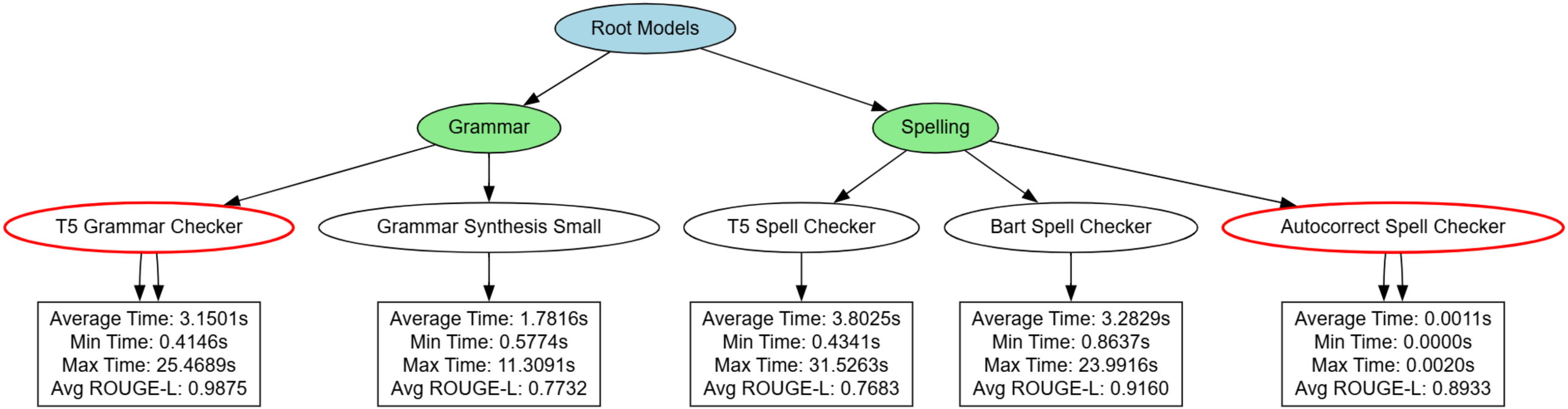
OCR from HF

execution time	max/min taken memory	ROUGE score
	6 images at a time:	
113.59 s	**1861/231 MiB**	
	4 images at a time:	
116.82 s	**1827/231 MiB**	**91,92%**
	2 images at a time:	
132.85 s	**1759/231 MiB**	

Quantized OCR

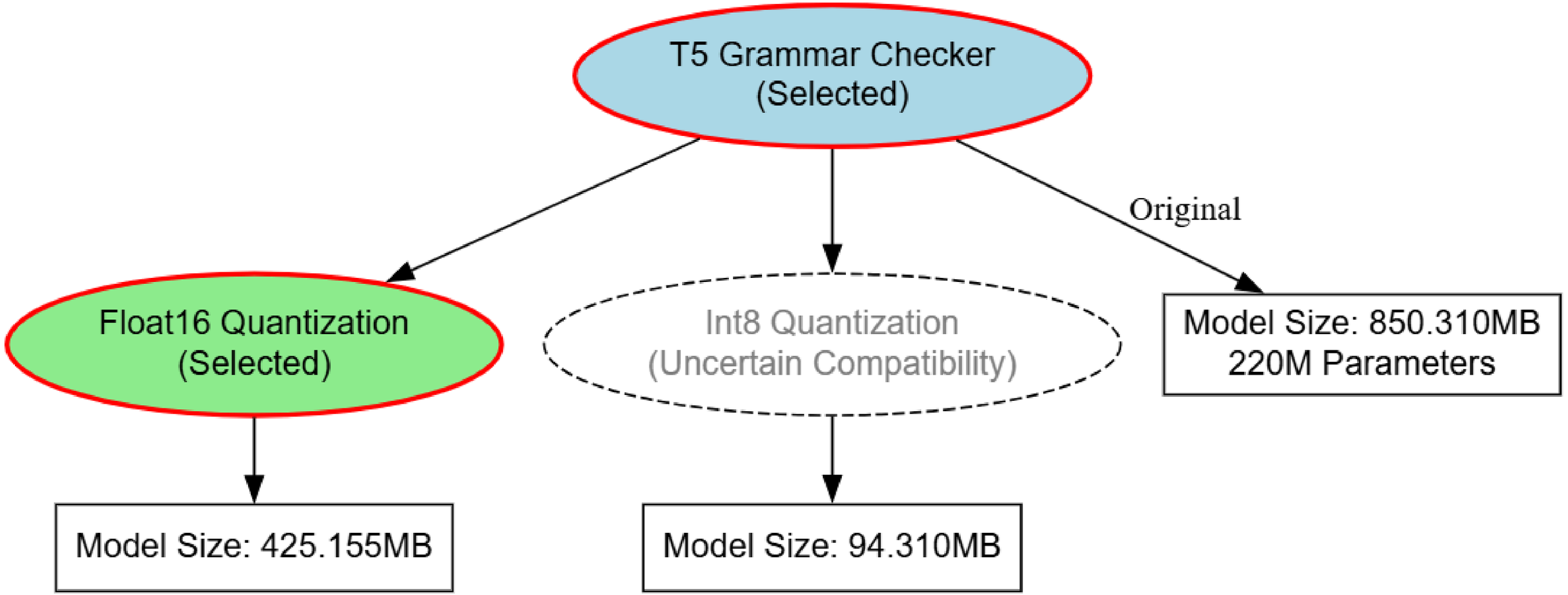
execution time	max/min taken memory	ROUGE score
	6 images at a time:	
57.15 s	**1205/231 MiB**	
	4 images at a time:	
64.44 s	**1171/231 MiB**	**91,95%**
	2 images at a time:	
102.36 s	**1089/231 MiB**	

