Counting with Limited Supervision

Michael A. Hobley

Keble College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas Term 2023

Michael A. Hobley
Keble College

Doctor of Philosophy

Michaelmas Term 2023

Counting with Limited Supervision

Abstract

Counting is among the first abstract analysis tasks that we learn. It is the most fundamental way we can quantitatively understand data. From an early age, people are able to accurately count objects with minimal direction, even when the type of object to be counted is completely unknown. While prior machine learning-based methods have addressed the problem of counting previously unseen kinds of objects, known as class-agnostic counting, they have all required both user input during deployment in the form of exemplar images to define the type to be counted and the locations of every object during training to act as supervision.

In this thesis, we aim to further automated counting methods with the goal of replicating the human ability to perform completely naive counting. To achieve this, we recognise that counting is composed of, at its heart, two different tasks: instance finding and repetition recognition. We explore these problems first in isolation and then together. Within this exploration, we introduce various paradigms to the field of class-agnostic counting including exemplar-free counting, weak supervision, simultaneous multi-class counting, and more abstract concepts which we believe should be considered such as valid-but-unknown counts and the distinction between intrinsic and non-intrinsic tasks.

Over the course of this thesis, we propose three methods which demonstrate that class-agnostic counting can be achieved with less information than previously postulated during both training and deployment. Specifically, we show that large sets of high dimensional data can be clustered flexibly and accurately using only relational pairwise labels, that robust counting can be achieved on novel classes without the requirement of exemplar images to define type during training or inference, and additionally that under certain conditions, such a method can be trained using only image-wise scalar count supervision.

We also propose two datasets to facilitate training and reliably evaluate the performance of said methods alongside other contemporary work. Together, these contributions create a strong base for counting in settings with limited supervision and minimal user input.

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Michael A. Hobley, Keble College

For those who accompanied me

when I needed it most

Acknowledgements

I first would like to express my greatest thanks to my supervisor Professor Victor Prisacariu, who not only convinced me to undertake this DPhil but supported me throughout it. I would also like to thank Professor Felix Leach and Professor Stephen Payne who have gone above and beyond in guiding me through my time in Oxford. Thank you to my examiners, Professor Minh Hoai Nguyen and Professor Ioannis Havoutis. I appreciate the time they have given to provide such valuable and thoughtful feedback on my research.

I am very thankful for my friends and collaborators in AVL, particularly Henry, Ryan, and Theo, who offered me advice, feedback, and incalculable amounts of tech support throughout the years. I would like to express my thanks to Jeremy for years of proof-reading my work and to Matt for his dedication and insightful feedback. Thanks to Alexey Pajitnov, without whom this DPhil would have been completed in much less time.

Thank you to my friends, teammates, colleagues and coaches. Thanks especially to Ali, Beth, Cat, Costi, Ellie, Emily, Hannah, Issy, Julia, the Katies, Liv, Mary, Molly, Thomas, and all the friends I made in KCBC and OUWBC, who not only made this DPhil possible but made my time doing it more enjoyable.

Finally, I would like to thank my brother, John, and my parents, Rosemary and Philip. I will be eternally grateful for the support and love they have given me throughout my life.

Contents

1 Introduction **1**
- 1.1 Objective . . . 1
- 1.2 Thesis Structure and Contributions . . . 3
- 1.3 Publications and Public Releases . . . 6

2 Background **7**
- 2.1 Network Architectures . . . 8
  - 2.1.1 Convolutional Neural Networks . . . 8
  - 2.1.2 Attention and Vision Transformers. . . . 12
- 2.2 Clustering . . . 14
  - 2.2.1 Formulation . . . 15
  - 2.2.2 K-means . . . 15
  - 2.2.3 DBSCAN . . . 17
  - 2.2.4 Mean shift . . . 19
  - 2.2.5 Spectral clustering methods . . . 21
  - 2.2.6 Deep Clustering . . . 21
  - 2.2.7 Side-Information . . . 24
- 2.3 Counting . . . 26
  - 2.3.1 Class-specific counting . . . 26
  - 2.3.2 Detection-based Counting . . . 27
  - 2.3.3 Classification-based Counting . . . 28
  - 2.3.4 Regression-based Counting . . . 28
  - 2.3.5 Weakly-supervised Counting . . . 29
  - 2.3.6 Class-agnostic Counting . . . 31
  - 2.3.7 Attention in Counting . . . 34
  - 2.3.8 Count Ambiguity . . . 36
  - 2.3.9 Conclusion . . . 38

3 Our Datasets **40**
- 3.1 FSC-147 . . . 41
  - 3.1.1 Limitations . . . 41
  - 3.1.2 Errors . . . 42
- 3.2 Proposed Solution (FSC-133) . . . 53
- 3.3 Multi-Class Class-Agnostic Counting Dataset . . . 53
  - 3.3.1 Multi-Class Counting Problem Definition . . . 55
  - 3.3.2 Multi-Class Class-Agnostic Synthetic Dataset . . . 56

3.3.3 Conclusion 64

4 Dataset-Agnostic Task-Specific Clustering Using Side-Information 66
4.1 Introduction 66
4.2 Method 70
4.2.1 Clustering Problem Definition 70
4.2.2 Mean Shift Kernel 70
4.2.3 DMS Kernel 72
4.2.4 DMS Algorithm 74
4.2.5 Training with Side-Information 78
4.3 Experiments 79
4.3.1 Datasets 79
4.3.2 Intrinsic vs Non-intrinsic Tasks 80
4.3.3 Metrics 81
4.3.4 Experiment Training 83
4.3.5 Results 83
4.3.6 Ablation 84
4.4 Conclusion 89

5 Exemplar-Free Weakly-Supervised Single-Class Class-Agnostic Counting 91
5.1 Introduction 91
5.2 Method 94
5.2.1 Self-supervised knowledge distillation. 94
5.2.2 Count Regression 96
5.2.3 Tiling Augmentation 98
5.3 Experiments 99
5.3.1 Architecture 99
5.3.2 Training 100
5.3.3 Evaluation Metrics 100
5.4 Results 101
5.4.1 Benchmark Methods 101
5.4.2 FSC-147 and FSC-133. 102
5.4.3 Cross-Dataset Generalisability 103
5.4.4 Feature Visualisation 106
5.4.5 Ablation Studies 109
5.4.6 Failure Cases and Limitations 112
5.5 Conclusion 116

6 Exemplar-Free Multi-Class Class-Agnostic Counting 117
6.1 Introduction 117
6.2 Related Work 119
6.3 Method 120
6.3.1 Density Map Regression 121
6.3.2 Matching 122
6.3.3 Example Discovery 124

6.3.4 Implementation 125
6.4 Results 130
6.4.1 Benchmarking Methods 130
6.4.2 Benchmarking Metrics 131
6.4.3 MCAC 132
6.4.4 Applicability to FSC-133/147/LVIS 133
6.5 Ablations 139
6.5.1 Validating Our Loss 139
6.5.2 Validating Our Matching Scaling 140
6.5.3 Validating Our Number of Predictions 141
6.5.4 Validating Valid-But-Unknown Predictions 143
6.6 Conclusion 144

7 Conclusion **145**
7.1 Contributions 146
7.2 Future Work 148
7.3 Closing Remarks 149
Bibliography 150

1

Introduction

1.1 Objective

One of the most fundamental ways we can quantitatively analyse the world around us is through counting. It is one of the first abstract tasks we learn as children and it is simple to explain. To count is in effect to answer the question 'How many instances are there of a given type?'. Even though the description of the task is simple and it is easy for humans to perform, it has broad and impactful applications in a variety of industries, including ecology with animal counting [1, 2], urban planning with vehicle [3–5] and crowd counting [4, 6–10], and various medical and biological disciplines that require cell counting [11, 12]. These industries would all benefit from being able to gather more accurate and reliable data in a shorter amount of time with less manual labour. While modern machine learning systems are increasingly able to complete more and more complex tasks, they are still

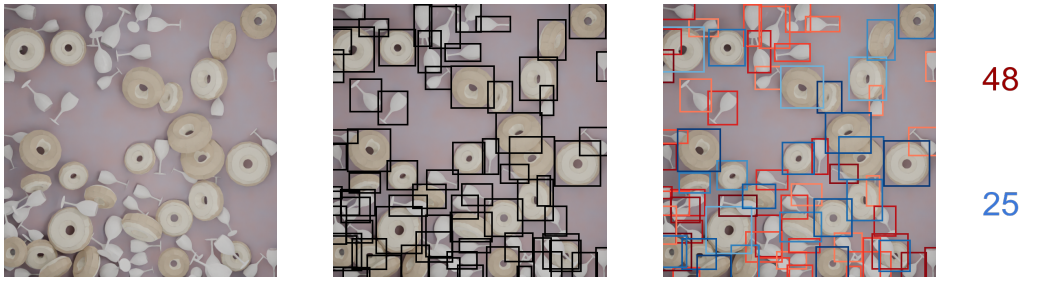


Figure 1.1: An example of the two counting stages on a multi-class image from our dataset, MCAC. Areas that could be 'worth counting' are located and compared to generate the final counts.

flawed in their ability to count, especially when the type and visual appearance of objects are not consistent between images.

This thesis furthers the abilities of automated enumeration methods. Until recently, counting, in a machine-learning context, has been focused on individual, specific tasks. Often, counting methods were formulated as the classification or detection of a known kind of object. While people may count in this way in simple, known cases, we believe that the human ability to count goes beyond this restricted approach. When counting, a person does not have to check each pixel or object against a catalogue of all known kinds of objects. They can identify areas of interest that they believe could contain something 'worth counting' and then draw comparisons between them. In this way, people are able to count instances of types of objects they have never seen before with limited-to-no guidance.

The work presented in this thesis is motivated by the desire to automate the human ability to count in novel situations with minimal guidance. Foundational to our work is the intuition that human counting is comprised of two stages: object detection and repetition recognition. See Figure 1.1 for a visual representation of these two stages. In this thesis, we first address these stages separately. We then develop methods which approach both stages simultaneously to gain a more holistic understanding of counting.

1.2 Thesis Structure and Contributions

In this section, we discuss the structure of this dissertation, outlining the context of each section and their contributions.

Chapter 2 provides context and motivation to our work as a whole. We first outline the important network architectures relevant to our work. Then, we summarise the existing clustering and counting methods, provide an overview of their limitations, and discuss how they motivate our work. The developments within the field of class-agnostic counting that occurred concurrent to our work are presented in the relevant chapters.

Throughout the work presented in this dissertation, we encountered errors, ambiguities, and limitations of the most popular class-agnostic counting dataset, FSC-147. In **Chapter 3**, we address these and present FSC-133, an improved version of the popular dataset without those errors or ambiguities. This improved dataset enables more effective training of class-agnostic counting methods and, more importantly, more accurate evaluation of them. We then go on to discuss the need for a new dataset to enable the training and evaluation of class-agnostic counting methods in settings with multiple objects and explain why a synthetic dataset is the most appropriate choice for this setting. We then present the first Multi-class Class-Agnostic Counting dataset (MCAC). Both FSC-133 and MCAC are used in Chapters 5 and 6 to train and validate our methods, as well as those of our contemporaries.

In **Chapter 4**, we concentrate on the repetition recognition stage of counting, separate from the detection stage. We address the question 'Given a set of single instances, how many are there of each type?' by framing it as a clustering problem. We recognise that to function robustly, many of the previous clustering approaches require complex or large-scale parameter tuning. This tuning often requires information such as the number of clusters present in the dataset or

some form of distance metric to define the boundary between points which are similar or dissimilar. This prior knowledge is often difficult to gather or completely unknown, and parameters used to define similarity in classical methods do not generalise well to high dimensional spaces. These earlier clustering methods also generally divide a set of points in a single way. We observe that, in many cases, there are numerous ways to group a set of data points depending on the features relevant to a given task. A method capable of addressing this complexity would be highly beneficial. We believe that the ability of machine learning methods to identify trends as well as function in high-dimensional feature spaces will enable a flexible clustering method which can solve varied clustering problems with much less contextual information than previous methods.

To this end, we developed a Differentiable Mean Shift algorithm (DMS). This iterative neural network mirrors the traditional Mean-Shift algorithm in a fully differentiable way by gradually improving its understanding of cluster centres. DMS is trained using small amounts of 'side-information' in the form of pairwise 'similar/dissimilar' labels. DMS is able to cluster the same set of features in different ways depending on the side-information provided. This enables it to be task-specific, identifying groupings that are not the most obvious, while still requiring minimal human input. While we present this work mostly in the context of clustering images based on various visual features, it could be applied to any feature space or task. As DMS is completely differentiable, it can be used within other end-to-end trainable methods as well as a stand-alone clustering method.

While Chapter 4 focused on solely the repetition recognition half of the counting problem, Chapters 5 and 6 address both the object detection and repetition in tandem, creating complete instance counting methods.

Traditionally, counting methods are class-specific; methods would be trained to count occurrences of a specific object in an image. Class-agnostic methods

aim to count objects of any type. Class-agnostic counters should be able to function in much more varied contexts than class-specific ones without the need for retraining, removing the requirement for large amounts of training data of each kind of object during deployment.

Prior to our work, class-agnostic counting methods required intervention, in the form of exemplar images, during training and at inference time. The exemplar images are used to define the type of object to be counted. In many contexts, including those presented in the main class-agnostic counting dataset, we do not think this is necessary. Presented with an image containing only one kind of object and asked to 'count', a person understands the task at hand, even if the object class is novel. In **Chapter 5**, we introduce RCC, a class-agnostic counter that operates without the need for exemplars, marking the first instance of a method replicating this behaviour. Not only can RCC count without needing exemplar images at training or inference time, but it can also be trained using only scalar image-wise counts as supervision rather than using instance-wise locational labels. RCC achieves results competitive with prior and concurrent class-agnostic counting methods that use exemplar images and that train with instance-wise labels.

The work in Chapter 5 shows that the human ability to perform 'uninformed counting' is automatable in single-class settings. However, we recognise that many real-world applications do not have the known constraint that only a single kind of object will be present in each image while also not knowing the object class. Similar to our observation about the ability of humans to perform uninformed counting in single-class settings, we believe that people can, in general, count objects of multiple types in an image without being told the class boundaries or being given exemplars of type. We show in this thesis that this behaviour is replicable in an automated system.

In **Chapter 6**, we present A Blind Counter (ABC123) the first multi-class

exemplar-free class-agnostic counter. ABC123 can count instances of multiple types simultaneously without human intervention. While it is generally hypothesised that exemplar-based methods should function well in multi-class settings, using the exemplar images to clarify ambiguities, this has not been shown robustly. Using MCAC, we demonstrate that many pre-existing methods perform poorly in multi-class settings and that ABC123 performs well in them. We also demonstrate that the understanding of counting learnt from MCAC, a synthetic dataset, is transferable to photographic data.

Chapter 7 provides an overview of the research presented in this thesis, emphasising our contributions. Additionally, we explore and discuss potential directions for future research.

1.3 Publications and Public Releases

Here, we present the publications and public releases of the work included in this thesis:

- **M. Hobley**, and V Prisacariu. "DMS: Differentiable Mean Shift for Dataset Agnostic Task Specific Clustering Using Side Information." Completed in 2019, *ArXiv e-prints*. 2023.
- **M. Hobley**, and V Prisacariu. "Learning to count anything: Reference-less class-agnostic counting with weak supervision." Completed in 2022, in the *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 2023.
- **M. Hobley**, and V Prisacariu. "ABC Easy as 123: A Blind Counter for Exemplar-Free Multi-Class Class-agnostic Counting" *ArXiv e-prints*. 2023. Currently Under Review for the *Conference on Computer Vision and Pattern Recognition*. 2024.

2

Background

This chapter offers a comprehensive survey of the research relevant to this thesis. Our objectives are to establish the contextual backdrop of previous counting and clustering methods as well as to introduce the fundamentals of the tools we use in our work. To achieve this, we present the two most significant architectures to this thesis, convolutional neural networks and transformers. We then outline many of the foundational clustering methods as well as their limitations before discussing deep-learning-based improvements to them. Finally, we delve into the recent advancements in the field of automated counting. We describe the task of class-specific counting and how its limitations prompted the advent of class-agnostic counting. In describing the relevant work on class-agnostic counting, we make clear the motivation behind our work in this area.

2.1 Network Architectures

In this section, we discuss the two main architectures utilised in this dissertation and by others working in similar fields, namely convolutional neural networks and transformers.

2.1.1 Convolutional Neural Networks

Inspired by Neocognition [13], Convolutional Neural Networks (CNNs) were introduced and effectively trained using back-propagation [14, 15]. The early CNNs, known as LeNets [14], were initially devised for the recognition of handwritten digits. These networks featured convolutional layers followed by sigmoid units to provide non-linearity, with average pooling layers. The networks were structured hierarchically, with the initial layers being responsible for extracting local features from the input and the later layers gradually amalgamating them into more complex global features. The high-dimensional convolutional features are finally summarised using a fully connected layer to produce an image classification.

CNNs are an example of parameter sharing. The learned feature extractors are applied across the whole input, which is more efficient and encourages the filters to learn to recognise general translation-invariant features. Early networks were trained using a combination of the mean squared error (MSE) and maximum a posteriori (MAP) losses. While LeNets excelled on the MNIST dataset [15], they encountered challenges when applied to real-world vision problems in the 1990s. Two primary obstacles emerged: first, training large-scale CNNs necessitated an impractical number of iterations to converge using stochastic gradient descent (SGD) due to limited computational resources at the time; second, despite spatial sharing of convolutional kernels, typical CNNs still required millions of parameters, rendering them susceptible to overfitting on datasets like MNIST.

The breakthrough for CNNs in computer vision was in 2012, with the advent of

large-scale datasets and high-performance Graphics Processing Units (GPUs). This, combined with a series of innovations in CNNs, rapidly improved their performance in computer vision applications. AlexNets [16] employed rectified linear units (ReLUs) as their non-linear activations. To decrease the resolution of their data, highlight the most distinctive features, and improve their spatial invariance, AlexNets also introduced max-pooling layers. To combat overfitting, dropout [17] and data augmentation techniques, including colour jittering and random cropping, are commonly incorporated during training. These augmentations disrupt the usual training of a network and create a more robust understanding of the true patterns in the images rather than of the exact images themselves.

From 2014, significant improvements were made by adopting deeper architectures [18, 19]. Many of the contemporary counting methods use Residual Networks (ResNets), which have not significantly changed since they were introduced in 2016 [20]. These networks, often with over a hundred layers, leveraged skip connections, where the output of a layer is added to the output of later layers. The incorporation of residual units facilitates more effective backpropagation of gradients.

To achieve pixel-level predictions such as semantic segmentation [21, 22] or the density map regression used in many counting methods [23–25] rather than image-level classifications, CNNs often employ two stages: downsampling and upsampling. The downsampling stage resembles the architecture of a classification network, summarising the data into a lower spatial resolution but higher dimensional, and so more informative, feature space. The upsampling stage increases the spatial resolution of this condensed feature back to that of the original image. This is achieved using either alternating upsampling and convolutional layers or 'transposed' convolutional layers. The training loss of these networks varies based on the task at hand but generally can be grouped into pixel-wise losses, such as pixel-wise mean squared error [23, 24], or feature-wise [26] or image-wise [27] perceptual losses.

CNNs, by their nature, find relationships between spatially local features, starting at a pixel level and gradually expanding their receptive field, the area of the input which affects a single feature in later layers. However, even with the increased receptive field of later layers, a deep feature is constructed hierarchically from features with smaller receptive fields leading to an understanding of the image that is by definition local-context-first. While this locally aware approach is useful for tasks like image classification where the object of interest is spatially contiguous, we show it is not well suited for other tasks such as counting where spatially disparate objects are as important as close objects.

There have been various approaches to increasing the receptive field of a given feature or network, such as dilation [28, 29] or multi-scale features [23]. In dilation, the weights of kernels are applied to samples which are no longer spatially adjacent, effectively introducing 'holes' to the convolutional kernel. While dilation does increase the receptive field faster than non-dilated kernels, they are either still inherently local-first as they do not take global context into account directly or they are so dilated that they are no longer functional as convolutional kernels. Multi-scale features are generally created by using a set of scaled versions of the input image or by combining features from different latent layers which have different spatial resolutions [30, 31].

Instead of parameter sharing across spatial dimensions (a characteristic of CNNs in image processing), an alternative approach is parameter sharing across a temporal dimension. This concept underlies Recurrent Neural Networks (RNNs), which were originally developed in 1986 by Rumelhart et al. [32] and which are most commonly used for text processing or generation. Each feature neuron in an RNN takes the output from the previous time steps as its input. While this arrangement endows RNNs with a basic form of memory, it also exacerbates issues related to vanishing and exploding gradients. The root cause of this problem lies

in the repeated multiplication of parameter values during gradient computation, leading to gradient explosions when parameters exceed 1 or gradients vanishing when parameters fall below 1.

Long Short-Term Memory (LSTM) networks, introduced by Hochreiter and Schmidhuber [33] in 1997, try to mitigate the amnesia caused by these gradient-related challenges. LSTMs incorporate input, output, and forget gates within each recurrent cell. These gates enable the network to determine when to update information within the cell and when to discard it rather than relying on the same parameter multiplication at every time step. In the case of RNNs, which process words sequentially, they consider the present word and other words from the past within a defined 'context window'. As a consequence, more recent words near the end of a sentence receive greater attention, while earlier words gradually get lost to volatile neural activations.

The introduction of the *attention* mechanism revolutionised the text processing paradigm by providing all words in a sequence equal access to all other parts of the sequence, concurrently and in a parallel fashion. This eliminates the time-consuming aspect of serial processing and also combats the amnesia of RNNs and LSTMs. Initially, the attention mechanism was integrated into sequential RNNs for language translation systems [34]. However, as the field evolved, developments in Transformer-based large language models dispensed with RNNs altogether and relied exclusively on the highly efficient parallel Attention scheme; this is further discussed in Section 2.1.2. Counting, as we see it, requires an understanding of features that may appear far from each other to gain an informed representation of the objects present. As such, the global-context-aware features generated by utilising attention mechanisms appear well suited to the task of counting. Due to this intrinsic capability, we utilise them in Chapters 5 and 6.

2.1.2 Attention and Vision Transformers.

Attention layers vary from convolutional or recurrent layers as they dynamically adjust the weighting of input values rather than using static weights for all sections of the input data. An attention layer operates on three main components, namely the queries, Q, the keys, K, and the values, V, in three base steps: alignment scores, weights, and context vectors.

Alignment scores are found between each query, q, and each key k_i:

$$z_{\mathbf{q},\mathbf{k_i}} = \mathbf{q} \cdot \mathbf{k_i} \tag{2.1}$$

These alignment scores are then used to find weights using a softmax:

$$\alpha_{\mathbf{q},\mathbf{k_i}} = \text{softmax}(z_{\mathbf{q},\mathbf{k_i}}) = \sigma(\mathbf{z})_i = \frac{e^{z_{\mathbf{q},\mathbf{k_i}}}}{\sum_{j=1}^{\mathbf{K}} e^{z_{\mathbf{q},\mathbf{k_j}}}} \tag{2.2}$$

The weights are then used to find a context vector from the value vectors $\mathbf{v_{k_i}}$:

$$\text{attention}(\mathbf{q}, \mathbf{K}, \mathbf{V}) = \sum_i \alpha_{\mathbf{q},\mathbf{k_i}} \mathbf{v_{k_i}} \tag{2.3}$$

Transformers proposed by Vaswani et al. [35] use this multiplicative attention as it is more efficient than the initially proposed additive attention. They also employ only attention layers instead of combinations of attention and convolutional or recurrent layers. Transformers utilise multi-head attention which learns multiple linear projections of the queries, keys, and values in parallel. The attention mechanism is then applied to these separately before they are combined and projected again:

$$\text{multi-head}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{concat}(\text{head}_1(\mathbf{Q}, \mathbf{K}, \mathbf{V}), ..., \text{head}_n(\mathbf{Q}, \mathbf{K}, \mathbf{V}))\mathbf{W} \tag{2.4}$$

where $\mathbf{W}$ is the final projection, concat is the concatenation function, and head_i is defined as:

$$\text{head}_i = \text{attention}(\mathbf{Q}\mathbf{w}_i^q, \mathbf{K}\mathbf{w}_i^k, \mathbf{V}\mathbf{w}_i^v) \tag{2.5}$$

where $\mathbf{w}_i^q$, $\mathbf{w}_i^k$ and $\mathbf{w}_i^v$ are the linear projections of head_i.

This multi-head approach allows a single attention layer to construct its output based on different representational spaces. This facilitates a much more complex and varied understanding of the features than would be possible with a single representational space.

In 2020, Dosovitskiy et al. [36] applied this transformer architecture, originally designed for text-based problems, to vision tasks. Vision Transformers (ViTs) [36] were first used for image recognition but have since been applied to many other vision tasks including detection [37], segmentation [38], and synthesis [39]. Instead of operating on the features of single pixels like CNNs or single word-embeddings like RNNs, ViTs operate on patches of pixels, which have been collapsed to a single feature vector. Patches are used to lower the number of features to which attention is applied. This is because the attention mechanism in the transformer architecture scales as $\mathcal{O}(n^2)$ as it compares all inputs with all other inputs. To facilitate the use of the positional information from the original image, learnt position embeddings are added to the patch embeddings at each attention layer.

We show that attention-based architectures that model global context inherently

are well suited for the task of counting due to their understanding of spatially disparate repetition. In counting, if two features are similar but far apart, it is significantly more likely that they are different instances of the same kind of object rather than different parts of the same instance than if the features are neighbouring.

2.2 Clustering

As discussed in Chapter 1, a fundamental component of counting is the ability to identify objects of a similar type. If each object instance is already separate and the set of object types is known and consistent, this can be formulated as a classification task. 'Find the objects that are of type x'. However, if the set of object types is not known, is not consistent, or is without a set of robustly labelled examples, a different approach must be taken. Clustering methods aim to group data points based on similarity, agnostic to their specific class. The general procedure for clustering is to first formulate the input data as a graph and then partition the graph into sub-graphs. Generally, this is achieved by developing a proxy metric for similarity using known instances and then applying this to unknown instances.

Classical deterministic clustering algorithms can be roughly grouped into spatial clustering [40, 41] and spectral clustering [42]. Spatial clustering methods use a metric function, typically either the L_1 distance, the Euclidean distance, or the cosine distance, to evaluate the similarity between two data points. Spatial clustering methods include k-means [40], DBSCAN [41], and mean shift [43]

We will outline some of the most common classical clustering methods, as well as their limitations, before discussing some recent ways deep learning has been utilised to address these limitations.

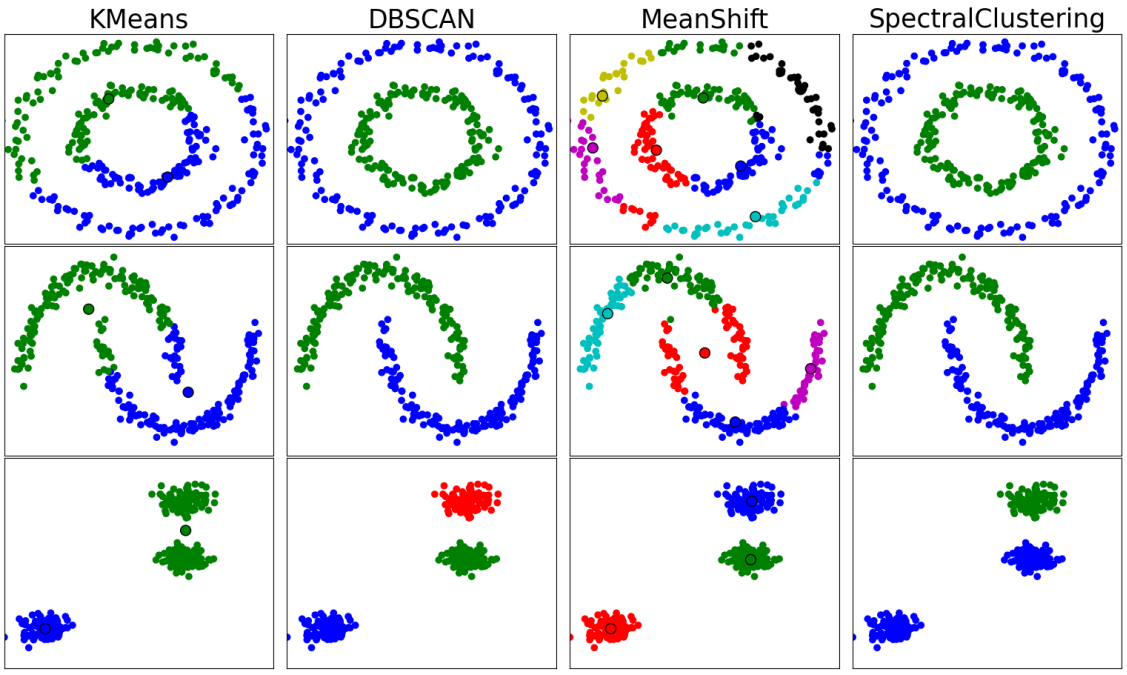


Figure 2.1: Examples of classical clustering approaches. In cases where the method predicts cluster centres, these are shown with an outlined circle.

2.2.1 Formulation

Clustering algorithms aim to group a set of n d dimensional points $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} = \mathcal{X}$ into $k \leq n$ sets $\{S_1, S_2, ...S_k\} = \mathcal{S}$ with centres $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k\} = \mathcal{M}$.

$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{\mathbf{x} \in S_i} \mathbf{x} \tag{2.6}$$

where $|S_i|$ is the number of points in cluster i.

2.2.2 K-means

The objective of k-means clustering is to divide a set of n observations into k clusters, with each observation being assigned to the cluster whose mean, often referred to as the cluster centre, is closest to it. This cluster centre acts as a representative of the cluster itself. The primary aim of k-means clustering is to minimise the intra-cluster variance, measured using L_2 distance, which consequently divides the

data space into Voronoi cells. Mathematically:

$$\underset{\mathbf{S}}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 . \tag{2.7}$$

Algorithm 1 k-means algorithm

function K-MEANS($\mathcal{X}$, k)
  Select $\{\bar{\boldsymbol{\mu}}_1^{(0)}, \bar{\boldsymbol{\mu}}_2^{(0)}, ..., \bar{\boldsymbol{\mu}}_k^{(0)}\} = \bar{\mathcal{M}}$ a set of cluster centre initial estimates
  while $\mathcal{S}^{(n)} \neq \mathcal{S}^{(n-1)}$ **do**
    Assign each data point to the cluster with the nearest centre:
    $S_i^{(n)} = \left\{ x_p : \left\| x_p - \mu_i^{(t)} \right\|^2 \leq \left\| x_p - \mu_j^{(t)} \right\|^2 \ \forall p, \forall j, 1 \leq p \leq n, 1 \leq j \leq k \right\}$
    Recalculate centres using the data points assigned to each cluste:
    $\boldsymbol{\mu}_i^{(n+1)} = \frac{1}{\left| S_i^{(n)} \right|} \sum_{\mathbf{x} \in S_i^n} \mathbf{x}$
  end while
end function

K-means and other similar algorithms are effective in cases where the number of clusters is known and the intra-cluster variation is lower than the inter-cluster variation. This is because they maximise the similarity of data points in the same cluster and maximise the difference between points in different clusters. However, this makes it unsuitable for cases where the clusters vary significantly in shape, size, or density. K-means is also very sensitive to the initialisation of the estimated means; see Figure 2.1 for examples of failure cases of k-means, where k was not correct or the manifold of the data was such that there was greater intra-class than inter-class variation.

K-means++ [44] improves the cluster centre initialisation by finding spread-out sample points. K points are randomly selected as before, but they are selected sequentially, and the selection of a given point is weighted based on the reciprocal of the squared distance between it and other already sampled points. Another improvement on k-means was developed by Xing et al. [45], who focused on learning a more appropriate distance metric using pairwise side-information. This idea

has since been extensively explored [46–50] and motivated our work in Chapter 4. K-means and its variants are useful because they are simple and robust. However, in practice, k is often unknown.

2.2.3 DBSCAN

Another branch of spatial methods is linkage methods. Linkage methods adopt a bottom-up strategy in which small clusters are merged using local neighbour criteria rather than information about the whole manifold. Examples are Ward's hierarchical clustering (AC-W) [51] and DBSCAN [41].

DBSCAN is a density-based clustering method that uses a maximum distance threshold, ϵ, and a minimum neighbours threshold, $minPts$, to define its clusters. DBSCAN initially generates a graph of points. Points are connected iff the distance between them is below ϵ. Points are classified as core points, non-core points, and outlier points. Core points are directly connected to more than $minPts$ numbers of points. A cluster is defined by a graph that includes all directly connected core points and all non-core points that are directly connected to one or more core points. Outlier points are points not connected to a cluster, *i.e.* any point that is neither a core point nor directly connected to a core point.

If the number of clusters and the minimum cluster size are known, then a search over ϵ can be used to find the appropriate clusters.

As DBSCAN and other similar linkage-based methods only look at local neighbourhood densities rather than using a broader dataset-wide viewpoint, they show good performance in cases when the shape of the clusters varies significantly. They also often work well in cases where there are outlier points or noise since unlike methods like k-means, they are not required to assign all points to a cluster. DBSCAN significantly differs from k-means as it can create clusterings where some intra-cluster distances exceed some inter-cluster distances by an arbitrary

Algorithm 2 An abstract description of DBSCAN

```
function DBSCAN(𝒳, ϵ, minPts)
    i = 0
    𝒮 = {}
    for each unvisited point x ∈ 𝒳 do
        mark x as visited
        𝒩 = regionQuery(x, ϵ), find the set of neighbouring points
        if |𝒩| ≥ minPts then
            𝒮_i = {x, }
            𝒮 = 𝒮 ∪ {𝒮_i}
            expandCluster(x, 𝒩, 𝒮_i, ϵ, minPts)
            i++
        else
            mark x as NOISE
        end if
    end for
end function

function expandCluster(x, 𝒩, 𝒮_i, ϵ, minPts)
    for each point x′ ∈ 𝒩 do
        if x′ is not visited then
            mark x′ as visited
            𝒩' = regionQuery(x′, ϵ)
            if |𝒩'| ≥ minPts then
                𝒩 = 𝒩 ∪ 𝒩'
            end if
        end if
        if x′ is not yet member of any cluster then
            𝒮_i = 𝒮_i ∪ x′
        end if
    end for
end function

function regionQuery(x, ϵ)
    return all points within x's ϵ-neighborhood (including x):
    𝒩 = {y|y ∈ 𝒳 ∧ ||y − x||_2 ≤ ϵ}
end function
```

amount; see Figure 2.1.

The performance of DBSCAN and other linkage-based methods are significantly degraded in higher dimensional feature spaces as their affinity measures are grounded in observations of low-dimensional data [52]. GDL [53] works in a similar fashion to DBSCAN but defines a more complex affinity metric using the average in-degree and out-degree of each data point. This makes it more robust to noise, and it proves more consistent in high-dimensional feature spaces.

2.2.4 Mean shift

Mean shift is a non-parametric 'mode-seeking algorithm'. In its simplest form, it iteratively locates an estimate of the maxima of a density function given discrete samples from that function. An estimate for the maxima is found by taking a previous estimate and a weighted average of points proximal to the previous estimate. The weights for the average are typically based on the distance from the current estimate using a Gaussian Kernel. The vector between one estimate and the next is called the *mean shift* [43]. This process is repeated until the algorithm has converged. There is currently no formal proof for the convergence of mean shift in high (>1) dimensional space using a general kernel [54].

In a clustering context, mean shift iteratively improves its estimate of the cluster's centre or sample mean by averaging the position of points it currently believes are inliers to that cluster. Data points that converge to the same sample mean are grouped together into a single cluster. In order to sort a whole dataset, the classical mean shift algorithm must be initialised from each data point.

Classical mean shift is based on two key assumptions: firstly, points that are similar belong to the same cluster, and, secondly, that Euclidean distance is a good proxy for similarity.

The mean shift algorithm is formulated as follows. Given a set of data points,

$\mathcal{X}$, an estimate of the cluster centre $\bar{\boldsymbol{\mu}}_i^{(n)}$ associated to point $\mathbf{x}_i$ is improved using the sample mean function m as:

$$\bar{\boldsymbol{\mu}}_i^{(n+1)} = m(\mathcal{X}, \bar{\boldsymbol{\mu}}_i^{(n)}) = \frac{\sum_{\mathbf{x}\in\mathcal{X}} \mathbf{x}K(\mathbf{x}, \bar{\boldsymbol{\mu}}_i^{(n)})}{\sum_{\mathbf{x}\in\mathcal{X}} K(\mathbf{x}, \bar{\boldsymbol{\mu}}_i^{(n)})}, \tag{2.8}$$

where $K(.)$ is a kernel function which evaluates the similarity of two n dimensional points in a 0 to 1, dissimilar to similar, range.

Given the sample mean function $m(.)$ defined on some collection of data with an initial value $\bar{\boldsymbol{\mu}}^{(0)}$, a cluster centre together with the cluster inlier data points can be calculated using the mean shift algorithm. The sample mean is iteratively updated until the distance of sample means between two consecutive iterations becomes sufficiently small, *i.e.* $\|\bar{\boldsymbol{\mu}}^{(n+1)} - \bar{\boldsymbol{\mu}}^{(n)}\| < \tau_c$, where τ_c is the convergence threshold; see Algorithm 3. The difference between an estimate of the sample mean $\bar{\boldsymbol{\mu}}^{(n)}$ and the initial estimate of the sample mean, $\bar{\boldsymbol{\mu}}^{(n)} - \bar{\boldsymbol{\mu}}^{(0)}$ is known as the *total mean shift*.

