

# **AI Based Electronic Component Identifier**

**Student:** Violet Concordia

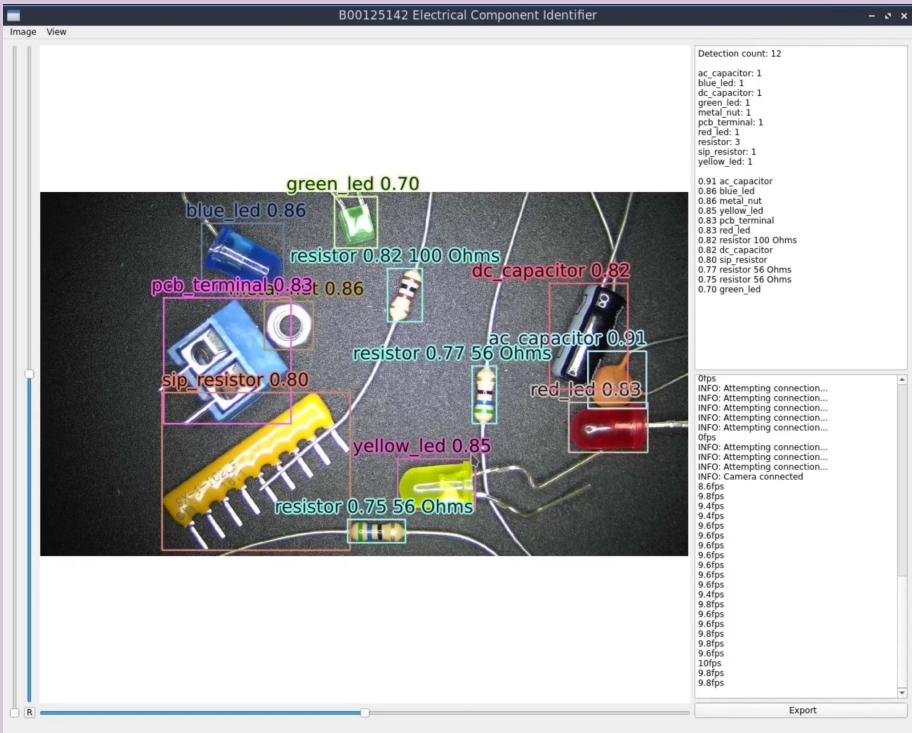
**Student number:** B00125142

**Supervisor:** Benjamin Toland

**Course ID:** TU807

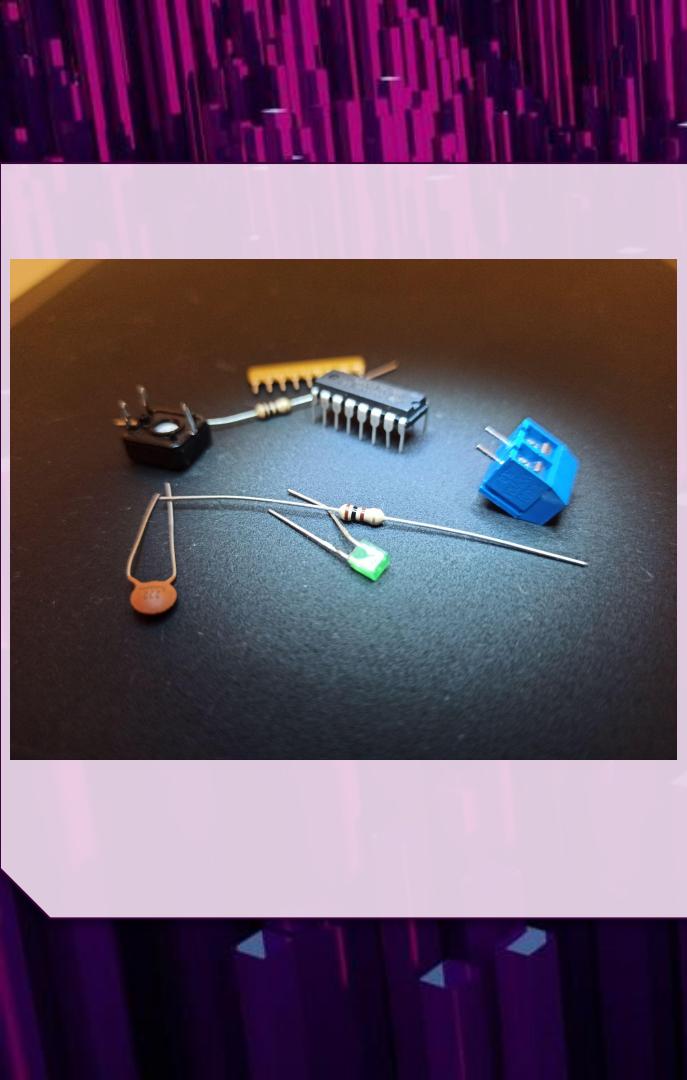
# Introduction

- Uses **AI** to detect objects from **images**.
- Detects **electronic components**.
  - **Identifies** their properties:
    - Type
    - Color/number code
    - Marking code
- Designed as a **tool for engineers**.
  - Assists bulk analysis.
- **Lightweight**.
  - Potential **mobile device** deployment.
- **Incredibly heavy to set-up**.
  - **Training** is done on a **powerful machine**.