Grammar & spelling checkers benchmarking:

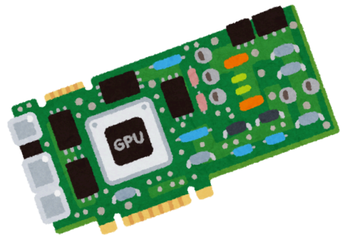


T5 Grammar Checker: prithivida/grammar_error_correcter_v1
Grammar Synthesis Small: prithivida/grammar_error_correcter_v1
T5 Spell Checker: Bhuvana/t5-base-spellchecker
Bart Spell Checker: oliverguhr/spelling-correction-english-base
Autocorrect: https://github.com/filyp/autocorrect

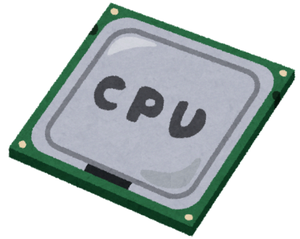
Quantization T5 Model



All process without preprocessing:

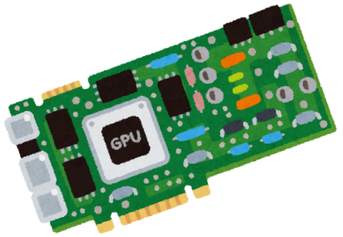


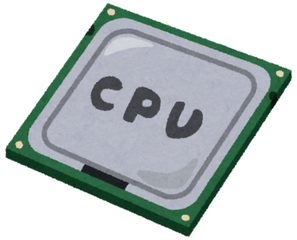
NVIDIA GeForce RTX 4060 8188MiB	
execution time	max/min taken memory
6 images at a time:	
212.71 s	**2009/231 MiB**
4 images at a time:	
215.9 s	**1999/231MiB**
2 images at a time:	
226.85 s	**1899/231 MiB**

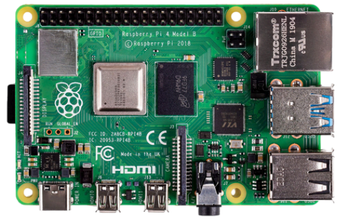


13th Gen Intel(R) Core(TM) i7-13700H
execution time
6 images at a time:
1130.37 s
4 images at a time:
1169.35 s
2 images at a time:
1189.27 s

Processing on one image







NVIDIA GeForce RTX 2060 Mobile	AMD Ryzen 7 4800HS	Raspberry Pi 4
execution time	execution time	execution time
TIme : 4.1s **Frequency** ~1365 MHz **Thermals :** ~40 degres	**TIme :** 1165,5s **Frequency** 2900Mhz **Thermals :** ~55 degrees	**TIme :** Aborted **Frequency** 1500 Mhz **Thermals :** ~65 degrees

Risks analysis

Hardware Risks

1. Raspberry Pi 4

May struggle with processing intensive AI models without additional optimization.

2. Raspberry Pi Camera

Captured image quality may **vary under poor lighting conditions**, affecting the accuracy of text recognition.

3. Google Coral TPU

Compatibility issues with certain AI models not optimized for the TPU.

4. Models Size

The models can still be heavy even after quantization

Models risks

- Errors can **propagate** between the models (e.g. incorrect Image2Text recognition leads to wrong spelling check)

Image2Text model

1. Preprocessing step can fail

Wrong lines cropping, noise removal, angle change - lead to the Image2Text model fail

2. Rare handwriting & words

The model can fail to recognize rare **handwriting type** or rare **words**

Spelling model

Model overcorrect

Can significantly alter rare words, occasionally replacing them with entirely different terms.

