

<https://doi.org/10.15388/vu.thesis.662>  
<https://orcid.org/0000-0002-7156-5995>

VILNIUS UNIVERSITY

Bernardas Čiapas

# Barcodeless Food Products Recognition for Retail Self-checkout Service

**DOCTORAL DISSERTATION**

Natural Sciences  
Informatics (N 009)  
Vilnius 2024

This dissertation was prepared between 2019 and 2023 at Vilnius University.

**Academic Supervisor:**

**Prof. Dr. Povilas Treigys** (Vilnius University, Natural Sciences, Informatics — N 009).

**Defence Panel:**

**Chair – Prof. Dr. Olga Kurasova** (Vilnius University, Natural Sciences, Informatics — N 009).

**Members:**

**Dr. Ernestas Filatovas** (Vilnius University, Natural Sciences, Informatics — N 009).

**Prof. Dr. Dalius Mažeika** (Vilnius Gediminas Technical University, Technology Sciences, Informatics Engineering — T 007).

**Assoc. Prof. Dr. Viktor Medvedev** (Vilnius University, Natural Sciences, Informatics — N 009).

**Prof. Dr. Audris Mockus** (The University of Tennessee, USA, Natural Sciences, Informatics — N 009).

The dissertation shall be defended at a public meeting of the Dissertation Defence Panel at 12:00 p.m. on the 30th of September, 2024 in Room 203 of the Institute of Data Science and Digital Technologies of Vilnius University. Address: Akademijos str. 4, LT-08412, Vilnius, Lithuania.

The text of this dissertation can be accessed at the Library of Vilnius University and on the website of Vilnius University:

<https://www.vu.lt/lt/naujienos/ivykiu-kalendorius>.

<https://doi.org/10.15388/vu.thesis.662>  
<https://orcid.org/0000-0002-7156-5995>

VILNIAUS UNIVERSITETAS

Bernardas Čiapas

# Maisto produktų be brūkšninio kodo atpažinimas savitarnos kasose

**DAKTARO DISERTACIJA**

Gamtos mokslai  
Informatika (N 009)  
Vilnius 2024

Disertacija rengta 2019–2023 metais Vilniaus universitete.

**Mokslinis vadovas:**

**prof. dr. Povilas Treigys** (Vilniaus universitetas, gamtos mokslai, informatika — N 009).

**Gynimo taryba:**

**Pirmininkė – prof. dr. Olga Kurasova** (Vilniaus universitetas, gamtos mokslai, informatika — N 009).

**Nariai:**

**dr. Ernestas Filatovas** (Vilniaus universitetas, gamtos mokslai, informatika — N 009).

**prof. dr. Dalius Mažeika** (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija — T 007).

**doc. dr. Viktor Medvedev** (Vilniaus universitetas, gamtos mokslai, informatika — N 009).

**prof. dr. Audris Mockus** (Tenesio universitetas, JAV, gamtos mokslai, informatika – N 009).

Disertacija ginama viešame Gynimo tarybos posėdyje 2024 m. rugsėjo 30 d. 12 val. Vilniaus universiteto Duomenų mokslo ir skaitmeninių technologijų institute Vilniuje, Akademijos g. 4, 203 auditorijoje.

Disertaciją galima peržiūrėti Vilniaus universiteto bibliotekoje ir Vilniaus universiteto interneto svetainėje adresu:

<https://www.vu.lt/lt/naujienos/ivykiu-kalendorius>.



## Acknowledgements

I want to express my appreciation to my advisor Prof. Dr. Povilas Treigys for guiding me through my PhD journey, for the support and encouragement during challenging moments, and for generously sharing his vast experience and expertise in the field.

I am thankful to the dissertation reviewers Prof. Dr. Olga Kurasova and Prof. Dr. Dalius Mažeika for constructive criticism and insightful suggestions that helped to improve my work.

I am grateful to the Information Technologies Open Access Center of the Mathematics and Informatics department of Vilnius University for providing high-performance computing (HPC) resources used in this research.

The research was funded under the Programme "University Excellence Initiatives" of the Ministry of Education, Science and Sports of the Republic of Lithuania (Measure No. 12-001-01-01-01 "Improving the Research and Study Environment").

Last but not least, I am very thankful to my wife Eglė, who has graciously supported my aspirations throughout my PhD studies, especially considering the responsibilities of raising our three children.

# Contents

<b>Glossary</b>	<b>11</b>
<b>List of Symbols</b>	<b>12</b>
<b>1 Introduction</b>	<b>13</b>
1.1 Problem Statement . . . . .	15
1.2 Research Object . . . . .	17
1.3 Research Goal and Objectives . . . . .	17
1.4 Contributions to Science and Practical Value . . . . .	18
1.5 Defended Claims . . . . .	19
1.6 Approbation of the Research . . . . .	20
1.7 Outline of the Thesis . . . . .	22
<b>2 Related Works</b>	<b>25</b>
2.1 Computer Vision Tasks in Retail . . . . .	25
2.2 Data Collection and Preparation . . . . .	26
2.3 Model Architecture and Training . . . . .	29
2.4 Verification Approaches . . . . .	32
2.5 Product Similarity Approaches . . . . .	34
<b>3 Data Collection and Preparation</b>	<b>37</b>
3.1 Self-checkout Flow and Choice of Camera Location . . . . .	37
3.2 Properties of Raw Data . . . . .	38
3.3 Image Labelling . . . . .	40
3.4 Product Taxonomy and Frequency of Sales . . . . .	41
3.5 Existing Retail Products Datasets . . . . .	43
3.6 Dataset Cleaning . . . . .	46
3.7 Image Pre-processing and Augmentation . . . . .	46
3.8 Dataset Structuring for ML Tasks . . . . .	53
3.9 Conclusions of the Section . . . . .	54
<b>4 Methods</b>	<b>57</b>
4.1 Image Fitness for Product Recognizability . . . . .	58
4.1.1 Deriving Minimum Viable Architecture . . . . .	58
4.1.2 Visibility Thresholding Strategies . . . . .	64
4.2 Product Classification . . . . .	65
4.2.1 Fully Automated Self-checkout Pipeline . . . . .	65
4.2.2 Filtering Empty Images . . . . .	66
4.2.3 Filtering Products of Low Visibility . . . . .	67
4.2.4 Ablation Studies in the Fully Automated Pipeline . . . . .	68
4.2.5 Architecture Alternatives . . . . .	68
4.2.6 Classifier Training and Evaluation . . . . .	70

4.3	Product Verification . . . . .	71
4.3.1	Concept of a Class Verification Task . . . . .	71
4.3.2	Class Verification Approaches . . . . .	73
4.3.3	Product Verification Using Distance from Class Centres . . . . .	76
4.3.4	Increasing Inter-class Distance . . . . .	81
4.3.5	Product Verifier Training and Evaluation Details . . . . .	82
4.4	Product Grouping by Similarity . . . . .	84
4.4.1	Motivation for Product Grouping by Similarity . . . . .	84
4.4.2	Approaches for Deciding Product Similarity . . . . .	87
4.4.3	Training Product Group Classifiers . . . . .	91
4.5	Conclusions of the Section . . . . .	92
4.6	Hardware and Software Frameworks . . . . .	94
4.7	Code repositories . . . . .	94
<b>5</b>	<b>Results</b>	<b>96</b>
5.1	Results of Determining Product Recognizability . . . . .	96
5.2	Results of Product Classification . . . . .	98
5.2.1	Ablation Studies . . . . .	98
5.2.2	Architecture Alternatives . . . . .	99
5.2.3	Other Datasets . . . . .	100
5.2.4	Filtering, Pre-processing and Training . . . . .	100
5.3	Results of Verifying Product Selection . . . . .	104
5.4	Product Classification into Groups of Similarity . . . . .	115
5.5	Conclusions of the Section . . . . .	120
<b>6</b>	<b>General Conclusions</b>	<b>124</b>
	<b>References</b>	<b>126</b>
	<b>Santrauka lietuviškai</b>	<b>139</b>
	<b>Ižanga</b>	<b>139</b>
	Problemos aprašymas . . . . .	140
	Tyrimo tikslas ir uždaviniai . . . . .	141
	Mokslinė svarba . . . . .	142
	<b>Duomenų paruošimas</b>	<b>143</b>
	Vaizdų žymėjimas . . . . .	143
	Prekių taksonomija ir pardavimų dažnis . . . . .	144
	Egzistuojančios maisto prekių vaizdų aibės . . . . .	145
	Duomenų aibės sudarymas mašininiam mokymui . . . . .	146

<b>Metodai</b>	<b>147</b>
Vaizdo tinkamumas prekei atpažinti . . . . .	147
Prekių klasifikavimo etapai, architektūros parinkimas . . .	148
Pirkėjo pasirinkimo verifikavimas . . . . .	149
Prekių grupavimas pagal panašumą . . . . .	150
<b>Rezultatai</b>	<b>151</b>
Matomų ir nematomų prekių atskyrimas . . . . .	151
Prekių klasifikavimas . . . . .	152
Pirkėjo prekės pasirinkimo patikrinimas . . . . .	153
Vaizdų klasifikavimas į panašių prekių grupes . . . . .	155
<b>Bendrosios išvados</b>	<b>156</b>

## Glossary

- AHE** Adaptive Histogram Equalization. An extension of HE that divides an image into smaller tiles and performs HE independently on each tile. 49, 51, 55
- AUC** Area Under Curve of ROC. A single scalar value that summarizes the performance of a binary classifier system across all possible threshold values. 153
- CLAHE** Contrast Limited Adaptive Histogram Equalization. An extension of AHE that limits the amplification of contrast in each local region. 19, 49, 51, 55, 124, 156
- Class prototype** A representative example that characterizes a particular class. It may refer to that example's embeddings learnt by training a neural network. Verification based on class prototype means comparing an input image to the claimed class prototype. Opposite: sample-to-sample based verification. 17, 19, 30–34, 71, 73, 75, 76, 104, 121, 124
- Clean images** Images that contain a single, well visible product, not covered by customer body parts or any other object. 15
- ECR** Efficient Consumer Response. A community of companies, organizations, and stakeholders within the consumer goods industry who collaborate to implement ECR principles and practices. This community includes manufacturers, retailers, distributors, suppliers, service providers, industry associations, and other entities involved in the supply chain. 13, 140
- EER** Equal Error Rate. A metric used to evaluate the performance of binary classification systems at the point at which the false acceptance rate (FAR) and false rejection rate (FRR) are equal. 153
- Embeddings** Image activations of a trained classification neural network's certain layer. In this dissertation, image embeddings are assumed to contain sufficient information about that image class, because the embeddings are the sole input to the network's succeeding layer. 19, 30–33, 35, 36, 71, 74–77, 80, 83, 88, 91, 94, 119
- Empty images** Images that contain only background of self-checkout scales, absent of a product or any other object. 16, 19, 26, 39, 45, 46, 65–68, 93, 98–101, 120, 124

- GMM** Gaussian Mixture Models. 16, 27, 46
- HE** Histogram Equalization. A technique used to enhance the contrast of an image. It adjusts the intensity values of the pixels in an image so that the histogram of the resulting image is spread out evenly across the entire intensity range. 49, 50, 55
- In-distribution** Products (or classes) that the classifier has been exposed to during training. Opposite: out-of-distribution. 17, 33, 34, 72
- OCC** One-Class Classification. 27, 124, 156
- Out-of-distribution** Products (or classes) that the classifier hasn't been exposed to during training. These samples don't belong to any class familiar to the classifier. Since softmax-based classifiers generate probabilities solely among the known classes, they cannot identify such objects. See also: in-distribution. 33, 34, 71, 72, 83
- Product group** A product group refers to a collection of various products typically seen as similar due to their common origin. In food retail, examples of product groups include vegetables, fruits, candies, nuts, etc. Each of these groups comprises multiple individual products; for instance, within the vegetables group, you may find cucumbers, tomatoes, etc. 43
- RFID tags** Radio Frequency Identification tag. A small electronic device that consists of a microchip attached to an antenna. These tags are used for identification and tracking purposes. 15, 140
- ROC** Receiver Operating Characteristic. A graphical plot used to evaluate the performance of a binary classifier system. The plot illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. 153, 154
- Sample-to-sample** A technique to determine if two inputs belong to the same class. Verification if an input belongs to a claimed class typically compares two images: one from a training set of that class and the input image. Opposite: class prototype based verification. 19, 30, 31, 33, 36, 71, 73, 75–77, 104, 121, 124
- SCO** See: self-checkout. 20, 45

- Self-checkout** A point of sale in a food retail store that allows customers to check out the contents of a shopping basket without a cashier. 13, 17, 18, 26, 33, 57
- Self-checkout images** Images of food products captured within self-checkout area of retail stores as customers individually identify and weigh each item. The moment of product identification by a customer represents the specific timeframe when a product is solely present within a particular camera's view. 18, 19, 26, 28, 38, 39, 46, 47, 58, 65, 67, 124
- Self-checkout instance** A single self-checkout device. The lighting and camera angles vary between images captured at different self-checkout instances. 13, 14, 47, 48
- SOM** Self Organizing Maps [1]. A clustering technique that preserves inter-cluster proximity. 19, 20, 35, 86, 87, 89–91, 94, 115, 116, 119, 123, 125, 143
- SVM** Support Vector Machines. 27, 33

# List of Symbols

$C$	number of channels of an input to a convolutional layer or its output
$c_i$	a centre of the $i$ -th class: a single datapoint that best resembles the samples of a single class. Centres are data points in the same space as image activations of a certain neural network's layer
$distCosine()$	cosine distance function between two vectors range -1 to 1
$FN$	false negative
$FP$	false positive
$H, W$	spatial dimensions height, width of an input to a convolutional layer or its output
$L_{CE}$	cross-entropy loss
$L_{contr}$	contrastive loss in siamese networks
$L_C$	centre-loss: a loss function component used in a verification task that measures sample distance to its class centre. Conceptually defined for Euclidean distance in the paper [2]. Variants $L_C^{Cosine}$ and $L_C^{Minowski}$ generalize the loss function for different distance types
$L_{Inter}$	inter-centre loss: a loss function component used in a verification task that measures distance between class centres
$L_{Proxy-NCA}$	a loss function in the verification task as defined in the paper [3]
$L_{triplet}$	triplet loss
$L_{Verification}$	total loss in the verification task
$M$	number of samples in a minibatch or in the dataset. $M_i$ denotes number of samples in the $i$ -th class
$N$	number on classes (products) in the dataset
$ReLU()$	a rectified linear unit function.
$s()$	cosine similarity function between two vectors range -1 to 1
$TN$	true negative
$TP$	true positive
$W, b$	weights and biases of a convolutional or a dense layer. In the case of a convolutional layer $w_{i,j}$ denotes the W's element in the $i$ -th row and $j$ -th column
$X$	input to either a convolutional or a dense layer. It is either an input image or activations of a preceding layer. In case of convolutional input, $x_{i,j}$ denotes X's element in the $i$ -th row and $j$ -th column
$Z$	output matrix of either a convolutional or a dense layer. In case of convolutional output, $z_{i,j}$ denotes output Z's element in the $i$ -th row and $j$ -th column
$\ \dots\ _p$	$p$ -th Norm (Minowski distance)



# 1 Introduction

Retail self-checkout machines help customers transact faster and save retailer costs. The estimated number of self-checkout machines worldwide was 325K [4] in 2019 and is growing by 13.3% [5]. An average of 7 products in a shopping basket and an average of 1,400 weekly transactions per self-checkout result in almost 10,000 images registered per week per self-checkout instance. Typically big retail stores can carry an assortment of up to 30,000 products. The assortment is constantly changing due to seasonality and supplier changes.

However, self-checkouts raised new problems for retailers: theft and long checkout duration of barcodeless products. According to the ECR self-checkout report [6] of 13 retailers, retail stores with 50% of transactions being processed through self-checkouts can expect their shrinkage losses to be 75% higher than the average rate found in grocery retailing. The same study revealed that 43.4% of all shopping baskets contain incorrectly chosen products. Malignant customers abuse self-checkouts in a variety of ways: they replace barcodes of expensive products with barcodes of cheaper ones, and intentionally pick cheaper products from the pick list menu. Shoplifting occurs through selecting the wrong item, barcode switching, failing to scan items, or leaving without paying. Benign customers suffer longer checkout times due to having to pick each barcodeless product from a picklist menu that contains many similar products and has a hierarchical structure of 3-5



Shopping basket, multiple products  $\Rightarrow$  Scanner/scales area, single product (area of research)  $\Rightarrow$  Bagging area, multiple products

Figure 1: Checkout flow

levels. A complex picklist menu often results in an unintentional selection of wrong products and the need for staff assistance. The prolonged checkout duration adds up to over 1,400 weekly transactions on average per self-checkout instance.

Figure 1 shows the flow of product movement during self-checkout process. A customer brings a shopping basket (left in the picture) or a trolley full of products to be purchased to the checkout area. Then he takes one product at a time from a basket/trolley and registers it in one of two ways: scans (products with barcode stickers - e.g. milk packs) or picks from a menu (barcodeless products - e.g. fruits). A scanner is usually located under the glass (green rectangle in the picture) and/or behind a glass in front of the customer (above the green rectangle in the picture). A picklist menu to select barcodeless products is displayed on a touch screen in front of a customer (above the green rectangle in the picture - not shown). Upon picking a barcodeless product from a menu, it is weighed by scales (green rectangle in the picture). Finally, after a product is registered, a customer moves it to the bagging area.

Figure 2 shows a timeline of two sample product selection moments and a few subsequent moments every 0.2 seconds (visual inspection of videos led to the conclusion that significant changes in the semantic structure of the camera image can be expected after not less than 0.2 seconds due to customer actions). A barcodeless product is registered after being placed on scales by selecting from a menu. At the selection moment and sometime after it, a customer's arm is extended towards the self-checkout screen, thus creating a likely interference of body parts in product images. Although some cases (Figure 2(a) at +0.6, +0.8 seconds

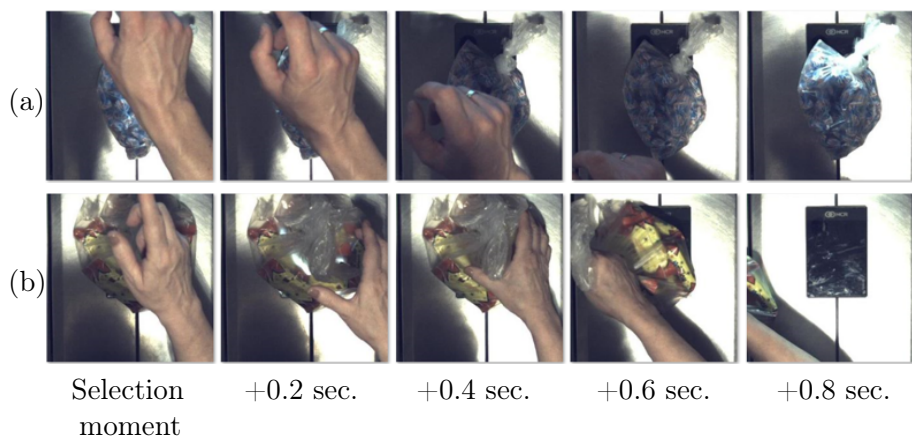


Figure 2: Checkout timeline depicts the change in image contents after two product selection events (a) and (b)

after selection) show clean images, generally a customer’s arm is always a subject of interference.

Although retail stores carry up to 30,000 products, most of them contain barcodes that are easily identified. The number of individual barcodeless products is 200—300 (194 classes used in this research) in self-checkouts by food retailers in stores of 800—1,200 square meters floor size. Often taxonomy of products is unknown, or product visual similarity has little correlation with class proximity in the taxonomic hierarchy. For example, among retail self-checkout products, red apples are more similar to tomatoes than red apples are to green apples, even though red apples and green apples might be in the same product category. Various candy sorts differ among themselves as much as they do from other product categories: candies’ similarity to other products is mostly determined by the wrapping paper.

Retailers try to tackle theft with security scales at self-checkout, which work for consistent-weight products but not variable-weight ones like fresh produce. Some attach RFID tags on high-value items, but this is costly and impractical for many products, especially unpacked fruits and vegetables. Security cameras, although usually monitor the self-checkout area, contain too much footage for real-time review and theft identification by security personnel. Retailers’ goal is to automatically identify mismatch events between a product placed on scales and a customer’s selection, and notify store personnel.

The hierarchical structure of a picklist menu to choose barcodeless products contains 3-5 levels. Retailers’ goal is to automatically suggest a product placed on scales for customer confirmation with high confidence. When a prediction is made with lesser confidence, it is acceptable to present a customer with a reduced list of products for selection.

## 1.1 Problem Statement

There is a need for a fully automated product recognition pipeline at self-checkouts. The pipeline should be capable of collecting product images, adding new products, labelling images with single-product labels, and rejecting images where a product is not present or is poorly visible. Any human interaction in either stage (e.g. dataset cleaning, manual labelling, etc.) is not sustainable due to frequent changes in assortment (due to seasonality and supplier change), and a high number of products and images per product.

The dataset can be automatically collected and labelled by integrating a software agent in self-checkout machines and receiving events of customer choice of products, although it would include some incorrect selections. Unlike an automatically collected image set at

self-checkout, most benchmark datasets (ImageNet, CIFAR[-10]-100], MNIST) only include images where the visibility of objects of interest is good. Synthetic retail product datasets, such as Fruits-360 [7], are not representative of images to be recognized at self-checkouts; recognition performance on such datasets is hardly transferrable to retail stores.

In the self-checkout environment, three areas of product residence may be identified: a shopping basket, a scanner/scales area, and a bagging area. Only the second area contains a single product at a time moment, whereas the basket and bagging area generally contain more. It is a much more complex task to recognize individual products in a shopping basket or bagging area, where multiple products are placed. In terms of computer vision, this would be an object detection task that requires labels with product location bounding boxes. This research refrains from detection tasks in basket and bagging areas, although solving it has a variety of applications.

The stage of rejecting images where a product cannot be recognized is the next step in the pipeline; the rejected images must not participate in any further stages, such as classifying the product for easier selection or customer selection verification. The recent advances in background removal using GMM and neural networks are promising techniques for empty images elimination. A sizable portion of images contain body parts (hands, heads) that interfere with product visibility. This complex task is possibly solvable using neural networks trained on a dataset labelled with product visibility properties.

Images categorized as having satisfactory product visibility must be classified to help solve business tasks, such as picklist menu assistance. The classification task must adhere to a relatively smaller number of samples of rarely sold products. For some business tasks not only Top-1, but also Top-2 to Top-5 accuracy is important (e.g. picklist menu can present a shortened list of 1-5 products). The trained neural network preferably should be light enough to run inference on low-powered, GPU-less self-checkout machines.

Visually similar classes present a challenge and an opportunity in computer vision tasks. The classes being similar usually results in more errors between these classes in a confusion matrix of a multi-class classifier that negatively affects the model accuracy. In some domains, where class similarity is high and/or data for training is limited, predicting a Top-1 class is impractical due to the lower boundary on accuracy drawn by applications. Some applications do not always require the exact class to be recognized as a Top-1, but often narrowing down the list of possible classes to a few similar ones is good enough. For example, in retail self-checkouts, a picklist assistant can narrow down a list of products to choose from to a few similar ones. This is even desirable

by retailers, should the accuracy of several presented products to choose from exceed the accuracy of showing the Top-1 product significantly.

Upon the customer selection of a product, a computer vision solution is necessary to verify if the product in the image matches the customer selection. Verification can only be performed when a chosen product is one of the in-distribution classes. Since customers' choices are limited to items in a picklist menu (which includes barcodeless items but excludes barcode-containing items), it is sufficient to train recognition of barcodeless products. If there's a high likelihood of a product mismatch, an attendant is alerted for visual confirmation. Such a solution would prevent placing a bottle of expensive liquor and choosing any barcodeless product from a menu. However, the solution would not detect replaced barcodes. Data collection of barcode-containing products is a more complex task (due to the absence of a stationary moment at the time of scanning) and is left out of this research. Promising techniques for product (class) verification are Siamese networks and class prototype learning networks (such as Centre-Loss, and Proxy-NCA).

The retail industry badly needs to solve these problems. Successful solutions would simplify product selection from the picklist menu and raise alerts upon scanning/selecting incorrect products.

Several tech companies have developed cashier-less stores: Amazon Go, Zippin, Standard Cognition, and Grabango. They implement a seamless checkout process by tracking customer movement, registering events of item removal from shelves, and assigning items to customers. The cashier-less stores rely on weight and depth sensors embedded in shelves to identify barcode-less products. This requires multiple hardware devices, and precise product placement, which leads to high initial implementation and maintenance costs. This dissertation offers automated product identification at a self-checkout device using a single camera. Although not implementing a seamless checkout as in the cashier-less stores, the proposed solution is by far more cost-effective.

## **1.2 Research Object**

Food retail self-checkout product images

## **1.3 Research Goal and Objectives**

The goal of the research is to propose and investigate an approach for a self-learning barcodeless product recognition pipeline in food retail store self-checkout service. To achieve the goal, the following objectives are identified:

- To explore the collected images of self-checkout products, both

quantitatively and qualitatively. To offer a schema needed to prepare a dataset ready for training neural networks. The schema needs to account for a sizable portion of images being empty or including customer body parts that interfere with product visibility, some products being in plastic bags, and high disbalance in product sales.

- To propose methods for evaluating image fitness for product recognizability and image emptiness, followed by testing their effectiveness through ablation studies. To develop a neural network architecture for product classification in self-checkout images, with comparisons against state-of-the-art architectures using authentic self-checkout data and assessments of generalizability on similar public datasets. The proposed architecture must suit generally low-powered, GPU-less self-checkout machines for running inference. To propose a deterministic and computationally effective method for verifying if a customer’s menu choice matches a product on the scales, and test its accuracy.
- To suggest such a method of grouping products by similarity so that the accuracy of predicting a group (of similar products) is maximized. To evaluate the lift of prediction accuracy against Top-1 accuracy.

## 1.4 Contributions to Science and Practical Value

The dissertation contributes to an area of visual product recognition research in a food retail self-checkout environment. The outline of main contributions and their practical value is as follows:

- An automated pipeline for self-checkout product recognition was introduced. It streamlines image collection, labelling, filtering, pre-processing, and classifier training, validated through ablation studies. Its key value lies in enabling frequent model retraining without manual effort, crucial for dynamic retail environments.
- A method for convolutional neural network architecture design tailored to self-checkout product recognition was presented. Comparative tests revealed its comparable or superior accuracy to established networks like EfficientNet and ResNet, validated on real self-checkout data and a public dataset Fruits-360. Notably, it enables efficient classification using smaller neural networks, ideal for low-powered self-checkout machines without GPUs.

- A class verification method employing the Centre-loss function was investigated, with accuracy comparable to sample-to-sample methods. Notably, it outperforms them in computational efficiency, by avoiding multiple forward passes and random sample selection. Its generalizability was confirmed through testing on the public dataset Fruits-360. In practical terms, it enables accurate verification of product selection through a deterministic algorithm, enhancing security for customer transactions.
- Three algorithms were investigated to measure product similarity and group products accordingly. They assess error contribution, the mean distance between embeddings, and SOM purity improvement. These methods streamline product selection by presenting similar product groups to customers, enhancing decision-making efficiency compared to navigating full product trees.

## 1.5 Defended Claims

1. To filter off empty images from an image set collected at self-checkout using an integrated agent that saves a camera view at the time of customer product selection, a two-balanced-class emptiness classifier should be trained. To filter off images having subpar product visibility, a two-class visibility classifier needs to be trained, where the threshold of product visibility is experimentally determined.
2. To pre-process self-checkout images for training, illumination differences in regions are reduced best by using CLAHE. Removing background has a negative effect on classification metrics. Augmenting oversampled images using both affine and perspective transformations improves classification accuracy over either one. Augmenting using three perspectives outperforms one perspective.
3. Highly accurate classification of self-checkout images doesn't necessitate well-known large neural network architectures. The proposed architecture tuned to self-checkout images is made by adding blocks of convolutional and dense layers until saturation is reached. The architecture generalizes to other similar datasets.
4. In the class verification task, a class prototype-based approach Centre-Loss tantamounts to sample-to-sample approaches in terms of accuracy. Euclidean distance in loss functions performs equally or better than other distance types (Manhattan, Minkowski, Cosine), although nearby Minkowski  $p$  values ( $p=1$  Manhattan,

$p=3$ ) and Cosine perform similarly. In a Centre-Loss architecture that has a dual loss function, the penultimate layer must be connected to the Centre-Loss layer. The optimum size of the penultimate layer positively correlates with the Minkowski  $p$  value.

5. A dual loss function neural network trained with Centre-Loss for class verification achieves classification accuracy similar to a conventional softmax-only network. This unexpected outcome prompts a single neural network's deployment to SCO machines that solve two tasks: verification and classification.
6. Classification-into-similarity-groups accuracy is generally best, when products are assigned to similarity groups using the SOM-Purity method; f-score is generally best when products are assigned using the Error-Contribution method. Classifiers trained on individual-class-labelled images, that predict a similar product group, generally outperform classifiers trained on similarity-group-labelled images.

## 1.6 Approbation of the Research

The results of this thesis were published in 2 scientific journals having an impact factor in Clarivate Analytics Web of Science, 1 peer-reviewed other scientific journal, and 2 peer-reviewed conference proceedings. Presentations of the results were made at 2 international and 2 national scientific conferences.

### **Papers in scientific periodic journals having an impact factor in Clarivate Analytics Web of Science**

- Ciapas B., Treigys P. "Centre-loss - a preferred class verification approach over sample-to-sample in self-checkout products datasets". IET Computer Vision (2024). Wiley. ISSN 1751-9632 | eISSN 1751-9640. DOI: <https://doi.org/10.1049/cvi2.12302>.
- Ciapas, B., Treigys P. "Automated barcodeless product classifier for food retail self-checkout images". The Visual Computer (2023): 1-15. Springer. ISSN 0178-2789 | eISSN 1432-2315. DOI: <https://doi.org/10.1007/s00371-023-03163-8>.

### **Papers in other peer-reviewed scientific periodic journals**

- Ciapas B., Treigys P. "High F-score Model for Recognizing Object Visibility in Images with Occluded Objects of Interest". Baltic J. Modern Computing, Vol. 9 (2021), No. 1, pp. 35–48. DOI: <https://doi.org/10.22364/bjmc.2021.9.1.3>.



## Papers in peer-reviewed conference proceedings

- Ciapas B., Treigys P. (2023). "Self-Checkout Product Class Verification using Center Loss approach". Computer Science Research Notes. Union Agency, Science Press. ISSN 2464-4617 | eISSN 2464-4625. DOI: <https://doi.org/10.24132/CSRN.3301.4>.
- Ciapas B., Treigys P. (2022). Retail Self-checkout Image Classification Performance: Similar Class Grouping or Individual Class Classification Approach. In: Ivanovic, M., Kirikova, M., Niedrite, L. (eds) Digital Business and Intelligent Systems. Baltic DB&IS 2022. Communications in Computer and Information Science, vol 1598. Springer, Cham. ISBN 978-3-031-09849-9 | eISBN 978-3-031-09850-5. DOI: [https://doi.org/10.1007/978-3-031-09850-5\\_12](https://doi.org/10.1007/978-3-031-09850-5_12).

## Presentations at scientific conferences

- Ciapas B., Treigys P. "Self-Checkout Product Class Verification using Center Loss approach". International Conference on Computer Graphics, Visualization and Computer Vision 2023, Plzen (Czech Republic), 2023 May 15-19d.
- Ciapas B., Treigys P. "Retail Self-Checkout Image Classification Performance: Similar Class Grouping or Individual Class Classification". 15th International Baltic Conference on Digital Business and Intelligent Systems (DB&IS), Riga (Latvia), 2022 July 4-6d.
- Čiapas B., Treigys P. "Prekių atpažinimo tyrimas naudojant giliuosius neuroninius tinklus savitarnos kasų vaizduose". XX mokslinė kompiuterininkų konferencija. Klaipėda, 2021 September 23–24 d.
- Čiapas B., Treigys P. "Expanding Convolutional Networks with SIFT Features to Classify Images Better". 11th international workshop on data analysis methods for software systems (DAMSS 2019), Druskininkai, Lithuania, 2019 November 28-30d. / Lithuanian Computer Society, Vilnius University Institute of Data Science and Digital Technologies, Lithuanian Academy of Sciences. Vilnius : Vilnius University Press, 2019. ISBN 978-609-07-0324-3 | eISBN 978-609-07-0325-0. p. 18. DOI: <https://doi.org/10.15388/Proceedings.2019.8>

## Presentations at national scientific institutions

- Čiapas B. "Prekių atpažinimo tyrimas naudojant giliuosius neuroninius tinklus savitarnos kasų vaizduose". Systems analysis seminar. Vilnius University. Institute of Data Science and Digital Technologies. 2022 October 10d.
- Čiapas B. "Prekių atpažinimo tyrimas naudojant giliuosius neuroninius tinklus savitarnos kasų vaizduose". Lietuvos mokslų akademijos (LMA) ir Lietuvos Dirbtinio Intelektu Asociacijos (LDIA) simpoziumas "Dirbtinio intelekto technologijų taikymai vaizdų analizėje". 2022 October 5d.
- Čiapas B. "Prekių atpažinimo tyrimas naudojant giliuosius neuroninius tinklus savitarnos kasų vaizduose". Systems analysis seminar. Vilnius University. Institute of Data Science and Digital Technologies. 2021 December 6d.
- Čiapas B. "Vaizdų ypatybių tyrimas sprendžiant atpažinimo uždavinius savitarnos kasose". Konferencija "Lietuvos magistrantų informatikos ir IT tyrimai". Vilnius University. 2021 May 14d.
- "Prekių atpažinimas savitarnos kasose". Jaunųjų tyrėjų tarptautinė mokslinė konferencija "Jaunasis tyrėjas išmaniajai visuomenei". Vilnius University, Šiauliai Academy. 2021 May 13d. Certificate no. MVG-VUŠA-2021-335.
- Čiapas B. "Prekės atpažinimas savitarnos kasų vaizduose". Systems analysis seminar. Vilnius University. Institute of Data Science and Digital Technologies. 2021 March 15d.
- Čiapas B. "Prekės atpažinimas savitarnos kasų vaizduose". Systems analysis seminar. Vilnius University. Institute of Data Science and Digital Technologies. 2020 May 11d.

## 1.7 Outline of the Thesis

The research schema is shown in Figure 3. The thesis is organized as follows:

- **Introduction** describes the context of the research area (retail self-checkout environment) and defines the tasks that could have practical value if solved. The section defines the aims and objectives of the research and states the scientific contributions and claims defended in the thesis. Publications and presentations by the author are listed.

- **Related Works** provide an overview of the latest research in areas related to this study. The main topics reviewed are computer vision tasks in retail, neural network architecture, discriminative feature learning, clustering and outlier detection, and others.
- **Data Preparation** introduces the flow at self-checkout (Step 1) and presents options for automated data collection and labelling (Step 2). The section presents the collected data’s quantitative and qualitative properties (Step 3). Existing image sets are compared to the collected authentic dataset. Dataset pre-processing techniques adopted for this research are presented (Step 4).
- **Methods** present theoretical investigation results and the design of the experiments. In *Image fitness for product recognizability*, a systematic way of iterative architecture design using building blocks is presented (Step 5) and thresholding strategies for recognizability filters are suggested. *Product classification* presents a fully automated product classification pipeline, which includes Emptiness (Step 7) and Recognizability filters, as proved by Ablation studies (Step 9). The subsection investigates selections of the proper architectural backbone (Step 10). *Product verification* investigates Discriminative Feature Learning (Step 13) and choice of distance type (Step 12) for verification of product selection by a customer. *Product grouping by similarity* suggests options for determining the most similar classes (Step 15).
- **Results** start with comparing recognizability classifiers (Step 6), the best of which are used to rid the dataset of poorly visible products. The product classification results (Step 11) conclude the most fit architecture and prove the need for filtering. Verification results (Step 14) conclude the proper loss function and distance type. Classification into groups of products (Step 16) quantitatively compares options for determining class similarity.
- **Conclusions** summarize the findings of the thesis.

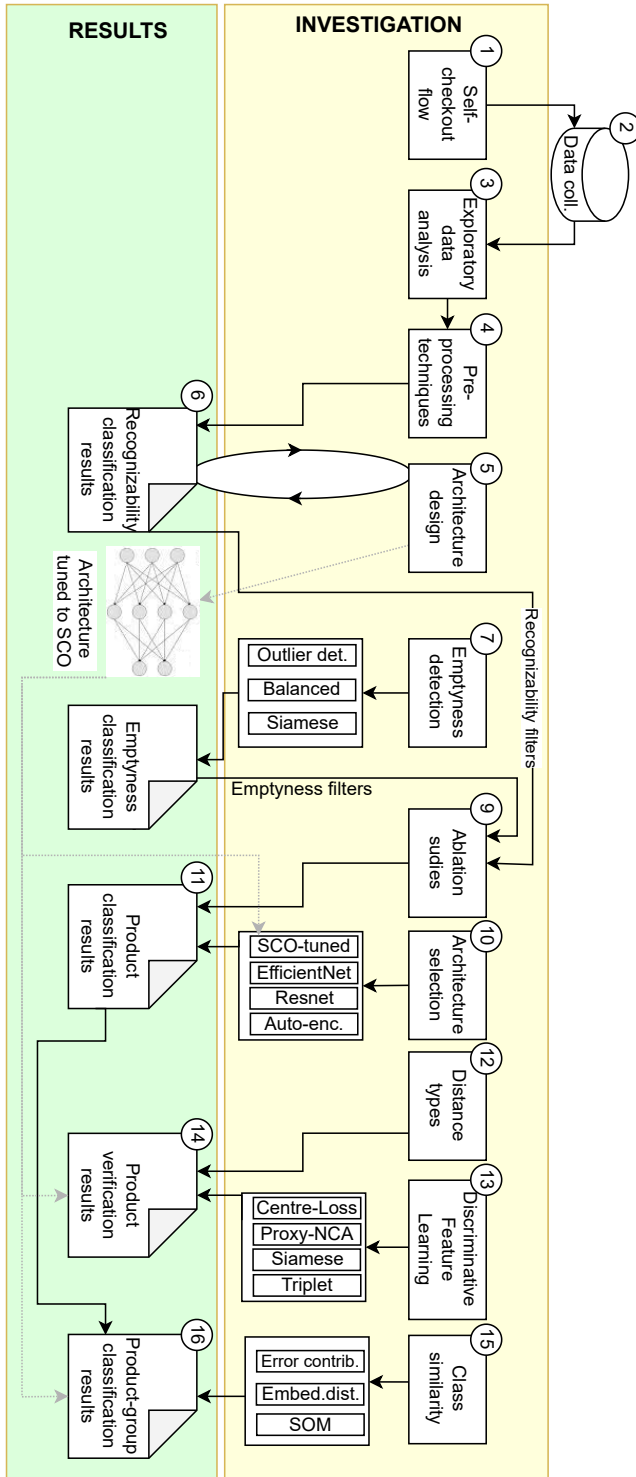


Figure 3: Outline of the thesis

## 2 Related Works

### 2.1 Computer Vision Tasks in Retail

Taxonomy for computer vision tasks in the retail industry by application offered in the paper [8] suggests the following categories: shopping assistance, out-of-stock detection, and planogram compliance; this dissertation relates to shopping assistance applications.

Product recognition tasks in retail using computer vision can be grouped by the number of products in images: single product vs. zero or more products. The latter task is more ubiquitous in store shelves, shopping carts, self-checkout security scales, and checkout conveyor belt image analysis. The papers [9], [10], [11] solve the latter task of object detection (localization and classification of multiple products within images). The metrics of detection tasks are different from classification tasks: detection tasks aim to predict a bounding box of every object of interest within an image and to classify the objects inside the bounding boxes, whereas classification tasks assume the presence of a single object of interest inside images and do not aim to localize them. Due to the difference in metrics of classification vs. detection tasks, the results are not compared against the cited papers. The former task - recognition in single-product images - is also frequently encountered in retail environments: self-checkout and regular checkout scanning areas, self-service, and assisted-service scales. The research [12] solves a product classification task of 8 products in DIY stores by combining images from the internet and regular stores into a training set. Although near-perfect accuracy is achieved, the number of products in the paper is a far reach from the number of products in retail stores. The research [13] classifies ~200 products in retail store shelves by using product brand and category proximity within shelves as an input feature. Such product proximity is irrelevant in the order products are weighed/scanned in self-checkout transactions. The research [14] solves a planogram compliance task using unsupervised methods by finding product patterns in a provided planogram and shelf images. The method does not require any reference images of the products. However, their method assumes that divergence from a planogram is minor, whereas in product recognition tasks at self-checkouts probabilities of products are more evenly distributed.

The image sets that resemble retail food products sold at self-checkouts, are discussed in detail in section 3.5. The results of this research were compared against the papers [15], [16], [17] on the Fruits-360 dataset. The paper [15] focuses not only on accuracy but also on the speed of training by introducing parallel convolutional layers that help with backpropagation and vanishing gradients problems. Since the

main focus of this dissertation is on accuracy, the not-so-popular parallel convolutional networks were left out. The paper [16] (also publishes the Fruits-360 dataset) uses a really basic network of 4 convolutional and 2 dense layers. Since this research focus is on accuracy, bigger networks with more parameters were considered. The paper [17] uses the best architecture on ImageNet: EfficientNet [18]. However, its accuracy is a notch lower than that of other methods.

