

# Fruit & Vegetable Recognition at Colruyt

Hendrik D'Oosterlinck



That's me!



Als abonnee kan je dit plusartikel lezen

09/08/2019 om 03:00 door Nina Bernaerts Joerie Dewagenaere

## Slimme camera weet wat u koopt: Colruyt scant groenten en fruit om afrekenen vlotter te laten verlopen

Winkelen



Elon Musk   
@elonmusk



If advanced fruit recognition hasn't been applied to manipulate social media, it won't be long before it is

♥ 76.5K 7:55 AM - Sep 26, 2019



💬 10.2K people are talking about this



**data**news

Rubrieken ▾

Het magazine

Voordelen  
voor abonnees

Abonneren

## Colruyt test slimme weegschaal die groenten en fruit herkent

nieuws

## Colruyt lanceert weegschaal die met A.I. groenten en fruit herkent

door Wouter Scholliers · 9 augustus



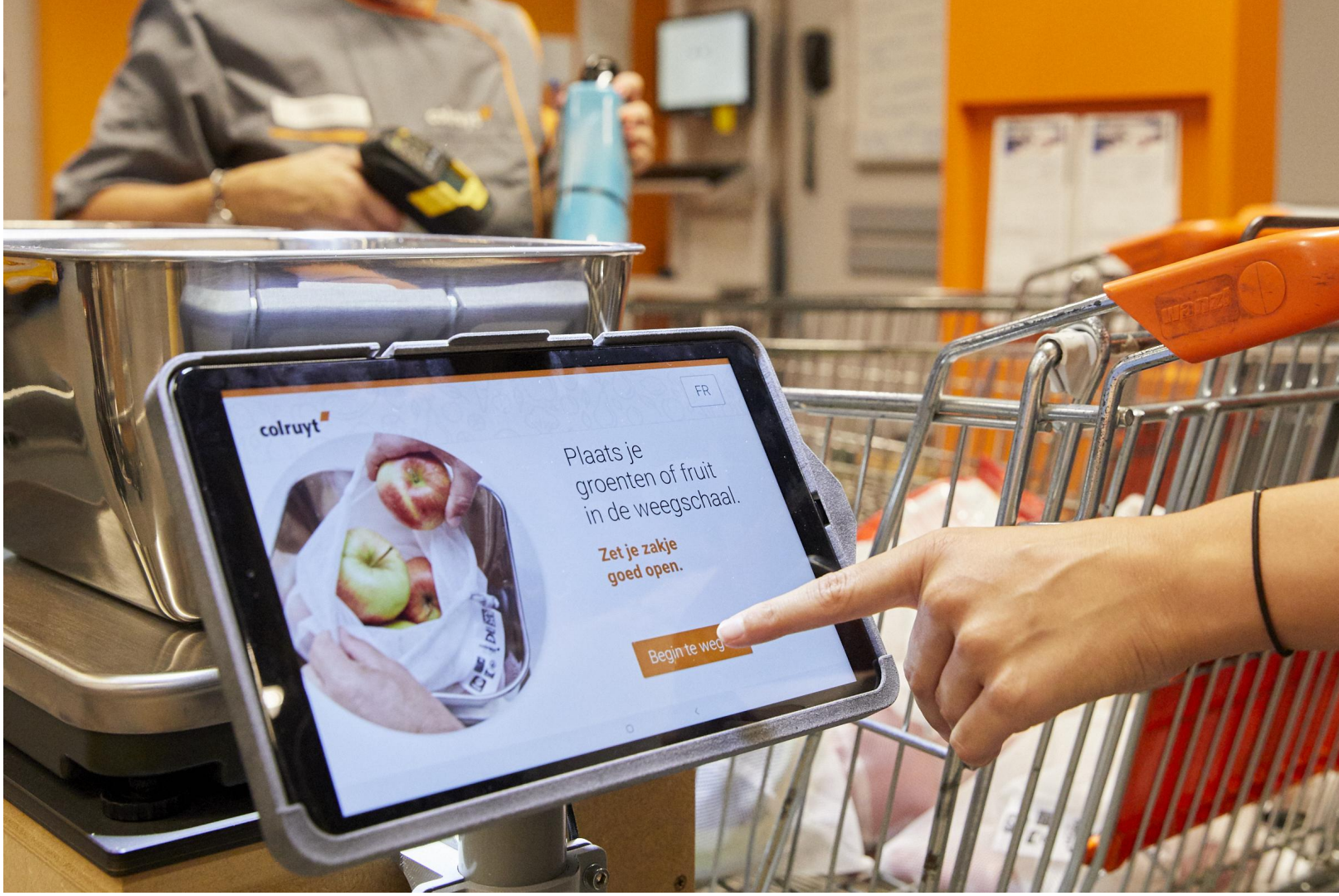
economisch nieuws

## Colruyt test weegschaal die groenten en fruit herkent

08/08/2019 om 12:12 door evg | Bron: BELGA



Foto: Jimmy Kets



colruyt

FR

Plaats je  
groenten of fruit  
in de weegschaal.

Zet je zakje  
goed open.

Begin te wegen...

# Challenges at checkout

Fruits and vegetables checkout is slow

There's lots of confusion between similar products

To avoid this confusion, produce is wrapped in plastic

Can we solve this using computer vision?

# Solution



Classify product, bagtype, closed bag

Data captation/inspection system

Fully retrainable, editable

Automatic backups, monitoring

# In sandbox

Jupyter Notebook + Tensorflow =




1. Download a toy dataset
2. Train a neural network
3. Apply to Colruyt data

But still missing some production features:

1. Maintain & retrain without an AI-team
2. No user control, backups
3. No cameras, no monitoring, no API, ...



# In production

Jupyter Notebook + Tensorflow = 

