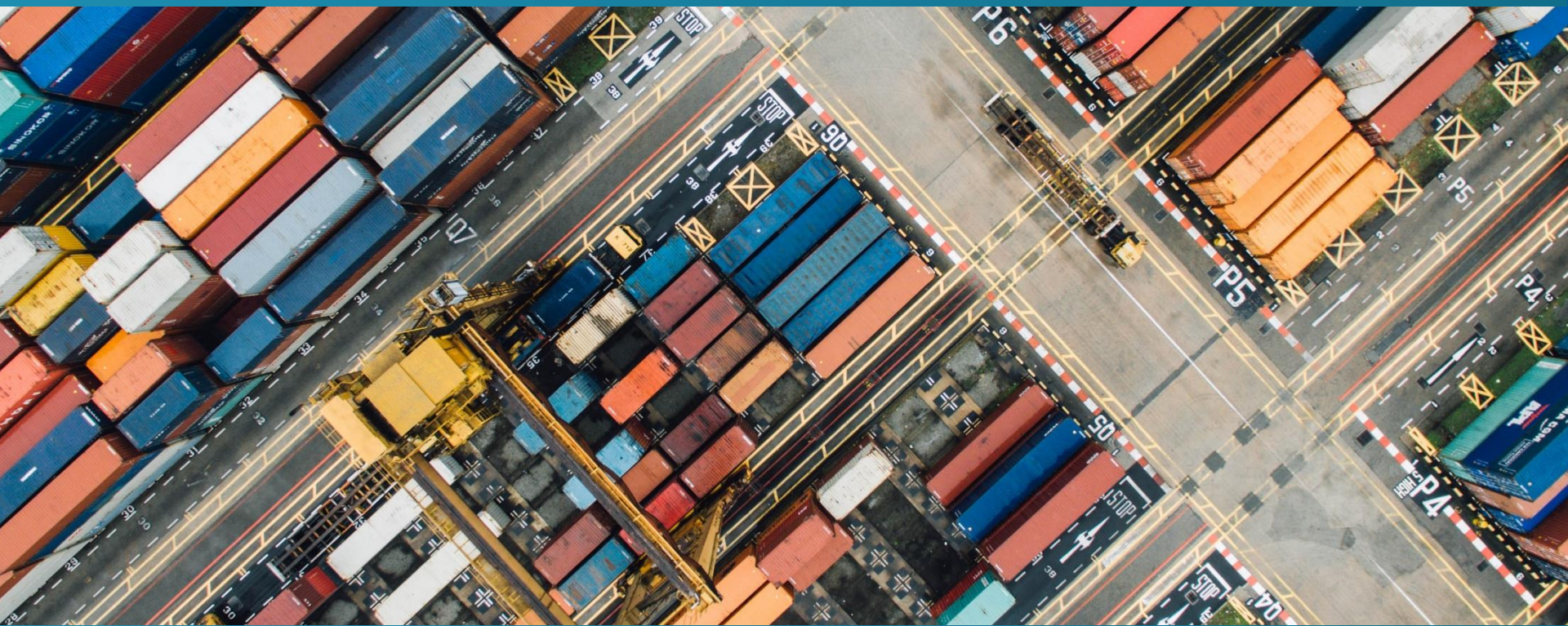


# Building Robust Industrial Applicable Object Detection Models Using Transfer Learning and Single Pass Deep Learning Architectures



VISAPP2018 – 27/01/2018 – Funchal, Madeira  
Steven Puttemans, Timothy Callemeyn & Toon Goedemé



**KU LEUVEN**

# Introduction

## Drawbacks during early deep learning years

- ✓ High computational cost
- ✓ Demanding and costly hardware (GPGPU)
- ✓ Need for enormous annotated datasets
- ✓ Adapting pre-built models to new object classes very challenging

