StampNet
Unsupervised Multi-Class Object Discovery

Visser, J.W.J.H.

Award date:
2020

Technische Universiteit Eindhoven University of Technology

Department of Mathematics and Computer Science
Data Mining Group

StampNet: Unsupervised Multi-Class Object Discovery

Master Thesis

Joost Visser

Supervisors
dr. Vlado Menkovski
dr. Alessandro Corbetta

Version 1.1

Eindhoven, February 2020

Abstract

Humans can group and locate different types of objects within an image, even when they do not recognise the object itself. Computer programs, however, typically need a large training set of images containing objects whose locations and categories are annotated to perform this task. These annotations often require major human effort and thus come with significant costs. In unsupervised object discovery, we want to discover and group objects without annotations, similar to what humans are capable of. Previous work in unsupervised object discovery mainly focuses on discovering and localising a single object in each image, but we aim to simultaneously find multiple objects in each image and group these objects into categories.

In this thesis, we propose StampNet, a novel autoencoding neural network that localises objects over a simple background in images and categorises them simultaneously. StampNet consists of a discrete latent space that is used to categorise objects and to determine the location of the objects. The object categories are formed during the training, resulting in the discovery of a fixed set of objects. We present a set of experiments that demonstrate that StampNet can localise and cluster multiple overlapping shapes with varying complexity including the digits from the MNIST dataset. We also present an application of StampNet in the localisation of pedestrians in overhead depth-maps. The code of StampNet is available online at `https://github.com/crowdflowTUe/stampnet`.

Preface

My thesis has been a very long journey, longer than I want to admit. I want to thank everyone that has helped me to make it this far and finish the research project that has way too many hours invested in it.

First and foremost, I want to thank both my supervisor dr. Vlado Menkovski and dr. Alessandro Corbetta for all the help and insights for this project. Your valuable feedback has helped me steer the project in the right direction and produce results. Many of the ideas you suggested are incorporated in this thesis and without your help, I would never be able to finish writing the paper. Thank you.

Moreover, I want to thank my friends that helped along the road of completion. You supported me mentally by being there. I want to thank Daniël in particular for help with proofreading some of the thesis texts and helping me with my presentation, all the way back in the days of the seminar.

Finally, my thanks will not be complete without the mention of my family. You have supported me all the way, even though it took just slightly longer than expected. I especially want to thank my brother Perry, for always calling me right when I was working on yet another long night of experimenting.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Finding and discovering objects in a digital image is an important task in image analysis. Consider a self-driving car driving around the road. When a person dashes onto the road, the car should be able to detect the person to break in time. Therefore, we want to detect *what* an object is, and *where* this object is located in an image. In image analysis, we call this *classification* and *localisation* respectively.

For humans, object classification and localisation is an easy task, as we do this in our daily lives by viewing the world with our eyes. Even when we see a digital image, we can recognise and localise objects of the image using our knowledge. When we encounter an object which we have never seen before, we can still figure out where this object is and detect other similar objects, using features like shape, contrast and colour.

For a computer program, however, this task becomes much more complex. It does not have any experience with viewing objects in the world as a person, nor can it differentiate between different objects. By giving a supervised learning algorithm large amounts of data with annotations of where each object is and what type of object it is, we can train models that can perform object detection and localisation [33]. The size of these datasets can go into the millions of images [9], requiring millions of annotations which can only be done by humans. Consequently, these annotations have a significant cost.

In contrast, *object discovery* deals with localisation when there are no annotations. In this setting, we want to find recurring patterns that define an object, localise these objects and ideally group them in clusters. Humans are capable of this, as we can discern and locate different objects which we never have seen before and we can group objects that look similar. The localisation allows us to discover notable objects while the concurrent grouping of objects enables efficient learning; in reality, humans do not require a label for an object like a bottle for each different possible rotation, setting and background.

Previous work on unsupervised localisation typically addresses one object in multiple images, which is referred to as object co-localisation [21, 41]. However, unsupervised localisation of multiple classes of objects remains a significant challenge. Unlike object co-localisation, where there is only a single object of interest in the image and thus only focuses on localisation, the model needs to simultaneously discover and form categories of the objects (because of the multi-class aspect) and learn to perform localisation.

Therefore, to remove the need of annotations in images with multiple classes, these methods cannot be used, as the multi-class aspect requires some form of clustering to be required to differentiate between classes.