Specifics in retail self-checkout images include unevenly distributed classes, many images containing poorly visible products due to semi-transparent bags, etc. It is discussed in detail in section 3.2.

## 2.2 Data Collection and Preparation

To collect images of products in the self-checkout environment, a camera had to be chosen, which ideally should see through objects other than the retail products. Advances in covered object recognition use a see-through terahertz beam such as the paper [19] and analyze reflection signal amplitude and phase differences in materials. Such terahertz cameras are far from ubiquitous and will hardly ever be, therefore data for this research was collected using more widespread Red-Green-Blue (RGB) image features.

Given the variety of computer vision tasks in the self-checkout environment, proper label types needed to be chosen. Some publicly known datasets such as ImageNet [20], Pascal Visual Object Classes (VOC - [21]) use rectangular bounding boxes as ground truth to mark object location and size. Others use even more precise object shape markings: Caltech 101 [22] and LabelMe [23] use closed boundaries and Microsoft Research (MSRC - [24]) uses pixel-level segmentation. Each of the above object marking ways such as rectangular bounding boxes, closed boundary shapes, and pixel-level segments are costly to label in new datasets. However, due to the nature of some domains, object location is bounded by a small area (such as products at the time of weighing at self-checkouts), and it is only relevant to predict object class. Tasks of this research only require class labels for images, thus making it less costly to label a new domain-specific dataset. Performance comparison with methods using location-specific labels cannot be made due to different label nature.

Given the significant occurrence of empty images within the self-checkout image dataset, entropy measurement was considered as a separating technique: non-empty images typically exhibit greater entropy compared to background-only. Entropy is widely used in signal pre-processing for automatic label generation: the paper [25] measures entropy between audio frames in order to extract time sequences

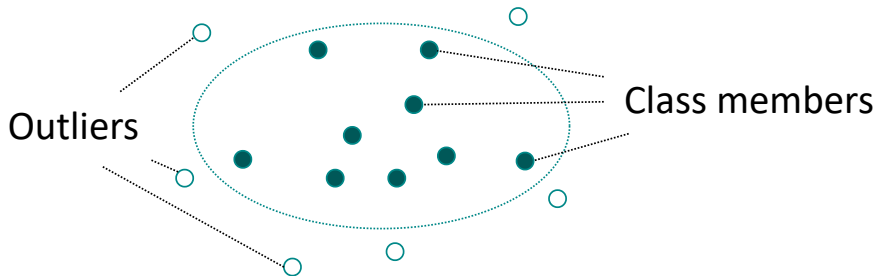


Figure 4: One-class classification

belonging to the same syllable; the paper [26] uses entropy to segment images. However, the background of the self-checkout scales is non-uniform, therefore it is expected to have high entropy. Instead, it was settled for manual image labelling, thus making it possible to formulate the task at hand - deciding if an image contains a visible enough object of interest - as a classification task. To filter out empty images, morphological operations were considered, such as background removal using GMM [27]. However, their proposed method trains a model on multiple empty images of the same static background, whereas different self-checkouts usually have slightly variant camera angles and illumination. Treating empty images as a positive class allows training a Siamese network that separates empty images from the rest. The paper [28] proposes a Siamese architecture and shows its effectiveness on human face verification where a single human's face photos are few. The algorithm trains a "distinction" function by feeding positive (i.e. the same person's photo) and negative (i.e. different person's photo) to the network. Another way to treat the task of separating empty images is OCC [29] - a task of finding a separating boundary between positive class samples and the rest, as shown in Figure 4. The survey of OCC methods [30] proposes using mostly positive examples, whereas negative examples are none, some poorly distributed, or unlabelled. Due to a huge, constantly changing variety of products in self-checkouts, OCC was considered to be a viable way to separate empty images by treating empty as a positive class. Other methods for separating a positive class from the rest include Gaussian processes [31] and SVM [32]; they still have to prove their validity on high dimensional data, such as images.

Other computer vision tasks are solved by researchers to which this dissertation results cannot be directly compared due to different metrics. Yet, they present ideas for solving issues specific to the retail environment. The paper [33] recognizes the lack of labelled data in the retail industry and offers modified weakly supervised learning to solve segmentation tasks using inaccurate and imprecise labels; depending

on the dataset, they present mean average precision in a detection task: ~42% (MVTec Densely Segmented Supermarket Dataset - [34]), ~98% (RPC: A Large-Scale Retail Product Dataset - [35]). The papers [36] and [37] aim to segment salient objects (SO) in remote sensing images. Although SO mask could be used to remove background, but self-checkout images background is static, which is easier removed using morphological operations. The paper [38] proposes to solve the detection of empty shelves by extracting high-level features (texture, colour, geometry) and supervised training methods; it presents ~85% accuracy in empty shelves detection. Both of the above do not take advantage of retail store setup where only a single product exists at a specific location at a time (such as checkout scanner/scales area).

Filtering out images with low product visibility is a complex task that requires interpreting images for plastic bag transparency, and separating intruding customer body parts from products. This task is specific to the retail domain that has not been widely researched. The paper [39] proposed a 6-class classifier that separates images by product visibility: the method suggests 4 ordinal categories by product's visible area percentage and 2 classes for products in bags, but does not conclude which could be reliably classified into individual products. To use this approach as a pre-processing step to rid the dataset of images having poor product visibility, one must determine the best threshold separating visibility categories into Visible/Invisible categories. The dataset was filtered using this method by experimentally choosing the best threshold. A more detailed discussion is in section 3.6.

The automatically collected self-checkout dataset was unbalanced; neural network classifiers trained on unbalanced sets are more likely to predict more common classes. Fighting this bias usually includes balancing the number of samples or introducing higher loss coefficients for underrepresented classes. The paper [40] concludes that over-sampling the less represented and under-sampling the more represented classes performs better than only under-sampling. They also show the benefit of over-sampling over introducing loss ratios for different classes. These findings were applied when training classifiers in this research.

The number of images per class in the self-checkout dataset was dwarfed by those found in ImageNet; neural networks require large image quantities to generalize to unseen data. Strategies for image augmentation to increase data variability, and reduce model overfitting to training data, are numerous. The paper [41] uses Generative Adversarial Networks (GAN) to generate more medical images and present results improvement in binary classification tasks for medical images over the no-augmentation approach. However, results for the multiclass classification task (object of this research) using this augmentation approach are not



presented. On the other hand, using GANs implies heavy hardware usage; augmentation in other domains than medicine, where labelled images are not so scarce, may be achieved using simpler techniques. The paper [42] uses a perspective transformation to augment images when annotated data is scarce or datasets are unbalanced and shows the effectiveness of such an approach on a detection task. The proposed perspective transformation was used to augment over-sampled images. Oversampling and augmenting are discussed in more detail in section 3.7.

## 2.3 Model Architecture and Training

Deep neural network architectures for image classification tasks abound. The pioneering paper in deep convolutional neural networks [43] suggests using a set of Convolutional layers followed by a set of Dense layers. Although its achieved accuracy (62.5%) on the ImageNet dataset of 1,000 classes was later improved by many researchers, most use his proposed sequence of layer types. The effectiveness of his approach on a self-checkout dataset remains to be investigated. Convolutional neural networks still appear in many leading papers of the ImageNet classification task, such as the papers [18], [44]. The paper [45] suggests a convolutional auto-encoder-based classifier, where an auto-encoder is first trained to learn features, then fixed encoder layers are appended with two Dense layers to train a classifier. They show the effectiveness of the technique on a 3-class painter classification task with >96% accuracy. The results of their proposed technique on a much greater number of classes are not presented. A different approach from convolutional networks to extract image features is Vision Transformers (ViT) introduced in the paper [46]. ViTs rely on the attention mechanism originating from language models, they lose spatial information of image patches in proximity. Yet, ViTs achieve impressive results in the ImageNet classification task in the paper [47]. Capsule Nets (CapsNets) introduced in the paper [48] is another competitor to convolutional nets for feature extraction. CapsNets show equivariance in object pose and location extraction, whereas convolutional nets lose location information in deeper stacked convolutional layers. However, CapsNets does not show competitive progress in classifying images of big datasets, such as ImageNet. Alternative architectures applied in this research are discussed in detail in section 4.2.5.

To optimize the neural network parameters, Cross-Entropy has been widely applied as a loss function in classification tasks since the pioneering artificial neural network papers [49], [43]. As opposed to Cross-Entropy, many researchers use entropy to create unsupervised

models: the paper [50] attempts to maximize entropy among different image background/foreground pixels; the papers [51] and [52] try to reduce entropy when selecting the next features in forming decision tree nodes; the paper [53] uses the entropy of output by competing translation systems in order evaluate translation quality.

A class verification task needs a different loss function because its objective is to ascertain whether an input image fits the class it is claimed to belong to. In class verification research, face identity verification takes the spotlight. The common approach involves comparing image features through sample-to-sample methods. A Siamese network [28] effectively learns a distinction function - whether two images belong to the same class (person) or not. It consists of two identical networks with shared weights and a distinction layer that measures the Euclidean distance between embeddings of a fully connected layer. A similar concept is employed in Triplet Loss [54], except that it uses three images to calculate a loss function: an anchor, a positive (same class/persons') and a negative (another person's). The Anchor+Positive pair is trained to output an opposite value than the Anchor+Negative pair. Circle Loss [55] introduced a variant of a Triplet Loss function by using not linear, but circular separating boundaries between positive and negative pairs similarities, and showed improved verification results on several major face datasets. Both distinction function-based methods - Siamese and Triplet - require reference images (or their embeddings) during inference. Although that is usually satisfiable when the number of images per class is small, using big training datasets faces several challenges: first, different reference images lead to different verification results; second, inference against several reference images requires aggregation, an area where research is lacking; third, inferring against a multitude of reference images is rarely feasible due to performance and storage reasons. Considering the proven accuracy of Siamese and Triplet networks, experiments in this research contrast them with class prototype-based methods. The loss function of class prototype-based methods measures the distance between the embeddings of a class prototype and a sample. SphereFace [56] and ArcFace [57] measure the angular distance between the embeddings of a class prototype and a sample, then modify the Cross-Entropy loss function to use angular distances. They show better discrimination of inter-class features than regular Cross-Entropy. Their unfold loss function does not allow adjusting classification vs. verification relative importance. Since the verification task's primary focus is verification (as opposed to classification), the Cross-Entropy loss is only included to preserve class separability (i.e. not to regress all class centres to the same point), thus it is important to adjust this relative importance. Another way to verify class is to derive a class prototype

during training and then compare an input sample against the prototype during inference. A class prototype is a generalized representation of a class used in class verification tasks. The research [58] creates prototypes based on activations and uses Earth Mover’s Distance (EMD). However, these prototypes often have high intra-class variation. In Discriminative Feature Learning [2], class centres are learned by averaging class samples in the same embedding space, pushing embeddings towards them with a two-fold loss function: in addition to Cross-Entropy, the other summand Centre-Loss pushes samples towards their respective class centre. The first member of such a loss function - Cross-Entropy - ensures that different class centres are separable, i.e. do not regress to the same point. Class centres are updated in every iteration, thus "learned". This method was adopted in experiments of this research, extending it to various distance types. Proxy-NCA [3] learns class centres using a loss function that pushes samples not only towards their own but also away from other class centres. It measures cosine distance, which, by definition, loses the scale component. Proxy-NCA’s spin-offs SoftTriple [59] uses several single class centres and Proxy-Anchor [60] trains more efficiently by minimizing both sample-to-centre and sample-to-sample distances. Artificial Immune Networks [61] form high-density clusters for each class but don’t explicitly minimize cluster size, resulting in expected high intra-class variance. Some researchers use text data to build class prototypes, as in the paper [62]. However, obtaining such data for self-checkout products is challenging.

Verification loss functions (such as Contrastive Loss) typically measure Euclidean distance but the use of non-Euclidean distance types remains relatively underexplored in research. Some language-focused researchers opt for metrics like Manhattan, as evident in the papers [63], [64], and [65], or Chebyshev distance, as in the paper [66]. In the realm of computer vision, the paper [67] conducts a comparison involving Manhattan, Euclidean, and Chebyshev distances using an emotion-labelled dataset. Its work is expanded upon by including Minkowski distances with varying  $p$  values (3 and 4) as well as Cosine distance, conducting experiments on the focus-of-interest self-checkout products dataset. While the paper [68] examines architectures using Manhattan and Chebyshev distances, it is important to note that their two architectures differ in other aspects, making direct distance type comparison inconclusive. On the other hand, the paper [69] undertakes a comprehensive comparison of various distance types (Manhattan, Euclidean, Minkowski, Chebyshev, Cosine) in an image retrieval task. The aim of this research is to conduct a similar comparison, albeit in a different context - a verification task.

Transfer learning is a technique of partial pre-trained network

transfer from one domain to another. The technique is popular when neural networks need to be trained in domains where training data is insufficient, classes are unbalanced, or both (the case in this research). The paper [70] concludes that earlier layers are more generic - therefore more transferrable - than later layers that are more domain-specific. The paper [71] cuts off the last layer from the pioneering Krizhevsky [43] architecture prior to adding 2 additional dense layers, and shows the effectiveness of this approach over random weight initialization. Cutting off more than the last layer is not discussed. In this research, strategies of initializing and fixing weights from well-known pre-trained models are applied.

In the task of separating images with insufficient product visibility, a proper evaluation metric had to be chosen. Most methods using datasets where object location is defined, such as the papers [72] and [73], use intersection-over-union (IoU) to measure the correctness of object localization and segmentation. Since in this research no datasets with annotations of object location have been used, class labels (Is/Isn't an object) were used in measuring correctness. F-score is widely used in information retrieval, such as search, document classification, or query performance evaluation. The paper [74] got 0.65 f-score measuring class match between searched vs. retrieved images in content-based image retrieval. The paper [75] used f-score to classify search query difficulty and received values up to 0.665. The research [76] classified textual documents into pre-defined classes and received f-score values up to 0.92. To evaluate models of product recognizability, the f-score was used.

## 2.4 Verification Approaches

A class prototype is a generalization of the data samples of a single class, which a new sample is compared against at inference time. Multiple research attempts have been made to derive a class prototype given a set of data samples. The paper [62] uses the term "class prototype in a semantic space", which is category vectors (one per category). They construct the category vectors by using auxiliary textual information about the classes of interest. In this research, textual or other information about the classes is not used to train category vectors - mostly because discriminative textual information is not easily obtainable for the country-, chain-, or store-specific classes of self-checkout products. To derive a class prototype, some authors use the space of embeddings of a certain trained neural network layer. To extract features from images, many authors train convolutional filters that are class-agnostic, but sensitive to an object's existence. A very similar concept - class-agnostic convolutional filters on object-containing

patches - was used in the paper [72]. The research [58] first extracts features using a fully convolutional network and then compares patches on an unlabelled dataset. In this dissertation, training is done on full images rather than object-containing crops (due to self-checkout dataset annotation nature). These methods generate region proposals, and then extract features from them: the research [77] extracts visual words from pixel level segments, then compares them to those of known object bounding boxes; the paper [78] finds closed boundary shapes. Both of the above methods imply having learnt features from a dataset annotated with object locations, which didn't exist in the dataset used in this research.

In a class verification task, one approach is to view it as outlier detection, identifying outliers as incorrect [image;claimed-class] pairs. Outlier detectors create a boundary to separate in-distribution from out-of-distribution samples, experimenting with various boundary shapes like hyperplanes in SVM [79], ellipsoids in robust covariance models [80], and any shape in isolation forests [81]. In this research, the choice of boundary shape depends on the distance type used: Manhattan leads to a hypercube boundary, Euclidean results in a hypersphere, and Cosine forms a cone. However, loss functions of this research verification task push latent space variables of any class to the same point ("class centre"), thus any centre-enclosing boundary suffices. Some outlier detection methods don't explicitly minimize intra-class distances, but enhance separability using techniques like Gaussian Radial Basis Function (RBF) kernels [82]. In this study, the aim is to minimize intra-class distances for all latent layer embeddings, a distinction from typical outlier detection methods.

The output of class verification must gauge the probability of an image belonging to a claimed class. Conversely, computer vision classifiers in the papers [83], [84], and [85] assign probabilities to known classes. However, many real-world applications, like self-checkout product verification, require identifying out-of-distribution samples, a task that classifiers aren't designed for. Verification, in contrast, distinguishes between in-distribution and out-of-distribution samples. Efforts to address this issue include the paper [86], which sets a lower threshold for the Top-1 classifier's prediction to consider a sample as out-of-distribution, and the paper [87], which modifies classifier architecture by adding a confidence branch. However, both approaches may struggle with datasets containing similar classes, like self-checkout products with multiple similar-looking tomato types, leading to complex probability distribution issues.

While numerous research papers delve into class prototype-based class verification and sample-to-sample-based verification separately,

there is a research gap when it comes to comparing these two approaches. Surprisingly, the investigation did not uncover any articles focused on a verification task that directly compares these two approaches while maintaining identical hyperparameters, including neural network architecture and dataset.

Researchers addressing verification tasks employ various methods to model distributions of class samples. For instance, Open Set Deep Networks [88] create a Weibull distribution for each class, enabling varying variance levels among different classes. Similarly, the paper [89] learns per-class distributions and establishes a fixed Mahalanobis distance from the class centres to determine a sample’s class membership. Both of these approaches are more flexible than the one used in this research, as they accommodate different variance levels per class. However, none of these studies train models to reduce intra-class variance. By training models that minimize intra-class variance, the necessity to model distinct per-class distributions can be reduced. Strategies of class verification based on distance from centres are discussed in section 4.3.3.

Class verification research is distinguished by the type of negative samples. In face verification, negative samples in the papers [56], [2], [90], [54], [91], [28], [57] typically consist of images associated with a different person’s identity. In the context of AI safety, like in the paper [92], negative samples are generated by GANs, while positive samples are real images. However, in self-checkout product verification, negative samples should be either images from a different out-of-distribution class or any out-of-distribution product not in the training data. Unfortunately, there’s a shortage of datasets for retail barcodeless products, so this research used only in-distribution data. Research in artificial intelligence safety attempts to verify if the input is consistent with known (out-of-distribution) samples. Deep Verifier Networks (DVN) [92] use an auto-encoder’s latent layer’s activations to estimate the density of known samples. Samples with latent activations inconsistent with the density model are rejected as adversarial. DVN does not attempt to model latent space where intra-class samples are clustered together. Thus DVN does not derive class prototypes. Since in the self-checkout domain "adversarial" samples are images of other than the declared class, deriving a class prototype is suggested.

## 2.5 Product Similarity Approaches

Image similarity is drawn from an image description in the paper [93]. Such a description is not available in self-checkout image sets.

The concept of similarity descriptor is presented in the paper [94]. It

uses pre-trained famous classifiers to draw feature maps of deep layers. The authors first train a neural network to choose the most proper deep layer to use as a feature set for image similarity. Then feature map histograms are used to measure image similarity by calculating Earth Mover's distance. In this dissertation, a pre-trained classifier was used to draw feature maps for further comparison. Instead of building a complex network to choose the deep layer, the pre-last dense layer was used: the paper [94] shows little difference in performance by using different layers of classifiers.

The ranking technique in the paper [95] learns image similarity by training a network on triplets that consist of 3 images: query, positive (similar) and negative (dissimilar). It relies on a self-created dataset, where similar candidates are Google image search results. It then calculates image pairwise similarity using an extensive list of features among images of the same search query results, whereas the similarity of images between different query results is set to 0. Using their method for this research presents an issue: since the task is to learn inter-class similarity, it does not make sense to label the pairwise similarity of images of different classes to be 0.

The image clustering techniques in the papers [96], [97], and [98] group images by exploiting some deep layer's embeddings. The resulting image clusters, however, do not map into groups of classes ("class clusters"): clusters usually contain images from a multitude of classes and images of a class are dispersed over a multitude of clusters. This means that image clustering cannot be used directly to measure class proximity, but may be used as the first step followed by some post-processing to determine class similarity.

The clustering of categorical datasets usually uses purity metric to determine the quality of clusters; subsequently cluster purity improvement by merging classes can be measured to determine class proximity. SOM [1] is a clustering technique that preserves inter-cluster proximity: data points that are close in high dimensional spaces are assigned to cluster centroids that are close in 2D space. Since this thesis investigates inter-class proximity, SOM is chosen to be one of the alternatives to group classes into clusters.

Determining class similarity requires some generalization based on the similarity of image samples. The average of all samples of a given class is used to calculate a "class vector" in the paper [99]. Cosine similarity between class vectors is used as a measure of class similarity. They use linear discriminant analysis (LDA) to come up with a feature space for calculating class vectors. In self-checkout products domain, samples within the same class may be quite different (e.g. a Snickers candy has 2 views depending on whether it is put front side up or

down), therefore calculating a class average may not be representative of a class. Instead, a sample-to-sample comparison is used. Like the authors of the paper [99], cosine similarity is used between embeddings. To come up with a lower-dimensional space of image embeddings, pre-last fully connected layer activations are used instead of LDA: this is not a significant difference as long as class-specific information is preserved within embeddings. A technique proposed in the paper [100] is for class-to-image comparison by training a Siamese-like network that takes extracted class and image features. Although a promising technique to identify if an image belongs to a specific class, it doesn't provide a way to compare how similar two classes are.

Hierarchical classification tasks present the challenge of picking the right classification strategy. Two ubiquitous strategies are local classifiers and multilabel classifiers. Local classifiers in the paper [101] can be split into classifiers-per-level (LCL), classifiers-per-parent-node (LCPN) and binary classifiers-per-node (LCN). Each local classifier strategy requires multiple classifiers to be trained and used during inference - a potential obstacle for not-so-powerful devices, such as self-checkout computers. A task of grouping in this research can be thought of as a hierarchical classification task of 2 levels, where leaf nodes are groups of products; therefore, the task converges to a classic multi-class classification problem. Multilabel classification tasks in the papers [102], [103], and [104] use multiple labels for every data point - one label per hierarchy level. The loss function of multilabel classifiers requires setting weights between levels. The grouping task of this research can be thought of as a multilabel classification task of 3 levels - root level, class group level and individual product level - where the weight of leaf (individual products) level is 0, and the weight of class group level is 1. Approaches for deciding class similarity are discussed in section 4.4.2.



### 3 Data Collection and Preparation

#### 3.1 Self-checkout Flow and Choice of Camera Location

The choice of camera location was based on the following characteristics: products should be fully present, contain minimum interference with customers' body parts, include minimum background, and the background should be mostly static. The camera mounted on top of a self-checkout screen in Figure 5(a) results in many images having products covered in large part by customers' arms and heads. The camera located below a self-checkout screen in Figure 5(c) faces horizontally, thus contains mostly store's background that is dynamic (customer movement) and, therefore, hard to remove. The camera placed on the side of a self-checkout screen in Figure 5(b) produces fewer images containing body parts than the camera on the top and has mostly static background of the scales area unlike the camera below the screen. After initial tests, the camera's location for data collection was chosen to be on the side of the screen as in Figure 5(b).

The crop of choice was based on the following characteristics: it must contain products, must lend itself to automatic labelling, and

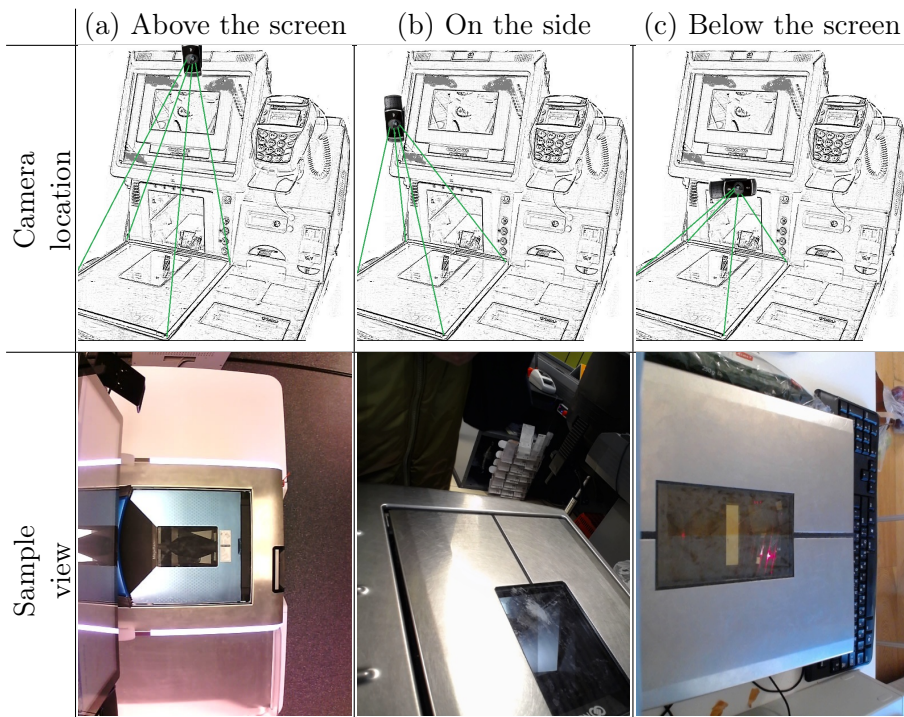


Figure 5: Camera location variants and sample views

products should be clearly visible and not be stacked on each other. The scanner/scales area usually contains a single product, whereas other areas (shopping basket/trolley, bagging) generally contain multiple products, covered fully or partially by other products. This suggests the scanner/scales area (green frame in Figure 1) to be the top choice for image set collection and further analysis. The red-circumvented area (Figure 6, left) represents the area of interest that was cropped and perspective-transformed to a rectangle (Figure 6, right). The camera position being static with respect to the self-checkout scales area, the position of the crop (the four corners of the scales) was set once upon installation of the camera, then used for the entire duration of image collection. The cropped area was used for all the experiments. The original image size was  $480 \times 640$ . The crop being of irregular shape, needed to be perspective-transformed into a square shape in order to fit as an input into most convolutional neural networks, that take square input shapes. The target square size was set to  $256 \times 256$ , so that the majority of information came from the camera (as opposed to "invented" by upsampling if a bigger target size was used).

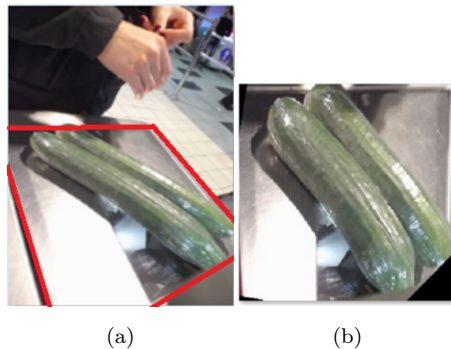


Figure 6: Original (a) and transformed (b) images. Cameras placed on the side of a self-checkout screen produce fewer images containing body parts but contain much undesirable background and skewed products

Images were collected from cameras placed over 4 distinct self-checkout machines in one food retail store of 800—1,200 square meters in Vilnius, Lithuania. The camera model was a Logitech C930w Business Webcam, having a  $90^\circ$  diagonal field of view, 4x digital zoom, available Full HD/30fps resolution.

### 3.2 Properties of Raw Data

Self-checkout images demonstrate such specifics:

- Very often products are covered by hand or other body part, i.e. product visibility is limited;
- Products are packed in plastic bags which limits product visibility as well;
- Products vary in size;
- Every self-checkout camera has different illumination properties, illumination varies during work hours and typically is non-uniform.

Real-life images, which need to be classified, often contain objects that are occluded to some degree. For example, most self-checkout images contain products partially covered by a customer's hand or another body part; about 10% of barcodeless products are sold in plastic bags that are semi-transparent and may have a high glare; specific locations within the scales area reflect light in a way that reduces recognizability, illumination differs during the day time, products differ in size, etc. In addition, self-checkout images collected at the moment of choosing a product from a picklist menu contain many empty images. Reasons for empty images are multiple: network latency, software bugs, no item present at the selection moment, etc.

Due to all these aspects, which are specific to self-checkouts, some images are likely to contain less information about the object of interest. Simply applying classification techniques on images with occluded objects is likely to result in low classification results, if they are not eliminated from the neural network training set. Prediction of products in empty images is meaningless.

Cameras, although stationary, do not always contain the same static background: they get shifted by customers; cameras installed in new locations contain new backgrounds. As a result, empty images cannot be eliminated by simple morphological operations. Deciding on whether images contain products visible enough for classification is an even more difficult task.

To obtain satisfactory classification results, images with occluded objects must be first categorized to see whether objects of interest are visible enough for classification, and only images containing well-visible objects need to be classified. Empty images must be discarded. Separating images containing visible enough products from the rest is a necessary step not only in preparing datasets for training but also in the inference pipeline in order to apply individual product recognition techniques only for images where products can be recognized.

### 3.3 Image Labelling

Two types of labels were used:

- Product ID was used to train product classifiers;
- Product visibility categories were used to train the product visibility classifier, which was subsequently utilized to remove images from the raw dataset where products were poorly visible.

Images were automatically labelled with customers' selected product IDs. To collect and label images, a software agent was installed on self-checkout machines. The software agent received events from the self-checkout software of a customer who has chosen a product from a picklist menu, took camera snapshot images and labelled them with customer selection (at the moment of customer selection, a product was usually present on the scales). Although some mistakes in customer product selection were spotted, no manual re-labelling was done: manual re-labelling would be unpractical in a rapidly changing store assortment.







The images were labelled with visibility-level labels manually. Due to the uncertainty of what portion of the product needs to be visible, it was decided to use multiple ordinal labels. Labelling images into a bigger number of product visibility categories would have given more flexibility when splitting data into Visible/Invisible categories. However, considering that a human labeller would make more mistakes if more categories were used, it was decided to limit the number of visibility labels to four: Q1, Q2, Q3, and Q4. The labels were assigned based on what percentage of the product area is within the image and not covered by other objects, such as the customers' hands. The Q1 category had up to 25% product's area visible, Q2 - from 25% to 50%, Q3 - from 50% to 75%, and Q4 - more than 75%. Images with products packed in plastic bags showed very different features from images with unpacked products in early analysis: plastic bags are easily recognizable, but products inside the bags are not necessarily recognizable. Therefore, it was decided to label images in plastic bags with different labels. Due to some images within plastic bags having intense light reflection that makes products unrecognizable, it was decided to split images with plastic bags into classes Bag (recognizable products in plastic bags) and BagR ("R" means reflection that makes a product not recognizable to humans). Finally, the images have been labelled into 6 exclusive visibility categories by applying these rules:

- By product visibility percentage (classes Q1–Q4) for products not in bags

- Products in bags (class Bag) when the product can be recognized by a human
- Products in bags with a reflection (class BagR) which makes a product unrecognizable

Samples of each data class are displayed in Table 1.

Table 1: Each visibility class samples and class ratios

					
Q1 32%	Q2 22%	Q3 15%	Q4 21%	Bag 7.3%	BagR 2.6%

Due to the uncertainty of how many images need to be labelled in order to create models that generalize, it was chosen to label a similar number of samples per class as the ImageNet dataset ( $\sim 1,000$ ), where the classification task was solved with high accuracy. The entire labelled dataset consists of  $\sim 6,000$  images. Images were randomly selected for labelling from the entire set with no pre-selection criteria. All the selected images were labelled by a single human labeller.

### 3.4 Product Taxonomy and Frequency of Sales

Food retail stores typically carry an assortment that consists of thousands of products and is highly dynamic: new products are introduced to test the market frequently; products appear and disappear due to changes in seasons. Most of the products contain barcodes, but several hundred are barcodeless (unpacked) - usually fruits, vegetables, nuts, and some sorts of candies. The specific number of products depends on a retail store size. A typical retail store of 800—1,200 square meters carries about 200—300 barcodeless products - such a floor size is approximate footage of Tesco Metro, all Aldi, a bit smaller than all Lidl, and half the size of Walmart Neighborhood Market stores. Some food retailers (such as Tesco, Walmart) manage several store sizes, with smaller stores usually carrying a subset of products. Same retailer and size stores usually share assortment except in geographically dispersed regions.

Filtering products that should be included in the recognition pipeline depends on several criteria: ease of selection for the customer, and image count available per class. Business problems for both types of products differ, e.g. products with barcodes are identified by scanning a barcode,

whereas barcodeless products are identified by customer selection from a picklist menu in self-checkouts, which is both time-consuming and error-prone. In this research, it is intended to help solve the problem of choosing the right barcodeless product. Only barcodeless (weighed) products were included in the selection for further experiments.

Frequent introduction of new products requires a boundary on a minimum number of images for a product to be included in the recognition pipeline. Products, that had less than 3 images, were filtered out (considering at least one image would need to be included in a train, a validation, and a test set). The total number of individual products came out to be 194.

The full set consists of 26,637 images that belong to 194 different product IDs. The image set classes are highly unbalanced as shown in Figure 7: the most frequently purchased products exceed the least frequent in the order of thousands. Seasonal products, products introduced for a short term to test the market, and supplier change contribute to high sales disbalance. Distribution of images among classes is extremely uneven: the biggest classes contain 3282 (bananas), 2760 (carrots), and 2181 (lemons) samples, whereas the smallest classes (rarely purchased candies) contain only 3 samples.

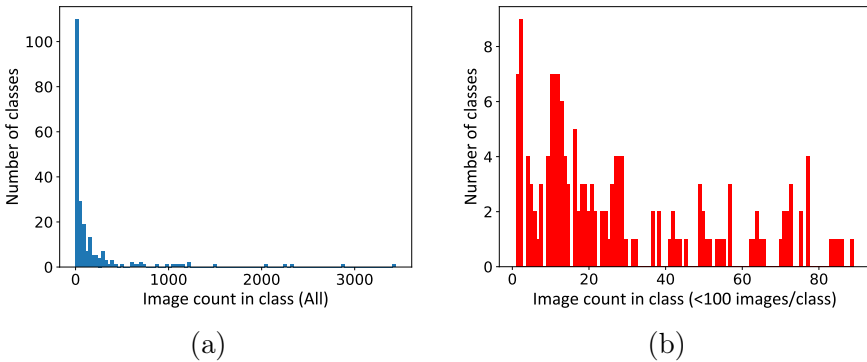
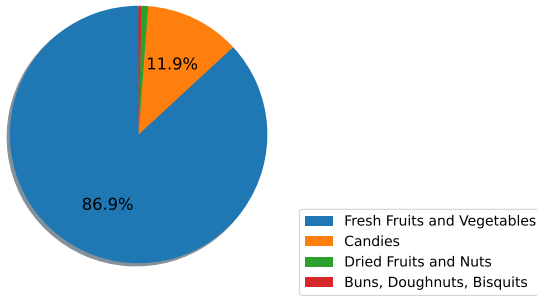


Figure 7: Image counts in classes - overall (a) and classes having less than 100 samples (b). Image count frequencies for different products in the authentic self-checkout dataset represent the real-world distribution of sales of different barcodeless products in retail stores. The dominant classes contain 3441 (bananas), 2857 (carrots), and 2357 (lemons) samples, whereas about half of the classes contain less than 50 samples per class

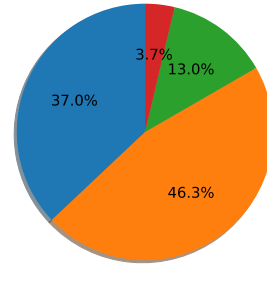
Self-checkout dataset distribution by category is shown in Figure 8. "Fresh Fruits and Vegetables" is the dominant category by sales (thus, by image count) - it makes up almost 87% of the images. However, the

Image count by product group



(a)

Product count by product group



(b)

Figure 8: Image count by product group (a) and product count by product group (b). Self-checkout dataset's dominant product group by sales is "Fresh Fruits and Vegetables" (a). However, this group is not the biggest by product count (b): "Candies" product group contains a similar number of products, whereas "Dried Fruits and Nuts" and "Buns, Doughnuts, Biscuits" are smaller, but significant

number of products in each of the high-level categories is more evenly distributed: "Candies" make up 46%, "Fresh Fruits and Vegetables" - 37%, and other categories are significant, too.

### 3.5 Existing Retail Products Datasets

There is a scarcity of representative datasets for real-world self-checkout products. A dataset representative of a self-checkout environment needs to possess the following properties:

- Contain barcodeless products sold in self-checkouts;
- Images either labelled with product ID (single-product images) or bounding boxes/masks in addition to product IDs (multi-product images);
- Background and illumination properties should be similar to ones encountered in self-checkouts at the times products need to be recognized;
- Some or all product images should be in transparent plastic bags, similar to how products are packaged for the duration of checkout.

In contrast to authentic self-checkout datasets, synthetic counterparts do not comprise items enclosed in plastic bags, objects

partially occluded by anatomical features, and variable lighting conditions. It is imperative to underscore that only computer vision solutions validated against authentic self-checkout datasets apply to real-world self-checkout environments. Many datasets of retail products consist mostly of barcode-containing products that are not in the scope of this research.

**Fruits-360** [7] stands out as a deep (~65,000 images) and wide (95 categories) dataset that contains products usually not identified by barcodes, but by customer’s selection at self-checkouts. Images within a category appear to be of the same sample (or a few samples) rotated in various ways, which implies a lower variability of images than it is encountered in the real world. Although the dataset contains the most popular fruits, mapping categories between Fruits-360 and the authentic self-checkout dataset remains a challenge due to differences in product names, and different categorization hierarchies between the datasets. In addition, Fruits-360 [7] only covers a small portion of barcodeless products processed through self-checkouts: it is a subset of the "Fruits and Vegetables" category that does not contain vegetables, candies, dried fruits, biscuits, etc. Therefore, Fruits-360 is not representative of the self-checkout products dataset and has limited use for the study of self-checkout product recognition.

A few publicly available image sets contain packaged products, which usually contain barcodes. Since barcode-carrying products are easily identifiable by scanning a barcode, the next three datasets are less applicable to the research of computer vision-based product selection assistance at self-checkouts.

**RPC: A Large-Scale Retail Product Dataset** [35] contains ~83,000 images and is by far the largest retail dataset open to the public. It was synthetically assembled in a sterile environment using cameras from the top, 45- and 30-degree angles, and horizontal views. The background is uniform and, therefore easily removable. This dataset which contains 200 categories is comparable to the dataset of our research. Images usually contain more than a single product and are labelled by bounding boxes, making this dataset more usable for tasks other than classification.

**MVTec Densely Segmented Supermarket Dataset** (MVTec D2S) [34] is a supermarket products dataset of ~21,000 images and 60 categories, which is a fraction of categories that supermarkets carry. The dataset has instance labels at the pixel level that allow extensive data augmentation techniques to be applied.

**GroZi-120** [9] image set contains ~12,000 images of 120 products, a combination of product images from the web and actual grocery store shelves. Several smaller retail product image sets are made available to



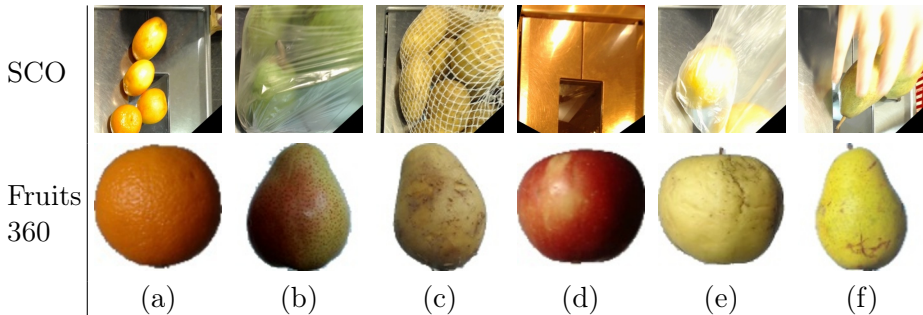


Figure 9: Image samples in the authentic SCO and Fruits-360 datasets

the public in the papers [105], [106]. These contain up to 10,000 images.

Of the described existing retail product image sets, Fruits 360 [7] is the closest match to the authentic self-checkout products database in terms of fitness to develop product recognition solutions for self-checkouts. Figure 9 illustrates sample images from the self-checkout dataset collected for this research in comparison to Fruits 360 [7]. SCO dataset, which is collected and filtered using only fully automated techniques still contains some products in bags (b, e), products covered by hands (f), and empty images (d); Fruits 360 excludes any of that and contains only well-visible products. The SCO dataset contains the background of checkout scales, but Fruits 360 has the background removed.

Table 2 compares the SCO dataset’s statistics against the most similar public dataset Fruits 360 [7]. The comparison is made after the elimination of empty images and images with poor product visibility from the former set. The SCO dataset (the automatically collected set) has a lot higher variance of image counts among different classes. The SCO dataset contains 5 times fewer images per class (137 on average) than Fruits 360 (690 on average) - a fact suggesting that performance metrics on the SCO dataset might be improved by collecting a larger set.

Table 2: SCO vs. Fruits 360 dataset

	<b>SCO</b>	<b>Fruits 360</b>
Classes	194	131
Total images (pre-balancing)	~26,600	~90,000
Images per class: Min/Avg/Max	3 / 137 / 3111	396 / 690 / 1312

### 3.6 Dataset Cleaning

Manual removal of empty images is not a viable solution due to constant change in assortment, resulting in the need to periodically refresh the dataset and retrain models.

