Real-Time Symbol Matching

Neural Network Agent for Competitive Dobble Play

Computer Vision Project

Elsa Rún Ólafsdóttir, Eyþór Mikael Eyþórsson and Margrét Eiríksdóttir

Where we started

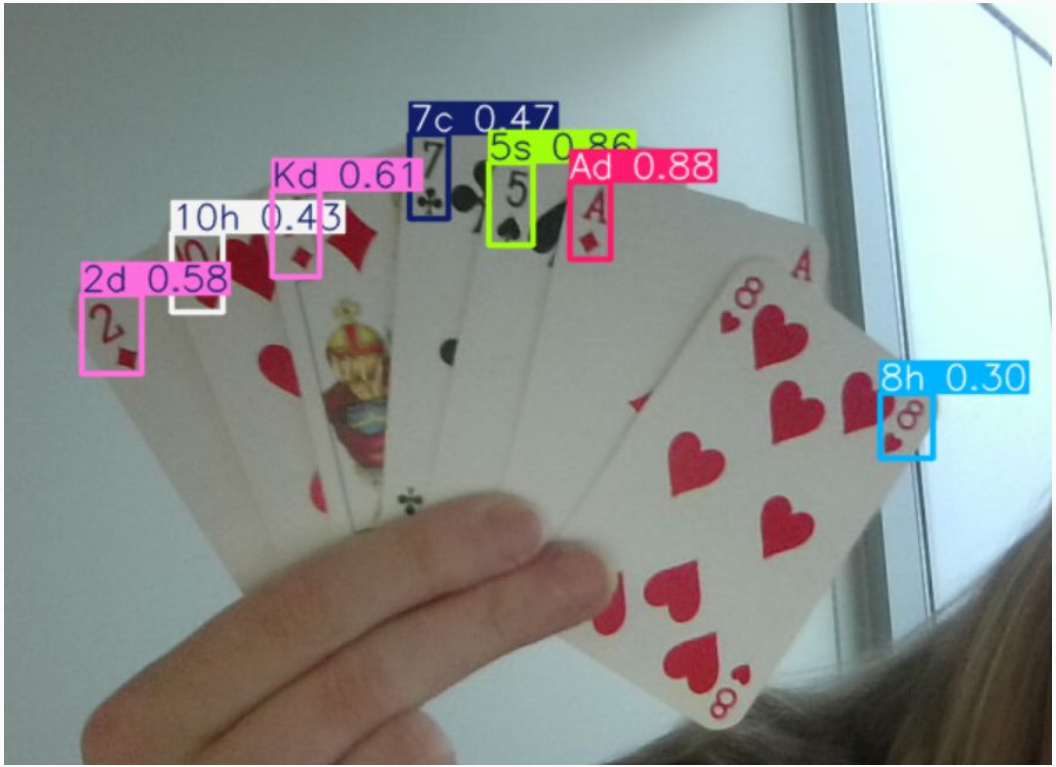
Playing card classification

- Detect rank, suit and bounding box of a hand of up to 7 playing cards anywhere inside RU in real-time

When the work started

- Found dataset on kaggle and notebook to train a model
- Model worked almost perfectly without any alterations needed
    - Only card orientation that had slight problems

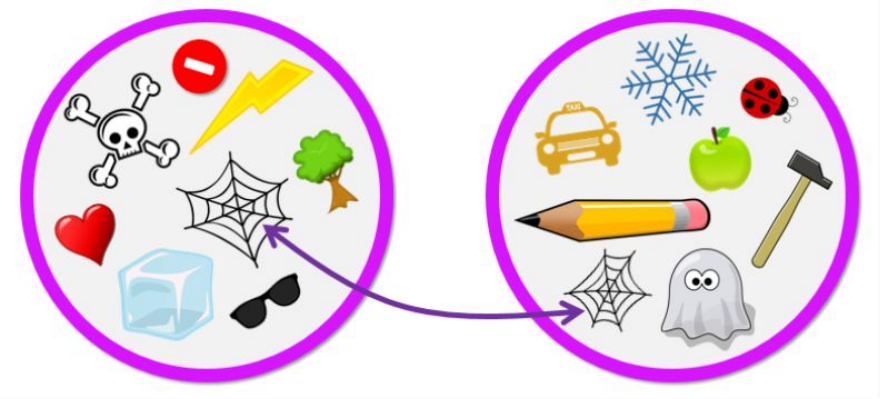
Need for broadening the scope -> Dobble card classification