Provide API to capture and predict  
Retrain, review labels  
Manage data, backups, users  
Full on-premise deploy



User, cash register communication  
Prediction triggering



Monitor and alert





## Pipelines

Prepare pipelines. Create models through labeling and training or select existing models.



## Deployments

Deploy pipelines. Manage deployed inference and training pipelines.



## Packages

Manage pipeline packages.



## Storage

Set up & manage S3 storage.



## Accounts

Create & manage users.



## Backups

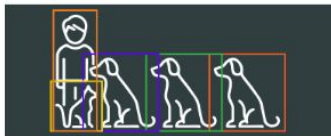
Backup & restore.



## Image Classification

Global labels

What is shown in this image?



## Object Detection

Bounding boxes

What items are in this image and where are they located?



## Instance Segmentation

Masks

What items are in this image and how are they shaped?



## Semantic Segmentation

Masks

What type of object are these shapes?



## Dummy Source

Any type of label

DummySource

# Data captation

- Set up lighting and cameras
- Captation app to take and review images
- Colruyt takes images at their test store
- After roll-out, customer interaction provides new images, classes and labels

# Data captation



# Robustness test @ Robovision

	Test store dataset	In production
Background	Standardized	Varying
Camera	Standardized	Varying
Lighting	Standardized	Varying
Focus	Standardized	Varying

Can't handle different setup at Robovision, real-life prediction are terrible at first.  
What if conditions change a bit during production? How do we deal with this?

# Data Augmentation



# Correlated data

For instance, nearly all mangos:

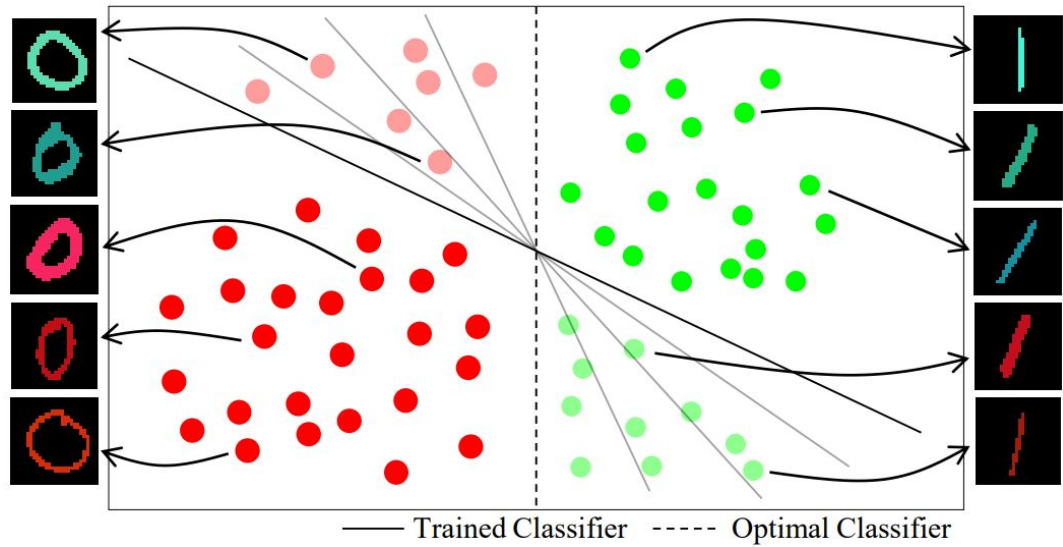
- Captured on the same day
- In the same weighing scale
- One of the lights wasn't working as intended