## Deep learning now

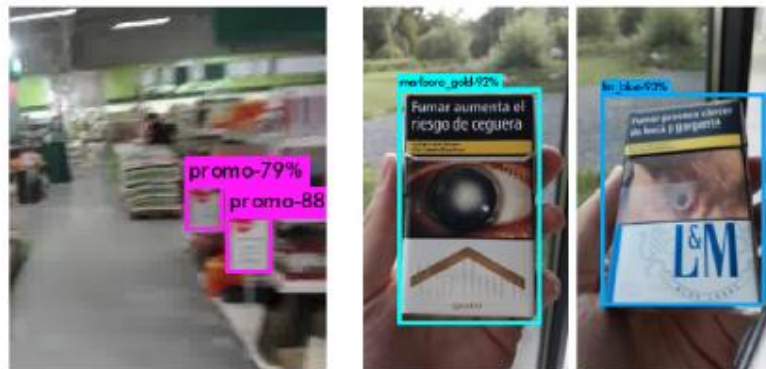
- ✓ Shift towards usability and real-time performance
- ✓ Affordable hardware available
- ✓ Model zoo's and a lot of publicly available datasets
- ✓ Transfer learning

# Introduction

Furthermore we noticed and explosion of stable and well documented open-source DL frameworks.

## Goal of this paper

- ✓ Investigate if non-traditional industrial cases are a good candidate for deep learned object detection, while ensuring real-time processing.
- ✓ Applied on 2 industrially relevant cases
  1. Detection of promotion signs in eye-tracking data to analyze shopping behaviour
  2. Detection of warehouse products for augmented advertisement purposes

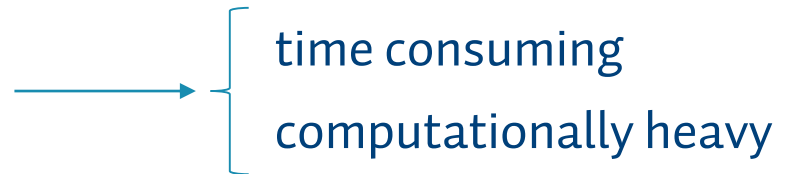




# Related work

## Deep convolutional neural networks (*Densenet201, InceptionV3*)

- + multi-scale image pyramid
- + sliding-window based processing



## Introduction of region proposal networks (*R-CNN*)

- ✓ Separate shallow net + mutual information → limited regions
- ✓ Possibility of running complex deep architectures at faster speed

## Improved approach: single shot detectors (*SSD, YOLOv2*)

- ✓ Include the region proposals into the pipeline
- ✓ Single pass of a complete image for both bounding box prediction and class probability generation

# Related work

If your application is close to academic classes, than you are good to go with pre-trained deep models.

If NOT, we need to apply transfer-learning

- ✓ Re-use convolutional layers and pre-trained weights
- ✓ We know they capture general features of several object classes
- ✓ Only the last convolutional layers combine these features
  - So let us re-train these layers with case specific data

Limited case-specific training data

- ✓ Industry moto: the less the better
- ✓ Using data augmentation we can extend this set drastically

# Framework

For training our case-specific deep learned object detectors we use the following set-up

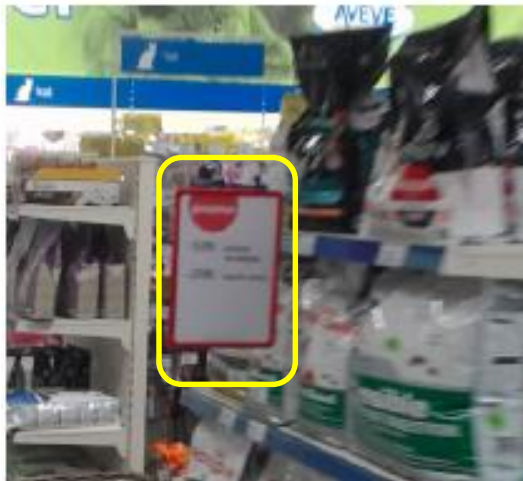
- ❖ Darknet framework (<https://github.com/pjreddie/darknet>)
- ❖ YOLOv2 architecture (YOLO9000 paper)
- ❖ Own version (<https://gitlab.com/EAVISE/darknet>)
- ❖ 2x Intel(R) Xeon(R) CPU E5-2630
- ❖ 32GB RAM
- ❖ NVIDIA Titan X (Pascal) GPU



# Datasets & framework

## Case 1: AVEVE – Belgian warehouse

- ✓ Costumers asked to shop with head-mounted eye-tracker
- ✓ Two object classes:
  - a small promotion board (*small\_sign*): 75 training samples / 420 test frames
  - a large promotion board (*large\_sign*): 65 training samples / 960 test frames



small\_sign



large\_sign

# Datasets & framework

## Case 2: cigarette box detection

- ✓ Presenting a cigarette box to the camera of your cellphone
- ✓ Locate the box and identify the brand
- ✓ Two models: general cigbox detector & brand-specific cigbox detector

Table 1: Number of cigarette samples per brand.