An example output of the playing card detection

Dobble - Playing cards

- Each card contains 8 symbols
- 55 cards in total in the deck
- Always one, and only one symbol in common between any two cards
- In total there are 57 different symbols
- Many versions exist of Dobble



[1]

Project Idea

Identification

Accurately identify all of the images on the Dobble card

Find a match

Find matching symbols of known class labels between two or more Dobble cards

Generalize

Use a different method to find any matching symbols between two or more Dobble cards

Goals

Min goal

- ✓ Detect all learned symbols on two dobble cards and find the pair
- ✓ Model trained on pre-existing dataset found online on dobble cards, using YOLOv8

Max goal

- ✓ YOLOv8 model trained on our own dataset on dobble cards
- ✓ Pre-trained Siamese network
- ✓ Detect all learned and unlearned symbols on two dobble cards
  - ○ Match pairs of unlearned symbols

Background material

1

Paper - Playing Cards Classification and Detection using Sequential CNN Model

- Uses CNN model for playing card categorization, identifying both rank and suit
- Uses object detection models like YOLO, SDD and Faster R-CNN.
- These models generate bounding boxes and classify cards simultaneously

2

Article - Having Fun with YOLOv8: How Good Your Model in Detecting Playing Card?

- Uses YOLOv8 for playing card detection
- Includes instructions and examples
- Includes links to a dataset and a notebook for the code to train the model

3

Paper - Evolution of object detection methods.

- Yolo first proposed in 2015, breakthrough in real-time object detection.
- YOLO is a CNN-based one-stage object detection model
- Prioritized speed and simplicity, enabling faster object detection for practical, real-world applications.

[2], [3], [4]

Background: Resources used

OpenCV

Used for image capturing and feature extraction

PyTorch

Backbone of YOLOv8 and ResNet50

Google Colab

Used to train the model to have GPU access

Roboflow

Access to existing open access dataset and dataset creation

Ultralytics

yolov8n.pt model from YOLO packet used as base model in training

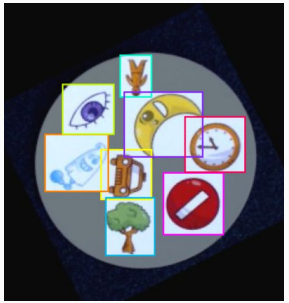
RU GPU cluster

Used to train ResNet50 on custom dataset

Dataset

Existing dataset

- Dobble dataset found on Roboflow
- Contains 39 classes
  - 18 missing
- 4422 train, 6 valid and 111 test images



[5], [6]

New dataset

- Includes all 57 classes of our dobble version
- 1640 labelled images
- 3699 train, 244 valid and 163 test images (4106 in total)
- Same background for all images
- Same augmentation and preprocessing steps used for both datasets:
  - Auto-orient
  - Stretch to 640 x 640
  - 3 Outputs per training example
  - 90° rotate: Clockwise, Counter-Clockwise, Upside down
  - Bounding Box brightness: -30% to 30%
  - Bounding box blur: Up to 2px

Dataset

Augmented custom dataset

- Created by cropping symbols from custom dataset
- 3962 labeled images
- 57 classes

```
31 0.25859375 0.50234375 0.196875 0.1390625
52 0.48515625 0.51796875 0.2453125 0.1625
11 0.740625 0.45078125 0.196875 0.10390625
6 0.24375 0.3890625 0.23671875 0.075
```

Dataset

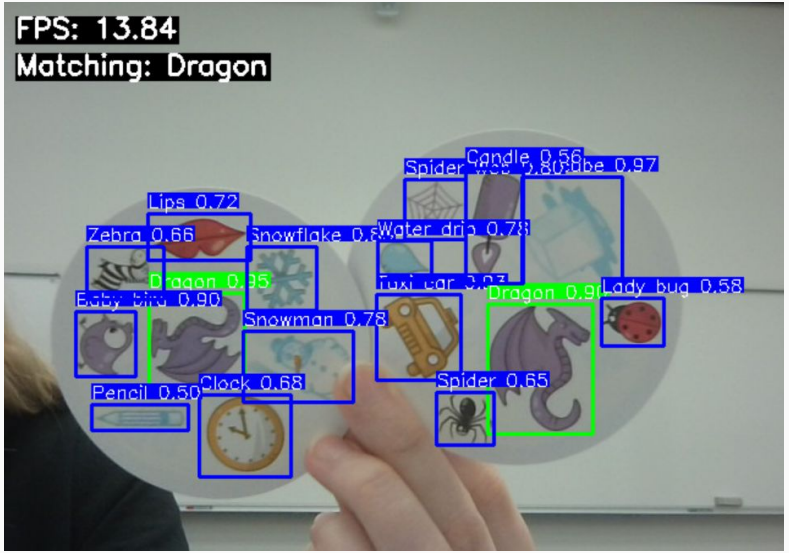
For transfer learning

- Cartoon dataset from Kaggle
- 10 classes
- 10.000 training images
- 2.000 testing images