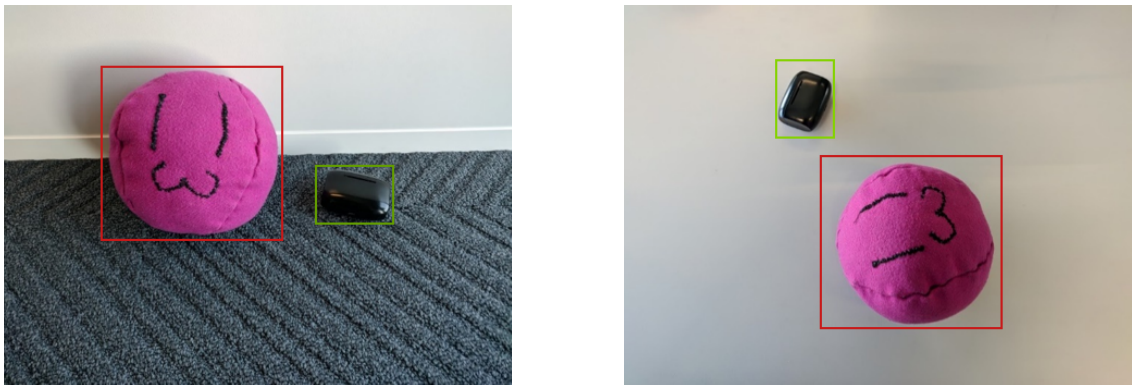


Figure 1.1: Humans can discover, localise and differentiate these different objects (bounded by the red and green bounding box) without ever having seen them before, i.e. without the use of any labels. We aim to do the same in unsupervised multi-class object discovery.

1.2 Problem description

Humans can analyse multiple objects in images without the need for any labels or prior knowledge about these objects. They can then interpret these objects: where these objects are, the size of the object, the type of the objects and more. Therefore, by creating a model that can perform this kind of unsupervised object discovery, we could interpret the model to obtain valuable information from the image.

However, as mentioned in Section 1.1, this is a problem for the latest unsupervised models due to the existence of multiple objects of different types. Besides having to differentiate between categories, the positioning and size of the objects are not predetermined, adding further to the complexity of the problem. Moreover, multiple objects can overlap, which further increases the difficulty of finding, localising and clustering these objects.

The goal of this thesis is to find out whether it is possible to create a model that can perform such unsupervised multi-class discovery, to obtain information from the image without the need for annotations. In other words, the problem statement that we tackle in this thesis can be summarised as follows:

> ***Problem statement:*** *Can we discover and localise multiple objects in images in an unsupervised manner, while simultaneously being able to differentiate between different classes of objects?*

1.3 Project goal

To tackle our problem statement, we first need to develop a method that is capable of performing unsupervised multi-class object discovery and that can differentiate between different classes. Developing this new technique is the first and main goal of the thesis.

However, we require a set of benchmarks to test how well the method performs. To the best of our knowledge, the problem of multi-class object discovery with multiple objects per image has never been tackled before. Thus, we do not know any standard benchmark or test-data. Therefore, the second goal consists of finding new datasets to test our method on, either by creating a new dataset or by augmenting existing datasets. Ideally, we can compare our methods to other methods in the literature, for instance by considering the simpler case when having a single class.

Finally, we want to show a real-world application of the developed method. We have a dataset available of overhead depth images containing pedestrians and the goal is to localise the pedestrians in an unsupervised manner. This problem contains two different classes, pedestrians and walls,

therefore requiring the need for multi-class object discovery. We want to analyse the performance of the model in terms of discovery and localisation of pedestrians.
Summarised, we focus on the following three goals:

> ***Goal 1:*** *Develop a method that is capable of performing unsupervised multi-class object discovery.*
>
> ***Goal 2:*** *Create new datasets for unsupervised multi-class object discovery, to test and analyse our method.*
>
> ***Goal 3:*** *Evaluate the performance of the method in the context of pedestrian localisation.*

1.4 Research scope

As mentioned before, previous work on unsupervised localisation typically address a single object on multiple images. The problem of discovering multiple objects of multiple classes in an unsupervised manner is vastly more complex: not only do we need to find multiple objects in a single image, which significantly increases the number of potential solutions, we also need to have some form of clustering to differentiate between objects. Another problem is that multiple objects can overlap in each image, which we should take into account when developing the model.