It was considered to apply background removal techniques for automatic image labelling with object visibility labels (big foreground mask meaning high visibility), but such techniques would have treated customer body parts as foreground objects. Due to the high variety of products, image segmentation for automatic labelling was disqualified.

The raw dataset was cleaned by removing images that had poor product visibility. The visibility predictions were obtained from the classifier trained on visibility labels as described in section 3.3.

### 3.7 Image Pre-processing and Augmentation

The bigger part of self-checkout images area usually contains a background that is irrelevant in recognizing products. In order to focus neural network training on the foreground features, experiments were performed removing static background using GMM [27] before training. Figure 10 depicts a sample with a background (left), a mask obtained using GMM (middle), and an image with the background removed (right). In order to eliminate small foreground patches and fill small foreground mask gaps within products, morphological opening/closing was applied on background masks. Experiments of removing static background by applying GMM [27] and further applying morphological opening/closing on a mask before training negatively impacted validation accuracy in early experiments and were excluded from the final pipeline.

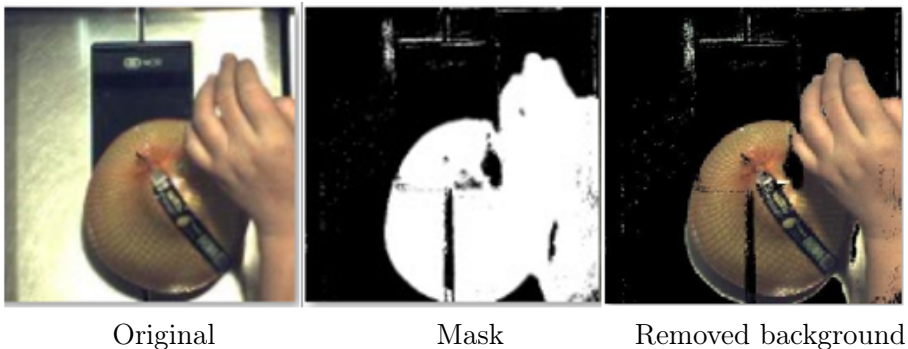


Figure 10: Background removal using GMM

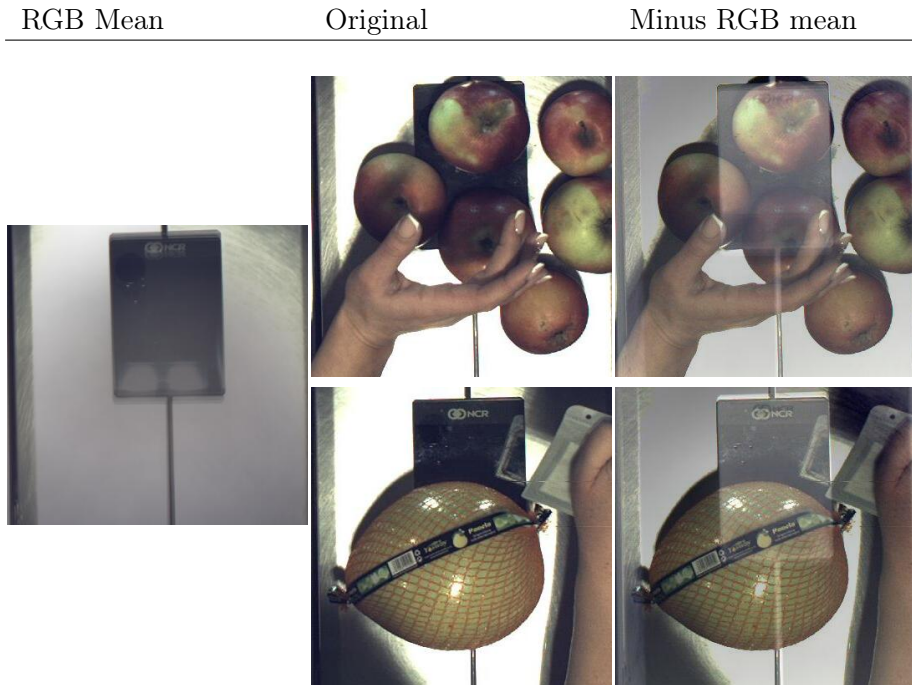


Figure 11: Subtracting RGB mean from the images. Since the mean image contains a darker spot in the scanner area in the middle, the resulting images tend to be lighter in that area. This distorts the resulting images unnaturally, making products lighter when placed inside the scanner area, and darker when placed outside of it

Almost all self-checkout images show a non-uniform illumination effect. The non-uniform illumination becomes a problem if not eliminated when training images are collected under different illumination than inference images. To reduce variance in image illumination intensity, the following pre-processing techniques (one at a time, in no particular sequence) were applied on the train, validation and test sets; then trained and evaluated classification models of each:

- Subtracted RGB mean<sup>1</sup> of the self-checkout instance; sample images in Figure 11. The darker scanner area (lighter in resulting images) tends to distort the resulting images unnaturally. The technique was omitted from further data preprocessing;

<sup>1</sup>A self-checkout instance mean image is generated with the same dimensions as the individual self-checkout images. Here, the pixel values at each (x, y) location are computed by averaging the corresponding pixels collected from that particular self-checkout instance within the dataset.

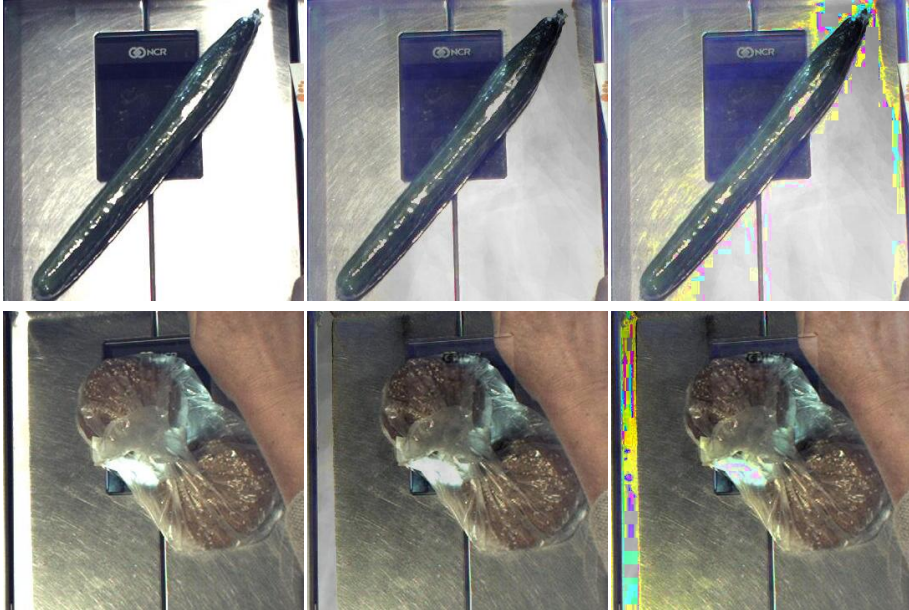


Figure 12: Subtracting the mean intensity. The original image on the top shows a high-intensity region on the bottom right, and the image on the bottom has a highly lit area on the right. By subtracting mean intensity (Value channel's mean in HSV or Luminance channel's mean in HLS colour spaces), the goal of eliminating highly lit areas is achieved. A natural look is preserved in HSV-mean(V), whereas unnatural patches of blue/yellow are observed in HLS-mean(L)

- Subtracted mean<sup>1</sup> V channel in Hue-Saturation-Value (HSV) colour space of the self-checkout instance; sample images depicted in Figure 12 column HSV minus V mean; the resulting images seem natural and non-uniform illumination issue is reduced. The technique was used in the final data preparation;
- Subtracted mean<sup>1</sup> L channel in Hue-Lightness-Saturation (HLS) colour space of the self-checkout instance; sample images depicted in Figure 12 column HLS minus L mean; the resulting images obtained unnatural colour patches. The technique was omitted from further research;
- Equalized image's intensity (channel V of HSV colour space) histogram as shown in Figure 13. Although the method equalizes intensity across different images, intensity across different regions

within the image remains uneven;

- Equalized image intensity histograms in smaller image patches using AHE. The sample in Figure 14 shows the effect of AHE using different size tiles: the bigger number of tiles (8x8 and more) results in overamplified contrast;
- Applied CLAHE [107] on HSV V channel. In addition to splitting an image into smaller regions (tiling), the method puts an upper limit on the contrast between the neighbouring pixels. Sample CLAHE-adjusted images are depicted in Figure 15.

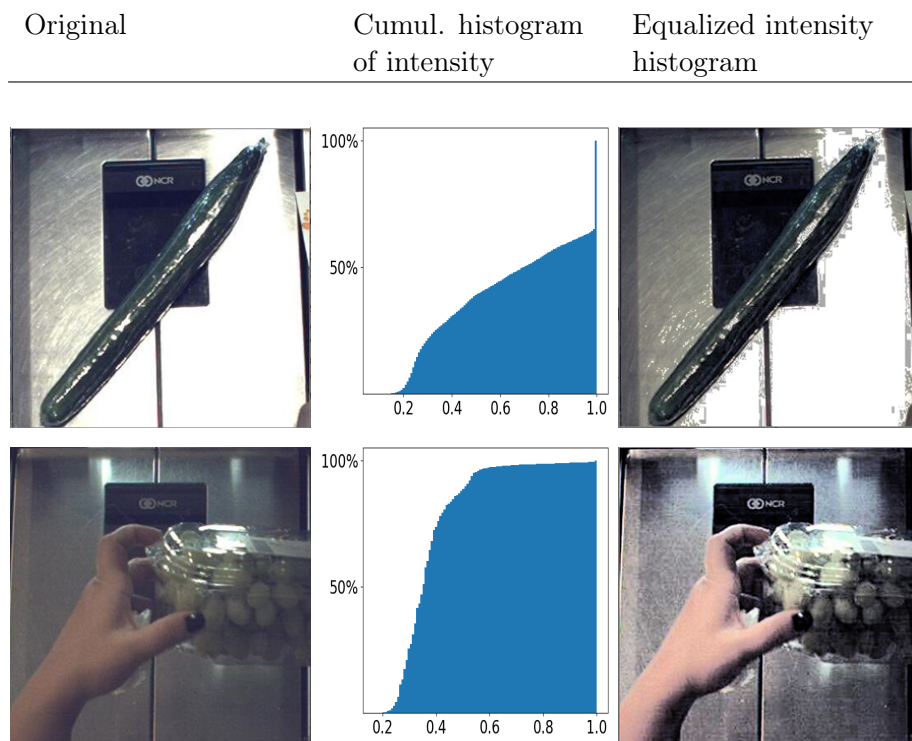


Figure 13: HE. Originally different-intensity images (left, top vs. bottom) gain similar intensity after intensity HE (right). Intensity differences in regions of images remain seen, such as two light reflections in the bottom image area at the top

The final dataset pipeline includes cropping the scanner/scales area and applying CLAHE [107] on the HSV channel V. Training on images without applying CLAHE showed a negative impact.



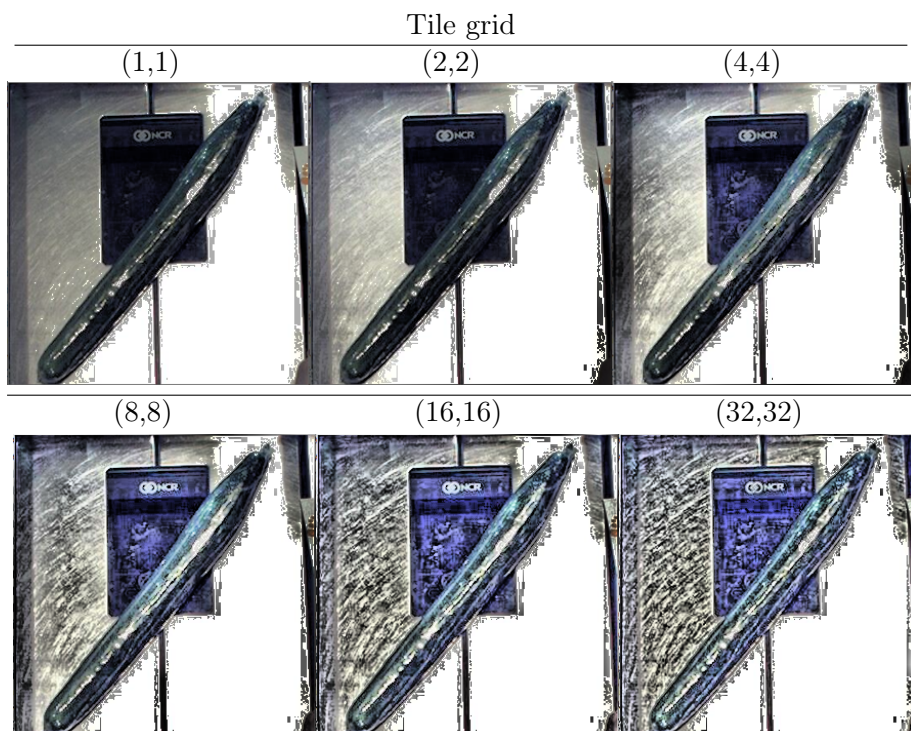


Figure 14: Adaptive Histogram Equalization. The tile grid size (1x1) corresponds to HE, whereas using a bigger grid size (smaller patches) equalizes smaller regions. Except for the flat-white patch in the right bottom, the intensity across regions is reduced with smaller patches (8x8 and more). The contrast is overamplified, and the image looks unnatural using smaller patches (8x8 and more)

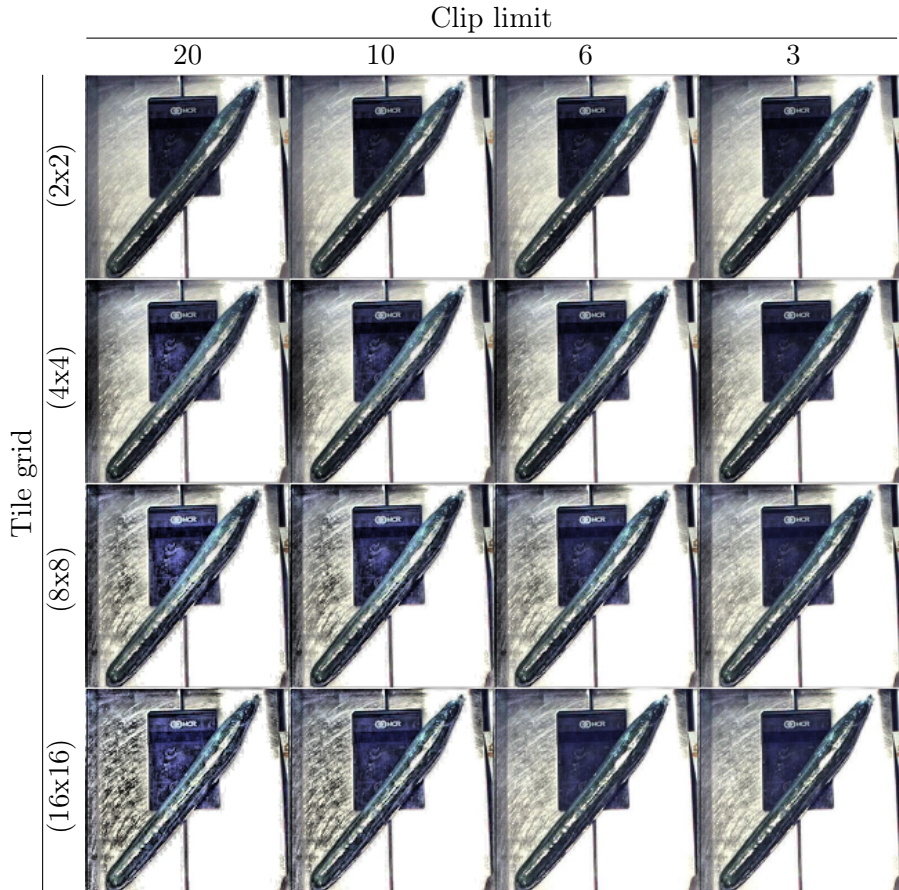


Figure 15: CLAHE [107]. Using a big clip limit corresponds to AHE, whereas a smaller clip limit de-amplifies contrast in overamplified-by-AHE regions. The method results in more natural-looking images than AHE, and reduces intensity differences across regions and differently-lit-images

Image corner distortion was applied as shown in Figure 16. Ranges of up to  $\pm 10$ ,  $\pm 20$ , and  $\pm 30$  pixels were used to choose a random distorted corner location. Eventually, two balanced datasets were made by concatenating: 1) affine transformed + 20-pixel corner distorted datasets and 2) affine transformed + 10, 20, and 30-pixel corner distorted datasets. The concatenated datasets contained  $\sim 1$  million and  $\sim 2$  million images respectively. Individual product classification results are reported using each dataset.

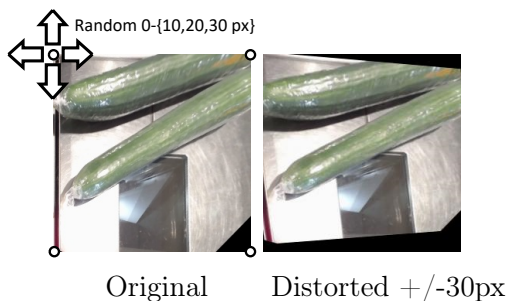


Figure 16: Corner distortion

The images were augmented in various ways in order to experiment with the augmentation technique’s impact on accuracy. The following basic affine transformations were applied: rotation (random up to 10 degrees), horizontal and vertical shift (random up to 32 pixels), zoom (random up to 10%), and horizontal flip (random 50% probability). Small enough augmentation parameters were chosen so that augmented images still mimic real photos taken by the checkout camera. Experiments of eliminating any one augmentation parameter or reducing the augmentation range by half led to a decline in validation accuracy.

Despite balancing the dataset through oversampling and augmenting the oversampled images, initial experiments revealed a significant accuracy gap between the training and validation data, indicating overfitting. To mitigate this issue, additional diverse training data was required. This was addressed by implementing dynamic augmentation: altering the underlying training minibatch randomly within specified ranges in every training iteration. This effectively used a different dataset in every epoch. The dynamic augmentation parameters were: rotation (random up to 10 degrees), shifting (random up to 32 pixels), zoom (random up to 10%), and horizontal flip (random 50% probability). In order to pick the optimal values, experiments were run with different dynamic augmentation parameters: they were doubled, tripled, halved, one was left out, augmentation was left out for validation set. Using



dynamic augmentation showed increased validation accuracy in early experiments.

### 3.8 Dataset Structuring for ML Tasks

Table 3 lists the datasets used in the self-checkout pipeline; Figure 17 shows how each was created. Dataset #1 is a raw image set of images

Table 3: Datasets used in the self-checkout pipeline

No	How obtained	Labels	Source of	Image count	
				Original	Balanced
#1	Auto-collected at SCO	Product ID	-	26.6K	-
#2	Manually labelled part of #1	Visibility (Q1-Q4, Bag, BagR)	Visibility classifier	6K	11.5K
#3	Filtered #1 using visibility classifier	Product ID	Product classifier, verifier, grouper	18.1K	500K

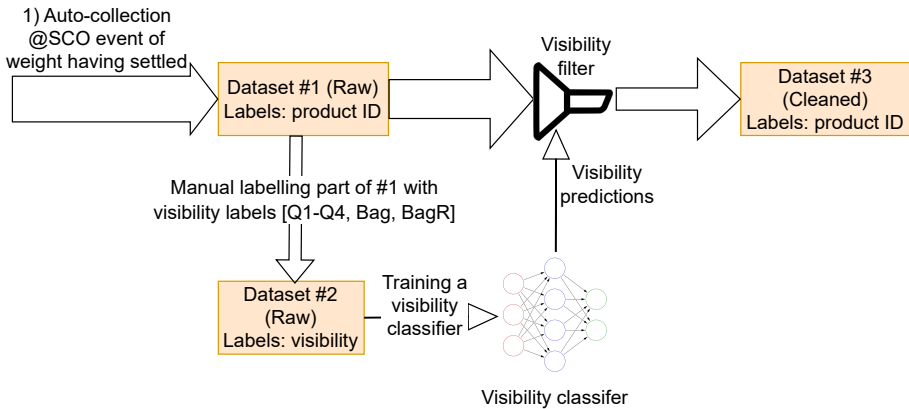


Figure 17: Flow for automatically preparing cleaned dataset #3, from the raw dataset #1 with a sizable portion of images containing invisible products, involves a proxy dataset #2 with manually set product visibility labels. A visibility classifier, trained on dataset #2, is used to filter the entire dataset #1 of poorly-visible-products containing images. Dataset #3 does not contain the worst visible category Q1, and experiments were run on whether the inclusion of the BagR visibility label is needed

collected at self-checkouts and labelled by customer-chosen product ID. Dataset #2 is a small subset of dataset #1 obtained by manually labelling images with product-visibility labels (described in section 3.3). A classifier trained on dataset #2 was then used to filter out poorly visible product images from dataset #1. Dataset #3 was used in all the further tasks of the research: product classification, verification, and grouping.

Datasets #2 and #3 were stratified and split into train, validation, and test subsets. First, 20% of the datasets were allocated for testing. Then, 20% of the remaining data was allocated for validation and 80% for training (effectively allocating 16% and 64% of the entire data for validation and training respectively). The train subsets were used to train classification models (dataset #2 to train the visibility classifier and dataset #3 to train the product classifier). The validation subsets were used to tune model hyperparameters and stop model training early. The test subsets were used to evaluate the trained models.

Both datasets #2 and #3 turned out unbalanced as shown in Table 1 for dataset #2 and Figure 7 for dataset #3. The train and validation subsets were balanced by oversampling [108] up to the count of images of the biggest class (as shown in Figure 18). The oversampled images were augmented. The test subsets were left intact since they contain a real-world representation of image distribution, therefore real-world classification metrics can be measured against it.

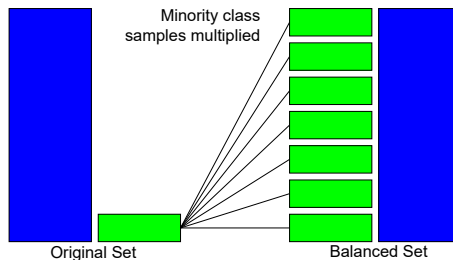


Figure 18: Oversampling

### 3.9 Conclusions of the Section

In this section, analysis of the most proper data collection strategy was done in order to create an authentic dataset of products at self-checkout. Images reviewed were taken from three different locations attached to the self-checkout device: above a self-checkout screen, below the screen, and on a side of the screen. The most proper location for camera mounting turned out to be on the side of a screen, whereas other locations

contained interfering body parts (when mounted above the screen) or a non-static store background (below the screen). The timeline of how a camera image changes after a customer has chosen a product has been investigated every 0.2 seconds, up to 1.0 seconds following the selection. The most proper moment for recognition turned out to be immediately following the selection moment. Although some images taken after a delay of 0.2—1.0 seconds have interfering body parts removed, often the salient object (sales item) is also removed. The task at hand being the recognition of a product, analysis was done for the most proper crop within a camera image. The resulting crop was chosen to fully contain the self-checkout scales area, which is where the product is usually placed at the time of customer selection.

Analysis of the collected authentic dataset showed the following characteristics. About 32% of the images contain body parts so that only up to 1/4 of a product is visible, the rest is covered. Approximately 10% of the products are in plastic bags. The bags, although transparent, sometimes reflect the light in such a way that a product is unrecognizable. Illumination of images is non-uniform; although similar between images of the same self-checkout, but differs greatly between self-checkouts. The technique that reduced differences in illumination, but preserved the natural look was CLAHE. Other techniques tried (HE, AHE, removing average intensity in various colour spaces) had a worse impact. Adopting CLAHE improved classification accuracy in the preliminary experiments.

The collected dataset had the following properties: a total of 26,637 images within 194 products. "Fruits and vegetables" was by far the biggest higher-level category by image count (86.9%), whereas split by the number of products within higher-level categories was more evenly distributed: candies contained 46.3% of the products; fruits and vegetables contained 37%; dried fruits and nuts, bakery contained less distinct products. The most frequent products in the dataset were bananas (12% of all the images), carrots (10%), and lemons (8%), whereas most categories contained less than 100 images. To balance the dataset, all except the biggest class images were oversampled and augmented using perspective and affine transformations. Augmenting oversampled images improved preliminary classification accuracy.

Augmenting images using both affine and perspective transformations showed the best preliminary recognition results. Using three perspectives outperformed a single perspective. Higher image variability using both affine and perspective transformation outperformed affine-only transformations and perspective-only transformations.

To prepare for training a recognizability classifier, a small part of the automatically collected dataset was labelled with recognizability

labels. After careful review of images, the labels were assigned by product visibility percentage (Q1-Q4), plus two labels for products in plastic bags (Bag - for easily recognizable products to the human eye; BagR - bags with light reflection that makes products unlikely to be recognized). This subset with recognizability labels was used to train a recognizability classifier, which in turn serves for a) filtering the full dataset off of unrecognizable product images and b) in the production pipeline to decide if a product can be recognized.

Existing datasets of retail food products that could potentially be a replacement for an authentic self-checkout products dataset, were reviewed. The review distinguished Fruits-360 (65,000 images within 95 products) as the closest match, although still too different to serve as a replacement.

## 4 Methods

The operational pipeline for product recognition, as illustrated in Figure 19, comprises several stages. The fully automated segment (highlighted with a green background) includes image collection, labelling based on customer selection of products, filtering out images with poor product visibility, training, and distributing trained models to self-checkout devices. This process is regularly run to accommodate changes in product list and appearance resulting from seasonal variations. Located at the bottom of the schema, the visibility classifier is trained using manually provided labels, which cannot be obtained automatically. Its primary functions are to filter the automatically collected dataset before training and to assess product visibility before classification or verification at self-checkout. Unlike other components, the visibility classifier is less dependent on updates to product lists or appearances and therefore does not require periodic retraining. The self-checkout system is responsible for classification, suggesting the most similar product or group of products to customers, and verification, alerting store staff when a selected product does not match its corresponding image, by utilizing machine learning artifacts from previous stages.

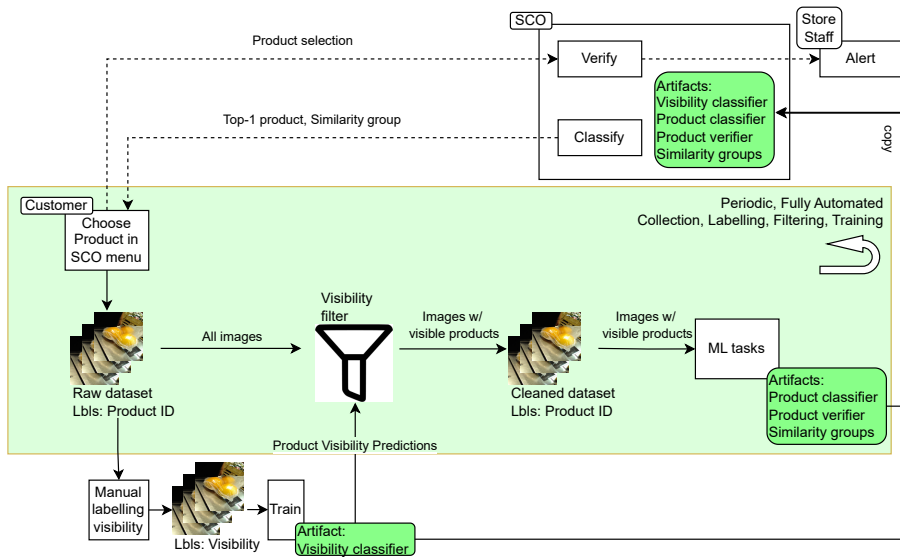


Figure 19: Proposed working pipeline in production

## 4.1 Image Fitness for Product Recognizability

To determine product recognizability in self-checkout images, the following questions were sought to be answered: 1) The minimum viable size of neural network architecture fit for recognition in the self-checkout images. 2) In what way the labels grouped into Visible/Invisible categories separate the images with the lowest error rate?

### 4.1.1 Deriving Minimum Viable Architecture

Image high-level feature extraction is a necessary step in any recognition task - classification, detection, and localization. A pixel is an image's feature of the lowest level. Object recognition in images based on individual pixel values is nearly impossible due to high dimensionality and high variance in pixels within images that contain the same objects. Therefore, extraction of higher-level features than pixels is needed to perform quality recognition. One of the leading ways to extract features from images is using convolutional filters, as in the papers [43] and [109]. A convolutional operation is visualized in Figure 20, and detailed in Eq. 1.

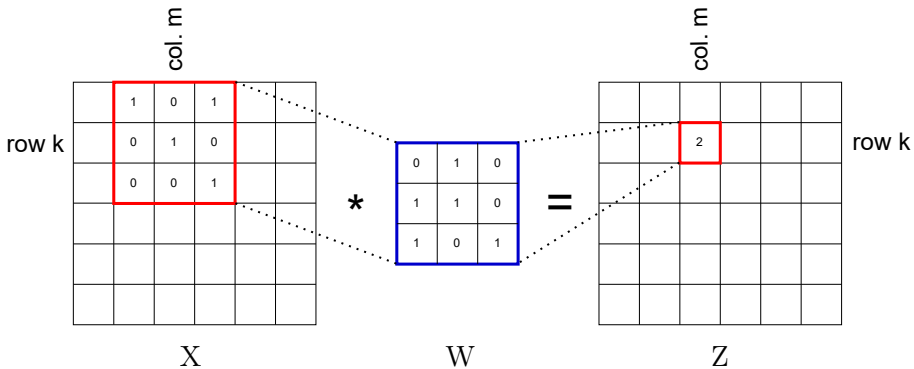


Figure 20: Convolution operation.  $X$  is the input matrix - either the input image or activations of previous layers;  $W$  is the convolutional filter of a size  $3 \times 3$  in this research;  $Z$  is the output matrix of the same shape as  $X$  (given the same padding and stride=1 is used)

Calculating convolutional output (when stride=1, filter size is odd)

$$z_{i,j} = \sum_{k=1}^s \sum_{m=1}^s (x_{i+k-\frac{s+1}{2}, j+m-\frac{s+1}{2}} \times w_{k,m}), \quad (1)$$

where

$s$  is a convolutional filter (kernel) height and width,

$x_{i,j}$  is an input  $X$  element in the  $i$ -th row,  $j$ -th column,

$w_{k,m}$  is a convolutional filter  $W$  element in the  $k$ -th row,  $m$ -th column.

Convolutional kernel size is the key feature of any neural network architecture. Generally, large kernels, such as 7x7 or 9x9, capture more global information and larger patterns in the input, whereas small kernels, such as 3x3 or 5x5, capture local features in the input data. Large kernels have more parameters, and thus are more computationally expensive to train. The kernels of size 1x1 do not capture interrelation of nearby pixels. Odd-size kernels have a "central" pixel that can be mapped from a deeper layer activations map into a previous layer; even size kernels do not have such a "central" pixel. Using even size kernels was shown to result in heavier distortions in deeper layers due to the latter property. Therefore, the smallest convolutional kernel size, that captures inter-relations of nearby pixels and has a "central" pixel is 3x3. The pioneering convolutional neural networks paper [43] used arbitrary filter sizes 3x3, 5x5, and 11x11 to classify ImageNet images. However, the authors of the VGG network [110] later showed on ImageNet that the smallest kernel size of 3x3 is sufficient, only if a neural network stacks enough convolutional layers. Due to these facts, kernel size 3x3 was chosen and other sizes were not experimented with; instead of using big convolutional filters, additional convolutional layers were stacked until training accuracy saturated.

Another hyperparameter of a convolutional kernel is stride - a step size that defines the movement of a kernel on an input layer, in order to compute a neighbouring pixel's value in the output layer. A minimum stride is 1 in either direction, which results in the same size output as input. Any bigger strides result in smaller feature maps and are computationally more efficient, but may discard critical fine-grained spatial information. On the other hand, the reduction of feature maps without losing the most important information can be achieved with pooling. Due to these facts, stride 1x1 was used in all the convolutional layers.

Images are very high dimensional data points (colour image dimensionality = height  $\times$  width  $\times$  3). Training effective neural networks

on high dimensional data requires either training data available in quantities correlated to dimensionality or lowering the dimensionality of the data. Since obtaining new data is often costly, the latter technique is commonly employed. Max pooling is a down-sampling operation used in convolutional neural networks for image data. It is primarily employed to reduce the spatial dimensions (width and height) of the input volume, thereby decreasing the computational complexity of the network and controlling overfitting. Max pooling operates independently on each channel of the input, and its main purpose is to retain the most important information while discarding less relevant details. It is worth noting that there are other pooling methods, such as average pooling, which takes the average value within each window instead of the maximum. Average pooling tends to "smooth out" the fine details, making it useful in tasks like style transfer. Max pooling, on the other hand, retains the most prominent feature or activation in each local region, making it a perfect fit for object recognition tasks. In this research, max pooling was used after each convolutional layer.

Fully connected (FC) layers are a type of artificial neural network layer where each neuron is connected to every neuron in the preceding layer. They enable the network to learn complex relationships and capture intricate patterns in the data. One or several FC layers typically follow convolutional layers in the paper [43]. The same approach was used in this research.

A non-linear activation function must generally follow every convolutional or dense layer in a neural network. Without activation functions (or with a linear activation function), a neural network, no matter how deep, would essentially behave like a linear regression model, and the model's capacity to learn and represent complex patterns would be severely limited. A commonly used activation function after both convolutional and dense layers (except the last output layers) is Rectified Linear Unit (ReLU) in Eq. 2, as in the papers [110], [111], [46]. The ReLU activation function is sometimes prone to so-called "dying ReLU" - a problem of some neurons stopping to learn due to negative input values. The problem occurs due to large learning rates, improper weight initialization, or badly conditioned inputs that contain mostly negative values. To solve the "dying ReLU" issue, variants of ReLU have been introduced that have a small non-zero slope when input is negative: leaky ReLU, Parametric ReLU (PReLU), and Exponential Linear Unit (ELU). In this research, ReLU was used after every convolutional and every dense layer (except the last), because it served the purpose of introducing non-linearity and the issue of "dying ReLU" did not occur during training.



$$\text{ReLU}(z) = \max(0, z), \quad (2)$$

where

$z$  is a convolution or dense layer output.

Activation functions that follow the last layer of a neural network differ in purpose from activation functions between convolutional or dense layers: they must "normalize" the input in order to prepare it for calculating the loss function. The generally accepted activation function for the  $N$ -class classification task is Softmax as in Eq. 3. Both the input  $Z$  and the output  $\sigma(Z)$  are vectors of length  $N$  (number of classes). The output vector  $\sigma(Z)$  is interpretable as probabilities for each of the  $N$  classes, and its sum equals 1. Softmax was used in all the classification tasks (visibility classification and product classification), as well as in a product verification task as explained later.

$$\sigma(z)_i = \left( \frac{e^{z_i}}{\sum_{j=1}^N e^{z_j}} \right), \quad (3)$$

where

$z_i$  is an element of the preceding layer output vector  $Z$  that corresponds with the  $i$ -th class,

$N$  is the number of classes.

Training deep neural networks is prone to a vanishing/exploding gradient problem unless some normalization between a preceding layer's output and a succeeding layer's input is introduced. A common normalization technique is called Batch Normalization (BatchNorm) as in Eq. 4. It norms values across the spatial dimensions and samples of a minibatch, but not across the channels. In addition, it has two trainable parameters scale  $\gamma$  and shift  $\beta$ , which help adjust the output values to different architectures and mitigate dependence on weight initialization. The Batch Normalization helping to solve the vanishing/exploding gradient issue was shown in the papers [112], [113]. A competing normalization technique to BatchNorm is Layer Normalization (LayerNorm). As opposed to BatchNorm, LayerNorm normalizes across spatial dimensions and channels. Because LayerNorm does not normalize across samples, it is useful when the data distribution varies across instances (such as NLP), whereas BatchNorm suits

scenarios where the training data has consistent statistics across batches. BatchNorm was used after most of the convolutional and dense layers.

$$BN(x) = \gamma \left( \frac{x - \mu(x)}{\sigma(x)} \right) + \beta, \quad (4)$$

where

- $x$  is input  $\in \mathbb{R}^{H \times W \times C \times M}$  (after convolutional layers),
- $\mu(x)$  and  $\sigma(x)$  are mean and standard deviation  $\in \mathbb{R}^C$ ,
- $\gamma$  and  $\beta$  are learnable parameters of the Batch Norm. layer  $\in \mathbb{R}^C$ ,
- $H$  and  $W$  are spatial dimensions height and width,
- $C$  is the number of channels,
- $M$  is the minibatch size.

Trained neural networks tend to perform better on data that was used during training than on unseen data, a phenomenon called overfitting. To reduce the gap in performance between training data and unseen data, various regularization techniques are used. The already discussed Batch Normalization often acts as a regularization technique that helps to prevent overfitting to training data as shown in the papers [112], [113]. Another technique, called L2 normalization, tends to reduce absolute values of trainable neural network parameters by adding a Euclidean norm of the weights to the loss function. Yet another technique called Dropout "cancels" neurons in each training iteration with probability  $p$ . The Dropout technique reduces the relative impact of any individual neuron, forcing each neuron to be more robust and less dependent on the presence of specific other neurons. The positive impact of Dropout was shown in the papers [114], [115]. In this research, dropout was used after every dense layer (except the last).

Training a neural network entails minimizing its loss function. The generally accepted loss function for an N-class classification task is called Cross-Entropy loss as in Eq. 5. Minimizing the Cross-Entropy loss encourages the model to make more accurate predictions by penalizing deviations from the true labels. Cross-Entropy was used in both visibility classification and product classification tasks, as well as in a class verification task as explained later.

$$L_{CE} = - \sum_{i=1}^N y_i \times \log(\sigma(z)_i), \quad (5)$$

where

$\sigma(z)$  is softmax vector  $\in \mathbb{R}^N$  (probabilities of classes) as in Eq. 3,  
 $y$  are class labels (one-hot vector),  $\in \mathbb{R}^N$ ,  
 $N$  is the number of classes.

The experiments were run to investigate different neural network setups. The classical convolutional neural network architecture [43] was used by varying numbers of convolutional and fully connected layers. At first, the focus was on reducing bias while leaving reducing variance for later: starting with one layer of each type, layers were added until training accuracy was saturated (validation accuracy not considered). The last dense layer contained a Softmax activation function, all others contained ReLu. A convolutional filter size was  $3 \times 3$ ; experiments of filter size  $5 \times 5$ , and  $7 \times 7$  were also performed. Network input was chosen to be  $256 \times 256$ , which is the nearest power of 2 smaller than the original image size. Every next convolutional layer was twice reduced in height and width using max-pooling and had about 2 times the number of convolutional filters (therefore, carried about 1/2 of the features of the previous layer). Next, the focus was on reducing variance and improving validation accuracy. Experiments were performed using Batch Normalization [112], Dropout [115], and L2 regularizations. Batch Normalization was initially applied after layers that showed sparse outputs, but then applied after all the other layers - both convolutional and dense. In addition to Batch Normalization, experiments were performed by adding dropout after dense (except last) layers. In addition to Batch Normalization and dropout layers, L2 regularization was applied and tested after various dense layers. The loss function was binary Cross-Entropy. The binary Cross-Entropy loss function was optimized using Adam [116] during training. At the end of each epoch of training, the model was evaluated using the validation set. Training the models was early-stopped after validation accuracy did not improve for the last 20 epochs, then parameters were reverted to those of the best epoch. Trained models were additionally trained by halving the learning rate. The final model was chosen by the best classification accuracy obtained on the validation set.

### 4.1.2 Visibility Thresholding Strategies

Given the dataset labelled with 6 visibility labels (#2 in Table 3), and a goal to filter out images having salient objects of subpar recognizability, the experiments were run to find the best-separating threshold between visible vs. invisible salient objects in images. Data labels were assigned in all possible ways into [Visible; Invisible] categories as shown in Figure 21 with the following restrictions:

- Q1 always Invisible;
- Q4 always Visible;
- Intermediate ordinal labels Q2, Q3 must adhere to their ordinal sequence: Q2 can't be Visible unless Q3 is, and vice versa;
- Similarly, BagR can't be Visible unless Bag is, and vice versa.

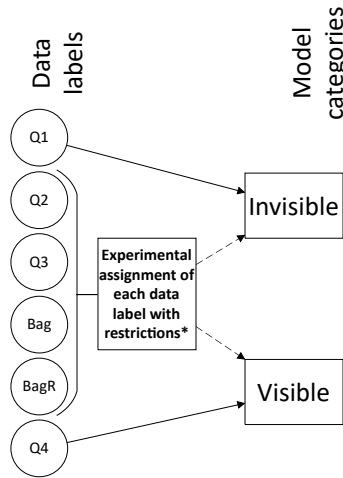


Figure 21: Visibility labels grouping strategy

In all experiments the same number of samples was used: data was undersampled when the model category [Visible; Invisible] contained more than a single label [Q1-Q4, Bag, BagR].