Label	Training	Validation
marlboro_red	63	163
marlboro_blue	14	149
marlboro_gold	29	157
marlboro_touch	14	167
chesterfield_red	75	157
chesterfield_blue	15	222
heets_gold	15	185
heets_red	15	188
heets_blue	15	229
lm_red	46	183
lm_blue	30	146
lm_gold	15	179
philip_morris_yellow	15	178
philip_morris_red	15	179
<b>TOTAL:</b>	<b>376</b>	<b>4279</b>





# Suggested approach

Started from the original YOLOv2 architecture on Pascal VOC 2007

- ✓ Physically change the network architecture
  - ❖ Force the number of output class to our needs
  - ❖ Adapt the number of filters in the final convolutional layer
  - ❖  $\text{\#filters} = (\text{\#classes} + 5) \times \text{\#anchors}$
- ✓ Adapt the anchor ratios (for bounding box prediction)
- ✓ Initialize weights of all but last convolutional layer with pre-trained weights. Final convolutional layer is initialized randomly.

Generally our transfer-learned models converge over multiple thousands iterations (*for most cases 10.000 iterations seems to do fine*), using 64 sample batch size at each iteration.

# Suggested approach

During training we allow the following data augmentations to our limited set of training samples

- ✓ Random rescaling (max. 30%) of the input size of the conv1 layer
- ✓ Random flips around the vertical axis
- ✓ Random deviation (max. 10%) of the hue pixel values
- ✓ Random saturation and exposure deviation (max. 50%)
- ✓ Random jitter/crop on the training samples annotations (max. 20%)

## Training time

- ✓ With a 416 x 416 input resolution
- ✓ Converging models mostly within 12 hours of training (overnight)
- ✓ Halting training when loss rate drops under 0.5, which seems to be a stable point for our applications

# Suggested approach

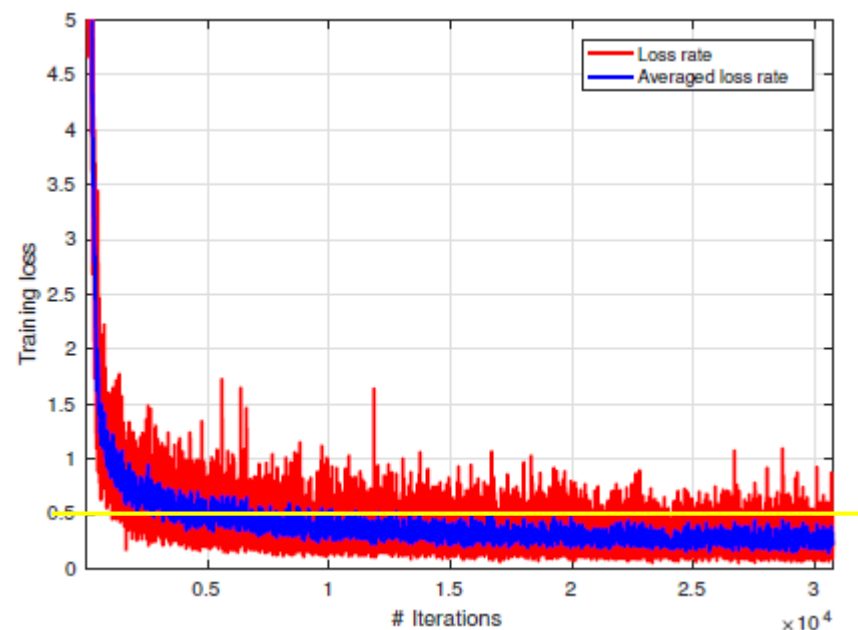
## Stopping criteria

- ✓ Loss rate
- ✓ Average loss rate

## Memory footprint

- ✓ All models from same basic architecture YOLOv2
- ✓ 270MB storage
- ✓ 450MB memory footprint

Watch out for overfitting, which is rather common due to the limited training dataset.