Therefore, our problem is several times more challenging than the problems solved in existing literature. For that reason, we add some constraints to the problem to reduce the scope of the thesis, such that the problem becomes less complex while remaining useful. We show the latter by providing a practical application of the problem.

The first set of constraints regards the clustering of objects. To separate classes in an unsupervised manner, there has to be something that determines what makes a class different from another class, a certain measure we can use. We assume that we can differentiate between different classes by means of the mean-squared error (MSE) per pixel, or Euclidean distance, between two different objects. The MSE measures how similar two objects are in terms of overlap and pixel value; if two objects are almost the same, they overlap a lot and their pixel values are similar, resulting in a small MSE and thus they should be in the same class. To fairly calculate this measure between different objects, we only consider objects of fixed sizes. Note that such measure is also used in e.g. k-means clustering.

In practice, two objects can be similar and should belong in the same class but have almost no overlap. Consider two dogs that are exactly the same, but one dog is rotated a quarter turn. They should belong to the same class *dog*, but the resulting MSE measure might be high because they do not look and stand exactly the same, thus they will end up in different clusters.

We can solve this problem by either requiring labels, which are not allowed in unsupervised learning, or by using a certain measure of similarity between these images that is not dependent on the error per pixel value. While we could develop new measures or use pretraining to attain a better sense of similarity, this deviates too much from the scope of our research. Thus, we only focus on using MSE as a measure to separate classes.

Another constraint is that we only consider neural network models for this thesis. While some of the datasets we are tackling might be solved by simple feature engineering or creating statistical models, most of the state-of-the-art research use neural networks in one way or another due to the effectiveness of convolutional layers. Using feature engineering or creating statistical models would also undercut the unsupervised theme of this thesis, as a person has to spend effort in creating these features. The goal of the simpler datasets is purely to analyse the capability of the model in a new setting.

1.5 Contributions

The main contributions of this thesis are:

1. We developed a novel neural network model, StampNet, that is capable of performing multi-class object discovery. In particular:
   (a) We showed a novel way of using the kernel of a convolutional layer by storing complete objects instead of shapes of an object.
   (b) We showed a new application for the Gumbel-Softmax [18], a trainable categorical sample approximator in neural networks, as a coordinate predictor in an autoencoder.
   (c) We established a new way of clustering with neural networks.
2. We created new datasets for testing this new class of problems. The simple shape dataset consists of simple shapes to test the basic functionality of the model and the T-MNIST and CT-MNIST datasets are more difficult augmentations of the MNIST dataset, a dataset containing images of handwritten digits [22].
3. We established an application of our new model in pedestrian tracking in overhead depth-maps.

Furthermore, a paper version of this research has been published in the IEEE International Conference on Image Processing (ICIP) of 2019 [44]. We include the published paper version in Appendix C.

1.6 Method of research

To satisfy the project goals and solve the problem statement, we take the following steps:

Explain relevant theory for understanding the method
We first introduce some specific concepts of neural networks in Chapter 2, such as an autoencoder. These concepts are required to understand the model we use. Moreover, we introduce the notation used throughout the thesis.

Perform a literature study on relevant topics
Additionally, in Chapter 2, we briefly highlight existing literature closely related to unsupervised object discovery, such as multi-class co-localisation. We investigate what is possible with current techniques developed by the research community.

Establish the model used to solve the problem
In Chapter 3 we explain the model developed for this problem. The architecture of the model will be described and justified step-by-step.

Test the performance of the model on various datasets
We test the model on various datasets in Chapter 4. The chapter explains how the model is tested and which datasets are used. There is also an application of our model in pedestrian tracking in overhead depth-maps at the end of the chapter.

Discuss the results and conclude
Finally, we conclude the results in Chapter 5. We examine the test results and evaluate to what extent the goals of our project are achieved. Moreover, we discuss the problem statement and recommend future work for this project.

Chapter 2

Background section

In this chapter, we elaborate on the preliminary knowledge of our model as well as discussing works related to the field of object discovery and object detection. We start by explaining the theoretical background knowledge in Section 2.1 and continue by highlighting the related works in Section 2.2.

2.1 Theoretical background

2.1.1 Autoencoder

An *autoencoder* is a type of neural network that can be used in an unsupervised setting. The network avoids labels by setting the input as its label. In other words, the network has to reconstruct the input image, given the input image.