F-score was used as the main metric to evaluate the models. The f-score is a harmonic mean of precision and recall. F-score measures classifier quality more appropriately when classes are unbalanced (such as data in the research [117] or data in this research) than the most popular classification metrics: accuracy, precision, recall (sensitivity), and specificity. On the other hand, the f-score measure is comparable

to accuracy, etc. when classes are balanced. Variation of f-score -  $F_\beta$  that gives different weights to precision vs. recall - is useful when the cost of different error types (false positive vs. false negative) differ (not in the scope of this research). Cross-Entropy, although relevant to measuring classifier quality for unbalanced classes, gives higher weights to high-confidence mispredictions, but in this research, both high and low-confidence mispredictions are treated the same.

## 4.2 Product Classification

To classify products in self-checkout images with optimum accuracy, the following questions were sought to be answered: 1) By applying the trained classifiers of product recognizability for filtering out images, to discover their impact on the full product classification pipeline; determine the optimum recognizability classifier 2) Given that empty self-checkout images lend themselves to being filtered out not only by using recognizability classifiers, but other techniques (e.g. Siamese) as well, to discover how using different techniques for filtering impact the full classification pipeline; determine the optimum technique for filtering out empty images 3) Both steps above being optional in the product classification pipeline, to determine their usefulness by performing an ablation study 4) To compare well-known neural network architectures vs. self-made architecture optimized on the self-checkout image set for product classification performance; determine the optimum architecture.

### 4.2.1 Fully Automated Self-checkout Pipeline

The experiment pipeline is shown in Figure 22. The stage "Auto labelled images" is a software agent integrated with self-checkout software,

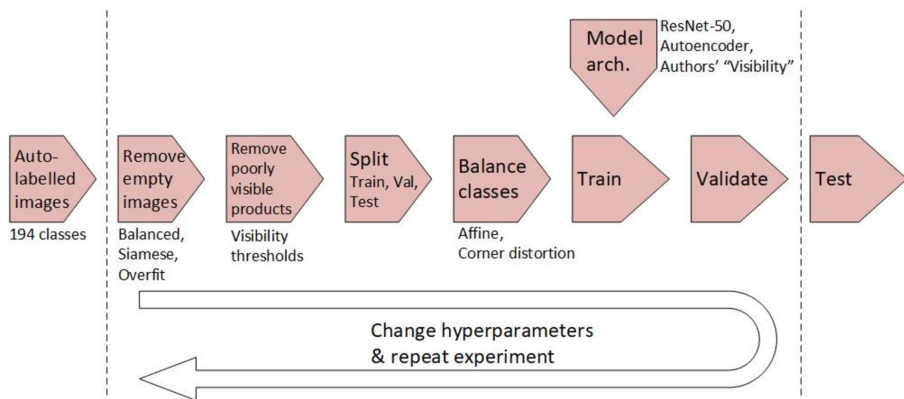


Figure 22: Experiment pipeline

capable of receiving events when a barcodeless product was chosen. The agent collected the dataset of image files labelled with product ID. Some of the images turned out empty (a customer had not placed a product on the scales). "Remove empty images" filtered out empty ones from the further steps of the pipeline (both during training and inference). The investigated techniques of classification (Balanced, Siamese, Overfit) are described in this section. A sizable portion of the remaining non-empty images contained products that were not visible enough for classification: interfering customer body parts and semi-transparent plastic bags with high glare were the primary reasons. Stage "Remove poorly visible products" in Figure 22 filtered out images unfit for product classification. The visibility thresholds were investigated using the resulting classifiers of recognizability subsection 4.1 as described in this section. The "Split" stage used stratified image assignment into the train, the validation, and the test sets. "Balance classes" stage oversampled and augmented under-represented classes of train, validation subsets to make the total number of samples per class approximately equal. "Model architecture" was chosen between off-the-shelf networks (Resnet-50, EfficientNet) and minimum viable architecture from the recognizability subsection 4.1 network tuned on the self-checkout dataset. "Train", "Validate", and "Test" are the next standard steps in the machine learning pipeline. The need for two stages that are not standard in the machine learning pipeline - "Remove empty" and "Remove poorly visible" - was justified by ablation studies (later in this section).

### 4.2.2 Filtering Empty Images

Three different techniques were involved to eliminate empty images when preparing the dataset for a further individual product classification task. First, a convolutional 2-class classifier (classes: empty; not empty) was constructed. The architecture of the model was taken from recognizability subsection 4.1, except the Softmax layer was set to 2 classes (Empty/Not Empty). Second, the Siamese architecture [28] was applied to create a "distinction" function between not empty and empty images. The Siamese architecture is detailed in Figure 23. It consists of two identical blocks (upper and lower in the diagram) that share weights. Although the inside of the twin blocks could be any neural network, a well-known Resnet-50 was chosen as it is relatively fast and accurate on well-known datasets. The first image of the Siamese network was always empty. The second image was alternated during training between semantically the same (empty) and semantically different (not empty). The difference between the twin blocks, followed by a Sigmoid, was labelled "1" when the second image was empty and "0" otherwise. Such a

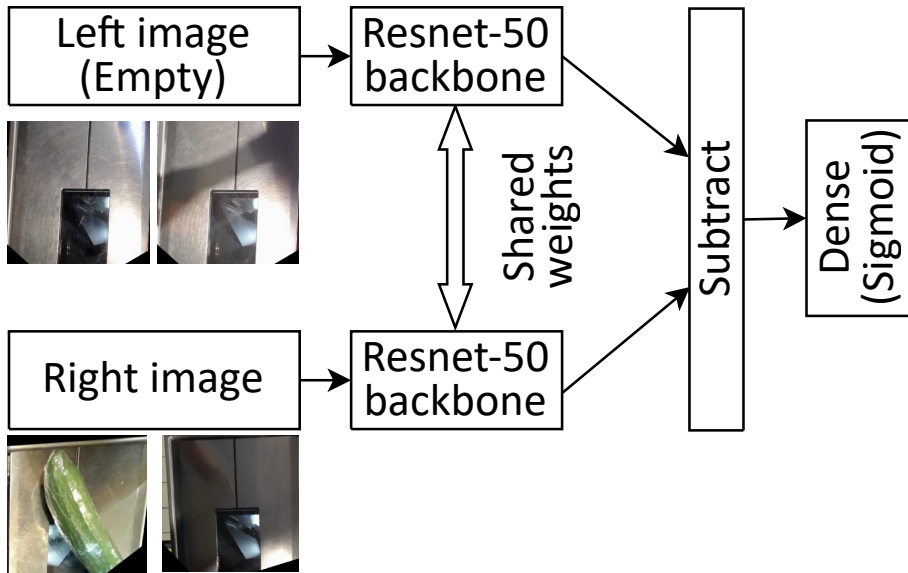


Figure 23: Siamese architecture for image emptiness

network learned a distinction function that could separate empty images from not empty ones. A small number of empty images was labelled, and it was assumed that the rest of the dataset was not empty images. Once the Siamese network was trained, it was used to separate empty images from the rest: each image in the dataset was compared to all of the labelled empty images. Third, the problem was framed as anomaly detection (not empty images labelled as anomalies) and it was solved by overfitting a neural network to empty images in the training set (such a technique resulted in a Sigmoid value of almost 1.0 for empty and lower than 1.0 for any other images). Each of the three techniques described above was used to remove empty images that resulted in 3 different datasets: Balanced, Siamese, and Overfit. Each dataset was used to train individual product classifiers and individual product classification results are reported on these 3 datasets in Results section 5.2.4.

### 4.2.3 Filtering Products of Low Visibility

Table 1 shows that ~32% self-checkout images have less than a quarter of a product visible, which implies that images must be filtered by product visibility before classifying in order to be useful. The resulting classifier of recognizability subsection 4.1 was integrated into the pipeline as a pre-processing step to filter out images having poor product visibility from further pipeline steps. Since 4.1 classified the dataset into 6 visibility categories (4 ordinal categories by product's visible part and 2 classes

for products in plastic bags), a hyper-parameter was introduced for which of the 6 visibility classes had to be included/excluded from individual product classification. Except for the obvious categories (worst visible Q1, best visible Q4), other categories (Q2, Q3, products in bags Bag, BagR) had to be experimentally checked for fitness to classify reliably into individual product classes. The recognizability study suggested two boundary options to decide if product visibility is sufficient: a) having less than a quarter of a product visible; b) having less than a quarter of a product visible and plastic bags with high glare. This resulted in 2 different datasets, one with all the images but the lowest visibility category Q1, and the other excluding plastic bags with high glare from the former set. Individual product classifiers were trained using each dataset; individual product classification results using these datasets are reported in the Results section 5.2.4.

#### 4.2.4 Ablation Studies in the Fully Automated Pipeline

Two stages in the pipeline of this study are not standard in machine learning: filtering out empty images and filtering out images where product visibility is unsatisfactory. To discover the usefulness of these stages, ablation studies we performed by removing each of them. The two stages - filtering out empty images and filtering images with poor product visibility - could possibly be built using a single classifier. The latter stage - filtering out poor product visibility - was trained using some empty images in the lowest visibility category Q1, thus could potentially separate out empty images. However, empty image properties lend themselves to techniques simpler than the multi-class classification that were investigated; using resulting classifiers of product recognizability subsection 4.1 allowed this research to focus on individual product classification. Discoveries are presented in the Results section 5.2.1.

#### 4.2.5 Architecture Alternatives

The architecture was investigated by using the final dataset (best accuracy showing dataset obtained by having eliminated images with unsatisfactory product visibility and having balanced under-represented classes using techniques described above). Four groups of architectures were investigated: ResNet-50 [111], EfficientNet [18], auto-encoder based, and own "Visibility" architecture resulting from recognizability subsection 4.1. ResNet-50 was chosen for its proven ability to fine-tune using pre-trained models on specific tasks with smaller datasets, such as in the papers [118], [119]. EfficientNet's recent success on ImageNet, yet its relatively small size (5.3M parameters) is a viable option on



low-powered self-checkout machines. Auto-encoder based architectures benefit from self-supervised training when labelled data is not sufficient, such as in this research. Using the simplest EfficientNet version B0 took much longer training time (~3 times longer per epoch) and showed initial results of comparable accuracy to the recognizability study’s best architecture. As a result, training duration also being a factor in real-world retail environments, even more complex EfficientNet versions B1-B7, or other architectures such as VGG [110] (133M parameters) were not investigated.

Training an auto-encoder and using it to pre-train a classifier is a strategy known as pretraining or transfer learning. This approach can be beneficial in scenarios where the amount of labelled data for the target classification task is limited - the case in this research. Pretraining an auto-encoder allows the model to learn useful features and representations that can then be transferred to the classification task. The trained encoder is used to extract features from the input data. The encoder’s output in the latent space serves as a compressed representation of the input. A new classification layer on top of the extracted features is attached. This layer is responsible for making predictions based on the encoded features. The entire model (auto-encoder + classification layer) is trained on labelled data.

- The resulting architecture of recognizability subsection 4.1 was applied by changing the last Softmax layer size to 194 (the number of classes used for this research). Unlike in the other architectures where pre-trained weights were used all the weights were randomly initialized. All the other architecture hyper-parameters such as layer depth, convolutional filter size and count, activation

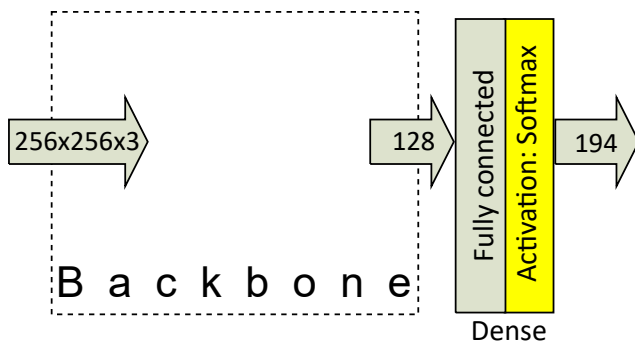


Figure 24: Product classification architecture contains the same backbone as recognizability architecture (Figure 34), but Softmax size is set to the number of classes

functions, max-pooling layer size and stride were left unchanged from the original architecture. The architecture is detailed in Figure 24.

- EfficientNet B0 with Softmax layer changed to 194 neurons (number of classes) was trained using pre-trained weights on ImageNet, but no layer weights were fixed.
- ResNet-50 with pre-trained convolutional layer weights on the ImageNet dataset. 1–2 dense layers were added (excl. Softmax) ranging in size from 128 to 1,024 neurons. A dropout layer was optionally added after each dense layer. Convolutional layer weights were left fixed.
- Auto-encoder-based architectures were created by first training a convolutional auto-encoder (U-Net). Each convolutional layer was followed by a max-pooling layer of size 2x2, stride 2; each next convolutional layer had about 2 times more filters (effectively, information size was cut in half after each convolutional block). Starting with the original image size of 256x256, experiments were performed with bottleneck layer sizes of 16x16 (32 filters) 8x8 (64), and 4x4 (128). Once auto-encoders were trained, the encoder part with fixed weights was appended with a few dense layers at the end: 1–2 dense layers of size 128–1,024 neurons plus Softmax layer was experimented on. A dropout layer was optionally added after each dense layer.

#### 4.2.6 Classifier Training and Evaluation

All of the experiments were carried out 10 times using all of the pipeline stages (reverse-arrow in Figure 22). Data was randomly re-sampled into train, validation, and test sets in every experiment (effectively similar to 10-fold cross-validation).

Models were trained using Adam [116] optimizer, default learning rate 1e-3. The training was performed for up to 100 epochs with early stopping if validation accuracy did not improve for the last 10 epochs. After training, the model’s weights were restored to those of the best epoch.

Training took 75-81 minutes per epoch, with early stopping due to saturated model metrics after no more than 30 epochs. Inference in a typical self-checkout computer with an Intel Core i3 CPU took 0.75 seconds.

Top-1 classification accuracy was measured on the unbalanced Test set. The results are reported in section 5.2.

## 4.3 Product Verification

To verify product selection in the self-checkout menu with optimum accuracy, the following questions were sought to be answered: 1) Given the concept of Centre-Loss layer in the paper [2], evaluate its effectiveness on the self-checkout image set for verifying product selection 2) Compare the effectiveness of class prototype-based verification technique using Centre-Loss against widely-used sample-to-sample techniques Siamese [28] and Triplet [54]. 3) Explore whether alternative distance metrics, beyond Euclidean, could enhance similarity measurement between image embeddings.

### 4.3.1 Concept of a Class Verification Task

Real-world computer vision tasks include a need to verify claims that an image contains a claimed type of object. A popular research area of class verification is face verification (the papers [90], [91], [28], [54], [2], [57], and [56]), where a class represents a person. In face verification, the computer vision task is to verify if a person in an image is the same person he/she claims to be. The negative samples are usually the ID of another person than in the image. Another popular research area in class verification is predicting image authenticity, given an image and a class in that image (the papers [120] and [121]). The negative examples are usually images generated by conditional generative adversarial networks (GAN). In food retail self-checkouts, a computer vision task attempts to check if a product in an image is the same product as a customer's chosen one. Negative samples are images of products other than the customer's choice.

The class verification task is a binary classification task that takes two inputs: an image and a class ID (a class is a product in this research and a person in the paper [2]). An image contains one of the following: a) an object of the same class as the input b) an object mismatched with the input class ID or c) out-of-distribution input (OOD - any object of the unknown class or no object at all). The goal of the class verification task is to separate a) ("Correct") from the rest ("Incorrect") - whether an object in an image belongs to the claimed class or not. Class verification does not need to distinguish between b) and c) whether an object in an image is of any other known class, an unknown class, or does not contain an object at all.

The class verification task differs from other classic computer vision tasks - classification and detection. Multi-class classification algorithms predict the distribution of probabilities only among a list of known classes. An image containing an object of an unknown class passed

to a classification neural network leads to unpredictable results, whereas the class verification task requires it to be rejected as "Incorrect" no matter what label is passed as input. Passing an image containing a known class' object to a classifier is likely to yield a higher probability for the correct class. Still, the probability boundary that separates correct class from incorrect is unknown. Thus, classification networks cannot be used for verifying class identities directly. Detection networks usually consist of two steps: 1) predicting object locations with the highest objectness probabilities (whether an object of a known class exists) and 2) predicting probability distributions of the patches between the known classes (classifying). Thus, detection algorithm errors in predicting objectness are penalized differently from errors classifying known objects. Nevertheless, class verification tasks are indifferent to the existence of objects other than the claimed class. Training detection networks require labels with bounding boxes around the objects, but class verification tasks are indifferent to object location within images - both during training and inference. Detection networks usually classify image crops having the highest objectness scores similarly to classification networks: they distribute class probabilities among the list of known classes. Therefore, a similar lack of boundary between correct and incorrect classes in detection networks makes them directly unusable for verifying classes.

Despite the lack of proper loss function for verifying classes in image classification and detection neural networks, their ability to extract image features has been widely demonstrated in the papers [122], [123], and [124]. Knowledge transfer is widely used between different tasks. Therefore, the backbones of neural classification or detection neural networks are likely useful in class verification if the loss function is changed.

The class verification task relates to conditional outlier detection task (as in the papers [125] and [80]). Outlier detection algorithms draw a boundary between in-distribution and out-of-distribution samples and judge new samples based on their relation to that boundary. However, outlier detection tasks do not make an explicit attempt to transform space in such a way that all samples (of a single class) are placed nearby.

The class verification task has a wide variety of applications. In the context of face verification, the suggested class comes from a presented ID, which must be confirmed by class verification. In the context of self-checkouts, the suggested class is a customer's chosen product from a picklist menu, which must be confirmed by class verification. The proposed computer vision solution aims to reduce theft through class verification. This task involves two inputs: an image and a claimed class, which must be among the known classes. The image can contain

an object of the claimed class, another known class, an unknown class, or no object at all. The verification task’s goal is to distinguish the claimed class object from the rest (excluding differentiation between other known classes, unknown classes, and no objects). Unlike verification, classification only uses the image as input, producing a probability vector across known classes. While verification tasks are well-studied in security with human-face datasets, research using other datasets is limited.

Different research domains use various strategies to select negative [image; claimed class] pairs. In computer vision safety, the goal is to differentiate real images from those generated by Generative Adversarial Networks (GAN), as in the papers [120] and [121]. Negative samples for this task consist of GAN-generated images. In face verification, classes represent different individuals, with research often using faces from the same dataset paired with other individuals’ identities as negative samples (as in the papers [54], [2], [57], [90], [91], [28], and [56]). In the self-checkout domain, negative samples should include two types of images likely to be incorrectly chosen in a self-checkout picklist menu: 1) barcodeless images in the self-checkout dataset and 2) images of expensive barcode-containing products sold in the same stores. Self-checkout datasets often contain hundreds of images per class, whereas face datasets typically have fewer images per identity (Digi-Face 1M [126] has 11, CelebA [127] has 20, and Wider Face [128] has 9 images-to-identities ratio), sometimes even less in security applications. This research explores verification methods using a self-checkout dataset.

### 4.3.2 Class Verification Approaches

Research in the realm of class verification tasks spans multiple directions. One approach involves employing sample-to-sample comparison using neural networks, like Siamese. Another pathway explores the learning of class prototypes during training. During inference, sample-to-sample methods assess the image being verified by comparing it to one (or possibly several) images of the same class from the training dataset. In contrast, class prototype-based methods evaluate the image being verified by comparing it to the prototypes that were learned.

Siamese networks get their name from two identical branches that share weights. A sample Siamese network is depicted in Figure 23. Siamese network is a comparison technique predicting if two data points belong to the same class. Siamese network’s input is a pair of images, whereas a verification task’s input is an image and a proposed class. To apply a Siamese network in solving a verification task, one may supply to the network: a) a verification task’s input image and b) an image from the training set of the verification task’s proposed class. Siamese

networks predict based on the distance between embeddings of the last layer of the two input images. The Siamese network loss function is called a Contrastive loss as in Eq. 6. The function aims to minimize the distance between two datapoints when they belong to the same class ( $Y=0$ ); it aims for the distance to be high (at least margin  $\alpha$ ) when the two datapoints belong to different classes ( $Y=1$ ).

$$L_{contr}(X1, X2) = (1 - Y) \times \|f(X1) - f(X2)\|_p + Y \times \max(0, \alpha - \|f(X1) - f(X2)\|_p), \quad (6)$$

where

$X1$  and  $X2$  are input images,  
 $Y$  is 1 when  $X1$  and  $X2$  belong to the same class; 0 otherwise,  
 $f(\dots)$  are the image activations of the last network layer,  
 $\|\dots\|_p$  is the  $p$ -th Norm (distance),  
 $\alpha$  is the margin.

The triplet network is a variation of Siamese networks. Unlike classic Siamese, Triplet contains three identical branches with shared weights and takes three images as input. Two input images (Anchor and Positive) belong to the same class, whereas the third (Negative) belongs to a different class. Accordingly, the Triplet loss function in Eq. 7 takes three images as input. Similarly to Contrastive loss, it uses distances between the Anchor image and either Positive or Negative images. Unlike the Contrastive loss, it does not aim for absolute values in distances, but aims for the difference between distances to be high (at least  $\alpha$ ). Whereas the Contrastive loss either reduces the distance between the same class images or increases the distance between different class images in a single optimization step, the Triplet loss does both.

$$L_{triplet}(A, P, N) = \max(\|f(A) - f(P)\|_p - \|f(A) - f(N)\|_p + \alpha, 0), \quad (7)$$

where

$A, P,$  and  $N$  are Anchor, Positive, and Negative images,  
 $f(\dots)$  are image activations (embeddings) of the last network layer,  
 $\|\dots\|_p$  is the  $p$ -th Norm (distance),  
 $\alpha$  is the margin.

Sample-to-sample methods yield pairwise distances but lack aggregation capability. In verification tasks, they require selecting representative training samples to compare during inference and aggregating results across multiple samples. To address this, the paper [88] models intra-class distance distributions and measures class probability based on distribution parameters. The papers [129] and [130] use the Earth Mover’s Distance (EMD) to quantify the dissimilarity between the training set’s intra-class distance distribution and the distance distribution of the test sample compared to same-class training samples.

In certain class verification applications, there’s a need to run inference on hardware with limited storage and processing power, and without GPUs, like retail self-checkout computers. Previous research found that on machines with an Intel iCore3 CPU, inference for a single image takes about 0.75 seconds. However, attempting to select random subsets of training samples and varying the number of samples during inference can lead to unpredictable results. Moreover, when using sample-to-sample methods to compare against multiple training images during inference, the demands on storage space and computation time increase proportionally with the number of training images. Consequently, conducting inference for multiple images on low-powered self-checkout machines makes sample-to-sample methods impractical.

Unlike sample-to-sample methods, class prototype-based approaches compare the image being verified only against the class prototype. This results in significantly faster computation times ( $1 \times 0.75$  seconds compared to  $M \times 0.75$  seconds with sample-to-sample methods, where  $M$  is the number of images). However, there’s a research gap in comparing the verification accuracy between these two approaches. This study aims to fill that gap by evaluating their accuracy. Only if the class prototype-based approach demonstrates comparable or superior accuracy will it become the preferred choice for low-powered inference machines.

Many researchers typically follow a two-step process when quantifying image differences: first, higher-level features are extracted, and then the Euclidean distance is calculated. However, the choice of distance metric can impact verification results. In the case of measuring the distance between neural network embeddings (i.e., higher-level image features), alternatives like Cosine and various Minkowski distances (including Manhattan, Euclidean, and Chebyshev) are valid options. In contrast, distance types like Hamming, Jaccard, and Dice are more suitable for comparing binary data. It is worth noting that Minkowski distances are scale-variant, while Cosine distance is not. The  $p$  parameter in Minkowski distances allows for flexibility, with special

cases like Manhattan ( $p=1$ ), Euclidean ( $p=2$ ), and Chebyshev ( $p=+\infty$ ). Despite the prevalence of Euclidean distance in research, there’s a dearth of studies comparing it to other distance metrics in class verification tasks. In this research, the aim is to investigate how the choice of distance metric between embeddings affects verification accuracy.

### 4.3.3 Product Verification Using Distance from Class Centres

Class prototype-based class verification approaches learn class proxies (or centres) - usually points in the space of neural networks latent space, that can be compared against image activations. The class prototype state-of-the-art approaches were analyzed: Proxy-NCA [3] and Centre-Loss [2].

Proxy-NCA is an approximation of Neighborhood Component Analysis as in Eq. 8. The numerator of the loss attempts to increase the cosine similarity between a data point’s embeddings and its class centre; the denominator attempts to decrease the similarity between a data point’s embeddings and all the other class centres. Proxy-NCA measures cosine distance, which, by definition, loses the scale component.

$$L_{Proxy-NCA}(x) = -\log \frac{e^{s(f(x), p^+)}}{\sum_{p^- \in P^-} e^{s(f(x), p^-)}}, \quad (8)$$

where

$f(X)$  are image activations (embeddings) of the last network layer,  
 $p^+$  and  $p^-$  are class proxies of the same(+) class as X or different(-),  
 $P^-$  is a subset of all class proxies other than X’s class centre,  
 $s(\dots)$  is Cosine similarity.

Centre-Loss, a departure from traditional sample-to-sample verification methods, focuses on learning virtual class centres (Figure 25). These class centres represent the mean point of all the images of a given class, within the same space as image embeddings of a given neural network layer. The Centre-Loss network’s loss function incorporates a distance measurement between an image’s embeddings and its respective class centre. Optimizing this loss function encourages embeddings to move closer to their corresponding class centres ( $L_C$  in Figure 25a). Class centres are continually updated to reflect changes in embeddings due to weight updates, effectively bringing image embeddings closer to their respective centres, i.e. learning class prototypes.

The referenced methods, including Siamese, Triplet, and Centre-Loss, commonly employ Euclidean distance to calculate differences between



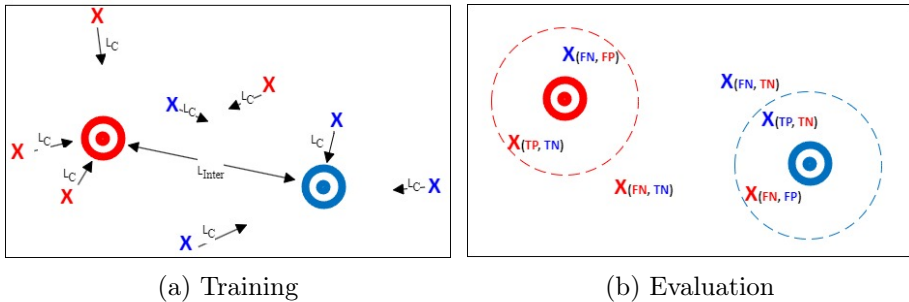


Figure 25: Centre-Loss, training and evaluation. The "targets" represent class centres (one per class). The "X"s represent data points (activations of a selected neural network layer). Different classes are represented by different colours. The arrows in (a) represent data point movement upon optimizing a loss function summand  $L_C$  in Eq. 9 and centre movement - upon optimizing  $L_{Inter}$  in Eq. 12. The dashed circles around class centres in (b) represent thresholds of verifying data points belonging to a class. Values in subscript at data points mark their verification predictions and correctness, e.g. the left-most red point  $X_{(TP, TN)}$  is correctly verified as a member of the red class (TP) and as a non-member of the blue class (TN)

sample-to-sample embeddings or sample-to-class embeddings. However, as one of the research goals was to compare various distance types, experiments were conducted, and the results are presented using not just Euclidean but also other distance metrics: Manhattan, Minkowski, and Cosine. Minkowski distance, which encompasses both Euclidean ( $p=2$ ) and Manhattan ( $p=1$ ) distances, offers numerous versions based on different integer values of  $p$ . For practical reasons, this research focused on  $p$  values within the range [1,4], taking into account resource and time constraints. The Results section illustrates how varying the  $p$  value impacts performance. In contrast, Cosine distance is scale-invariant, making its values restricted when measuring differences between data points with unknown scales. Consequently, Cosine distance was included in the research. However, other distance types like Hamming, Jaccard, and Dice were excluded due to their inability to quantify distances for non-binary values.

The Centre-Loss architecture experiments were started by reusing the classifier's backbone that is explained in detail in recognizability subsection 4.1. Presumably, the backbone's demonstrated performance on the same dataset for other tasks (recognizability and classification), implies that the architecture is fit to carry enough information through the network layers about the classness of sample images. In addition,

the architecture contains little parameters (3.2mln) compared to leading architectures on big sets like ImageNet - CoCa [84] (2100mln), ViT-G/14 [47] (1843mln), EfficientNet [18] (11mln and up). At the end of the backbone, two layers were added: a Softmax layer and a Centre-Loss layer, similar to the paper [2]. The Centre-Loss (CL) layer in Figure 26 takes two inputs: activations  $\in \mathbb{R}^{M \times Cnt}$  ( $M$  - number of minibatch samples;  $Cnt$  - count of neurons in the extra dense layer) and image labels ( $M$  one-hot vectors). It has an internal parameter of class centres  $\in \mathbb{R}^{N \times Cnt}$  ( $N$  - number of classes). The output of the CL layer is the difference vector  $\in \mathbb{R}^{M \times Cnt}$  between samples' corresponding class centres and samples' activations of the extra dense layer. After initial experiments in order to keep satisfactory metrics, an Extra Dense layer was added to the head of the model backbone: training without the Extra Dense layer did not saturate the loss function and did not achieve satisfiable separation in distances between "Correct" and "Incorrect" classes. The final model architecture is shown in Figure 26. Experiments

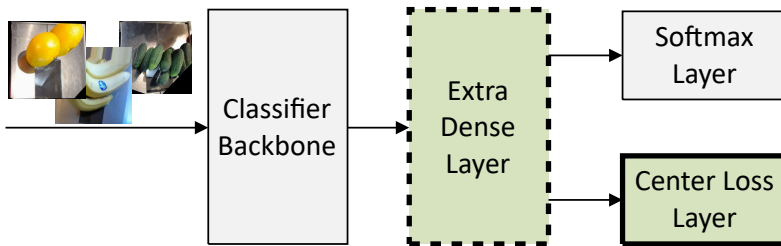


Figure 26: Centre-Loss model architecture. The backbone was reused from recognizability subsection 4.1 and classification subsection 4.2 shown in Figure 34. Besides the Softmax layer, the Centre-Loss layer outputs the sample embeddings distances from their respective class centres

were performed, and results were reported using different sizes of the Extra Dense layers.

The Centre-Loss approach defines the loss function but does not define a neural network architecture (such as pre-Centre-Loss layer selection, neuron count in pre-Centre-Loss layer) and requires tuning hyperparameters specific to Centre-Loss (such as Centre-Loss weight). Tuning the architecture and hyperparameters is described in this section below.

In the reference paper [2], The Centre-Loss architecture has two outputs: Softmax and Centre-Loss that are used to calculate  $L_{CE}$  and  $L_C$  in Eq. 9 respectively. The Centre-Loss layer is connected to the

last layer before Softmax (solid line in Figure 27), experiments were conducted to assess if better results could be achieved by connecting the Centre-Loss layer to different shallower layers. Ten experiments were performed, connecting the Centre-Loss layer to various layers beyond the Convolutional backbone (dashed lines in Figure 27).

$$L_{Verification} = L_{CE} + \lambda_1 \times L_C, \quad (9)$$

where

$L_{CE}$  is Cross-Entropy loss as in Eq. 5,

$\lambda_1$  is Centre-Loss weight,

$L_C$  is Centre-Loss ( $L_C^{Cosine}$  or  $L_C^{Minkowski}$ ).

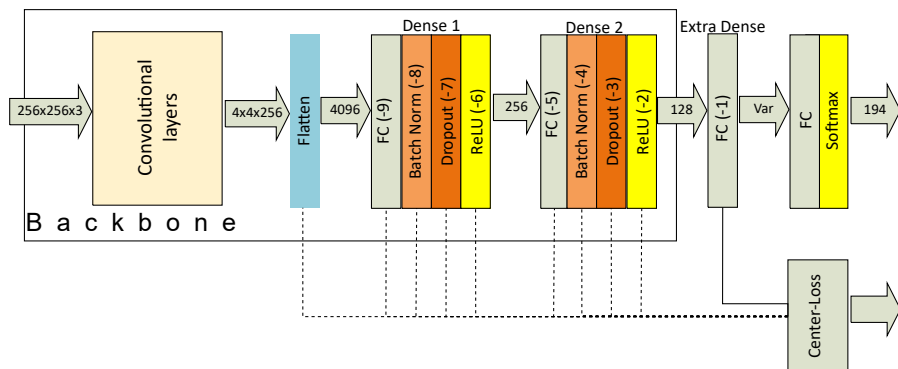


Figure 27: Architecture experiments of connecting Centre-Loss to various layers. Negative numbers in parenthesis signify layer indexes (referred to in the pre-Centre-Loss layer experiments). Lines connecting the Centre-Loss layer to its predecessor are marked in one solid line (best result) and nine alternative dashed lines (experiments performed on). The full structure of the backbone is shown in Figure 34

Optimizing the Centre-Loss architecture involved selecting the appropriate pre-Centre-Loss layer size, represented by the Dense 3 block’s FC layer in Figure 27. The experiments began with a small 2-neuron layer, then incrementally doubled its size. This process was continued until either metrics reached saturation or hardware limitations were encountered, with the maximum size being 2048 neurons.

In the Centre-Loss part of this research, the overall loss function (Eq. 9) comprises the weighted sum of two components. The first component, denoted as  $L_{CE}$ , utilizes the well-known Cross-Entropy loss

of the Softmax function (Eq. 5). Its role is to prevent all class centres from collapsing into a single point. The second component,  $L_C$  (Centre-Loss), imposes penalties on data samples based on their distances from class centres. Cross-Entropy loss (Eq. 5) preserves differences between classes. Without Cross-Entropy loss, the centres of all classes are likely to regress to one point. Cross-Entropy loss does not minimize differences between various samples of the same class. The absence of Centre-Loss leads to a classifier that provides class separability, but not sample concentration in the embeddings space.

The Centre-Loss component of the loss function, denoted as  $L_C$  in Eq. 9, aims to minimize the distance between various intra-class samples. A concept of the class centre is introduced: it is an average vector of all samples in that class of the extra dense layer’s activations. The sample’s distance from the respective class centre is calculated. The Centre-Loss component varies based on the chosen distance type. Minkowski distance types, including Manhattan ( $p=1$ ) and Euclidean ( $p=2$ ), are detailed in Eq. 10. In contrast, Cosine distance is specified in Eq. 11. Optimizing the Centre-Loss involves two main steps: 1) moving sample embeddings closer to their respective class centres, and 2) shifting class centres toward sample embeddings within a minibatch. The Minkowski-distance-based Centre-Loss (Eq. 10) generalizes the approach used in the paper [2] to accommodate any value of the parameter  $p$  for Minkowski distance, whereas the research [2] is limited to  $p=2$  (Euclidean). The Cosine-distance-based Centre-Loss (Eq. 11) computes negative cosine similarity within a range of 0—2.

$$L_C^{Minkowski} = \frac{1}{M} \sum_{i=1}^M \|x_i - c_{y_i}\|_p, \quad (10)$$

where

$M$  is the number of samples,

$x_i$  is the  $i$ -th sample’s activations of the extra dense layer,

$y_i$  is the  $i$ -th sample’s label,

$c_{y_i}$  is the centre of the  $y_i$ -th class,

$\|\dots\|_p$  is the  $p$ -th Norm (distance).

$$L_C^{Cosine} = \frac{1}{M} \sum_{i=1}^M \left(1 - \frac{x_i \times c_{y_i}}{\|x_i\| \times \|c_{y_i}\|}\right), \quad (11)$$

where

$M$  is the number of samples,

$x_i$  is the  $i$ -th sample’s activations of the extra dense layer,

$y_i$  is the  $i$ -th sample’s label,

$c_{y_i}$  is the centre of the  $y_i$ -th class.

#### 4.3.4 Increasing Inter-class Distance

While Centre-Loss aims to minimize distances between samples and their centres, it doesn’t attempt to increase the distance between centres of different classes. A possible enhancement was investigated to the Centre-Loss loss function to push class centres apart during training ( $L_{Inter}$  in Figure 25a), denoted in Eq. 12. The third component, referred to as  $L_{Inter}$  (Inter-Centre-Loss), applies penalties to class centres according to their cosine similarity within the range of 0—2, and is outlined in Eq. 13. The optimization of this particular loss component drives class centres away from each other.

The Inter-Centre-Loss component necessitates a relative weight hyperparameter, denoted as  $\lambda_2$  in Eq. 9. To determine its optimal value, initially the range of  $\lambda_2$  values akin to that of  $\lambda_1$  (as both pertain to distance in the same dimensional space) was explored. Notably, improved metrics were observed with smaller  $\lambda_2$  values.  $\lambda_2$  was sequentially reduced by a factor of 3.0 until results showed negligible differences from the case of  $\lambda_2$  being set to 0.

$$L_{Verification} = L_{CE} + \lambda_1 \times L_C + \lambda_2 \times L_{Inter}, \quad (12)$$

where

$L_{CE}$  is Cross-Entropy loss as in Eq. 5,

$\lambda_1$  is Centre-Loss weight,

$L_C$  is Centre-Loss ( $L_C^{Cosine}$  or  $L_C^{Minkowski}$ ),

$\lambda_2$  is Inter-Centre-Loss weight,

$L_{Inter}$  is Inter-Centre-Loss.

$$L_{Inter} = \frac{1}{M \times (N - 1)} \sum_{i=1}^M \sum_{\substack{j=1 \\ j \neq y_i}}^N \left(1 + \frac{c_{y_i} \times c_j}{\|c_{y_i}\| \times \|c_j\|}\right), \quad (13)$$

where

$M$  is the number of samples,

$N$  is the number of classes,

$y_i$  is the  $i$ -th sample’s label,

$c_{y_i}$  and  $c_j$  are the centres of the  $y_i$ -th and  $j$ -th class

#### 4.3.5 Product Verifier Training and Evaluation Details

The dataset of self-checkout products filtered out of invisible products (#3 in Table 3) was used in the verification study. The Centre-Loss models require data generators that produce two inputs [image;product ID] and an output [product ID]. The input product ID is used by the Centre-Loss layer to calculate distances from the respective class centres. The output product ID is used to calculate the Cross-Entropy error of the Softmax.

The Centre-Loss layer does not have any trainable parameters updatable by gradient descent. Yet, the internal parameter of class centres was updated in each iteration: only centres of the classes represented by the samples in a minibatch were updated, whereas unrepresented class centres were left untouched. Class centres were updated as shown in Eq. 14.

$$Centre = Centre + \alpha \times (Activations - Centre), \quad (14)$$

where

$Centre$  is the centre of a sample’s class  $\in \mathbb{R}^{cnt}$ ,

$Activations$  are the extra dense layer’s activations  $\in \mathbb{R}^{cnt}$ ,

$\alpha$  is the centres’ learning rate (hyperparameter),

$cnt$  is the number of neurons in the extra dense layer (hyperparameter).

Models were trained for up to ten epochs with a patience criteria of five epochs (i.e. training was stopped and best weights restored if the five last epochs of training did not improve the validation loss function value). Training on a relatively big training set of two million images (about 10,000 per class) usually saturated the training accuracy

in the first 1—2 epochs. Therefore a maximum of 10 epochs was never reached. The criteria for the best weights selection and early stopping was total validation loss, which is a weighted sum of Softmax layer, the Centre-Loss, and Inter-Centre-Loss. Adam [116] optimizer with a default learning rate of 0.001 was used. The training duration of 1 epoch on about two million images varied between 45 and 55 minutes.

The Centre-Loss function relies on a hyperparameter  $\lambda_1$  in Eq. 9 - relative weight of Centre-Loss component in the loss function. A low  $\lambda_1$  value prioritizes the softmax's Cross-Entropy loss, undermining the goal of bringing class embeddings closer to their respective centres. Conversely, a high  $\lambda_1$  value risks collapsing all class centres into a single point, rendering class differentiation impossible. In the original Centre-Loss paper [2],  $\lambda_1$  was empirically determined to be  $3e-3$ , with similar results achieved in the range of  $1e-4$  to  $5e-2$ . In this research, training was started within this range and systematically expanded by a factor of 3.0 in both directions until metric saturation was observed.

The training complexity of the suggested Centre-Loss approach is  $O(MN)$  (where  $M$  - the overall number of samples, and  $N$  - the number of classes): every sample's distance is calculated to every class' centre. The Siamese training complexity is  $O(M^2/B)$  (where  $B$  - number of batches), as every pair of samples is compared, but pairs are limited to the samples within a batch. The Triplet's complexity is  $O(M^3/B^2)$ , as every sample/anchor is compared to every positive sample and every negative sample. Still, triplets are limited to sample combinations within a batch, and every pass through data places every sample as an anchor once. The actual training ranged between 48—53 minutes/epoch for Centre-Loss, 68—83 for Siamese, 125—138 for Triplet networks.

The weights in the neural network backbones were initiated from the pre-trained classifier on the same dataset. At first, training was performed with no fixed weights. However, a decline in performance was observed during the initial epochs, prompting us to consider weight fixation. To make this choice, it was drawn from common practices in transfer learning tasks, where researchers often fix weights in shallower layers while training deeper ones (such as the papers [131] and [132]).