Two different methods of matching cards

Object detection

- Yolov8 model
- Real-time video capture
- Identify different symbols
- Display bounding boxes, labels and confidences for each object
- If two or more objects belong to the same class in the current frame it is considered a match
- Green bounding box for a match
- Blue bounding box for other symbols
- Only works for symbols/class labels that the model knows



Two different methods of matching cards

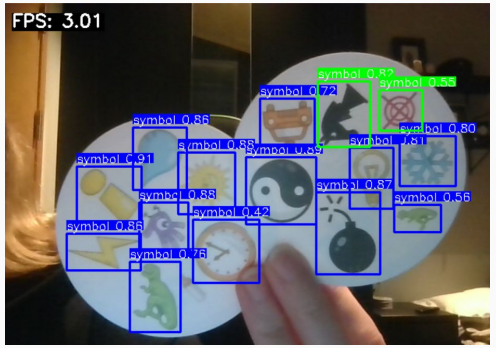
Pattern Matching

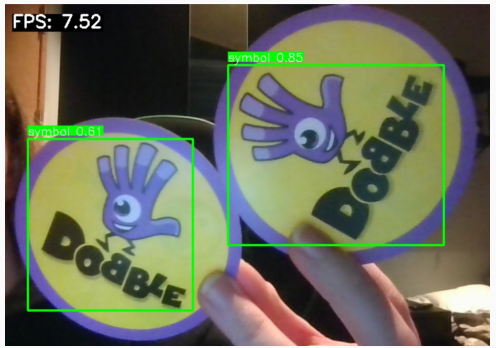
Goals:

- Match unlearned symbols

Method

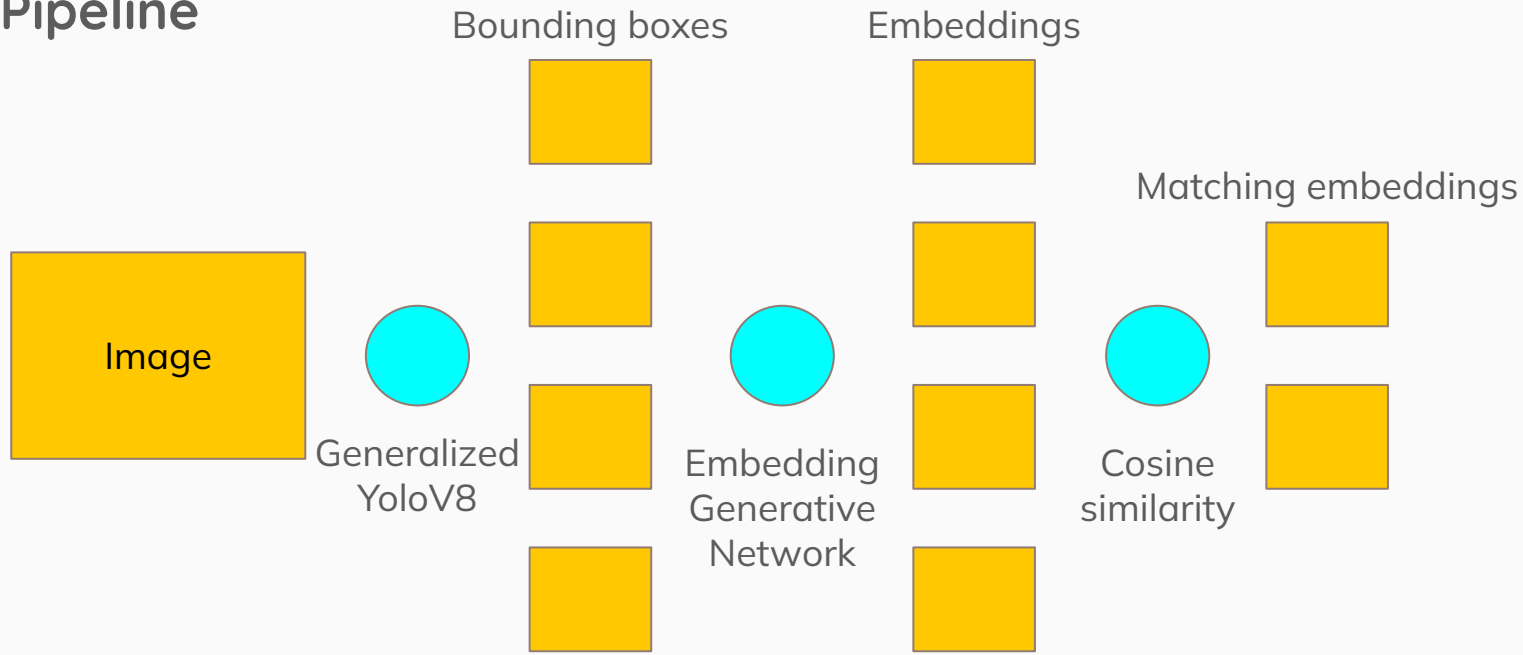
- Generalized object detection with Yolov8 (all labels changed to 0)
- Frames within bounding boxes fed to custom CNN to generate embeddings
- Cosine similarity of embeddings determines correlation
- Bounding boxes with highest correlation are displayed