The autoencoder reproduces the input image by automatically extracting useful features from the image. These extracted features are then used for the reconstruction of the image, without the need for any labels. The automatic feature extraction of images and unsupervised properties of an autoencoder make it an ideal method to perform multi-class unsupervised object discovery. In multi-class unsupervised object discovery, we are interested in the location of each object as well as the class it belongs to. If we design the network such that it can only reproduce the image by predicting the locations and the class each object belongs to, e.g. by placing the predicted objects on the predicted locations, we can extract both the location and the class of each object automatically.

One way for the network to reconstruct the input image is to output the input image, the identity function. However, the network does not learn anything other than the identity function. We avoid this by constraining the network in a certain manner. For example, we could limit the number of neurons in one of the hidden layers, which we illustrate in Figure 2.1. Because the number of neurons in the hidden layer is limited, the network must learn an encoding to reproduce the input image. Consequently, the network architecture contains an *encoder* that encodes the input image to a certain code, while the *decoder* reads this code to reproduce the input image.

For images, the encoder generally consists of several convolutional layers and pooling layers. The convolutional operations allow the network to extract spatial features and the pooling layers reduce the size of the tensor.

The decoder generally contains deconvolutional layers and upsampling layers. The upsampling layers increase the size of the image by duplicating the pixels, whereas the deconvolutional layers perform a transposed convolutional operation. A transposed convolutional operation, also known as fractionally strided convolutional operation [25], is similar to standard convolutional operation. However, instead of only convolving over the inside of the image, we convolve over the outside of the image as well (with zero-padding). We illustrate this in Figure 2.2.

In summary, the autoencoder optimises for the reconstruction of the image. Thus, to minimise the loss, it extracts useful spatial features using the encoder. The decoder uses these features

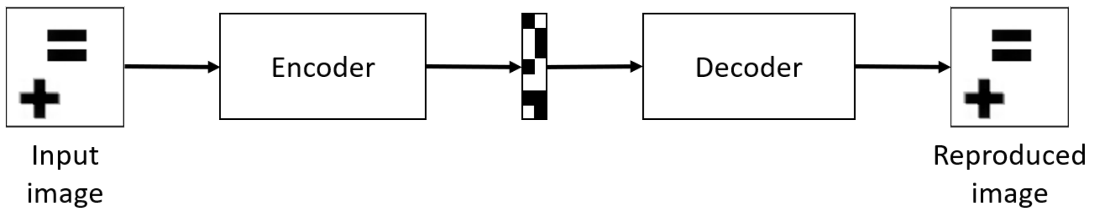


Figure 2.1: An example of an unsupervised neural network architecture called an autoencoder. Since the input image is used as a label, there is no need for labels. The network first encodes the input image into a compressed representation, after which it decodes this code into an output image.

to reproduce the image as accurately as possible. This results in a network that automatically extracts features of the image and creates an encoding, hence the name autoencoder. In our case, we design the decoder such that it has to predict the location and class of each object to reconstruct the image. This way, we can obtain the information without the need for any labels.

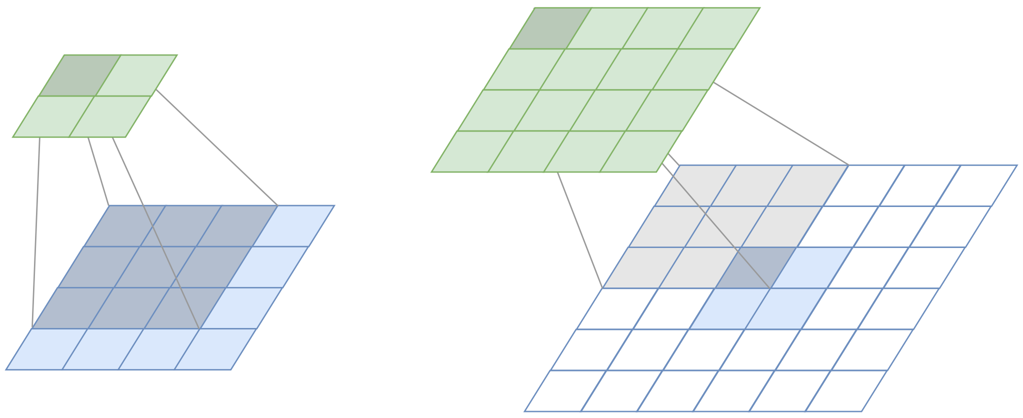


Figure 2.2: A convolutional operation (left) and a transposed convolutional operation (right). The input is in blue and the output is in green. On the left side, we perform a convolutional operation on a 4×4 image, resulting in an output image of size 2×2. Note that in this case, the sliding window happens inside the image. In contrast, in a transposed convolutional operation we also slide over the outside (using zero-padding), resulting in an output image of 4×4 while the input image is of size 2×2. In both cases the size of the kernel is 3×3.