Grammar model

High Text Error Rates

The model sometimes fails to correct the text with **numerous errors**

	Insignifiante	Mineure	Significative	Majeure	Sévère
Presque Certain	Moyen	Elevé	Très Elevé	Extreme	Extreme
Probable	Moyen	Moyen	Elevé (1)	Très Elevé	Extreme
Modéré	Faible	Moyen	Moyen	Elevé (6)	Très Elevé (2) (5)
Improbable	Tres Faible (3)	Faible	Moyen (4)	Moyen (8)	Elevé
Rare	Tres Faible	Tres Faible	Faible	Moyen (7)	Moyen

1. Raspberry Pi 4- AI model takes a lot of time to run
2. Raspberry Pi Camera - Poor external conditions affect image quality
3. Google Coral TPU - Compatibility issues with AI models
4. Energy consumption because of the Model Size - Heavy even after quantization
5. Image2Text - Preprocessing failure
6. Image2Text - Rare handwriting/words unrecognized
7. Spelling Model - Overcorrects rare words and gives wrong results
8. Grammar Model - High text error rates

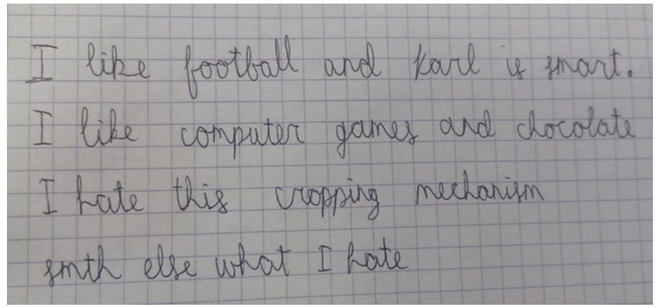
Preprocessing problems:

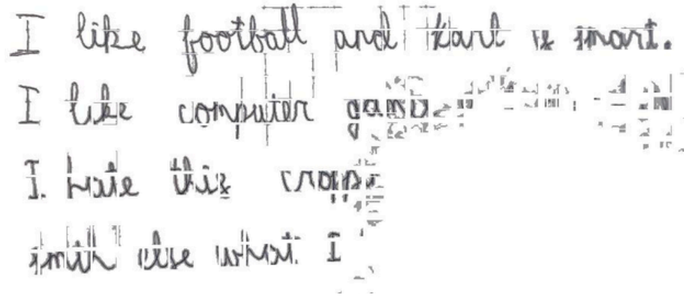
Background removal

Preprocessing problems:

Background removal

1. Small contrast between the text and background:

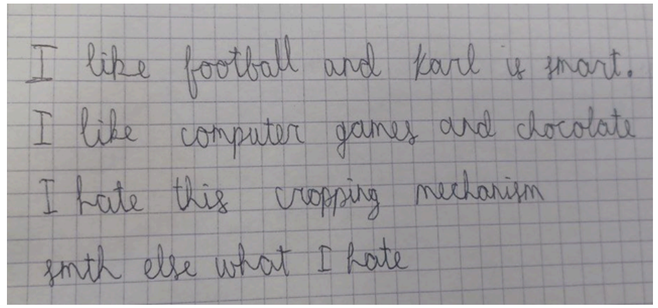


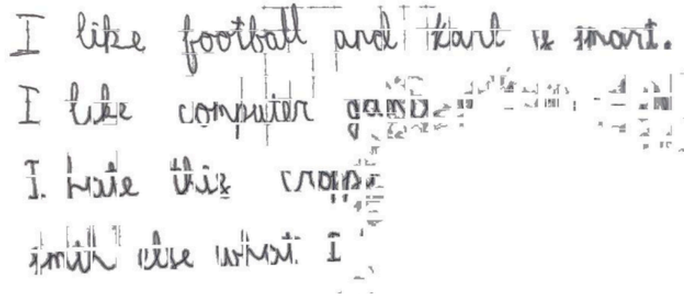


Preprocessing problems:

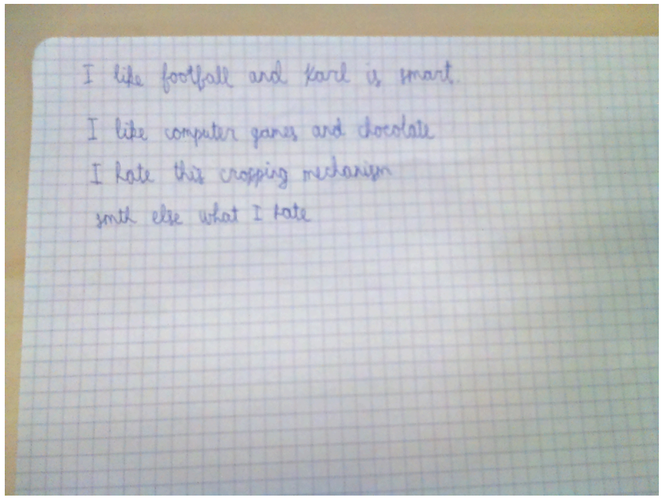
Background removal

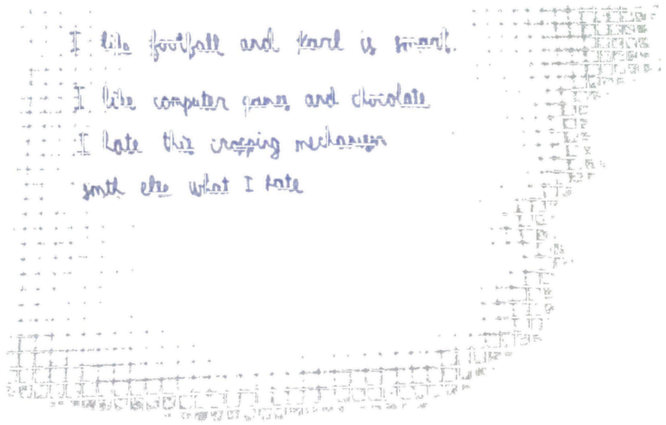
1. Small contrast between the text and background:





2. Image quality

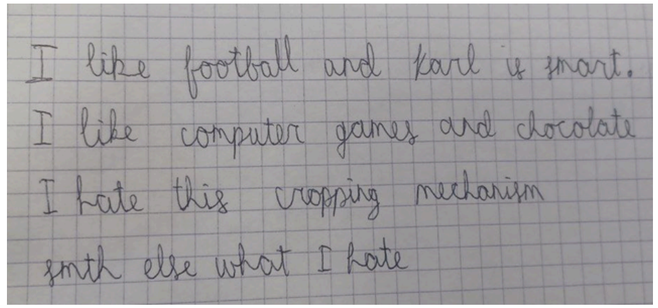


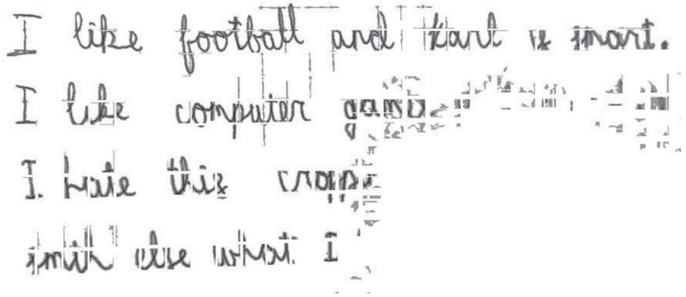


Preprocessing problems:

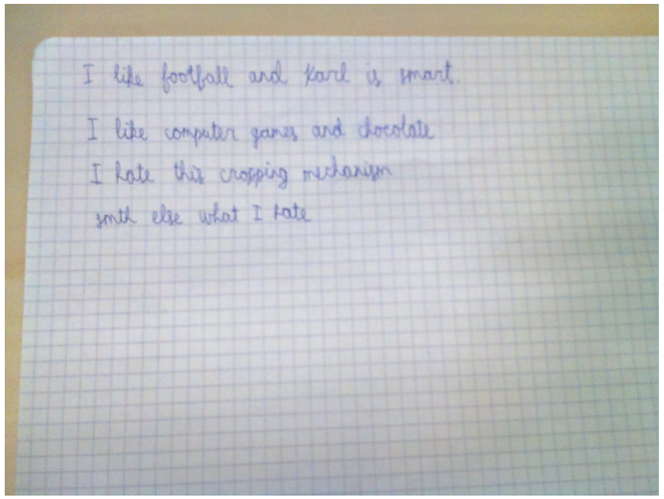
Background removal

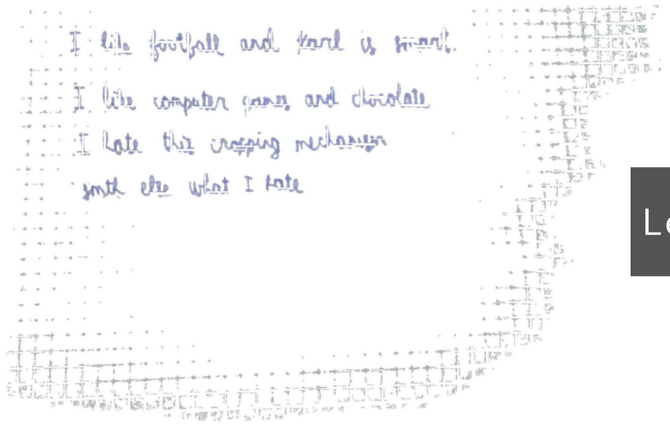
1. Small contrast between the text and background:





2. Image quality





Leads to

Cropping

line_1.

line_2.

line_3.

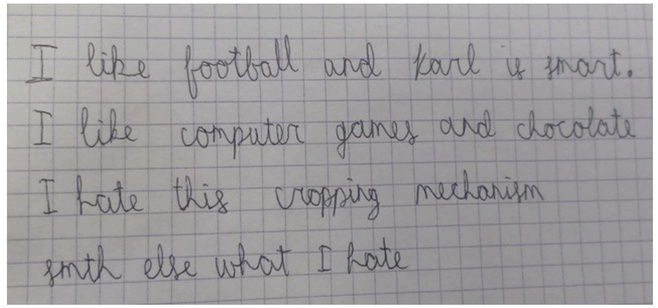
.....