Two different methods of matching cards

Pipeline



Performance measures and goals

For model:

- Confusion matrix
- F1-score: 0.8
- Recall: 0.8
- Precision: 0.8
- mAP for .70: 0.90

For real-time detection:

- Accuracy: 0.8
- Detection rate: 1.0
- FPS: 15

Performance of ResNet50 for classification

Trained on (10 epoch, lr=1e-4)

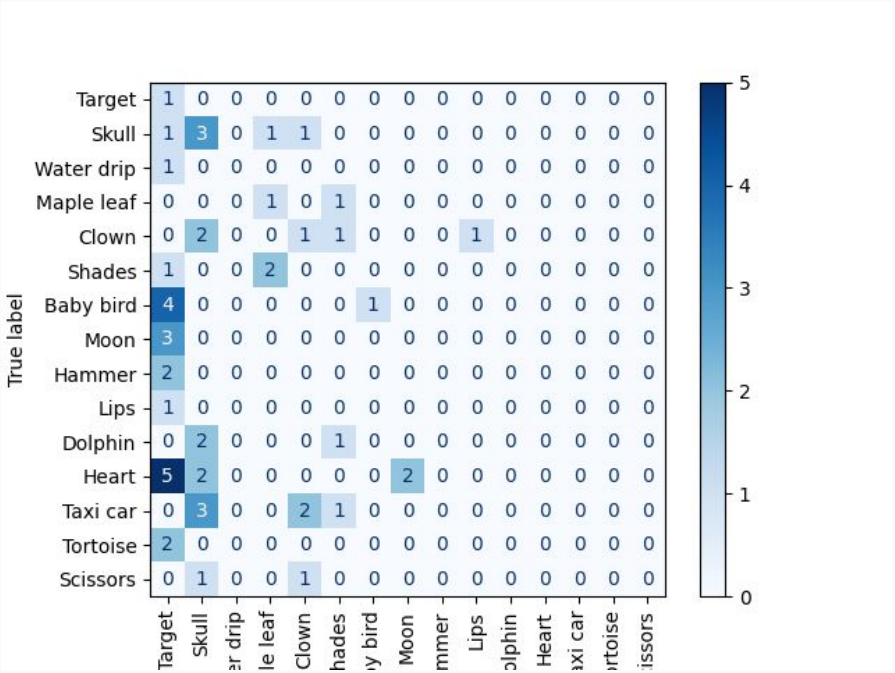
- ImageNetV2
- Custom dataset (augmented)

Tested on

- Custom dataset (augmented)

For confidence threshold 0.3

- F1-score: 0.0215
- Recall: 0.0347
- Precision: 0.0206
- mAP: 0.0679



Performance of ConvNeXt for classification

Trained on (6 epoch, lr=1e-3)