The test set (#3 in Table 3) did not include out-of-distribution samples due to difficulties in collecting them. It is recognized that including out-of-distribution samples might be helpful in further research.

For every test image, the distance was measured from every class centre. That gave  $M \times N$  measurements ( $M$  - dataset size;  $N$  - number of classes), of which  $M$  samples were positive ("Correct" selections) and  $M \times (N - 1)$  were negative ("Incorrect" selections). The results were calculated by giving weight  $(N - 1)$  to the positive measurements so that

the "Correct" and the "Incorrect" classes were balanced.

During evaluation, the distance between each data point and every class centre was computed. A concept of maximum-distance-from-centre threshold (dashed line in Figure 25b) was established. Data points were classified as positive (inside the circles) or negative (outside the circles). Correct predictions involve data points inside the same-class circle and outside the circle of another class. By incrementally adjusting the threshold, Receiver Operating Characteristics (ROC) were derived.

Throughout the experiments, the primary performance metric employed was the Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC). Furthermore, the verification Equal Error Rate (EER) is provided, which corresponds to the point on the ROC curve where the False Positive Rate (FPR) matches the False Negative Rate (FNR).

## 4.4 Product Grouping by Similarity

In order to assign products to groups, so that groups become separable with the optimum accuracy, the following questions were sought to be answered: 1) Given the non-trivial nature of image class similarity, to discover in what similarity-way image classes need to be grouped so that the maximum group separability accuracy is achieved 2) To identify which technique yields optimum group separability. Classification into known groups (by similarity) of classes can be done by (a) training group-classifiers or (b) training individual-product-classifiers and assigning Top-1 product's group.

### 4.4.1 Motivation for Product Grouping by Similarity

When the classification of individual classes is unpractical, one must find a way to find "class clusters" - groups of classes such that classes within clusters are more similar than classes in different clusters. Resulting class clusters become labels for classifying images into these clusters. The class assignment to clusters must be done in a way to maximize the accuracy of image classification into these "class clusters".

The goal of this research is to investigate the "class clustering" mechanism such that metrics of classification into these "class clusters" are maximized. This research investigates two aspects of classifying into "class clusters": first, determining class similarity; second, whether it is beneficial to train "cluster" classifiers.

Determining class similarity drives the order of class assignment into clusters. As opposed to classic clustering tasks that produce "data clusters", grouping classes into clusters cannot be achieved by using such



clustering techniques as K-means or similar. Once classes are grouped into clusters, the quality of data classification into these "class clusters" is measured.

The second aspect of this research is to investigate the usefulness of training cluster classifiers - classifiers that predict class clusters instead of individual classes. Cluster classifiers are trained on data labelled with cluster IDs rather than individual class IDs. The performance of cluster classifiers is compared with the performance of individual class classifiers.

Drawing an upper boundary on the number of classes, merged into a cluster, depends on the application. For example, a self-checkout picklist assistant may display 3—10 items for a customer to choose from. This research refrains from optimizing the number of clusters.

The findings of this research apply to domains where 1) Top-1 image classification accuracy does not meet minimum requirements and 2) predicting a few similar classes is useful. One example is a retail self-checkout picklist assistant for barcodeless products. Another example is species classification of entire animal or plant kingdoms in random environments, such as Google Lens.

The following terms are used throughout the section:

- Class cluster - a group of individual classes (1 or more) that are deemed similar by one of the similarity techniques; a target category in a cluster classifier; each individual class is assigned to one and only one class cluster.
- Group of classes - used as a synonym for "class cluster".
- Individual Class - a target category in a classifier, representing a single product offered by a retailer. Examples: plum tomatoes, big oranges, persimmons, bananas. Used interchangeably with terms "Class", and "Product".
- Metrics hypothetically merged - given an individual class classifier and similar class clusters, evaluate the accuracy and f-score as if classification errors between classes in the same cluster were correct predictions.
- Metrics actually merged - given class clusters, and dataset labelled with individual class ID, change labels to cluster ID and train a classifier. Evaluate the accuracy and f-score of the classifier-into-clusters.
- Taxonomic hierarchy - a picklist menu tree in retailers' self-checkout menu for the selection of barcodeless products. A menu tree usually represents a biological hierarchy. E.g. the

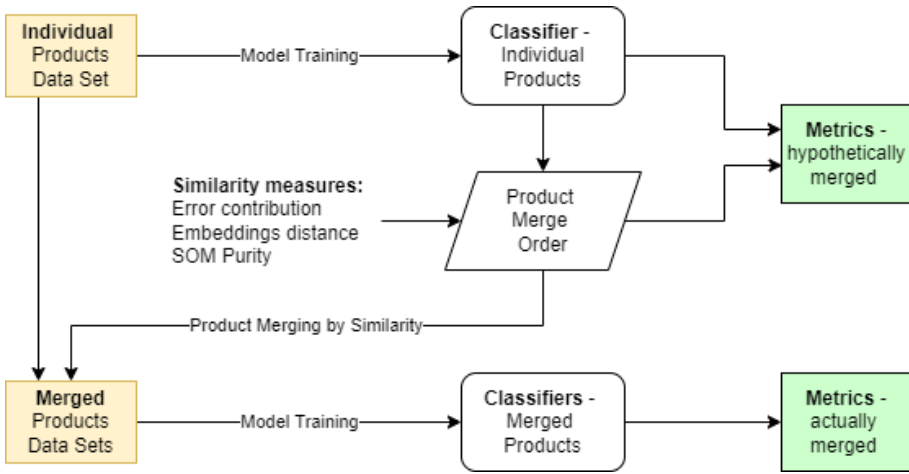


Figure 28: Comparing Actual vs. Hypothetical metrics of merged similar classes

individual product category "plum tomatoes" is a leaf of the category "all tomatoes", which is a branch of a higher-level category "Vegetables".

- Taxonomic Proximity - number of steps required to traverse between 2 products in the Taxonomic hierarchy graph.

The step *Classifier - Individual Products* in Figure 28 is based on the classifier of individual products resulting from classification subsection 4.2.

The inter-class similarity is determined using three different methods (details below in this chapter): Error-Contribution, Embeddings-Distance, and SOM-Purity. Based on class similarity using each of the 3 methods, an agglomerative clustering scheme is produced - i.e. the order of how classes are merged into clusters by similarity (Figure 28, *Product Merge Order*).

Using datasets labelled with cluster IDs, classifiers are trained (Figure 28, *Classifiers - Merged Products*) that predict the cluster of classes that the image belongs to. The architecture of all the classifiers is identical except for the last Softmax layer (which differs in the number of neurons only). The cluster classifier architecture is also identical to the individual product classifier.

*Metrics - hypothetically merged.* The individual product classifier's Top-1 predictions are the basis for the individual classifier's confusion matrix (Figure 29, Left). Given class clusters (In Figure 29, classes I and J belong to cluster X), confusion matrix of classification into clusters (Figure 29, Right), row/column X is produced by summing rows/columns that represent classes [I, J]. Essentially, this treats the

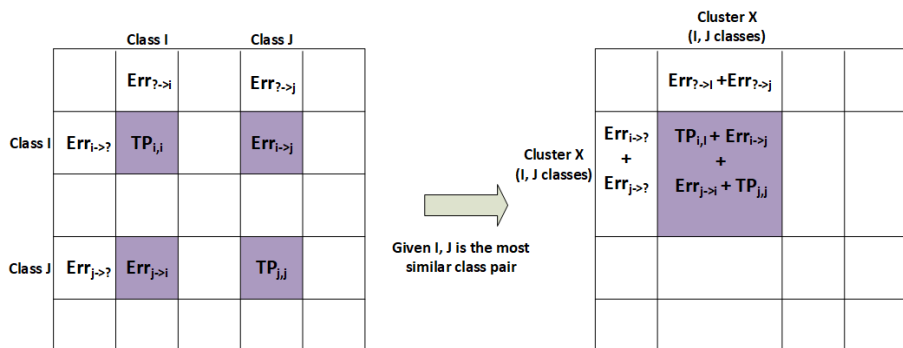


Figure 29: Calculating confusion matrix based on hypothetical class merge

Top-1 predictions of either class [I, J] of the individual classifier as correct if the true class is either of classes [I, J], and incorrect otherwise. The confusion matrix of classification into clusters (Figure 29, Right) is the basis for other metrics of hypothetically merged classes: accuracy, f-score.

*Metrics - actually merged.* Given class clusters, images of classes that belong to the same cluster, are merged into the same folder before training (essentially labelling the dataset with cluster IDs instead of individual class IDs). The cluster classifier is trained on the dataset labelled with cluster ID. Confusion matrix of a cluster classifier in the basis for other metrics of actually merged classes: accuracy, f-score.

#### 4.4.2 Approaches for Deciding Product Similarity

Class merge order is determined by class similarity: more similar classes are merged earlier. However, similarity between classes is subjective: measuring class similarity using different techniques yields different results. Similarity between classes was measured in three distinct ways, later used to determine class merge order:

- Biggest error contributors;
- Embeddings distance;
- SOM purity improvement.

*Biggest error contributors.* A higher number of errors between two classes in a classifier's confusion matrix implies these classes are more similar. The validation set's confusion matrix was used to determine inter-class similarity. Figure 30 shows representative samples of the Top-3 most similar class pairs using the "biggest error contributors" method.

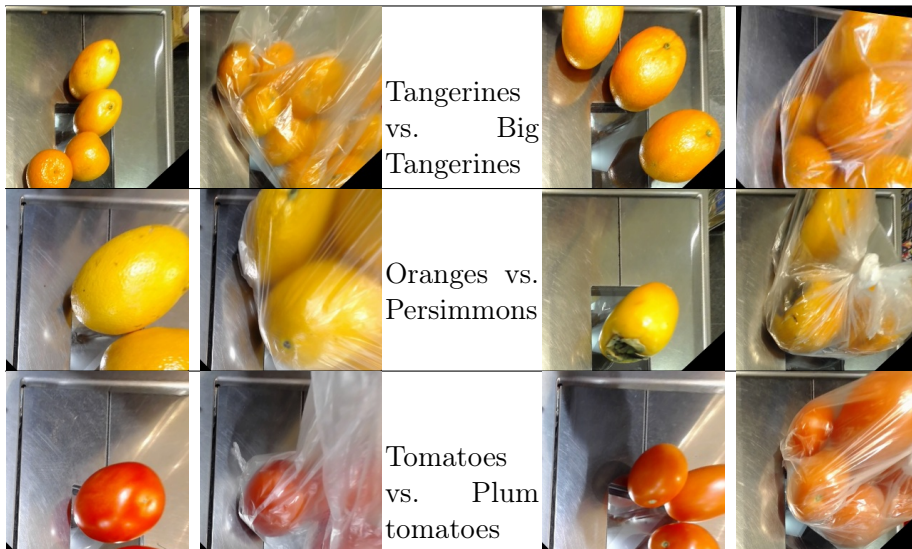


Figure 30: Representative samples of the biggest error-contributing class pairs in a confusion matrix

These class pairs indeed seem similar to the human eye: different-sized tangerines, oranges and persimmons, and different sorts of tomatoes.

*Embeddings distance.* Measuring image similarity by direct pixel comparison is impractical due to the high dimensionality of image data. In addition, even slight variation in an object’s position yields a high difference between pixels in two images. Instead of direct pixel comparison, any higher-level, lower-dimensional features are preferable for comparing images. For this research, the higher-level features were Individual Product Classifier’s activations of the pre-last layer (embeddings) as shown in Fig 31. The high accuracy of the classifier implies that enough information about classes is carried in all the layers of the classifier network. The pre-last dense layer was chosen for comparing image similarity for several reasons. First, deeper layers in sequential networks carry higher-level feature information. Second, deeper dense layers usually have lower dimensionality, which is preferred for performance reasons. Third, the last layer (Softmax) carries individual class probabilities, which implies that using the last layer embeddings of an ideal classifier would yield equal similarity between any pair of classes that contradicts the goal of measuring inter-class similarity. Single image embeddings vector took shape:  $embeddings \in \mathbb{R}^{128}$ . Figure 31 depicts how image embeddings were extracted from the classifier for inter-class similarity measurement. Cosine distance was used to measure the similarity between image embeddings. Using

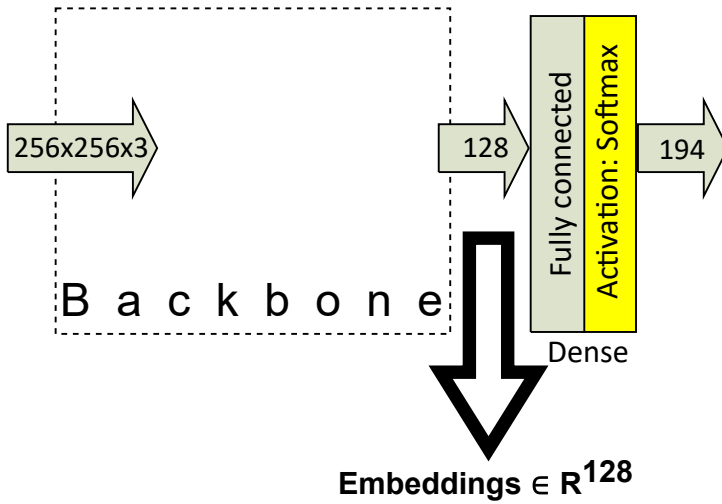


Figure 31: Embeddings for inter-class distance measurement are pre-softmax layer activations of the product classifier (Figure 24)

other distance types (Euclidean, Manhattan) showed very similar results, therefore research was limited to a single distance type (cosine) in all experiments.

Finally, mean distances between all class pairs were calculated by averaging the distance between all images in class A and all images in class B (where  $A \neq B$ ), as shown in Eq. 15. This resulted in a half-matrix of mean distances among all classes. Figure 32 shows the Top-3 class pairs that have the lowest inter-class distance when the "Embeddings-distance" technique is used for class similarity.

$$Mat\_Emb\_Dist_{i,j} = \frac{1}{M_i \times M_j} \sum_{k=1}^{M_i} \sum_{l=1}^{M_j} distCosine(E_k, E_l), \quad (15)$$

where

$N$  is the number of classes in the dataset,

$M_i$  is the number of samples in the class  $i$ ,

$Mat\_Emb\_Dist_{i,j}$  is the mean distance between all the samples in class  $i$  and class  $j$ .

*SOM purity improvement.* SOM [1] is a type of artificial neural network used for unsupervised learning and dimensionality reduction. SOMs are known for preserving the topological structure of the input space in the reduced-dimensional map. Neurons or nodes in the SOM are arranged in a grid, often in a 2D configuration. During training,



Figure 32: Representative samples of the smallest mean distance between image embeddings having class pairs

input vectors are presented to the SOM, and neurons compete to be the "winning" neuron that best represents the input. SOM spans various applications, such as visualization (in the research [133]) and clustering (in the research [134]). As opposed to other clustering techniques, SOM preserves an inter-cluster grid, where nearby cluster centroids imply more similar data points attached to them - an important factor in the investigation of class similarity. Ideal SOM trained on an image set of various classes should result in a) images of the same classes falling under the same cluster; b) images of different classes falling under different clusters, provided there are at least as many clusters as there are classes. Although training SOM on real data rarely results in such clusters, images of similar classes tend to fall under the same clusters more frequently than images of dissimilar classes. Therefore, merging two similar classes should result in bigger cluster purity (Eq. 16 [135])

improvement than merging two classes that are not similar.

$$Purity = \frac{1}{M} \sum_{i=1}^k \max_j |c_i \cap t_j|, \quad (16)$$

where

$M$  is the number of samples in the dataset,

$k$  is the number of SOM clusters,

$c_i$  is the  $i$ -th cluster,

$t_j$  is a classification which has the max count for cluster  $c_i$ .

Training SOM on an image set requires a few hyper-parameters. First, input data should be decided on: like for most clustering techniques, direct clustering of image pixels of high dimensionality ( $d = 256 \times 256 \approx 64K$ ) is irrelevant performance-wise and data-quantity-wise. Instead of image pixel clustering, the classifier embeddings from Figure 31 were chosen as input to train SOM. Second, SOM grid size should be selected. For images of each class to fall under a different cluster, at least as many clusters as there are classes must exist. The grid size was set to  $15 \times 15$ , making  $\sim 225$  clusters (a little less if hexagon grid cells are used), which exceeded the number of classes (194). Top-3 pairs' representative samples using the SOM-Purity similarity measure are shown in Figure 33.

All three similarity techniques - biggest error contributors (Figure 30), smallest mean distance between embeddings (Figure 32), most SOM purity improvement (Figure 33) - suggested that the most similar pairs appear similar to the human eye, although all three techniques suggested different class pairs in the top of the similarity list.

#### 4.4.3 Training Product Group Classifiers

Drawing the upper bound on the number of classes within a cluster usually depends on applications and is not part of this research. However, investigating the performance of classifiers trained on "class clusters" requires choosing the number of clusters. Although it is possible to train cluster classifiers on each number of merged classes in the range  $[2; N - 1]$  ( $N$  - number of classes), such a task requires an enormous amount of resources. Since the goal was to investigate in what order merged classes yield the best classification results, it makes sense to choose the number of clusters to be at the intersection points of important metrics using hypothetical classification results. The metrics measured were accuracy



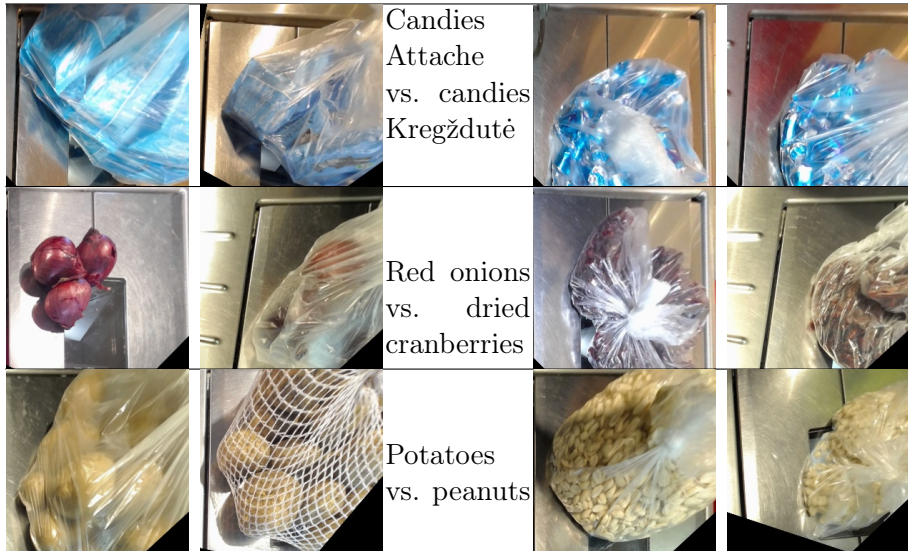


Figure 33: Representative samples of the class-pairs that most improve SOM purity if merged

and f-score. It also makes sense to choose local maxima points of f-score - meaning that a smaller number of classes from local maxima yields a lower f-score, making the classifier of the next-smaller number of classes inferior in terms of both f-score and number of classes.

The item *Individual Products Dataset* in Figure 28 is the self-checkout image dataset labelled with product ID and cleaned off of empty frames and invisible products (#3 in Table 3).

All the models were trained for 100 epochs unless validation accuracy did not improve for the last 10 epochs. Adam optimizer [116] was used with learning rate 1e-3.

Metrics are compared for classification-into-clusters of:

- A) individual products classifier (Figure 28, *Metrics - hypothetically merged*);
- B) cluster classifiers (Figure 28, *Metrics - actually merged*).

All the class grouping experiments were followed by training cluster classifiers. Both accuracy and f-score were measured (accuracy represents the real-world class disbalance in the self-checkout domain).

## 4.5 Conclusions of the Section

The image fitness for the recognizability section outlines the creation of a minimum viable neural network architecture for classifying product



recognizability on a self-checkout dataset. The architecture includes convolutional layers with 3x3 kernels, Batch Normalization, ReLU activation, and max-pooling. The number of convolutional and dense layers is determined through iterative training until performance saturates. Regularization techniques like dropout are employed to reduce validation error. A thresholding strategy is proposed for categorizing product recognizability labels, with special consideration for transparent plastic bags. The resulting classifier is versatile, serving purposes in recognizability experiments, dataset filtering, and as a backbone for other neural network tasks.

The product classification section outlines a fully automated pipeline for training a product classifier in retail environments, starting from data collection and labelling based on customer-chosen product IDs without manual review. The pipeline includes phases for rejecting images where product visibility is insufficient for recognition, such as removing empty or poorly visible products. To implement the removal of empty images, three strategies were proposed: an emptiness classifier trained on balanced data, an anomaly detector trained on empty data with non-empty treated as anomalies, and a Siamese network comparing target images to empty self-checkout scales area images. For removing poorly visible products, a recognizability classifier from a previous section was suggested, applying thresholds to separate visible from invisible labelled images. An ablation study was proposed to demonstrate the necessity of these rejection phases. Four different backbones for product classifiers were proposed: initializing with pre-trained weights from a recognizability classifier, using EfficientNet-B0, ResNet-50, and training an auto-encoder on the self-checkout dataset and using its encoder part with pre-trained weights for classification. The resulting trained product classifier was utilized for classification experiments on authentic self-checkout and Fruits-360 datasets, and to initialize weights for neural networks in verification and grouping tasks.

The product verification section delves into comparing two main approaches for tackling a class verification task. The first approach involves training a network to compare multiple samples during inference, utilizing contrastive or triplet loss functions. The second approach entails learning a class prototype during training, which is then compared against during inference. This method utilizes Centre-Loss or Proxy-NCA loss functions, aiming to minimize the distance between batch image activations and class centres while moving these centres closer to the data points of their respective classes. Proxy-NCA loss also works to maximize distances from other class centres. To prevent class centres from collapsing into a single point, networks employing Centre-Loss require additional components in the loss function. This

separation is achieved through Cross-Entropy loss or Inter-Centre-Loss summands. Different alternatives for measuring distances between samples or sample-to-centre are presented, ensuring a fair comparison by utilizing the same neural network backbone across all approaches.

The product grouping by similarity section proposes a method to classify products into similar product groups rather than individual products, aiming to maximize accuracy. Three approaches to discovering product similarity pairs are presented. The first approach is based on the number of errors in the confusion matrix between two products. The second approach relies on the average embeddings distance of a latent neural network layer. The third approach involves training a SOM network and offering pairs that increase SOM node purity the most if classes are merged. Both the second and third approaches utilize a trained product classifier's latent layer embeddings. Once the order of similar pairs is determined, the section suggests training classifiers into product groups. Experiments are outlined to compare the accuracy of classifiers-into-groups against the accuracy of individual classifiers.

## 4.6 Hardware and Software Frameworks

This section reports relevant hardware parameters and software frameworks used for training in all the experiments. As the hardware granted was different by experiment, it is reported accordingly.

Product recognizability experiments were carried out using Keras 2.2.4-tf and Tensorflow 1.15.0. All the experiments were performed on a PC with a single GPU Nvidia GeForce GTX 1070.

For product classification experiments Keras 2.4.0-tf, Tensorflow 2.4.1, CUDA 11.2 were used. Training experiments were performed on a PC with two Nvidia GeForce GTX 1080 GPUs, and 64GB RAM.

Product verification neural network training was executed on a GPU Nvidia Tesla V100-SXM2-32GB. The training duration of 1 epoch on about two million images varied between 45 and 55 minutes.

Nvidia GeForce GTX 1080 GPU and 64GB RAM were used for all the experiments of product grouping by similarity. The deep learning frameworks were Keras 2.4.1-tf, CUDA 11.2.

## 4.7 Code repositories

Code is available at:

- Recognizability study: <https://github.com/bernardas78/Visible>;
- Classification study: [https://github.com/bernardas78/IsKnown\\_Code](https://github.com/bernardas78/IsKnown_Code);

- Verification study: [https://github.com/bernardas78/Product\\_Verify](https://github.com/bernardas78/Product_Verify);
- Grouping study: [https://github.com/bernardas78/IsKnown\\_Code](https://github.com/bernardas78/IsKnown_Code).

## 5 Results

### 5.1 Results of Determining Product Recognizability

The final model resulted in a deep neural network architecture shown in Figure 34.

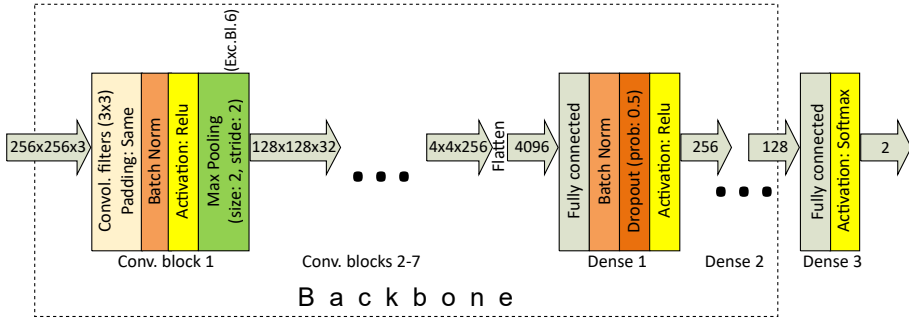


Figure 34: Final model architecture for product recognizability

Experiments with less convolutional and dense layers showed significant bias (training error); more layers of either kind did not further decrease bias. Larger convolutional filters ( $5 \times 5$ ,  $7 \times 7$ ) were experimented with and a decline in validation accuracy of  $-3.0\%$  was observed. Finally, convolutional filters were all chosen to be of size  $3 \times 3$ . Upon adding batch normalization after various layers, generally increased validation accuracy of  $-0.6\%$  -  $+2.2\%$  was observed. The final model includes batch normalization layers after each convolutional and dense layer. Experiments of adding dropout regularization after dense layers (except the last) showed significant improvement in validation accuracy of  $+2.2\%$  -  $+2.5\%$ . The final model contains a dropout layer after each dense layer

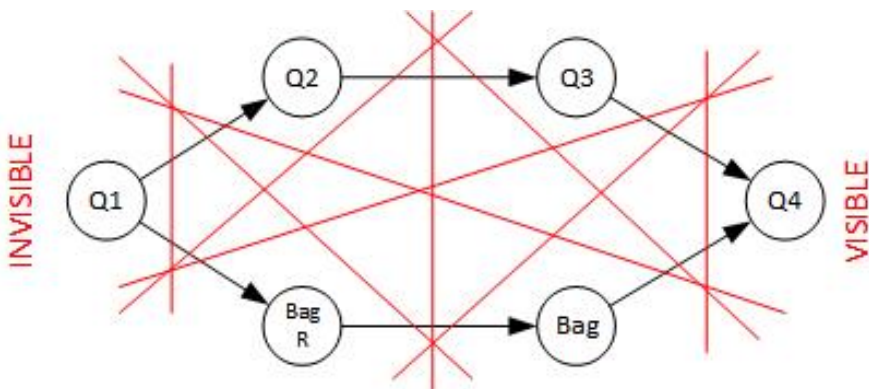


Figure 35: Product visibility directed graph. Red lines show potential thresholds separating images fit for product recognition from the rest. Product visibility increases in the direction of arrows

Table 4: Highest f-score models for each grouping

Labels in category		F-score	Accuracy	Precision	Recall
Visible	Invisible				
Q2, Q3, Q4, Bag, BagR	Q1	<b>0.906</b>	0.874	0.931	0.883
Q2, Q3, Q4, Bag	Q1, BagR	0.895	0.86	0.897	0.892
Q2, Q3, Q4	Q1, Bag, BagR	0.854	0.826	0.839	0.869
Q3, Q4, Bag, BagR	Q1, Q2	0.793	0.78	0.707	0.903
Q3, Q4, Bag	Q1, Q2, BagR	0.781	0.794	0.732	0.837
Q3, Q4	Q1, Q2, Bag, BagR	0.723	0.752	0.606	0.895
Q4, Bag, BagR	Q1, Q2, Q3	0.667	0.762	0.581	0.784
Q4, Bag	Q1, Q2, Q3, BagR	0.661	0.782	0.581	0.766
Q4	Q1, Q2, Q3, Bag, BagR	0.565	0.757	0.437	0.8

(except the last). In addition to dropout, trying L2 regularization did not help, and L2 was excluded from the final model.

The quality of models was evaluated in terms of how well they separate images into [Visible; Invisible]. The product recognizability classifier outputs a vector of six recognizability categories (visibility ordinal categories Q1-Q4 and two categories for products in bags), but which recognizability categories are fit for product recognition, remains unknown. The potential thresholds separating six categories into Visible/Invisible are shown in Figure 35. Table 4 presents the f-score for the best threshold, as well as all the other thresholds. Next to the f-score, less relevant metrics - accuracy, precision, recall, and Cross-Entropy are presented.

Figure 36 presents the confusion matrix of the best f-score achieving model (Q1 vs. the rest). Although false negatives (8.1%) exceed false positives (4.5%), unbalanced (real world) test set classes (32% Invisible) make predicting Invisible less likely.

ACTUAL	Invisible	26.3 %	4.5 %
	Visible	8.1 %	61.1 %
		Invisible	Visible
		PREDICTED	

Figure 36: Best model's (Q1 vs. the rest) confusion matrix

## 5.2 Results of Product Classification

The main result of the research in Table 5 shows test accuracy on the self-checkout dataset using the best-performing pipeline as well as alternative neural network architectures and alternative pipelines having certain phases removed.

Table 5: Accuracy of the best pipeline compared against alternative architectures and ablated phase pipeline

		Alternative architecture			Ablation study	
Test accuracy	Proposed pipeline	EfficientNet B0	Resnet-50	Auto-encoder	Empty Removed	Invisible Removed
Mean	80.5%	80.2%	72.9%	58.1%	79.1%	78.8%
St.dev.	1.2%	2.4%	1.9%	2.5%	0.8%	2.0%

Results on other similar datasets are summarized later in this section, which is divided into the following parts:

- **Ablation Studies.** This part discusses the results of the pipeline having one of the phases removed;
- **Architecture Alternatives** shows the results using alternative state-of-the-art architectures;
- **Other Datasets** compares results of the proposed method on the most similar public dataset;
- **Pre-processing and Training** details the results of experiments preparing the dataset for classification and training.

### 5.2.1 Ablation Studies

Ablation studies on the stages of the pipeline "Remove empty images" and "Remove poorly visible images" were performed. One of the mentioned stages was removed in each study, thus leaving either empty or images having unsatisfactory product visibility in the dataset. Each ablation study consisted of 10 experiments. The results are summarized in Figure 37. The average test accuracy of 80.5% using all the pipeline

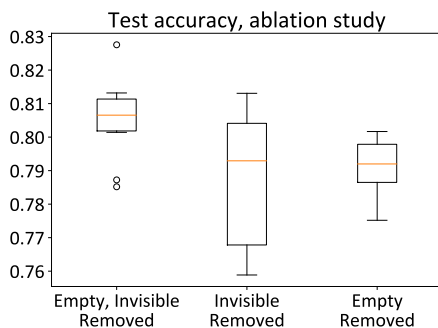


Figure 37: Ablation study showed the usefulness of the two stages in the pipeline that remove empty images and images having unsatisfactory product visibility

phases dropped down to 79.1% with the "Remove empty images" phase eliminated, and down to 78.8% with the "Remove poorly visible products" phase eliminated. Ablation studies have shown that both phases are necessary for the pipeline.

## 5.2.2 Architecture Alternatives

Individual product classifier results showed dependence on model architecture by a high margin. The results of 10 experiments of each architecture on the final dataset are summarized in Figure 38. All auto-encoder-based models performed below average. The self-made architecture in recognizing product visibility (Figure 34) consistently outperformed not only auto-encoder-based, but also ResNet-50-based models by a margin, and showed very little variance. EfficientNet B0 showed very similar performance as the self-made architecture but took

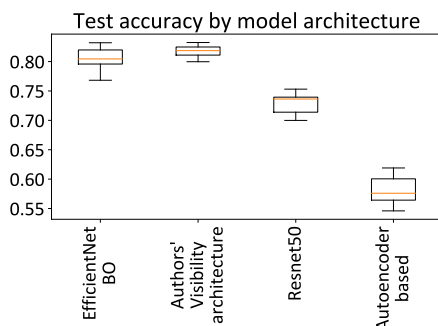


Figure 38: Model architecture's impact on the accuracy of individual product classifier

~3 times longer to train (230 min vs. 70 min per epoch). The self-made architecture yielded 80.5%+/-1.2% accuracy over 10 experiments. The full confusion matrix is available. <sup>1</sup>

### 5.2.3 Other Datasets

Classification results on the most similar public dataset the Fruits 360 [7] are compared against other authors in Table 6. The proposed pipeline by this research showed better or the same test accuracy as other authors. This suggests the generalization of the method of this research on other comparable datasets.

Table 6: Comparing results on Fruits 360 dataset

Method	Test accuracy, %
[17]	95.7
[16]	98.7
[15]	99.6
<b>Proposed method</b>	<b>99.6</b>

The classification results using the method of this research on authentic self-checkout vs. Fruits 360 datasets are compared in Table 7. The notable difference in test accuracy between the datasets suggests the future direction of work is further improving the authentic self-checkout dataset.

Table 7: Accuracy: authentic self-checkout vs. Fruits 360 dataset. \*10 experiments were performed on the authentic self-checkout dataset

Dataset name	Test accuracy, %
Authentic self-checkout	80.0 - 83.4*
Fruits 360	99.6

### 5.2.4 Filtering, Pre-processing and Training

This paragraph reports the results of the filtering empty images task. Experiments on the three Emptiness classifiers described in the Methods section showed that *Balanced* classifier (98.8% accuracy separating empty from the rest) outperformed *Siamese* and *Overfit* by a high margin. Results in Figure 39 show individual product classifier accuracy on datasets prepared by filtering out empty images using 3 emptiness classifiers described above (*Balanced*, *Siamese*, *Overfit*). The *Balanced*

<sup>1</sup>[https://github.com/bernardas78/IsKnown\\_Code/blob/main/Dataset\\_descr/conf\\_mat\\_div10\\_80dpi.png](https://github.com/bernardas78/IsKnown_Code/blob/main/Dataset_descr/conf_mat_div10_80dpi.png)



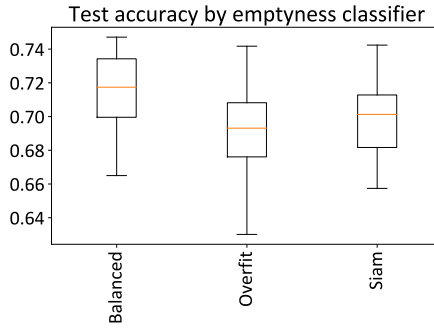


Figure 39: Empty images elimination technique’s impact on individual product classification accuracy

technique produced the best dataset, whereas *Siamese* showed only a slightly better result than *Overfit*. A total of 60 experiments (20 per technique) were performed. The final pipeline used the *Balanced* dataset.

The results of the task of selecting images with visible products in Figure 40 show the effect on the accuracy of removing images containing plastic bags with high glare. Both datasets have Q1 (having  $<1/4$  product area visible) images removed. A total of 10 experiments per dataset were performed. Removing bags with high glare only marginally improved accuracy on the test set from a mean of 70.0% to a mean of 70.2%.

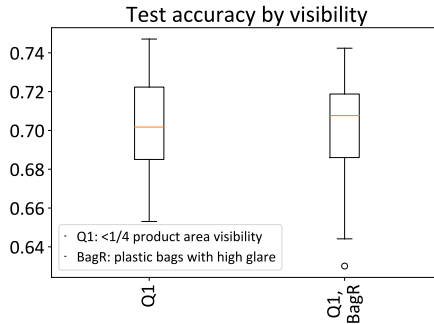


Figure 40: Effect on accuracy by eliminating various groups of images with poor product visibility

The impact of augmentation options on product classification accuracy is reported in this paragraph. All datasets contained balanced by oversampling classes using a) Basic affine transformations and b) Images with corners distorted in the range of  $\pm 20$ px. Figure 41 shows the improvement in accuracy by additionally distorting image corners in

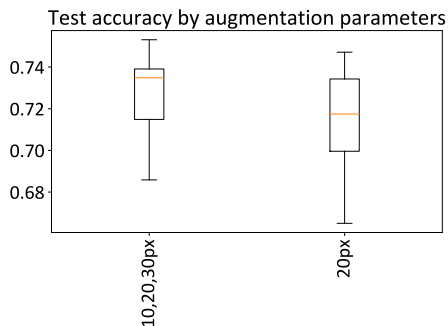


Figure 41: Additional augmentation impact on accuracy. Experiments were performed using only the Balanced emptiness classifier

the ranges of  $\pm 10\text{px}$ , and  $30\text{px}$ .

Experiments using only basic affine transformations (no corner distortion) showed worse accuracy by 1.3—2.8%. Experiments using only corner distortion (no basic affine transformations) showed worse accuracy by 1.3—3.9%. A total of 20 experiments using ( $\pm 20\text{px}$ ) and 10 experiments using ( $\pm 10\text{px}$ ,  $20\text{px}$ ,  $30\text{px}$ ) datasets were performed: an average improvement of 1.1% in test accuracy was observed by additionally distorting corners. The best-performing dataset was chosen for the final pipeline: concatenated affine transformations + corner distortions in ranges ( $\pm 10\text{px}$ ,  $20\text{px}$ ,  $30\text{px}$ ).

Figure 42 shows learning curves for training models on the authentic self-checkout and Fruits 360 datasets. Both loss and accuracy saturated in the first 2—3 epochs on both datasets. The model trained on Fruits 360 generalized well on the validation set (over 99% accuracy), and the model trained on the authentic self-checkout dataset reached  $\sim 50\%$  validation accuracy. The notable difference in the authentic self-checkout dataset between the validation set and the test set is that the validation set is balanced and the test set preserves the real-world class distribution.

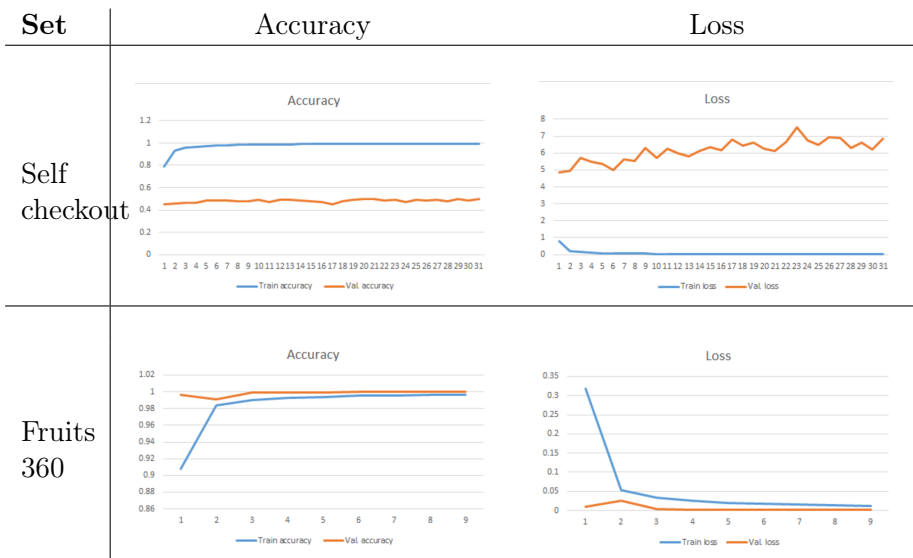


Figure 42: Learning curves: authentic self-checkout and Fruits 360 datasets

### 5.3 Results of Verifying Product Selection

The primary outcome of the selected product verification study revolves around the performance evaluation of two distinct class verification approaches: sample-to-sample and class prototype-based methods. To conduct this assessment, three different neural network models were employed in the context of barcodeless product verification using a self-checkout dataset: Centre-Loss, Proxy-NCA (both representing the class prototype-based approach), Siamese, and Triplet (both representing the sample-to-sample methods). Figure 43 and Table 8 showcase the

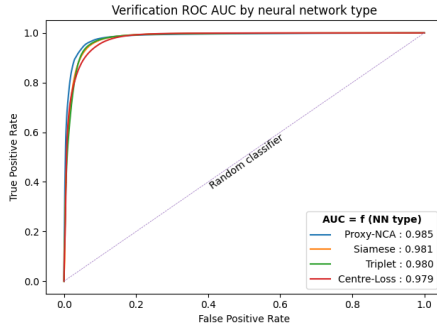


Figure 43: Verification ROC by type of neural network

Receiver Operating Characteristics (ROC) curve and provide detailed information regarding the verification Equal Error Rate (EER) and the ROC Area Under Curve (AUC) for each neural network type. The reported results utilize the optimal hyperparameter values for each network type, including distance type (for all network types), the number of trainable layers (Siamese and Triplet), pre-Centre-Loss layer index, neuron count in the pre-Centre-Loss layer,  $\lambda_1$ , and  $\lambda_2$  values (all four for Centre-Loss). Remarkably, the differences in performance among all the network types are marginal. This suggests that sample-to-sample comparing networks (Siamese and Triplet) do not exhibit superior performance compared to a class prototype-based network (Centre-Loss).