# Object Detection

- Important in
  - Security
    - Notifying concerns
      - People
      - Animals
      - Flora
  - Production
    - Discarding defects
      - Damage
  - Analysis
    - Quality inspection
      - Scratches
      - Spots
    - Classification
      - Resistor
      - Capacitor
    - Total and class count



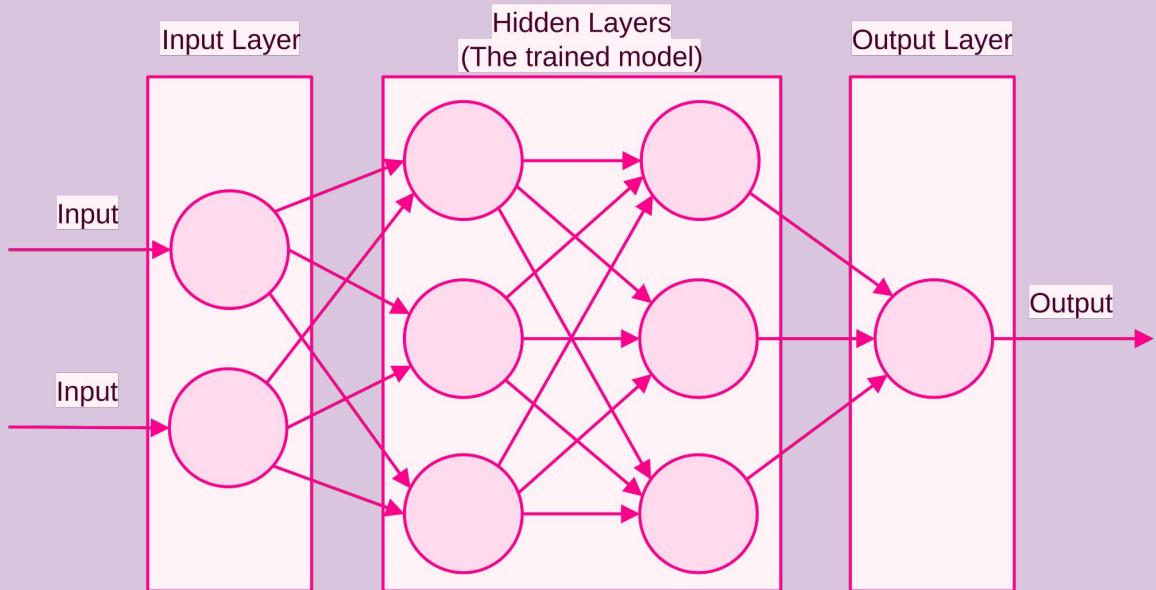
# The Problem

## Object detection from images:

- Natural to **intelligent** creatures.
  - Designed for **object detection** through evolution.
    - Second nature.
- **Binary data to computers.**
  - Has no concept of object, image, or color.
    - Everything is perceived the same.