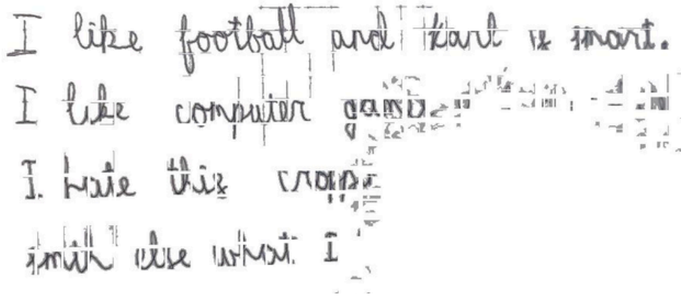
10 more lines

Preprocessing problems:

Background removal

1. Small contrast between the text and background:





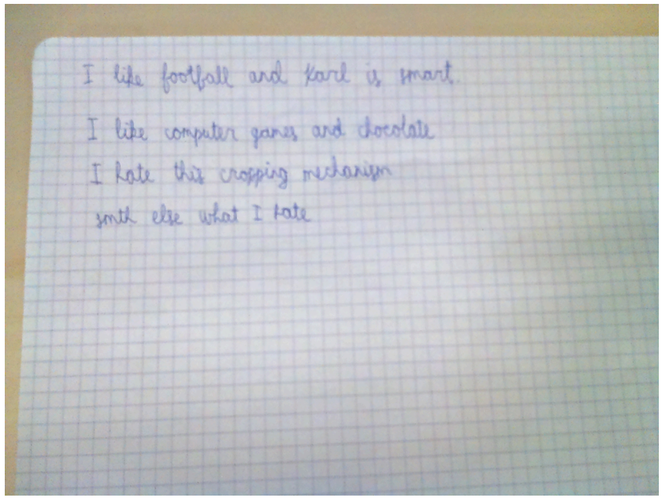
OPEN QUESTIONS:

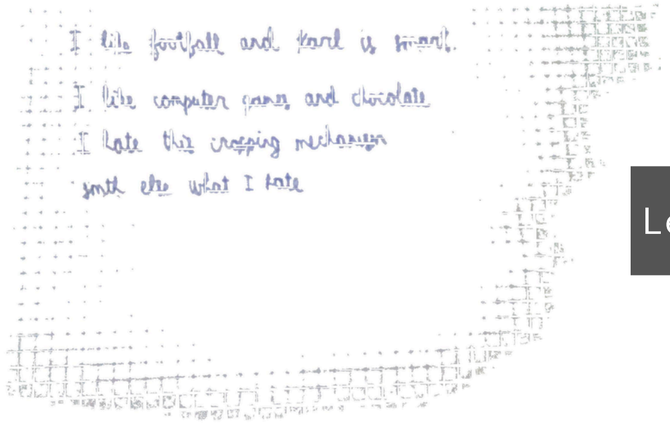
3. How to detect the skewed image?

<u>OR EVEN WORSE</u>

4. What should we do if writings from different angles are present?

2. Image quality





Leads to

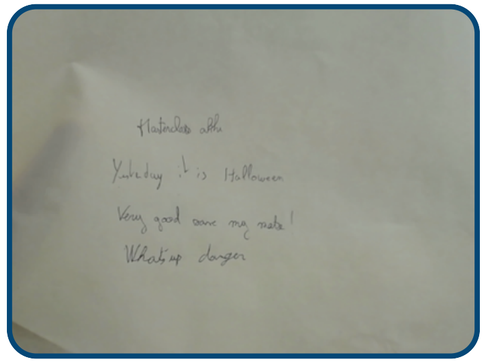
Cropping

.....

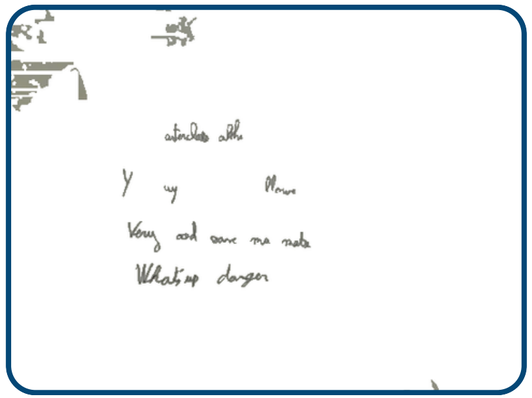
10 more lines

Image quality

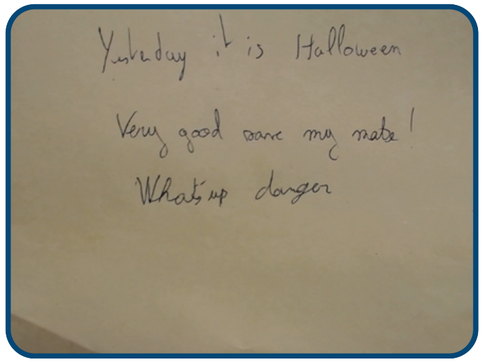
Adjusting the camera settings and zoom is important to capture a clear picture that the model can recognize. Depending on future tests, the camera might need to be changed.