Algorithm 3 Mean-Shift Algorithm

function MEAN-SHIFT($\mathcal{X}$, τ_c)
  Initialise a set of centre estimates $\{\bar{\boldsymbol{\mu}}_1^{(0)}, \bar{\boldsymbol{\mu}}_2^{(0)}, ..., \bar{\boldsymbol{\mu}}_k^{(0)}\} = \bar{\mathcal{M}}$
  for each $\bar{\boldsymbol{\mu}}_i^{(0)} \in \bar{\mathcal{M}}$ **do**
    while $\left\|\bar{\boldsymbol{\mu}}_i^{(n)} - \bar{\boldsymbol{\mu}}_i^{(n-1)}\right\| \geq \tau_c$ **do**
      $\bar{\boldsymbol{\mu}}_i^{(n+1)} = m(\mathcal{X}, \bar{\boldsymbol{\mu}}_i^{(n)})$
    end while
  end for
end function

To cluster all points, the algorithm must be initialised from each point $\bar{\boldsymbol{\mu}}_i{}^{(0)} = x_i$ for all $x_i \in \mathcal{X}$ (i. e. $k = n$). The data points are then grouped based on their estimated centres after convergence, *i.e.* if $\left\|\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j\right\| < \tau_s$, then x_i and x_j are in the same cluster, $S = \{i, ..., j, ...\}$.

Mean shift can also be used on a subset of points to find centres which then act as initialisation points for another clustering method such as k-means. In Chapter 4,

we present a method inspired by the iterative approach of cluster finding used in mean-shift. Our method, however, uses a neural-network-based similarity kernel which allows it both to work more effectively in high-dimensional feature spaces and to address different tasks more flexibly.

2.2.5 Spectral clustering methods

Spectral clustering methods [42, 55–58] aim to reduce the dimensionality of a feature space before clustering it using any of the above methods. This is achieved by converting an $n \times n$ data point distance graph into its Laplacian and taking the k most significant eigenvectors to form a $n \times k$ *spectral embedding*; see Algorithm 4. This spectral embedding can be used to improve the performance of other clustering methods, such as k-means.

Algorithm 4 An abstract description of Spectral Clustering

Generate a similarity matrix $\mathbf{A}$ of all points, $\mathbf{A}_{ij} \geq 0$
Calculate the Laplacian $\mathbf{L}$ (or the normalised Laplacian) $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D}$ is the diagonal matrix $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$
Compute the eigenvectors associated with the k least eigenvalues of matrix $\mathbf{L}$
Construct the spectral embedding, the $n \times k$ matrix taken from the first k eigenvectors
Cluster the n graph nodes in this new spectral feature space

As spectral clustering methods decrease the dimensionality of the data, they handle cases with high-dimensionality or non-standard manifolds well. However, their performance in cases with outliers or noise is significantly reduced as the eigenvectors of the Laplacian are very sensitive to perturbations [59].

2.2.6 Deep Clustering

Most classical clustering approaches use simple metric functions and work well with low-dimensional data. However, they generalise poorly to higher dimensions given the added complexity of finding affinity in higher-order feature spaces [60]. This

phenomenon is closely related to the 'curse of dimensionality' [61, 62].

Prior deep learning methods attempt to either (i) encode data points into a features space that can be more effectively clustered using classical methods or (ii) mimic and improve upon conventional spatial/spectral clustering approaches with a set of differentiable layers [63].

DCN [64] trains a deep neural network to perform an optimised, non-linear dimensionality reduction and then uses k-means clustering. They train their dimensionality reduction stage using a k-means loss. This enforces a condition that the assigned data points are evenly distributed around the cluster centres. Jule [65] uses an agglomerative clustering loss to find an appropriate and reduced feature space. Jule uses an iterative approach where data points are clustered based on an affinity metric. Their network is then trained to further optimise the affinity in the now-merged clusters. This process repeats, resulting in a feature space best optimised for hierarchical methods like agglomerative clustering or DBSCAN.

Deep Embedding for Clustering (DEC) [66] and DEPICT [67] use a combination of an autoencoder-reconstruction loss to reduce the dimensionality of the feature space and a cluster-assignment loss. The cluster-assignment losses aim to make soft cluster predictions align with a set of target predictions, pushing points deemed similar closer together in the lower dimensional feature space. Similarly, DAC [68] creates a feature space which it iteratively improves by driving similar points together using a binary pairwise classification loss.

While linear projections and autoencoders are the most commonly used dimensionality reduction approaches, other methods have been proposed to similar effect. Hu et al. [69], Hsu and Lin [70], and Vittal Premachandran and Yuille [71] utilise either self-augmentation or Generative Adversarial Networks (GANs) to reduce the dimensionality of their respective feature spaces while maintaining or improving the understanding of data point similarity. The impact of these

improved feature spaces is clear.

Most of the above methods boast significant improvements on high-dimensional data as compared to the classical methods upon which they are based. However, these methods generally focus solely on improving the suitableness of the feature space for classical clustering rather than changing the method of clustering itself.

Deep clustering approaches that are based on classical clustering methods generally have the same flaws as those classical methods such as requiring supervision in the form of dataset-specific parameters. This parameter-tuning significantly decreases the usefulness of these clustering approaches in applications where end-to-end training is required as well as in deployment contexts where users may not have the knowledge needed to tune these parameters.

Robust Continuous Clustering (RCC) [72] and Deep Continuous Clustering (DCC) [73] both create an improved feature embedding and then generate the clusters directly. They use a locality-preserving loss to learn a more robust feature embedding that is consistent with a classical k-NN graph, bringing mutual k-NN samples closer to create tighter clusters. Shah and Koltun [72] also proposed RCC-DR which utilises a linear autoencoder for dimensionality reduction. Instead of using a classical clustering method on the embedded feature space to generate discrete cluster definitions, both RCC and DCC generate clusters in a continuous fashion. This allows them to be more easily incorporated into methods trained in an end-to-end fashion.

In Chapter 4, we focus on clustering a feature space without explicitly embedding it. This is because it is not always possible or efficient to train an encoder for the clustering stage, such as when the clustering is added as a head to a system that is required to be memory and time-efficient. While many methods need fully-labelled training data, we do not wish to impose this requirement as it would severely limit the viable deployment scenarios of our method. While the work presented in

Chapter 4 focuses explicitly on clustering difficult-to-divide datasets using simple untrained features, the above encoding methods could be complimentary as they would likely improve the input feature space making clustering simpler.

2.2.7 Side-Information

Completely unsupervised methods, although highly appealing as they do not require the laborious labelling of data, cannot exploit externally applied semantic concepts. This can result in learned representations that prove challenging to interpret and may not align with a user's understanding. Side-information based approaches aim to strike a middle ground, allowing a user to align the learned insights to their knowledge of the data but not requiring large-scale data labelling. Side-information generally refers to information known about the data that cannot be directly used to supervise training, *i.e.* it is not the ground truth label. Pairwise co-cluster labelling as introduced by Xing et al. [45] is a good example of side-informational supervision. Pairwise labels define a list of pairs of data points that are 'similar' and a list of pairs that are 'dissimilar' without labelling any other information. These pairwise relational labels are, in many cases, easier to gather than more descriptive labels like the ground-truth class of a data point. Pairwise information can be used to find, or act as a substitute for, other required information such as a distance tolerance.

Pairwise labels are commonly utilised in two separate ways, Metric Learning [45, 74–76] and Constrained Clustering [77, 78]. In the former, a feature space is learnt prior to clustering which minimises the intra-cluster variance of similar points while maximising the inter-cluster variance of dissimilar points. In the latter, the clustering algorithm itself is altered to penalise grouping dissimilar points and not grouping similar points.

Deep-learning-based clustering methods that use pairwise side-information also follow these two paradigms. Hsu and Kira [79] use a neural network to predict a

cluster membership probability vector for each data point which is then evaluated using a constrained clustering loss. The pairwise supervision is then used in combination with a contrastive KL-divergence term to penalise similar pairs having different cluster membership probabilities and dissimilar pairs having similar cluster membership probabilities. Fogel et al. [80] improves upon RCC [72] by using pairwise constraints to strengthen or remove the links between data points found using K-NN. They utilise a Siamese autoencoder network to generate representations of pairs of data points. They then use a clustering loss in combination with a reconstruction loss to encourage diverse and meaningful latent representations which are proximal for similar pairs and distant for dissimilar pairs. Shukla et al. [81] is similar to Fogel et al. [80] in that they train a convolution autoencoder with a combination of a metric learning loss term and a reconstruction loss. It encodes latent representations suitable for k-means clustering, encouraging low intra-cluster variance, while simultaneously regressing cluster centres. They also incorporate a cluster membership KL divergence term as in Hsu and Kira [79]. Various works have looked into the resilience of clustering methods to errors in pairwise supervision [48, 82] as well as into how they are affected by having non-binary 'similar'/'dissimilar' boundaries [49].

As our clustering method, presented in Chapter 4, must be lightweight, it does not employ the metric learning approaches of many other methods and instead takes a constrained clustering approach. The metric learning stages are generally used prior to a clustering stage which uses a classical distance metric and so do not scale well to high-dimensional spaces. Instead, our method learns a minimal, but high-dimensional, kernel which can estimate the similarity of data points, in the original high-dimensional feature space.

2.3 Counting

As mentioned in Section 1.1, we believe counting is comprised of two processes: identifying objects that could be of interest and enumerating the repetitions of them. The latter process can be approached as clustering. In the established literature, *'Counting'* is used to describe systems which include both the instance finding stage as well as the clustering or repetition recognition stage. Most commonly, automated counting is applied in the context of computer vision and image processing, though it could in theory be applied to other data modalities.

2.3.1 Class-specific counting

Generally, methods that have aimed to complete the whole task of counting, that is finding and enumerating objects, have focused on counting a single or small set of known classes of objects. These class-specific counters have been useful in a variety of industries, including ecology with animal counting [1, 2], urban planning with vehicle [3–5] and crowd counting [4, 6–10], and various medical and biological disciplines that require cell counting [11, 12]. These industries all benefit from being able to gather accurate and reliable data in a short amount of time with minimal manual labour.

Class-specific methods are trained to locate a specific type of object. They require an individually trained network for each type of object to be counted, with limited to no capacity to adapt to novel classes. Individually trained networks require gathering new data and retraining whenever a new type is considered, which is difficult, time-consuming, and expensive. Additionally, these methods generally aim to explicitly localise instances before enumerating them, requiring point-level annotations to supervise the training. These class-specific and point-level supervised systems are only feasible when the composition and appearance of object types will remain the same indefinitely and point-level annotations exist, which is often not

the case in real-world applications. Counting methods can be broadly grouped into detection-based, regression-based, and classification-based methods.

2.3.2 Detection-based Counting

Detection-based methods use standard object detection or segmentation approaches [83, 84] to find all objects of a given type and then enumerate them. This can be used on either a single class [85] or on multiple classes [86]. Detections can be found using the original input image or a constructed feature space such as local-maxima detection on a regressed density map [3, 10, 87–89]. Detection-based methods are often trained using bounding box labels which are more expensive to gather than the point- or image-level annotations used by some other branches of methods. To bridge this gap, some methods aim to use topological information to improve their results when only using point annotations [87, 89–91]. For example, Liu et al. [91] generate pseudo-ground truth bounding boxes from their point-level annotations, and Abousamra et al. [90] combine density map regression with a detection stage which learns topological information about the probability that people are near each other and uses this to inform more accurate predictions. These methods, however, generally have worse performance than those that use full bounding box supervision.

Detection-based methods work best in settings where individual objects are easily distinguishable. Object detection and segmentation, separate from counting applications, are broad fields with significant recent developments that can be beneficial in counting applications. Their performance in cases with high overlap or occlusion has improved materially in recent years. However, detection-based counting methods are still unsatisfactory in high-density applications, applications with complex or highly textured objects, or applications with large intra-class variation and low inter-class variation.

2.3.3 Classification-based Counting

Classification-based methods [5, 60] generate a discrete classification of an image's global count. Their most obvious flaw is that they treat each 'count class' independently rather than as a continuous variable. This means all incorrect counts, independent of their proximity to the ground-truth count, are treated as equally wrong. This makes training difficult as you need data from each discrete count-class, meaning you either need large amounts of data and/or a low maximum count. Liu et al. [92] partially solved this by using a nonlinear quantisation strategy and making block-wise classifications to generate a 'coarse count level' rather than the exact count for areas of an image. This produced a fairly accurate classification counter. However, it was imprecise, generating only a range that the count fell into, not the exact value.

2.3.4 Regression-based Counting

Regression-based counting methods aim to regress a single image-wise count [6, 7, 93] or a pixel-wise density map prediction [9, 11, 23, 94]. The pixel-wise density map prediction is such that when an area is integrated, it gives a count value for the corresponding area of the image.

Regression methods that aim to find density maps are often cheaper to train than detection-based approaches as they only require point annotations rather than bounding boxes. Point annotations already exist for datasets in many fields as the process of 'dotting' or 'pointing' is standard practice for manual counting [95–100]. The approximate ground-truth density map is then generated by placing a Gaussian kernel centred at each point annotation.

Density map regression methods have been around for a long time [8] and operate in various ways using features ranging from simpler low-level classical features [93] such as texture [8, 101, 102] to complex features generated from neural networks

[96, 103]. As regression-based methods generate a decimal count rather than a definite integer, they were originally only used to estimate the overall count in cases where the objects were hard to detect individually. However, recent methods are more precise and can better locate specific instances in complex settings rather than estimating the density of the instances. A regressed density map can also be used as a rudimentary object detector, followed by cross-correlation with an exemplar to find instances of the desired class, such as in Sokhandan et al. [104].

While density map regression-based methods can produce accurate counts, they are often blurry and cannot accurately locate individual instances in dense environments [88, 89] due to their use of a Gaussian kernel for ground truth generation.

In this thesis, we approach counting as a regression task. We show that for single-class images, as posed in most of the prior literature, density map regression is not needed and that scalar image-wise count regression is sufficient. We then go on to show that in more complex, multi-class images, density map regression is useful for disambiguating counts during training and for quantitative analysis.

2.3.5 Weakly-supervised Counting

The majority of the counting methods outlined above use some form of instance-wise positional information, which is costly to gather. Weakly-supervised counting methods aim to generate an accurate count with a reduced annotation burden. They are trained with minimal [105, 106] or no point-level annotations [7, 107, 108]. The methods to facilitate training of an accurate model with reduced supervision vary greatly.

Borstel et al. [107] proposed the use of Multiple Instance Regression (MIR) to reduce the normal labelling requirements. This formulation uses a set of annotated counts corresponding to regions of the image. The exact location of the instances is not labelled. The model then learns to assign pixel-level densities to match

the region-level counts.

MATT [105] is trained predominantly using count-level annotations but with a small amount of point-level annotations. They found that training a single network with a single count value was not sufficient in constraining their problem. Instead, they found that training multiple networks which all have to generate the same count given the same input added a coherence constraint sufficient to make their results competitive.

Yang et al. [108] proposed a soft-label sorting network that is trained mostly by trying to sort images by count. They found this helped them generate more representative features that were then used to directly regress the image-wise count.

Another approach to minimising the annotation burden is to conduct most of the training in an unsupervised way. Sam et al. [106] took this approach, optimising 99.9% of their stacked auteoencoders' parameters with unlabelled data. The remaining 0.1% of parameters were optimised using point-level annotations.

Liu et al. [109] use a self-supervised approach to gain a meaningful understanding of counting by utilising completely unlabelled crowd data during training. They do this by creating a ranking network that orders a set of hierarchical crops from a crowd image. This is built on the fact that a cropped version of an image must have an equal or lower associated count than the uncropped image. This proxy task, which uses easy-to-gather unlabelled images, improved their accuracy by an average of 10%.

We endeavour, where possible, to minimise the amount of labelled training data required to learn to accurately count. To this end, we show in Chapter 5 that in settings with only a single kind of object present and so with minimal ambiguity, weak supervision is all that is needed to learn to count, even when the kind of object is novel.

2.3.6 Class-agnostic Counting

In contrast to the class-specific methods above, which require a static class composition, class-agnostic counting methods, introduced in 2018 [4], aim to learn a general understanding of counting that can be applied to arbitrary classes. This means they can, in theory, count objects of types they did not encounter during training, which makes them useful for deployment in many more fields, where gathering large amounts of data may be prohibitively expensive or time-consuming. In general, class-agnostic counting has been achieved by creating a sufficiently general feature space and then applying some form of matching to the whole feature map [24, 110] or to proposed regions of interest [23, 104]. This matching utilises an exemplar image, which is provided at both training and inference time, to serve as an example of the type of object to be counted.

The first class-agnostic counting method, Generic Matching Network (GMN) [111], introduced the idea of performing counting in an extract-then-match way. They had a very simple approach to this. They embed the exemplar and query images into the same feature space, concatenate them, and then match them using an architecture comprised of a (3×3) convolutional layer and a (3×3) convolutional transpose layer in order to generate a density map of the same resolution as the input image. They utilised tracking video data to train a backbone with a general feature space which they then fine-tuned to specific domains using labelled data on the order of dozens to hundreds of images. GMN struggles with completely novel classes without fine-tuning [23].

Ranjan et al. [23] improved upon the class-agnostic paradigm laid out by Lu et al. [111] with FamNet, which can achieve counting on previously unseen classes without the need for domain-specific adaptation. They achieve this using a multi-scale feature extraction module, which generates meaningful general features, and a density prediction module, which matches features and regresses a density map.

They achieve multi-scale feature matching by performing ROI pooling on the convolutional feature maps from sequential blocks from their pre-trained backbone that are at different resolutions. Ranjan et al. [23] also incorporated two test-time adaptations to improve their results, both of which were based on the exemplar images being drawn from the query image. They firstly apply a scaling based on the knowledge that the exemplar bounding box must contain ≥1 instance. Secondly, they apply an idea similar to correlation filters which are used in in tracking [112]. The idea is that as perturbations to the bounding box increase, the target response should decrease, in this case following a Gaussian distribution. These two adaptations are combined and applied as a loss at test time to simulate a single-query fine-tuning.

Shi et al. [24] expand upon the extract-then-match paradigm of previous methods with BMNet and BMNet+, which learns a similarity metric and loss jointly with the feature representation. Their learnt similarity loss is inspired by metric learning [113], specifically that a good similarity metric should be low for features drawn from different categories and high for features drawn from the same category.

At the point of conducting the work in Chapter 5, there were no exemplar-free counting methods, and all class-agnostic counting methods utilised instance-wise annotations, *i.e.* there were no weakly-supervised class-agnostic counters. We believe that such a method is possible and desirable for three reasons. First, given the human ability to arbitrarily count objects, why should algorithmic counting methods need exemplar images? Second, removing the need for exemplar images increases the deployment utility of our method as human intervention is no longer needed. Third, given the success of weakly-supervised class-specific counting methods, the requirement of instance-wise locational annotations in class-agnostic counting is unnecessary. In Chapter 5, we show all of this to be the case. That is, a weakly-supervised exemplar-free class-agnostic counting method is not only possible but competitive with exemplar-based and strongly-superverised contemporary methods

in single-class settings.

It should be noted that class-agnostic counting is distinct from multi-class counting, where objects of different classes appear within the same image. Before our work, class-agnostic counting had only been approached in single-class settings. Lu et al. [4] qualitatively demonstrated their abilities on images with a single class of object present and quantitatively evaluated their method on cell microscopy images [94, 114, 115] and drone-collected car images [85], all of which only have one kind of object present.

Following its introduction in 2021 [23], FSC-147 has become the default class-agnostic counting dataset [23–25, 116, 117]. Almost all of the images in FSC-147 contain only a single type of object, and in the few cases with multiple object types, only one is labelled. This has created a situation where class-agnostic counting methods are incentivised to develop an understanding of counting which is based on the idea of 'generic object detection' rather than having a discriminative approach which only counts similar objects.

While some exemplar-based methods were assumed to have the ability to function robustly in settings with more than one class of object present this has not been proven. In fact, we found this not to be the case; see Chapter 6 for details. Our work, detailed in Chapter 5, shows that a method with no class-direction (*i.e.* no exemplar images) can achieve state-of-the-art results on FSC-147, demonstrating that it contains few images in which it is ambiguous what is to be counted.

As it is clear that a good dataset encourages the development of methods to address it, and it is of course useful to have tools that function in multi-class settings, we created the first multi-class class-agnostic counting dataset, presented in Section 3.3. We utilise this dataset in the development of the first multi-class exemplar-free class-agnostic counting method, detailed in Chapter 6. Unlike existing methods, this method simultaneously counts objects of multiple classes and

is robustly shown to differentiate between different classes rather than functioning as a generic object detector.

2.3.7 Attention in Counting

Inspired by the success of attention-based architectures and transformers in other tasks, various methods aim to utilise these advancements in counting. Liu et al. [118] propose the Attention Map Generate (AMG) which generates two priors for a CNN, a mask of important 'candidate' regions and a 'congestion degree' which estimates a range of likely count values. The former aids the following stages in focusing on areas of high importance, for example, ignoring noisy backgrounds. The latter allows the later stages to generate more precise answers by focusing it on a smaller range of possible values.

Jiang et al. [119] works in a similar way, proposing regions of interest with a Density Attention Network (DANet) and then using an Attention Scaling Network (ASNet) to correct for systemic under- and overestimating errors. This method multiplies the counts by an attention score, effectively generating different counts for different areas and then summing them. Zhang et al. [120] proposed the Relation Attention Network (RANet) which utilises local and global self-attention on CNN features. This creates more informative features which can capture dependencies over the whole image.

In 2021, Liang et al. [121] proposed CLTR, the first method to apply transformers to crowd counting with limited or no point-wise labelling, *i.e.* weakly-supervised crowd counting. This was followed by various other crowd-counting methods including CrowdFormer [122], JCTNet [123], and TDCrowd [124]. CrowdFormer [122] is a transformer-based method which addresses issues of people appearing at varying scales in a single image. It utilises a pyramid-structured vision transformer to capture multi-scale information. This is similar to many multi-scale CNN methods,

but it incorporates the global receptive field of a transformer.

Wang et al. [123] recognised that patching images during counting tasks can cause significant issues. If an object spans a patch boundary, it could be counted in both or neither, causing erroneous results. Their method, JCTNet, combats this issue by utilising the fact that convolutional network features are locally aware. They use convolutional layers to create locally aware features which are then passed to a transformer architecture to perform the final counting. This significantly improves their results when objects spanned patch boundaries.

TDCrowd [124] is similar to the above methods in that it can be trained in a weakly-supervised pipeline, but it can also be trained with point-level labels. When trained with weak-supervision it achieved similar results to CLTR [121]. However, when it is trained with instance labels, it is ~15% more accurate, showing that even with a more contextually aware backbone, locational supervision is beneficial to counting tasks.

All of the above attention-based methods focus on crowd-counting, a class-specific task with limited ability to generalise to class-agnostic contexts. Lao-net [117] was the first method to use attention in a class-agnostic setting. It utilises a feature correlation module which is composed of Self-Attention and Correlative-Attention modules to learn inner-relations and inter-relations. The Self-Attention module creates more informative globally-aware features. The Correlative-Attention module then takes the place of the feature-matching convolution of most other class-agnostic counting methods. They achieve significantly better results (~35% better MAE) than the prior, non-attention-based class-agnostic counting methods. However, they require both exemplar images and point-wise supervision.

In this thesis, we recognise the power of the vision transformer architecture and how it is uniquely suited to counting tasks. As such, we utilise it to its fullest. In Chapter 5, we show that the features generated from a well-trained

vision transformer are sufficient to perform class-agnostic counting in single-class scenarios without exemplar images or instance-wise supervision. In Chapter 6, we expand on this, proposing an exemplar-free method which works in settings with multiple objects present, counting each simultaneously.

2.3.8 Count Ambiguity

Different counting tasks have significantly different levels of associated ambiguity. Class-specific methods have very little ambiguity. The task is posed as 'How many instances of type x are in the query image?'. As long as one can train a network to have a robust understanding of the complete variation of the desired type, this has no ambiguity. However, in settings without an expectation of visual coherence between the training and inference time images, this approach cannot be taken.

An exemplar-based class-agnostic counting method is easily understood and relatively unambiguous. The task is posed as 'Given an exemplar image, how many of this type are present in this query image?'. In theory, a well-trained exemplar-based method can be applied to images with either a single type of object present or many kinds of object present. However, in Chapter 6, we found that current exemplar-based methods [23–25] were not discriminative enough of object-type to achieve this robustly, mainly due to the available training data. While this setting is usually well-defined, it is clear that the exemplar images will not capture the full distribution of the possible visual appearance of the objects. This introduces an ambiguity. An informed guess must be made on what is of the 'same type' as the exemplars. For example, given two exemplars of blue cars, should the network count all cars present in an image or just the blue ones?

In a known single-class exemplar-free setting, the question being posed is also relatively simple and unambiguous, 'How many objects of the *main* type appear in this image?'. While there is some ambiguity around 'main', in single-class settings

with only one kind of object, this is not an issue. This is shown in Chapter 5 where RCC achieves results on the FSC-147 dataset competitive with exemplar-based methods without the need for exemplar images.

In an exemplar-free setting with multiple types of objects, the objective is a lot less clear. When multiple types of objects are present and no guidance is given, there is an ambiguity of which count or counts are the desired output. An end user may want a single count, a specific set of counts, or all of the possible counts. There are also various ways to divide up a set of objects (size, colour, shape, semantic information) which would lead to multiple counts being associated with a single object.

To provide meaningful information to a user or to evaluate the accuracy of a method, it is important to be able to disambiguate the counts. This is achieved by either providing enough information to explain the count or by matching the predicted counts to the known class-count labels. This matching is non-trivial and likely not exclusive to those defined by the dataset creator. It is likely the data contains what we call *valid-but-unknown counts*, which in turn the network will find.

Valid-but-unknown counts are associated with the ways of grouping objects differently from the known target labels. They are likely to arise in any class-agnostic counting problem but are especially impactful in the context of exemplar-free multi-class class-agnostic counting. There are many types of valid-but-unknown counts, for example:

- **Super-set counts**, it would be valid to count 4 dogs and 3 cats as 7 'animals'.
- **Sub-set counts**, 6 chess pieces could be counted as 2 white pieces and 4 black pieces.
- **Orthogonal-counts**, 2 red cars, 2 blue cars, 3 red boats and 3 blue bloats could be counted as 5 red objects and 5 blue objects or as 4 cars and 6 boats.

The idea of valid-but-unknown counts is closely related to our intrinsic and non-intrinsic task paradigm posed in Chapter 4.

2.3.9 Conclusion

This thesis was motivated by the mentioned prior work, their successes, and their limitations. Prior clustering methods are often inflexible, requiring hyper-parameter tuning, large amounts of knowledge about the dataset as a whole, or only finding the intrinsic clusters. In Chapter 4, we expand upon current clustering methods with DMS. DMS is a flexible and fully-differentiable clustering algorithm. It utilises pairwise side-information to develop a task-specific understanding of similarity.

Prior to our work, counting methods required a static class composition or exemplar images during training and inference to define the type of object to be counted. Most of them also generally utilised point-wise instance labels. In Chapters 5 and 6, we propose RCC and ABC123. RCC was the first exemplar-free class-agnostic counting method and the first weakly-supervised class-agnostic counting method. It not only demonstrates that counting objects of novel type without an exemplar is possible and that class-agnostic methods can be trained using only image-wise counts for supervision, but it also shows that they can be competitive with exemplar-based methods in single-class environments.

ABC123 is currently the only exemplar-free multi-class class-agnostic counting method. Unlike prior methods, it can generate counts for multiple novel types of objects simultaneously. Chapter 6 also introduces the idea of example finding. This paradigm turns the prior class-agnostic counting methods on their head. Instead of providing exemplars of the type to be counted, it performs exemplar-free counting and then generates examples of the types that have been counted.

In order to train and evaluate RCC and ABC123, we need robust and reliable datasets. As discussed, FSC-147, the most popular class-agnostic counting dataset,

only addresses single-class situations. It also contains errors. As such, we present FSC-133 and MCAC in Chapter 3. FSC-133 is an updated version of FSC-147, which removes its errors and ambiguities. MCAC is the first multi-class class-agnostic counting dataset.

3

Our Datasets

FSC-147 [23] is the most popular dataset in the field of class-agnostic counting and has been crucial for the development of the field. However, it does have its limitations, and includes some errors. In this chapter, we outline these errors and limitations. We then propose solutions in the form of FSC-133 and MCAC. FSC-133 is an updated version of FSC-147 that removes the errors and ambiguities we found. We utilise this in Chapters 5 and 6 to train and fairly evaluate our methods' performances in single-class settings.

MCAC is a new dataset that overcomes the most significant limitation of FSC-147, that each image only contains a single labelled class. It is the first multi-class class-agnostic counting dataset. In Chapter 6, we use MCAC to train a multi-class method as well as to demonstrate that previous class-agnostic counting methods do not function in these settings as previously assumed.

3.1 FSC-147

FSC-147 [23] is a recent few-shot counting (FSC) dataset. It was introduced in 2021 and has since become the most widely used dataset in the field of class-agnostic counting. This is due to its diverse range of images, which contain varied classes, compositions, and cameras. The two most significant features of a class-agnostic counting dataset are (i) its ability to measure the accuracy of a method at finding the correct count and (ii) its ability to test how well a method generalises to previously unseen classes. The former requires an accurate knowledge of the number of objects in each image. The latter is achieved by having complete mutual exclusivity between the classes seen during training, validation, and testing. FSC-147 includes 6135 images. Each image has point annotations for each object instance and three random instance bounding box annotations, all of which were manually gathered. FSC-147 is a single-class dataset; only one labelled type of object is present per image. While there are some examples which contain more than one kind of object, these are rare and only one class is labelled.

3.1.1 Limitations

While FSC-147 has proven invaluable as a dataset for counting, the limited annotations do not allow for methods to develop a complex understanding of the task. These limitations include the small number of bounding box annotations per image, the lack of pixel-wise segmentations of any objects, and no information on the occlusion of objects in an image. The lack of information about instance occlusion can lead to methods struggling to gain a coherent understanding of counting in high-occlusion scenarios. Most methods that use FSC-147 utilise a loss function based on a regressed density map prediction. Instead of their loss comparing the generated density map to a ground truth density map, they must compare to a pseudo-ground-truth density map which features a Gaussian kernel

centred at each instance annotation. These Gaussian kernels lose a lot of the salient information about the objects they represent, like shape and scale, which is then lost to the method. Methods also develop an inconsistent view of which objects should be counted when there is occlusion involved. This is due to the fact it is not possible to construct a set of rigorous rules about which objects are too occluded to be counted, meaning these decisions are made subjectively and therefore inconsistently. This is made clear with the examples we present in Section 3.1.2.

The single-class nature of this dataset means that methods trained on it are not incentivised to learn discriminative features. The tasks can often be solved by counting all objects present rather than focusing on whether the objects in the image are similar to each other or the exemplar. This leads to the methods becoming 'generic object detectors' rather than counters which locate repeated instances of the same type. The images are also relatively simple, with a clear class to count. This is made clear by our work in Chapter 5 which achieves state-of-the-art results on this dataset with no user input. The lack of multi-class and highly complex images limits the capacity of methods using this dataset to be developed or tested in many realistic settings where more than one kind of countable object could be present.

3.1.2 Errors

There are errors in FSC-147, including incorrect annotations and, more significantly, the repetition of images and the overlap between training, validation, and test sets. Some of these errors are presented in Figure 3.1. These errors undermine the judgements that can be made about the accuracy of a given method as well as a method's ability to generalise to unseen classes.

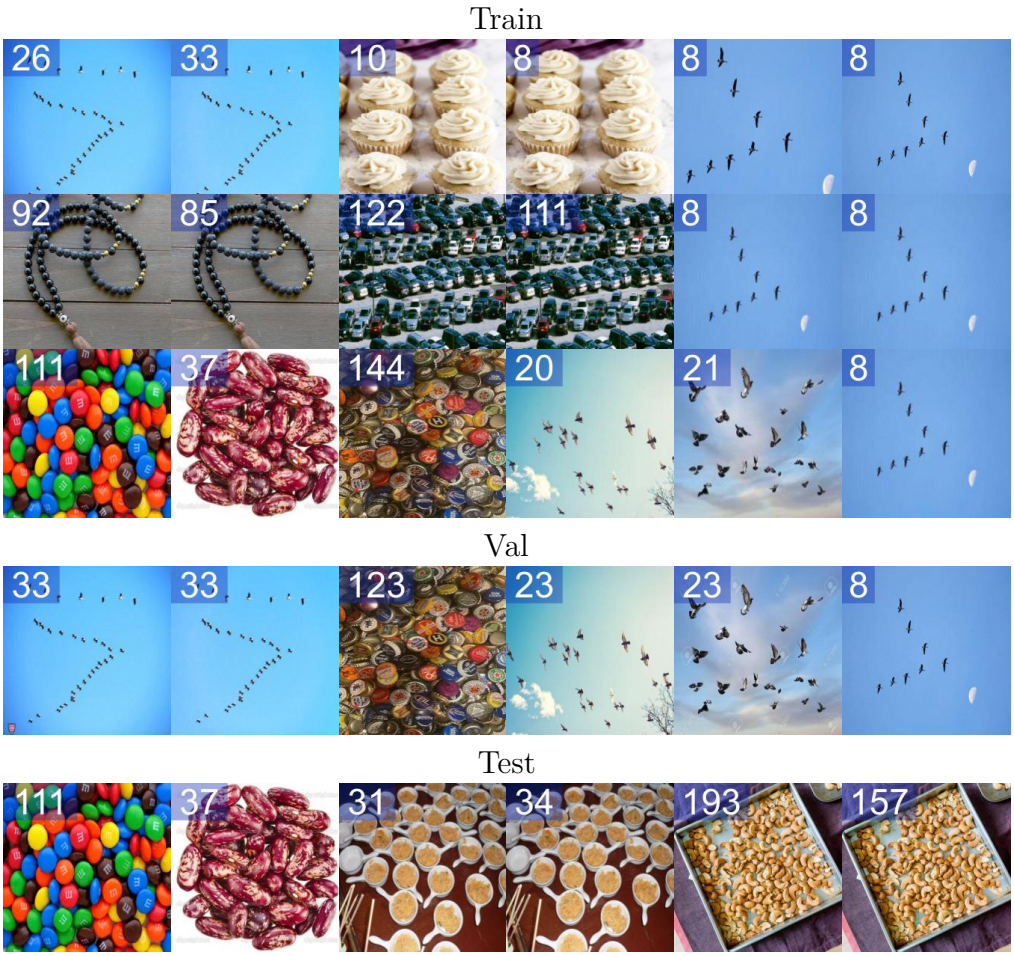


Figure 3.1: Examples of some of the 448 identical or close to identical images that appear in FSC-147 with different image IDs and their associated 'ground truth' counts. Duplicates can occur with different count labels and/or in different splits.

Annotation Errors

Annotation errors can be grouped into various categories. We will focus on (i) Missing counts, (ii) Additional counts, and (iii) Ambiguous annotations. Missing counts are cases where the labels do not include an object that is very similar to those that have been labelled; see Figure 3.2 for examples. Additional counts are cases where instance annotations are added either to objects that are not semantically similar, to objects that are so occluded as to be not significantly visible in the picture, or to erroneous points; see Figure 3.3 for examples of each. There are various cases of ambiguous labels where instance annotations or bounding boxes

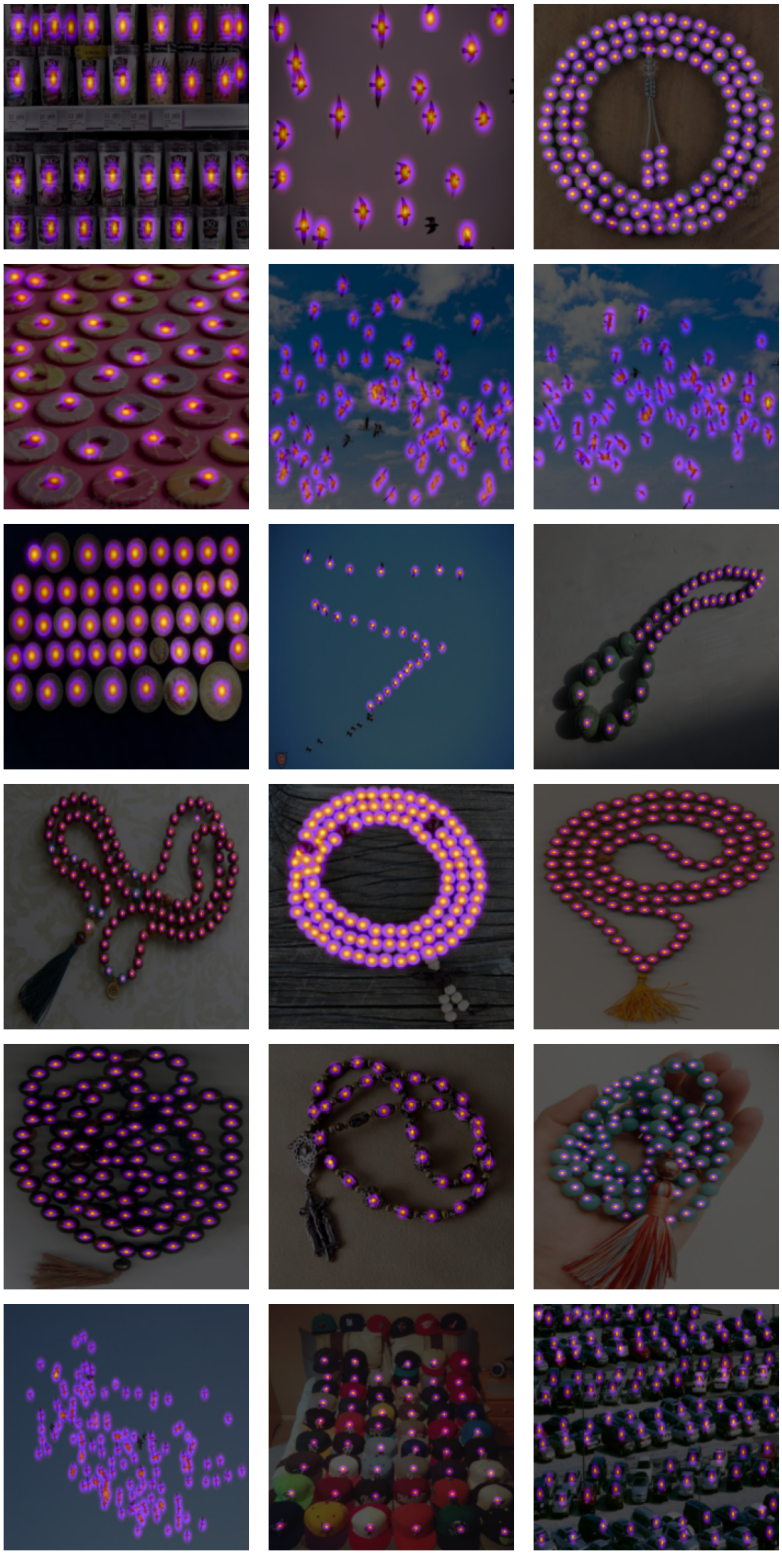


Figure 3.2: Examples of cases in the FSC-147 with missing annotations. There are object instances in these images that we believe are semantically similar enough to be included in the count.