# Simple Neural Network Diagram



**Chosen architecture**



# The Solution

## Neural Networks

- Complex networks of **neurons**.
- Each **neuron**:
  - Holds **weight**,
  - **biases** the output.

## Inference

- Utilises a **trained model** to **detect objects**.
  - Architecture used: **YOLO** (**You Only Look Once**) [1]

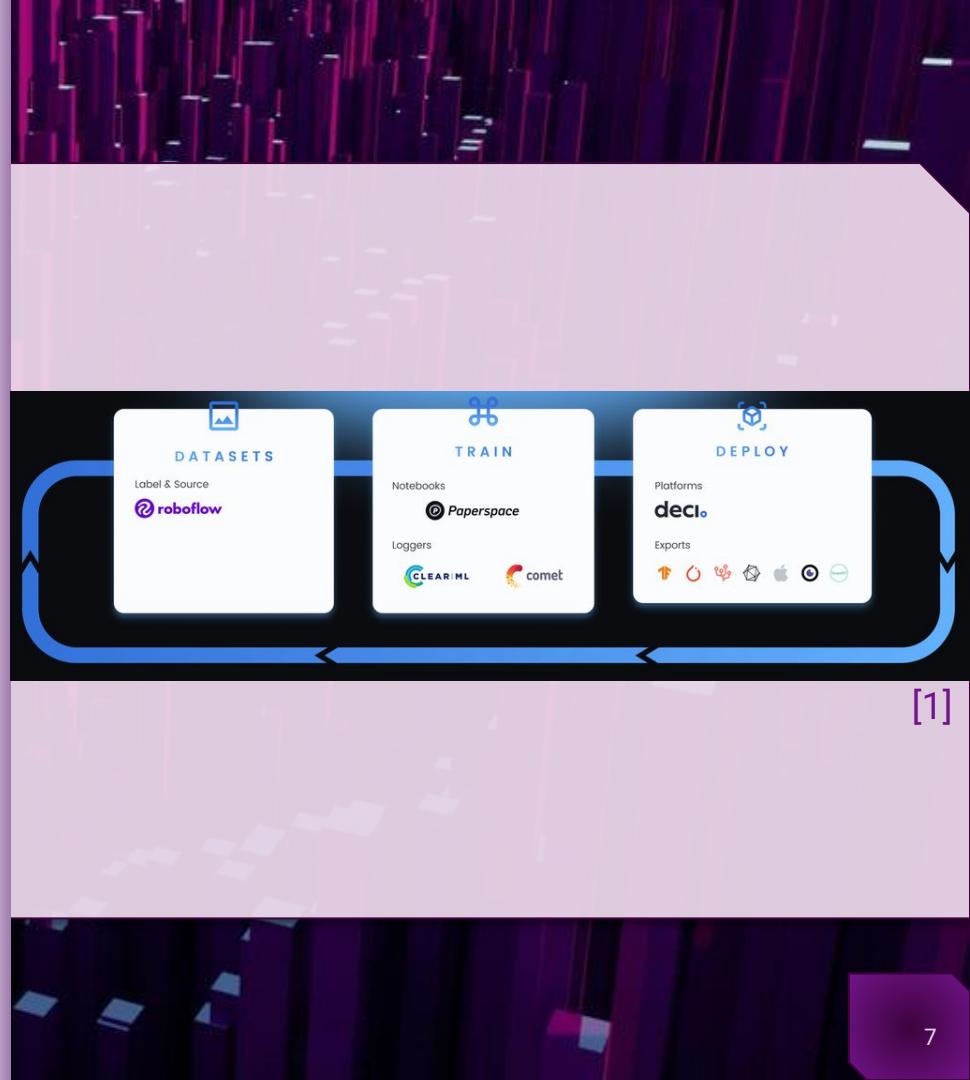
# Model Training

- **Images must be labeled manually.**
  - A **label** represents **where** and **what class** is contained.
- **Each picture** should have some **variation** in conditions:
  - **Position,**
  - **Rotation,**
  - **Lighting.**