# CASE 1: AVEVE promotion boards

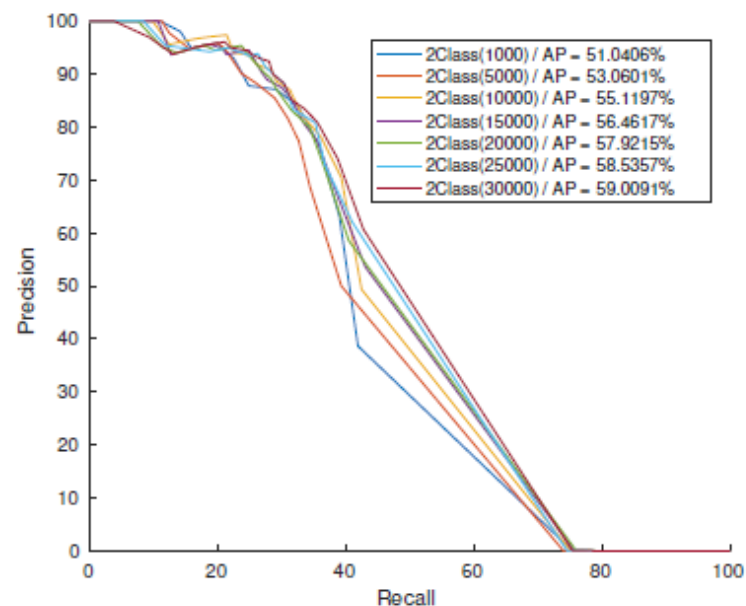
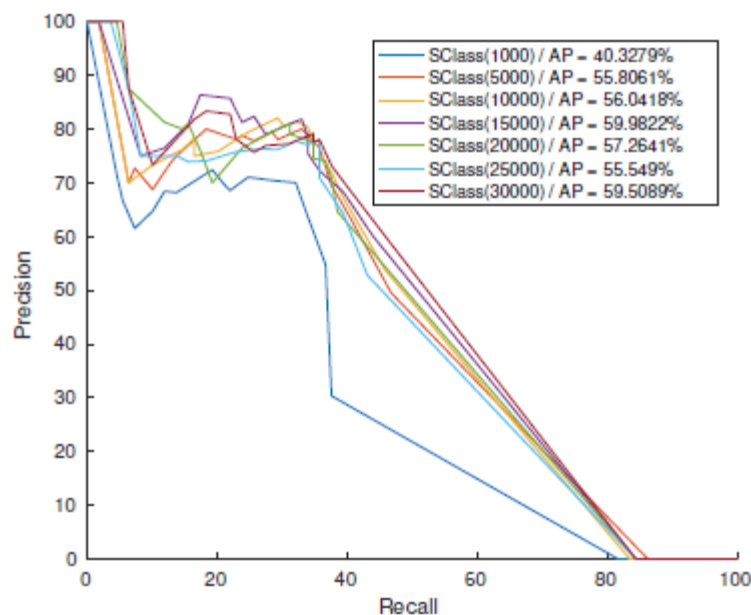
We have a look at two main questions

1. Can we analyse complete eye-tracker experiments by linking the output of deep learned object detectors to eye-tracker data, removing the need for manual analysis?
2. Can we reduce the amount of annotations as much as possible without losing too much average precision on the obtained detectors?

We built two separate models

- ✓ Single-class detector for small\_sign
  - Is it possible? Does it work?
- ✓ Two-class detector for small\_sign and large\_sign combined
  - Can we combine detection and classification?
  - Using a single run through a single model?