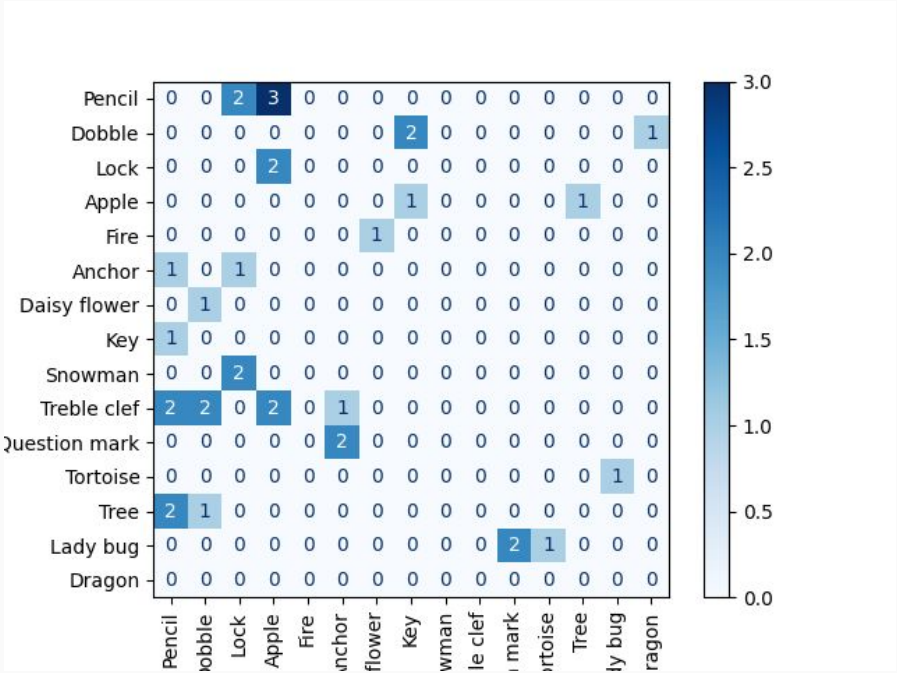
- ImageNetV2
- Custom dataset (augmented)

Tested on

- Custom dataset (augmented)

For confidence threshold 0.3

- F1-score: 0.0303
- Recall: 0.0636
- Precision: 0.0206
- mAP: 0.0571



Performance of ConvNeXt for classification

Trained on (20 epoch, lr=1e-3)

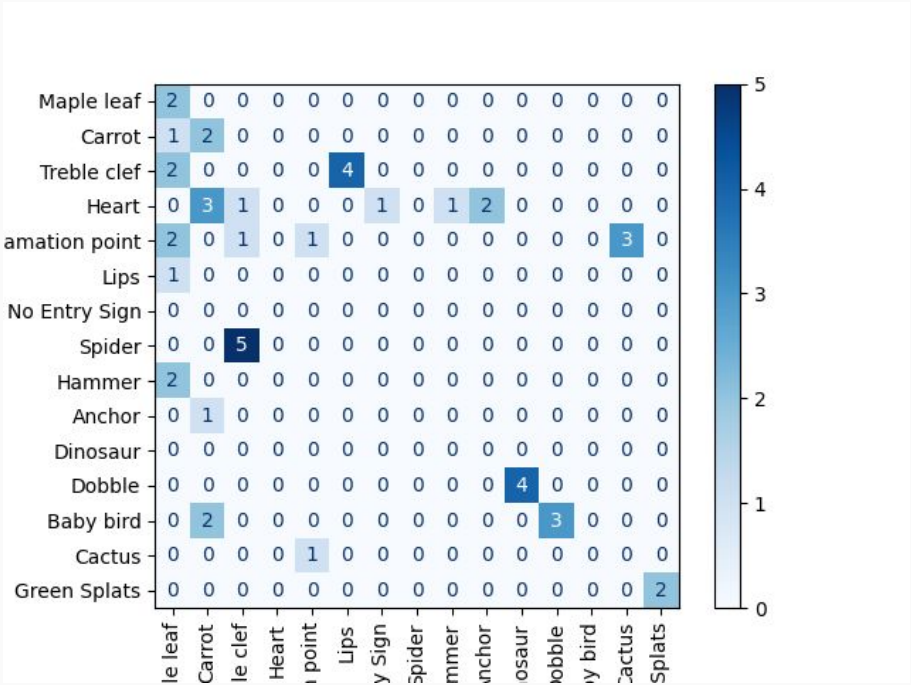
- ImageNetV2
- Custom dataset (augmented)

Tested on

- Custom dataset (augmented)