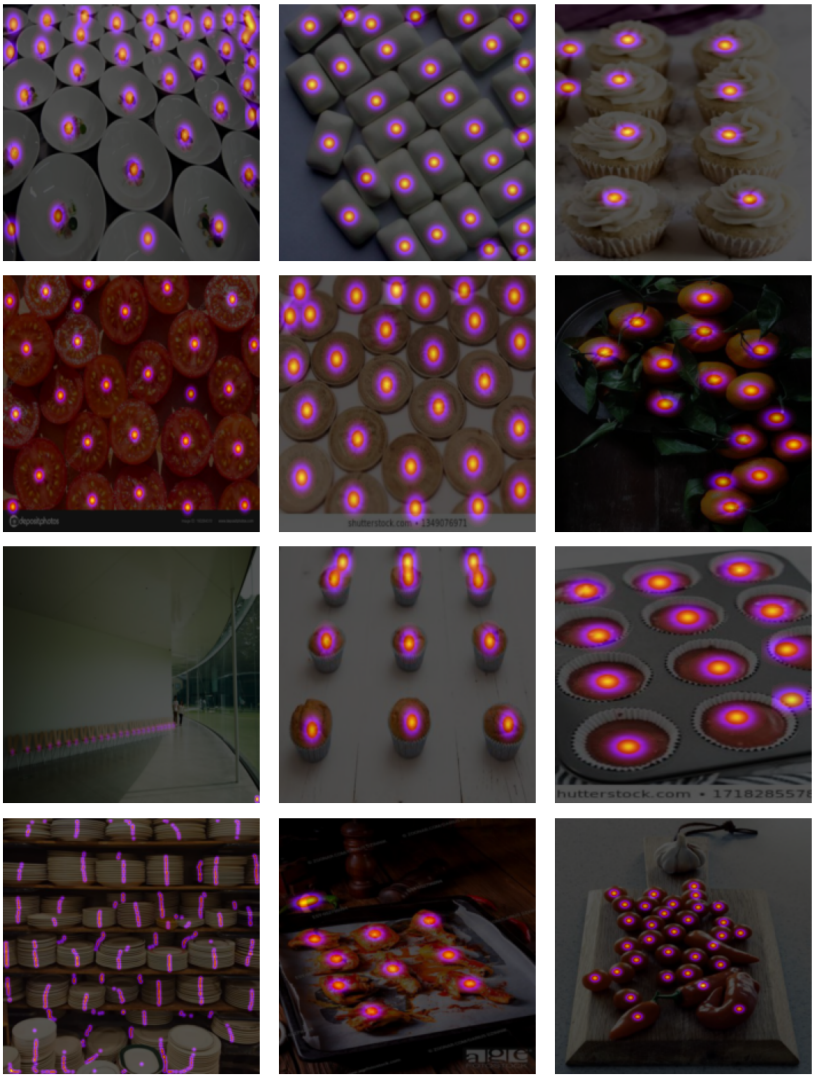


Figure 3.3: Examples of cases in FSC-147 with additional instance annotations. These either include highly occluded instances, include annotations for objects of different classes (*e.g.* chilli pepper vs chicken wing), or have erroneous annotations.

do not align with each other. Examples of this include bounding boxes that include more than one instance in situations where this was avoidable, instance annotations associated with highly occluded objects, or bounding box annotations for repeated component parts of objects with instance annotations corresponding to the whole objects; see Figure 3.4 for examples of each.

Many of the missing or additional annotation cases are due to either a lack of semantic understanding, which causes people to group non-similar objects or ignore similar objects, or difficulties with judging occlusion. One of the criteria of

Figure 3.4: Examples of ambiguous annotations in FSC-147. These include bounding box annotations that include more than one point annotation, point annotations and bounding box annotations defining different classes, arbitrarily selected classes to count, or arbitrarily placed point annotations.

the dataset is that images have "no severe occlusion" such that it would "prevent humans from accurately counting the objects" [23]. However, this is subjective, leading to annotation errors. The lack of consistency associated with manual counting, especially concerning occluded instances, is especially clear when the same image appears multiple times in the dataset with different associated counts. This phenomenon is further discussed in Section 3.1.2.

Repetition Recognition

Repeated images in a dataset are problematic for multiple reasons. The least significant effect is that unbalanced datasets reduce the performance of methods as they are often less generalisable. The most impactful effect occurs when the

Figure 3.5: Images that have a low pixel-wise difference that we deemed to be different.

Figure 3.6: Examples of images at the upper bound of pixel-wise difference we considered that we deemed to be the same image. The pixel-wise differences of the images in each column when resized to 224×224 are 5207, 5403, 7551, 8224, and 9396 respectively.

same image is present during training and evaluation. This undermines the test of a method's generalisability.

We identified repeated images using the L1 pixel-wise difference of the two 8-bit images at a resolution of 224 × 224 pixels. We manually examined each pairing with a difference of less than 10,000 to assess the similarity. This manual stage was needed as many images with low pixel-wise difference were clearly different and many with high difference were clearly the same; see Figure 3.5 and Figure 3.6 for examples of each phenomenon. The cases where the same image has a high pixel-wise difference generally occurred when the image had been compressed, cropped, translated small amounts, or had a colour filter or watermark added.

Image A				Image B				Count Diff		
ID	Class	Count	Set	Id	Class	Count	Set	Abs	Rel	Kept
4399	cranes	26	train	4549	flamingos	33	val	7	0.21	4399
4399	cranes	26	train	4719	seagulls	33	val	7	0.21	4399
2891	caps	144	train	1896	bottle caps	123	val	21	0.15	1896
4386	cranes	20	train	7415	birds	23	val	3	0.13	4386
6567	pigeons	21	train	929	birds	23	val	2	0.09	6567
4664	geese	11	train	6873	birds	11	val	0	0.00	4664
4613	geese	26	train	6714	birds	26	val	0	0.00	6714
4350	cranes	8	train	4704	seagulls	8	val	0	0.00	4683
4350	cranes	8	train	4707	seagulls	8	val	0	0.00	4683
3506	m&m pieces	111	train	3698	candy pieces	111	test	0	0.00	3506
3791	kidney beans	37	train	3494	red beans	37	test	0	0.00	3791

Table 3.1: The 17 cases where the same image appears in more than one of the training, validation, or test sets. The right column denotes which of these images (or if a third image) appears in FSC-133.

FSC-147 is said to contain 6135 unique images from 147 distinct classes. We found that 159 images occur more than once with a pixel-wise difference of 0, for a total of 334 images in the dataset. If we include images that are close to identical but do not have a zero pixel-wise difference, these numbers increase to 211 images that appear 448 times. See Figure 3.1 for illustration of some of the duplicates.

Not only do some images appear multiple times in the dataset, but 11 images appear in the training set and either the validation set or the testing set; see Table 3.1 for all of the cases. This situation should not be possible as the dataset divisions are made along class boundaries. Since the purpose of having these splits is to test the ability of a method to generalise to previously unseen classes, this issue is especially significant.

Another issue is that there are 71 cases where the same image appears more than once with a different count. The count discrepancies are up to 25%, including 5 which appear in both the training set and the validation set with 9%-21% discrepancy. Figure 3.1 presents examples of these errors, and Table 3.2 contains the complete breakdown of all the cases we found.

We believe these errors stem from the fact the images in FSC-147 were scraped

Image A				Image B				Count Diff		
ID	Class	Count	Set	ID	Class	Count	Set	Abs	Rel	Kept
5223	donuts tray	9	val	5664	donuts tray	12	val	3	25%	5223
613	seagulls	7	val	623	seagulls	9	val	2	22%	623
4399	cranes	26	train	4549	flamingos	33	val	7	21%	4683
4399	cranes	26	train	4620	geese	33	train	7	21%	4683
4399	cranes	26	train	4719	seagulls	33	val	7	21%	4683
3812	cupcakes	10	train	5238	cupcakes	8	train	2	20%	5238
5142	cashew nuts	193	test	5801	cashew nuts	157	test	36	19%	5801
4634	geese	18	train	6143	geese	15	train	3	17%	4634
4634	geese	18	train	6669	geese	15	train	3	17%	4634
6850	cereals	25	train	7337	cereals	21	train	4	16%	7337
6839	watches	58	test	989	watches	69	test	11	16%	6839
1896	bottle caps	123	val	2891	caps	144	train	21	15%	1896
2717	oranges	94	train	6573	oranges	110	train	16	15%	2717
3648	cashew nuts	35	test	5141	cashew nuts	30	test	5	14%	5141
3014	mini blinds	26	train	7307	mini blinds	30	train	4	13%	3014
4386	cranes	20	train	7415	birds	23	val	3	13%	4386
2970	fishes	16	train	6953	fishes	18	train	2	11%	6953
5929	eggs	76	test	6852	eggs	68	test	8	11%	6852
4295	green peas	110	test	5504	green peas	122	test	12	10%	4295
6567	pigeons	21	train	929	birds	23	val	2	9%	6567
2743	oranges	57	train	7363	oranges	52	train	5	9%	7363
6138	cars	122	train	6864	cars	111	train	11	9%	6138
4220	biscuits	11	train	5422	biscuits	10	train	1	9%	5422
3282	finger foods	31	test	3285	finger foods	34	test	3	9%	3282
3462	beads	92	train	5638	beads	85	train	7	8%	3462
4040	beads	117	train	5760	beads	108	train	9	8%	4040
3492	polka dots	25	val	5738	polka dots	27	val	2	7%	3492
3486	polka dots	81	val	5740	polka dots	87	val	6	7%	3486
6917	cereals	29	train	7353	cereals	27	train	2	7%	7353
5209	donuts tray	17	val	5656	donuts tray	16	val	1	6%	5656
4013	beads	115	train	5754	beads	108	train	7	6%	5754
524	geese	30	train	635	cranes	32	train	2	6%	635
4300	green peas	83	test	5506	green peas	88	test	5	6%	4300
3287	macarons	16	train	428	macarons	17	train	1	6%	3287
4634	geese	18	train	4675	geese	17	train	1	6%	4634
6602	pencils	38	train	7400	pencils	36	train	2	5%	7400
3475	beads	108	train	5637	beads	103	train	5	5%	3475
2357	bricks	276	train	2401	bricks	263	train	13	5%	2401
4044	beads	115	train	5335	beads	109	train	6	5%	4044
4019	beads	105	train	5771	beads	109	train	4	4%	4019
4049	beads	26	train	5755	beads	27	train	1	4%	5755
3956	candles	46	train	5309	candles	44	train	2	4%	5309
3969	candles	23	train	5317	candles	24	train	1	4%	3969
3950	candles	24	train	5312	candles	25	train	1	4%	3950
3020	mini blinds	34	train	7630	mini blinds	35	train	1	3%	3020
3668	toilet paper rolls	31	val	5154	toilet paper rolls	32	val	1	3%	3668
3679	buns	35	train	5026	bread rolls	36	train	1	3%	3679
4014	beads	108	train	5749	beads	105	train	3	3%	4014
3645	cashew nuts	60	test	5797	cashew nuts	58	test	2	3%	3645
5148	cashew nuts	36	test	5803	cashew nuts	35	test	1	3%	5148
7640	apples	32	test	7665	apples	33	test	1	3%	7665
3819	cupcakes	36	train	5241	cupcakes	35	train	1	3%	5241
3754	pearls	131	train	5185	pearls	127	train	4	3%	5185
3666	toilet paper rolls	32	val	5157	toilet paper rolls	31	val	1	3%	5157
3639	jade stones	31	train	5134	jade stones	32	train	1	3%	3639
6200	coins	52	train	6228	coins	53	train	1	2%	6200
267	beads	59	train	3242	beads	60	train	1	2%	267
6798	birds	87	val	976	birds	85	val	2	2%	976
3795	kidney beans	53	train	5547	kidney beans	52	train	1	2%	5547
3955	candles	60	train	5306	candles	61	train	1	2%	3955
2671	bowls	60	train	6724	bowls	59	train	1	2%	6724
6738	mini blinds	53	train	7130	mini blinds	52	train	1	2%	7130
5053	beads	139	train	5644	beads	136	train	3	2%	5644
3426	polka dots	219	val	5046	polka dots	215	val	4	2%	5046
3122	coffee beans	90	train	3134	coffee beans	89	train	1	1%	3134
6961	cartridges	178	train	7680	cartridges	179	train	1	1%	6961
4016	beads	109	train	5333	beads	108	train	1	1%	5333
3469	beads	74	train	5050	beads	75	train	1	1%	3469
4021	beads	106	train	5759	beads	107	train	1	1%	5759
5149	cashew nuts	111	test	5807	cashew nuts	110	test	1	1%	5807
4044	beads	115	train	5767	beads	116	train	1	1%	4044

Table 3.2: The 71 cases where the same image appears more than once in FSC-147 with different associated counts. The right column denotes which of these images (or, in some cases a third image) appears in FSC-133.

from Flickr, Google, and Bing search engines and then manually annotated [23]. We assume the images appeared on either multiple sources or were duplicated from the same source.

Split overlap

While images appearing in both the training and evaluation sets would be a problem for any machine learning application, it is especially impactful to the evaluation of class-agnostic counting methods. To facilitate testing a method's generalisability, the data splits are along class boundaries. Images that are incorrectly classified can end up in the wrong data split and harm the reliability of this evaluation.

There are a few instances in FSC-147 where images are clearly incorrectly classified. Some of these errors are obvious and stem from a misapplication of the class labels. For example, 'bottle caps' being classified as 'caps', a category designed for baseball caps. However, the majority of incorrect classifications stem from ambiguities in the class labels. FSC-147 has sets of classes which are visually very similar such as 'red beans (test)' and 'kidney beans (train)' or many of the bird classes. Identifying the bird classes is particularly challenging as it is difficult to recognise which of the categories is present when the instances are small or in silhouette. There are also hierarchical arrangements of classes. For example, 'cranes (train)', 'seagulls (val)', and 'geese (train)' could all also be classified as 'birds (val)', and 'buns (train)' and 'baguette rolls (train)' could also be classified as 'bread rolls (train)'. See Figure 3.7 for visual examples of these similarities.

These errors could occur when there is a language, expertise, or cultural difference in the labelling population. For example, 'Cranes' and 'geese' look very similar. Presented with an image and a set of ambiguous class labels, a human annotator may well assume the image is of the type with which they are more familiar. Similarly, the 'bread rolls', 'buns', and 'baguette rolls' categories depend on a linguistic and

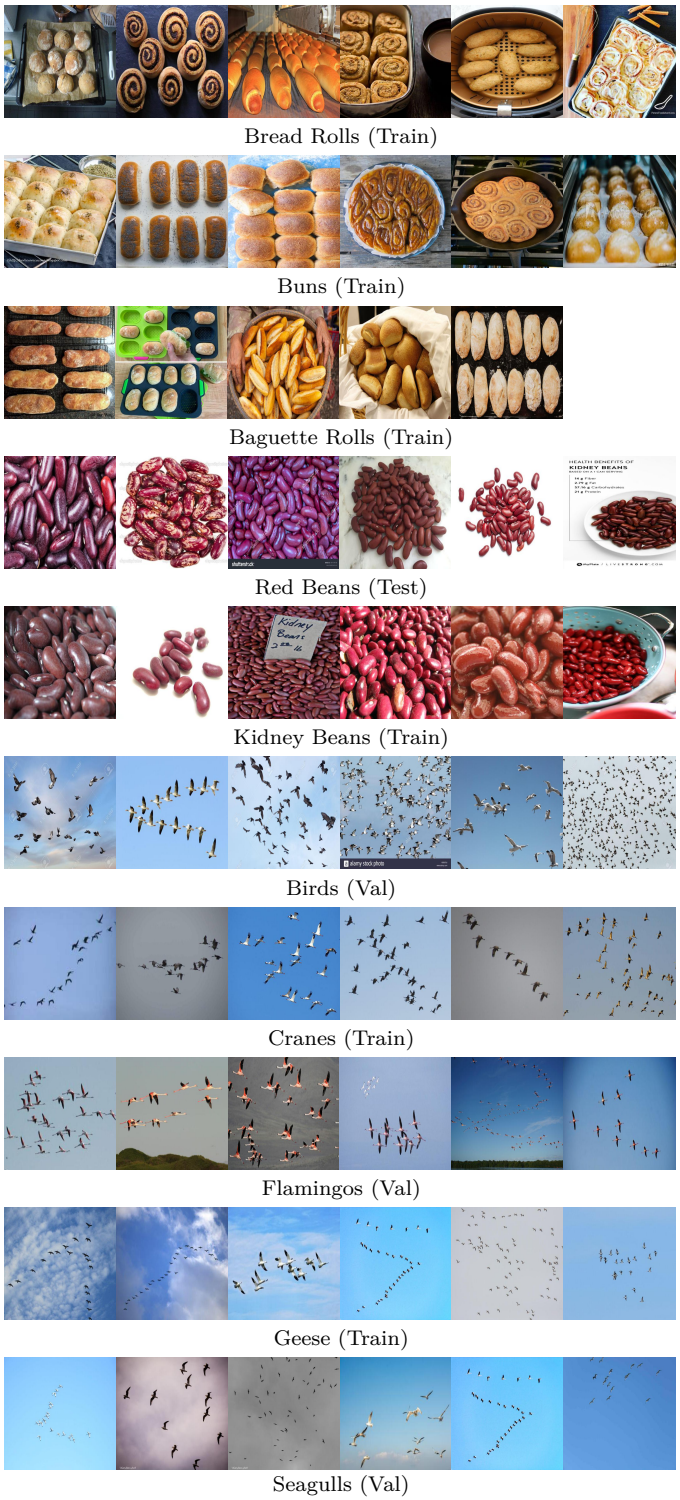


Figure 3.7: Examples of clear overlap or similarity between classes. Each row has a different class label in FSC-147. In FSC-133, these are simplified to Bread Rolls (Train), Kidney Beans (Train), and Birds (Train).

cultural understanding of the difference between these categories. We suggest combining categories that are similar enough that they could be easily mistaken for each other. This is both to disambiguate the dataset but also because we believe they do not fit the 'few-shot counting' description, *e.g.* training on images of geese in silhouette against a blue sky and then evaluating on cranes in silhouette against a blue sky is not a good test of the generalisability of a method.

Count discrepancy

As there is no rigorous way to determine the occlusion of objects from scraped images, there is no rigorous way of deciding if an instance should be included in the count or not. When an object is occluded by either another object or the boundary of the image, it is subjective whether that object is 'worth counting', which can lead to inconsistent labelling. This subjectivity is particularly highlighted when the same image appears more than once with a difference in the associated count value. Figure 3.1 shows examples of some of the cases where the same image appears with a discrepancy in the count and Table 3.2 details all of the cases. These cases are clearly errors, and we were able to identify them due to the image repetition. However, there are likely other images with incorrect annotations in the dataset that we were not able to identify.

While Ranjan et al. [23] note that they remove most images where 'severe occlusion prevents humans from accurately counting the objects', it is clear that some of these images remained. We do not suggest removing these images completely because we believe that while it is difficult to get an accurate human count for them, they are of value to the dataset. Instead, we evaluated which count we believed to be most accurate and removed the duplicates.

3.2 Proposed Solution (FSC-133)

We propose a revised version of this dataset, FSC-133, which corrects the issues outlined above. To disambiguate the dataset and help it more reliably measure a method's generalisability, FSC-133 combines categories that are similar enough to be easily confused; see Table 3.3 for the details on which classes were moved or combined.

FSC-133 has 5898 images in 133 classes. The training, validation, and testing sets have 3877, 954, and 1067 images from 82, 26, and 25 classes respectively. When combining classes that overlapped data splits, we merged them into the training split. This allows methods trained on FSC-147 to still be tested on FCS-133. While these methods would be disadvantaged, the tests still fairly evaluate their ability to count completely unseen classes.

As FSC-133 is a subset of FSC-147, it still has many of the same limitations. The most significant is the lack of thorough instance labels. There is only a small set of bounding boxes per image, there is no information about occlusion, and each image only has one kind of object labelled. All of these limitations restrict methods trained with it from learning a meaningful understanding of counting in complex single-class settings, and there is no reason to believe that methods trained on this dataset will function robustly in multi-class settings. These limitations are addressed in MCAC, our dataset, presented in Section 3.3.

3.3 Multi-Class Class-Agnostic Counting Dataset

In this section, we detail our new dataset, MCAC. MCAC is the first dataset designed for multi-class class-agnostic counting. As discussed previously, prior to 2023, class-agnostic counting methods, including our own, presented in Chapter 5 and Hobley and Prisacariu [125], focused on enumerating instances in images where only a

Train				
alcohol bottles	**baguette rolls**	balls	bananas	beads
bees	birthday candles	biscuits	boats	bottles
bowls	boxes	bread rolls	bricks	buffaloes
buns	calamari rings	candles	cans	caps
cars	cartridges	cassettes	cement bags	cereals
chewing gum pieces	chopstick	clams	coffee beans	coins
cotton balls	cows	**cranes**	crayons	croissants
crows	cupcake tray	cupcakes	cups	fishes
geese	gemstones	go game	goats	goldfish snack
ice cream	instant noodles	jade stones	jeans	kidney beans
kitchen towels	lighters	lipstick	m&m pieces	macarons
matches	meat skewers	mini blinds	mosaic tiles	naan bread
nails	nuts	onion rings	oranges	pearls
pencils	penguins	pens	people	peppers
pigeons	plates	polka dot tiles	potatoes	rice bags
roof tiles	screws	shoes	spoon	spring rolls
stairs	stapler pins	straws	supermarket shelf	**swans**
tomatoes	watermelon	windows	zebras	
Val				
ants	birds	books	bottle caps	bullets
camels	chairs	chicken wings	donuts tray	**flamingos**
flower pots	flowers	fresh cut	grapes	horses
kiwis	milk cartons	oyster shells	oysters	peaches
pills	polka dots	prawn crackers	sausages	**seagulls**
shallots	shirts	skateboard	toilet paper rolls	
Test				
apples	candy pieces	carrom board pieces	**cashew nuts**	comic books
crab cakes	deers	eggs	elephants	finger foods
green peas	hot air balloons	keyboard keys	legos	marbles
markers	nail polish	potato chips	**red beans**	**sauce bottles**
sea shells	sheep	skis	stamps	sticky notes
strawberries	sunglasses	tree logs	watches	

Table 3.3: The breakdown of classes in FSC-147. Note there are identical categories, *e.g.* 'red beans' and 'kidney beans' are semantically the same as are 'baguette rolls', 'buns', and 'bread rolls'. There are also hierarchical categories, *e.g.* 'cranes', 'geese', 'pigeons', 'seagulls', 'crows', 'swans', and 'flamingos' could be classified as also 'birds'. We show the classes that are combined in FSC-133 in **bold** and show the classes that are moved to the training set in FSC-133 in blue.

single class of object is present to count as this was the available data. This training leads to methods with very broad intra-class definitions, approximating a 'generic object detector'. In real-world applications, it is not necessarily the case that the only objects present are of the class to be counted. While many prior methods have been assumed to generalise to multi-class settings, this has not been shown robustly. A multi-class dataset would facilitate the evaluation of these methods on multi-class images as well as facilitate the training of new multi-class counting methods.

3.3.1 Multi-Class Counting Problem Definition

The goal of a multi-class class-agnostic counting method is to take a single image that contains multiple instances of multiple classes and return the associated counts of each. We recognise that '*exemplar-free* multi-class class-agnostic counting' is not a clearly defined task and could be interpreted as multiple different problems, each fitting a different set of requirements. A user may only want to gain a better understanding of the distribution of objects in an image. However, it is likely that they would want to know a more directed piece of information, 'How many objects are there of each of a set of classes that I care about?'. While in some contexts, it would make sense to have the user specify this preference in the form of an exemplar image or a textual description, there are cases where this would be laborious and time-consuming. As such, we create a dataset which facilitates the training of methods to address either of the above set of requirements. We also provide extra information potentially useful for approaches we have not considered.

While the deployment query scenario, 'Count given an unlabelled image of objects', is natural, the training and quantitative evaluation of methods to address it is not. To facilitate training and evaluation of methods in multi-class settings, we need images with multiple objects of multiple types. To evaluate a method's generalisability to unseen object types, the classes present in the images need to

be mutually exclusive between training, validation, and testing. It is infeasible to gather natural images with (a) a wide variety of classes, (b) a wide variety of the number of times an object appears in an image, and (c) no repetition of the types of object between the train, test, and validation splits.

While it may seem that datasets created for detection and segmentation tasks could be utilised, they generally focus on small sets of classes. Images from these datasets are also not likely to be separable into sets with mutually exclusive classes. The labelling of these classes is also generally not sufficient for many of the current class-agnostic counting methods.

Another possible solution would be to tile existing single-class images, similar to the approach taken by Liu et al. [25]. While not focused on multi-class counting, Liu et al. [25] aimed to improve their method's class-discrimination using a tiling modification. They combined up to four images from FSC-147 to create a composite image. This was used to create a system that is more likely to count only objects similar to the exemplars rather than acting as a generic 'object counter' as many previous methods were. While this modification effectively helped with their goal of greater discrimination, it is not a robust simulation of real-world multi-class images. This is because since the classes are spatially separable, a method can address the four corners of the image independently. As such, this solution should not be used to evaluate a method's multi-class capabilities.

3.3.2 Multi-Class Class-Agnostic Synthetic Dataset

We address the problems of the previous datasets and data augmentation approaches with a new Multi-Class Class-Agnostic Counting (MCAC) dataset. In this section, we discuss the implementation of our dataset. We explain our reason for choosing a synthetic rather than a photographic dataset. We then discuss how the dataset was generated and the ways we think it could be used most effectively for exemplar-free

and exemplar-based training.

Synthetic Choice

There are three major constraints placed on a multi-class, class-agnostic counting dataset:

1. **Multi-class** - Images should contain multiple instances of multiple types of object.

2. **Class-agnostic** - Classes must be mutually exclusive between training, validation and inference time.

3. **Diverse** - Diverse classes are required to facilitate effective training and to validate a method's generalisability.

Gathering and labelling images that match all three constraints is prohibitively difficult. Due to the difficulty of producing a real dataset with accurate and consistent annotations and the flaws of augmenting single-class images, we opted to utilise synthetic data generation. While a model trained on a purely synthetic dataset may have limited application to real-world problems, we believe this gives an upper bound on the ability of a method to deal with multi-count images. We also show in Section 6.4.4 that a counting method trained on purely simple synthetic data can generalise well to a photographic dataset. So it is a general and valuable tool for work on automated counting.

Two significant advantages of a synthetic dataset are the level of control we have over the composition of each image as well as the accuracy and precision of the information we can gather about the images. The number of instances, their exact locations, their pixel-wise segmentations, and their degree of occlusion are all known to a level that would not be possible with manual labelling. This is all achieved with minimal work, making this approach cheap, and scalable.

Dataset Generation

We generate our dataset using Blender, an open-source 3D computer graphics tool. Blender includes rendering, texturing, and physical simulation capabilities, making it ideal for our application. Instead of trying to find a valid packing of a large number of complex, non-convex objects without collisions or errors, we use a physics simulator to 'drop' the objects from a non-colliding initialisation into a box, shown in Figure 3.8. This creates layouts of objects with random locations and orientations, that have objects in a relatively narrow height without clipping and with realistic occlusion. As objects in real settings often vary in size, we vary the size of objects by ±50% from a random nominal size. We also vary the number, location, and intensity of lights present. This somewhat evenly lights the scene allowing all objects to be seen while maintaining variation of shadow. We use a randomly generated multi-coloured texture as the background with random noise texture varying the surface roughness of the floor. This minimises large specular highlights while also not being simple for a network to model.

The object models are drawn from ShapeNetSem [126], a subset of ShapeNet [127]. Each object in ShapeNetSem is labelled hierarchically by type. They are also pre-textured, leading to diverse and realistic appearances. We maintain the labels from ShapeNetSem.

To enable training of various current methods [23, 25, 116] and for maximum utility in the future, our dataset is labelled in various ways. Each object has associated with it a bounding box, a center-of-geometry point annotation, a final image pixel-wise segmentation, and an unoccluded segmentation. This is significantly more annotation information than any other class-agnostic counting dataset, as discussed in Section 3.2, and more than is required by any of the contemporary methods [24, 25, 116], as discussed in Section 3.3.2 and chapter 6.

We render a zoomed-out version of the final image, the final instance segmen-

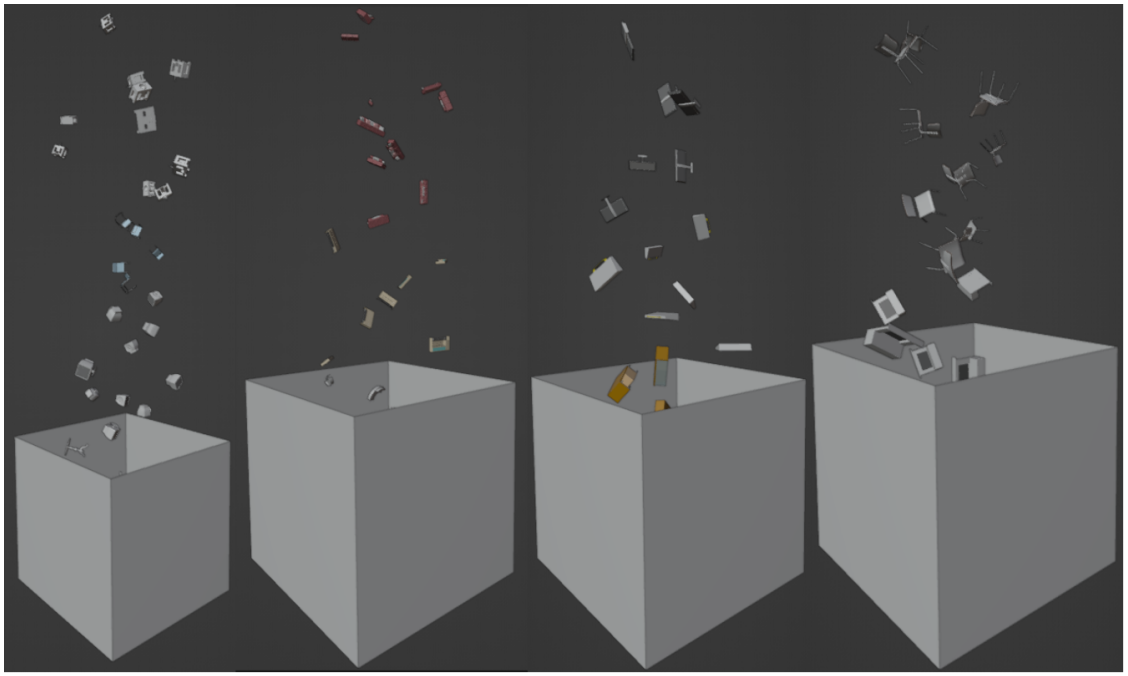


Figure 3.8: Examples of the arrangement of objects before running the physics simulation. Each object is placed at a unique height to ensure no collisions at initialisation. A physics simulator is then run, 'dropping' the objects into the box.

tation, and unoccluded pixel-wise segmentations of each object separately. These zoomed-out images are such that all objects are completely within the boundaries of the image. From this, the occlusion percentage is calculated as:

$$\text{Occlusion} = 1 - \frac{A_0}{A_1} \tag{3.1}$$

where A_0 is the number of pixels in the final image segmentation and A_1 is the number of pixels that would be seen if the object was unoccluded and was completely within the bounds of the image. The final images are cropped versions of the zoomed-out image. The bounding boxes are generated from the unoccluded segmentation by finding the tightest box that includes the instance and is within the bounds of the final image. See Figure 3.9 for examples from the dataset and Figure 3.10 for labelled examples.

Figure 3.9: Examples from the Multi-Class Synthetic Dataset.

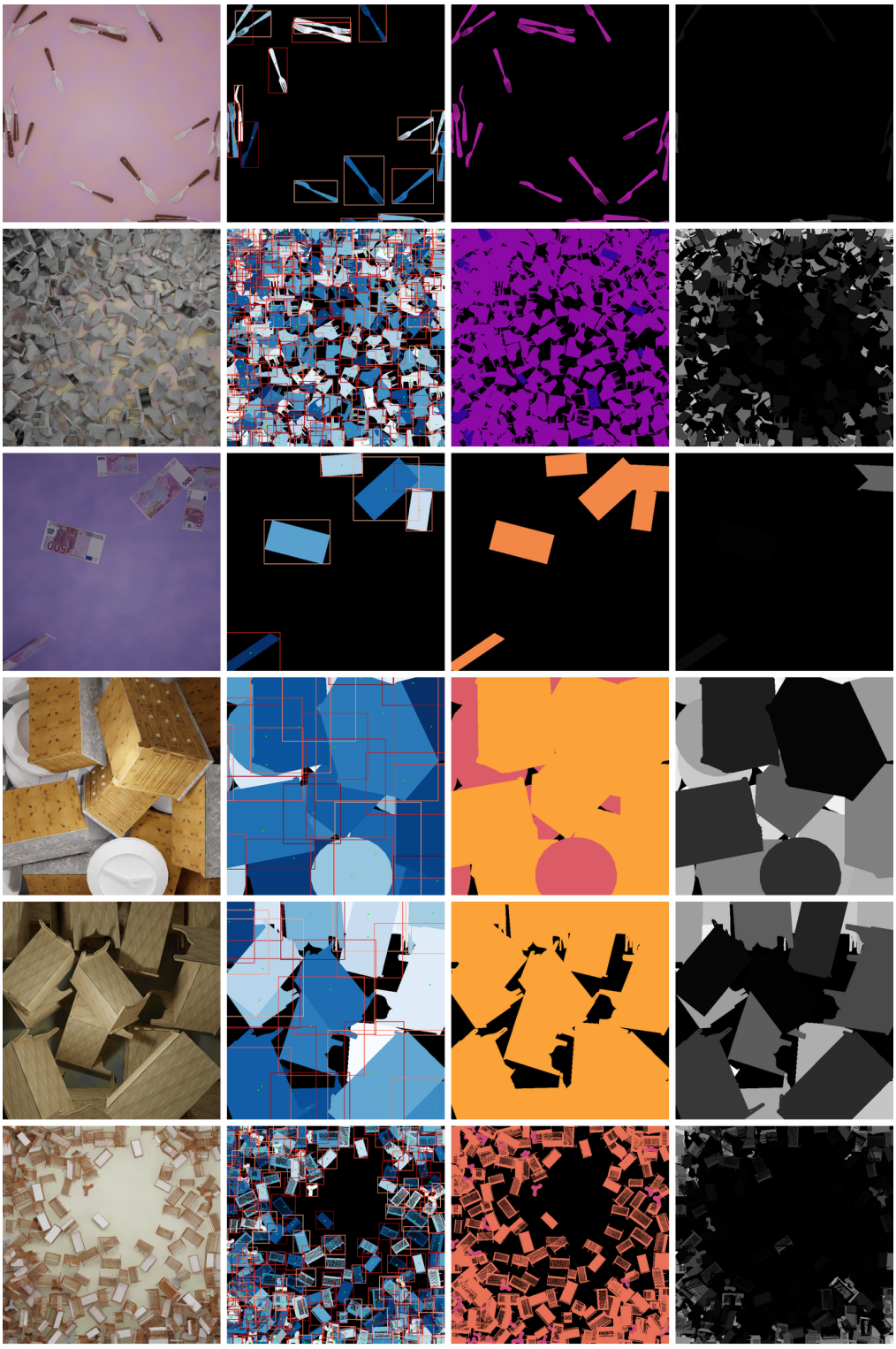


Figure 3.10: Labelled examples from the Multi-Class Synthetic Dataset. Each row contains a rendered image, the associated instance labels with bounding boxes and centre points, the class labels, and the individual instance occlusions.

		Count Value				Num Classes			
	#	Min	Median	Mean	Max	Min	Median	Mean	Max
MCAC									
Train	8298	1	23	46.87	298	1	2	1.75	4
Val	4286	1	24	46.52	297	1	2	1.78	4
Test	3640	1	25	50.80	296	1	2	1.72	4
Total	16224	1	23	47.66	298	1	2	1.75	4
MCAC-M1									
Train	2186	5	102	112.42	298	1	1	1	1
Val	1060	3	96	110.56	297	1	1	1	1
Test	1013	1	117	124.89	291	1	1	1	1
Total	4259	1	102	114.89	298	1	1	1	1

Table 3.4: The basic statistical breakdown of our multi-class dataset, MCAC, and the single-class variant MCAC-M1.