Finally, we require a loss function to enable training, as neural network needs a function to optimise. We use the mean of squared errors (MSE) per pixel for the loss function:

$$\mathcal{L}(X) = \frac{1}{n}\sum_{i=1}^{n}(f(X^{(i)}) - X^{(i)})^2$$

Where $X^{(i)}$ is the i-th input image and f is the autoencoder, so $f(X^{(i)})$ is the reconstruction of the neural network for this image.

2.1.2 Gumbel-Softmax

As mentioned in the previous chapter, we want to predict the location and class of each object in the image during the decoding step. The location of an object is a coordinate (x, y) and the class selector is an integer s. We model these variables as discrete random variables for three reasons: first of all, the domain of both the pixel coordinate and the class selector are discrete, so we can

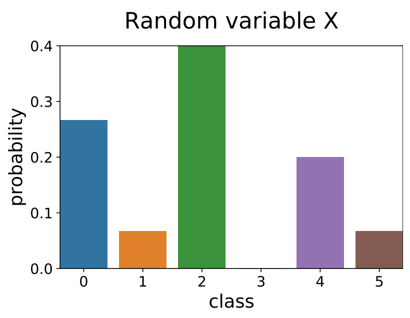


Figure 2.3: A categorical distribution of random variable X. Each category or class has a probability of π_i.

use discrete random variables. Secondly, as these variables are modelled as random variables, we can use probability theory. Finally, if we interpret these variables as probability distributions, we can obtain a one-hot sampling while still having non-zero probabilities for the classes that are not sampled, which is useful for training and backpropagation.

We model these discrete random variables as a categorical distribution. The categorical distribution is a discrete probability distribution that defines a random variable which can take one out of k possible categories. For example, consider a random variable X with $k = 6$ categories, such as in Figure 2.3. If we sample from this random variable, we sample the category according to the probability distribution. In our example: $\Pr[X = 2] = \pi_2 = 0.4$.

However, there is a small problem if we want to use categorical distributions in a neural network: we use backpropagation to compute the gradients, but it is not possible to backpropagate through sampling due to the randomness of the sampling. The Gumbel-Softmax [18], also known as the concrete distribution [26], circumvents this problem by re-expressing a sampling such that the gradients can flow without encountering any stochastic nodes; a *reparameterisation trick* for the categorical distribution.

In other words, the Gumbel-Softmax is a categorical distribution that is trainable in neural networks. To be more precise, it is a distribution that smoothly deforms into a categorical distribution, since a categorical distribution requires an argmax and the argmax function is not differentiable. The formula for the Gumbel-Softmax is as follows:

$$\frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{N} \exp((\log(\pi_j) + g_j)/\tau)}$$

Where $g_i \sim \text{Gumbel}(0, 1)$ is gumbel noise sampled from the Gumbel distribution and τ is a temperature variable that allows us to control how closely samples approximate those from the categorical distribution. As the temperature τ goes to 0, the softmax becomes an argmax and

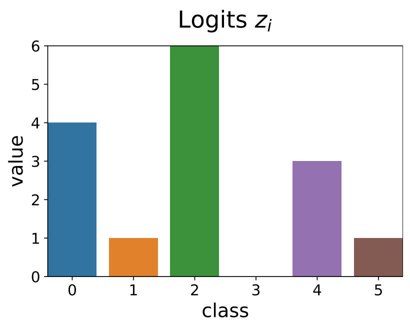


Figure 2.4: Logits z_i which we use as input to illustrate the effects of different temperatures for the Gumbel-Softmax distribution in Figure 2.5.