Table 8: Verification ROC AUC and EER by type of neural network

Neural Network	ROC AUC	EER
Proxy-NCA	0.985	0.054
Siamese	0.981	0.063
Triplet	0.980	0.060
Centre-Loss	0.979	0.073

The study investigated the impact of various distance types on accuracy metrics, which measure the dissimilarity between samples (Siamese, Triplet) or between a sample and class centre (Centre-Loss). Experiments encompassed Manhattan, Euclidean, Minkowski ( $p=3, 4$ ), and Cosine distance types. Tables 9 and 10 summarize ROC AUC and equal error rate (EER) by distance type, respectively. Detailed ROC curves for each distance type are in Figure 44a (Proxy-NCA), Figure 44b (Centre-Loss), Figure 44c (Siamese), and Figure 44d (Triplet). For Siamese and Triplet networks, all distance types exhibited similar performance, except for a slight degradation observed with the Cosine distance type in the Triplet network. In Centre-Loss, most distance types (Cosine, Manhattan, Euclidean, and Minkowski) performed similarly, although degradation was noted for higher Minkowski  $p$  values ( $p=3, 4$ ).

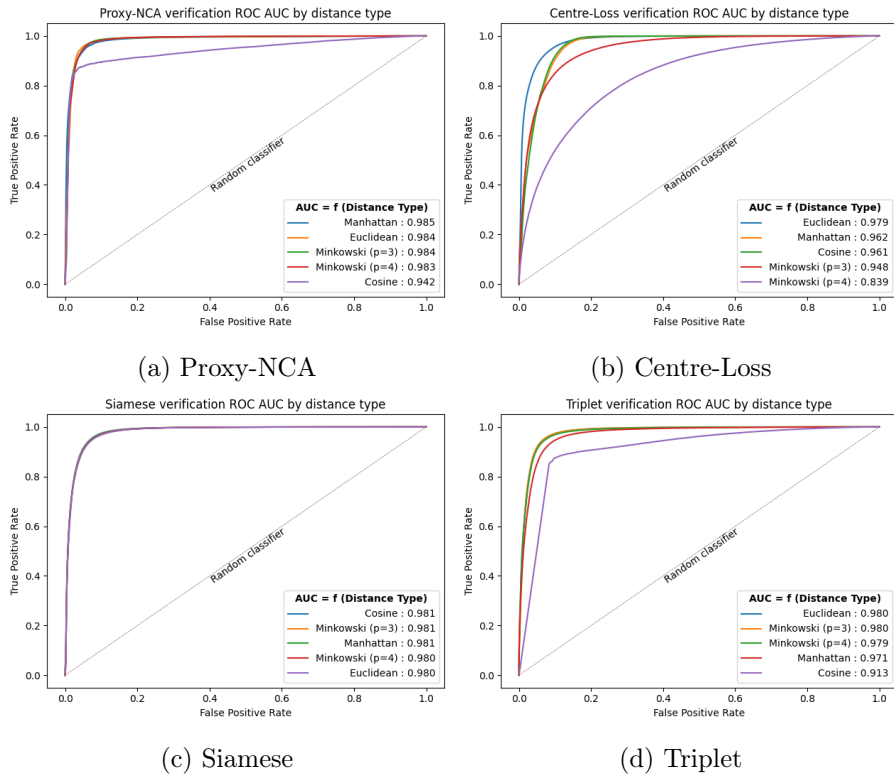


Figure 44: Verification ROC by distance-between-embeddings type

Table 9: Verification ROC AUC by distance-between-embeddings type

Distance Type	Network Type			
	Proxy-NCA	Centre-Loss	Siam.	Triplet
Manhattan	<b>0.985</b>	0.962	<b>0.981</b>	0.971
Euclidean	0.984	<b>0.979</b>	0.980	<b>0.980</b>
Minkowski ( $p=3$ )	0.984	0.948	<b>0.981</b>	<b>0.980</b>
Minkowski ( $p=4$ )	0.983	0.839	0.980	0.979
Cosine	0.942	0.961	<b>0.981</b>	0.913

Table 10: Verification Equal Error Rate by distance-between-embeddings type

Distance Type	Network Type			
	Proxy-NCA	Centre-Loss	Siam.	Triplet
Manhattan	0.054	0.099	0.063	0.078
Euclidean	<b>0.047</b>	<b>0.073</b>	0.065	<b>0.060</b>
Minkowski ( $p=3$ )	0.050	0.121	0.063	0.061
Minkowski ( $p=4$ )	0.051	0.241	0.064	0.064
Cosine	0.104	0.095	<b>0.062</b>	0.116

Training all layers in Siamese and Triplet architectures led to declining metrics after the first epoch. To address this, experiments were conducted by fixing the weights of the last one and the last two Dense blocks (see Figure 27) that were initialized from the original classifier. As depicted in Figures 45a (Siamese) and 45b (Triplet), the best ROC AUC was achieved when the last two Dense blocks were fixed in both Siamese and Triplet networks. Conversely, performance deteriorated when no weights were fixed or when only the last Dense block was fixed. Table 11 summarizes the verification EER and ROC AUC by the number of fixed blocks. In contrast, training all layers in the Centre-Loss architecture did not exhibit declining metrics; thus, all layers were trainable.

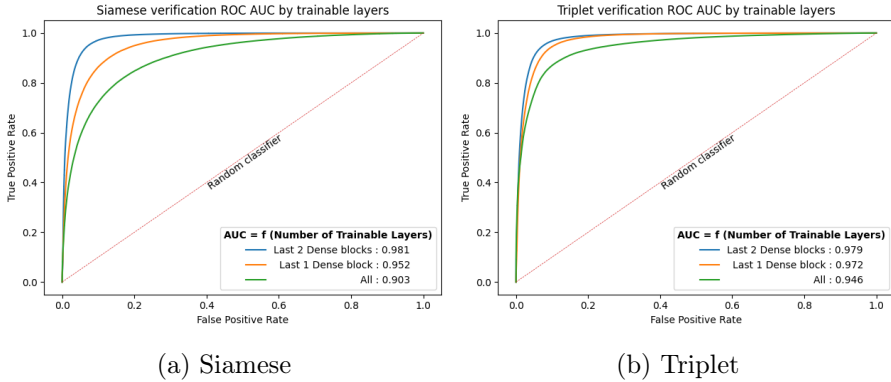


Figure 45: Verification ROC by number of trainable layers

Table 11: Verification ROC AUC and EER by number of trainable layers

Trainable Layers	ROC AUC		EER	
	Siamese	Triplet	Siamese	Triplet
Last 2 Dense blocks	<b>0.981</b>	<b>0.979</b>	<b>0.063</b>	<b>0.065</b>
Last 1 Dense block	0.952	0.972	0.116	0.079
All	0.903	0.946	0.175	0.115

To optimize the Centre-Loss architecture, a series of experiments was conducted. Initially, Centre-Loss architectures with the Centre-Loss layer attached to various layers of the original classifier were compared. Figure 46 and Table 12 provide a summary of ROC AUC and EER for ten different architectures, with the Centre-Loss layer attached from the last (-1st) to the 10th from the end (-10th) layer in the original classifier. The results highlight that the pre-last (-1st) layer is the optimal choice for Centre-Loss measurement.

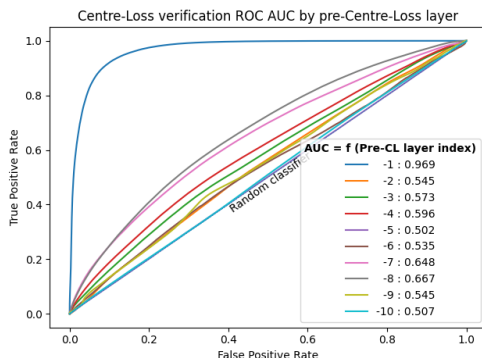


Figure 46: Centre-Loss Verification ROC by pre-Centre-Loss Layer

Table 12: Centre-Loss Verification ROC AUC and EER by pre-Centre-Loss Layer

Pre-Centre-Loss Layer	ROC AUC	EER
-1	<b>0.969</b>	<b>0.090</b>
-2	0.545	0.465
-3	0.573	0.449
-4	0.596	0.431
-5	0.502	0.499
-6	0.535	0.470
-7	0.648	0.393
-8	0.667	0.380
-9	0.545	0.467
-10	0.507	0.495



Secondly, the optimal number of neurons in the pre-Centre-Loss layer was empirically determined. Figure 47 illustrates the growth and saturation of verification ROC AUC with respect to the neuron count in the pre-Centre-Loss layer. The detailed class verification Equal Error Rates are shown in Table 13 and Receiver Operating Characteristic's Area Under Curve (ROC AUC) is shown in Table 14. ROC AUC demonstrates a positive correlation with neuron count for all distance types until it reaches saturation. The saturation point for ROC AUC occurs with a smaller number of neurons when the Minkowski coefficient  $p$  is low: Manhattan ( $p=1$ ) saturates at 256 neurons, Euclidean ( $p=2$ ) at 512-768 neurons, while higher  $p$  values ( $p=3$  and 4) do not saturate even at 2048 neurons.

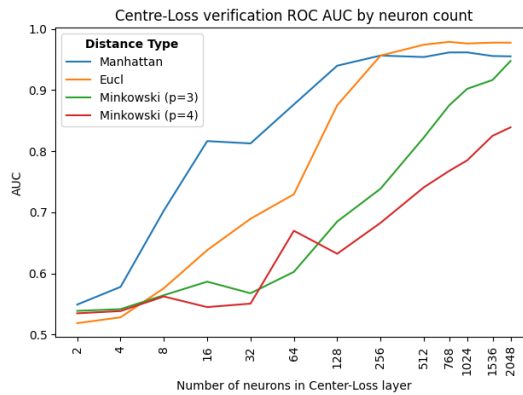


Figure 47: Centre-Loss Verification ROC AUC by Neuron Count in the Centre-Loss layer

Table 13: Accuracy at Equal Error Rate for various neuron counts in the Extra Dense layer

Neuron Count	Equal Error Rate			
	Manhattan	Euclidean	Minkowski ( $p=3$ )	Minkowski ( $p=4$ )
2048	0.110	<b>0.073</b>	0.121	0.241
1536	0.105	<b>0.073</b>	0.161	0.256
1024	0.100	0.076	0.176	0.284
768	0.099	<b>0.073</b>	0.204	0.298
512	0.115	0.076	0.252	0.322
256	0.102	0.110	0.324	0.367
128	0.134	0.207	0.364	0.405
64	0.201	0.330	0.427	0.378
32	0.262	0.365	0.453	0.469
16	0.271	0.404	0.438	0.467
8	0.356	0.445	0.455	0.456
4	0.439	0.480	0.477	0.479
2	0.470	0.489	0.464	0.478

Table 14: ROC Area Under Curve (AUC) for various neuron counts in the Extra Dense layer

Neuron Count	ROC AUC			
	Manhattan	Euclidean	Minkowski ( $p=3$ )	Minkowski ( $p=4$ )
2048	0.955	0.978	0.948	0.839
1536	0.956	0.978	0.917	0.825
1024	0.962	0.976	0.902	0.785
768	0.962	<b>0.979</b>	0.875	0.768
512	0.954	0.974	0.823	0.741
256	0.957	0.956	0.739	0.683
128	0.940	0.875	0.685	0.632
64	0.877	0.730	0.603	0.670
32	0.813	0.689	0.568	0.551
16	0.817	0.638	0.587	0.545
8	0.703	0.576	0.564	0.562
4	0.578	0.528	0.542	0.538
2	0.549	0.519	0.539	0.535

Several experiments were conducted to determine the optimal hyperparameter values for Centre-Loss: the Centre-Loss relative weight ( $\lambda_1$  in Eq.9) and the Inter-Centre-Loss relative weight ( $\lambda_2$  in Eq.9). Figure 48a displays the verification ROC, while Table 15 provides detailed ROC AUC and EER figures for various  $\lambda_1$  values. The results indicate that  $\lambda_1$  values between 0.1 and 3.0 produce similar metrics, while values outside this range lead to degraded performance. This underscores the importance of both summands in the Centre-Loss function: the Cross-Entropy summand for class separation and the Centre-Loss summand for reducing intra-class distances. The robust performance across a wide range of  $\lambda_1$  values suggests that the Centre-Loss function is not highly sensitive to variations within this range. However, the inclusion of the Inter-Centre summand in the loss function did not yield positive effects. Table 16 and Figure 48b show that the optimal value for  $\lambda_2$  is 0; increasing  $\lambda_2$  led to lower ROC AUC and higher EER values.

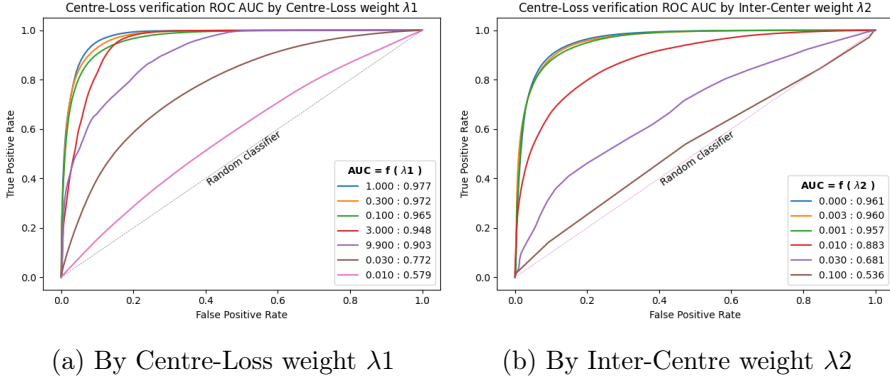


Figure 48: Centre-Loss Verification ROC by Centre-Loss hyperparameters  $\lambda_1$  and  $\lambda_2$

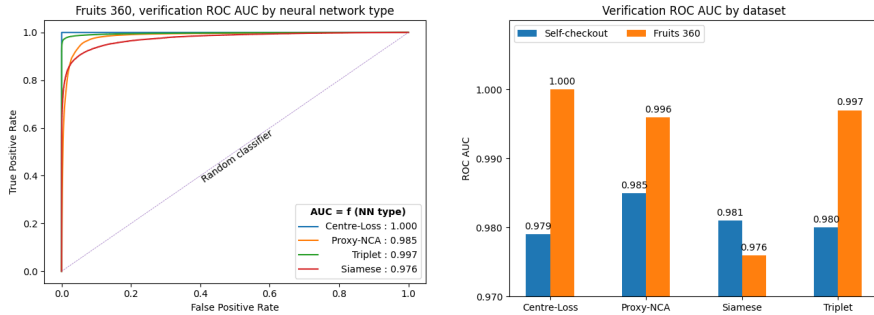
Table 15: Centre-Loss Verification by weight  $\lambda_1$

Centre-Loss Weight $\lambda_1$	ROC AUC	EER
1.000	<b>0.977</b>	<b>0.075</b>
0.300	0.972	0.086
0.100	0.965	0.098
3.000	0.948	0.117
9.900	0.903	0.195
0.030	0.772	0.298
0.010	0.579	0.444

Table 16: Centre-Loss Verification by weight  $\lambda_2$

Inter-Centre Weight $\lambda_2$	ROC AUC	EER
0.000	<b>0.961</b>	<b>0.100</b>
0.003	0.960	0.105
0.001	0.957	0.110
0.010	0.883	0.201
0.030	0.681	0.381
0.100	0.536	0.467

In addition to the authentic self-checkout dataset, the method’s performance was assessed using the Fruits 360 dataset [7]: models of each network type in question (Centre-Loss, Proxy-NCA, Siamese, and Triplet) were trained on Fruits 360 training subset and evaluated on its test subset. The verification ROC curves for Fruits 360 are illustrated in Figure 49a, and a comparison of ROC AUC between Fruits 360 and our self-checkout dataset is presented in Figure 49b. Except Siamese, most methods exhibited improved performance on Fruits 360, primarily due to the dataset’s clean and synthetic images.



(a) Fruits 360 dataset, verification ROC by type of neural network (b) Authentic Self-Checkout dataset vs. Fruits 360 [7]

Figure 49: Comparing verification results on Fruits 360 [7] vs. authentic self-checkout dataset

Figure 50 illustrates a set of sample images alongside their corresponding distances from selected class centres. Correct selections are denoted by green dashes, while incorrect ones are indicated by red dashes. The blue dashed line represents the equal error rate (EER) threshold, delineating the boundary between correct (below the threshold) and incorrect (above the threshold) selections.

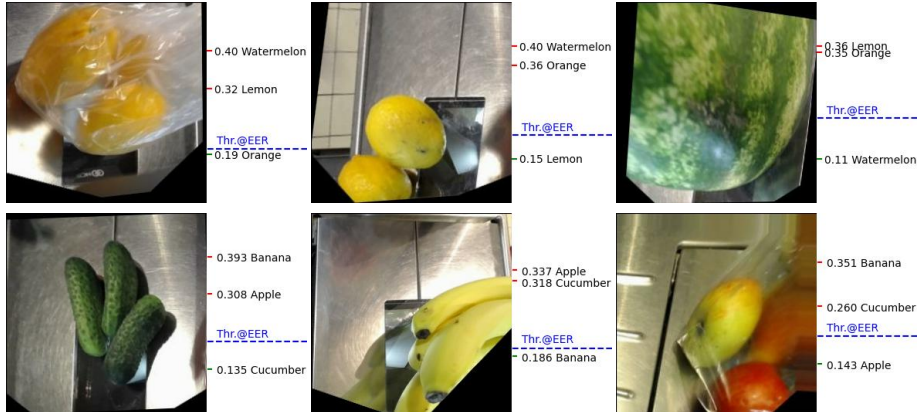


Figure 50: Sample images and their distances from selected class centres

Figure 51 depicts sample distance distributions for the selected number of neurons in the Extra Dense layer. The separation between distances from samples' own class centres ("Correct" classes) versus from other class centres ("Incorrect" classes) increases with the increase of neurons in the Extra Dense Layer until saturation is reached. The mean distance of both - Correct and Incorrect - increases with the number of neurons.

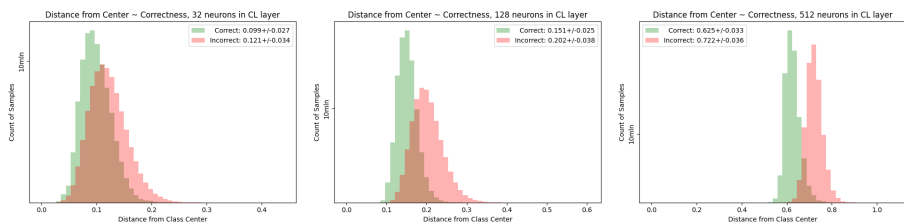


Figure 51: Distance distribution of CL layer activations from the same (Correct) vs other (Incorrect) class centres

The verification architecture (Figure 26) with Centre-Loss contains a Softmax layer, which is usable for product classification. It is interesting to evaluate the degradation of the product classifier accuracy by having the Centre-Loss function included: an additional component of Centre-

Loss in the loss function was expected to decrease the accuracy of individual class classification. The classifier-only (without Centre-Loss) performed at 73.2% accuracy on the validation set (blue line in Figure 52). However, the graph shows approximately the same individual classification accuracy when the Extra Dense layer contains at least 8 neurons:  $73.2\% \pm 0.8\%$ .

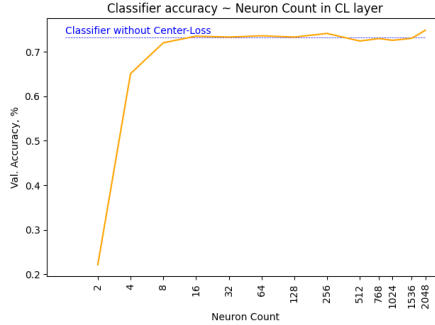


Figure 52: Classification accuracy dependency on the number of neurons in the extra dense layer

Figure 53 shows the relative importance of the Loss function summands:  $L_{CE}$  (Softmax' Cross-Entropy loss) and  $L_C$  (CL layer loss). The Softmax's Cross-Entropy loss gains minimum value at approximately 8—16 neurons in the Extra Dense layer, then stays at about the same rate upon adding more neurons. This relates to the fact that classification accuracy also saturates at about the same 8 to 16 neurons. Centre-Loss obtains its minimum values between 8 and 128 neurons, then rises steeply outside this range, mainly due to the rising dimensionality of space where distances are calculated.

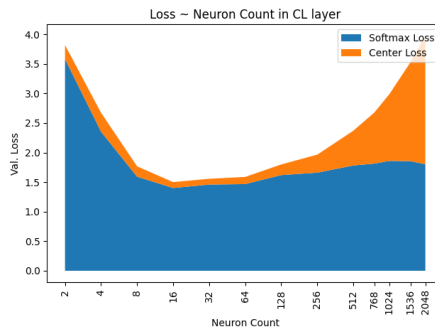


Figure 53: Loss (Cross-Entropy and Centre) relative values and dependency on the number of neurons in the extra dense layer

## 5.4 Product Classification into Groups of Similarity

The main result of this research is comparing the following:

- Different techniques to determine visual class similarity. Based on similarity, classes are merged into "class groups" and classification metrics are measured;
- Classifiers trained on individual classes vs. classifiers trained on class groups, where classes are merged into groups by similarity.

Table 17: Top-5 class pairs having most errors in a confusion matrix

Class A	Class B	Errors %
Tangerines	Big tangerines	6.5%
Oranges	Persimmons	5.1%
Tomatoes	Plum tomatoes	3.0%
Bananas	Apples Golden	2.6%
Bananas	Lemons	2.5%

The class pairs contributing the most errors are depicted in Table 17. The column "Errors %" represents false predictions both ways (class A instead of class B and vice versa) as a percentage of total errors in a confusion matrix. Almost 20% of all errors are caused by 5 class pairs - a significant percentage in a confusion matrix of size 194x194 (194 - class count in self-checkout dataset).

Table 18: Top-5 class pairs having the lowest inter-class mean distance between image embeddings

Class A	Class B	Normalized Similarity
Candies Minky	Candies Crazy Bee	8.3
Bean sprouts	Green grapes	6.8
Candies Vkusia	Candies Lokys	6.7
Candies Murkiny	Candies Verze	5.9
Candies Murkiny	Candies Kregždutė	5.7

Embeddings-Distance similarity technique is summarized in Table 18: it shows the Top-5 class-pairs that have the lowest mean inter-sample distance (i.e. highest inter-class similarity, where  $similarity = \frac{1}{distance}$ ).

The SOM-Purity improvement technique suggested the class pairs that improve cluster purity the most when merged. The original (before merging) SOM cluster structure is shown in Figure 54. Circle sizes are proportional to the number of samples in those clusters. The biggest

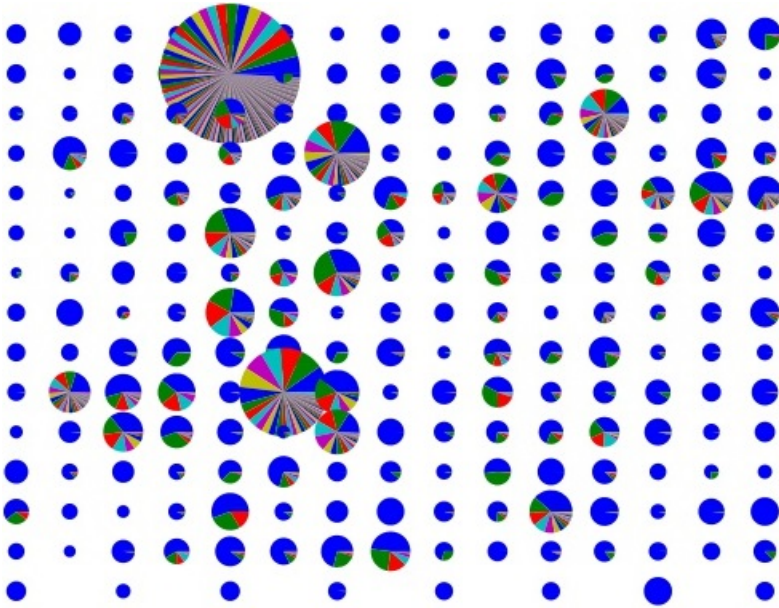


Figure 54: SOM cluster structure. The circle size represents a number of samples. Colours indicate different classes within clusters, blue being the most frequent class

class of every cluster is shown in blue, whereas every colour within a cluster represents a different class. Colour angle is proportional to the class number of samples in a cluster. Although most clusters contain mostly images of a single class, the biggest clusters appear to consist of a multitude of classes.

The number of group classifiers was limited as described in the Methods section. Figure 55 depicts the choice for the number of groups using accuracy intersection point (top-left), f-score intersection points (top-right) and f-score local maxima (bottom).

Table 19 compares the accuracy of classification into class groups. The best accuracy is achieved using the SOM-Purity merging technique for a smaller number of "class groups", but the Embeddings-Distance merging technique outperforms the rest when the number of classes is high ( $>160$ ). In all cases except one, individual class classifiers outperformed "class group" classifiers.

Table 20 compares the f-score of the "class group" classification. In all cases except one, the Error-Contribution merging technique outperformed the rest. Individual classifiers always performed better than "class group" classifiers.



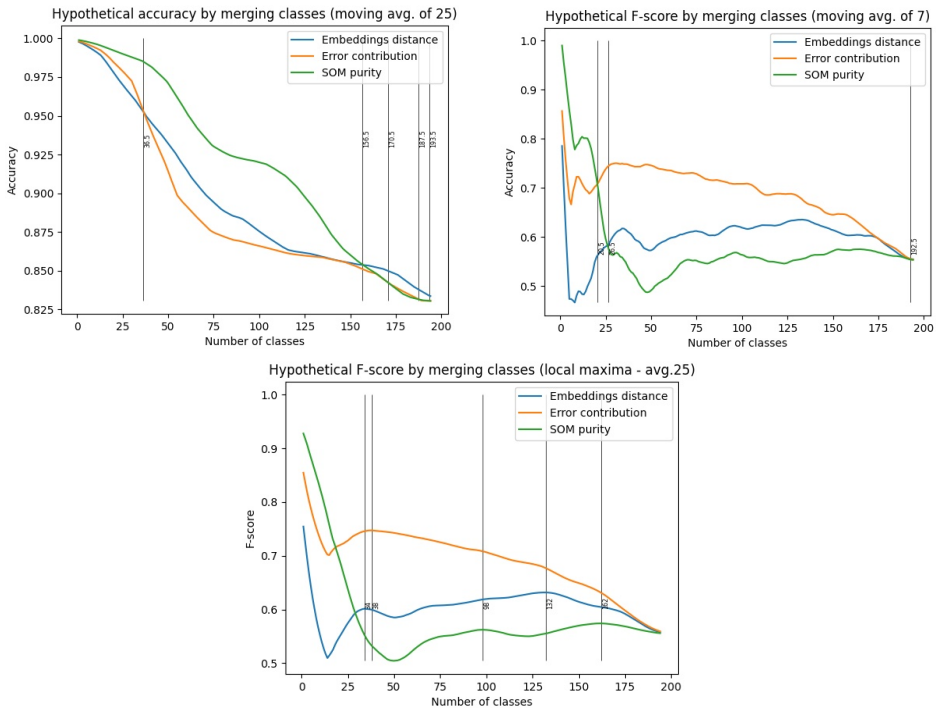


Figure 55: Choosing the number of merged classes in 3 ways: the intersection of accuracy, f-score, and local f-score maxima

Table 19: Accuracy using various methods for merging classes into groups

Num. classes	Hypothetical merge			Actual merge		
	Embeddings distance	Error contribution	SOM purity	Embeddings distance	Error contribution	SOM purity
20	0.982	0.987	<b>0.992</b>	0.965	0.983	0.981
26	0.965	0.976	<b>0.990</b>	0.929	0.954	0.977
36	0.952	0.964	<b>0.985</b>	0.900	0.953	0.957
38	0.952	0.962	<b>0.984</b>	0.937	0.908	0.974
132	0.860	0.858	<b>0.890</b>	0.844	0.850	0.863
156	0.854	0.853	0.855	<b>0.862</b>	0.839	0.847
162	<b>0.854</b>	0.848	<b>0.854</b>	0.843	0.806	0.825
170	<b>0.852</b>	0.845	0.843	0.850	0.822	0.828
187	<b>0.842</b>	0.831	0.831	0.837	0.831	0.814

Table 20: F-score using various methods for merging classes into groups

Num. classes	Hypothetical merge			Actual merge		
	Embeddings distance	Error contribution	SOM purity	Embeddings distance	Error contribution	SOM purity
20	0.557	0.702	<b>0.731</b>	0.373	0.491	0.383
26	0.584	<b>0.748</b>	0.562	0.421	0.597	0.318
36	0.618	<b>0.749</b>	0.534	0.435	0.633	0.335
38	0.610	<b>0.744</b>	0.528	0.516	0.534	0.371
132	0.635	<b>0.681</b>	0.552	0.597	0.659	0.450
156	0.607	<b>0.649</b>	0.573	0.606	0.630	0.528
162	0.606	<b>0.634</b>	0.575	0.578	0.595	0.545
170	0.601	<b>0.610</b>	0.575	0.566	0.587	0.516
187	0.563	<b>0.569</b>	0.562	0.525	0.553	0.528

Figure 56 shows the margin between various methods of merging classes into groups (different colours); between individual classifiers (lines) and "class group" classifiers (dots); accuracy (left) and f-score (right). Merging classes using the SOM-Purity technique generally outperforms other methods when accuracy optimization is key (e.g. in retail self-checkouts, where class distribution is uneven). Merging classes using the "Error contribution" technique generally outperforms other methods when f-score optimization is key. Individual classifiers outperform "class group" classifiers with few exceptions (lines above same-coloured dots).

In addition to the three techniques of merging classes described above (Error-Contribution, Embeddings-Distance, SOM-Purity), results are presented of the individual classifier using "barcode structure" class merging (red line in Figure 56). This merging technique underperforms all other proposed merging techniques. This suggests that higher-level category predictors in retail stores would not work as well as predictors of similarity-based class groups.

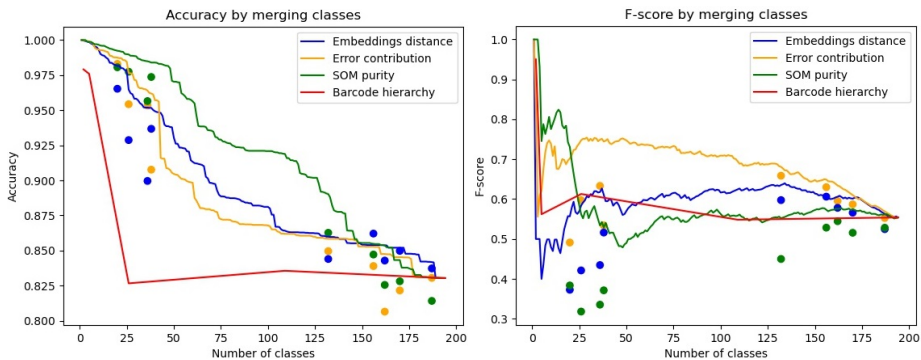


Figure 56: Actual vs. Hypothetical metrics by merging similar classes (dots represent metrics of actually merged datasets)

The inference of trained classifiers was tested on a self-checkout-like computer with 2GB RAM, no GPU, and a 2GHz CPU. Inference of an image took up to 100 milliseconds, having the classifier model pre-loaded into RAM. A single classifier model takes less than 50MB of disk space and requires to be placed on an inference computer. These requirements are acceptable for most self-checkout computers. Class similarity analysis and choosing the upper limit for the number of classes, followed by creating groups of similar classes, should be done outside of store computers, similar to our described infrastructure.

## 5.5 Conclusions of the Section

The experimental investigation was carried out on two datasets that resemble the self-checkout environment the most: an authentic self-checkout products dataset that was collected and labelled automatically in the self-checkout environment and Fruits-360. The task of determining product recognizability only applied to the former dataset, whereas the latter dataset only contained the well-visible, recognizable products.

**Image fitness for recognizability.** The recognizability study measured how well images with well-visible products can be separated from insufficiently visible products, where thresholds of visibility sufficiency are several. The best separation f-score of 0.906 was achieved between the least visible salient objects (Q1: less than 25% visibility) and the rest. Considering the authentic self-checkout dataset contains only 1/3 of Q1 images, it is important to note the model’s accuracy of 0.874, FPR of 0.146, and FNR of 0.117. The second-best f-score of 0.895 separates [Q1; BagR (products in bags with high light Reflection) ] from the rest, the value of the f-score still being in the 4th quartile. Both thresholds are worth exploring in filtering images for the product classification study. The other thresholds separate images with an f-score of 0.854 or less - significantly below the best two thresholds.

The architecture of choice consists of seven convolutional blocks followed by three fully connected blocks. Notably, adding more blocks of either type did not improve accuracy, whereas removing blocks of either type significantly degraded accuracy.

**Automated classification pipeline.** This stage investigated image filtering of empty images and images with subpar product visibility, and classification model architecture alternatives.

The optimal proposed pipeline contains stages for empty image removal and removal of images with insufficient product visibility and uses architecture tuned to the authentic self-checkout dataset. It achieved  $80.5\% \pm 1.2\%$  classification accuracy over 10 experiments.

The ablation study showed the usefulness of the two stages in the pipeline that remove empty images and images having unsatisfactory product visibility. Excluding the stage that removes empty images decreases accuracy by 1.4%, whereas not filtering out images using a recognizability classifier reduces the accuracy by 1.7%.

The necessary step of empty image filtering in dataset preparation was best achieved by using a two balanced class classifier: empty images were eliminated with a commanding 98.8% confidence. Filtering empty images using other emptiness classifiers (Overit and Siamese) resulted in worse product classification accuracy.

Comparable classification results were shown by eliminating images

with low product visibility using different visibility thresholds: eliminating images that contained bags with high glare marginally improved accuracy by 0.2%.

Neural network architecture tuned to self-checkout images worked better (80.5%+/-1.2% accuracy) than well-known models (EfficientNet-B0: 80.2% and Resnet-50: 72.9%) with pre-trained weights. Auto-encoder-based classifiers underperformed in comparison with all.

The proposed architecture was evaluated on another retail image set of barcodeless products Fruits-360, and then performance was compared against other authors'. The proposed architecture yields a comparable (better or similar) test accuracy of 99.6% to other authors'. This implies a good choice of architecture and a future research direction in how images are collected, filtered, and prepared within a fully automated pipeline.

**Verification of product selection.** Two distinct approaches to solving a class verification task on self-checkout product datasets were compared: class prototype-based and sample-to-sample. The optimal approach (Centre-Loss) learns class centres during training and then compares a sample to the declared class centre during inference. This is in contrast with the sample-to-sample approaches that compare an incoming sample to arbitrary samples during inference and require aggregation. The optimal approach uses a dual loss function that consists of cross-entropy loss and Centre-Loss functions. The Centre-Loss function uses Euclidean distance. The Centre-Loss layer is connected to the penultimate layer, its size of 256 neurons suffices. The relative weight of Centre-Loss within the total loss is 1.0.

The verification metrics between class prototype-based (Proxy-NCA, Cente-Loss) and sample-to-sample (Siamese, Triplet) approaches were almost identical on the authentic retail self-checkout barcodeless products dataset: the Proxy-NCA ROC AUC was 0.985, Centre-Loss: 0.979, Siamese: 0.981, Triplet: 0.980.

It has been demonstrated experimentally that using Euclidean distance in loss functions to measure sample-to-sample or class-centre-to-sample distances always results in equal or better accuracy over other distance types (Manhattan, Minkowski, Cosine). Although, using nearby Minkowski  $p$  values ( $p=1$  Manhattan,  $p=3$ ) performs similarly. However, higher Minkowski  $p$  values require more neurons in the penultimate layer to achieve saturation. In the Centre-Loss approach, using Euclidean distance achieved 0.979 ROC AUC, whereas nearby Minkowski  $p$  values resulted in similar ROC AUC of 0.962 ( $p=1$  Manhattan) and 0.948 ( $p=3$ ), and Cosine ROC AUC was very close at 0.961. The Siamese network showed almost identical ROC AUC values of 0.980-0.981 using all distance types (Minkowski  $p$  between 1 and 4, Cosine), whereas

in the Triplet network, Cosine (ROC AUC: 0.913) underperformed all Minkowski values (0.971-0.980).

Experiments with Centre-Loss architecture revealed the penultimate layer as the layer of choice to minimize intra-class distances upon training (ROC AUC 0.969), while connecting the Centre-Loss layer to any other layer past the convolutional backbone resulted in low ROC AUC values of 0.507-0.667. The EER also increases using any other layer (0.380-0.499) in comparison to the penultimate layer (0.090). A minimum number of neurons is necessary for both classification accuracy and class verification accuracy to saturate. The optimum size of the penultimate layer depends on Minkowski  $p$  value:  $p=1$  (Manhattan) requires 128 neurons,  $p=2$  (Euclidean): 256,  $p>=3$  did not saturate even at 2,048 neurons. Once saturated, adding more neurons does not improve classification or verification accuracy. The relative weight  $\lambda_1$  of the Centre-Loss function within the total loss was 1.0 (ROC AUC: 0.977), although similar metrics were observed for  $\lambda_1$  in the range 0.1—3.0 (ROC AUC:  $\geq 9.65$ ). This shows the robustness of the loss function to variations of  $\lambda_1$ .

The suggested update in the Centre-Loss function  $L_{Inter}$  to increase Inter-Class distance did not give a positive result: higher  $\lambda_2$  values generally resulted in decreased ROC AUC, the optimal value of  $\lambda_2$  being 0.0. This concludes that having Cross-Entropy loss as part of the total loss function provides sufficient class centre separability.

Although all three types of networks (Centre-Loss, Siamese, Triplet) had their weights initialized from a pre-trained classifier, only Centre-Loss showed error reduction during the first epochs of training, when all the layers were trainable. Siamese and Triplet required fixing all the convolutional layer weights for training to achieve optimal validation error. The Siamese ROC AUC improved from 0.903 to 0.981, Triplet improved from 0.946 to 0.979 by leaving only the last two dense blocks trainable.

The Centre-Loss approach, having included Cross-Entropy as part of the loss function, can be used as a classifier, not just a verifier. The unexpected result of the research was the maintained strong accuracy of the classifier, by having included a Centre-Loss summand in the total loss function: with or without the Centre-Loss, the accuracy of the classifier was  $73\% \pm 0.8\%$ .

The three approaches (Centre-Loss, Siamese, Triplet) were evaluated on another retail image set of barcodeless products Fruits-360. The Centre-Loss demonstrated outstanding verification ROC AUC of 1.000, whereas Siamese and Triplet underperformed (ROC AUC: 0.976 and 0.977, respectively). ROC AUC metric on Fruits-360 generally was better than on the authentic self-checkout dataset, likely due to well-cleaned, synthetic data.

**Product grouping by similarity.** Three distinct approaches were investigated to determine class similarity. Based on determined class similarity, group classifiers were trained, and their performance was compared against the performance of the individual class classifier.

All three similarity approaches proposed different class pairs in the Top-5 similar pairs list, but image pairs from each proposed approach seem very similar to the human eye. Using all three approaches, the Top-5 pairs stand out in terms of similarity: in Error-Contribution, Top-5 pairs contribute to over 20% classification errors; in Embeddings-Distance, the Top-5 normalized similarity ranges between 5.7 and 8.3.

The experiments showed inconclusive results for the class similarity method to be used when merging classes into clusters. The best similarity approach depends on the count of "class groups" and the performance metric used (accuracy vs. f-score). Using the Error-Contribution similarity method improved the f-score by  $9.9\% \pm 9.1\%$  over SOM-Purity and  $7.8\% \pm 6.4\%$  over Embeddings-Distance. Although the SOM-Purity similarity method generally improved accuracy by  $1.2 \pm 1.5\%$ , its significance waned having a higher number of groups.

Classifiers trained on individual classes that predict "class group" usually outperform classifiers trained on "class groups". The accuracy of individual classifiers was  $1.8\% \pm 1.5\%$ , f-score was  $10.3\% \pm 9.0\%$  higher than that of group-classifiers.

## 6 General Conclusions

1. In the dataset preparation step of filtering empty images, a two-class emptiness classifier eliminated them with a commanding 98.8% confidence. Other techniques for emptiness (Siamese, OCC) performed worse. The separation of images having good vs. bad product visibility was achieved using an experimentally determined separating threshold with an f-score 0.906.
2. Experiments revealed CLAHE as the most effective method among those that were employed to reduce illumination differences. Higher image variability by augmenting images using both affine and perspective transformation outperformed affine-only transformations by 1.3%—2.8% and perspective-only transformations by 1.3%—3.9%. Using three perspectives outperformed a single perspective on average by 1.1%.
3. Product classification experiments showed that the proposed neural network architecture of 7 convolutional and 3 dense layers (accuracy 80.5%  $\pm$ 1.2%) tuned to self-checkout images outperformed well-known models: EfficientNet B0 (80.2% $\pm$ 1.2%), ResNet-50 (58%  $\pm$ 2.5%). Notably, the performance of the self-made architecture degrades by removing any convolutional or dense layer and no longer improves by adding layers. The proposed architecture’s accuracy on a synthetic Fruits-360 yields a comparable test accuracy of 99.6% to other authors.
4. Verification metrics of class prototype-based (Proxy-NCA, Centre-Loss) vs. sample-to-sample (Siamese, Triplet) were similar: ROC AUC of Proxy-NCA 0.985, Centre-Loss 0.979 vs. Siamese 0.981, Triplet 0.980; EER of Proxy-NCA 0.047, Centre-Loss 0.073 vs. Siamese 0.063 and Triplet 0.060. Euclidean distance in loss functions did not cede to other distance types (Manhattan, Minkowski, Cosine) in accuracy, although nearby Minkowski  $p$  values ( $p=1$  Manhattan,  $p=3$ ) followed closely. In the Centre-Loss approach, using Euclidean distance achieved 0.979 ROC AUC, whereas nearby Minkowski  $p$  values resulted in 0.962 ( $p=1$  Manhattan) and 0.948 ( $p=3$ ). Experiments with Centre-Loss architecture revealed the penultimate layer as the layer of choice. The size of the penultimate layer saturates depending on the Minkowski  $p$  value: higher Minkowski  $p$  values require more neurons to achieve saturation. A suggested update in the Centre-Loss function to increase Inter-Class distance did not give a positive result.