Dataset Breakdown

MCAC contains images with between 1 and 4 classes of object and between 1 and 400 instances per class. The distribution of classes per image and instances per class are shown in Figure 3.11 and Table 3.4. MCAC has three data splits. The training split has 4756 images (8298 counts) drawn from 287 classes, the validation split has 2114 images (4286 counts) drawn from 37 classes, and the testing split has 2413 images (3640 counts) drawn from 19 classes.

Many current methods were developed using images which only contain one kind of object, and so are specifically suited to this setting. In light of this, we encourage comparisons to be made not only using MCAC but also MCAC-M1. MCAC-M1 is the subset of MCAC images that contain only a single class, the breakdown of which is also shown in Table 3.4. Of course, the problems posed by MCAC-M1 are simpler than those posed in MCAC, where the number of classes is unknown. As is further discussed in Section 6.4, we believe the discrepancy in the performance of a given method trained on MCAC and MCAC-M1 demonstrates the ability of the method to discriminate between objects based on type rather than acting as a generic object counter.

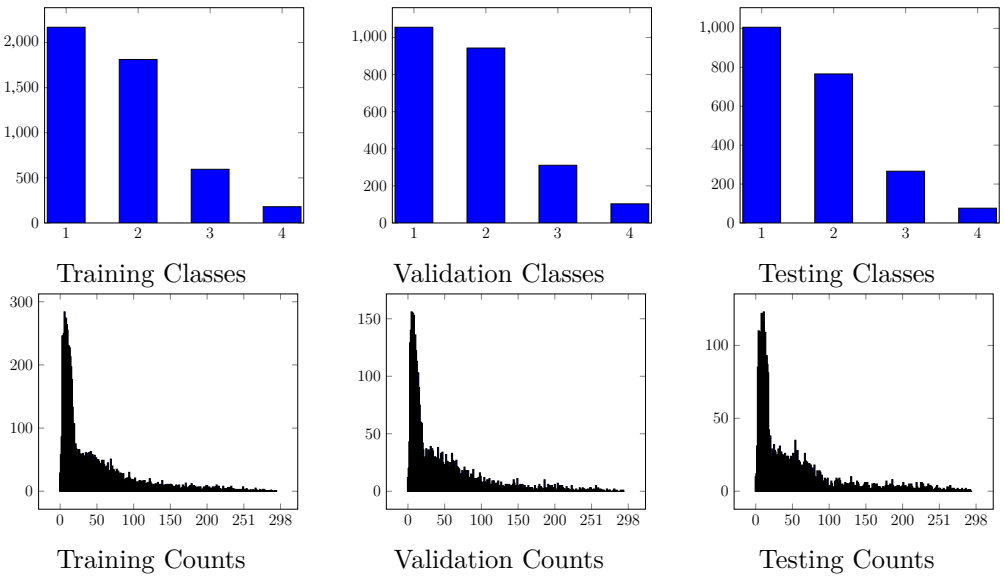


Figure 3.11: The distributions of the number of classes in an image and of the number of instances of each class per image across the dataset.

Ambiguity

Both exemplar-based and exemplar-free methods bump into problems of ambiguity. If there are objects of varied levels of generality, which boundary should be used? For example, on a chess board with a single white pawn as the exemplar, should the count be of all the pieces, all the white pieces, all the white pawns, all the pawns, and so on?

Given the infeasibility of defining every possible way of grouping the objects present in an image, we define a single way of grouping them: an identical mesh and texture, independent of size or orientation. We do, however, acknowledge the existence of other *valid-but-unknown* counts, the unlabelled ways of grouping the objects, introduced in Section 2.3.8. As the original class hierarchy labels are preserved, future works could count based on more complex definitions of similarity than we utilise.

Dataset Usage

Here, we outline a set of standards we suggest for the dataset when it is used in future works and that we follow in our later work. These are by no means the only

way the dataset can be used, and the segmentations, hierarchical class labels, and occlusion labels allow for much more complex and interesting use cases.

We group objects as the same if they share the same geometry and texture, independent of their orientation or size. As was asserted by Ranjan et al. [23], an object that can barely be seen or recognised should not be counted. While it is difficult to define these terms robustly for photographic data, our use of synthetic data allows us to set a consistent, quantitative boundary. We exclude from the final count all objects that are more than 70% occluded by either other objects or the edge of the frame. Though true density maps could be utilised as we have instance-wise segmentations, we use ground-truth pseudo-density maps to align with prior methods' approaches [23, 25, 110, 117]. These pseudo-density maps have a Gaussian kernel centred on the centre of geometry of each object with $\sigma = 8$. As is standard, when the Gaussians from different instances overlap, we sum them rather than taking the maximum value. As some of the Gaussian kernels are cut off by the edge of the frame, the sum of the density maps will be lower than the desired integer count labels. We scale each density map so the sum over it is the correct value. This adjustment to the training data improved our results on the test data by $\sim 5\%$.

To ensure that meaningful but varied exemplars are used during the training of exemplar-based methods, we take bounding boxes randomly from instances with less than 30% occlusion. We evaluate these methods using the bounding boxes of the three least occluded instances. In cases with more than three equally occluded instances, we use those with the lowest instance IDs.

3.3.3 Conclusion

In this chapter, we have proposed two datasets, FSC-133 and MCAC. FCS-133 improves upon the current most popular class-agnostic counting dataset, FSC-147, by removing errors and ambiguities. We believe these improvements not only

facilitate more representative evaluation but also lead to methods being able to learn a more consistent understanding of counting. MCAC addresses what we see as the most significant limitation of current class-agnostic counting datasets, that they are single-class. MCAC is the first multi-class class-agnostic counting dataset. It features diverse object types and counts and is labelled with a high level of accuracy and consistency. As seen in Chapter 6, MCAC enables the training of exemplar-based and exemplar-free multi-class counting methods. It also allowed us to show that methods previously believed to generalise from single-class training into multi-class settings do not in fact achieve this. Both datasets provide significant utility to our work as well as the broader field of class-agnostic counting as a whole.

4

Dataset-Agnostic Task-Specific Clustering Using Side-Information

4.1 Introduction

The ability to identify repetition is one of the two key components of counting, alongside object detection. In this chapter, we address this repetition recognition independently of detection by framing it as a clustering task. Clustering, also known as cluster analysis, involves the process of organising a collection of items in a manner that items within the same cluster exhibit greater resemblance to one another than they do to items in different clusters. Automated clustering is one of the fundamental problems in machine learning due to its varied use cases and minimal requirement for labelling. Unlike classification, most clustering methods group data points by

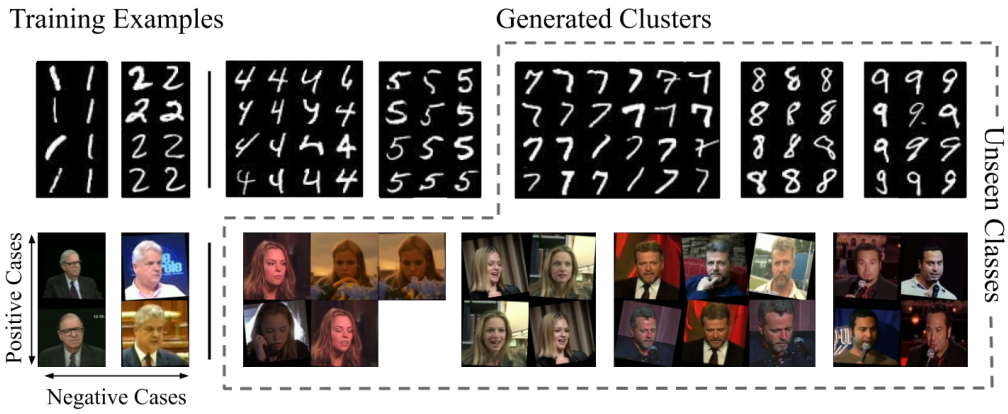


Figure 4.1: Examples of learnt clusters given side-information. Positive examples contain two data points known to be from the same class and negative examples are from different classes. Here, the same class is either the same digit (top) or the same person (bottom) and negative examples are of two different classes. Our method learns to correctly identify unseen classes. For example, the 7, 8, and 9 clusters are not present when training on MNIST (top).

type with little to no supervision and no ground-truth class labels. This enables them to operate in a class-agnostic way. They can discover previously unseen classes and be utilised in applications where there is limited labelled data available.

As discussed in Section 2.2, to separate data points correctly, classical clustering approaches normally require some kind of general holistic information about the data. This can take various forms, but most commonly it revolves around either a known number of clusters as in algorithms like k-means [40] and DBSCAN [41] or a predefined boundary for a proxy similarity metric, as in methods like mean shift [43]. Usually, this proxy is a simple metric, such as the L_1, Euclidean, or Cosine distance, in an embedded feature space. In many real-world cases, the exact number of clusters may be unknown, and it is often difficult to specify an informative distance tolerance to separate intra-class variation from inter-class variation. Defining an informative proxy similarity metric is especially difficult in high-dimensional or complex feature spaces since the usual metrics scale poorly with dimensionality. These simple metrics inherently prioritise the feature dimensions with the greatest magnitude variation, meaning they will always group the same data in the same, 'most obvious', way.

While datasets generally provide a single label per image, it is clear that many datasets could be clustered in multiple valid ways. We refer to the 'most obvious' clustering task of a dataset, which is usually the task the labels refer to, as the *intrinsic* task, and the other valid divisions as the *non-intrinsic* task; examples of possible non-intrinsic tasks are presented in Figure 4.5. This is closely linked to the idea we proposed in Section 2.3.8 of valid-but-unknown counts. Up to this point, clustering methods generally focus on identifying intrinsic tasks and have little to no ability to perform non-intrinsic clustering, *i.e.* finding groupings where the most salient features do not have the greatest variance.

At the time of writing this work, deep-learning-based methods used these classical algorithms either to supervise their training [72, 73] or to directly cluster a projected feature space which has been optimised for the task at hand [64, 68]. These approaches suffer from the same problems as their respective classical methods and generally require significant parameter tuning.

In this chapter, we propose Differentiable Mean Shift (DMS), a method inspired by both work that uses pairwise similarity-based side-information to cluster data [45, 128, 129] and mean-shift [43], an iterative clustering algorithm which uses a Euclidean distance metric as a proxy for similarity. DMS is the natural progression of these two approaches. It removes the overly simplistic similarity proxy of Euclidean distance of the original mean-shift algorithm and instead uses side-information to learn a task-specific similarity kernel. This kernel identifies the features salient to a specific task, generating an understanding of the clusters directly from the side-information. We believe that using side-information in the form of choosing a small set of 'similar' or 'dissimilar' pairs is more intuitive and easier for a user than defining an abstract distance tolerance in an embedded feature space and also works in use cases where the number of clusters is not known by an end-user.

We train the differentiable kernel to evaluate the similarity of a given pair of

points in the context of a specific task. It adjusts its understanding of the importance of different aspects of the data, learning a task-specific definition of what it means to be within the same cluster. Because of this, DMS can accurately cluster the same dataset in multiple valid ways depending on the side-information provided.

DMS is able to generate accurate clusters without the need for either a clustering-specific embedded feature space; dataset-specific hyper-parameter tuning present in other methods [46, 79, 130, 131]; a wider knowledge of the dataset such as the number of clusters, their centres, or their spatial distribution; or even all of the classes to be present during training. As our method operates as a single forward pass through a lightweight network which does not require hyper-parameter tuning, dimensionality reduction, or an understanding of the whole dataset, it is trivial to include it within another deep learning method either as a final head or integrated into the end-to-end training of another task. We do not train backbones or feature encoders; instead, we operate on the input data features directly.

In this chapter, we show that our method outperforms prior approaches in both the intrinsic and non-intrinsic dataset tasks, that it is possible to train a strong cluster definition without enforcing a constraint that each cluster must be presented during training, and that the learnt definition is flexible, learning various clustering tasks from side-information alone. This accuracy and flexibility paired with the lightweight nature of the network make DMS well-suited to various deployment situations.

Our main contributions are:

- A new approach to clustering, which directly clusters points trained using only side-information rather than using distance tolerances, a known number of clusters, or any other laborious hyper-parameter tuning.
- A differentiable, mean-shift-inspired framework that can be easily added to other machine learning methods.
- A state-of-the-art algorithm that outperforms other contemporary methods

on intrinsic tasks and is capable of performing non-intrinsic task clustering on various datasets.

The remainder of this chapter is structured as follows. Section 4.2 outlines and formalises the mathematics of the classical mean shift algorithm and our differentiable mean shift process. Section 4.3 discusses the details of our method and evaluates it in the context of other recent clustering approaches. We then validate sections of our method in Section 4.3.6.

4.2 Method

In this section, we outline the formulation of the task of clustering and summarise the original mean shift algorithm. We then outline our proposed method, DMS, which is a fully differentiable, neural network-based method. Finally, we explain our use of side-information during training and how this facilitates task-specific clustering.

4.2.1 Clustering Problem Definition

Clustering algorithms aim to group a set of n, d dimensional points $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} = \mathcal{X}$ into $k \leq n$ sets $\{S_1, S_2, ...S_k\} = \mathcal{S}$, with centres $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k\} = \mathcal{M}$.

$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{\mathbf{x} \in S_i} \mathbf{x} \tag{4.1}$$

where $|S_i|$ is the number of points in the i^{th} cluster, S_i.

4.2.2 Mean Shift Kernel

The mean shift algorithm, as presented in Algorithm 3 in Section 2.2.4, iteratively improves an estimate, $\bar{\boldsymbol{\mu}}_i$, of each cluster centre. This is achieved by finding inlier pre-

dictions using the current estimate of the cluster centre and a similarity kernel, $K(.)$.

$$\bar{\boldsymbol{\mu}}_i^{(n+1)} = m(\mathcal{X}, \bar{\boldsymbol{\mu}}_i^{(n)}) = \frac{\sum_{\mathbf{x}\in\mathcal{X}} \mathbf{x}K(\mathbf{x}, \bar{\boldsymbol{\mu}}_i^{(n)})}{\sum_{\mathbf{x}\in\mathcal{X}} K(\mathbf{x}, \bar{\boldsymbol{\mu}}_i^{(n)})}, \tag{4.2}$$

In the original mean shift formulation, the Euclidean distance flat kernel $K_F(.)$ was used. This is defined as a λ-ball $\in \mathbb{R}^d$ centred at the current estimated cluster mean $\bar{\boldsymbol{\mu}}$, where d is the dimensionality of the feature space. Points within this ball are considered inliers to the cluster.

$$K_F(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} 1 & \text{if } \|\mathbf{x}_1 - \mathbf{x}_2\|_2 \leqslant \lambda \\ 0 & \text{if } \|\mathbf{x}_1 - \mathbf{x}_2\|_2 \geqslant \lambda \end{cases} \tag{4.3}$$

$K_F(.)$ has a strong neighbourhood constraint making it easy to implement and fast to run, but it is not differentiable and requires a good knowledge of the spread of points within each cluster. Other commonly used similarity kernels are the unit Gaussian Kernel, K_G, and the truncated unit Gaussian kernel, K_T, [132].

$$K_G(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{2\pi}\sigma^d} \mathbf{e}^{-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2}} \tag{4.4}$$

$$K_T(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} \beta K_G(\mathbf{x}_1, \mathbf{x}_2) & \text{if } \|\mathbf{x}_1 - \mathbf{x}_2\|_2 \leqslant \lambda \\ 0 & \text{if } \|\mathbf{x}_1 - \mathbf{x}_2\|_2 \geqslant \lambda \end{cases} \tag{4.5}$$

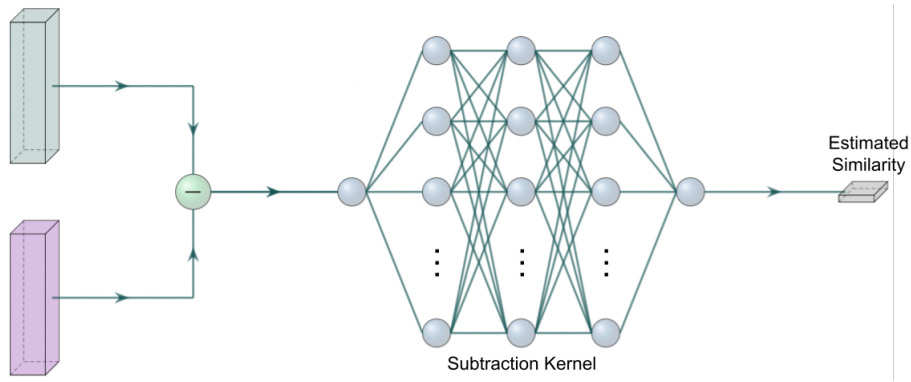
Since the Gaussian Kernel has no discrete neighbourhood characteristic, all points in $\mathcal{X}$ affect the next centre prediction. However, K_G scales the impact of various points based on their estimated inlier-ship to the current cluster, where inlier-ship is

based solely on Euclidean distance. While the Gaussian Kernel is differentiable, its representation capabilities are limited as it is symmetrical in all dimensions. While this could be improved by having a learnt scaling of different dimensions based on some other metric, it would still scale poorly to more complex feature spaces as the Euclidean distance does not generalise well to higher dimensions.

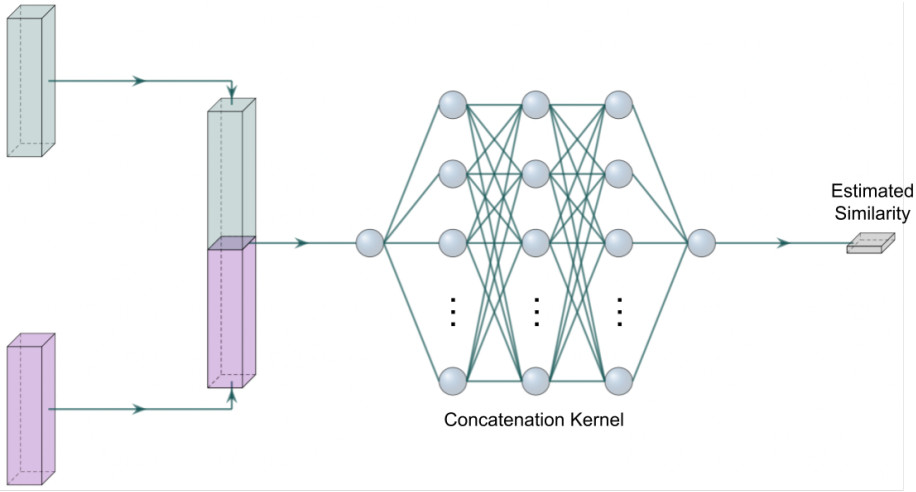
4.2.3 DMS Kernel

Instead of using Euclidean distance as a proxy for similarity, we propose a differentiable kernel based on a neural network, K_D, which learns to predict similarity more directly. The differentiable kernel $K_D(.)$ produces a probability score, predicting the level of similarity between either two arbitrary data points, $(\mathbf{x}_1, \mathbf{x}_2)$, or a data point and an estimated sample mean, $(\mathbf{x}, \bar{\boldsymbol{\mu}})$. The more similar the points, the closer the output of $K_D(.)$ is to 1. Due to the flexibility of the neural network architecture, $K_D(.)$ can learn a deeper understanding of the specific task and, as such, can give more informed predictions of similarity than the classical analytical metrics, such as the aforementioned flat or Gaussian kernels. A single K_D similarity prediction is presented visually in Figure 4.2.

In high-dimensional spaces, it is often desirable to act non-isotropically across a feature space's dimensions based on a task's requirements. This would help in cases where a dimension has the largest variance but the desired task is invariant to this variation. For example, clustering tasks which focus on semantic class should be invariant to changes in brightness, rotations, or translations. A neural network is ideal for these applications as each dimension can be acted on independently to find those most salient to a given task. In contrast, since simpler metrics, such as the Euclidean norm, are not able to apply the same contextual knowledge of the feature space, always evaluate dimensions with greater absolute variation as more significant. This ability of neural networks decreases the need for explicit



(a) Subtraction-Based



(b) Concatenation-Based

Figure 4.2: The DMS kernel predicts the similarity of two points from either (a) the difference of their features or (b) their concatenated features. All of the latent layers have batch normalisation and ReLu activations. The final layer utilises a sigmoid.

dimensionality reduction and removes the problems of poor representation inherent in such a reduction. The context-based dimensional independence also allows a single feature space to be clustered differently depending on the task.

The latent layers of K_D have rectified linear activations applied, and the output of K_D is bounded to $[0, 1]$ by a sigmoid function.

$$\mathrm{Sig}(x) = \frac{1}{1 + e^{-x}} \tag{4.6}$$

These activations provide non-linearities to the feature space allowing for much

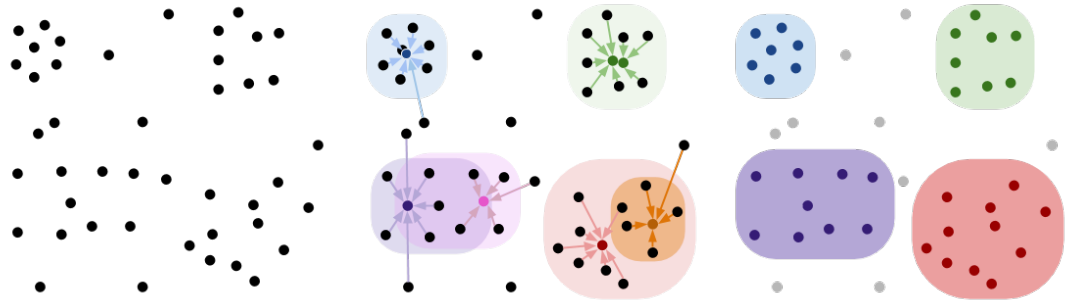


Figure 4.3: Algorithm Stages. First, the centres are located with an initial approximation of the clusters. Then, given the centres, inliers are found, the clusters are refined, similar clusters are combined, subset clusters are incorporated, and noise points are removed.

greater complexity of analysis as well as scaling the results to more closely resemble the behaviour of the traditional Gaussian kernel, increasing and decreasing the impacts of the areas close to and far from the sample mean respectively.

4.2.4 DMS Algorithm

Our method, which in practicality is a single pass through a neural network, is comprised of two stages: i) iterative centre finding and ii) refined inlier prediction. A visual layout of these stages is shown in Figure 4.3 and Algorithm 5 outlines the DMS algorithm in pseudo-code. During the first stage, the cluster centres are found by iteratively taking the weighted average of their inliers predictions, $K_D(\mathbf{x}_i, \bar{\boldsymbol{\mu}})$. In the second stage, similar centres are merged. During this conglomeration, two centres are said to be the same if their inlier predictions of all of the points in $\mathcal{X}$ are similar. The second stage also refines the inlier predictions as well as defining outlier points.

To estimate the sample mean vector of a cluster, we initialise $\bar{\boldsymbol{\mu}}^{(0)} = \mathbf{x}_i$, where $\mathbf{x}_i \in \mathcal{X}$. We then predict the sample mean iteratively, $\bar{\boldsymbol{\mu}}^{(n+1)} = m(\mathcal{X}, \bar{\boldsymbol{\mu}}^{(n)})$, where n is the iteration step index; one step can be seen in Figure 4.4. Unlike classical mean shift, we do not determine convergence by defining a distance tolerance between the centres found in consecutive iterations. Instead, we use the prediction of inliers given

Algorithm 5 The DMS Algorithm

function DMS($\mathcal{X}$)
  $\mathcal{X}_{init} \in \mathcal{X}$, select a set of initialisation points
  for each point $\mathbf{x}_i \in \mathcal{X}_{init}$ **do**
    $\bar{\boldsymbol{\mu}}_i$ = findCentre($\mathbf{x}_i$, $\mathcal{X}$)
  end for
  $\mathcal{M} = \{\mu_0, \mu_1, ..., \mu_i, ...\}$, the set of all estimate centres
  $\mathcal{M}$ = combineCentres($\mathcal{M}$, $\mathcal{X}$, τ), remove duplicate centres
  for each point $\mathbf{x}_i \in \mathcal{X}$ **do**
    $\hat{S}(i) = \underset{j}{\text{argmax}}(K_D(\mathbf{x}_i, \hat{\boldsymbol{\mu}}_j))$, find the most confident cluster
    if $K_D(\mathbf{x}_i, \hat{\boldsymbol{\mu}}_j) < (1 - \tau)$ **then**
      $\hat{S}(i)$ = outlier, if no clusters are confident the point is an 'outlier'
    end if
  end for
end function

function FINDCENTRE($\mathbf{x}_i$, $\mathcal{X}$)
  $\bar{\boldsymbol{\mu}}_i^{(n)} = \mathbf{x}_i$
  $\bar{\boldsymbol{\mu}}_i^{(n+1)} = m(\mathcal{X}, \bar{\boldsymbol{\mu}}_i^{(n)}) = \frac{\sum_{\mathbf{x}\in\mathcal{X}} \mathbf{x} K(\mathbf{x}, \bar{\boldsymbol{\mu}}_i^{(n)})}{\sum_{\mathbf{x}\in\mathcal{X}} K(\mathbf{x}, \bar{\boldsymbol{\mu}}_i^{(n)})}$
  if $\frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}\in\mathcal{X}} ||K_D(\mathbf{x}, \bar{\boldsymbol{\mu}}_i^{(n)}) - K_D(\mathbf{x}, \bar{\boldsymbol{\mu}}_i^{(n-1)})|| < 10^{-6}$ **then**
    $\bar{\boldsymbol{\mu}}_i = \bar{\boldsymbol{\mu}}_i^{(n)}$
  else
    $\bar{\boldsymbol{\mu}}_i$ = findCentre($\bar{\boldsymbol{\mu}}_i^{(n+1)}$, $\mathcal{X}$, τ)
  end if
end function

function COMBINECENTRES($\mathcal{X}$, $\mathcal{M}$)
  $\mathbf{H} = [|\mathcal{X}| \times |\mathcal{M}|]$
  $\mathbf{S} = [|\mathcal{M}| \times |\mathcal{M}|]$
  for each point $\mathbf{x}_i \in \mathcal{X}$ **do**
    for each centre $\boldsymbol{\mu}_j \in \mathcal{M}$ **do**
      $\mathbf{H}_{i,j} = K_D(\mathbf{x}_i, \bar{\boldsymbol{\mu}}_j)$
    end for
  end for
  $\tilde{\mathbf{H}}$ = biStochasticNormalisation($\mathbf{H}$) [133]
  $\mathbf{S} = \tilde{\mathbf{H}}\tilde{\mathbf{H}}^\top > 0.5$
  Comp = $\{\{i, j, k\}, \{p, q, r\}, ...\{x, y, z\}\}$ = findConnectedComponents($\mathbf{S}$) [134]
  $\mathcal{M} = \{\boldsymbol{\mu}_i, \boldsymbol{\mu}_p, ..., \boldsymbol{\mu}_x\}$ = removeDuplicateCentres($\mathcal{M}$, Comp)
end function

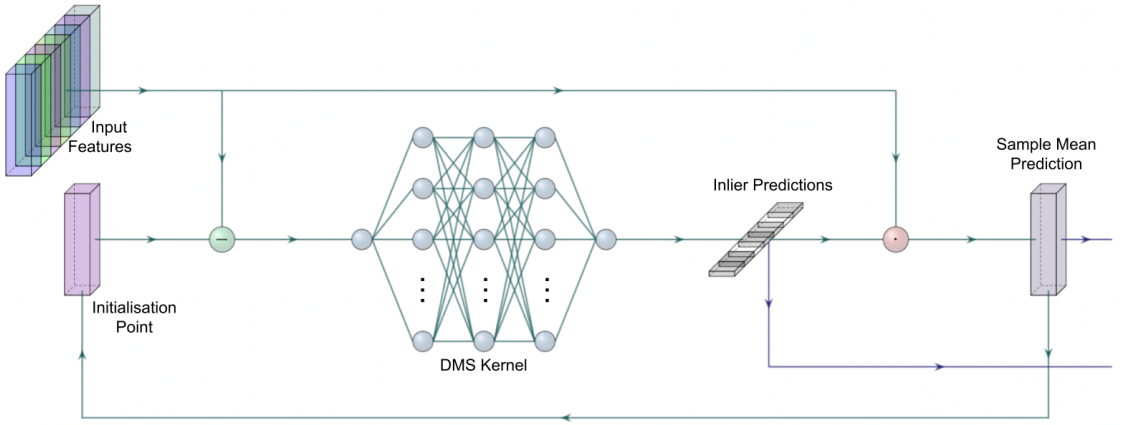


Figure 4.4: The DMS framework for iterative cluster centre finding and inlier prediction.

the current and previous sample mean. If two consecutive sample means, $\bar{\boldsymbol{\mu}}^{(n-1)}$ and $\bar{\boldsymbol{\mu}}^{(n)}$, predict the same points to be inliers and outliers, the algorithm has converged, and the cluster centre has been found. Formally, the algorithm has converged if

$$\frac{1}{|\mathcal{X}|}\sum_{\mathbf{x}\in\mathcal{X}}||K_D(\mathbf{x},\bar{\boldsymbol{\mu}}_i^{(n)}) - K_D(\mathbf{x},\bar{\boldsymbol{\mu}}_i^{(n-1)})|| < \tau \tag{4.7}$$

where τ is the convergence threshold. In this way, we can calculate M sample means, $\bar{\boldsymbol{\mu}}_i$, initialised from $\mathbf{x}_i$, where $i = 1,..,M$.

We evaluate sample mean similarity in the same way as iterative convergence. This is necessary because while in most cases similar samples converge to an identical cluster centre, *i.e.* $\|\bar{\boldsymbol{\mu}}_i - \bar{\boldsymbol{\mu}}_j\|_2 < 10^{-6}$, we observe that samples that belong to the same ground-truth cluster frequently converged to different but similar sample means. To manage this discrepancy, we define two centres to be the same if both clusters predict the same points to be inliers and outliers with similar confidences.

Formally, we calculate a confidence matrix $\mathbf{H} \in [0,1]^{|\mathcal{X}|\times M}$ where the element $\mathbf{H}_{i,j}$ represents the confidence that the data point $\mathbf{x}_i$ belongs to the candidate cluster j.

$$\mathbf{H}_{i,j} = K_D(\mathbf{x}_i, \bar{\boldsymbol{\mu}}_j) \tag{4.8}$$

A prediction similarity matrix over the candidate sample means can be calculated $\mathbf{S} = \tilde{\mathbf{H}}\tilde{\mathbf{H}}^\top$, where $\tilde{\mathbf{H}}$ is the bi-stochastic normalisation of $\mathbf{H}$ [133]. Each element $\mathbf{S}_{i,j}$ measures the similarity between the sample means $\bar{\boldsymbol{\mu}}_i$ and $\bar{\boldsymbol{\mu}}_j$. We find the final unique cluster centres by binarising $\mathbf{S}$ and extracting the connected components [134]. The same binarising tolerance $\tau = 0.5$ is used for all datasets. However, we found that any value between 0.1 and 0.95 will provide similar results as the values are driven to either 1 or 0 by the training loss; see Equation 4.10. Each data point is then assigned to the cluster with the highest inlier confidence for that point.

$$\hat{S}(i) = \begin{cases} \underset{j}{\text{argmax}}(K_D(\mathbf{x}_i, \hat{\boldsymbol{\mu}}_j)) & \text{if} \quad K_D(\mathbf{x}_i, \hat{\boldsymbol{\mu}}_j) > (1-\tau) \\ \text{outlier} & \text{else} \end{cases} \tag{4.9}$$

In datasets with high variance, it can be the case that a point which originally converged to a particular centre is not assigned to the associated cluster during the final stage. This is either due to a different centre being more confident of its inliership in the final stage or because no cluster is confident of its inliership. Generally, this is desirable, as points not associated with any cluster or on the border of two clusters may converge to an incorrect centre during our first stage due to the lack of a hard neighbourhood constraint in K_D. The most obvious way to understand this is that a trail of data points could gradually lead the predicted mean a greater distance from the initialisation point than another, closer, possible cluster mean that has fewer of these 'stepping stone' points.

Following this stage, the clusters could be further refined in various ways. For example, trivial single-point solutions could be removed if it is known that this case does not occur in the data. During our testing, we do not remove these trivial solutions as this implies knowledge of the dataset.

It is worth noting that while classical mean shift needs to be initialised from

every point in the dataset, our algorithm only needs to be initialised from at least one point per cluster. Classical mean shift classifies a point as belonging to a cluster if it converges to the associated centre; our algorithm only needs to find each cluster centre as our second stage finds all inliers of all of the clusters.

4.2.5 Training with Side-Information

During training, we generate data as follows. A given dataset, $\mathcal{X}$, and task configuration contains a set of categories, $\mathcal{C}$, and each category $c \in \mathcal{C}$ contains a number of samples, $|c|$. This cardinality may vary between categories. These category boundaries are task-specific. For example, the categories for dividing by colour will be different from those for dividing by shape. We divide the dataset into a set of training points, $\mathcal{X}_{tr}$, and evaluation points, $\mathcal{X}_{ev}$, where $\mathcal{X} = \mathcal{X}_{tr} \cup \mathcal{X}_{ev}$. We assume that we have side-information in the form of relational 'similar/dissimilar' labels for the points in $\mathcal{X}_{tr}$ but not for those in $\mathcal{X}_{ev}$.

One training instance is composed of a random data point, $\mathbf{x}$, called the initialisation point, a set of positive samples, $\mathcal{P}$, a set of negative samples, $\mathcal{N}$, and a set of unlabelled samples, $\mathcal{U}$. As the datasets we use include ground-truth labels instead of pairwise side-information, we generate the side-information from the ground-truth labels, then discard the ground-truth labels. $\mathcal{P}$ is composed of points drawn from the same class as the initialisation point, $\mathcal{N}$ is drawn from the points of different classes to the initialisation point, and $\mathcal{U}$ is drawn from all points in $\mathcal{X}$.

The specific class labels of all of the points are discarded once these categories, which define the side-information, are created. The use of pairwise relational labels as side-information instead of using the specific class label during training allows the clustering network to learn a representation of similarity independent of the specific class.

Only points in $\mathcal{X}_{ev}$ are used during testing. While the unlabelled samples, $\mathcal{U}$,

are taken from the entire dataset, $\mathcal{X}$, they are excluded from the training loss. The unlabelled data points serve only to increase the cardinality and variance of the clusters during training, facilitating more consistent cluster convergence. All classes and data points in $\mathcal{P}$, $\mathcal{N}$, and $\mathcal{U}$ are randomly and independently selected. Points can be repeated in a training instance, facilitating training in situations where there are not enough available data points to create sufficiently large positive class sizes.

During training, we use four mean shift iterations; the impact of this is further discussed in Section 4.3.6. We used the Binary Cross Entropy [135] of the inlier predictions and the side-information of the labelled points as the training loss, see Equation 4.10. Predictions of the unlabelled points are not used in the loss.

$$\text{Loss} = -\sum_{i \in \mathcal{P}, \mathcal{N}} (y_i \log(p_i) + (1 - y_i) \log(1 - p_i)) \tag{4.10}$$

where $y_i = 1$ if $x_i \in \mathcal{P}$ and $y_i = 0$ if $x_i \in \mathcal{N}$, and p_i is the prediction confidence that x_i is in $\mathcal{P}$. This loss consistently pushes predictions to the extremes of the unit interval. It is rare to have values between 0.05 and 0.95 predicted after training.

4.3 Experiments

In this section, we evaluate DMS using standard datasets and metrics for both intrinsic and non-intrinsic tasks before discussing our architectural and training choices.

4.3.1 Datasets

We draw datasets from various domains, including object detection, facial recognition, handwriting comprehension, and text classification based on word frequency. To ensure a fair comparison, we used the standard benchmark descriptors and reduced datasets where appropriate, as in the experimental section of Shah and Koltun

[73]. We compare our method to other methods using MNIST [136], COIL100 [137], YouTube Faces (YTF) [138], and RCV1 [139].

MNIST is a popular dataset of handwritten digits which are normally categorised into 10 classes. It contains 70,000 images. Each image is 28×28 pixels (784 dimensions). COIL100 is a dataset of 100 objects each in one of 72 orientations. It contains a total of 7,200 RGB images. Each image is 3×128×128 (49,152 dimensions). YTF is a dataset comprised of videos of faces. In clustering settings [72, 73], the first 40 subjects were sorted in chronological order as used, for a total of 10,056 frames. Each image is an RGB image that has 3×55×55 pixels (9,075 dimensions). RCV1 is a dataset of Reuters newswire articles. It contains 800,000 articles. While it contains a complex hierarchical labelling system, it is standard for only the four root categories to be used for clustering. Articles labelled with more than one root category are removed. We use the same randomly sampled subset of 10,000 articles as Shah and Koltun [73]. We also use the same normalized 'Term Frequency - Inverse Document Frequency' features [140]. As these TF-IDF features are taken from the 2,000 most frequently occurring word stems, there is a dimensionality of 2,000.

4.3.2 Intrinsic vs Non-intrinsic Tasks

We recognise that in many cases, there are numerous useful ways to divide a dataset. While there is not always a clear boundary between these, in the contexts of the datasets we use, we divide these into intrinsic tasks and non-intrinsic tasks.

Intrinsic tasks are those which are most obvious given all facets of the data. This generally aligns with clustering based on object class, reflecting the way a person would likely group the data points first; see the top row of Figure 4.5. In all of our datasets, we believe the provided labels are aligned with the intrinsic task.

Non-intrinsic tasks are the less obvious groupings. Generally, they require some alterations to the default importance of some facets of a dataset. These tasks can

be hierarchical to the intrinsic task, orthogonal to it, or somewhere in between. A hierarchical task is one where multiple complete classes from the intrinsic task are grouped to form 'parent classes' or classes are split further to create 'child classes'. For example, we define a set of parent classes based on the broader root classes of the objects in COIL100 (toy cars, drinking vessels, boxes, etc.). We also define a set of parent classes based on colour. Visual examples are shown in the second and third rows of Figure 4.5.