# Model Training

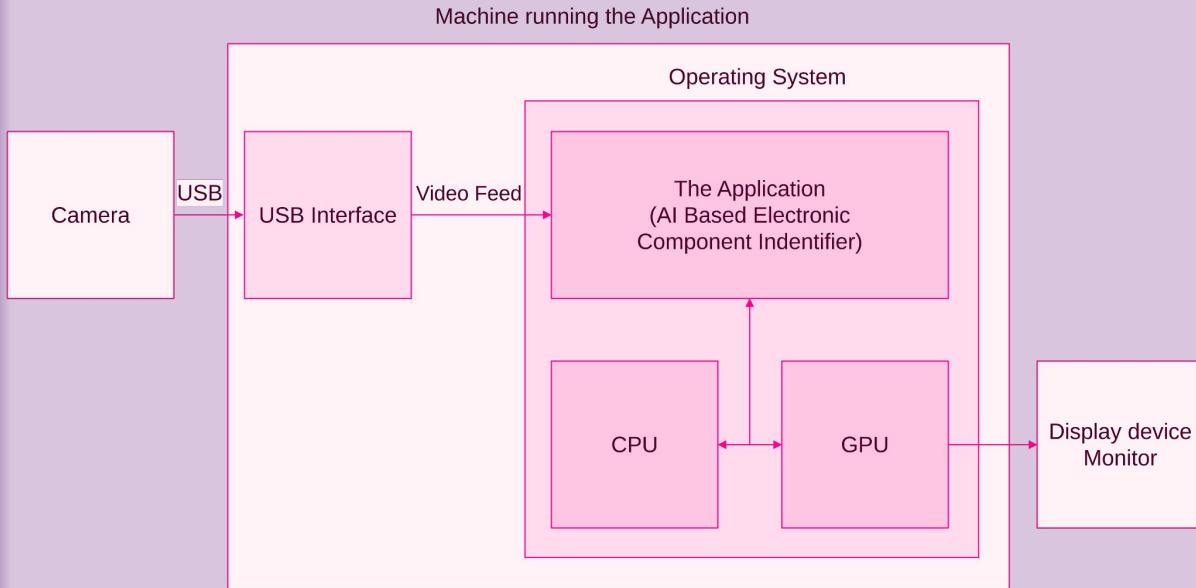
- Training requires **two labeled datasets** to train on:
  - Bigger: for **training**,
  - Smaller: for **evaluation**.
- Trained on over **3000** images.
  - Manually **labeled**.
  - **Classes:**
    - Resistor (Single, SIP)
    - Capacitor (AC, DC)
    - LED (Red, Green, Blue, Yellow, Clear)
    - Metal Nut
    - Integrated Circuits
    - Light Dependant Resistor
    - Diode



# System Block Diagram

## Concept Diagram

- **Camera** is External.
- Communication through **USB**.
- The application utilises both **CPU** and **GPU**.