5. The same Centre-Loss neural network is applicable for two distinct tasks: classification and verification. This is due to the unexpected finding that employing a dual loss function did not impair classification accuracy:  $73.2\% \pm 0.8\%$  (vs  $73.2\%$  without Centre-Loss).
6. Experiments of grouping products by similarity, so that the accuracy of classifying into groups is maximized, showed that SOM-Purity generally improved accuracy by  $1.2 \pm 1.5\%$ , although its significance waned having a higher number of groups. The Error-Contribution similarity method improved the f-score by  $8.9\% \pm 7.7\%$ . The best similarity measure depends on the number of groups and performance metric used (accuracy vs. f-score). Classifiers trained on individual-class labels that predict a similar product group outperform classifiers trained on similarity-group labels. The accuracy of individual classifiers was  $1.8\% \pm 1.5\%$ , f-score was  $10.3\% \pm 9.0\%$  higher than the accuracy of group-classifiers.

## References

- [1] Teuvo Kohonen. “The self-organizing map”. In: *Proceedings of the IEEE* 78.9 (1990), pp. 1464–1480.
- [2] Yandong Wen et al. “A discriminative feature learning approach for deep face recognition”. In: *European conference on computer vision*. Springer. 2016, pp. 499–515.
- [3] Yair Movshovitz-Attias et al. “No fuss distance metric learning using proxies”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 360–368.
- [4] NCR. *Self-Checkout: a Global Consumer Perspective*. 2019. URL: [https://www.ncr.co.jp/wp-content/uploads/files/solutions/self/fl/fl\\_wpa/RET\\_SCO\\_wp.pdf](https://www.ncr.co.jp/wp-content/uploads/files/solutions/self/fl/fl_wpa/RET_SCO_wp.pdf).
- [5] Grand View Research. *Self-checkout Systems Market Size, Share and Trends Analysis Report By Components (Systems, Services), By Type (Cash Based, Cashless), By Application, By Region, And Segment Forecasts, 2020 - 2027*. 2020. URL: <https://www.grandviewresearch.com/industry-analysis/self-checkout-systems-market>.
- [6] Adrian Beck. *Self-Checkout in Retail: Measuring the Loss*. 2018. URL: <https://www.researchgate.net/publication/330214157>.
- [7] Mihai Oltean. *Fruits 360 dataset: new research directions*. 2021. URL: [https://www.researchgate.net/publication/354535752\\_Fruits\\_360\\_dataset\\_new\\_research\\_directions](https://www.researchgate.net/publication/354535752_Fruits_360_dataset_new_research_directions).
- [8] Bikash Santra and Dipti Prasad Mukherjee. “A comprehensive survey on computer vision based approaches for automatic identification of products in retail store”. In: *Image and Vision Computing* 86 (2019), pp. 45–63.
- [9] Michele Merler, Carolina Galleguillos, and Serge Belongie. “Recognizing groceries in situ using in vitro training data”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8.
- [10] Kyota Higa, Kota Iwamoto, and Toshiyuki Nomura. “Multiple object identification using grid voting of object center estimated from keypoint matches”. In: *2013 IEEE International Conference on Image Processing*. IEEE. 2013, pp. 2973–2977.
- [11] Marian George et al. “Fine-grained product class recognition for assisted shopping”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2015, pp. 154–162.

- [12] Timothy Chong, Idawati Bustan, and Mervyn Wee. “Deep learning approach to planogram compliance in retail stores”. In: *Semantic Scholar* (2016), pp. 1–6.
- [13] Ipek Baz, Erdem Yoruk, and Mujdat Cetin. “Context-aware hybrid classification system for fine-grained retail product recognition”. In: *2016 IEEE 12th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*. IEEE. 2016, pp. 1–5.
- [14] Song Liu et al. “Planogram compliance checking based on detection of recurring patterns”. In: *IEEE MultiMedia* 23.2 (2016), pp. 54–63.
- [15] Laith Alzubaidi et al. “A deep convolutional neural network model for multi-class fruits classification”. In: *International Conference on Intelligent Systems Design and Applications*. Springer. 2019, pp. 90–99.
- [16] Horea Murecsan and Mihai Oltean. “Fruit recognition from images using deep learning”. In: *arXiv preprint arXiv:1712.00580* (2017).
- [17] Dang Thi Phuong Chung and Dinh Van Tai. “A fruits recognition system based on a modern deep learning technique”. In: *Journal of physics: conference series*. Vol. 1327. IOP Publishing. 2019, p. 12050.
- [18] Mingxing Tan and Quoc Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.
- [19] Dayong Wang et al. “Expanding the field-of-view and profile measurement of covered objects in continuous-wave terahertz reflective digital holography”. In: *Optical Engineering* 58.2 (2019), pp. 1–7. DOI: 10.1117/1.OE.58.2.023111. URL: <https://doi.org/10.1117/1.OE.58.2.023111>.
- [20] J Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. 2009, pp. 248–255.
- [21] Mark Everingham et al. “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88.2 (2010), pp. 303–338.
- [22] Li Fei-Fei, R Fergus, and P Perona. “One-shot learning of object categories”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.4 (Apr. 2006), pp. 594–611. ISSN: 1939-3539.

- [23] Bryan C Russell et al. “LabelMe: a database and web-based tool for image annotation”. In: *International journal of computer vision* 77.1-3 (2008), pp. 157–173.
- [24] Nouman Ali and Bushra Zafar. *MSRC-v2 image dataset*. Aug. 2018. DOI: 10.6084/m9.figshare.6075788.v2. URL: [https://figshare.com/articles/dataset/MSRC-v2\\_image\\_dataset/6075788/2](https://figshare.com/articles/dataset/MSRC-v2_image_dataset/6075788/2).
- [25] Jelena Liutvinavičienė and Olga Kurasova. “Multi-level Massive Data Visualization: Methodology and Use Cases”. In: *Baltic Journal of Modern Computing* 6.4 (2018), pp. 321–334. ISSN: 2255-8942.
- [26] V Nežerka and J Trejbal. “Assessment of aggregate-bitumen coverage using entropy-based image segmentation”. In: *Road Materials and Pavement Design* 21.8 (2019), pp. 1–12. ISSN: 1468-0629.
- [27] Zoran Zivkovic and Ferdinand Van Der Heijden. “Efficient adaptive density estimation per image pixel for the task of background subtraction”. In: *Pattern Recognition Letters* 27.7 (2006), pp. 773–780. ISSN: 01678655.
- [28] Yaniv Taigman et al. “DeepFace: Closing the gap to human-level performance in face verification”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. June 2014, pp. 1701–1708. ISBN: 9781479951178. DOI: 10.1109/CVPR.2014.220.
- [29] Bartosz Krawczyk and Michał Woźniak. “Experiments on distance measures for combining one-class classifiers”. In: *2012 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE. 2012, pp. 89–92.
- [30] Shehroz S Khan and Michael G Madden. “A survey of recent trends in one class classification”. In: *Irish conference on artificial intelligence and cognitive science*. Springer. 2009, pp. 188–197.
- [31] Michael Kemmler et al. “One-class classification with Gaussian processes”. In: *Pattern recognition* 46.12 (2013), pp. 3507–3518.
- [32] Yunqiang Chen, Xiang Sean Zhou, and Thomas S Huang. “One-class SVM for learning in image retrieval”. In: *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*. Vol. 1. IEEE. 2001, pp. 34–37.
- [33] Xichun Bi and Lifang Wang. “Performing Weakly Supervised Retail Instance Segmentation via Region Normalization”. In: *IEEE Access* 9 (2021), pp. 67761–67775.

- [34] Patrick Follmann et al. “MVTec D2S: Densely Segmented Supermarket Dataset”. In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Cham: Springer International Publishing, 2018, pp. 581–597. ISBN: 978-3-030-01249-6.
- [35] Xiu-Shen Wei et al. “RPC: A large-scale retail product checkout dataset”. In: *arXiv preprint arXiv:1901.07249* (2019).
- [36] Qi Wang et al. “Hybrid feature aligned network for salient object detection in optical remote sensing imagery”. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–15.
- [37] Yanfeng Liu et al. “Distilling Knowledge from Super Resolution for Efficient Remote Sensing Salient Object Detection”. In: *IEEE Transactions on Geoscience and Remote Sensing* (2023).
- [38] Luis Rosado et al. “Supervised learning for Out-of-Stock detection in panoramas of retail shelves”. In: *IST 2016 - 2016 IEEE International Conference on Imaging Systems and Techniques, Proceedings*. IEEE, 2016, pp. 406–411. ISBN: 9781509018178. DOI: 10.1109/IST.2016.7738260.
- [39] Bernardas Ciapas and Povilas Treigys. “High F-score Model for Recognizing Object Visibility in Images with Occluded Objects of Interest”. In: *Baltic Journal of Modern Computing* 9.1 (2021), pp. 35–48.
- [40] Nitesh V Chawla et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.
- [41] Maayan Frid-Adar et al. “GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification”. In: *Neurocomputing* 321 (2018), pp. 321–331. ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2018.09.013>. URL: <https://www.sciencedirect.com/science/article/pii/S0925231218310749>.
- [42] Ke Wang et al. “Perspective Transformation Data Augmentation for Object Detection”. In: *IEEE Access* 8 (2020), pp. 4935–4943. DOI: 10.1109/ACCESS.2019.2962572.
- [43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems* 25. Ed. by F Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. ISBN: 9781420010749. URL: <http://papers>.

- nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.
- [44] Siddharth Srivastava and Gaurav Sharma. “OmniVec: Learning robust representations with cross modal sharing”. In: *arXiv preprint arXiv:2311.05709* (2023).
  - [45] Omid E. David and Nathan S. Netanyahu. “DeepPainter: Painter classification using deep convolutional autoencoders”. In: *Proceedings of the International conference on artificial neural networks*. Springer. 2016, pp. 20–28. ISBN: 9783319447803. DOI: 10.1007/978-3-319-44781-0\_3. arXiv: 1711.08763.
  - [46] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *International Conference on Learning Representations (ICLR)* (2021).
  - [47] Xiaohua Zhai et al. “Scaling vision transformers”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12104–12113.
  - [48] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. “Dynamic routing between capsules”. In: *Advances in neural information processing systems* 30 (2017).
  - [49] Mingsheng Long et al. “Unsupervised domain adaptation with residual transfer networks”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 136–144. arXiv: 1602.04433.
  - [50] Shibai Yin, Yiming Qian, and Minglun Gong. “Unsupervised hierarchical image segmentation through fuzzy entropy maximization”. In: *Pattern Recognition* 68 (2017), pp. 245–259. ISSN: 0031-3203. URL: <http://www.sciencedirect.com/science/article/pii/S0031320317301115>.
  - [51] Sergejs Kodors. “Detection of Man-Made Constructions using LiDAR Data and Decision Trees”. In: *Baltic Journal of Modern Computing* 7.2 (2019), pp. 255–270.
  - [52] J R Quinlan. “Induction of decision trees”. In: *Machine Learning* 1.1 (1986), pp. 81–106. ISSN: 1573-0565. DOI: 10.1007/BF00116251. URL: <https://doi.org/10.1007/BF00116251>.
  - [53] Matīss Riktērs. “Hybrid Machine Translation by Combining Output from Multiple Machine Translation Systems”. In: *Baltic Journal of Modern Computing* 7.3 (2019), pp. 301–341. ISSN: 2255-8942.

- [54] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “Facenet: A unified embedding for face recognition and clustering”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.
- [55] Yifan Sun et al. “Circle loss: A unified perspective of pair similarity optimization”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 6398–6407.
- [56] Weiyang Liu et al. “Sphereface: Deep hypersphere embedding for face recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 212–220.
- [57] Jiankang Deng et al. “Arcface: Additive angular margin loss for deep face recognition”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4690–4699.
- [58] Chi Zhang et al. “Deepemd: Few-shot image classification with differentiable earth mover’s distance and structured classifiers”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 12203–12213.
- [59] Qi Qian et al. “Softtriple loss: Deep metric learning without triplet sampling”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6450–6458.
- [60] Sungyeon Kim et al. “Proxy anchor loss for deep metric learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 3238–3247.
- [61] Mehdi Sadeghi, Keivan Maghooli, and Mohammad Shahram Moein. “Using artificial immunity network for face verification.” In: *Int. Arab J. Inf. Technol.* 11.4 (2014), pp. 354–361.
- [62] Huajie Jiang et al. “Learning Class Prototypes via Structure Alignment for Zero-Shot Recognition”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.
- [63] Nouha Othman, Rim Faiz, and Kamel Smaili. “Manhattan siamese LSTM for question retrieval in community question answering”. In: *On the Move to Meaningful Internet Systems: OTM 2019 Conferences: Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21–25, 2019, Proceedings*. Springer. 2019, pp. 661–677.

- [64] Kareem Amin et al. “Advanced similarity measures using word embeddings and siamese networks in CBR”. In: *Intelligent Systems and Applications: Proceedings of the 2019 Intelligent Systems Conference (IntelliSys) Volume 2*. Springer. 2020, pp. 449–462.
- [65] Zainab Imtiaz et al. “Duplicate questions pair detection using siamese malstm”. In: *IEEE Access* 8 (2020), pp. 21932–21942.
- [66] Zhongguo Wang and Bao Zhang. “Chinese Text Similarity Calculation Model Based on Multi-Attention Siamese Bi-LSTM”. In: *Proceedings of the 4th International Conference on Computer Science and Software Engineering*. 2021, pp. 93–98.
- [67] Hong Zeng et al. “Siam-GCAN: A Siamese Graph Convolutional Attention Network for EEG Emotion Recognition”. In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–9.
- [68] Andreas Bühler et al. “Deep Unsupervised Common Representation Learning for LiDAR and Camera Data using Double Siamese Networks”. In: *arXiv preprint arXiv:2001.00762* (2020).
- [69] S Öztürk. “Comparison of Pairwise Similarity Distance Methods for Effective Hashing”. In: *IOP Conference Series: Materials Science and Engineering*. Vol. 1099. 1. IOP Publishing. 2021, p. 012072. DOI: 10.1088/1757-899x/1099/1/012072.
- [70] Jason Yosinski et al. “How transferable are features in deep neural networks?” In: *arXiv preprint arXiv:1411.1792* (2014).
- [71] Maxime Oquab et al. “Learning and transferring mid-level image representations using convolutional neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1717–1724.
- [72] Bharat Singh et al. “R-FCN-3000 at 30fps: Decoupling Detection and Classification”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Ed. by Jyoti Aneja, Aditya Deshpande, and Alexander G. Schwing. IEEE, June 2018, pp. 1082–1090.
- [73] Ming-Ming Cheng et al. “BING: Binarized normed gradients for objectness estimation at 300fps”. In: *Computational Visual Media* 5.1 (2019), pp. 3–20. ISSN: 2096-0662. URL: <https://doi.org/10.1007/s41095-018-0120-1>.



- [74] D Jeyabharathi and A Suruliandi. “Performance analysis of feature extraction and classification techniques in CBIR”. In: *2013 International Conference on Circuits, Power and Computing Technologies*. IEEE, 2013, pp. 1211–1214.
- [75] Zhongmin Zhang, Jiawei Chen, and Shengli Wu. “Query performance prediction and classification for information search systems”. In: *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*. Ed. by Xu J. Cai Y., Ishikawa Y. Springer, 2018, pp. 277–285.
- [76] M Mowafy, A Rezk, and H El-Bakry. “An efficient classification model for unstructured text document”. In: *American Journal of Computer Science and Information Technology* 6.1 (2018), p. 16.
- [77] B C Russell et al. “Using Multiple Segmentations to Discover Objects and their Extent in Image Collections”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2. June 2006, pp. 1605–1614.
- [78] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. “Measuring the objectness of image windows”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (Nov. 2012), pp. 2189–2202. ISSN: 01628828.
- [79] Sarah M Erfani et al. “High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning”. In: *Pattern Recognition* 58 (2016), pp. 121–134.
- [80] Xin Sun et al. “Conditional Gaussian Distribution Learning for Open Set Recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.
- [81] Zhiguo Ding and Minrui Fei. “An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window”. In: *IFAC Proceedings Volumes* 46.20 (2013), pp. 12–17.
- [82] Meng Joo Er et al. “Face recognition with radial basis function (RBF) neural networks”. In: *IEEE transactions on neural networks* 13.3 (2002), pp. 697–710.
- [83] Xiangning Chen et al. “Symbolic discovery of optimization algorithms”. In: *arXiv preprint arXiv:2302.06675* (2023).
- [84] Jiahui Yu et al. “Coca: Contrastive captioners are image-text foundation models, 2022”. In: *arXiv preprint arXiv:2205.01917* (2022). arXiv: 2205.01917.

- [85] Mitchell Wortsman et al. “Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 23965–23998.
- [86] Ziwei Liu et al. “Large-scale long-tailed recognition in an open world”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 2537–2546.
- [87] Terrance DeVries and Graham W Taylor. “Learning confidence for out-of-distribution detection in neural networks”. In: *arXiv preprint arXiv:1802.04865* (2018).
- [88] Abhijit Bendale and Terrance E Boult. “Towards open set deep networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1563–1572.
- [89] Zhengxiang Wang, Yiqun Hu, and Liang-Tien Chia. “Image-to-class distance metric learning for image classification”. In: *European Conference on Computer Vision*. Springer. 2010, pp. 706–719.
- [90] Ghaliya Alfarsi et al. “Techniques for face verification: Literature review”. In: *2019 International Arab Conference on Information Technology (ACIT)*. IEEE. 2019, pp. 107–112.
- [91] Azzam Sleit, R Abu-Hurra, and Wesam Almobaideen. “Lower-quarter-based face verification using correlation filter”. In: *The Imaging Science Journal* 59.1 (2011), pp. 41–48.
- [92] Tong Che et al. “Deep verifier networks: Verification of deep discriminative models with deep generative models”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 8. 2021, pp. 7002–7010.
- [93] Yiming Shen et al. “DSRPH: Deep semantic-aware ranking preserving hashing for efficient multi-label image retrieval”. In: *Information Sciences* 539 (2020), pp. 145–156.
- [94] Liangliang Wang and Deepu Rajan. “An image similarity descriptor for classification tasks”. In: *Journal of Visual Communication and Image Representation* 71 (2020), p. 102847.
- [95] Jiang Wang et al. “Learning fine-grained image similarity with deep ranking”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1386–1393.
- [96] Yazhou Ren et al. “Deep density-based image clustering”. In: *Knowledge-Based Systems* 197 (2020), p. 105841.

- [97] Tsung Wei Tsai, Chongxuan Li, and Jun Zhu. “Mice: Mixture of contrastive experts for unsupervised image clustering”. In: *International Conference on Learning Representations*. 2020.
- [98] Xu Ji, Joao F Henriques, and Andrea Vedaldi. “Invariant information clustering for unsupervised image classification and segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9865–9874.
- [99] Pooran Singh Negi, Mohammad Mahoor, et al. “Leveraging Class Similarity to Improve Deep Neural Network Robustness”. In: *arXiv preprint arXiv:1812.09744* (2018).
- [100] Keunyoung Park and Doo-Hyun Kim. “Accelerating image classification using feature map similarity in convolutional neural networks”. In: *Applied Sciences* 9.1 (2019), p. 108.
- [101] Rodolfo M Pereira, Yandre M G Costa, and Carlos N Silla. “Handling imbalance in hierarchical classification problems using local classifiers approaches”. In: *Data Mining and Knowledge Discovery* 35.4 (2021), pp. 1564–1621.
- [102] Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. “Hierarchical multi-label classification networks”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5075–5084.
- [103] Haomin Chen et al. “Deep hierarchical multi-label classification of chest X-ray images”. In: *International conference on medical imaging with deep learning*. PMLR. 2019, pp. 109–120.
- [104] Brendan Kolisnik, Isaac Hogan, and Farhana Zulkernine. “Condition-CNN: A hierarchical multi-label fashion image classification model”. In: *Expert Systems with Applications* 182 (2021), p. 115195.
- [105] Marian George and Christian Floerkemeier. “Recognizing products: A per-exemplar multi-label image classification approach”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 440–455.
- [106] Philipp Jund et al. “The freiburg groceries dataset”. In: *arXiv preprint arXiv:1611.05799* (2016).
- [107] Karel Zuiderveld. “Contrast Limited Adaptive Histogram Equalization”. In: *Graphics Gems*. Ed. by Paul S Heckbert. Academic Press, 1994, pp. 474–485. ISBN: 978-0-12-336156-1. URL: <http://www.sciencedirect.com/science/article/pii/B9780123361561500616>.

- [108] Roweida Mohammed, Jumanah Rawashdeh, and Malak Abdullah. “Machine learning with oversampling and undersampling techniques: overview study and experimental results”. In: *2020 11th International Conference on Information and Communication Systems (ICICS)*. IEEE, 2020, pp. 243–248.
- [109] Matthew D. Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision*. Ed. by D. Fleet et al. Springer, 2014, pp. 818–833. ISBN: 9783319105895. arXiv: 1311.2901.
- [110] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *3rd International Conference on Learning Representations*. Ed. by Yoshua Bengio and Yann LeCun. Morgan Kaufmann Publishers Inc., 2015, pp. 1–14. arXiv: arXiv:1409.1556v6. URL: <http://arxiv.org/abs/1409.1556>.
- [111] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, 2016, pp. 770–778. ISBN: 9781467388504. arXiv: 1512.03385. URL: <https://doi.org/10.1109/CVPR.2016.90>.
- [112] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis R Bach and David M Blei. JMLR.org, 2015, pp. 448–456. ISBN: 9781510810587. arXiv: 1502.03167. URL: <http://proceedings.mlr.press/v37/ioffe15.html>.
- [113] Shibani Santurkar et al. “How does batch normalization help optimization?” In: *Advances in neural information processing systems* 31 (2018).
- [114] Geoffrey E Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580* (2012).
- [115] Nitish Srivastava et al. “Dropout: A simple way to prevent neural networks from overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. ISSN: 15337928. URL: <http://jmlr.org/papers/v15/srivastava14a.html>.
- [116] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014). URL: <https://arxiv.org/abs/1412.6980>.

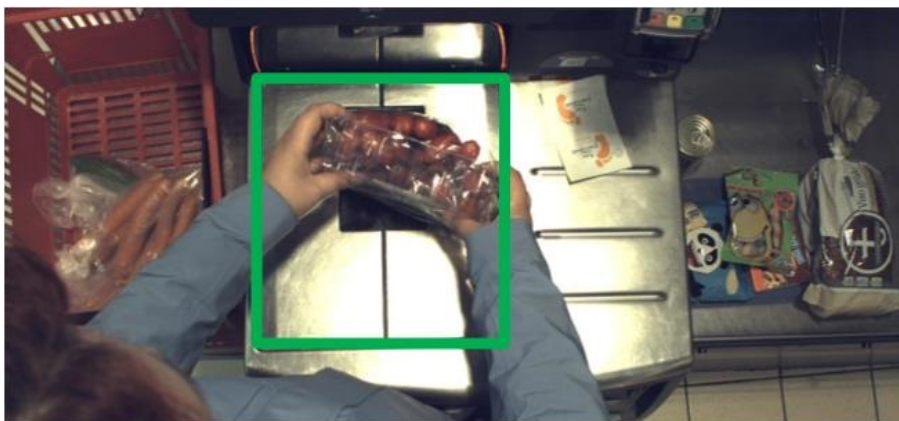
- [117] Andrea Dal Pozzolo et al. “Calibrating probability with undersampling for unbalanced classification”. In: *2015 IEEE Symposium Series on Computational Intelligence*. 2015, pp. 159–166.
- [118] Kaiming He et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.
- [119] Barret Zoph et al. “Learning transferable architectures for scalable image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8697–8710.
- [120] Xiaowei Huang et al. “Safety verification of deep neural networks”. In: *International conference on computer aided verification*. Springer. 2017, pp. 3–29.
- [121] Sydney M Katz et al. “Verification of image-based neural network controllers using generative models”. In: *Journal of Aerospace Information Systems* 19.9 (2022), pp. 574–584.
- [122] Fuzhen Zhuang et al. “A comprehensive survey on transfer learning”. In: *Proceedings of the IEEE* 109.1 (2020), pp. 43–76.
- [123] Srikanth Tammina. “Transfer learning using vgg-16 with deep convolutional neural network for classifying images”. In: *International Journal of Scientific and Research Publications (IJSRP)* 9.10 (2019), pp. 143–150.
- [124] G Jignesh Chowdary et al. “Face mask detection using transfer learning of inceptionv3”. In: *International Conference on Big Data Analytics*. Springer. 2020, pp. 81–90.
- [125] Xiuyao Song et al. “Conditional anomaly detection”. In: *IEEE Transactions on knowledge and Data Engineering* 19.5 (2007), pp. 631–645.
- [126] Gwangbin Bae et al. “DigiFace-1M: 1 Million Digital Face Images for Face Recognition”. In: *2023 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2023.
- [127] Ziwei Liu et al. “Deep learning face attributes in the wild”. In: *Proceedings of the IEEE International Conference on Computer Vision*. Vol. 2015 Inter. Dec. 2015, pp. 3730–3738. ISBN: 9781467383912. DOI: 10.1109/ICCV.2015.425. arXiv: 1411.7766.
- [128] Shuo Yang et al. “Wider Face: A Face Detection Benchmark”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5525–5533.

- [129] Mateusz Pabian, Dominik Rzepka, and Mirosław Pawlak. “Supervised Training of Siamese Spiking Neural Networks with Earth Mover’s Distance”. In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 4233–4237. DOI: 10 . 1109 / ICASSP43922 . 2022 . 9746630.
- [130] Sachin Kumar, Soumen Chakrabarti, and Shourya Roy. “Earth Mover’s Distance Pooling over Siamese LSTMs for Automatic Short Answer Grading.” In: *IJCAI*. 2017, pp. 2046–2052.
- [131] Hasan Tercan, Alexandro Guajardo, and Tobias Meisen. “Industrial transfer learning: Boosting machine learning in production”. In: *2019 IEEE 17th international conference on industrial informatics (INDIN)*. Vol. 1. IEEE. 2019, pp. 274–279.
- [132] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. “What makes ImageNet good for transfer learning?” In: *arXiv preprint arXiv:1608.08614* (2016).
- [133] Olga Kurasova and Alma Molytè. “Quality of quantization and visualization of vectors obtained by neural gas and self-organizing map”. In: *Informatica* 22.1 (2011), pp. 115–134.
- [134] Julius Venskus. “Semi-supervised and Unsupervised Machine Learning Methods for Sea Traffic Anomaly Detection”. PhD thesis. Vilniaus universitetas, 2021.
- [135] C Manning and P Nayak. *Introduction to Information Retrieval-Evaluation*. 2013.

## Ižanga

Mažmeninės prekybos savitarnos kasos padeda klientams greičiau apsipirkti, o mažmenininkams – sumažinti kaštus. Savitarnos kasų skaičius pasaulyje 2019 m. sudarė apie 325 000 [4] ir augs 13,3 % [5] kasmet. Vidutiniškai 7 prekės pirminių krepšelyje ir vidutiniškai 1 400 operacijų kiekvienoje savitarnos kasoje per savaitę sudaro apie 10 000 prekių pardavimų. Įprastai didelės mažmeninės prekybos parduotuvės gali turėti iki 30 000 prekių asortimentą. Dauguma prekių turi lengvai atpažįstamus brūkšninius kodus. Vis tik tipinėje 800–1200 kvadratinį metrų ploto parduotuvėje yra 200–300 prekių be brūkšninio kodo (šiam tyrimui naudotos 194 klasės). Asortimentas nuolat keičiasi dėl sezoniškumo ir tiekėjų pasikeitimų. Dažnai prekių vizualinis panašumas mažai koreliuoja su priklausymu tam tikrai prekių grupei. Pavyzdžiui, raudoni obuoliai yra panašesni į pomidorus nei į žalius obuolius, nors raudoni ir žali obuoliai gali būti toje pačioje prekių kategorijoje. Įvairių saldinių rūšių panašumą į kitus gaminius lemia vyniojamasis popierius.

57 pav. vaizduojamas prekės judėjimas atsiskaitant. Pirmiausia klientas padeda pirminių krepšėlį (kairėje). Tada paima po vieną prekę iš krepšelio ir registruoja vienu iš dviejų būdų: nuskaito (prekes su brūkšninio kodo lipdukais, pvz., pieno pakuotes) arba pasirenka (be brūkšninio kodo, pvz., vaisius) iš meniu, kurio struktūra yra 3–5 lygių hierarchinis medis. Dėl sudėtingo meniu ir daugelio panašių prekių dažnai pasirenkamos neteisingos prekės, o klientai atsiskaitydami



Pirminių krepšelis  
(kelios prekės)



Svarstyklių zona  
(viena prekė,  
tyrimo sritis)



Pakavimo zona  
(kelios prekės)

57 paveikslas: Atsiskaitymo eiga.

užtrunka ilgiau. Iš meniu išsirinkus prekę be brūkšninio kodo, ji pasverinama svarstyklėmis (57 pav. žalias stačiakampis). Galiausiai, užregistravęs prekę, klientas perkelia ją į pakavimo zoną (dešinėje).

Remiantis ECR savitarnos kasų ataskaita [6], kuri apėmė 13 mažmenininkų, mažmeninės prekybos parduotuvėse, kuriose 50 % operacijų atliekama savitarnos kasose, nuostoliai dėl vagysčių yra 75 % didesni. Tas pats tyrimas atskleidė, kad 43,4 % visų pirkinių krepšelių yra neteisingai pasirinktų prekių. Piktavališki klientai savitarnos kasomis piktnaudžiauja įvairiais būdais: brangių prekių brūkšninius kodus pakeičia pigesnių, o pigesnes prekes tyčia renkasi iš meniu pasirinkimo sąrašo. Vagystės iš parduotuvės įvyksta pasirinkus netinkamą prekę, pakeitus brūkšninį kodą, nenuskenavus dalies prekių arba išėjus nesumokėjus. Mažmenininkai bando apsisaugoti nuo vagysčių naudodami apsaugines svarstyklas, kurios tinka fiksuoto svorio prekėms, bet ne kintamo svorio, pvz., šviežiams vaisiams ir daržovėms. Kai kurie pritvirtina RFID tags žymas ant didelės vertės prekių, tačiau dažnai tai brangu ir nepraktiška, pvz., nesupakuotų vaisių ir daržovių atveju. Apsaugos kameros, nors paprastai stebi savitarnos zoną, generuoja per daug filmuotos medžiagos, kad apsaugos darbuotojai galėtų jas stebėti realiuoju laiku ir nustatyti vagystes.

## Problemos aprašymas

Mažmeninės prekybos įmonės siekia išspręsti savitarnos kasose aktualias problemas – vagystes ir ilgai trunkantį prekių be brūkšninio kodo identifikavimą. Sėkmingi sprendimai supaprastintų prekių pasirinkimą ir įspėtų apie neteisingus prekių pasirinkimus. Nebūtina visiškai automatizuoti atsiskaitymo proceso, kaip parduotuvėse be kasų ir kasininkų, tačiau siūlomas sprendimas privalo būti ekonomišką, pageidautina pasikliauti tik viena kamera ir apsieiti be kitos papildomos aparatūrinės įrangos, pvz., galingos vaizdo plokštės. Prekių atpažinimo sprendimas turi būti visiškai automatizuotas: bet kokia žmonių sąveika bet kuriame etape (pvz., rankinis duomenų rinkinio valymas, žymėjimas ir t. t.) yra nepraktiška dėl dažno asortimento keitimosi (dėl sezoniškumo ir tiekėjų kaitos) ir didelio prekių bei vaizdų skaičiaus. Sprendimas turėtų apimti prekių vaizdų rinkimą, naujų prekių įtraukimą, vaizdų žymėjimą vienos prekės žyma ir vaizdų, kuriuose prekės nėra arba ji blogai matoma, šalinimą.

Duomenų aibė prekių atpažinimo sprendimui turi atitikti savitarnos kasų vaizdų ypatybes: netolygų prekių pasiskirstymą, skirtingą kasų apšvietimą, dalis vaizdų turi būti uždengti pirkėjų kūno dalimis. Dauguma lyginamųjų duomenų aibių („ImageNet“, CIFAR[-10]-100], MNIST) apima tik tuos vaizdus, kuriuose dominančių objektų



matomumas yra geras. Sintetinės mažmeninės prekybos prekių duomenų aibės, pvz., „Fruits 360“ [7], neatspindi vaizdų, kuriuos reikia atpažinti savitarnos kasose; vargu ar galima pasiekti tikslų tokių duomenų aibių atpažinimą mažmeninėse parduotuvėse.

Didelėje vaizdų dalyje yra kūno dalių (rankų, galvų), kurios iš dalies ar visiškai uždengia prekę. Atpažinimo sekos pradžioje turi būti vaizdų, kuriuose prekė negali būti atpažįstama, atmetimo etapas. Atmesti vaizdai neturi dalyvauti jokiuose tolesniuose etapuose, pvz., klasifikuojant prekes, kad būtų lengviau pasirinkti, ar patikrinant kliento pasirinkimą.

Prekės pasirinkimui palengvinti sprendimas turi atrinkti vieną ar keletą panašiausių prekių iš viso parduodamų prekių, neturinčių brūkšninio kodo, sąrašo. Reikia atsižvelgti į santykinai mažesnę retai parduodamų prekių pardavimų skaičių. Svarbus ne tik panašiausios, bet ir kelių (2–5) panašiausių prekių klasifikavimo tikslumas (pvz., pasirinkimo sąrašo meniu gali būti pateiktas sutrumpintas 1–5 prekių sąrašas). Būtina galimybė sprendimą įdiegti į mažos galios, neturinčius grafikos procesoriaus savitarnos kasų įrenginius.

Klientui pasirinkus prekę iš meniu, reikia patikrinti, ar vaizde esanti prekė atitinka kliento pasirinkimą. Kadangi klientų pasirinkimas apsiriboja meniu elementais (t. y. prekėmis be brūkšninio kodo), pakanka atpažinti, ar vaizdas atitinka pasirinktą prekę be brūkšninio kodo. Jei yra didelė neatitikimo tikimybė, apsaugos darbuotojas turi būti įspėjamas vizualiai patvirtinti pasirinkimą. Toks sprendimas neleistų į krepšelį įsidėti brangaus gėrimo butelio ir iš meniu pasirinkti bet kokią prekę be brūkšninio kodo. Tačiau toks sprendimas neaptiktų pakeistų brūkšninių kodų.

## Tyrimo tikslas ir uždaviniai

Tyrimo tikslas – pasiūlyti ir ištirti automatizuotą prekių be brūkšninio kodo atpažinimo eigą mažmeninės prekybos maisto parduotuvių savitarnos kasose. Tyrimo uždaviniai:

- ištirti surinktus savitarnos prekių vaizdus tiek kiekybiškai, tiek kokybiškai; pasiūlyti schemą duomenų aibei, kuri būtų tinkama neuroniniams tinklams mokyti, parengti. Schemoje turi būti atsižvelgta į tai, kad didelė vaizdų dalis yra tuščia arba apima kliento kūno dalis, kurios uždengia dalį prekės, kai kurios prekės yra plastikiniuose maišeliuose, o prekių nuotraukų skaičius yra iškreiptas;
- pasiūlyti metodus įvertinti vaizdo tinkamumą produkto atpažįstamumui, patikrinti jų efektyvumą atliekant abliacijos

tyrimus. Sukurti neuroninio tinklo architektūrą prekėms klasifikuoti savitarnos kasų vaizduose, palyginti su moderniausiomis architektūromis naudojant autentiškus savitarnos kasų vaizdus, įvertinti bendrinamumą panašioms viešosioms duomenų aibėms. Siūloma architektūra turi tikt mažos galios, grafinės plokštės neturintiems savitarnos kasų įrenginiams. Taip pat pasiūlyti deterministinį ir skaičiavimo požiūriu našų metodą patikrinti, ar kliento meniu pasirinkimas atitinka prekę ant svarstyklų, išmatuoti jo tikslumą;

- pasiūlyti prekių grupavimo pagal panašumą metodą, kad būtų maksimaliai padidintas (panašių prekių) grupės prognozavimo tikslumas; įvertinti prognozavimo tikslumo padidėjimą lyginant su panašiausios prekės prognozavimo tikslumu.

## Mokslinė svarba

Disertacija prisideda prie kompiuterine rege paremtų prekių atpažinimo tyrimų mažmeninės prekybos maistu savitarnos kasų aplinkoje. Toliau pateikiami pagrindiniai indėliai ir jų praktinė vertė:

- pristatytas automatiškai atnaujinamas prekių atpažinimo procesas. Jis apima vaizdų rinkimą, žymėjimą, filtravimą, išankstinį apdorojimą ir klasifikatoriaus mokymą. Proceso etapai patvirtinti abliacijos tyrimais. Pagrindinė nauda – galimybė dažnai atnaujinti modelį be žmogaus įsikišimo, o tai svarbu dinamiškoje mažmeninės prekybos aplinkoje;
- pristatytas konvoliucinio neuroninio tinklo architektūros projektavimo metodas, pritaikytas prekėms savitarnos kasų vaizduose atpažinti. Lyginamieji bandymai atskleidė, kad metodas yra toks pat tikslus arba tikslesnis už gerai žinomus tinklus, pvz., „EfficientNet“ ir „ResNet“, naudojant autentišką savitarnos kasų vaizdų rinkinį ir viešąjį duomenų rinkinį „Fruits 360“. Tai leidžia efektyviai klasifikuoti naudojant mažesnius neuroninius tinklus, tinkamus mažos galios savitarnos kasų įrenginiams;
- iširtas klasės patikros metodas naudojant atstumo nuo centro tikslo funkciją, kurio tikslumas panašus į vaizdų palyginimo metodų. Pažymėtina, kad atstumo nuo centro patikros metodo skaičiavimo efektyvumas pranoksta vaizdų lyginimo („Siamo“) metodo, nes išvengiama kelių vaizdų lyginimo ir atsitiktinio vaizdų parinkimo. Bendrinamumas patvirtintas naudojant viešąjį duomenų rinkinį „Fruits 360“. Tai leidžia tiksliai patikrinti kliento prekės pasirinkimą taikant deterministinį algoritmą;

- ištirti trys algoritmai, siekiant įvertinti prekių panašumą ir atitinkamai sugrupuoti prekes. Jie paremti klaidų indėlių klasifikavimo matricoje, vidutiniu atstumu tarp aktyvacijų ir SOM grynumo pagerėjimu. Šie metodai supaprastina prekės pasirinkimą, parodant panašiausios grupės prekes klientui pasirinkti vietoje viso pasirinkimo medžio.

## Duomenų paruošimas

### Vaizdų žymėjimas







Buvo naudojamos dviejų tipų žymos:

- prekės ID; žymos naudojamos prekių klasifikatoriui mokyti;
- prekės matomumo kategorijos; žymos naudojamos prekės matomumo klasifikatoriui, kuris vėliau naudotas vaizdams su prastai matomomis prekėmis pašalinti, mokyti.

Visi vaizdai automatiškai sužymėti prekių žymomis pagal apsiperkančių klientų pasirinkimą.

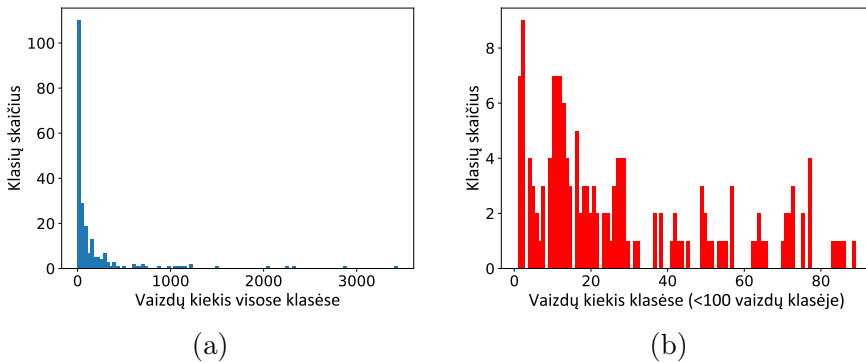
Dalis vaizdų rankiniu būdu sužymėti matomumo lygio žymomis. Buvo naudojamos keturios ordinalinės žymos (Q1, Q2, Q3, Q4) pagal matomą prekės dalį vaizde: Q1 turėjo iki 25 % matomumą, Q2 nuo 25 % iki 50 % ir t. t. Vaizdai su prekėmis plastikiniuose maišeliuose sužymėti atskiromis žymomis dėl neaiškumo, ar juose esančias prekes pavyks atpažinti, suskirstant juos į „Bag“ (žmogui atpažįstami) ir „BagR“ (neatpažįstami dėl šviesos atspindžio). Kiekvienos klasės pavyzdžiai pateikiami 21 lentelėje.