Orthogonal tasks, on the other hand, are tasks in which every intrinsic cluster contains one instance from the new task. For example, the images of cars from COIL100 could be separated based on orientation (agnostic to the specific object). This task is orthogonal to separating the specific objects (agnostic to the orientation). We do not test our method on orthogonal tasks in this work as there was no obvious orthogonal task that had clear utility in these datasets. Other methods, which have no side-information input, perform very poorly on these tasks producing unfair comparisons.

4.3.3 Metrics

We report results using the standard metrics: clustering accuracy (ACC) [141] and Normalised Mutual Information (NMI) [142]. As NMI has been shown to be biased towards the generation of finer-grained clusters and ACC has been shown to be unreliable in datasets where there is significant variation in cluster cardinality [143], we also include Adjusted Mutual Information (AMI). ACC, NMI, and AMI are all in the range $[0, 1]$, where 1 indicates ideal performance, and are defined by Equations 4.11, 4.12 and 4.13 respectively. ACC is defined as:

$$\mathrm{ACC}(\mathbf{L}, \hat{\mathbf{L}}) = \frac{1}{|M|} \sum_{l \in \mathbf{L}, \hat{l} \in \hat{\mathbf{L}}} \delta(\hat{l}, l) \tag{4.11}$$

where $\mathbf{L}$ and $\hat{\mathbf{L}}$ are the matched predictions and ground-truth labels, respectively. $\delta(x, y) = 1$ if $x = y$ and $\delta(x, y) = 0$, otherwise. The matching is achieved by maximising the ACC over all possible one-to-one assignments. In practicality, the optimal mapping is found efficiently using the Jonker-Volgenant algorithm [144] to robustly solve the Hungarian matching algorithm [145]. For NMI and AMI we have

$$\mathrm{NMI}(\mathbf{L}, \hat{\mathbf{L}}) = \frac{\mathrm{I}(\mathbf{L}, \hat{\mathbf{L}})}{\sqrt{\mathrm{H}(\mathbf{L})\mathrm{H}(\hat{\mathbf{L}})}} \tag{4.12}$$

$$\mathrm{AMI}(\mathbf{L}, \hat{\mathbf{L}}) = \frac{\mathrm{I}(\mathbf{L}, \hat{\mathbf{L}}) - \mathbb{E}[\mathrm{I}(\mathbf{L}, \hat{\mathbf{L}})]}{\sqrt{\mathrm{H}(\mathbf{L})\mathrm{H}(\hat{\mathbf{L}})} - \mathbb{E}[\mathrm{I}(\mathbf{L}, \hat{\mathbf{L}})]} \tag{4.13}$$

where $\mathrm{I}(\cdot, \cdot)$ is the mutual information between its inputs, $\mathrm{H}(\cdot)$ is the entropy, and $\mathbb{E}[\cdot]$ is the expectation function.

$$\mathrm{I}(X; Y) = \mathrm{H}(X) - \mathrm{H}(X|Y) \tag{4.14}$$

$$\mathrm{H}(X) := -\sum_{x \in \mathcal{X}} p(x) \log p(x) = \mathbb{E}[-\log p(X)] \tag{4.15}$$

$$\mathrm{H}(X|Y) = -\sum_{x,y \in \mathcal{X} \times \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(y)} \tag{4.16}$$

where $p(x) = P[X = x]$, $p(y) = P[Y = y]$, and $p(x, y) = \mathbb{P}[X = x, Y = y]$

4.3.4 Experiment Training

During training, 5% of the total dataset had pairwise side-information, $|\mathcal{X}_{tr}| = \frac{|\mathcal{X}|}{20}$. The total number of data points used in a training instance, $|\mathcal{P}| + |\mathcal{N}| + |\mathcal{U}|$, is a random integer between 200 and 300 and varies between different batches. The ratio of the number of points with side-information to those without in a training instance, $|\mathcal{P} \cup \mathcal{N}| : |\mathcal{U}|$, varies uniformly between $[\frac{1}{3}, 1]$; the ratio of positive pairs to points with side-information, $|\mathcal{P}| : |\mathcal{P} \cup \mathcal{N}|$, varies uniformly between $[\frac{1}{20}, \frac{1}{10}]$. For datasets with 10 or fewer classes with similar cardinalities, this means the positive set is always under-represented during training, and for cases with more than 20 classes, the positive set is always over-represented. This did not appear to have any significant effect on performance. Each batch is comprised of 96 training instances.

We found that for datasets where clusters were similar, *i.e.* comparable cardinality, inter-cluster variance, and intra-cluster variance, a significantly smaller amount of training data could be used with no detriment to performance. COIL100, for instance, could be trained with side-information present for only 0.7% of the data, approximately one point per class from each of 25 of the total 100 clusters, without significant performance differences to using the entire training set. As it is unlikely that the points were drawn at one per class, this shows the network's ability to generalise and identify unseen classes; this is further shown in Section 4.3.6, where we evaluate the performance of the method with limited class exposure. Conversely, when there are few and unique clusters with regard to cardinality and variance, it is important that each cluster is well represented in the training data.

4.3.5 Results

In the ideal case, we would initialise our method from all data points in the test set at inference time. However, we found that as long it was likely a point was initialised in each cluster, there was no detriment to performance from reducing

the number of initialisation points. As we did not want to assume knowledge of the number of clusters in the dataset, we initialised the method on all datasets from 500 data points randomly selected from the test set. We did not ensure that each ground-truth cluster was represented by the initialisation points. We conducted mean shift iterations which included all points in the test dataset from these initialisation points until the centres converged.

We significantly outperform all contemporary methods on all four datasets on their intrinsic tasks as shown in Table 4.1. Either other methods only find the most obvious clusters (the 'intrinsic' task) or in cases where other tasks can be suggested, they require abstract parameter tuning in terms of the number or variance of clusters. Our method, in contrast, is versatile and unbiased towards any task. We compare our method to other contemporary methods on two hierarchical non-intrinsic tasks. To ensure fairness, we performed multi-variate hyper-parameter tuning of the benchmark methods to maximise their chance of success on each of the non-intrinsic tasks. As other approaches generally find the divisions associated with the intrinsic task of a dataset, they perform worse than chance on orthogonal tasks, and it would be unfair to compare these results. DMS significantly outperforms other methods on the less obvious, 'non-intrinsic', tasks; see Table 4.2.

4.3.6 Ablation

In this section, we evaluate components of our training approach and network architecture. We vary the number of mean shift iterations used during training, the number of layers used in the kernel, and the robustness of subtraction and concatenation-based kernels to limited training data.

Figure 4.5: Different clustering tasks using the same data points. The intrinsic task is the most 'obvious' division of a dataset this is the task other approaches generally solve. As our method is example-led, it has no bias towards the intrinsic task over other problems.

	MNIST			COIL100			YTF			RCV1		
Algorithms	**ACC**	**NMI**	**AMI**	**ACC**	**NMI**	**AMI**	**ACC**	**NMI**	**AMI**	**ACC**	**NMI**	**AMI**
Classical												
K-means++	0.532	0.5	0.5	0.621	0.835	0.803	0.624	0.788	0.783	0.529	0.355	0.355
AC-W	0.571	0.679	0.679	0.697	0.876	0.853	0.647	0.806	0.801	0.554	0.364	0.364
DBSCAN	0.000	0.000	0.000	0.921	0.458	0.399	0.675	0.756	0.739	0.571	0.017	0.014
SEC	0.545	0.469	0.469	0.648	0.872	0.849	0.562	0.760	0.745	0.425	0.069	0.069
LDMGI	0.723	0.761	0.761	0.763	0.906	0.888	0.332	0.532	0.518	0.667	0.382	0.382
Deep Learning												
GDL	n/a	n/a	n/a	0.825	0.965	0.958	0.497	0.664	0.655	0.444	0.020	0.020
RCC	0.876	0.893	0.893	0.831	0.963	0.957	0.484	0.850	0.836	0.356	0.138	0.138
RCC-DR	0.698	0.827	0.828	0.825	0.963	0.957	0.579	0.882	0.874	0.676	0.442	0.442
RCC-DR(SGD)	0.696	0.827	0.827	0.855	0.967	0.961	0.473	0.845	0.830	0.354	0.106	0.106
DCN	0.560	0.570	0.570	0.600	0.830	0.810	0.620	0.810	0.790	0.730	0.470	0.470
DEC	0.867	0.853	0.840	0.815	0.645	0.611	0.643	0.811	0.807	0.683	0.504	0.500
JULE	0.800	0.900	0.900	0.911	0.983	0.979	0.342	0.587	0.574	-	-	-
DEPICT	0.968	0.919	0.919	0.402	0.678	0.667	0.586	0.790	0.785	-	-	-
DCC	0.963	0.915	0.913	0.858	0.967	0.962	0.699	0.908	0.903	0.563	0.498	0.495
DMS (*subtract*)	**0.976**	**0.939**	**0.939**	**0.990**	0.996	**0.994**	**0.923**	**0.949**	**0.912**	**0.760**	0.587	**0.532**
DMS (*concat*)	0.958	0.914	0.896	**0.990**	**0.997**	**0.994**	0.832	0.896	0.820	0.582	**0.603**	0.453

Table 4.1: The clustering accuracy of our approach compared against 14 baselines on the intrinsic task of each dataset. Accuracy is measured using the standard accuracy metric (ACC), Normalised Mutual Information (NMI), and Adjusted Mutual Information (AMI). Results other than our own are as found in Shah and Koltun [73].

	Intrinsic Task			Root Class			Dominant Colour		
Algorithms	**ACC**	**NMI**	**AMI**	**ACC**	**NMI**	**AMI**	**ACC**	**NMI**	**AMI**
RCC-DR	0.471	0.544	0.461	0.825	0.963	0.957	0.415	0.476	0.359
DMS (sub)	**0.990**	**0.996**	**0.994**	**0.990**	**0.997**	**0.996**	**0.817**	**0.790**	**0.751**

Table 4.2: The ability of our method and other contemporary methods to learn various clustering tasks from the same COIL100 features. All tasks use the same features, but the clustering focuses on a different aspect of the semantic information.

Training Iterations

The optimal performance of DMS is when 4 iterations are used during the training of our kernel; the comparison to other numbers of iterations are shown in Table 4.3. We hypothesise that when a small number of iterations are used, the network does not have sufficient flexibility to learn the nuance of the given task. Conversely, with a large number of iterations, the network is not incentivised to effectively define the clusters as it has many opportunities to improve its prediction. We found that the number of iterations required for convergence at inference time increased proportionally with both the number of training iterations and the complexity of the dataset. This verifies the idea that a single pass through the kernel is less able to distinguish inliers from outliers given more training iterations. Given four training iterations, the time taken to converge at inference time varied between one and five iterations depending on the complexity of the dataset.

Kernel Depth

Here, we evaluate the effect of varying the number of fully connected layers used in the differentiable kernel. All networks are trained for the same number of epochs. We use a sufficient number of epochs to ensure the training of all networks had converged. As shown in Table 4.4, increasing the depth of the kernel improves the performance of the method. After three layers, the performance improvements are negligible while training time and likelihood of overfitting increase. For this

Layers	ACC	NMI	AMI
2	0.818	0.920	0.847
4	**0.923**	**0.949**	**0.912**
6	0.890	0.915	0.883
8	0.780	0.837	0.813

Table 4.3: The performance of our differentiable kernel with varied training iterations on the YTF dataset.

Layers	ACC	NMI	AMI
1	0.069	0.000	0.000
2	0.920	0.934	0.895
3	**0.923**	0.939	**0.912**
4	0.905	**0.942**	0.898

Table 4.4: The performance of our differentiable kernel with a varied number of fully connected layers on the YTF dataset.

reason, we use three fully connected layers for all other tests.

Concatenation and Subtraction Kernels

Here, we test two differentiable kernels, the subtraction-based kernel, which takes the difference of the data point and sample mean, $(\mathbf{x} - \bar{\boldsymbol{\mu}}_i)$, and an alternative concatenation-based kernel, which takes a concatenation of the data point and sample mean, $concat(\mathbf{x}, \bar{\boldsymbol{\mu}}_i)$. The concatenation kernel requires the addition of an extra fully-connected layer compared to the subtraction-based kernel. We use the same architecture as the original differential kernel after this.

While we hypothesised the use of a concatenation and a fully-connected layer would be more effective than the simpler subtraction as a greater amount of information could be used, this turned out to not be the case. As shown in Table 4.1, the results of both kernels were comparable on datasets where it was likely all clusters would be represented. However, the results varied significantly when this was not the case. It should be noted that the concatenation and the additional first layer could simply learn the subtraction function.

As it is clear that the concatenation-based kernel performs worse than the

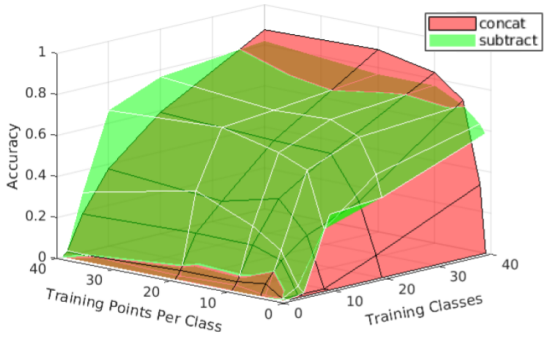


Figure 4.6: The effect of varying the distribution of training data on the performance of our subtraction and convolution-based kernels. For this test, we explicitly limit the number of classes and the maximum number of points per class with side-information during training. This shows the method's ability to discover novel classes as there is very little decrease in performance of the subtraction-based method when half of the classes are absent during training.

subtraction-based kernel on more sparse datasets, we further test their performance when less side-information is available. Here, unlike our other experiments where the points with side-information are randomly selected, we explicitly limit the number of classes and points per class with side-information. The results of this experiment appear in Figure 4.6.

The performance of both kernels degrades similarly when fewer data points per class are seen during training. There is little effect on the performance unless the number of data points is significantly restricted. However, while the concatenation-based kernel performs better in cases where all of the ground-truth classes are present in the training data, its performance is significantly reduced when this is not the case.

The subtraction-based kernel, on the other hand, has little degradation in performance when decreasing the classes present during training, which demonstrates that it is better able to generalise to unseen data. We believe this is because the subtraction-based kernel, unlike the concatenation kernel, is only presented with the relative position of points to the current sample mean estimate. During training, the subtraction-based kernel is unable to use the global position of the training clusters as a crutch and must learn to find a better description of similarity. This

experiment clearly shows the method's ability to correctly identify and cluster novel classes, as both kernels maintain high accuracy when $\sim 75\%$ of the classes are not present during training.

4.4 Conclusion

In this chapter, we present a novel approach to clustering which learns to cluster points directly from side-information in the form of pairwise similarity. Unlike previous work, this approach does not require any other knowledge of the dataset as a whole; it does not require knowledge of the number of clusters, their centres, or any kind of distance metric as a prior. We demonstrate this approach using DMS, a differential, iterative clustering method inspired by the mean shift algorithm.

DMS learns the requirements of a task and uses this to outperform state-of-the-art deep-learning-based clustering approaches on both intrinsic and non-intrinsic tasks without the need to specify any task or dataset-specific hyper-parameters. This flexibility, along with DMS's performance in high-dimensional feature spaces and its lightweight nature, enables it to be seamlessly incorporated into other deep learning applications for end-to-end training or as a head on top of an already trained backbone.

It should be noted that the work presented in this chapter was completed soon after the introduction of the attention mechanism and before the introduction of the vision transformer. While using attention mechanisms for clustering is difficult due to their $O(n)$ complexity, we believe that utilising an attention mechanism in the kernel on a subset of inlier points to further refine inlier predictions would likely improve our method's accuracy.

While the identification of similarity that DMS provides is crucial for counting, it is limited to counting separately provided instances. Most of the current computer vision literature defines counting as the identification *and* enumeration of similar

instances within an image rather than the enumeration of already separated instances in distinct images. In order to use DMS in these settings, an additional detection or segmentation stage would be required. Instead of implementing a two-stage detect-then-count framework, we go on to focus on addressing both stages simultaneously to develop a single-stage exemplar free class-agnostic counter.

5

Exemplar-Free Weakly-Supervised Single-Class Class-Agnostic Counting

5.1 Introduction

Counting is an easy task to understand but is time-consuming to perform manually at scale and difficult to automate. Simple though it is, instance counting has diverse applications including crowd-counting, traffic monitoring, conservation, microscopy, and inventory management. Significantly, whereas people can generally count objects without a prior understanding of the type of object to be counted, most current automated methods cannot.

The work in this chapter is based on the intuition that the ability to count is, at its core, comprised of two components:

- Object Detection: Where are object instances we may want to count?
- Repetition Recognition: Is the class of this given instance repeated elsewhere?

This intuition is gained from the fact that when presented with a set of novel objects and asked to 'count', a person knows what is to be counted. They do not require an exemplar or prior understanding of object type to clarify that we do not want to find the repetitions of, say, a self-similar background.

While the work in Chapter 4 addressed the problem of repetition recognition with minimal labels, it would need to be paired with a detection stage to complete the full counting task. The work presented in this and the subsequent chapter addresses both stages simultaneously.

Until the publication of 'Class-Agnostic Counting' [4] in 2018, methods focused on specific domains and developed feature spaces useful for counting a single or small set of classes of objects. Class-agnostic counting methods aim to learn general features and matching approaches that can then be applied to previously unseen object types. However, Lu et al. [4] and many following works [23, 104, 110] still need human intervention at inference time, in the form of a set of exemplar images. These exemplar images are sometimes called 'reference images' [125].

Our work, presented here, as well as a few concurrent or follow-up works [25, 116, 146] aim to minimise the amount of human intervention needed during deployment. Specifically, we all try to demonstrate that class-agnostic counting can be achieved without the need for exemplar images. In this chapter, we focus on performing this in settings with only a single class present per image. We acknowledge that FSC-147 and FSC-133 guided our, and many others, view of class-agnostic counting significantly. The single-class nature of these datasets allows exemplar-free methods to perform well on it.

We demonstrate that self-supervised vision transformer features, specifically their use of self-attention with a global receptive field, are a useful tool for meeting both

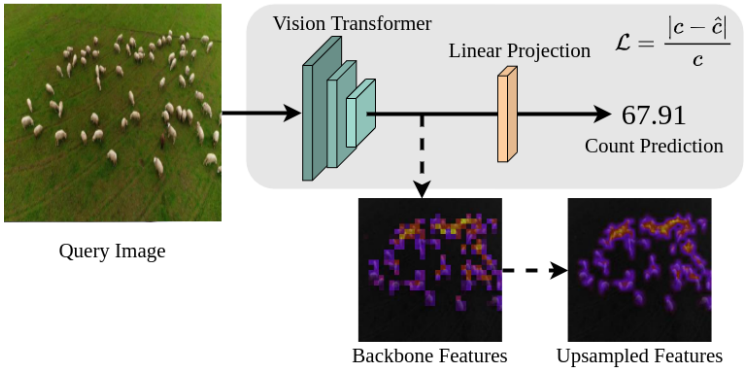


Figure 5.1: The RCC pipeline. Our method learns to count objects of novel classes without exemplar images, using only the ground truth count, c, as supervision. We also visualise these features to show they suffice to localise instances of the counted object.

of our conditions for counting and can replicate the behaviour observed in human counting. We prove experimentally that, given general and globally contextual features, enumerating instances is simple, requires only minimal training, and can be achieved using only image-wise scalar supervision rather than instance-wise locational labels. We also demonstrate that while using only scalar supervision, the learnt latent features are sufficient to localise the instances of the relevant counted objects.

The key contributions of this chapter are as follows:

1. We propose RCC, the first exemplar-free class-agnostic counter.

2. We show that in single-class settings, using image-wise count supervision rather than instance-wise supervision is beneficial and allows the network to learn its own idea of positional salience.

3. We demonstrate that RCC outperforms the exemplar-free class-agnostic counting methods that followed it and that it is competitive with current methods that use exemplar images as a prior and train with full point-level supervision.

5.2 Method

In this chapter, we approach the challenging task of counting instances of unseen classes without exemplar images or point-level supervision. Unlike other contemporary methods [116, 146] which have separate object discovery and matching stages of counting, we solve these tasks simultaneously. Object detection, in a class-agnostic context, requires a general, informative feature space. We identify that self-supervised knowledge distillation is well-suited to learning such a feature space. Repetition recognition requires an understanding of the global context of an image. Vision transformers, in particular their use of attention, are perfectly suited to this task.

5.2.1 Self-supervised knowledge distillation.

As vision transformers generally require large amounts of data to learn a general, informative feature space and the standard counting dataset is small, we wish to train the vision transformer on other datasets first. This pretraining means that the backbone will already have a meaningful understanding of general image features and the only learning that is required is the counting-specific information, which is relatively simple. For our evaluation on previously unseen classes to be fair, this pretraining needs to take place either on only the training classes from the final dataset or on a wider range of classes but without the use of labels. We chose to take the latter approach as it increases the flexibility of our pretraining data requirements which amounts to utilising more data during pretraining.

We use a training methodology of self-supervised knowledge distillation based on Caron et al. [38] for our pretraining stage. We learn informative representations of images by encouraging consensus between a 'teacher' network, g_t parameterised by θ_t, and a 'student' network, g_s parameterised by θ_s. The teacher network operates on large 'global' crops, V_G, of an image, while the student network operates on

a set, V_L, of smaller, 'local' crops of the same image. Both networks generate probability distributions P_t and P_s respectively, defined in Equation 5.1. These probability distributions function similarly to classification predictions in supervised classification tasks. The goal is to have the student, which can only see portions of the image, make the same 'prediction' about what is in the image as the teacher, which can see a larger portion of the image. Similar to classification tasks, this is achieved by minimising the cross-entropy between the probability distributions of the student network and the 'oracle' labels of the teacher network, as in Equation 5.2.

$$P_s(x)^{(i)} = \frac{e^{g_s(x)^{(i)}}/\tau_s}{\sum_{d=0}^{d_p} e^{g_s(x)^{(d)}}/\tau_s} \quad , \quad P_t(x)^{(i)} = \frac{(e^{g_t(x)^{(i)}} - C)/\tau_t}{\sum_{d=0}^{d_p} e^{(g_t(x)^{(d)}} - C)/\tau_t} \tag{5.1}$$

$$\min_{\theta_s} \sum_{x_g \in V_G} \sum_{x_l \in V_L} -P_t(x_g) \log P_s(x_l) \tag{5.2}$$

Here, d_p is the dimensionality of the probability distribution, $\tau_s > 0$ and $\tau_t > 0$ are the temperature parameters of the student and teacher, and C is a centring parameter. The temperature and centring parameters are used for sharpening and centring the predicted distribution. Sharpening and centring both act to decrease the chance of modal collapse without the need for contrastive losses or batch normalisation, both of which require batch-wise assumptions about the data. Sharpening discourages one form of modal collapse; it keeps the network from generating a uniform prediction distribution for all images. However, it encourages the network to generate a single value with a high probability for all images. Centring operates in the other direction, minimising the chance the network will generate an identical prediction for all images. This is achieved by centring the teacher's predicted probabilities around an exponential moving average of all its predictions in a batch.

$$C \leftarrow mC + (1-m)\frac{1}{B}\sum_{i=0}^{B} P_t(x_i) \tag{5.3}$$

where $m > 0$ is a rate parameter controlling the momentum of the average, and B is the batch size.

Different to standard distillation, the teacher network is constructed from the student model. It is a momentum teacher [147] whose weights are iteratively updated with an exponentially moving average of the student's weights, which are scheduled to increase throughout training.

5.2.2 Count Regression

In the context of counting, whether two objects are the same should be independent of their location within an image. A related observation is that if two features are similar and they are far apart, then it is more likely that they are repetitions of a class of objects rather than parts of the same instance, meaning they are more likely to be the desired countable. While CNNs have been shown to work well at object detection, we believe their local-first formulation is not well suited to repetition recognition. Instead, we believe transformer architectures with their global attention are perfectly suited to counting. The crucial global context afforded by transformers stems from their use of an attention mechanism. As discussed in Section 2.1.2, attention generates a new feature from linear projections of all other features based on their similarity. This means all features are affected by all other features, rather than prioritising spatially local features as in a convolutional structure. Inspired by Vaswani et al. [35], our vision transformers, g_t and g_s, use multiple attention heads to generate informative, diverse features.

In contrast to the established literature and the methods that followed ours, we use this behaviour alone to conduct the counting. We directly regress a single,

scalar count prediction, $\hat{c}$, for the whole input image, x, from the latent features of $g_s(x)$ using a linear projection, $\mathbf{F}$.

$$\hat{c} = \mathbf{F} \cdot g_s(x) \tag{5.4}$$

We train this using scalar count supervision alone, without using point-level annotations. Simply put, we ask the method to generate an answer to the question 'How many things are there in an image?' and motivate it only by telling it how far off it was from the correct number. Although it may seem that a location-based loss should aid in training, we found that point-level annotations and Gaussian density maps were unnecessary and often detrimental in a single-class class-agnostic setting such as that presented in FSC-133/147; see Section 6.5.1 for further details. The positional annotations in FSC-133/147 are often arbitrarily placed and contain at most limited information about the size and shape of an object. Most methods generate a pseudo-density map from these points by placing Gaussian kernels of a standard size at each one of the annotated points. As the point annotations are not necessarily centred on the object and as it is not possible for the Gaussian kernel to be the correct size and shape to be representative of all the instances in the dataset, the generated density maps are flawed approximations. Due to this, teaching this information as 'ground-truth' supervision for a regression stage is not helpful. Consider, say, a method which identified a part of each of the correct objects that is different to the part the annotations label such as counting the erasers on pencils rather than their centres or tips. This would be severely punished by a positional loss function while being a correct evaluation of the problem. This arbitrary punishment hinders the network's ability to gain an understanding of the counting task, especially in a class-agnostic setting where the labels are more likely to be arbitrarily placed.

Instead, we allow the network to develop its own conceptual representation of the

task by regressing an estimate for the count directly. To this end, we use one of the simplest possible loss functions, the Absolute Percentage Error (APE), defined as:

$$\mathcal{L}_{APE} = \frac{|c - \hat{c}|}{c} \tag{5.5}$$

where c is the ground truth count and $\hat{c}$ is the predicted count. This was a slight improvement over Absolute Error when applied to FSC-133/147 as it limits the disproportionate effect of very high-density images on the gradients; see the ablation tests in Section 5.4.5. We found that the network, without the imposition of human ideas of positional salience, still learnt to implicitly localise instances of the counted class in a meaningful way in its latent layers. We further discuss this, and its usefulness, in Section 5.4.4.

5.2.3 Tiling Augmentation

We found that while our method had reasonable performance on most images, it struggled in cases where the objects were small and densely packed. This issue has been noted in many other methods [23, 25, 116]. Various approaches have been taken to creating multi-scale systems which can improve the detection of objects of varying sizes. These approaches generally require a complex spatial loss [148] or non-maximal suppression [83], which have significant computational costs. As our method regresses a single count rather than a set of object locations and is trained using only scalar supervision, these approaches are not feasible.

To allow the network to better understand multiple scales and densities, we instead increase the diversity of image densities at training time by tiling resized versions of the input image into a (2×2) grid. We apply this augmentation randomly to 50% of the training instances. This increases the representation of high-density images and improves our method's results; the MAE and RMSE were decreased by

~10% and ~20% respectively. The effect of this augmentation is further discussed in the ablation section 5.4.5.

A similar adaptation was adopted in a later work by Liu et al. [25], who used it to both increase the number of high-density images but also to create a rudimentary multi-class dataset to increase the discrimination ability of their method.

We also tested cropping areas of images out to increase the number of images with low counts. We found that this had no positive effect on our results and also that it cannot be applied to datasets where the locations of objects are not known.

5.3 Experiments

In this section, we discuss the details of our implementation, training, and the metrics we use for evaluation.

5.3.1 Architecture

We use a ViT-small backbone inspired by DeiT-S [149]. This was chosen due to its efficiency and to provide a good comparison to other counting methods which use a ResNet-50 backbone. ViT-S has a similar number of parameters (21M vs 23M), throughput (1237im/sec vs 1007im/sec), and supervised ImageNet performance (79.3% vs 79.8%) as ResNet-50 [149], which is used by contemporary counting methods.

As even data-efficient transformers require relatively large amounts of data to train and the datasets on which we evaluate are small, we initialised our transformer backbone, g_s, using weights from Caron et al. [38]. As discussed in Section 5.2.1, this self-supervised pre-training gives the network an understanding of meaningful image features without supervision and before exposure to our limited datasets, minimising the chance of overfitting.

Our linear count projection, F, projects from a $p \times d_m$ feature space to a scalar count prediction, where d_m is the dimensionality of the transformer features and p is the number of patches of the vision transformer. We found $d_m = 384$ and $p = 28^2$ are sufficient to achieve competitive results while also being lightweight enough to be trainable on a single 1080Ti Nvidia GPU. It should be noted that this vision transformer configuration limits the resolution of our input image to (224×224) as opposed to the ∼(384×384) used by contemporary methods with ResNet-50 or larger ViT backbones.

5.3.2 Training

We used a batch size of 2 distributed over 2 GPUs (Nvidia 1080Ti). We trained for 80 epochs with a learning rate of $3e-5$, which takes 2.5 hours. To increase the diversity of object densities at training time, we applied our 2×2 tiling augmentation to 50% of the iterations. We applied random reflections and rotations to the images and, when tiled, to each tile independently. Colour-based augmentations (colour-jitter, Gaussian blur, and solarisation) had a negligible effect on our results. We did not apply random crops as this would require point-level annotations to adjust the count.

5.3.3 Evaluation Metrics

In accordance with previous works on class-agnostic counting [24, 116], we use Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to evaluate our performance,

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|c_i - \hat{c}_i| \tag{5.6}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(c_i - \hat{c}_i)^2} \tag{5.7}$$

where c_i and $\hat{c}_i$ are the ground truth and predicted counts for image x_i, and n is the number of images in the validation or test set.

5.4 Results

In this section, we show that RCC dramatically outperforms other exemplar-free class-agnostic counting methods and is competitive with exemplar-based methods. Since our method does not need exemplar images or point-level annotations, it can be applied to broader real-world applications where the objects to count are ever-changing. We validate this by showing our method's ability to generalise to a novel domain. We also use our learnt latent features to localise instances in an image. Not only may this have real-world utility, it also helps substantiate that RCC uses meaningful information to count. We then discuss the failure cases and limitations of our method to inform and motivate possible future research. We finally validate specific components of our method.

5.4.1 Benchmark Methods

We evaluate our method against two trivial baseline methods which predict the same value, $\hat{c}$, for all test images, as follows:

$$\hat{c}_{\text{mean}} = \frac{\sum C_{\text{train}}}{n_{\text{train}}} \qquad \hat{c}_{\text{median}} = C_{\text{train}}\left[\frac{n_{\text{train}}}{2}\right] \tag{5.8}$$

where C_{train} is an ordered list of all ground truth counts for the training set

and n_{train} is the number of images in the training set.

We also compare our method to two few-shot detection methods (FR [150] and FSOD [151]), seven exemplar-based class-agnostic counting methods (GMN [4], MAML [152], FamNet [23], CFOCNet [110], BMNet [24], LaoNet [117] and CounTR [25]), and RepRPN-Counter [116], a concurrent work which is the only other class-agnostic counting method that does not require exemplar images during training. CounTR was released after our method. It trains using a mix of exemplar images and exemplar-free examples applying the learning from the exemplars to the exemplar-free cases.

We also include comparisons to exemplar-free modifications of exemplar-based methods implemented by Ranjan and Hoai [116], denoted by a * in the results tables 5.1 and 5.2. Since Ranjan and Hoai [116] have not released the implementations of either their method or the modified exemplar-based methods they evaluate, the experiments we could run comparing to them were limited. To ensure a fair comparison, we also train recent, high-performing methods with released implementations using the same vision transformer backbone, ViT-S, as our method, denoted by † in the results tables 5.1 and 5.2.

5.4.2 FSC-147 and FSC-133.

As seen in Table 5.1, we achieve significantly better results than the concurrent work RepRPN and all exemplar-free versions of exemplar-based methods on FSC-147 without the need for point-level annotations. We are also competitive with previous few-shot or exemplar-based methods on FSC-147 and FSC-133 without the need for exemplar images, point-level annotations, or test-time adaptation; see Tables 5.1 and 5.2.

These results demonstrate that the combination of an architecture well-suited for counting and a simple, count-centred objective function can learn a meaningful

conceptual representation of counting without any location-based information and so does not require exemplar images. As will be discussed in Section 5.4.6, we believe that the discrepancy in FSC-147 test-set RMSE is due to poor performance on the high-density images in the test set, as these outliers have a greater effect on RMSE than MAE. This is likely caused by our network operating at a lower resolution than the other methods as well as the fact that we cannot use the test-time adaptation tricks they utilise due to a lack of exemplar data.

We found that methods that were modified to use a ViT-S backbone in place of their ResNet-50 backbones performed worse; this is likely due to the significant decrease in the resolution of both the input image and the latent features. Since FamNet previously used features from two sequential layers of a ResNet backbone, this modification also halved the dimensionality of the features used to regress the count, decreasing their scale-invariant performance.

In general, the methods perform better on FSC-133 than on FSC-147 with the exception of the validation MAE. The greater validation MAE is likely due to the removal of duplicate images and similar classes that allowed the network to 'cheat', having seen these images during training. The other metric improvements are likely due to a few high-density images being moved to the training set in the dataset reformulation.

5.4.3 Cross-Dataset Generalisability

To confirm our method was domain invariant and had strong generalisability, we evaluate it on CARPK [85]. CARPK is a car counting dataset comprised of birds-eye views of car parks. These aerial views significantly differ in appearance from any objects found in FSC-147 or FSC-133. To ensure we are fairly testing the generalisability of the methods, we exclude the 'car' class from our FSC-133 pretaining.

Method	Shots	Val Set		Test Set	
		MAE	RMSE	MAE	RMSE
Mean	N/A	53.38	124.53	47.55	147.67
Median	N/A	48.68	129.7	47.73	152.46
Exemplar-based Training					
MAML [152]	3	25.54	79.44	24.90	112.68
FR [150]	3	45.45	112.53	41.64	141.04
FSOD [151]	3	36.36	115.00	32.53	140.65
GMN [4]	1	29.66	89.81	26.52	124.57
FamNet [23]	1	26.55	77.01	26.76	110.95
FamNet [23]	3	24.32	70.94	22.56	101.54
FamNet+ [23]	3	23.75	69.07	22.08	99.54
FamNet† [23]	3	37.90	109.76	33.51	137.79
FamNet+† [23]	3	37.77	109.04	33.18	137.20
CFOCNet [110]	3	21.19	61.41	22.10	112.71
LaoNet [117]	1	17.11	56.81	15.78	97.15
BMNet [24]	3	19.06	67.95	16.71	103.31
BMNet+ [24]	3	15.74	58.53	14.62	91.83
BMNet† [24]	3	19.29	68.58	18.34	115.31
BMNet+† [24]	3	17.21	60.18	16.90	107.66
CounTR [25]	3	**13.13**	**49.83**	**11.95**	**91.23**
CounTR† [25]	3	22.10	73.08	19.52	111.50
CounTR [25]	0	**17.40**	**70.33**	**14.12**	**108.01**
CounTR† [25]	0	28.96	90.66	25.58	110.39
Exemplar-free Training					
MAML* [152]	0	32.44	101.08	31.47	129.31
GMN* [4]	0	39.02	106.06	37.86	141.39
FamNet*(pretrained) [23]	0	39.52	116.08	39.38	143.51
FamNet* [23]	0	32.15	98.7 5	32.27	131.46
RepRPN-Counter [116]	0	29.24	98.11	26.66	129.11
CounTR†* [25]	0	33.15	100.80	28.95	125.31
RCC (ours)	0	**17.49**	**58.81**	**17.12**	**104.53**

Table 5.1: Comparison to state-of-the-art methods on FSC-147. We outperform other exemplar-free methods and achieve competitive results with methods which use exemplar images and test-time adaptation. We highlight the best results for few-shot and zero-shot exemplar-based-training and exemplar-free-training methods. † denotes methods trained using the same backbone as ours. * indicates an exemplar-free modification of an exemplar-based method. Methods in grey were made public after this work.

Method	Shots	Val Set		Test Set	
		MAE	RMSE	MAE	RMSE
Mean	N/A	54.39	112.68	44.76	104.28
Median	N/A	51.29	119.29	43.15	109.83
Exemplar-based Training					
FamNet [23]	3	25.49	68.21	21.05	43.48
FamNet+ [23]	3	24.75	67.04	20.20	41.76
FamNet† [23]	3	36.89	92.76	30.09	90.95
FamNet+† [23]	3	36.36	89.22	30.79	90.71
BMNet [24]	3	20.16	54.83	14.61	41.11
BMNet+ [24]	3	**16.54**	**50.65**	**13.85**	**40.61**
BMNet† [24]	3	22.06	61.01	16.38	54.13
BMNet+† [24]	3	18.78	59.76	13.87	47.27
CounTR [153]	3	**15.34**	52.38	**9.78**	**35.46**
CounTR† [25]	3	22.19	62.54	16.80	55.03
CounTR [25]	0	20.32	69.61	11.97	38.68
CounTR† [25]	0	32.69	74.10	25.62	71.86
Exemplar-less training					
CounTR†* [25]	0	29.80	81.06	22.07	66.47
RCC (ours)	0	**17.87**	**41.69**	**14.79**	**34.48**

Table 5.2: Comparison to available state-of-the-art methods on FSC-133. We are competitive with exemplar-based methods without exemplar images or test-time adaptation. † denotes methods trained using the same backbone as ours. Methods in grey were made public after this work.