# Post-Processing

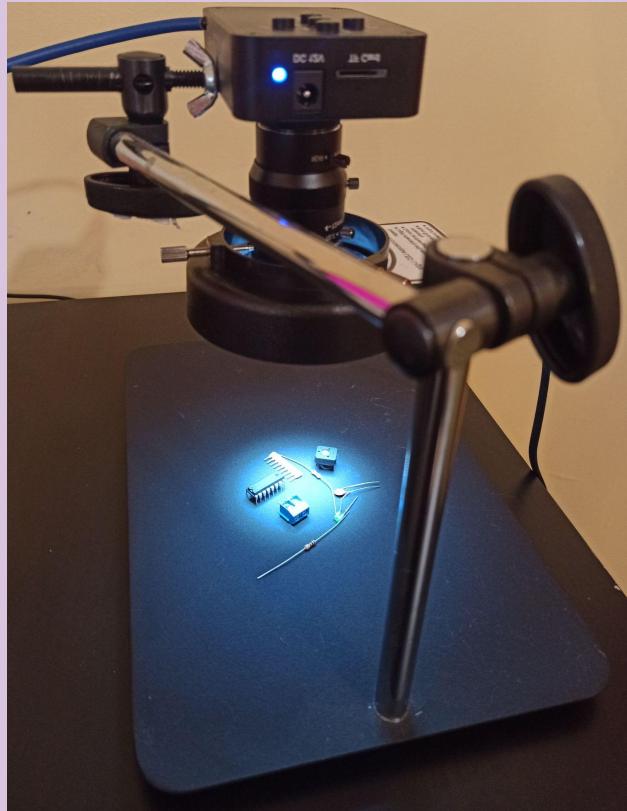
Additional processing of the **output** provided by the **inference**.

**Identification** of:

- resistor **color code**
- LED **color**

# State of the Project

- **Rig**
  - **Fully set up.**
  - Has an attachable **ring light**.
  - Used to **gather** all of the **training data**.
    - Taken over **300 labeled images** of **each class**.
- **Model Training**
  - Achieving confidence **up to 99%**.
    - Usually **above 80%**.
- **Inference with post-processing**
  - Running at **12fps**.



# Timetable

Name	Begin date	End date	Duration	Duration (Weeks, rounded up)
<b>Camera rig</b>				
Delivery times (Rig, Camera)	9/20/22	10/4/22	15	3
Physical rig setup with camera	9/20/22	10/3/22	14	2
<b>GUI</b>	10/4/22	10/4/22	1	1
Basic GUI (Console, smart camera connectivity, display, interface)	9/1/22	10/19/22	49	7
Camera feed to GUI interfacing (Includes resolution/framerate/frame handling, QoL methods)	10/5/22	10/19/22	15	3
<b>Research</b>	9/1/22	10/19/22	49	7
Optimal for inference camera research	8/1/22	1/19/23	172	25
AI object detection research and picking the most optimal architecture for the project (YoloV5 was picked)	9/1/22	9/12/22	12	2
Optimal camera rig research	8/1/22	9/19/22	50	8
Devices to potentially deploy on research	9/13/22	9/19/22	7	1
Color code identification from images research	12/6/22	1/4/23	30	5
<b>Post processing</b>	1/6/23	1/19/23	14	2
Optical Character Recognition (OCR)	1/20/23	3/1/23	41	6
Color code reading	1/20/23	3/1/23	41	6
<b>Object detection implementation</b>	1/20/23	3/1/23	167	24
First, basic dataset gathering (manual for early, asynchronous Inference testing) and model training	9/1/22	2/14/23	1	1
Bare minimum YoloV5 training and Inference implementation	10/5/22	10/5/22	60	9
Further data gathering and training of the model on various components	9/1/22	10/31/22	107	16
Deployment on a more mobile device	3/3/23	4/18/23	47	7

[2]

# Discussion

- The **project** runs entirely on **C++**, and is **cross-platform**.
- **Inference** is capable of extracting an **extremely high** level of detail from an input image.
- **Post-processing** is considerably more intensive than **Inference**.
- The **training** of the **model** for **300 epochs** took over **12 hours**.

# Conclusion

- The **inference** and **post-processing** results are more than **satisfactory**.
- While the **project** is mostly **software focused**, a **high quality rig setup** was essential in gathering a **high quality dataset**.
- The **project** has achieved its goals, and is **ready to be used**.

# References

- [1] YoloV5 <https://github.com/ultralytics/yolov5>, accessed on 6th of November, 2022
- [2] Gantt Project <https://www.ganttproject.biz/>, accessed on 6th of February, 2023



# The End

## Any questions?