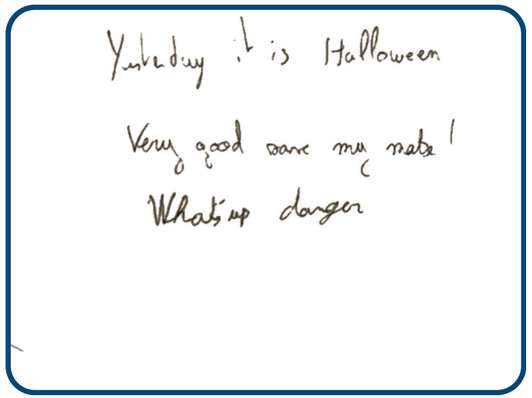
Processing



In this example changing the camera from a logitech camera to a pi camera with adjustable focus and better lighting yields betters results



Processing

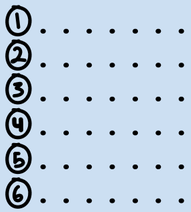


AI Limitations

The project is only for **English language** recognition and correction

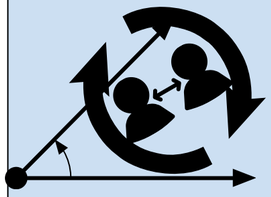
The models perform optimally on **legible** and standard **handwriting**

If the text is skewed - it cannot be recognized by the model

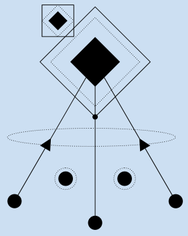


The number of lines per one image is **limited** + The **margin** between the lines **is preferable**

White background without watermarks is **preferable**



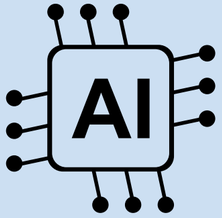
Distance -> as close as possible,
Angle -> as right as possible
Rotation -> As straight as possible



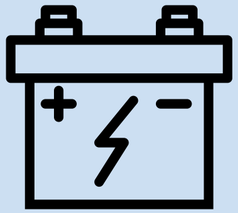
Add a model or framework to correct **syntax** errors. They are **poorly handled** now.

The grammar model is not performing *the best* when **numerous errors are present**

IOT Limitations

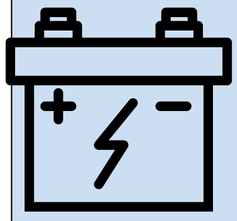


The AI model needs to be optimised to run on a raspberry pi, we may also need to switch the OS from 32 bits to 64 bits

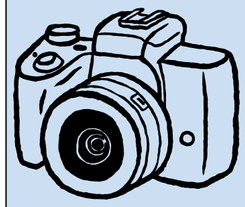


Hardwares may require a stable power supply

Fixed resolution and field of view of Raspberry Pi Camera limits its ability to capture certain handwritten text layouts