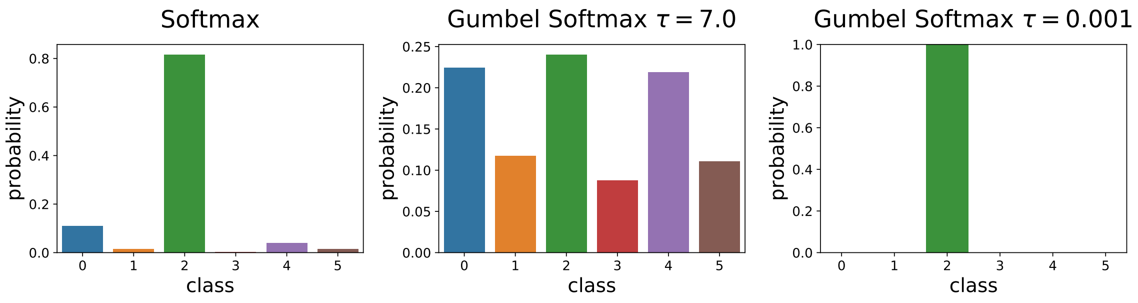


Figure 2.5: The output of three different softmax functions with the logits z_i as input (Figure 2.4). A high-temperature Gumbel-Softmax (middle) has a more uniform probability distribution and allows the gradients to be backpropagated, whereas a low-temperature Gumbel-Softmax (right) results in a one-hot output, similar to a sampling of the categorical distribution. We illustrate the regular softmax function (left) as a comparison. The output of the softmax function is deterministic, but as the Gumbel-Softmax contains noise, its output is not deterministic.

it approaches the categorical distribution. With a high temperature, the Gumbel-Softmax acts similar to a uniform distribution.

During training, we use an annealing schedule for τ. First, we let $\tau > 0$ to allow gradients past the Gumbel-Softmax. Then, we gradually anneal the τ close to 0 to make it more a categorical distribution.

For better intuition, we illustrate the difference in the softmax function, high-temperature and low-temperature Gumbel-Softmax function. As input, we use logits that look similar to the distribution of Figure 2.3. We show the values of the logits in Figure 2.4. In Figure 2.5, we show the output of the regular softmax, a high-temperature Gumbel-Softmax ($\tau = 7.0$) and a low-temperature Gumbel-Softmax ($\tau = 0.001$) function over these logits respectively. Since the Gumbel-Softmax uses noise to determine its output, these outputs are not deterministic (another calculation would result in a different graph). We observe that the higher temperature results in a more uniform output, allowing the gradients to flow through. The lower temperature only outputs 1 for a single class (mainly 2, sometimes 0), which is a better representation of a sampling of the categorical distribution.

2.2 Related Works

There have been many studies related to unsupervised object discovery in computer vision and machine learning. For a comprehensive overview of the current literature, we start by explaining the related works of the supervised variant of our problem, called object detection. Next, we continue to discuss studies with less supervision until we arrive at the latest works of unsupervised object discovery.

Each object in supervised object detection is labelled with its class and its bounding box, containing the coordinates and bounded size of the object. A standard solution where we predict these labels using a convolutional neural network (CNN) for each does not work as the number of objects is not fixed per image. In general, there are two solutions to this problem: either by generating proposal regions and classify the potential object [14, 13, 34, 24, 12, 23] or by subdividing the image into parts, such as a grid, and classify whether the centre of an object is within a part [31, 32, 33]. Our method can best be compared to an automated region proposal method, as the model proposes the regions using the predicted coordinates, although the model does not explicitly use region proposals.

In R-CNN [14], the authors generate a set of 2000 proposal regions, also called regions of interest (ROIs), where there could potentially be an object. These region proposals are generated using a selective search algorithm that recursively combines similar candidate regions until there are 2000 regions left. These regions are fed into a classifier that predicts the bounding boxes and whether there is an object in the ROI. An improved version called Faster R-CNN [34] lets the network learn the region proposals instead and single-shot detectors (SSD) [24] combine the region proposals and label prediction into a single step.

The authors of YOLO [31] take a different approach by dividing the image into a grid. In each grid, they generate a number of bounding boxes with their respective class probability and select an object if their class probability is above a certain threshold. Later versions [32, 33] contain various improvements such as predicting the offset with respect to anchor boxes, which are prior bounding boxes obtained by a k-means clustering of the dataset, instead of predicting the size of the bounding box themselves.

A step down in terms of supervision is weakly supervised object localisation (WSL), where the images are annotated with binary labels indicating the presence or absence of an object. Most approaches are based on multiple instance learning (MIL) [10, 3, 20], where positive images contain at least one true bounding box for a target class, whereas negative images only contain false boxes.

WSL methods generally consist of two steps: first, they generate regions proposals or ROIs (also known as candidate windows in WSL), after which they improve these in an iterative manner. There are a number of strategies to initialise the ROIs. A simple strategy is to take an initial region that almost covers the entire image [30, 36], making it very likely that the true bounding box is in this region. Other methods, however, use selective search [42] to initialise the ROIs [6, 20].