We significantly outperform FamNet with and without fine-tuning, and we are competitive with BMNet and CounTR without fine-tuning. We believe the fine-tuning result discrepancy with BMNet and CounTR results from out-of-distribution high-density images. This is because 70% of the test images in CARPK have more instances than the maximum count seen during training.

The other methods use exemplar images which aid in the cross-dataset generalisation and with issues of out-of-distribution high-density test images. This ability to deal with high-density images is significantly improved by the test-time adaptation schemes utilised in FamNet+ and BMNet+. Both methods make adjustments to their count based on knowledge about the exemplar patch, namely that the exemplar patch must contain at minimum one instance; if less than one instance

Method	Shots	Fine-Tuned	MAE	RMSE
Mean	N/A	N/A	65.63	72.26
Median	N/A	N/A	67.88	74.58
Exemplar-based				
FamNet [23]	3	×	28.84	44.47
BMNet [24]	3	×	14.61	24.60
BMNet+ [24]	3	×	10.44	13.77
CounTR [25]	3	×	**10.20**	**12.67**
CounTR [25]	0	×	11.33	14.60
FamNet [23]	3	✓	18.19	33.66
BMNet [24]	3	✓	8.05	9.70
BMNet+ [24]	3	✓	5.76	7.83
CounTR [25]	3	✓	**5.75**	7.45
CounTR [25]	0	✓	5.84	**7.43**
Exemplar-less				
RCC (ours)	0	×	**12.31**	**15.40**
RCC (ours)	0	✓	**9.21**	**11.33**

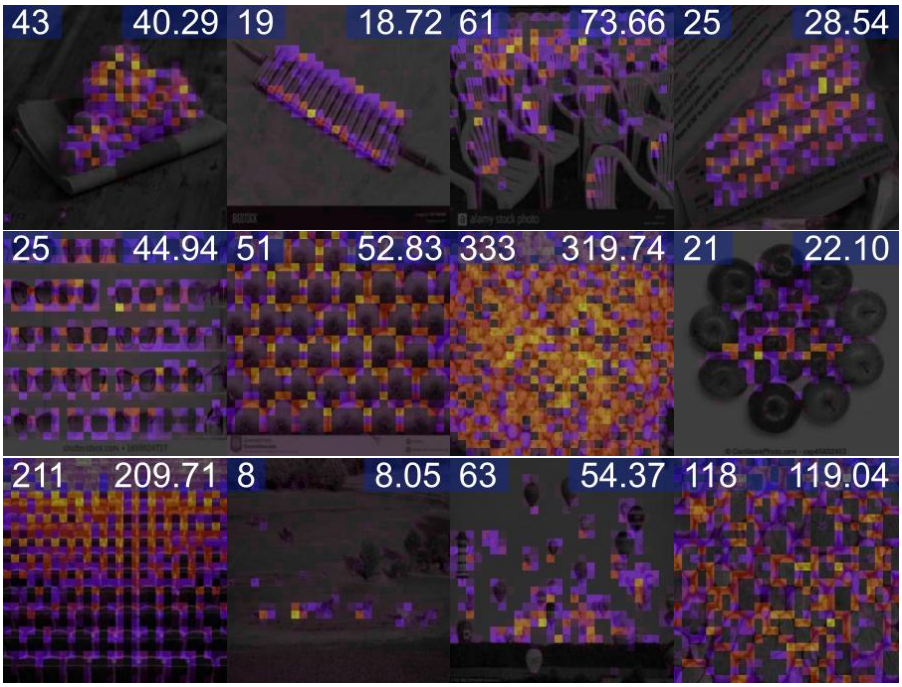
Table 5.3: Generalisation performance on CARPK. "Pretrained" models were trained on FSC-147 or, for our method, FSC-133. "Fine-tuned" models were further trained on the CARPK dataset. We highlight the top results with and without exemplar images and with and without fine-tuning. Methods in grey were made public after this work.

is predicted the whole density map is scaled until this is the case, increasing the overall count in high-density cases.

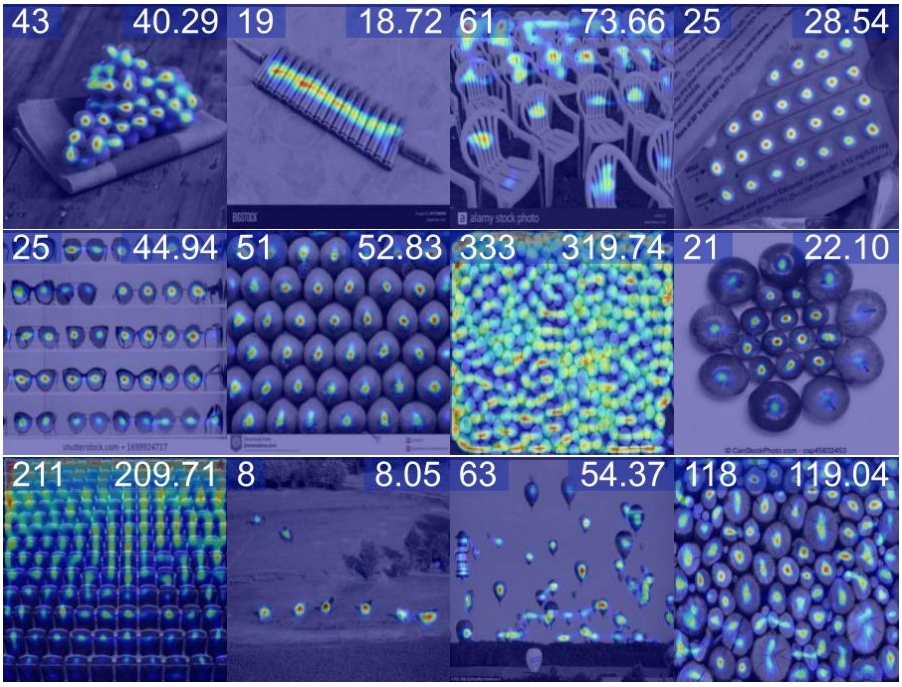
Further, while our method improves after fine-tuning, showing the benefit of task-specific information, the improvements are smaller than with the other methods. This indicates our method's generalisability is relatively optimised for other tasks before fine-tuning.

5.4.4 Feature Visualisation

To validate that our network is learning to recognise meaningful information, we visualise the most significant value from a singular value decomposition of the latent features from the final transformer layer weighted by the linear projection weights; see Figure 5.2a. It is clear from these visualisations that the features are localised

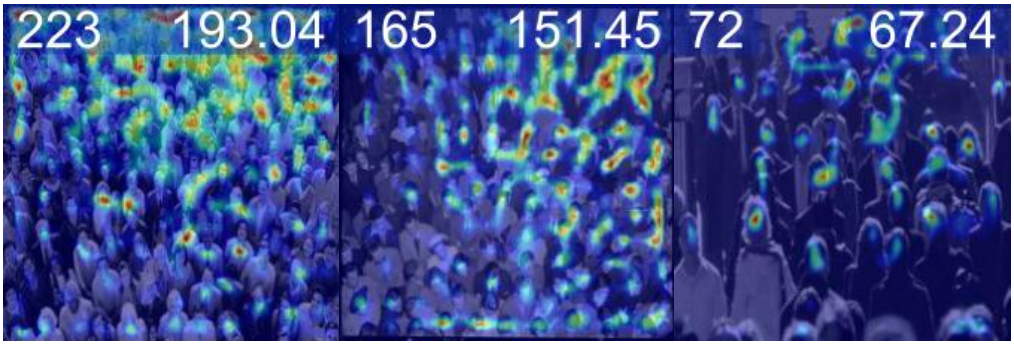


(a) PCA Feature Localisation

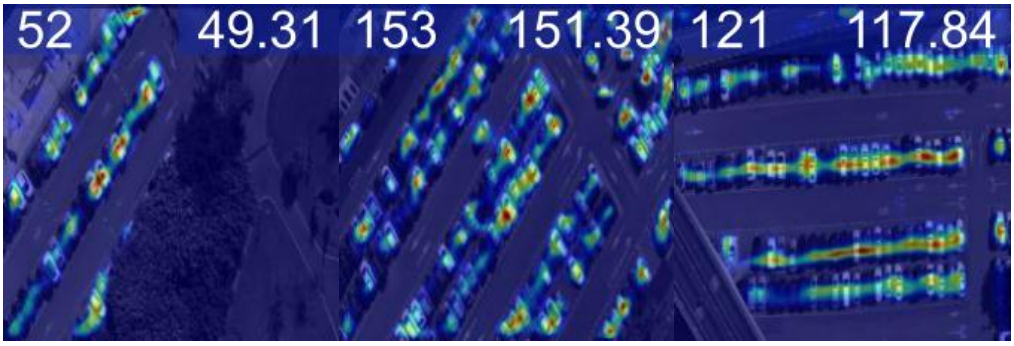


(b) Localisation Head Predictions

Figure 5.2: Localisation of latent counting features on unseen dataset classes from FSC-133. (a) Patch-wise principal components of the counting backbone features. (b) Pixel-wise localisation predictions of unseen dataset classes from FSC-133. The latent counting feature density maps are generated by a localisation head trained on FSC-133. The count and prediction of each image are in the top left and the top right, respectively.



(a) ShanghaiTech



(b) CARPK

Figure 5.3: Localisation of latent counting features on unseen datasets. Pixel-wise localisation predictions of unseen classes from unseen datasets CARPK and ShanghaiTech respectively. The latent counting feature density maps are generated by a localisation head trained on FSC-133. The count and prediction of each image are in the top left and the top right, respectively.

around the objects which have been counted. The network is attending to a single type of object rather than just all objects present. This behaviour makes sense as given the limited ground truth counts from the training dataset, the network must find 'the most obvious' thing to count, ignoring other, unlabelled, classes. As our patch-wise counting features are relatively low resolution and there may be utility in more accurately localising instances in an image, we trained a lightweight localisation head. This head, comprised of 3 Conv-ReLU-Upsample blocks, increases the patch-wise resolution of the trained counting features (28×28) to a pixel-wise density map prediction (224×224); see Figure 5.2b for examples. This head is trained using the pixel-wise mean squared error of the predicted density map and

a ground truth Gaussian density map as is standard in the field [23, 24].

Since this training is not weakly-supervised, we freeze the feature backbone to ensure we are testing if the weakly-supervised features are sufficient for high-resolution localisation. This training takes 10 epochs and has significantly better results than the same process conducted on a backbone frozen after the self-supervised pretraining. This further validates that our network is learning the ability to count rather than just detect objects as happens after only the self-supervised pretraining. We also show that the backbone and localisation head trained on FSC-133 are generalisable to other domains by testing it on the CARPK and ShanghaiTech [96] datasets; see Figure 5.3.

5.4.5 Ablation Studies

In this section, we validate the various components of our method, namely that the global context of a vision transformer backbone is crucial, that the counting head is effective without the need for a complex architecture, that our loss function is more effective than the commonly used loss functions, and that our tiling augmentation has a positive effect on our system's performance.

Attention-less Features

To demonstrate that the global context provided by the attention mechanism present in vision transformers is a critical component to exemplar-free class-agnostic counting, we evaluate RCC with ResNet-50 [20] and ConvNeXt [154] backbones in place of our transformer. We initialised both architectures with standard weights learnt from classification pre-training on ImageNet [155]. While using weights generated by training on a supervised task that overlaps classes with our test and validation sets means that the results are not directly comparable to our self-supervised method, this comparison serves as a best-case for these architectures.

When trained comparably, Resnet-50 has a comparable accuracy to our vision transformer backbone and ConvNeXt has a higher classification accuracy than our architecture on image classification tasks [154]. To enable comparison across backbones, we use the same input feature size (224×224) and take latent features at the same (28×28) resolution as the patched features used in our backbone. As seen in Table 5.4, even with the advantageous supervised pretraining, the attention-less features perform significantly worse than the self-supervised vision transformer, validating the idea that the attention mechanism is more significant in counting than in classification tasks.

Count Regression Complexity

We assert that given sufficiently general and globally aware features, regressing an accurate count should be straightforward. To validate this, we compare our linear projection against two count regression heads. These count regression heads are a simple architecture, Conv(3×3)-ReLU-Linear-Relu-Linear, and a more complex architecture comprised of four (3×3) convolutional layers followed by three linear layers with ReLU activations.

We found that more complex counting heads achieved worse results than our linear projection on all three backbones during inference, as shown in Table 5.4. It appears that the extra computational capacity of the complex architecture is not only unnecessary for the task of counting but is in fact detrimental as it overfits to the training classes. This is validated by both alternatives having a lower counting error than our linear projection during training.

Loss Function

We tested the two most common loss functions, Absolute Error (AE) and Squared Error (SE), against our loss function, Absolute Percentage Error. APE has the best performance on FSC-133, as shown in Table 5.5. We believe the main contributing

		Val Set		Test Set	
Backbone	Head	MAE	RMSE	MAE	RMSE
ResNet-50	Projection	31.80	78.87	24.99	75.87
	Simple	36.21	98.74	26.51	110.36
	Complex	33.63	88.50	24.02	89.31
ConvNeXt	Projection	30.30	82.16	21.58	87.21
	Simple	24.41	67.26	23.07	111.70
	Complex	25.94	89.48	24.60	134.66
ViT-Small	Projection	**17.87**	**41.69**	**14.79**	**34.48**
	Simple	23.08	66.19	17.46	78.60
	Complex	20.73	57.67	15.22	52.24

Table 5.4: Performance of different feature backbones, and two more complex count regression head architectures on FSC-133.

	Val Set		Test Set	
Loss	MAE	RMSE	MAE	RMSE
Squared Error	37.76	78.74	29.92	71.56
Absolute Error	21.93	64.85	14.90	**32.86**
Absolute Percentage Error	**17.87**	**41.69**	**14.79**	34.48

Table 5.5: Performance of RCC on FSC-133 with different loss functions.

factor to this is that using the APE decreases the significance of the very high-density images to the training. During training, these images create high gradients during the backpropagation, often an order of magnitude or more above the average. We believe that this gradient disturbance compromises the training significantly. The performance increase discrepancy is slightly (∼5%) more significant on FSC-133 than on FSC-147, which contains more high-density images in its test and validation sets but fewer during training than FSC-133.

Tiling Augmentation

We validate the use of our image tiling augmentation by testing various frequencies and tiling configurations, as shown in Table 5.6. We found that introducing the denser (4×4) tiling had detrimental effects. At this density of tiling, each object

	Val Set		Test Set	
Frequency	MAE	RMSE	MAE	RMSE
(2×2)				
75%	20.10	60.23	13.72	**31.26**
50%	**19.84**	**55.81**	14.23	43.83
25%	20.95	64.44	14.98	47.08
0%	21.76	68.68	15.72	77.07
(4×4) or (2×2)				
75%	20.73	63.21	**13.38**	38.61
50%	20.00	65.08	15.44	37.47
25%	20.51	63.73	15.71	42.77
0%	21.76	68.68	15.72	77.07

Table 5.6: Performance results for varying sizes and frequencies of our tiling augmentation on FSC-147. Images are tiled in either a (2×2) grid or in one of a (4×4) or (2×2) grid with equal probability.

instance is very small within the total image. These very small instances, in the context of our patched features, are no longer interpretable, providing little meaningful training. We also found that when a majority of training instances were tiled, the network's accuracy on lower-count images decreased due to these cases now being under-represented during training. This decrease in performance on the low-density images eventually outweighs the positive gains on the higher-density images as while they generally contribute less to the overall metric per image, they are much more numerous.

5.4.6 Failure Cases and Limitations

The most significant limitations of this work are its poor performance in high-density situations and that it can only generate a single count for a given image. The latter of which is addressed by our proposed method in Chapter 6.

High-Density

High-density situations are a common failure mode for counting methods, and while this method could be improved by, for example, (a) running upsampled sections of images through the counter, (b) increasing the input resolution of the transformer backbone, or (c) increasing the density of images in the training stage, we have left this as possible future work.

A clear failure case of RCC is its difficulty with high-density images. As shown in Table 5.7, the metrics on FSC-133 improve dramatically when the few images with over 1000 objects are removed from the evaluation. These images, presented in Figure 5.4, constitute 0.31% of the validation set and 0.09% of the test set. This is further verified by the results shown in Figure 5.5, where the effect of excluding higher values is shown more continuously. (Figure 5.5 also shows that the vast majority of images contain less than 200 images.) This failure case is likely because our patch-based approach limits the functional resolution to a (28×28) feature map.

With many of these high-density images, the same object is found in all patches and across all patch boundaries equally, making the features associated with individual instances indistinguishable. Ideally, we would use a smaller patch size, but due to computational constraints, we could not test this. As a proxy for this test, we split the high-density images into multiple sub-images that were processed independently and combined their counts. This improved the MAE and RMSE by 45.6% and 64.9% respectively. However, this does not generalise in a principled manner to all cases, especially very low-density cases

This failure case could also be attributed to the underrepresentation of very high-density images in the training data; only 9 (0.23%) of the 3877 training counts are over 1000. This is supported by the fact that these effects appear more dramatically in FSC-147 where high-density images have even lower representation. This is also supported by the improvements found from applying the tiled image augmentation,

	Val Set				Test Set			
Limit	#	%	MAE	RMSE	#	%	MAE	RMSE
None	0	0.00	19.84	55.81	0	0.00	14.23	43.83
1000	3	0.31	17.94	43.99	1	0.09	12.96	26.69
500	12	1.26	16.11	34.80	7	0.66	12.12	23.02

Table 5.7: The effect of removing high-density images from FSC-133. Excluding the small number of very dense images significantly improves our results, showing that these are a weakness of our method. # and % denote the total number and percentage of images excluded respectively.

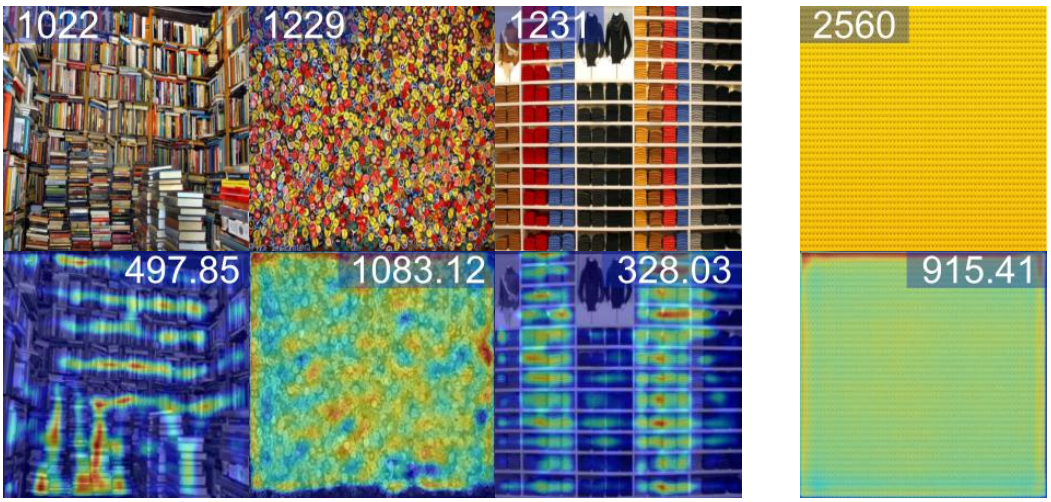


Figure 5.4: High-density images from FSC-133. The three images from the validation set (left) and the single test image (right) with associated counts of over 1000.

which artificially inflates the count of some training iterations.

Single-Class

While our method performs competitively on FSC-133/147 with exemplar-based methods, this is not reflective of its performance in many deployment situations. As discussed in Chapter 3, one of the significant flaws of FSC-133/147 is that the images are single-class. This enables a method such as our own to assume there will be only objects of a single type present and generate a single count. In reality, many situations will have objects of multiple types present so the assumption, and the performance of methods that utilise it, breaks down. As it stands, if multiple

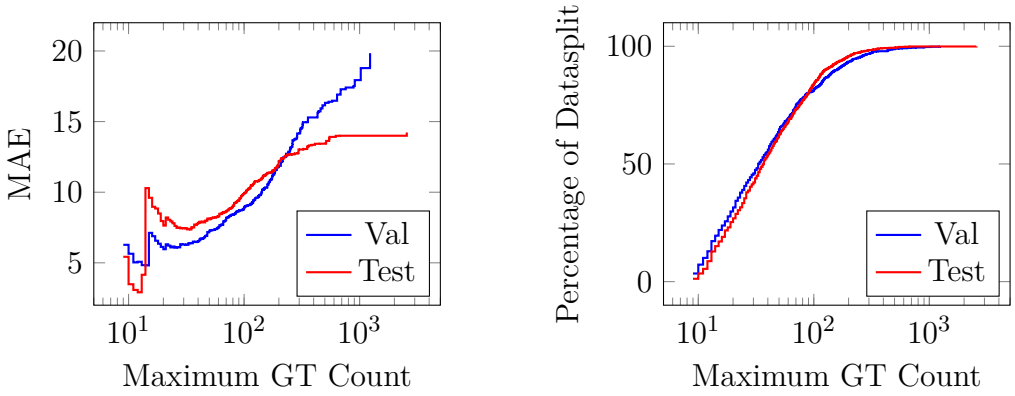


Figure 5.5: The MAE of RCC when operating on FSC-133, excluding counts above a given value (left), and the percentage of the data split that test includes (right).

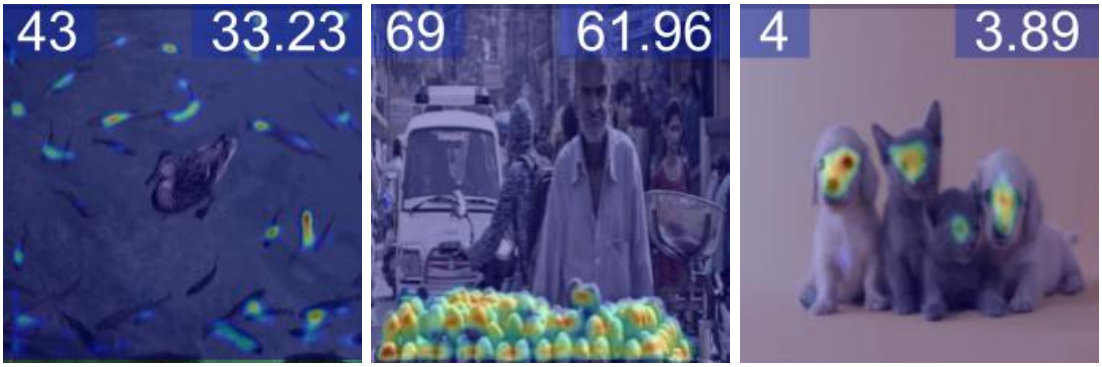


Figure 5.6: Localisation of counting features with multiple classes present. The network either ignores all but one class (left and centre) or it groups very similar classes and counts the superset (right). The left and centre images have high levels of occlusion, which leads to low predictions.

types of objects are present in an image, RCC will either count the super-set of all objects, or it will 'choose' one class to count; see Figure 5.6. This significantly limits its real-world deployability as it is only suitable for applications where it is known only one class will be present. It is unlikely this would be known while also not knowing the visual appearance of the class, and so a class-agnostic method would likely not be required.

While this is the case, the core achievement of this work is negating the requirement for exemplar images and point-level annotations. There are also a wide range of applications to singular classes of novel objects, e.g. medical imaging and microscopy [11].

5.5 Conclusion

In this work, we present RCC, the first exemplar-free class-agnostic counting method, and show that it can be trained without point-level annotations. This is based on the confirmed intuition that well-trained vision transformer features are both general enough and contextually aware enough to implicitly understand the underlying basis of counting, namely object detection and repetition identification.

We demonstrate on FSC-133/147 that RCC is superior to the exemplar-free methods that followed it and is competitive with current exemplar-based class-agnostic counting approaches that use full point-level supervision. We also show its cross-domain generalisability on CARPK and that it can localise appropriate object instances without any location-based intervention during training.

There is clear utility in a method which does not have a reliance on object class priors, exemplar images, and positional annotations. RCC proves such a method is possible. However, this proof is constrained by there being only a single class present per image during deployment. While there are deployment situations where it is known only one class of object will be present while also not knowing the visual appearance of this class, we recognise that they are much less common than tasks where the number of classes is not guaranteed. In Chapter 6, we build on this method to remove the single-class constraint, creating the first exemplar-free multi-class class-agnostic counter.

6

Exemplar-Free Multi-Class Class-Agnostic Counting

6.1 Introduction

Class-agnostic counting methods enumerate objects of an arbitrary class, providing tremendous utility in many fields. Prior works have limited usefulness as they require a set of examples of the type to be counted.

In Chapter 5, we proposed a solution to exemplar-free counting in a single-class class-agnostic setting. Concurrent works [25, 116] also addressed this problem, though they used an exemplar-based method as their backbone and either aimed to generate reasonable exemplars at inference time or trained with exemplars and applied this learning to exemplar-free settings. While our work and these

methods provide reasonable results on FSC-133/147, they do not scale to images with multiple classes. This single-class limitation severely restricts the real-world deployment of these methods.

We believe that similar to our assumption in Chapter 5, humans possess the ability to count multiple objects of multiple types in an arbitrary image with limited to no direction. We show in this work that this behaviour is automatable. An automated counting method provides significant utility as it is faster and less labour-intensive than manually counting or using exemplar-based methods.

To address the task of multi-class counting in a class-agnostic setting, we propose A Blind Counter (ABC123), a method that can count multiple types of objects simultaneously without using examples of type during training or inference. To help users to understand the generated outputs, ABC123 introduces a new paradigm where instead of requiring exemplars to guide the enumeration, examples are found after the counting stage. We show that ABC123 outperforms contemporary methods on MCAC, our novel multi-class dataset proposed in Section 3.3, without the requirement of human-in-the-loop annotations. We also show that this performance transfers to FSC-133/147, the standard class-agnostic counting dataset.
Our contributions are:

- We propose ABC123, the first exemplar-free multi-class class-agnostic counter.
- We introduce the idea of *example finding* to exemplar-free counting and demonstrate its utility in aiding a user in understanding what has been counted.
- We show that prior methods do not perform as expected in multi-class settings and that ABC123 tackles multi-class counting effectively.

6.2 Related Work

The work in Chapter 5, RCC, addressed exemplar-free counting in single-class settings, removing the need for human intervention during deployment. In this section, we will outline the developments in exemplar-free counting which happened parallel to our work and helped to motivate the method presented in this chapter. Specifically, we will discuss the concurrent works proposed by Ranjan and Hoai [116], Liu et al. [25], Xu et al. [146], and Paiss et al. [156].

RepRPN [116] is a two-step exemplar-free method. It proposes regions likely to contain an object of interest and then uses them for an exemplar-based density map regression method. It proposes more than one bounding box and enumerates them separately.

ZSC [25] uses a multi-stage process to achieve counting from a text input instead of using exemplar images. First, the text input is used to generate a generic image of the type to be counted. This generic image is then used to find exemplar patches. Finally, the found exemplar patches are used as the input to an exemplar-based method, specifically BMNet [24].

CounTR [146] uses a large vision-transformer encoder-decoder to regress a density map of instance locations. It is trained using both few-shot and zero-shot cases, applying understanding gained from exemplar-based problems to exemplar-free cases. While it was not originally intended to be trained on only zero-shot problems, we verify that this is possible on single-class data and achieves reasonable but not state-of-the-art results.

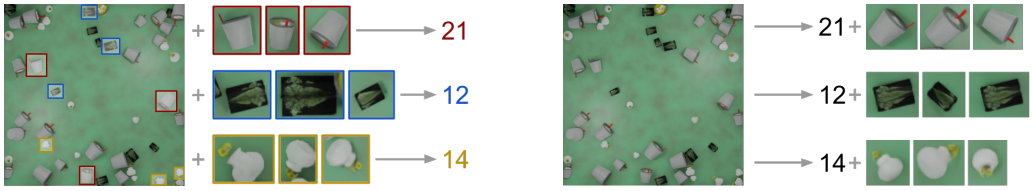
It has been assumed that exemplar-based methods can function in multi-class settings. However, this has not been proven rigorously because, as discussed in Chapter 3, the main dataset for class-agnostic counting (FSC-133/147 [23, 125]) contains only one labelled class per image. In fact, we show in Section 6.4 that these methods perform poorly in contexts with multiple types present.

While large models with image inputs like ChatGPT [157] and SAM [158] would seem to be able to effectively count objects of arbitrary types, in fact these methods have poor numerical understanding [159]. SAM, for example, performs unsatisfactorily on all counting tasks but is especially inaccurate on images with small objects or a high density of objects [159]. Paiss et al. [156] introduce a method that teaches a CLIP [160] model to count. They achieve this by creating a new dataset with labels of the form "a photo of {number} {objects}". They then use this dataset, along with known counterfactuals, generated by changing the {number}, to finetune a pre-trained visual language model (VLM) using a discriminative task as a proxy for the counting. In this way, they are able to encourage the VLM to learn a latent representation that has the count as a significant contribution to the features. They demonstrate impressive results, but they are only able to achieve this understanding for numbers up to ten and with very large computational costs.

6.3 Method

Our method, ABC123, takes an image with multiple instances of objects of multiple types and regresses the count of each type. This is achieved *blind*, *i.e.* on objects of arbitrary classes with no requirement to have seen the object classes during training or to have an exemplar image to define the type during training or inference. We achieve this by first regressing density maps for each type and then enumerating the instances using integration.

To facilitate training and evaluating ABC123 in an exemplar-free way, we propose a matching stage which matches the unguided counts to the known ground truth labels. To further increase the interpretability and utility of the outputs of ABC123, we design an example discovery stage which finds specific instances of the counted object.



(a) Exemplar-Based Counting **(b)** Our Exemplar-Free Counting

Figure 6.1: ABC123 counts objects of multiple unseen types. Our method not only does not need exemplars to define the type to count but also finds examples of each type it has counted.

6.3.1 Density Map Regression

Our method, like many before [23–25, 116], is based on density map regression. For each image, there are m classes present, each with an associated ground truth count y and density map d. We regress $\hat{m}$ counts and density map predictions, $\hat{y}$ and $\hat{d}$ respectively. $\hat{m}$ acts as upper-bound of the number of counts ABC123 can regress.

$$\hat{y} = \sum_{h,w} \hat{d}_{(h,w)} \tag{6.1}$$

where $\hat{d}_{(h,w)}$ denotes the density value for pixel (h, w). We achieve this by using $\hat{m}$ convolutional up-sampling heads on top of a vision transformer backbone. We use a vision transformer backbone due to its globally receptive field and self-attention mechanism, which we showed is crucial to generating a complex understanding in counting settings in Chapter 5. Each head regresses a single pixel-wise density map prediction and count prediction from a patch-wise low-resolution high-dimensional feature space. All of the count regression heads are identical and share their weights.

Similar to other contemporary methods [23, 25, 110, 117], we use the pixel-wise distance as our loss for the regression stage. Specifically, we use $||d - \hat{d}||_1$ where d and $\hat{d}$ are the ground truth and predicted density maps. This produced more accurate results than using the more common L_2 distance between the d and $\hat{d}$; see the ablations tests in Section 6.5.1 for the full comparison.

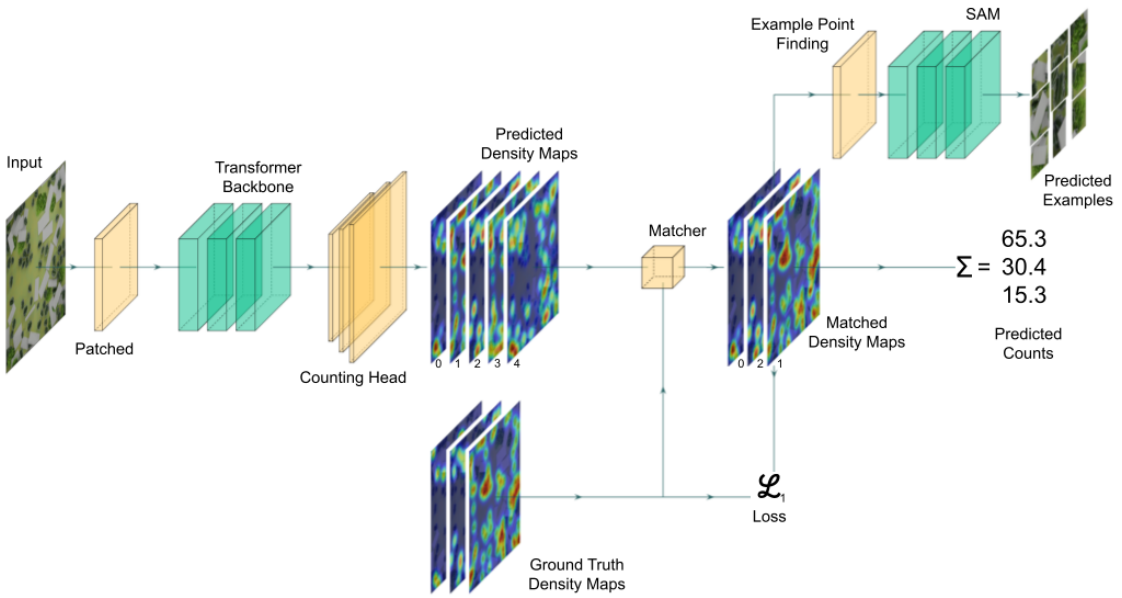


Figure 6.2: The ABC123 pipeline. Our method learns to count objects of multiple novel classes without needing exemplar images. During training and quantitative evaluation, the matcher aligns the unguided predictions to the ground truth labels. The example prediction stage locates instances associated with each generated count.

6.3.2 Matching

In single-class or class-specific settings, there is a single prediction-label pair, one generated count and one ground truth class. In exemplar-based multi-class settings there is an introduced ambiguity as the exemplar images may not reflect the complete variation of the desired objects' appearances, *e.g.* if the exemplars are of blue cars, should the network count only blue cars or all cars? In multi-class exemplar-free settings, this ambiguity is increased significantly. There are multiple predictions as well as multiple labels, without a clearly defined pairing. This resembles class-discovery [141, 161] and clustering [66] problems, where the number and cardinality of new classes are not necessarily known.

In keeping with these fields, we find correspondences between the set of m known counts and the set of $\hat{m}$ predicted counts. These correspondences are required to facilitate training and to evaluate the accuracy. The binary correspondence matrix is defined as $\mathcal{X} = \{0,1\}^{m \times \hat{m}}$ where $\mathcal{X}_{i,j} = 1$ iff prediction j is assigned to

label i. A problem instance is described by an $m \times \hat{m}$ cost matrix C, where $C_{i,j}$ is the cost of matching ground truth label i and prediction j. The goal is to find the complete assignment of predictions to labels with minimal cost. Formally, the optimal assignment has a cost

$$\min_{\mathcal{X}} \sum_{i=0}^{m} \sum_{j=0}^{\hat{m}} C_{i,j} \cdot \mathcal{X}_{i,j} \tag{6.2}$$

Specifically, our cost function is defined as the pixel-wise distance of the normalised ground truth density map d_i and the predicted density map $\hat{d}_j$.

$$C_{i,j} = \left\| \frac{d_i}{||d_i||_2} - \frac{\hat{d}_j}{||\hat{d}_j||_2} \right\|_2 \tag{6.3}$$

The normalisation ensures the matching is done on the locality of the counted objects rather than the magnitude of the prediction itself. We found empirically that this stricter constraint improves the convergence speed of the network to generate density maps that localise the instances. We use the Hungarian algorithm [145], specifically the Jonker-Volgenant algorithm outlined in Crouse [144], to solve for $\mathcal{X}$ robustly.

The supervision loss for each image is the sum of the L_1 difference of all of the ground truth density maps and their matched predictions as:

$$\mathcal{L} = \sum_{i,j}^{m,\hat{m}} ||d_i - \hat{d}_j||_1 \cdot \mathcal{X}_{i,j} \tag{6.4}$$

It should be noted that every label has an associated prediction, but the inverse is not the case, as generally $\hat{m} > m$. This means we do not impose a loss on

the unmatched density maps. This allows the network to generate more nuanced count definitions as it does not punish valid-but-unknown counts which are likely present in any counting setting.

As is usual [66, 141, 161, 162], we use the same matching procedure to quantitatively evaluate our performance at inference-time. It should be noted that as this matching stage uses the ground-truth density maps, it could be used to significantly benefit a method's quantitative results without improving its deployment capabilities. Specifically, a method could in theory predict all possible density maps and use the matching stage to pick the correct one.

As MCAC has a maximum of 4 classes present in an image, we limit our method to generating 5 density maps to minimise this behaviour. The effect of this is explored in the ablation tests in Section 6.5.3. We also normalise the density maps between 0 and 1 to ensure we are only matching to the locations of objects rather than the counts themselves. We explore the impact of this normalisation in more detail in the ablation tests in Section 6.5.2.

6.3.3 Example Discovery

While exemplar-free counting saves a user time, as no manual intervention is required, it does require the user to interpret the results. A set of scalar count values can be unclear as it is not always obvious which count corresponds to which type of object in the input image. Density maps can also often be difficult to interpret, especially in high-density situations; see Figure 6.4 for examples of difficult-to-interpret counts and density maps.

To aid the user in understanding to which class a generated count corresponds, we propose flipping the usual exemplar-based paradigm. Instead of using exemplar images to define the type to count, we find examples of the type counted. We achieve this by finding points that are likely to belong to instances of the counted

class from the predicted density map and using them as seed inputs for a pre-trained class-agnostic segmentation method [158]. For class i, we find a set of representative points, $\mathcal{P}_i = \{p_{i,0}, p_{i,1}, \dots, p_{i,n}\}$, by locating pixel locations that have a high activation in the density map for class i, $\hat{d}_i$, and a low activation in the density maps for the other predicted classes, $\hat{d}_j$. Formally:

$$p_{i,0} = \underset{h,w}{\operatorname{argmax}} \left(\hat{d}_{i,(h,w)} - \max_{j \neq i} \left(\hat{d}_{j,(h,w)} \right) \right) \tag{6.5}$$

where $\hat{d}_{i,(h,w)}$ is the value of pixel (h, w) of the regressed density map for class i.