21 lentelė: Prekės matomumo klasės, pavyzdžiai ir procentinė dalis.

					
Q1 32 %	Q2 22 %	Q3 15 %	Q4 21 %	Bag 7.3 %	BagR 2.6 %

## Prekių taksonomija ir pardavimų dažnis

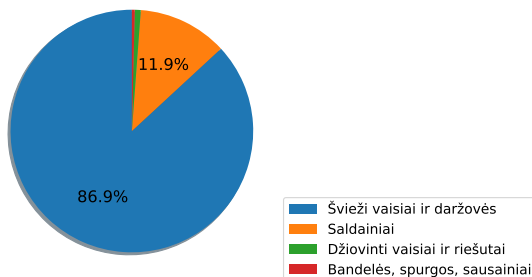
Maisto parduotuvių asortimentą daugiausia sudaro prekės su brūkšniniu kodu, tačiau keli šimtai prekių rūšių jo neturi (nesupakuoti vaisiai, daržovės, riešutai, sveriami saldainiai ir pan.). Šiame tyrime nagrinėjamas prekių be brūkšninio kodo atpažinimas siekiant pagreitinti prekės pasirinkimą ir sumažinti neteisingų pirkėjų pasirinkimų skaičių. Surinktą duomenų aibę sudaro 26 637 vaizdai, priklausantys 194 prekėms. Duomenų aibė yra išbalansuota kaip parodyta 58 pav.: aibėje yra 3 282 bananų, 2 760 morkų, 2 181 citrinų vaizdai, o rečiausių prekių vaizdų yra tik po 3.



58 paveikslas: Vaizdų kiekis pagal prekę - bendras (a) ir prekių, turinčių iki 100 vaizdų (b).

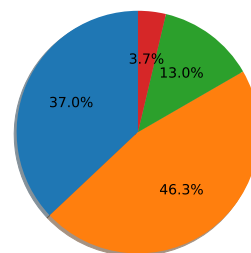
Duomenų aibės pasiskirstymas pagal prekių grupę pavaizduotas 59 pav. Pagal pardavimą dominuoja „švieži vaisiai ir daržovės“ (87 %). Pasiskirstymas pagal prekių kiekį grupėse tolygus.

Vaizdų skaičius pagal prekių grupę



(a)

Prekių skaičius pagal prekių grupę



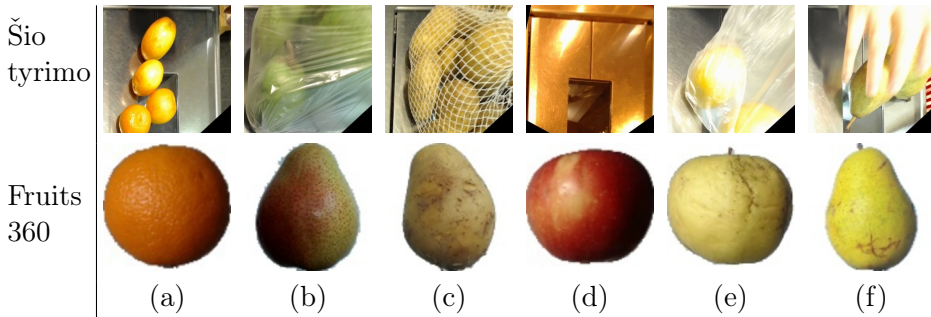
(b)

59 paveikslas: Vaizdų kiekis pagal prekių grupę (a) ir prekių skaičius pagal prekių grupę (b).

## Egzistuojančios maisto prekių vaizdų aibės

Vaizdų aibės, kurios atspindi realiai matomus vaizdus savitarnos kasose, viešai nėra prieinamos arba jų nėra. Kai kuriose aibėse, kurių vaizdai panašūs į savitarnos kasų vaizdus, didelę dalį sudaro prekės su brūkšniniais kodais. Kitose vaizdai surinkti sterilioje aplinkoje, kur prekės gerai matomos, neuždengtos kūno dalimis, neįdėtos į plastikinius maišelius.

**Fruits 360** [7] iš viešai prieinamų aibių panašiausia į savitarnos kasų vaizdus; prekės aibėje parduodamos be brūkšninio kodo. Aibę sudaro 65 000 vaizdų ir 95 prekės. Vis tik „Fruits 360“ [7] apima tik mažą dalį prekių be brūkšninių kodų - tik tam tikras kategorijos „Vaisiai ir daržovės“ prekes be daržovių. Aibėje nėra saldinių, džiovintų vaisių, sausainių ir pan. 60 pav. palyginti šio tyrimo metu surinktos aibės ir „Fruits 360“ [7] vaizdai. 22 lentelėje statistiškai palygintos šio tyrimo metu surinktos ir išvalytos bei „Fruits 360“ [7] aibės.



60 paveikslas: Vaizdų palyginimas šio tyrimo metu surinktos aibės (viršuje) ir „Fruits 360“ (apačioje).

22 lentelė: Šio tyrimo metu surinkta aibė ir „Fruits 360“.

	Šis tyrimas	„Fruits 360“
Klasės	194	131
Iš viso vaizdų (prieš balansavimą)	~26 600	~90 000
Vaizdų klasėje: Min./Vid./Maks.	3 / 137 / 3111	396 / 690 / 1312

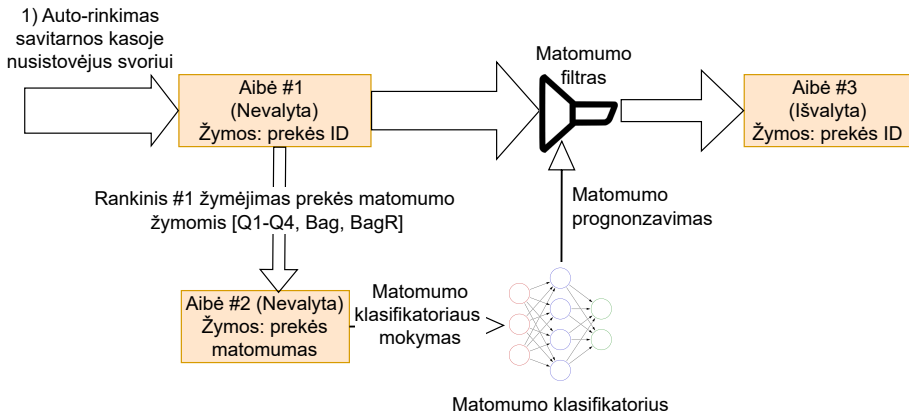
**RPC: A Large-Scale Retail Product Dataset** [35] aibę sudaro 83 000 vaizdų ir 200 prekių. Vaizdai surinkti sterilioje aplinkoje. **MVTec Densely Segmented Supermarket Dataset (MVTec D2S)** [34] aibę sudaro 21 000 vaizdų ir 60 prekių, t. y. maža dalis savitarnos kasose parduodamų prekių be brūkšninio kodo. **GroZi-120** [9] aibę sudaro 12 000 vaizdų ir 120 prekių, surinktų dalinai iš interneto. Keletą mažesnių aibių straipsniuose [105], [106] sudaro iki 10 000 vaizdų.

## Duomenų aibės sudarymas mašininiam mokymui

23 lentelėje surašytos vaizdų aibės, naudotos šiame tyrime. 61 pav. paaikškinta aibių sukūrimo eiga.

23 lentelė: Duomenų aibės naudotos atpažinimo eigoje.

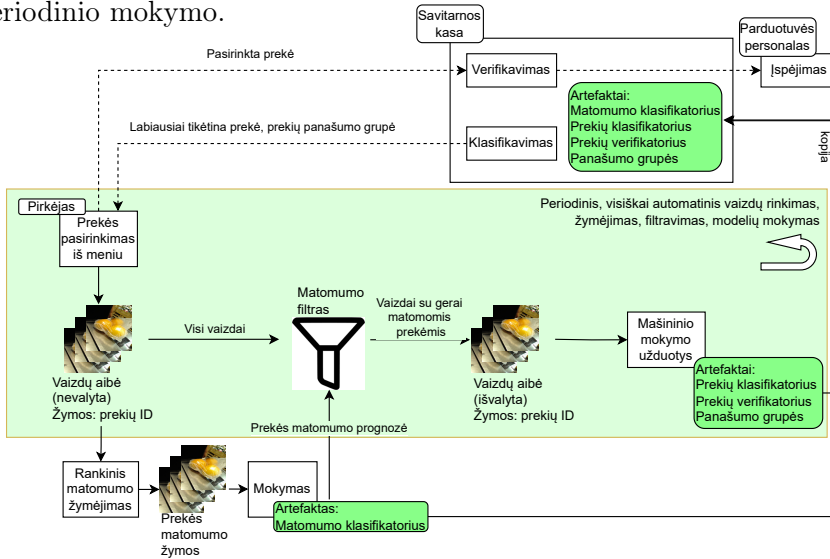
Nr.	Kaip gauta	Žymos	Kam naud.	Vaizdų skaičius	
				Original.	Balans.
#1	Automatiškai kasoje	Prekės ID	-	26,6 tūkst.	-
#2	Rankomis sužymėta dalis #1	Matomumo kategorija	Matomumui klasifikuoti	6 tūkst.	11,5 tūkst.
#3	Filtruota #1 naudojant matomumo klasifikatorių	Prekės ID	Prekėms klasifikuoti, verifikuoti, grupuoti	18,1 tūkst.	500 tūkst.



61 paveikslas: Tinkamai mokyti duomenų aibei paruošti (#3) iš automatiškai surinktos aibės (#1), kurioje didelė dalis vaizdų yra prastos kokybės, reikalinga pagalbinė aibė (#2), kurioje vaizdai sužymėti prekių matomumo žymomis.

# Metodai

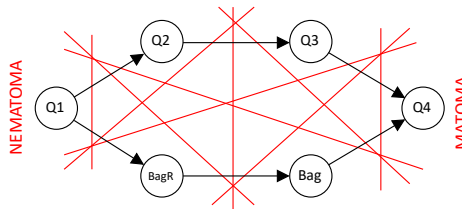
Periodiškai vykdoma prekių atpažinimo proceso automatizuota dalis (pažymėta žalsvai 62 pav.) apima vaizdų rinkimą, žymėjimą pagal klientų pasirinktas prekes, prastai matomų prekių vaizdų šalinimą, modelių mokymą. Schemos apačioje matomumo klasifikatorius mokomas naudojant matomumo žymas. Jo pagrindinės funkcijos yra filtruoti automatiškai surinktą duomenų rinkinį prieš mokymą ir įvertinti prekės matomumą prieš klasifikavimą. Matomumo klasifikatorius mažiau priklauso nuo prekių sąrašo ar išvaizdos pokyčių, todėl nereikalauja periodinio mokymo.



62 paveikslas: Atpažinimo proceso įgyvendinimas gamyboje.

## Vaizdo tinkamumas prekei atpažinti

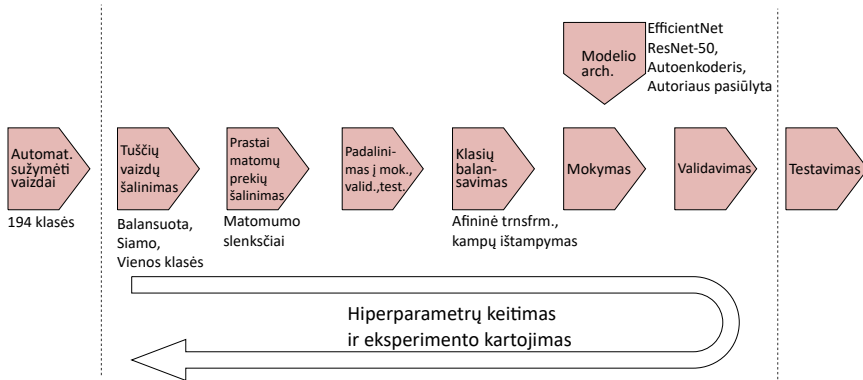
Turint duomenų rinkinį, sužymėtą 6 matomumo žymomis (#2 23 lentelėje), ir siekiant rasti geriausią atskyrimo slenkstį tarp matomų ir nematomų prekių vaizduose, duomenų žymos priskirtos visais įmanomais būdais [Matoma; Nematoma] kategorijoms, kaip parodyta 63 pav.



63 paveikslas: Galimi prekės matomumo žymų grupavimo būdai (raudonos linijos).

## Prekių klasifikavimo etapai, architektūros parinkimas

Siekiant klasifikuoti prekes savitarnos kasų nuotraukose, buvo nagrinėjami šie klausimai: 1) kaip prekės matomumo klasifikatoriai veikia prekių klasifikavimo tikslumą? 2) kokia technika geriausia tuščioms nuotraukoms filtruoti? 3) kaip nustatyti abiejų žingsnių reikalingumą atliekant abliacijos tyrimą? 4) kaip nustatyti optimalią architektūrą palyginant gerai žinomas neuroninių tinklų architektūras su šiame tyrime sukurtą ir savitarnos kasų nuotraukoms optimizuota architektūra?



64 paveikslas: Klasifikavimo eksperimento eiga.

Eksperimentas (64 pav.) apima du nestandartinius etapus: tuščių vaizdų ir prasto prekių matomumo vaizdų pašalinimą. Norint nustatyti šių etapų naudingumą, atlikti abliacijos tyrimai, pašalinant kiekvieną iš etapų. Šie du etapai galėtų būti įgyvendinti vienu klasifikatoriumi, nes prekių matomumo klasifikatoriai buvo mokomi naudojant kai kurias tuščias nuotraukas (Q1 kategorija). Vis tik tuščioms nuotraukoms atskirti tinka ir paprastesnės, galimai didesnio tikslumo technikos.

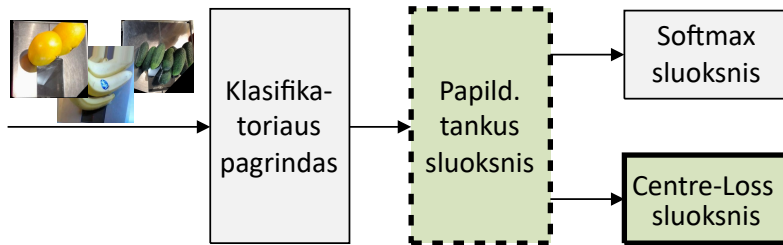
Modelio architektūra atrinkta tiriant keturias architektūras: „ResNet-50“ [111], „EfficientNet“ [18], autoenkoderiais pagrįsta ir šiame tyrime pasiūlyta architektūra. „ResNet-50“ pasirinkta dėl gebėjimo puikiai prisitaikyti prie mažesnių duomenų aibių naudojant iš anksto apmokytus modelius [118], [119]. „EfficientNet“ pasirinkta dėl sėkmės klasifikuojant „ImageNet“ ir palyginus mažo dydžio (5,3 M parametrų), tinkamo naudoti mažo galingumo savitarnos kasose, vaizdus. Autoenkoderiais pagrįstos architektūros labai tinka save prižiūrintiems mokymams, kai trūksta sužymėtų duomenų. Sudėtingesnės „EfficientNet“ versijos B1-B7 ir kitos architektūros, kaip VGG (133 M parametrų), nebuvo nagrinėjamos dėl ilgos mokymo trukmės ir netinkamumo mažo galingumo savitarnos įrenginiams.



## Pirkėjo pasirinkimo verifikavimas

Norint patikrinti, ar pirkėjo pasirinkimas atitinka prekę ant svarstyklių, buvo siekiama: 1) palyginti klasės prototipu pagrįstų patikros technikų („Centre-Loss“ [2], „Proxy-NCA“ [3]) tikslumą su plačiai verifikavimui naudojamomis kontrastinio vaizdų palyginimo technikomis („Siamo“ [28], „Triplet“ [54]) naudojant savitarnos kasų vaizdų aibę ir panašias vaizdų aibes; 2) ištirti, ar euklidiniam atstumui alternatyvūs tipai galėtų pagerinti vaizdų panašumo matavimą.

„Centre-Loss“ architektūra turi dvilypę tikslo funkciją ir dvi išvestis (65 pav.): „Softmax“ ir „Centre-Loss“ sluoksni, kaip nurodyta straipsnyje [2]. Nors verifikuoti naudojamas atstumas nuo klasės centro gaunamas iš „Centre-Loss“ sluoksnio, „Softmax“ sluoksnis reikalingas mokymo metu klasių atskiriamumui užtikrinti.



65 paveikslas: „Centre-Loss“ modelio architektūra.

Siekiant išaiškinti tinkamą atstumo tipą, tyrimo metu buvo eksperimentuojama ne tik su euklidiniu, bet ir su kitais atstumo tipais: manhatano, minkovskio ir kosinuso. Minkovskio atstumo  $p$  reikšmės buvo ribojamos diapazone [1,4]. Kiti atstumo tipai, tokie kaip Hammingo, Jaccardo ir Dice, netirti dėl savo netinkamumo vertinti nebinarinius atstumus.

Mokymo metu „Centre-Loss“ sluoksnyje klasių centrai atnaujinami, kaip parodyta 17 lygtyje.

$$Centras = Centras + \alpha \times (Aktivacijos - Centras), \quad (17)$$

kur:

$Centras$  yra tam tikros klasės centras  $\in \mathbb{R}^{cnt}$ ,

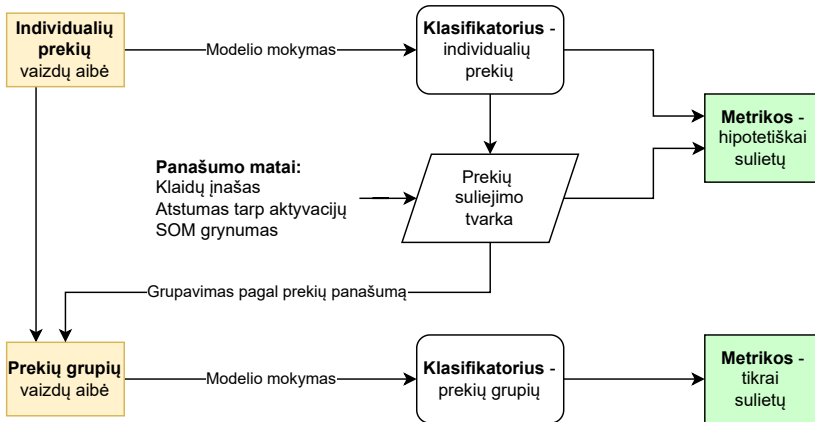
$Activations$  yra tam tikros klasės duomenų aktyvacijos  $\in \mathbb{R}^{cnt}$ ,

$\alpha$  yra mokymosi koeficientas,

$cnt$  yra neuronų skaičius prieš „Centre-Loss“ esančiame sluoksnyje.

## Prekių grupavimas pagal panašumą

Siekiant sugrupuoti prekes taip, kad klasifikuoti į grupes būtų galima kuo tiksliau, buvo bandoma: 1) nustatyti, kaip reikia lyginti vaizdų klases, siekiant maksimalaus grupių klasifikavimo tikslumo, 2) klasifikuoti į grupes (pagal panašumą) naudojant duomenis, sužymėtus (a) grupių žymomis ar (b) atskirų prekių žymomis, priskiriant labiausiai tikėtinos prekės grupę. Šio tyrimo išvados taikomos sritims, kur 1) vaizdų klasifikavimo tikslumas neatitinka minimalių reikalavimų ir 2) yra naudinga spėti panašių klasių grupę. Vienas iš pavyzdžių – mažmeninės prekybos savitarnos kasų prekių pasirinkimo meniu.



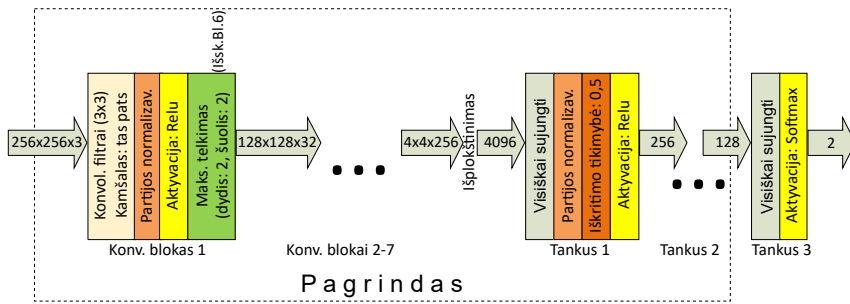
66 paveikslas: Klasifikatorių palyginimas apmokius naudojant grupių žymas ir individualių prekių žymas.

Grupavimo tyrimo eiga pavaizduota 66 pav. Žingsnis *Klasifikatorius* – *individualios prekės* yra prekių klasifikatorius. Tarpklasinis panašumas nustatomas naudojant tris skirtingus metodus: „Klaidos indėlis klasifikavimo matricioje“, „Vidutinis atstumas tarp aktyvacijų“ ir „SOM grynumas“. Remiantis šiais metodais, sukuriama aglomeracinio klasterizavimo schema, nurodanti klasių susiliejimo tvarką pagal panašumą (*Produktų susiliejimo tvarka*). *Klasifikatoriai* - *prekių grupės* prognozuoja vaizdo priklausymą tam tikrai prekių grupei. Visi klasifikatoriai yra vienodos architektūros, išskyrus paskutinį „Softmax“ sluoksnį (skiriasi tik neuronų skaičiumi). Grupių klasifikatoriai yra identiški atskirų prekių klasifikatoriams. *Rodikliai* - *individualios prekės* pagal prognozuojamą prekę nustato prekių grupę ir išmatuoja klasifikavimo į grupes rodiklius (tikslumas, F-rodiklis). *Rodikliai* - *sulietos prekės* išmatuoja klasifikavimo į prekių grupes rodiklius.

# Rezultatai

## Matomų ir nematomų prekių atskyrimas

Galutinė tinklo architektūra pavaizduota 67 pav. Eksperimentai su mažiau konvoliucinių ir tankių sluoksnių lėmė didesnę mokymo aibės klaidų skaičių; daugiau bet kurios rūšies sluoksnių nepažėjo sumažinti mokymo aibės klaidų.



67 paveikslas: Galutinė modelio architektūra.

Modelių kokybė buvo vertinama pagal tai, kaip gerai jie atskiria vaizdus su geriau matomomis prekėmis nuo vaizdų su blogiau matomomis. 24 lentelėje pateikiamos metrikos visiems leistiniams slenksčiams.

24 lentelė: Gerai ir blogai matomų prekių atskyrimo F-rodiklis ir kitos metrikos, leistiniais būdais grupuojant matomumo žymas.

Matomumo žymos		F-rodiklis	Tikslumas	Preciziškumas	Atmintis
Matoma	Nematoma				
Q2, Q3, Q4, Bag, BagR	Q1	<b>0,906</b>	0,874	0,931	0,883
Q2, Q3, Q4, Bag	Q1, BagR	0,895	0,86	0,897	0,892
Q2, Q3, Q4	Q1, Bag, BagR	0,854	0,826	0,839	0,869
Q3, Q4, Bag, BagR	Q1, Q2	0,793	0,78	0,707	0,903
Q3, Q4, Bag	Q1, Q2, BagR	0,781	0,794	0,732	0,837
Q3, Q4	Q1, Q2, Bag, BagR	0,723	0,752	0,606	0,895
Q4, Bag, BagR	Q1, Q2, Q3	0,667	0,762	0,581	0,784
Q4, Bag	Q1, Q2, Q3, BagR	0,661	0,782	0,581	0,766
Q4	Q1, Q2, Q3, Bag, BagR	0,565	0,757	0,437	0,8

## Prekių klasifikavimas

Klasifikavimo tyrimo rezultatai pateikti 25 lentelėje. Pasiūlytos eigos prekių klasifikavimo tikslumas  $80,5 \% \pm 1,2 \%$  palygintas su tikslumu naudojant kitas architektūras ir nenaudojant vieno iš filtravimo etapų: tuščių vaizdų ar vaizdų su prastai matomomis prekėmis.

25 lentelė: Pasiūlytos atpažinimo eigos ir alternatyvių architektūrų bei abliacijos tyrimų tikslumo lyginimas.

Tikslumas	Pasiūlyta eiga	Alternatyvios architektūros			Abliacijos tyrimai	
		EfficientNet B0	Resnet-50	Auto-enkoderis	Tušti pašalinti	Nematomi pašalinti
<b>Vidurkis</b>	<b>80,5 %</b>	80,2 %	72,9 %	58,1 %	79,1 %	78,8 %
<b>Dispersija</b>	<b>1,2 %</b>	2,4 %	1,9 %	2,5 %	0,8 %	2,0 %

Panašiausio viešo duomenų rinkinio „Fruits 360“ [7] klasifikavimo rezultatai naudojant autoriaus architektūrą palyginti su kitų autorių darbais 26 lentelėje. Šio tyrimo pasiūlyta eiga parodė geresnę arba tokį patį tikslumą, kaip ir kitų autorių. Tai rodo šio tyrimo metodo bendrinamumą kitiems palyginamiems duomenų rinkiniams.

26 lentelė: Lyginimas su kitų autorių darbais naudojant „Fruits 360“ aibę.

Metodas	Tikslumas, %
[17]	95,7
[16]	98,7
[15]	99,6
<b>Pasiūlyta eiga</b>	<b>99,6</b>

Klasifikavimo rezultatai naudojant šio tyrimo metu surinktą duomenų aibę ir „Fruits 360“ palyginti 27 lentelėje.

27 lentelė: Tikslumas: šio tyrimo metu surinkta duomenų aibė ir „Fruits 360“. \*atlikta 10 eksperimentų.

Duomenų aibė	Tikslumas, %
Autoriaus surinkta	80,0–83,4*
„Fruits 360“	99,6

## Pirkėjo prekės pasirinkimo patikrinimas

Pagrindinis rezultatas yra dviejų skirtingų klasės tikrinimo metodų - vaizdų palyginimu („Siamo“, „Triplet“) ir klasės prototipu („Centre-Loss“, „Proxy-NCA“) pagrįstų - tikslumo palyginimas. 28 lentelėje pateikta ROC AUC ir EER visiems tirtiems tinklų tipams. Pažymėtina, kad visų tipų tinklų tikslumo skirtumai yra nežymūs. Tyrime išnagrinėtų įvairių

28 lentelė: Verifikavimo ROC AUC ir EER pagal tinklo tipą.

Tinklo tipas	ROC AUC	EER
Proxy-NCA	<b>0,985</b>	<b>0,054</b>
Siamo	0,981	0,063
Triplet	0,980	0,060
Centre-Loss	0,979	0,073

atstumų tipų poveikis tikslumui pateiktas 29 ir 30 lentelėse. „Siamo“ ir „Triplet“ tinkluose visi atstumų tipai rodė panašų rezultatą, išskyrus šiek tiek blogesnę kosinuso atstumą „Triplet“ tinkle. „Centre-Loss“ modelyje dauguma atstumų tipų (kosinuso, manhatano, euklido ir minkovskio) rodė panašius rezultatus, nors su aukštesnėmis minkovskio  $p$  reikšmėmis ( $p=3, 4$ ) rezultatas prastesnis.

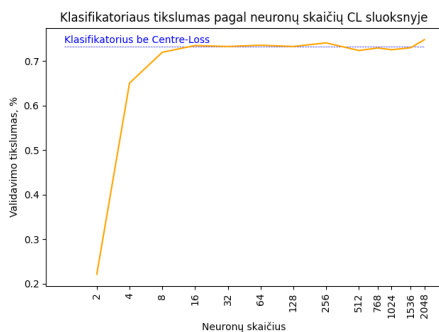
29 lentelė: Verifikavimo ROC AUC pagal atstumo tipą.

Atstumo tipas	Tinklo tipas			
	Proxy-NCA	Centre-Loss	Siam.	Triplet
Manhatano	<b>0,985</b>	0,962	<b>0,981</b>	0,971
Euklido	0,984	<b>0,979</b>	0,980	<b>0,980</b>
Minkovskio ( $p=3$ )	0,984	0,948	<b>0,981</b>	<b>0,980</b>
Minkovskio ( $p=4$ )	0,983	0,839	0,980	0,979
Kosinuso	0,942	0,961	<b>0,981</b>	0,913

30 lentelė: Verifikavimo EER pagal atstumo tipą.

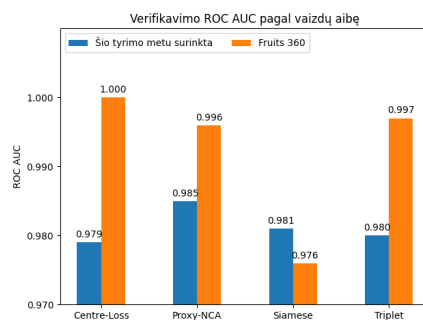
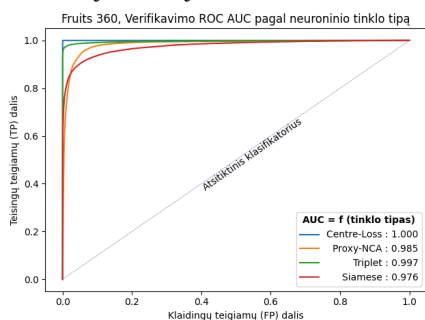
Atstumo tipas	Tinklo tipas			
	Proxy-NCA	Centre-Loss	Siam.	Triplet
Manhatano	0,054	0,099	0,063	0,078
Euklido	<b>0,047</b>	<b>0,073</b>	0,065	<b>0,060</b>
Minkovskio ( $p=3$ )	0,050	0,121	0,063	0,061
Minkovskio ( $p=4$ )	0,051	0,241	0,064	0,064
Kosinuso	0,104	0,095	<b>0,062</b>	0,116

Verifikavimo „Centre-Loss“ architektūra (65 pav.) turi „Softmax“ sluoksnį, kuris gali būti naudojamas produktams klasifikuoti. Klasifikatorius (be „Centre-Loss“) pasiekė 73,2 % tikslumą validavimo aibėje (mėlyna linija 68 pav.), tačiau su „Centre-Loss“ (oranžinė linija) prekių klasifikavimo tikslumas išlieka maždaug toks pat. Tai rodo, kad tas pats tinklas gali atlikti dvi užduotis neprarandant tikslumo: prekių klasifikavimą ir pirkėjo pasirinktos prekės verifikavimą.



68 paveikslas: Klasifikavimo tikslumo priklausomybė nuo neuronų skaičiaus prieš „Centre-Loss“ esančiame sluoksnyje (oranžinė linija); palyginimas su klasikiniu klasifikatoriumi be „Centre-Loss“ (mėlyna linija).

Metodo tikslumas įvertintas naudojant „Fruits 360“ duomenų aibę [7]: kiekvieno tinklo tipo modeliai („Centre-Loss“, „Proxy-NCA“, „Siamese“ ir „Triplet“) apmokyti su „Fruits 360“. „Fruits 360“ prekės patikrinimo ROC kreivės pateiktos 69a pav., o ROC AUC palyginimas tarp „Fruits 360“ ir autoriaus surinktos duomenų aibės pateiktas 69b pav. Dauguma metodų parodė geresnę veikimą su „Fruits 360“ dėl švarių, sterilioje aplinkoje surinktų vaizdų.



(a) „Fruits 360“ aibė, patikrinimo ROC pagal tinklo tipą. (b) Patikrinimo rezultatai naudojant šio tyrimo metu surinktą aibę ir „Fruits 360“. [7]

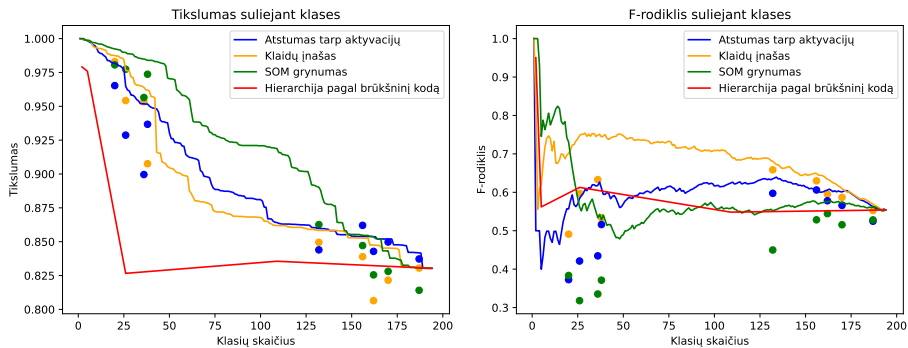
69 paveikslas: Verifikavimo rezultatų palyginimas naudojant „Fruits 360“ [7] ir šio tyrimo metu surinktą aibę.

## Vaizdų klasifikavimas į panašių prekių grupes

Šio tyrimo rezultatai:

- nustatytas efektyviausias būdas klasių panašumui nustatyti. Remiantis panašumu, klasės sujungiamos į „klasių grupes“ ir matuojamos klasifikavimo į grupes metrikos;
- klasifikatoriai, apmokyti naudojant atskirų klasių žymas, palyginti su klasifikatoriais, apmokytais naudojant klasių grupių žymas, kur klasės sujungtos į grupes pagal panašumą.

70 pav. pateikiamas klasifikavimo metrikų lyginimas: tikslumas (kairėje) ir F-rodiklis (dešinėje). Taškai rodo klasifikatorių, apmokytų naudojant grupių žymas, metrikas; kreivės - klasifikatorių, apmokytų naudojant individualių prekių žymas, metrikas. Pastarieji klasifikatoriai dažniausiai pranoksta pirmuosius, t. y. norint tiksliai prognozuoti grupę, klasifikatorius reikia mokyti naudojant individualių prekių žymas. Grafiko spalvos reiškia skirtingus prekių panašumo matavimo būdus. „SOM grynumas pagerėjimo“ būdas paprastai pranoksta kitus, kai svarbu optimizuoti tikslumą (pvz., mažmeninės prekybos savitarnos kasose, kur klasių pasiskirstymas yra netolygus). „Klaidos įnašo klasifikavimo matricioje“ būdas paprastai pranoksta kitus, kai svarbiausia yra optimizuoti F-rodiklį.



70 paveikslas: Klasifikavimo į panašių klasių grupes metrikos: taškai rodo klasifikatorių, apmokytų naudojant grupių žymas, metrikas, o kreivės - klasifikatorių, apmokytų naudojant individualių prekių žymas, metrikas.

## Bendrosios išvados

1. Dviejų klasių tuštumo klasifikatorius tuščių vaizdų šalinimo etape pašalino tuščius vaizdus su didžiuliu 98,8 % tikslumu. Kiti tuštumo metodai („Siamo“, OCC) pasirodė prasčiau. Vaizdai su geru ir blogu prekės matomumu atskirti naudojant eksperimentiškai nustatytą atskyrimo slenkstį, kurio F-rodiklis yra 90,6 %.
2. Eksperimentai parodė, kad CLAHE yra veiksmingiausias būdas sumažinti kasų apšvietimo skirtumus. Didesnis variabilumas augmentavus vaizdus naudojant ir afininę, ir perspektyvinę transformaciją 1,3 %-2,8 % pralenkė tik afines transformacijas ir 1,3 %-3,9 % – tik perspektyvines transformacijas. Naudojant tris perspektyvas, rezultatai buvo geresni vidutiniškai 1,1 % nei naudojant vieną perspektyvą.
3. Produktų klasifikavimo eksperimentai parodė, kad pasiūlyta 7 konvoliucinių ir 3 tankių sluoksnių neuroninio tinklo architektūra (tikslumas 80,5 %  $\pm$  1,2 %), apmokyta naudojant savitarnos kasų vaizdus, pralenkė gerai žinomas architektūras: „EfficientNet B0“ (80,2 %  $\pm$  1,2 %), „ResNet-50“ (58 %  $\pm$  2,5 %). Pasiūlytos architektūros našumas pablogėja pašalinus bet kokį konvoliucinį ar tankių sluoksnį ir nebe gerėja pridendant sluoksnius. Siūlomoms architektūroms tikslumas naudojant sintetinį rinkinį „Fruits 360“ pasiekia panašų 99,6 % tikslumą kaip ir kitų autorių darbuose.
4. Klasės patikrinimo tikslumas klasės prototipu („Proxy-NCA“, „Centre-Loss“) ir vaizdų palyginimu („Siamo“, „Triplet“) paremtais metodais buvo panašus: „Proxy-NCA“ ROC AUC 0,985, „Centre-Loss“ 0,979 ir „Siamo“ 0,981, „Triplet“ 0,980; „Proxy-NCA“ EER 0,047, „Centre-Loss“ 0,073, palyginti su „Siamo“ 0,063 ir „Triplet“ 0,060. Euklidinis atstumas tikslo funkcijose tikslumu nenusileido kitiems atstumo tipams (manhatano, minkovskio, kosinuso), nors gretimos minkovskio  $p$  reikšmės ( $p = 1$  manhatano,  $p = 3$ ) buvo šalia. Taikant „Centre-Loss“ metodą, naudojant Euklido atstumą, pasiektas 0,979 ROC AUC, o naudojant gretimas minkovskio  $p$  vertes buvo 0,962 ( $p = 1$  manhatano) ir 0,948 ( $p = 3$ ). Eksperimentai su „Centre-Loss“ architektūra atskleidė, kad priešpaskutinis sluoksnis labiausiai tinka klasės prototipui formuoti. Priešpaskutinio sluoksnio dydis prisotinamas priklausomai nuo minkovskio  $p$  reikšmės: didesnėms minkovskio  $p$  reikšmėms prisotinti reikia daugiau neuronų. Pasiūlyta „Centre-Loss“ tikslo funkcijos modifikacija, siekiant padidinti atstumą tarp klasių centrų, teigiamo rezultato nedavė.



5. Tas pats „Centre-Loss“ architektūros neuroninis tinklas yra tinkamas dviem skirtingoms užduotims: prekėms klasifikuoti ir pasirinktai prekei patikrinti. Taip yra dėl netikėto atradimo, kad dvilypės tikslo funkcijos naudojimas nepablogino klasifikavimo tikslumo:  $73,2 \% \pm 0,8 \%$  (palyginti su  $73,2 \%$  be „Centre-Loss“).
6. Produktų grupavimo pagal panašumą eksperimentai, kad klasifikavimo į grupes tikslumas būtų didžiausias, parodė, kad „SOM grynumo“ būdas paprastai pagerino tikslumą  $1,2 \% \pm 1,5 \%$ , nors jo pranašumas buvo mažesnis esant didesniam grupių skaičiui. „Klaidų klasifikavimo matricoje“ panašumo metodas pagerino F-rodiklį  $8,9 \% \pm 7,7 \%$ . Geriausias prekių panašumo nustatymo būdas priklauso nuo grupių skaičiaus ir naudojamos metrikos (tikslumo ar F-rodiklio). Klasifikatoriai, apmokyti naudojant individualių prekių žymas, pranoksta klasifikatorius, apmokytus naudojant grupių žymas: individualių klasifikatorių tikslumas buvo  $1,8 \% \pm 1,5 \%$ , o F-rodiklis buvo  $10,3 \% \pm 9,0 \%$  didesnis nei grupinių klasifikatorių tikslumas.

## **About the author**

Bernardas Čiapas was born in Klaipėda, Lithuania, in 1978. In 1996, he graduated from Vytautas the Great High School in Klaipėda. He obtained a B.Sc. in Informatics in 2000 at Kaunas University of Technology and an M.Sc. in Informatics in 2002 at California State University, Fullerton. From 2019 to 2023, he studied in the Ph.D. program in Informatics at Vilnius University.

## **Apie autorių**

Bernardas Čiapas gimė Klaipėdoje, Lietuvoje, 1978 metais. 1996 metais jis baigė Vytauto Didžiojo gimnaziją Klaipėdoje. 2000 metais Kauno technologijos universitete jis įgijo informatikos bakalauro laipsnį, o 2002 metais Kalifornijos valstijos universitete Fullertone įgijo informatikos magistro laipsnį. Nuo 2019 iki 2023 metų jis studijavo Vilniaus universiteto informatikos doktorantūros programoje.

## NOTES

Bernardas Čiapas

**Barcodeless Food Products Recognition for Retail  
Self-checkout Service**

DOCTORAL DISSERTATION

Natural Sciences

Informatics N 009

Thesis Editor: Zuzana Šiušaitė

**Maisto produktų be brūkšninio kodo atpažinimas savitarnos  
kasose**

DAKTARO DISERTACIJA

Gamtos mokslai

Informatika N 009

Santraukos redaktorė: Jorūnė Rimeisytė-Nekrašienė

Vilniaus universiteto leidykla  
Saulėtekio al. 9, III rūmai, LT-10222 Vilnius  
El. p.: [info@leidykla.vu.lt](mailto:info@leidykla.vu.lt), [www.leidykla.vu.lt](http://www.leidykla.vu.lt)  
Tiražas 20 egz.