For confidence threshold 0.3

- F1-score: 0.0271
- Recall: 0.0462
- Precision: 0.0275
- mAP: 0.0687



Performance of ConvNeXt for classification

Trained on (12 epoch, lr=1e-3)

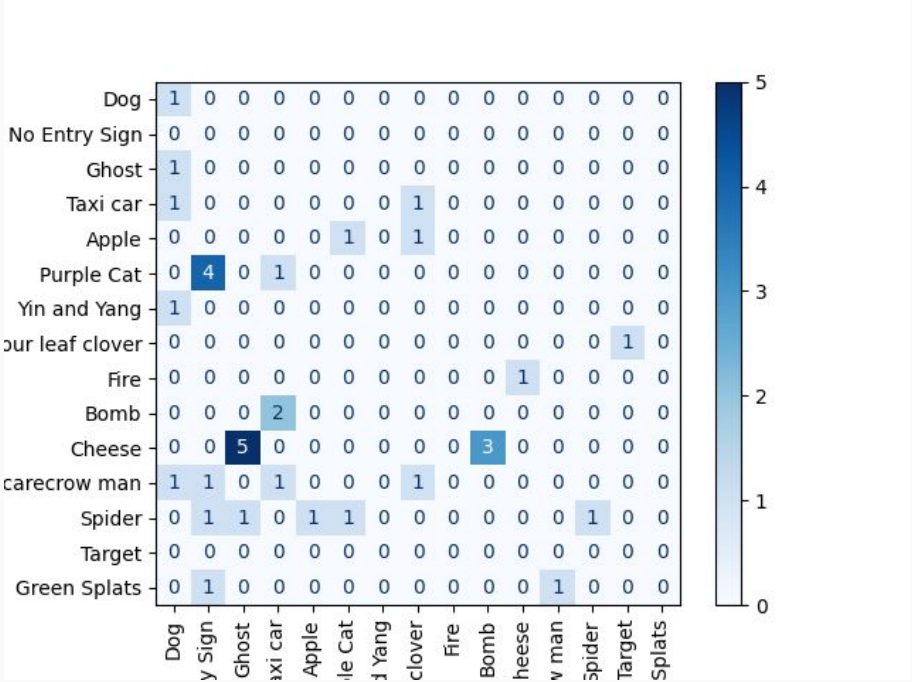
- ImageNetV2
- Cartoon dataset

Tested on

- Custom dataset (augmented)

For confidence threshold 0.3

- F1-score: 0.0163
- Recall: 0.0231
- Precision: 0.0160
- mAP: 0.0692

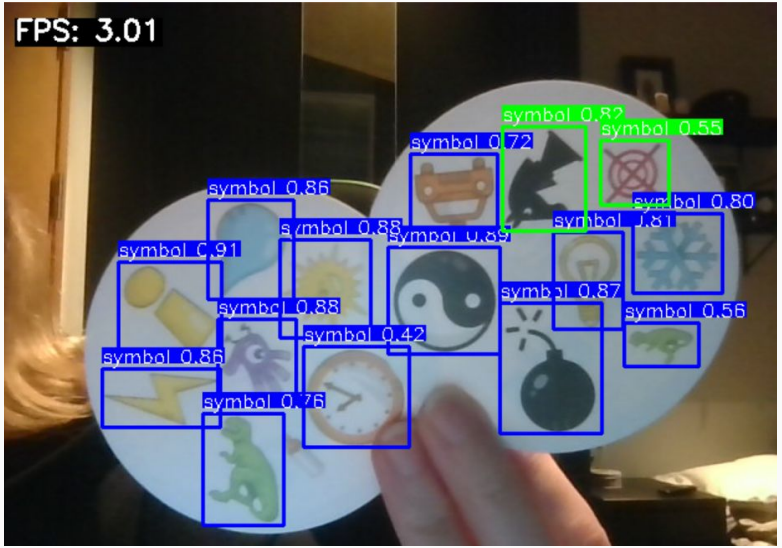


Performance

Pattern matching

Best case (ConvNext)