To find a diverse set of initialisation points for multiple instance examples, we exclude an area around $p_{i,0}$ when searching for $p_{i,1}$. To capture the range of variation of the counted instances, we select the three points from $\mathcal{P}_i$ which have the largest latent feature distance in the final layer of the transformer backbone to use as our 3 initialisation points. We use these points as seed inputs for a pre-trained segmentation method, SAM [158].

The end user is presented with cropped areas of the query image centred on these segmentations. See Figure 6.6 for visual presentations of cases with counts and found examples. We use this approach because SAM is often not accurate enough to segment a singular whole object given only a singular point, as is shown in Figure 6.3. However, the expanded bounding boxes provide more context, allowing an end user to infer the object counted.

6.3.4 Implementation

We use the ViT-Small [36] architecture for our backbone due to its lightweight nature and for a fair comparison to methods that use the Resnet-50 backbone, such as FamNet[23] and BMNet [24]. ViT-S has a similar number of parameters, throughput, and supervised ImageNet performance as ResNet-50 [149]. Due to

Figure 6.3: Examples from a naive example finding stage using peak finding and a pre-trained segmentation method [158]. Often, the found segmentations are not a representation of the whole counted object and so are not informative for the user.

this lightweight backbone, ABC123 is trainable in less than eight hours using two Nvidia 1080Ti GPUs. It takes less than two hours to train just the head with a frozen pre-trained backbone (ABC123❄).

Since vision transformers typically demand substantial training data, we initialised our transformer backbone with weights sourced from Caron et al. [38] in the same way as in Chapter 5. This self-supervised pre-training process endows the network with an understanding of meaningful image features before exposure to our dataset and without supervision. This approach reduces the risk of overfitting when the model is then trained on our dataset.

Our counting heads increase the patch-wise resolution of the trained counting features from $k \times (28 \times 28)$ to a pixel-wise density map prediction of $\hat{m} \times (224 \times 224)$, where k is the dimensionality of the transformer features and $\hat{m}$ is the maximum number of predicted counts. For ABC123, $k = 384$. Each counting head is comprised of a linear projection from $384 \times (28 \times 28)$ to a $(\hat{m} \times 16) \times (28 \times 28)$ feature space. This is then split into $\hat{m}$ separate $16 \times 28 \times 28$ feature maps, each of which is then upsampled to regress $\hat{m}$ pixel-wise density maps. The upsampling is comprised of 3 Conv-ReLU-Upsample blocks followed by a convolutional layer. Each upsampling operation

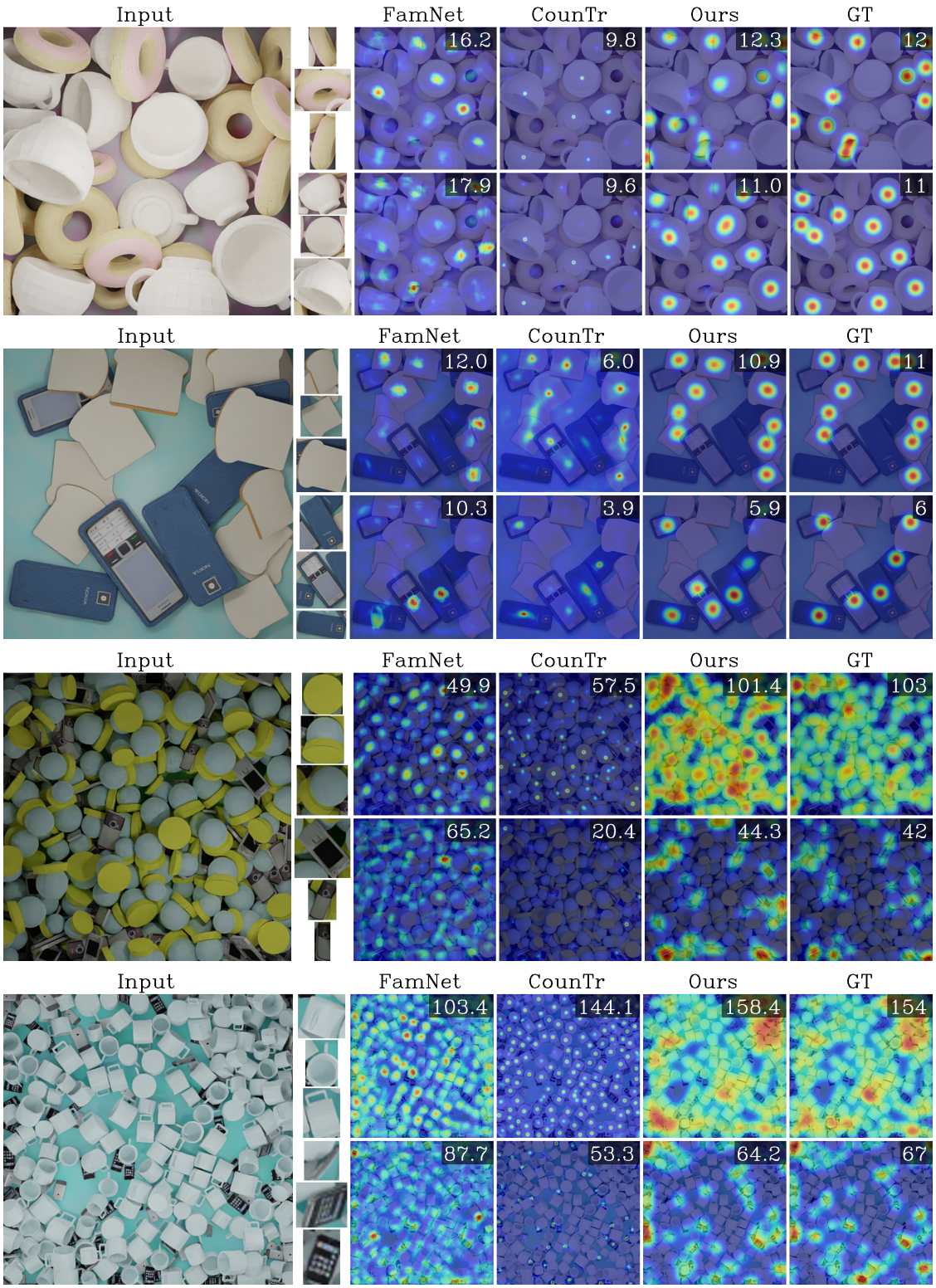


Figure 6.4: Comparison to other methods on MCAC. ABC123 produces more accurate results than the exemplar-based methods without using exemplar images. The ground truth (GT) and predicted counts are shown in the top right corner of their respective density maps.

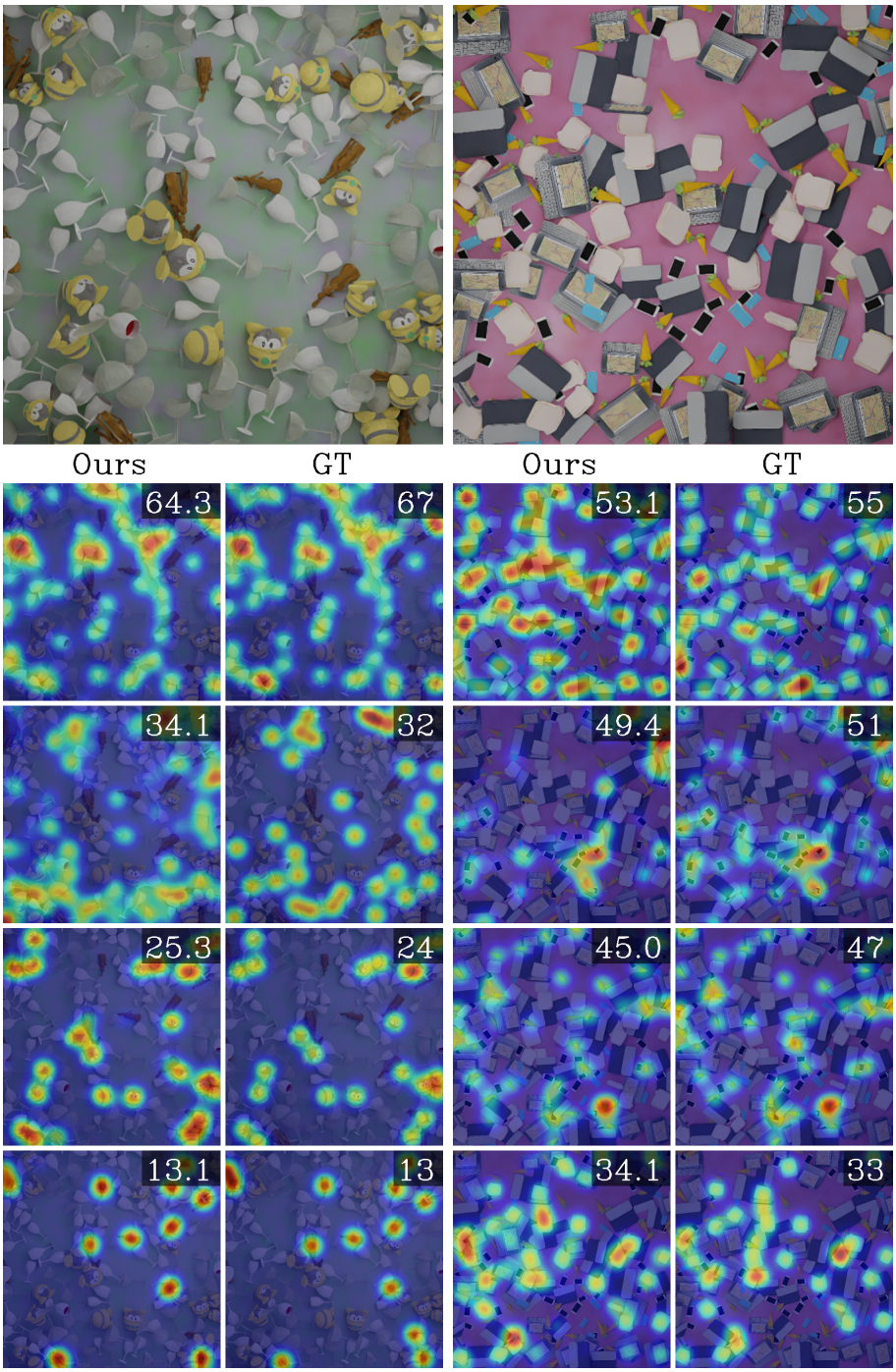


Figure 6.5: Results of our method on images with 4 classes. ABC123 can generate accurate counts and meaningful density maps from images with four novel classes. MCAC has between one and four classes of objects per image.

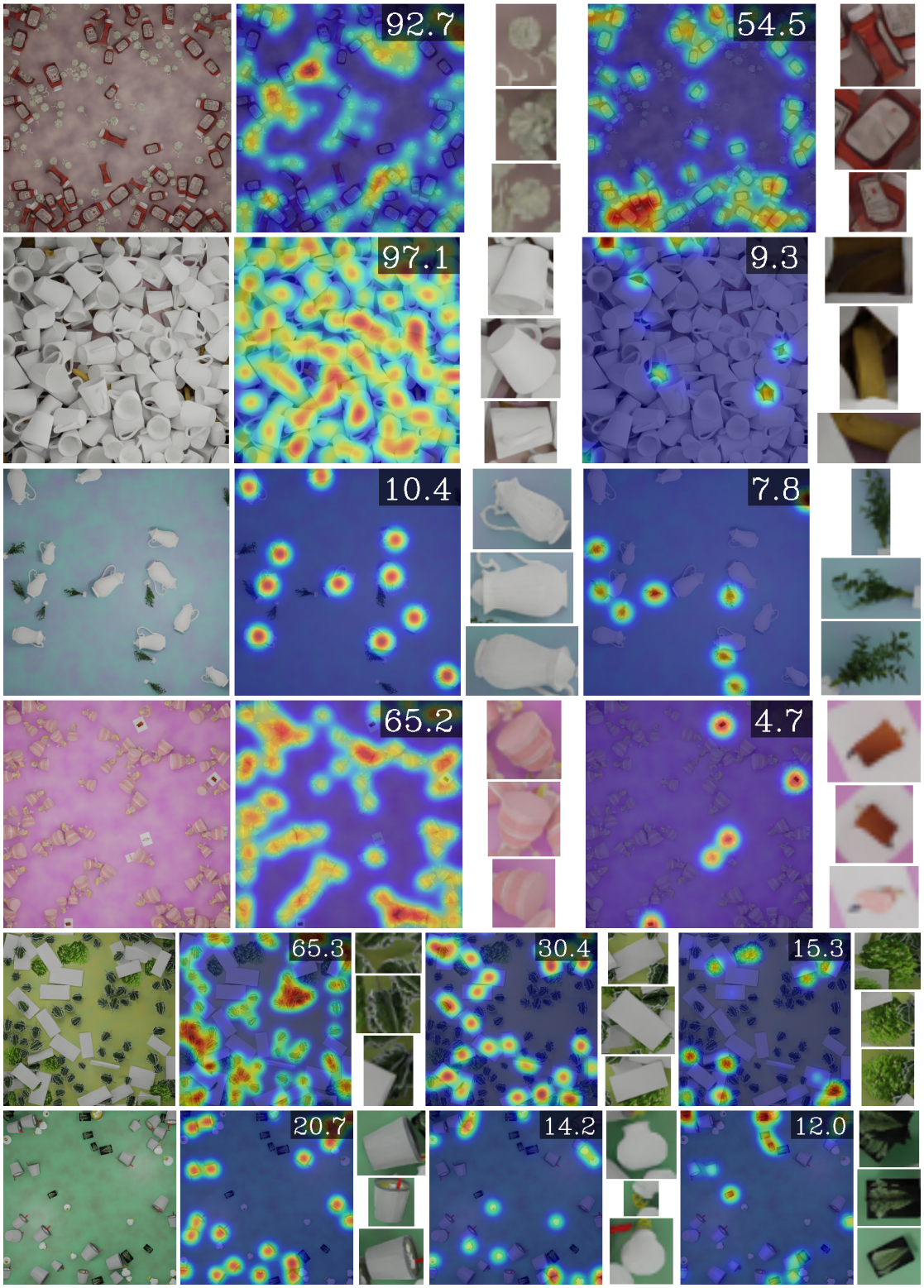


Figure 6.6: Example Finding. Using our density map predictions, we generate example bounding boxes to aid a user in understanding what has been counted.

bilinearly interpolates features increasing the resolution by 2×. Each convolutional layer decreases the feature dimensionality by $\frac{1}{2}$. We use a 7×7 convolutional kernel for each stage in order to increase the receptive field of this upsampling.

As the largest number of classes in a single image in MCAC is four ($m \leq 4$), we set $\hat{m} = 5$. This is the minimum number that ensures that the method has the capacity to generate a count for every defined class in a given image as well as at least one valid-but-unknown count, the value of which is discussed in more detail in Section 6.5.4. The effect of varying $\hat{m}$ on both quantitative results and deployment time utility is further discussed in the ablation tests in Section 6.5.3.

Our choice of the ViT-Small backbone limits the resolution of our input image to (224×224) as opposed to the ≥(384×384) resolution used by contemporary methods with ResNet-50 or larger ViT backbones. However, it increases the ability to train the method on a broader range of hardware and allows for a fairer comparison to other contemporary methods. For the example discovery stage, we use a frozen pre-trained ViT-B SAM model backbone [158] to generate the original segmentations.

6.4 Results

In this section, we outline how we will fairly compare our method to other recent counting methods before demonstrating the superiority of ABC123 to those methods.

6.4.1 Benchmarking Methods

We evaluate our method against two trivial baseline methods which predict the training-set mean or median count for all inference images, as in Section 5.4.1. As there are no previous multi-class exemplar-free class-agnostic counting methods, we compare ABC123 to exemplar-based methods using separate exemplars from each of the classes present. We compare our method to FamNet [23], BMNet [24], and CounTR [25] on MCAC.

To provide greater context to our method's performance in exemplar-free settings, we compare to the above methods as well as CounTR in its zero-shot configuration and RCC on MCAC-M1, the subset of MCAC with only a single type of object present per image. These are the only exemplar-free class-agnostic counting methods with publicly available implementations.

To ensure a fair comparison, we follow the suggested procedure laid out in Section 3.3.2, counting only instances which are less than 70% occluded and evaluating the exemplar-based methods using the bounding boxes of the three least occluded instances of each class.

6.4.2 Benchmarking Metrics

As in Section 5.3.3, we use Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to compare the performance of various methods. Following Nguyen et al. [163], we also use Normalised Absolute Error (NAE) and Squared Relative Error (SRE).

$$NAE = \frac{1}{nm} \sum_{j=1}^{n} \sum_{i=1}^{m_j} \frac{|y_i - \hat{y}_i|}{y_i} \tag{6.6}$$

$$SRE = \sqrt{\frac{1}{nm} \sum_{j=1}^{n} \sum_{i=1}^{m_j} \frac{(y_i - \hat{y}_i)^2}{y_i}} \tag{6.7}$$

where n is the number of test images, m_j is the number of classes in image j, and y_i and $\hat{y}_i$ are the ground truth and the predicted number of objects of class i in image j respectively.

In contrast to the absolute error metrics (MAE and RMSE), the relative error metrics (NAE and SRE) provide a more practical perspective on visual counting.

Specifically, these relative errors acknowledge that the impact of the same count error is more significant in images with fewer objects compared to those with a greater quantity of objects, *e.g.* an error of 5 is much more significant when there are 4 objects present in an image than when there are 400.

6.4.3 MCAC

We achieve significantly better results than contemporary exemplar-based methods, FamNet, BMNet, and CounTR, on MCAC both quantitatively and qualitatively without needing exemplars; see Table 6.1 for results and Figure 6.4 for comparative visual examples. As seen in Figure 6.4, FamNet often fails to discriminate between objects of different kinds when they are visually similar or in high-density applications. Both quantitatively and qualitatively, BMNet and CounTR outperform FamNet. However, in many cases, they appear to count the 'most obvious' objects in the image regardless of the provided exemplar images even when their visual appearance is dissimilar.

Our method performs well on images with 4 classes even when they have high intra-class appearance variation and low inter-class variation; see Figure 6.5. High intra-class appearance differences are usually due to the object in question having different colours on different sides or significantly different silhouettes depending on their orientation. Beyond their poor performance, another downside to current exemplar-based class-agnostic counting methods is that while they have some multi-class capabilities, they all take a single exemplar at a time, producing only one count. Due to this, the time per image scales as $O(m)$ with the number of classes. This is slow and inefficient as compared to our method, which generates all counts simultaneously, $O(1)$.

As would be expected, the performance of all of the methods improves when evaluating on MCAC-M1, the images from MCAC with only a single class present;

see Table 6.2 for the complete results. This is due to a reduction in the ambiguity of the type of object to be counted. This was more significant when the methods were trained on MCAC-M1 instead of MCAC. In this training configuration, all of the other methods learnt a broader definition of similarity because there was no chance that they would accidentally combine classes or count instances from another class.

RCC performs well on MCAC-M1, showing the strength of the simple count-wise loss in cases where there is little ambiguity as to what is to be counted. In contrast to other methods, ABC123 trained on MCAC-M1 has a similar performance to when it is trained on the full MCAC dataset. It does not learn a significantly different understanding of what is and is not a relevant object in this setting. This further demonstrates our method's ability to deal with high intra-class variance and avoid combining classes. Training ABC123 with only a single head ($\hat{m} = 1$) and no matching stage has very similar performance to using its default ($\hat{m} = 5$) configuration with a matching stage. This further confirms that the matching head in our default configuration does not provide an unfair advantage to our method's quantitative results.

6.4.4 Applicability to FSC-133/147/LVIS

ABC123 trained on only MCAC, a synthetic dataset, produces accurate results when applied to FSC-133/147/LVIS, the standard photographic datasets, as seen in Figure 6.7 and Tables 6.3 and 6.4. However, it often finds valid-but-unknown counts. As seen in Figure 6.8, the generated counts are correct for the type of object counted, but the type counted is often not aligned with the labels in the original dataset. Classes are often divided into sub-classes, and unlabelled classes are discovered.

There are three clear discrepancies between MCAC and FSC that could affect the applicability of a method trained on MCAC and then applied to FSC-133/147/LVIS, namely cross-domain difference, class boundary definitions and

maximum count value.

Cross-Domain Image Differences. As has been noted in many papers, applying a method solely trained on synthetic data to real photographic data is likely to have negative results [164]. Since images in the MCAC dataset are synthetic and simple, they do not reflect all of the variation found in FSC. For example, all images in MCAC are taken from above with no camera noise or harsh shadows. That being said, as is seen in Figure 6.7, the generalised counting ability gained from training on MCAC clearly translates into the more varied and complex images in FSC.

Inter- and Intra-Class Definitions. The frequent discovery of valid-but-unknown counts is due to a difference in the definition of what is similar in the datasets. As discussed in Section 3.3, MCAC associates a count with objects of the same mesh and texture. In contrast, FSC, which is labelled by hand and uses high-level semantic understanding, often groups objects with significantly different geometries, colours, or textures. This means that ABC123 when trained on MCAC and applied to images from FSC finds unlabelled classes and generates multiple identical counts because it enumerates different parts of the same objects, or subdivides classes by colour or geometry such as splitting 'chairs' into 'red chairs' and 'grey chairs'. Interestingly, an unguided segmentation method, SAM [158], which identifies instances' relations, often finds the same class divisions as ABC123; see Figure 6.8.

Novel class-discovery and duplicate part counts do not affect quantitative results on FSC as the matching stage removes them. However, sub-class counting causes erroneous numerical results.

To generate quantitative results, we borrow the approach of other open set methods [158, 159, 165], combining sub-classes. We perform this mapping by combining the density maps of sub-class counts, either by the summation of both

Method	Shots	Val Set				Test Set			
		MAE	RMSE	NAE	SRE	MAE	RMSE	NAE	SRE
Mean	N/A	39.87	53.56	3.07	11.40	42.67	59.68	2.79	10.93
Median	N/A	36.25	58.15	1.51	6.70	39.81	65.36	1.38	6.73
Exemplar-based									
FamNet+ [23]	3	24.76	41.12	1.12	6.86	26.40	45.52	1.04	6.87
BMNet+ [24]	3	15.83	27.07	0.71	4.97	17.29	29.83	0.75	6.08
CounTR [25]	3	15.07	26.26	0.63	4.79	16.12	29.28	0.67	5.71
Exemplar-free									
ABC123 ❄	0	14.64	23.67	0.46	2.97	15.76	25.72	0.45	3.11
ABC123	0	**8.96**	**15.93**	**0.29**	**2.02**	**9.52**	**17.64**	**0.28**	**2.23**

Table 6.1: Comparison to SOTA methods on MCAC. We significantly outperform methods which use exemplar images and test-time adaptation without requiring them. ABC123❄ denotes our method trained with a frozen pre-trained backbone.

the separate counts or by trying to combine the density maps using the maximum density at a given point and then counting the instances. The two approaches only differ in cases where sub-class density maps overlap. The maximum density map configuration produces results which are competitive with other contemporary methods while the density map summation is clearly SOTA, as presented in Table 6.3.

Maximum Count Values. As the standard benchmark evaluation metrics rely on absolute count error they are all very sensitive to even a small number of very dense images, as discussed in Section 5.4.6. We found that the errors on the few images with counts between 300 (the largest count in MCAC) and 3000 (the largest count in FSC) corrupted the metrics, making comparison to other literature more difficult. For this reason these images are excluded from the quantitative evaluation in Table 6.3. These exclusions amount to 3.0% of the validation set and 1.1% of the test set. It should be noted that the relative rankings of the methods with and without this exclusion remain the same.

This adjustment was not made for the results presented in Table 6.4 facilitating better comparison to other methods. This is possible because there are no images in the testing set of FSCD-LVIS with counts significantly over 300.

Method	Train m	Shots	$\hat{m}$	Val Set MAE	Val Set RMSE	Val Set NAE	Val Set SRE	Test Set MAE	Test Set RMSE	Test Set NAE	Test Set SRE
Mean	N/A	N/A	N/A	53.36	67.14	3.53	13.46	58.54	75.58	3.37	13.27
Median	N/A	N/A	N/A	45.98	76.64	1.08	6.68	51.35	86.61	1.03	7.00
Exemplar-based											
FamNet+ [23]	1 – 4	3	1	24.97	48.63	0.36	3.79	28.31	54.88	0.35	3.97
FamNet+ [23]	1	3	1	12.54	24.69	0.37	4.71	13.97	26.19	0.25	2.12
BMNet+ [24]	1 – 4	3	1	11.70	23.08	0.26	2.39	11.57	22.25	0.24	1.96
BMNet+ [24]	1	3	1	6.82	12.84	0.25	2.95	8.05	14.57	0.19	1.43
CounTR [25]	1 – 4	3	1	11.44	21.37	0.33	2.36	10.91	21.70	0.29	2.01
CounTR [25]	1 – 4	0	1	13.57	25.53	0.30	2.48	13.09	25.72	0.29	2.41
CounTR [25]	1	3	1	9.00	16.91	0.41	3.56	9.96	18.92	0.38	2.93
CounTR [25]	1	0	1	9.16	17.13	0.42	3.56	10.10	19.10	0.40	3.02
Exemplar-free											
CounTR†[25]	1	0	1	11.46	21.24	0.35	2.78	12.54	23.84	0.31	2.38
RCC [125]	1	0	1	7.78	15.40	0.24	2.71	8.81	16.92	0.19	1.73
ABC123 ❄	1	0	1	11.38	19.73	0.40	3.51	14.31	25.40	0.37	2.79
ABC123 ❄	1	0	5	10.78	18.83	0.28	1.97	13.23	24.57	0.29	2.39
ABC123 ❄	1 – 4	0	5	10.98	18.85	0.30	1.93	13.13	23.93	0.29	2.18
ABC123	1	0	1	5.85	12.91	0.24	3.37	7.53	15.69	0.22	2.19
ABC123	1	0	5	**5.82**	**11.74**	**0.15**	**1.22**	7.54	15.30	0.21	1.87
ABC123	1 – 4	0	5	6.08	12.62	0.16	**1.22**	**6.82**	**14.70**	**0.16**	**1.51**

Table 6.2: Comparison to SOTA methods on MCAC-M1. Methods are either trained on the full multi-class dataset ($1 \leq m \leq 4$) or MCAC-M1 ($m = 1$). 'Shots' denotes the number of exemplar images per query at inference time and $\hat{m}$ denotes the number of predictions the method generates per query. CounTR† is an exemplar-free adaption of CounTR. ABC123 outperforms other methods when trained in both single or multi-class settings.

Method	Shots	Sub-Class Combine	Val Set MAE	Val Set RMSE	Val Set NAE	Val Set SRE	Test Set MAE	Test Set RMSE	Test Set NAE	Test Set SRE
Exemplar-based										
FamNet+	3	N/A	25.83	46.31	0.50	4.29	28.05	45.59	0.48	4.33
BMNet+	3	N/A	29.47	53.15	0.51	4.72	30.74	52.00	0.47	4.68
CounTR	3	N/A	21.22	40.28	0.47	3.85	21.09	40.79	0.38	3.66
LOCA	3	N/A	25.70	48.64	0.45	4.30	29.93	49.89	0.48	4.62
Exemplar-free										
CounTR †	0	N/A	23.50	45.83	0.42	4.06	23.57	42.00	0.39	3.85
LOCA	0	N/A	29.37	54.01	0.51	4.84	33.96	56.66	0.53	5.17
ABC123	0	Max	*19.56*	46.71	*0.20*	*3.54*	*22.43*	47.35	*0.22*	*3.70*
ABC123	0	Sum	**11.13**	**34.47**	**0.12**	**2.44**	**11.75**	**33.41**	**0.11**	**2.38**

Table 6.3: Comparison to SOTA methods when trained on MCAC and applied to the cases in FSC147 with fewer than 300 objects. Combining sub-class density maps with a sum rather than a max is more effective as it is more accurate in cases where instances of the sub classes are spatially close or overlapping as both instances are counted completely. CounTR † is an exemplar-free modification of CounTR.

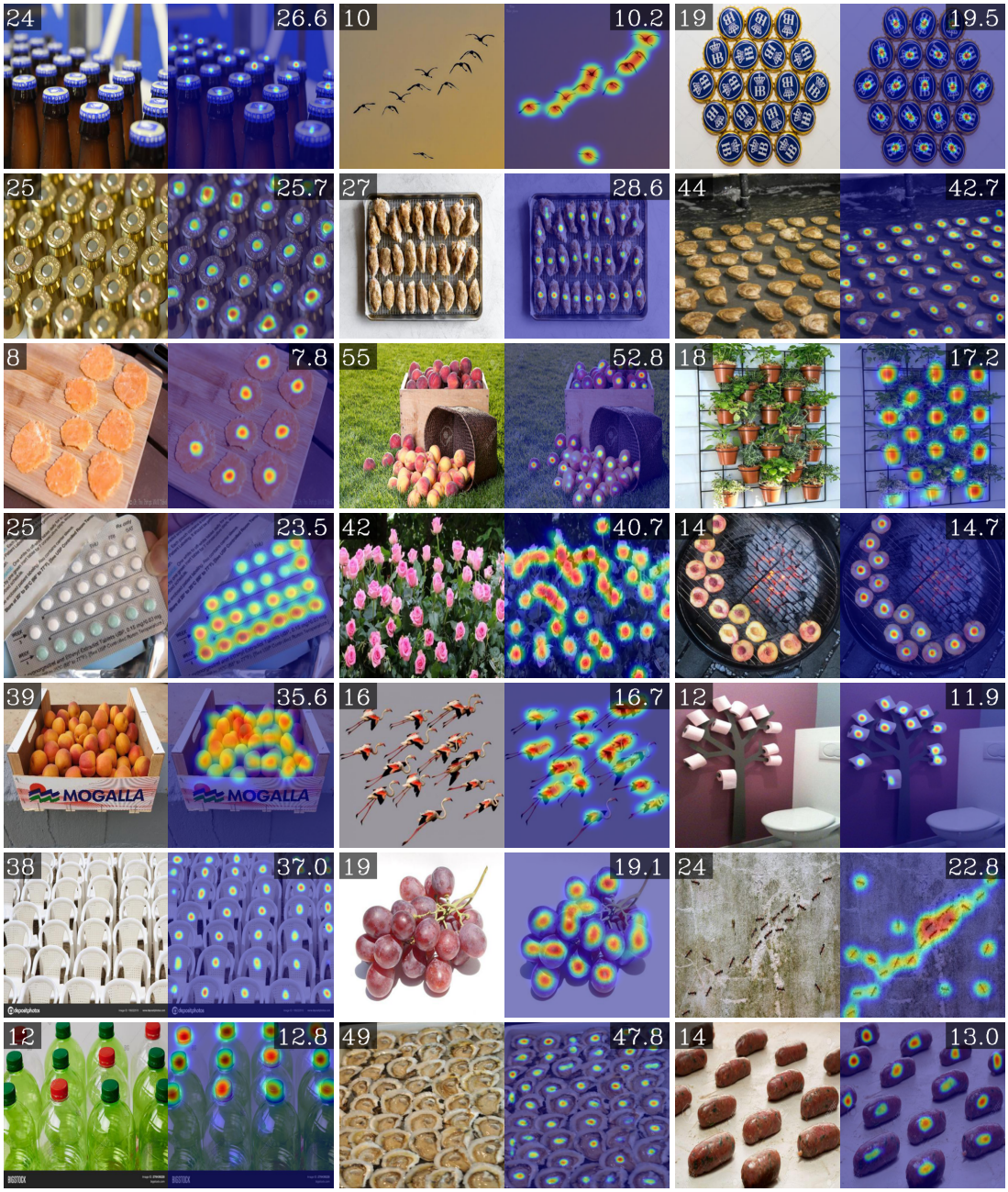


Figure 6.7: ABC123 trained on MCAC generates accurate counts on unambiguous images from FSC. The input image and ground truth count (top left) and the regressed density map with the predicted count (top right) are shown.

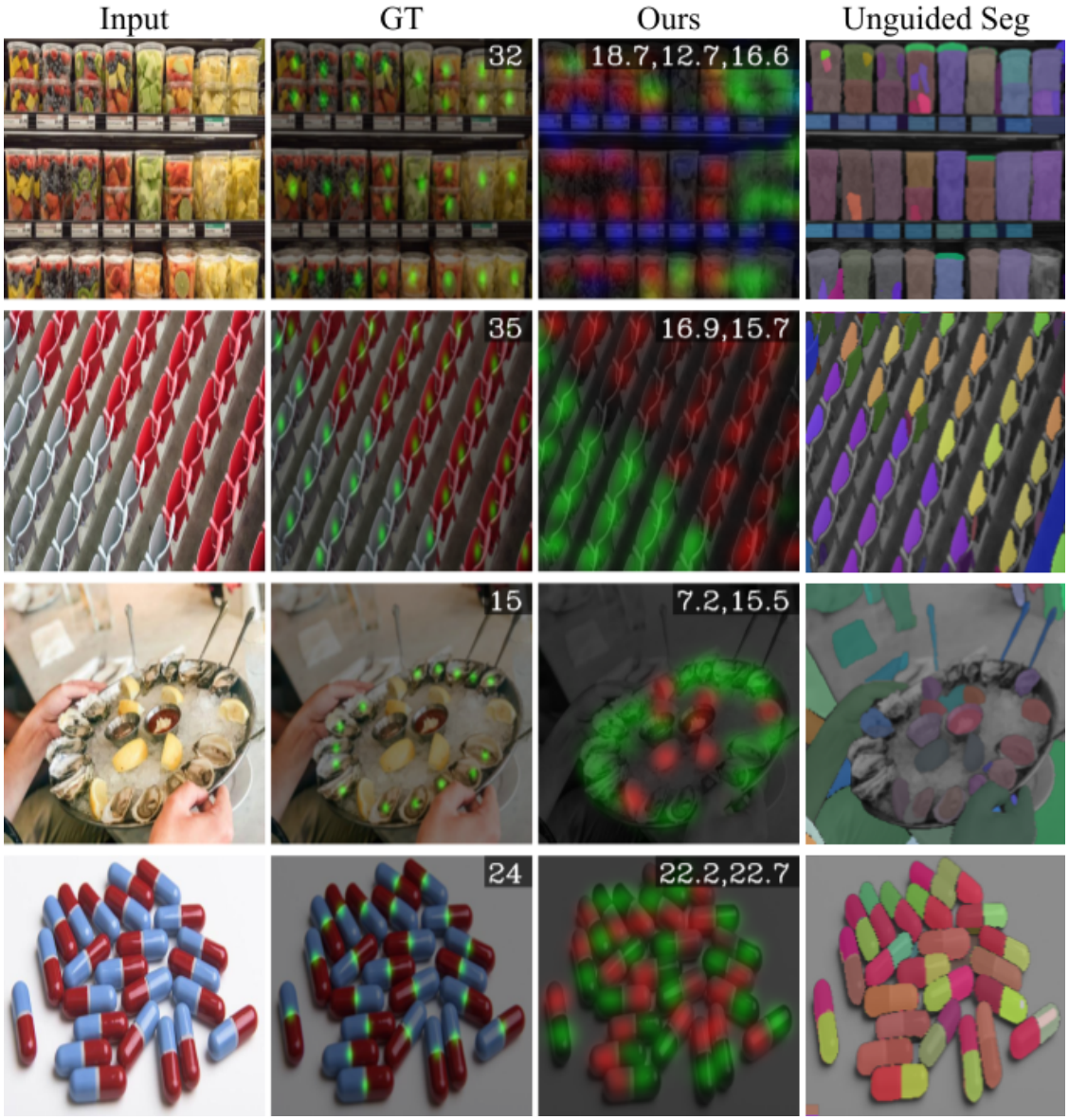


Figure 6.8: ABC123 trained on MCAC applied to ambiguous images from FSC-133. ABC123 generates accurate counts for the classes it locates. However, these classes are often not aligned with the human annotations. ABC123 often discovers unlabelled classes, divides labelled classes into sub-classes by colour or shape, or counts the same object multiple times by subdividing each object into component parts. Column 2: the ground truth (GT) density map is overlaid in green, Column 3: the predicted density maps are overlaid in red, green, and blue, with the associated counts in the the top corner, Column 4: The outputs of an unguided segmentation method, coloured by feature similarity.

Method	MAE	RMSE	NAE	SRE
Exemplar-based				
FamNet+	60.53	84.00	1.82	14.58
FSOD	61.31	64.10	1.02	6.94
FSDetView [166]	24.89	31.34	0.54	4.46
Counting-DETR	18.51	**24.48**	0.45	3.99
Exemplar-free				
ABC123	**16.63**	35.30	**0.17**	**3.02**

Table 6.4: Comparison to the relevant SOTA methods when trained on MCAC and applied to the test set of FSCD-LVIS [163]. Combining sub-class density maps with a max operation.

6.5 Ablations

In this section, we explain and validate various design decisions including our loss and matching stage scaling. We also show the effects of generating large numbers of predictions and penalising valid-but-unknown counts on quantitative benchmarks.

6.5.1 Validating Our Loss

We showed in Chapter 5 that using image-wise count loss functions, like the absolute count error, can be beneficial over pixel-wise loss functions as they allow a network to learn its own idea of positional salience. To validate our choice of a pixel-wise loss function for this more complex task, we test our loss, the pixel-wise error $||d - \hat{d}||_1$, against both the pixel-wise error squared $||d - \hat{d}||_2^2$, as in [23, 25, 146], and the image-wise count percentage error $|y - \hat{y}|/y$ where d and $\hat{d}$ are the ground truth and predicted density maps, and y and $\hat{y}$ are the ground truth and predicted counts.