For this weighing scale 50% of the pictures were mangos

For this lighting condition 99% of the pictures were mangos

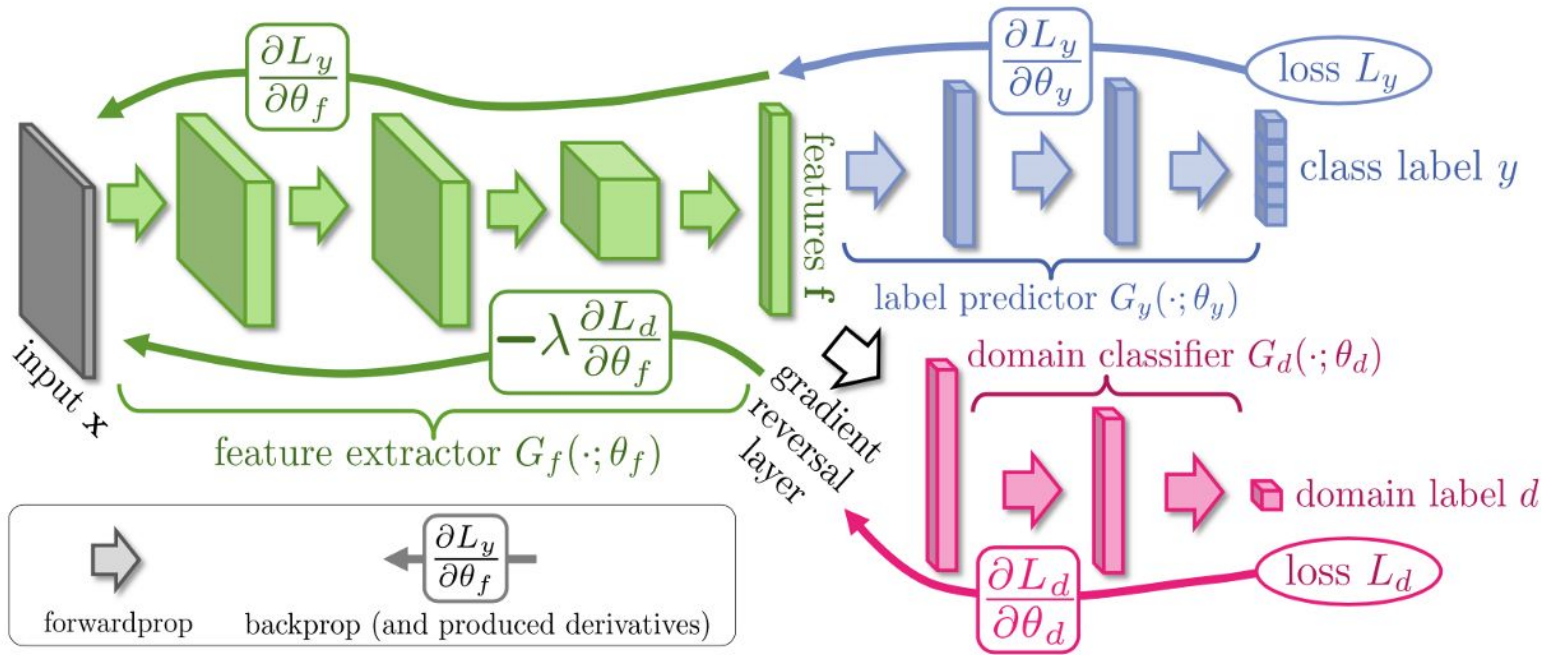
If same light breaks again, model is heavily biased towards predicting mango.