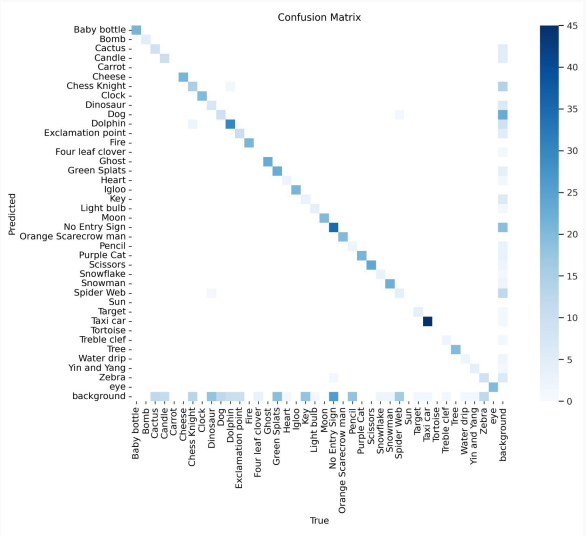
- Accuracy: 0.0
- Detection rate: 0.0
- FPS: ~5

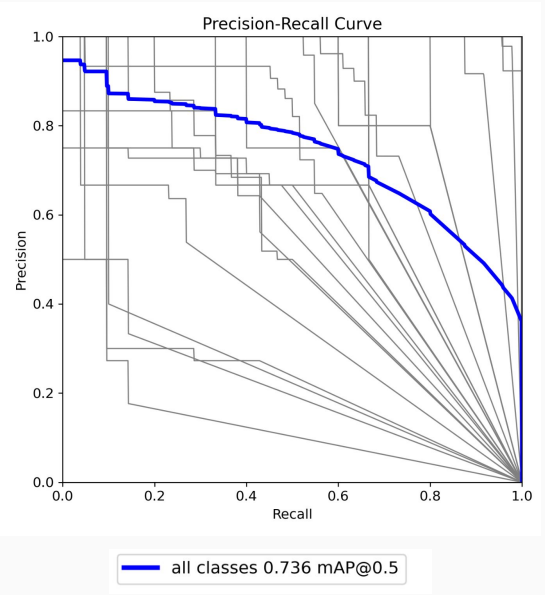


Performance of the 39 class model based on object detection method

For confidence threshold 0.7

- F1-score: 0.65
- Recall: 0.6
- Precision: 0.85
- mAP: 0.67

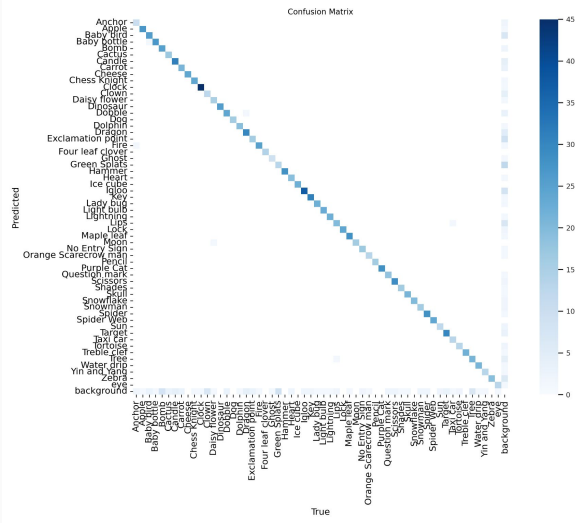


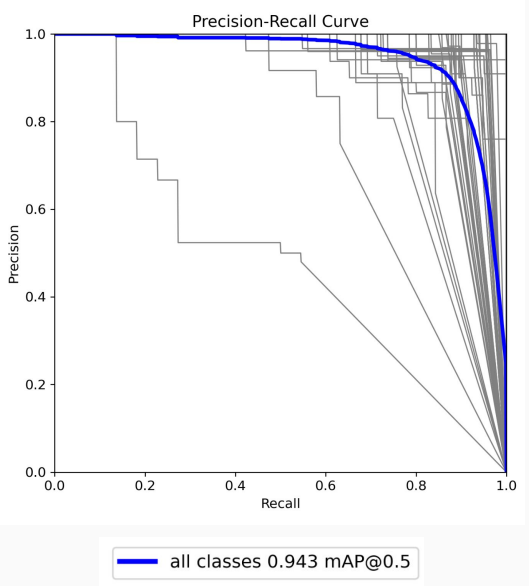


Performance of the 57 class model based on object detection method

For confidence threshold 0.7

- F1-score: 0.9
- Recall: 0.9
- Precision: 0.95
- mAP: 0.93

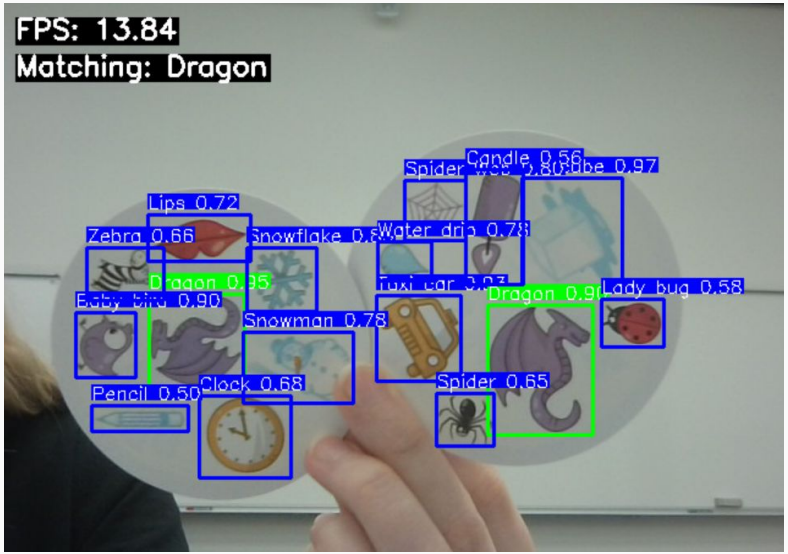




Performance

Object detection

- Accuracy: 1.0
- Detection rate: 1.0
- FPS: ~15



Code Demonstration

Work involved: Division of labor

Elsa

- Image capture for dataset
- Creation of new dataset
- Labelling images
- Report work
- Work on the object detection method

Eyþór

- Increase performance of general object detection network
- Set up and train classification networks
- Work on the pattern matching method

Margrét

- Image capture for dataset
- Labelling images
- Model training
- Detection and matching algorithm for object detection method

Work involved: Time and challenges

Challenges

- Balancing generalization and specialization
- No dataset found that included all the classes of our dobble version
- Labeling images for new dataset
- Maintaining open sessions on Colab to ensure finished runs

Time consumption

- Training datasets on Colab
- Labeling datasets in Roboflow
- Tweaking hyperparameters

Conclusion and future work

What worked well the first-time:

- Training the model and detecting symbols

What we would do differently:

- Start earlier to create our own dataset
- Spend more time on the pattern matching method
- Utilize transfer learning
- Look into ViT

Ideas for future work:

- Update model to use latest version of YOLO in training and compare performance
- Get more functionality into the pattern matching method
- Algorithmic implementations

Thank you!
Any questions?

References

[1] The Dobble Algorithm - 101 Computing, en-US, Dec. 2019. [Online]. Available: https://www.101computing.net/the-dobble-algorithm/ (visited on 12/08/2024).

[2] K. Mittal, K. S. Gill, R. Chauhan, M. Sharma, and G. Sunil, “Playing Cards Classification and Detection Using Sequential CNN Model Through Machine Learning Techniques Using Artificial Intelligence,” in 2024 International Conference on E-mobility, Power Control and Smart Systems (ICEMPS), Apr. 2024, pp. 1–4. DOI: 10 . 1109 / ICEMPS60684 . 2024 . 10559365. [Online]. Available: https://ieeexplore.ieee.org/document/10559365/?arnumber=10559365&tag=1 (visited on 12/02/2024).

[3] Having Fun with YOLOv8: How Good Your Model in Detecting Playing Card? | by Saskia Dwi Ulfah | Medium. [Online]. Available: https://medium.com/@sdwiulfah/having- fun- with-yolov8- how- good- your- model- in- detecting- playing- card- a468a02e4775 (visited on 12/08/2024).

[4] Y. Sun, Z. Sun, and W. Chen, “The evolution of object detection methods,” Engineering Applications of Artificial Intelligence, vol. 133, p. 108458, Jul. 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095219762400616X

[5] Dobble Cards Object Detection - v2 2022-06-12 9:15am, en. [Online]. Available: https://universe.roboflow.com/aravind-ellapu/dobble-cards-object-detection (visited on 12/08/2024).

[6] “dobble_card_data - v4 2024-12-10 6:56pm.” [Online]. Available: https://universe.roboflow.com/dobblecards/dobble_card_data/dataset/4

[7] “Cartoon Classification 2024-12-13 8:00am.” [Online]. Available: https://www.kaggle.com/datasets/volkandl/cartoon-classification