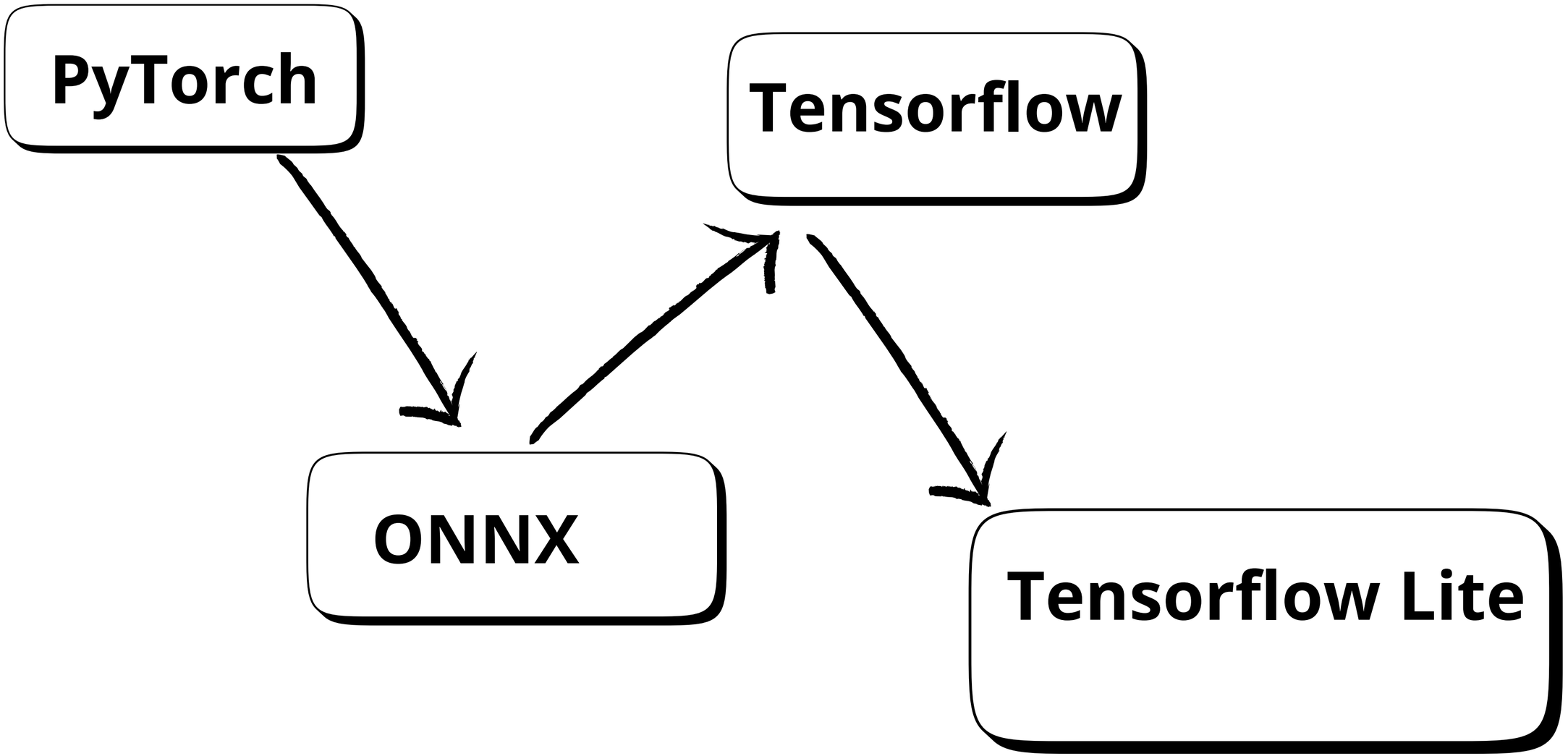


Not enough Raw CPU Power

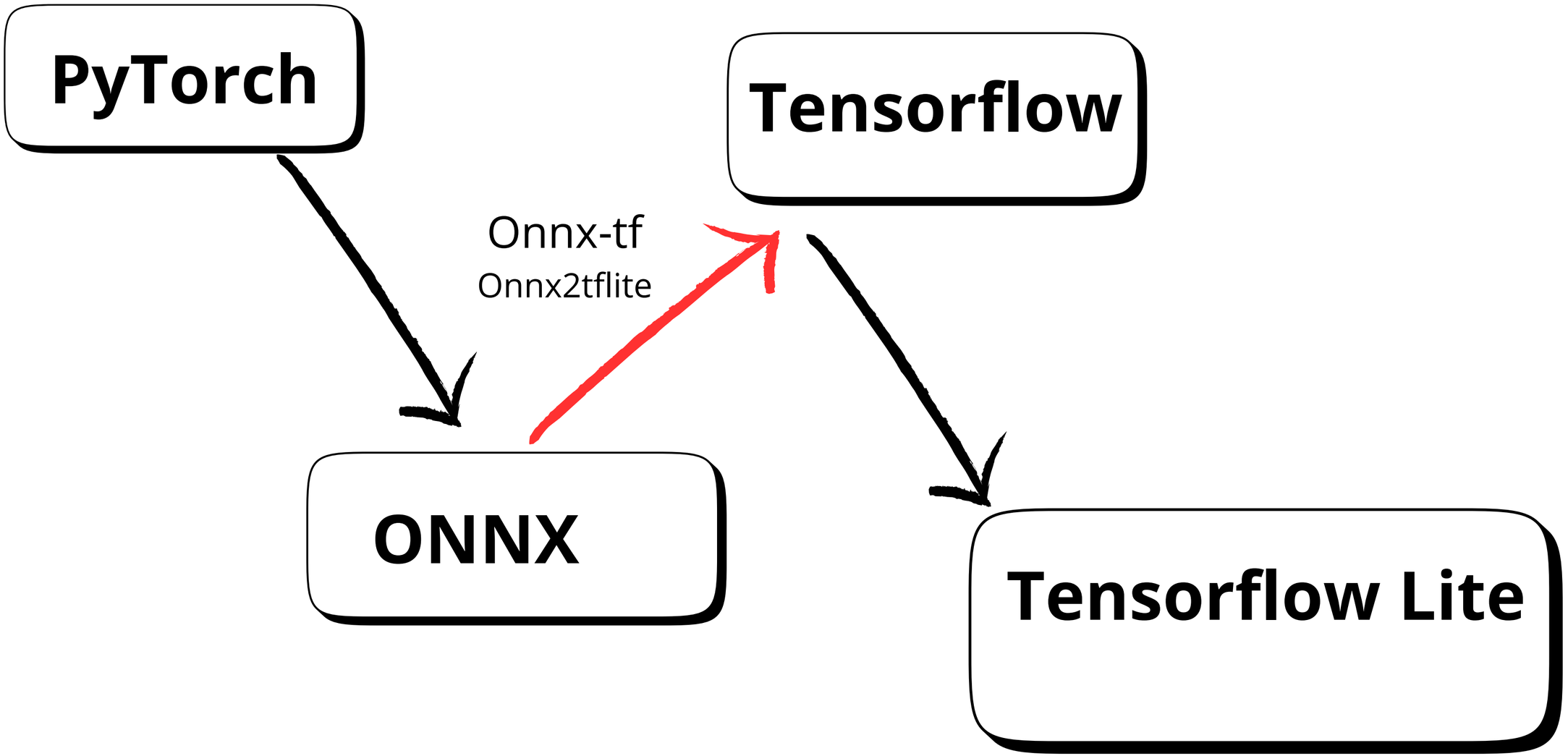


High camera quality and resolution is capital

TPU : Incompatibility



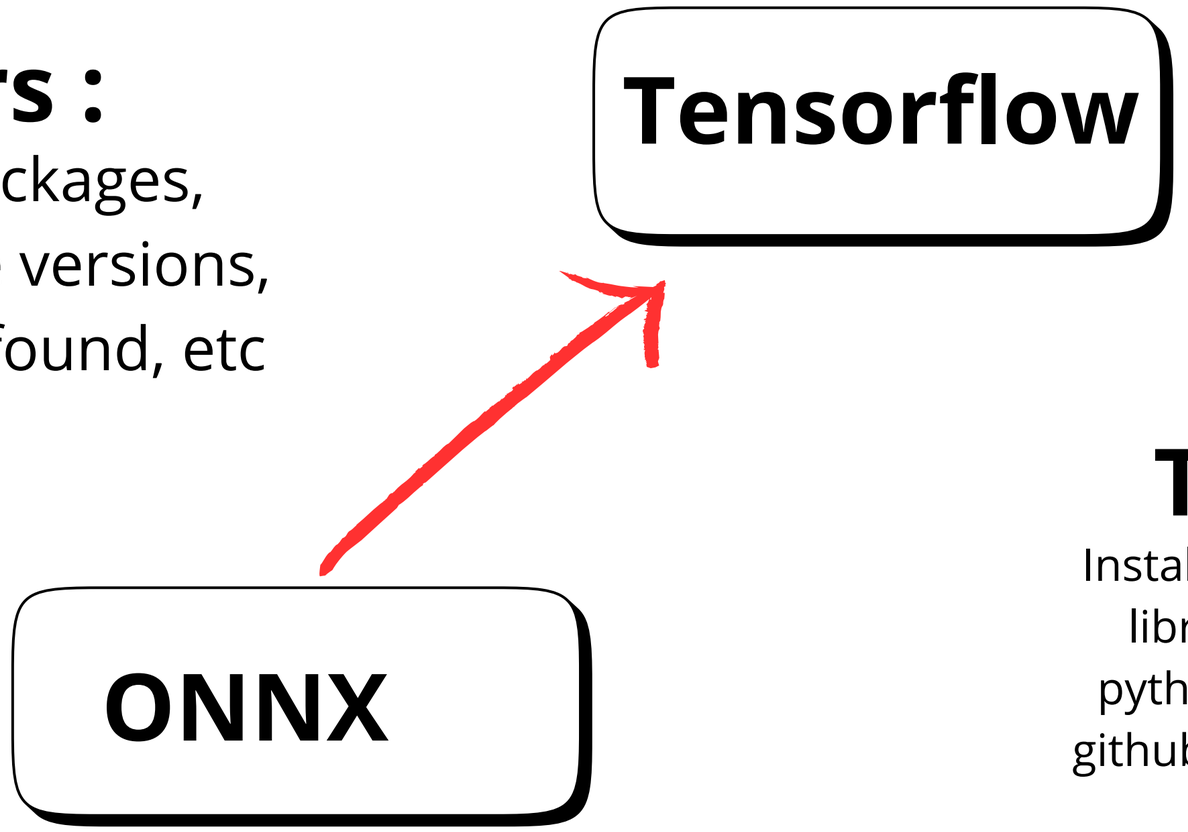
TPU : Incompatibility



TPU : Incompatibility

Errors :

Missing packages,
Incompatible versions,
module not found, etc



Tentatives :

Install previous versions of some libraries, different version of python, Installing libraries from github, different environnements, etc

Thank you for your attention!