How do we learn *not* to use these features for prediction?





# Help from domain adaptation...



# Too clean training dataset

What will the model predict the first time it sees hands, fingers?



*the limits of augmentation*

Continuously retrain with production data once deployed

Capture a small & dirty testset to know what your real-life performance will be

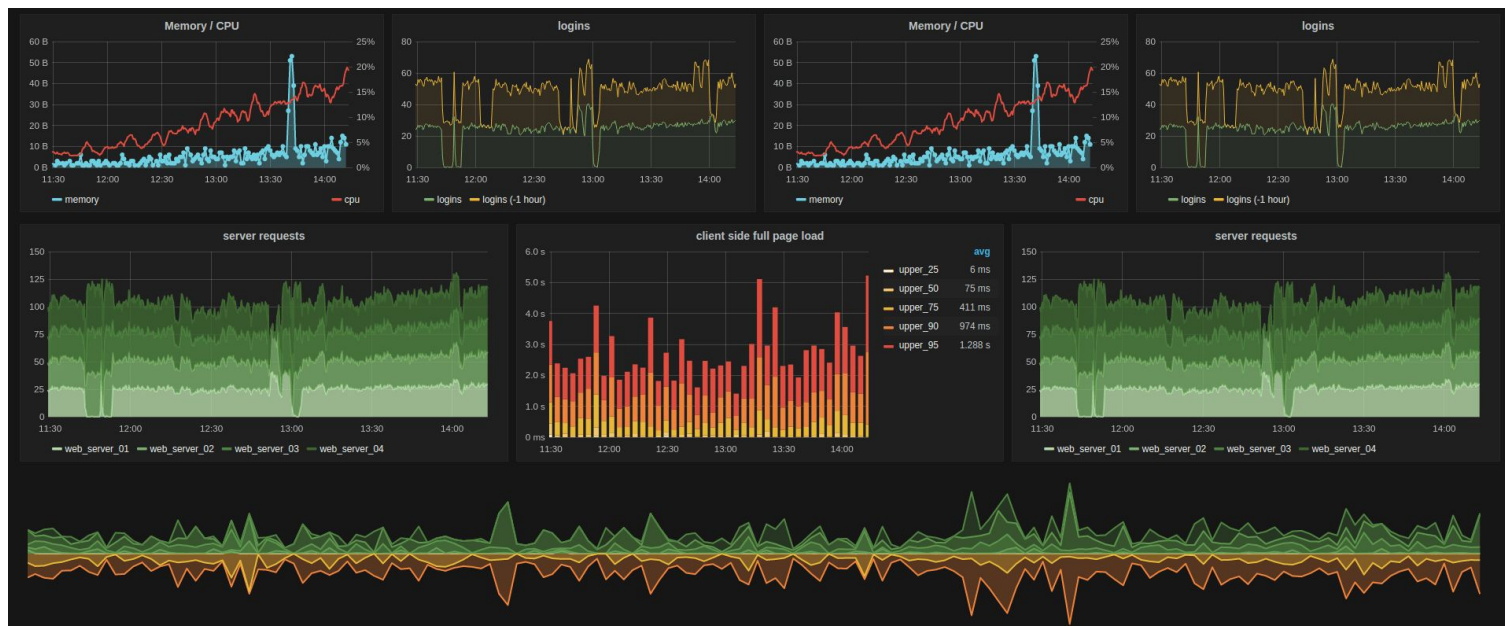
# Hardware issue

Months after roll-out, classes are added and the system keeps learning more, when we get an alert on Slack:



# Monitoring

Grafana tells us when API calls fail, server load goes above warning point, etc.  
We notify Colruyt that a cable has likely come loose, which was the case



# Lessons learned

- Get a testset as close to production
- Don't clean data that will be dirty in production
- Check for possible correlations in metadata
- Augment/unlearn these things
- Monitor your machines remotely
- Discuss remote access to machines upfront

# Next Steps

- System is currently running in production in Kortrijk on 9 checkouts
- Accuracy and amount of products keep going up as it learns
- Other projects with Colruyt are being explored



Thank you