# CASE 1: AVEVE promotion boards



- ✓ Evaluation at different iterations of training
- ✓ Raising the iterations yields an increase in average precision on the validation set
- ✓ Once the average precision drops again, we halt further training



# CASE 1: AVEVE promotion boards

We reckon that our models seem to be underperforming

- ✓ 59,98% average precision is not great
- ✓ Possible reasons:
  - Heavy motion blur of the recorded eye tracker data
  - Unlearned object dimensions (different from training data)
  - YOLOv2 has issues with detecting very small objects in relation to the image dimensions it is located in, while this is no issue for annotater

Given the context, eye-tracker data analysis

- ✓ This is more than good enough
- ✓ We only need to know if and when a costumer has noticed the sign, so if the costumer heads closer to the sign, we actually detect the sign robustly

# CASE 1: AVEVE promotion boards

## Evaluation / inference speed

- ❖ NVIDIA Titan X GPU
- ❖ 55 FPS at 1280 x 720 pixel resolution



# CASE 2: warehouse product packages

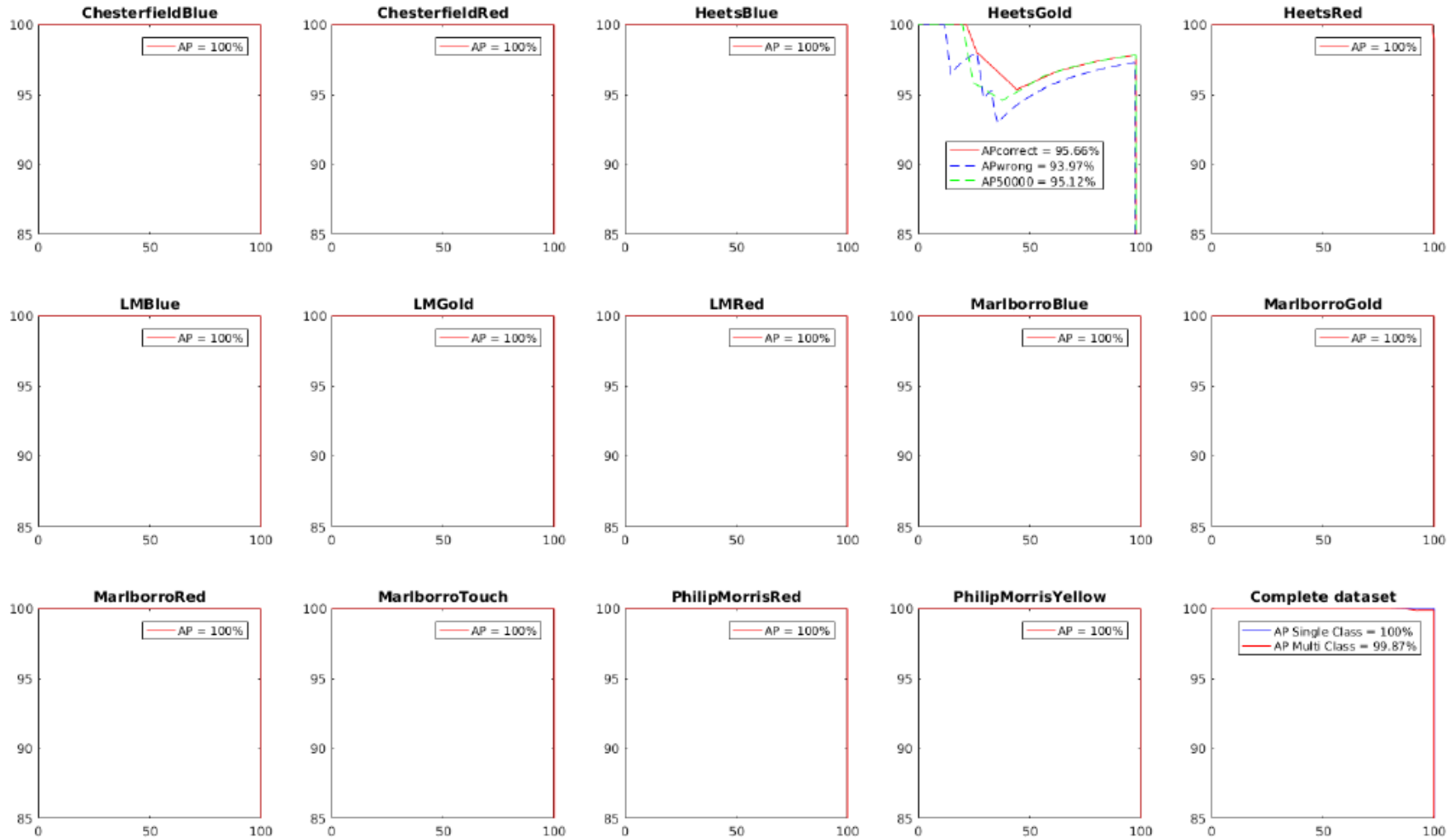
Is it possible to robustly locate the packages and possibly return the brand simultaneously?