Using an image-wise count loss creates poor results with somewhat arbitrary density maps and high count errors; see Table 6.5. This is likely due to two factors: an increased level of ambiguity and the matching stage now operating on latent features. As the multi-class tasks have significantly more ambiguity than their single-class counterparts, it is more difficult for a network to learn a robust

Method	Loss	Val Set MAE	Val Set RMSE	Val Set NAE	Val Set SRE	Test Set MAE	Test Set RMSE	Test Set NAE	Test Set SRE
ABC123 ❄	Count-MAPE	19.88	38.20	0.48	3.62	25.32	49.18	0.52	4.28
ABC123 ❄	Pixel L_2	16.46	26.54	0.78	4.79	17.92	28.88	0.75	4.77
ABC123 ❄	Pixel L_1	**14.64**	**23.67**	**0.46**	**2.97**	**15.76**	**25.72**	**0.45**	**3.11**
ABC123	Count-MAPE	15.95	29.71	0.50	3.54	16.28	29.82	0.49	3.58
ABC123	Pixel L_2	10.76	19.31	0.49	3.84	11.91	21.42	0.51	4.21
ABC123	Pixel L_1	**8.96**	**15.93**	**0.29**	**2.02**	**9.52**	**17.64**	**0.28**	**2.23**

Table 6.5: The effect of different loss functions during training. Pixel-wise L_1 loss produces significantly better results than the pixel-wise L_2 loss and the count-wise MAPE.

idea of the correct positional salience with only integer labels. The significant change over the pixel-wise supervision is that the matcher which usually operates on the density map to which the loss is directly applied now operates over an unconstrained latent feature. This, along with the added complexity of multi-class images, makes the problem significantly more difficult to understand from purely integer supervision. Matching to the scalar count while using pixel-wise supervision has similar results to matching to the latent features with the overall accuracy being approximately 7% worse.

Using a pixel-wise L_2 loss slightly degrades the performance compared to using the L_1 distance between the ground truth and predicted density maps. Still, the predicted density maps are qualitatively meaningful. This discrepancy is likely due to the increased significance the higher-density edge cases have on the training. See Table 6.5 for the quantitative comparison.

6.5.2 Validating Our Matching Scaling

Normalising the density maps during training and testing is beneficial. We hypothesise that this is because it forces the network to generate the correct localisation early in the training as well as the correct magnitude. This normalisation also decreases the significance of the issue of the matcher cheating to find the

Method	Normalisation		Val Set				Test Set			
	Train	Eval	MAE	RMSE	NAE	SRE	MAE	RMSE	NAE	SRE
ABC123 ❄	None	None	20.31	32.40	0.78	5.40	22.67	36.63	0.92	7.25
ABC123 ❄	None	L_2	21.38	35.08	0.86	6.43	25.46	43.37	1.05	8.67
ABC123 ❄	None	L_∞	20.13	32.15	0.81	5.86	22.76	36.78	0.95	7.64
ABC123 ❄	L_2	None	41.20	66.04	0.87	6.22	45.04	73.41	0.87	6.51
ABC123 ❄	L_2	L_2	**14.64**	**23.67**	**0.46**	**2.97**	**15.76**	**25.72**	**0.45**	**3.11**
ABC123 ❄	L_2	L_∞	17.21	27.94	0.51	3.28	17.97	29.36	0.48	3.30
ABC123 ❄	L_∞	None	22.40	35.80	0.73	4.89	23.93	38.59	0.80	5.86
ABC123 ❄	L_∞	L_2	17.92	28.98	0.67	5.12	20.40	33.39	0.77	6.24
ABC123 ❄	L_∞	L_∞	19.33	30.75	0.67	4.77	20.84	33.61	0.74	5.91
ABC123	None	None	15.46	28.49	0.60	5.12	16.15	29.82	0.69	6.50
ABC123	None	L_2	14.83	27.25	0.68	6.15	17.04	31.88	0.84	8.06
ABC123	None	L_∞	14.75	27.41	0.65	5.92	16.54	30.80	0.77	7.45
ABC123	L_2	None	35.29	61.12	0.67	5.55	38.15	67.89	0.64	5.78
ABC123	L_2	L_2	**8.96**	**15.93**	**0.29**	**2.02**	**9.52**	**17.64**	**0.28**	2.23
ABC123	L_2	L_∞	9.95	18.16	0.32	2.22	9.89	18.55	0.29	**2.20**
ABC123	L_∞	None	15.72	28.83	0.59	4.69	16.11	30.22	0.67	6.16
ABC123	L_∞	L_2	12.17	22.81	0.57	5.09	13.97	27.66	0.71	7.05
ABC123	L_∞	L_∞	12.46	23.00	0.56	4.90	13.96	27.12	0.66	6.37

Table 6.6: Effect of using normalised density maps for the matching stage. Normalisation improves the accuracy of ABC123. L_2 normalisation is more effective than L_∞ during training and evaluation.

correct count. Instead of selecting the correct count from a large set of predicted counts, the network must generate a density map with a relevant localisation. Using the L_2 normalisation over the L_∞ has a slight performance increase for training the network and during evaluation. We believe this is due to the decreased importance of very high-density areas in images having an overly strong effect on the training. See Table 6.6 for a quantitative comparison of the normalisations during the matching stage.

6.5.3 Validating Our Number of Predictions

As hypothesised, generating much larger numbers of predictions ($\hat{m} > 5$) increases the quantitative performance of the network; see Table 6.7 for the complete results. In these cases, the network generates a more diverse set of counts and uses the matching

	Val Set				Test Set				
$\hat{m}$	MAE	RMSE	NAE	SRE	MAE	RMSE	NAE	SRE	Head Utilisation
4	9.43	17.42	0.32	2.57	10.19	19.44	0.33	2.81	100%
5	8.96	15.93	0.29	2.02	9.52	17.64	0.28	2.23	100%
10	8.39	14.93	0.28	1.97	9.08	16.96	0.27	2.15	100%
20	7.78	13.75	0.26	1.82	8.29	15.53	0.24	1.95	85%
50	7.26	12.80	0.25	1.69	7.99	15.14	0.24	1.81	68%
100	7.11	12.81	0.23	1.59	7.43	14.48	0.21	1.72	39%

Table 6.7: Effect of using more count regression heads. Increasing the number of pre-matching predictions improves the quantitative results of our method. However, as the number of heads increases, the percentage of heads that are frequently (> 0.4%) matched decreases.

stage to select the best one. We believe, however, that this increased quantitative performance does not align with a more useful network in most real-world deployment situations. The numerical gain derives purely from the matching stage, which is not present during deployment. In fact, during deployment, these configurations would correspond to outputs which are much more difficult to interpret as a user would have to figure out which of the many outputs was most relevant to their problem.

While increasing the number of heads increases the options that can be generated, the return on useful heads diminishes. When high numbers of heads are used, fewer than half of the generated counts are used, *i.e.* the outputs of some heads were never picked by the matcher. See Table 6.7 for the head utilisation data. This is likely due to these heads not being matched frequently during training, meaning a loss is rarely propagated back through them.

There is also significant redundancy between the heads. The predictions of certain heads over the whole dataset are clearly similar and can be grouped. During our testing, of the 39 utilised heads, there were three groups of, respectively, 13, 6, and 4 heads that were very similar, lowering the effective number of utilised heads to 19.

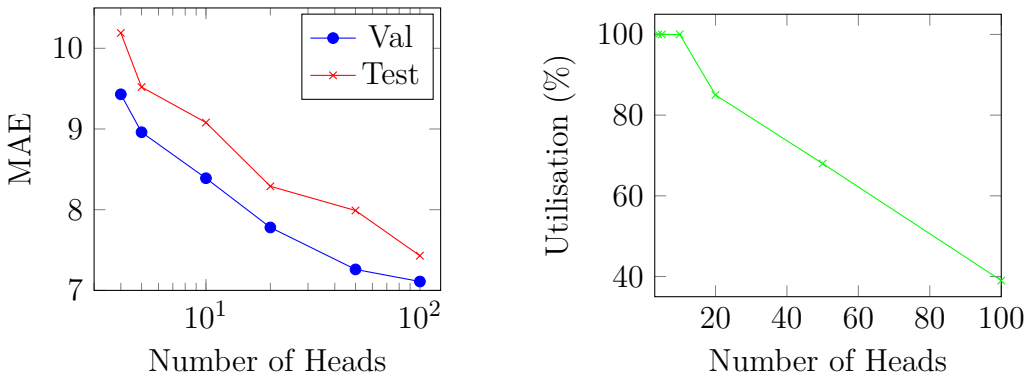


Figure 6.9: Effect of using more count regression heads on the accuracy of ABC123 and the head utilisation.

6.5.4 Validating Valid-But-Unknown Predictions

Here, we show the importance of allowing the generation of valid-but-unknown counts to the success of exemplar-free multi-class class-agnostic counting methods. To test the effect of allowing the network to generate predictions for unlabelled classes uninhibited, we evaluate the performance of ABC123 when the loss is applied to all $\hat{m}$ predictions and as before applying it to only the m matched predictions. When the method is encouraged to generate blank density maps and zero counts for everything but the labelled counts the accuracy is significantly worse. The network generates counts and density maps which are consistently too low. Not only does forcing the network to generate zeros for unlabelled counts harm the networks ability to gain its own nuanced understanding of counting, it also lowers the signal-to-noise ratio of meaningful counting information during training. As most of the images present in MCAC have less than half of the maximum number of types of objects, see Figure 3.11, the network is motivated to generate 0 most of the time during training. The results are presented in Table 6.8.

This shows the clear advantage of allowing the network to generate counts associated with divisions that were not specified by the dataset labels. This is because the network is able to generate more nuanced representations of counting when it is not encumbered with arbitrary rules of what is a 'valid' count. It

		Val Set				Test Set			
$\hat{m}$	loss applied to	MAE	RMSE	NAE	SRE	MAE	RMSE	NAE	SRE
4	all	33.66	49.15	0.87	6.83	36.41	54.88	0.91	7.14
4	m	9.43	17.42	0.32	2.57	10.19	19.44	0.33	2.81
5	all	23.08	42.66	0.84	5.35	24.99	47.87	0.75	5.64
5	m	8.96	15.93	0.29	2.02	9.52	17.64	0.28	2.23

Table 6.8: Effect of applying the loss to all regressed counts compared to only the matched counts. When the loss is only applied to the matched m predictions, it is able to generate counts associated with unlabelled divisions. When the loss is applied to all $\hat{m}$ predictions, all of the counts not associated with a count are forced to 0. This strict approach damages the training of ABC123 as it reduces the capacity of the network to generate nuanced representations of counting.

should be noted that even with the stricter loss, ABC123 is still superior to contemporary methods.

6.6 Conclusion

In this chapter, we present ABC123, a multi-class exemplar-free class-agnostic counter, and show that it is superior to prior exemplar-based methods in multi-class settings. We also show that the understanding of counting ABC123 gains from being trained on synthetic data can be applied to photographic images. ABC123 requires no human input at inference-time, works in complex settings with more than one kind of object present, and outputs easy-to-understand information in the form of examples of the counted objects. Due to this, it has good potential for deployment in various fields.

7

Conclusion

Counting is a fundamental and powerful task. We learn to count early in our childhood and utilise the skill on a daily basis in an unlimited number of settings. While people have the capacity to perform this task manually, in many contexts it is time-consuming and laborious. Simple though it is for people to generalise the idea of counting to objects of unknown types, this ability has proven difficult to automate.

Prior to the work presented in this thesis, automated counting in computer vision relied on either having a known set of classes to count or a large set of information, often in the form of exemplar images and instance-wise labelling, to aid a counter in understanding the task at hand.

In this thesis, we present methods which significantly lower the supervision burden during training, significantly lower the amount of user input required during the deployment of a counter, and significantly increase the abilities of counters to function in multi-class settings. This is achieved by first establishing a set of

paradigms and then proving those paradigms as possible. In this chapter, we summarise the paradigms we proposed and the methods we created which verified them before discussing the scope for future work which expands upon our work and, finally, summarising the overall thesis.

7.1 Contributions

Establishing and Exploring Paradigms

In this work, we establish and formalise some ideas about possible counting abilities as well as rethinking the way counting and clustering problems are usually presented and how this could be improved.

Inspired by human counting abilities, we propose the idea of exemplar-free class-agnostic counting. This concept, which has since been used by other works in the field [25, 116, 146], aims to remove the laborious user input at deployment time. Not only are we able to show that these exemplars of type are not needed during deployment, but we also do away with them during training.

Following the successes of weakly-supervised crowd counting methods, we introduce weak-supervision to class-agnostic counting. The ability to effectively supervise a method with only a count value would significantly reduce the burden on the training-set generation.

We introduce the idea of multi-class class-agnostic counting and exemplar-free multi-class class-agnostic counting. While previously it had been postulated that exemplar-based single-class class-agnostic counting methods could function in settings where multiple classes of objects would be present, it had not been demonstrated. These methods were also all single-class, identifying only one kind of object at a time, scaling as $O(n)$.

Multi-class counting, as we propose it, is the idea that all of the counts are

generated simultaneously, scaling as $O(1)$. Exemplar-free multi-class class-agnostic counting is the goal of achieving this with no prior on type or exemplars of type during deployment. We recognise that in multi-class settings the utility of a method correlates with the interpretability of the produced result. A count without context of what has been counted is not very useful. As such, we introduce the concept of example-finding, a reversal of exemplar-based methods. Instead of specifying exemplar instances to define the type to be counted, we propose finding examples of instances that have been counted, allowing a user to quickly and easily interpret the counts.

It became clear during our work that there were certain abstract ambiguities in counting which we have not seen discussed or even noted in other works. We propose the ideas of valid-but-unknown counts and intrinsic vs non-intrinsic clusterings. Both of these concepts acknowledge the ambiguity inherent to class-agnostic methods. If a method is developed to work on grouping instances of objects of classes it has not been specifically trained to identify, there will always be a level of uncertainty about the 'inliership' of these instances. In both clustering and class-agnostic counting, the specific interpretation of intra- and inter-class boundaries significantly changes the outcomes.

Demonstrating the Paradigms

Beyond proposing various paradigms, we create methods which utilise them and achieve robust and accurate performances.

In Chapter 4, we introduce DMS, a fully differentiable clustering method inspired by the classical mean shift algorithm. We demonstrate that training DMS with pairwise side-information achieved state-of-the-art performance over other machine learning-based clustering methods. We also show that the use of side-information as supervision allowed DMS to be flexible, learning task-specific representations

of similarity. In Chapter 4, we also validate the idea of non-intrinsic clustering, demonstrating the existence of other ways to divide datasets than the distributed labels and showing how these divisions are learnable using only side-information.

In Chapter 5, we introduce RCC which demonstrates that given the constraint that images include a single, or clear 'main', class, state-of-the-art class-agnostic counting can be achieved without exemplars during both training or deployment. We then show that not only can single-class exemplar-free class-agnostic counting methods be trained using this single scalar value supervision, but it is in fact beneficial, allowing a network to learn its own ideas of positional salience. The ability to effectively supervise a method with only a count value significantly reduces the burden on the training-set generation. RCC validates that exemplar-free counting is possible as well as that weak-supervision has a place in the field of class-agnostic counting.

In Chapter 6, we expand upon the capabilities of RCC with ABC123. ABC123 proves that exemplar-free counting is possible in multi-class class-agnostic settings. ABC123 generates state-of-the-art results in these settings, outperforming exemplar-based methods previously assumed to work on multi-class images. The work in Chapter 6 also shows the benefit of allowing a network to consider valid-but-unknown counts. In this chapter, we also demonstrate example finding, showing not only that it is possible but that it has utility.

7.2 Future Work

This thesis lays the groundwork for creating effective counting methods which function in situations with less information than previously posited, though the exploration of this area has only just begun. There is of course scope for methods which generate more accurate results, use even less information during training, or increase the usability and interpretability of the systems in deployment situations.

We believe that the weak-supervision proven to work in Chapter 5 could be utilised in multi-class settings. We also think that introducing the ideas presented in Chapter 4 of hierarchical and orthogonal class relationships to class-agnostic counting could lead to methods which are more accurate and can provide much more interesting and nuanced insights into the distribution of objects found in a single image or across a dataset.

7.3 Closing Remarks

This thesis present a set of works which demonstrate that counting methods can achieve accurate counting results on previously unseen classes with less input than has previously been demonstrated. This takes the form of pair-wise side-information taking the place of class labels in clustering, the removal of both priors on class and exemplars from counting, and, in single-class counting settings, the removal of labelling beyond a single scalar value. We trust that the work presented has contributed to the important area of exemplar-free counting and has demonstrated empirically that this now widely adopted paradigm will have a substantial impact on automated counting systems in real-world applications.

Bibliography

[1] Hyojun Go, Junyoung Byun, Byeongjun Park, Myung-Ae Choi, Seunghwa Yoo, and Changick Kim. Fine-grained multi-class object counting. In *International Conference on Image Processing*, 2021.

[2] Carlos Arteta, Victor Lempitsky, and Andrew Zisserman. Counting in the wild. In *European conference on computer vision*, 2016.

[3] Carlos Arteta, Victor Lempitsky, J Alison Noble, and Andrew Zisserman. Interactive object counting. In *European conference on computer vision*, 2014.

[4] Erika Lu, Weidi Xie, and Andrew Zisserman. Class-agnostic counting. In *Asian Conference on Computer Vision*, 2018.

[5] T Nathan Mundhenk, Goran Konjevod, Wesam A Sakla, and Kofi Boakye. A large contextual dataset for classification, detection and counting of cars with deep learning. In *European conference on computer vision*, 2016.

[6] Meng Wang and Xiaogang Wang. Automatic adaptation of a generic pedestrian detector to a specific traffic scene. In *Computer vision and pattern recognition*, 2011.

[7] Chuan Wang, Hua Zhang, Liang Yang, Si Liu, and Xiaochun Cao. Deep people counting in extremely dense crowds. In *International conference on Multimedia*, 2015.

[8] Aparecido Nilceu Marana, SA Velastin, LF Costa, and RA Lotufo. Estimation of crowd density using image processing. *Image Processing for Security Applications*, 1997.

[9] Xinkun Cao, Zhipeng Wang, Yanyun Zhao, and Fei Su. Scale aggregation network for accurate and efficient crowd counting. In *European conference on computer vision*, 2018.

[10] Olga Barinova, Victor Lempitsky, and Pushmeet Kholi. On detection of multiple object instances using hough transforms. *Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[11] Weidi Xie, J Alison Noble, and Andrew Zisserman. Microscopy cell counting and detection with fully convolutional regression networks. *Computer methods in biomechanics and biomedical engineering: Imaging & Visualization*, 2018.

[12] Roberto Morelli, Luca Clissa, Roberto Amici, Matteo Cerri, Timna Hitrec, Marco Luppi, Lorenzo Rinaldi, Fabio Squarcio, and Antonio Zoccoli. Automating cell counting in fluorescent microscopy through deep learning with c-resunet. *Scientific Reports*, 2021.

[13] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 1980.

[14] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 2012.

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 2014.

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer vision and pattern recognition*, 2015.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer vision and pattern recognition*, 2016.

[21] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *International Conference on Computer Vision*, 2017.

[22] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic Instance Segmentation for Autonomous Driving. In *Computer Vision and Pattern Recognition Workshops*, 2017.

[23] Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. Learning to count everything. In *Computer vision and pattern recognition*, 2021.

[24] Min Shi, Hao Lu, Chen Feng, Chengxin Liu, and Zhiguo Cao. Represent, compare, and learn: A similarity-aware framework for class-agnostic counting. In *Computer vision and pattern recognition*, 2022.

[25] Chang Liu, Yujie Zhong, Andrew Zisserman, and Weidi Xie. Countr: Transformer-based generalised visual counting. *arXiv preprint arXiv:2208.13721*, 2022.

[26] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, 2016.

[27] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *Advances in Neural Information Processing Systems*, 2014.

[28] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Computer vision and pattern recognition*, 2018.

[29] Shuai Bai, Zhiqun He, Yu Qiao, Hanzhe Hu, Wei Wu, and Junjie Yan. Adaptive dilated network with self-correction supervision for counting. In *Computer vision and pattern recognition*, 2020.

[30] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2net: A new multi-scale backbone architecture. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[31] Bekhzod Olimov, Barathi Subramanian, Rakhmonov Akhrorjon Akhmadjon Ugli, Jea-Soo Kim, and Jeonghong Kim. Consecutive multiscale feature learning-based image classification model. *Scientific Reports*, 2023.

[32] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 1986.

[33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[34] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017.

[36] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[37] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, 2020.

[38] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *International Conference on Computer Vision*, 2021.

[39] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Computer vision and pattern recognition*, 2022.

[40] Stuart P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 1982.

[41] Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 1998.

[42] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2007.

[43] Keinosuke Fukunaga and Larry D. Hostetler. The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. *IEEE Transactions on Information Theory*, 1975.

[44] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[45] Eric Xing, Michael Jordan, Stuart J Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 2002.

[46] Michele Ceccarelli and Antonio Maratea. Improving fuzzy clustering of biological data by metric learning with side information. *International Journal of Approximate Reasoning*, 2008.

[47] Nir Ailon, Anup Bhattacharya, and Ragesh Jaiswal. Approximate correlation clustering using same-cluster queries. In *Latin American Symposium on Theoretical Informatics*, 2018.

[48] Tuhinangshu Choudhury, Dhruti Shah, and Nikhil Karamchandani. Top-m clustering with a noisy oracle. In *National Conference on Communications*, 2019.

[49] Taewan Kim and Joydeep Ghosh. Relaxed oracles for semi-supervised clustering. *arXiv preprint arXiv:1711.07433*, 2017.

[50] Arya Mazumdar and Barna Saha. Clustering with noisy queries. In *Advances in Neural Information Processing Systems*, 2017.

[51] Fionn Murtagh and Pierre Legendre. Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion? *Journal of Classification*, 2014.

[52] Levent Ertöz, Michael Steinbach, and Vipin Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *SIAM International conference on data mining*, 2003.

[53] Wei Zhang, Xiaogang Wang, Deli Zhao, and Xiaoou Tang. Graph degree linkage: Agglomerative clustering on a directed graph. In *European Conference on Computer Vision*, 2012.

[54] Youness Aliyari Ghassabeh. A sufficient condition for the convergence of the mean shift algorithm with gaussian kernel. *Journal of Multivariate Analysis*, 2015.

[55] Feiping Nie, Zinan Zeng, Ivor W. Tsang, Dong Xu, and Changshui Zhang. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *Transactions on Neural Networks*, 2011.

[56] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Transactions on pattern analysis and machine intelligence*, 2000.

[57] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, 2002.

[58] Shi. Multiclass spectral clustering. In *International Conference on Computer Vision*, 2003.

[59] Leo Grady and Eric L Schwartz. Isoperimetric graph partitioning for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2006.

[60] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *The International Conference on Database Theory*, 2001.

[61] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, oct 2012.

[62] E. M. Wright and Richard Bellman. Adaptive Control Processes: A Guided Tour. *The Mathematical Gazette*, 1962.

[63] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, Maximilian Strobel, and Daniel Cremers. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*, 2018.

[64] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. Towards K-means-friendly spaces: Simultaneous deep learning and clustering. In *International Conference on Machine Learning*, 2017.

[65] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Computer vision and pattern recognition*, 2016.

[66] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, 2016.

[67] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization. In *International Conference on Computer Vision*, 2017.

[68] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep Adaptive Image Clustering. In *International Conference on Computer Vision*, 2017.

[69] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *International Conference on Machine Learning*, 2017.

[70] Chih Chung Hsu and Chia Wen Lin. CNN-Based joint clustering and representation learning with feature drift compensation for large-scale image data. *IEEE Transactions on Multimedia*, 2018.

[71] Vittal Premachandran and Alan L. Yuille. Unsupervised Learning Using Generative Adversarial Training and Clustering. *International Conference on Learning Representations*, 2017.

[72] Sohil Atul Shah and Vladlen Koltun. Robust continuous clustering. *Proceedings of the National Academy of Sciences of the United States of America*, 2017.

[73] Sohil Atul Shah and Vladlen Koltun. Deep continuous clustering. *arXiv preprint arXiv:1803.01449*, 2018.

[74] Aharon Bar-Hillel, Tomer Hertz, Noam Shental, Daphna Weinshall, and Greg Ridgeway. Learning a mahalanobis metric from equivalence constraints. *Journal of machine learning research*, 2005.

[75] Mikhail Bilenko, Sugato Basu, and Raymond J Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *International conference on Machine learning*, 2004.

[76] Hong Chang and Dit-Yan Yeung. Locally linear metric adaptation for semi-supervised clustering. In *International conference on Machine learning*, 2004.

[77] Noam Shental, Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall. Computing gaussian mixture models with em using equivalence constraints. *Advances in neural information processing systems*, 2003.

[78] Zhengdong Lu and Todd Leen. Semi-supervised learning with penalized probabilistic clustering. *Advances in neural information processing systems*, 17, 2004.

[79] Yen-Chang Hsu and Zsolt Kira. Neural network-based clustering using pairwise constraints. *arXiv preprint, arXiv:1511.06321*, 2015.

[80] Sharon Fogel, Hadar Averbuch-Elor, Daniel Cohen-Or, and Jacob Goldberger. Clustering-Driven Deep Embedding With Pairwise Constraints. *Computer Graphics and Applications*, 2019.

[81] Ankita Shukla, Gullal S Cheema, and Saket Anand. Semi-supervised clustering with neural networks. In *International Conference on Multimedia Big Data (BigMM)*, 2020.

[82] Arya Mazumdar and Barna Saha. Query complexity of clustering with side information. In *Advances in Neural Information Processing Systems*, 2017.

[83] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 2015.

[84] Claudio Michaelis, Ivan Ustyuzhaninov, Matthias Bethge, and Alexander S Ecker. One-shot instance segmentation. *arXiv preprint arXiv:1811.11507*, 2018.

[85] Meng-Ru Hsieh, Yen-Liang Lin, and Winston H Hsu. Drone-based object counting by spatially regularized regional proposal network. In *International Conference on Computer Vision*, 2017.

[86] Chaitanya Desai, Deva Ramanan, and Charless C Fowlkes. Discriminative models for multi-class object layout. *International journal of computer vision*, 2011.

[87] Dingkang Liang, Wei Xu, Yingying Zhu, and Yu Zhou. Focal inverse distance transform maps for crowd localization. *IEEE Transactions on Multimedia*, 2022.

[88] Hisham Cholakkal, Guolei Sun, Salman Khan, Fahad Shahbaz Khan, Ling Shao, and Luc Van Gool. Towards partial supervision for generic object counting in natural scenes. *Transactions on Pattern Analysis and Machifne Intelligence*, 2020.

[89] Chenfeng Xu, Dingkang Liang, Yongchao Xu, Song Bai, Wei Zhan, Xiang Bai, and Masayoshi Tomizuka. Autoscale: learning to scale for crowd counting. *International Journal of Computer Vision*, 2022.

[90] Shahira Abousamra, Minh Hoai, Dimitris Samaras, and Chao Chen. Localization in the crowd with topological constraints. In *AAAI Conference on Artificial Intelligence*, 2021.

[91] Yuting Liu, Miaojing Shi, Qijun Zhao, and Xiaofang Wang. Point in, box out: Beyond counting persons in crowds. In *Computer vision and pattern recognition*, 2019.

[92] Liang Liu, Hao Lu, Haipeng Xiong, Ke Xian, Zhiguo Cao, and Chunhua Shen. Counting objects by blockwise classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020.

[93] Antoni B Chan and Nuno Vasconcelos. Bayesian poisson regression for crowd counting. In *International conference on computer vision*, 2009.

[94] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. *Advances in neural information processing systems*, 2010.

[95] M. Marsden, K. McGuinness, S. Little, C. E. Keogh, and N. E. O'Connor. People, penguins and petri dishes: Adapting object counting models to new visual domains and object types without forgetting. In *Computer Vision and Pattern Recognition*, 2018.

[96] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Computer vision and pattern recognition*, 2016.

[97] Vishwanath A Sindagi, Rajeev Yasarla, and Vishal M Patel. Jhu-crowd++: Large-scale crowd counting dataset and a benchmark method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[98] Joseph Paul Cohen, Genevieve Boucher, Craig A Glastonbury, Henry Z Lo, and Yoshua Bengio. Count-ception: Counting by fully convolutional redundant counting. In *International Conference on Computer Vision workshops*, 2017.

[99] Philipp Kainz, Martin Urschler, Samuel Schulter, Paul Wohlhart, and Vincent Lepetit. You should use regression to detect cells. In *The Medical Image Computing and Computer Assisted Intervention Society*, 2015.

[100] Guangshuai Gao, Qingjie Liu, and Yunhong Wang. Counting from sky: A large-scale data set for remote sensing object counting and a benchmark method. *IEEE Transactions on Geoscience and Remote Sensing*, 2021.

[101] A.N. Marana, L. Da Fontoura Costa, R.A. Lotufo, and S.A. Velastin. Estimating crowd density with minkowski fractal dimension. In *International Conference on Acoustics, Speech, and Signal Processing*, 1999.

[102] A NSAV Marana, Sergio A Velastin, L da F Costa, and RA Lotufo. Automatic estimation of crowd density using texture. *Safety Science*, 1998.

[103] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *Computer vision and pattern recognition*, 2015.

[104] Negin Sokhandan, Pegah Kamousi, Alejandro Posada, Eniola Alese, and Negar Rostamzadeh. A few-shot sequential approach for object counting. *arXiv preprint arXiv:2007.01899*, 2020.

[105] Yinjie Lei, Yan Liu, Pingping Zhang, and Lingqiao Liu. Towards using count-level weak supervision for crowd counting. *Pattern Recognition*, 2021.

[106] Deepak Babu Sam, Neeraj N Sajjan, Himanshu Maurya, and R Venkatesh Babu. Almost unsupervised learning for dense crowd counting. In *AAAI Conference on Artificial Intelligence*, 2019.

[107] Matthias von Borstel, Melih Kandemir, Philip Schmidt, Madhavi K Rao, Kumar Rajamani, and Fred A Hamprecht. Gaussian process density counting from weak supervision. In *European conference on computer vision*, 2016.

[108] Yifan Yang, Guorong Li, Zhe Wu, Li Su, Qingming Huang, and Nicu Sebe. Weakly-supervised crowd counting learns from sorting rather than locations. In *European conference on computer vision*, 2020.

[109] Xialei Liu, Joost Van De Weijer, and Andrew D Bagdanov. Leveraging unlabeled data for crowd counting by learning to rank. In *Computer vision and pattern recognition*, 2018.

[110] Shuo-Diao Yang, Hung-Ting Su, Winston H Hsu, and Wen-Chin Chen. Class-agnostic few-shot object counting. In *Winter Conference on Applications of Computer Vision*, 2021.

[111] Erika Lu, Weidi Xie, and Andrew Zisserman. Class-agnostic counting. In *Asian Conference on Computer Vision*, 2019.

[112] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 2014.

[113] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. *Advances in neural information processing systems*, 2003.

[114] Carlos Arteta, Victor Lempitsky, J Alison Noble, and Andrew Zisserman. Detecting overlapping instances in microscopy images using extremal region trees. *Medical image analysis*, 2016.

[115] Antti Lehmussola, Pekka Ruusuvuori, Jyrki Selinummi, Heikki Huttunen, and Olli Yli-Harja. Computational framework for simulating fluorescence microscope images with cell populations. *IEEE transactions on medical imaging*, 2007.

[116] Viresh Ranjan and Minh Hoai. Exemplar free class agnostic counting. *arXiv preprint arXiv:2205.14212*, 2022.

[117] Hui Lin, Xiaopeng Hong, and Yabin Wang. Object counting: You only need to look at one. *arXiv preprint arXiv:2112.05993*, 2021.

[118] Ning Liu, Yongchao Long, Changqing Zou, Qun Niu, Li Pan, and Hefeng Wu. Adcrowdnet: An attention-injective deformable convolutional network for crowd understanding. In *Computer vision and pattern recognition*, 2019.

[119] Xiaoheng Jiang, Li Zhang, Mingliang Xu, Tianzhu Zhang, Pei Lv, Bing Zhou, Xin Yang, and Yanwei Pang. Attention scaling for crowd counting. In *Computer vision and pattern recognition*, 2020.

[120] Anran Zhang, Lei Yue, Jiayi Shen, Fan Zhu, Xiantong Zhen, Xianbin Cao, and Ling Shao. Attentional neural fields for crowd counting. In *International conference on computer vision*, 2019.

[121] Dingkang Liang, Xiwu Chen, Wei Xu, Yu Zhou, and Xiang Bai. Transcrowd: Weakly-supervised crowd counting with transformer. *arXiv preprint arXiv:2104.09116*, 2021.

[122] Siddharth Singh Savner and Vivek Kanhangad. Crowdformer: Weakly-supervised crowd counting with improved generalizability. *arXiv preprint arXiv:2203.03768*, 2022.

[123] Fusen Wang, Kai Liu, Fei Long, Nong Sang, Xiaofeng Xia, and Jun Sang. Joint cnn and transformer network via weakly supervised learning for efficient crowd counting. *arXiv preprint arXiv:2203.06388*, 2022.

[124] Phuc Thinh Do. Attention in crowd counting using the transformer and density map to improve counting result. In *NAFOSTED Conference on Information and Computer Science*, 2021.

[125] Michael Hobley and Victor Prisacariu. Learning to count anything: Reference-less class-agnostic counting with weak supervision. *arXiv preprint arXiv:2205.10203*, 2022.

[126] Manolis Savva, Angel X Chang, and Pat Hanrahan. Semantically-enriched 3d models for common-sense knowledge. In *Computer vision and pattern recognition*, 2015.

[127] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[128] Kai Han, Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Andrea Vedaldi, and Andrew Zisserman. Automatically discovering and learning new visual categories with ranking statistics. *arXiv preprint arXiv:2002.05714*, 2020.

[129] Kai Han, Andrea Vedaldi, and Andrew Zisserman. Learning to discover novel visual categories via deep transfer clustering. In *International Conference on Computer Vision*, 2019.

[130] Hongfu Liu and Yun Fu. Clustering with Partition Level Side Information. *International Conference on Data Mining*, 2015.

[131] Fengfu Li, Hong Qiao, and Bo Zhang. Discriminatively boosted image clustering with fully convolutional auto-encoders. *Pattern Recognition*, 2018.

[132] Yizong Cheng. Mean Shift, Mode Seeking, and Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[133] Ronald R. Coifman and Matthew J. Hirn. Bi-stochastic kernels via asymmetric affinity functions. *Applied and Computational Harmonic Analysis*, 2013.

[134] David J Pearce. An improved algorithm for finding the strongly connected components of a directed graph. *Victoria University, Wellington, NZ, Tech. Rep*, 2005.

[135] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 1958.

[136] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[137] S Nene, S Nayar, and H Murase. Columbia Object Image Library (COIL-20). *Technical Report*, 1996.

[138] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *Computer vision and pattern recognition*, 2011.

[139] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 2004.

[140] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[141] Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing*, 2010.

[142] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - A knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 2003.

[143] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 2010.

[144] David F. Crouse. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 2016.

[145] Harold W. Kuhn. The Hungarian method for the assignment problem. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, 2010.

[146] Jingyi Xu, Hieu Le, Vu Nguyen, Viresh Ranjan, and Dimitris Samaras. Zero-shot object counting. In *Computer vision and pattern recognition*, 2023.

[147] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Computer vision and pattern recognition*, 2020.

[148] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Computer vision and pattern recognition*, 2016.

[149] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, 2021.

[150] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *International Conference on Computer Vision*, 2019.

[151] Qi Fan, Wei Zhuo, Chi-Keung Tang, and Yu-Wing Tai. Few-shot object detection with attention-rpn and multi-relation detector. In *Computer vision and pattern recognition*, 2020.

[152] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 2017.

[153] Dingkang Liang, Wei Xu, and Xiang Bai. An end-to-end transformer model for crowd localization. *arXiv preprint arXiv:2202.13065*, 2022.

[154] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Computer vision and pattern recognition*, 2022.

[155] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer vision and pattern recognition*, 2009.

[156] Roni Paiss, Ariel Ephrat, Omer Tov, Shiran Zada, Inbar Mosseri, Michal Irani, and Tali Dekel. Teaching clip to count to ten. *arXiv preprint arXiv:2302.12066*, 2023.

[157] OpenAI. Gpt-4 technical report, 2023.

[158] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[159] Zhiheng Ma, Xiaopeng Hong, and Qinnan Shangguan. Can sam count anything? an empirical study on sam counting. *arXiv preprint arXiv:2304.10817*, 2023.

[160] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 2021.

[161] Colin Troisemaine, Vincent Lemaire, Stéphane Gosselin, Alexandre Reiffers-Masson, Joachim Flocon-Cholet, and Sandrine Vaton. Novel class discovery: an introduction and key concepts. *arXiv preprint arXiv:2302.12028*, 2023.

[162] Michael A Hobley and Victor A Prisacariu. Dms: Differentiable mean shift for dataset agnostic task specific clustering using side information. *arXiv preprint arXiv:2305.18492*, 2023.

[163] Thanh Nguyen, Chau Pham, Khoi Nguyen, and Minh Hoai. Few-shot object counting and detection. In *European conference on computer vision*, 2022.

[164] Yingzhou Lu, Huazheng Wang, and Wenqi Wei. Machine learning for synthetic data generation: a review. *arXiv preprint arXiv:2302.04062*, 2023.

[165] Michael A Hobley and Victor A Prisacariu. Dms: Differentiable mean shift for dataset agnostic task specific clustering using side information. *arXiv preprint arXiv:2305.18492*, 2023.

[166] Yang Xiao, Vincent Lepetit, and Renaud Marlet. Few-shot object detection and viewpoint estimation for objects in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.