- ✓ We apply this as a cigarette box detection case
- ✓ All packages have general cigarette box properties
- ✓ The print on the boxes differentiate the brands

We again build two different models

- ✓ A general cigbox detector, ignoring the brand itself
- ✓ A 14-class brand specific cigbox detector, incorporating the brand

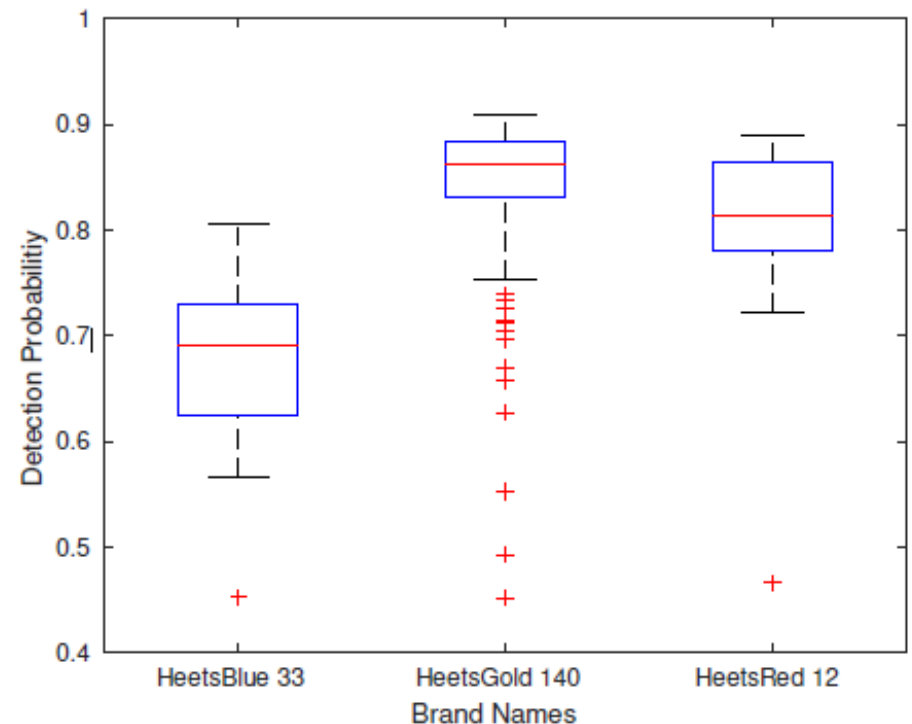
# CASE 2: warehouse product packages



# CASE 2: warehouse product packages

Drop in accuracy for 14-class detector generated by a single brand HeetsGold, thus we looked into the reason behind it.

Visual inspection of classes shows that there are less brand specific details, except from a very small range of coloured stripes.





# CASE 2: warehouse product packages

## Evaluation / inference speed

- ❖ NVIDIA Titan X GPU
- ❖ 70 FPS at 1280 x 720 pixel resolution



# Discussion

## Drawbacks on using YOLOv2

- ✓ Unable to robustly detect very small objects
- ✓ Grid and probability based proposal system sometimes forces neighbouring regions to group together as single detection

We notice that our cigbox detector has issues with rectangular shapes that are not cigboxes.


- ✓ This is expected, since we never provided other boxes as learning data
- ✓ By adding some hard negatives (other rectangular shaped boxes) we drastically reduce the class probability outcome for those objects
- ✓ This allows setting a firm threshold to remove these detections

# Discussion

We proved that

- ✓ Using deep learning for industrial object applications is feasible
- ✓ Using off-the-shelf pre-trained weights and transfer learning
- ✓ On affordable hardware, like the Titan X (Pascal)
- ✓ We do not always need a 100% succesful classifier (promotion boards)
- ✓ Because we can use the context and needs of the experiment
- ✓ It is possible to have accurate and robust detectors with minimal training data, up to only 15 labelled images per class.





Thank you for your attention!

Any questions?