These ROIs can be improved in a variety of ways. For example, some works improve these ROIs by using pairwise similarities between regions, e.g. in the k-NN clustering-based link analysis approach [21]. Other works improve the ROIs by alternating between selecting the region with the highest objectness measure and training a classifier for the positive and negative samples [6]. The objectness measure quantifies how likely it is for a region to contain an object of any class [1].

Co-localisation [41, 19] and co-segmentation [43, 35] are problems with even less supervision, where the only assumption is made that an object of a single class is in the majority of the images. Unlike in WSL, these negative images are not labelled. Nevertheless, similar to techniques used in WSL, Tang et al. [41] first generate ROIs using the objectness measure. After that, they optimise for a similarity measure to select the best candidate box in each image. At the same time, they output whether there is an object in the image at all.

When we remove the assumption that only an object of a single class is in the dataset, we obtain the multi-class variants of the problem: multi-class co-localisation [21, 5, 29] and multi-class co-segmentation [45]. Multi-class co-localisation is the same as unsupervised object discovery, as no labelling or assumption is made of the dataset.

In most cases, these methods first generate a set of candidate boxes and iteratively optimise these. Cho et al. [5] generate a set of candidate boxes using an off-the-shelf ROI proposal system [27], then alternately match these ROIs with respect to spatial and appearance consistency and select new ROIs that stand out the most using standout scores. Kim et al. [21] builds a similarity network of the ROIs and finds the best ranked ROIs using a link analysis with PageRank.

Murasaki et al. [29] take a different approach by using a pre-trained neural network. The authors extract deep features and cluster these. However, the use of a pretrained network does make the method more supervised, as such a network is already able to classify the objects it wants to localise.

In terms of clustering, both the work of Murasaki et al. [29] and Kim et al. [21] are able to cluster the localised objects. However, to the best of our knowledge, no work has yet fully tackled the problem of unsupervised multi-class object discovery with multiple objects in each image. While the method of Kim et al. [21] occasionally selects and outputs more than one promising ROI in each image, it does not do so consistently and does not output two ROIs that overlap too much (as these are merged).

Chapter 3

StampNet model

3.1 Problem formulation

Before we introduce the architecture of the model in detail, we first introduce the problem more formally. We consider a dataset of n images. Each image $X^{(i)}$ in the dataset has a size of $\phi_x \times \phi_y$ and contains a fixed number of m objects.

Each of the m objects are bounded by a bounding box B of fixed size $\psi_x \times \psi_y$. We index each object with j, so the j-th object of image i has a bounding box $B^{(i,j)} = (x_t, y_t, \psi_x, \psi_y)$. We illustrate this notation in Figure 3.1.

As the size of the bounding boxes ψ_x and ψ_y are fixed, we only need to predict the (x_t, y_t) coordinates of the object (and calculate corresponding measures using the known sizes of the bounding box). We use the upper-left coordinate of the bounding box for this. Then, the goal in terms of localisation is to predict the (x_t, y_t) coordinates of each object m.

Lastly, there are a total of M different classes of objects, which are indexed by l, so $1 \leq l \leq M$. In terms of clustering, we want to cluster all objects such that each cluster s ideally only contains objects of a single class l.

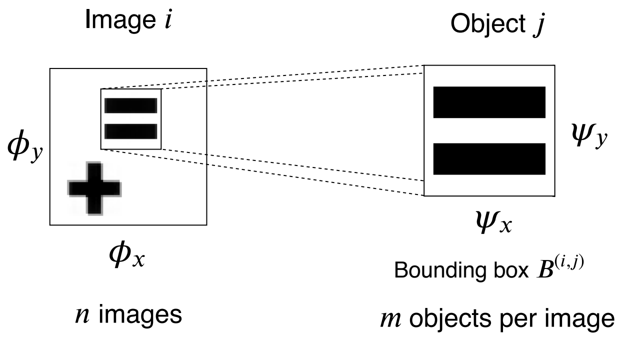


Figure 3.1: An illustration of the notation used throughout the thesis. Each image i of size (ϕ_x, ϕ_y) has m objects j. These objects j are bounded by their bounding boxes $B^{(i,j)}$ with size (ψ_x, ψ_y).

3.2 Architecture overview

StampNet is an autoencoder (Section 2.1.1) that simultaneously localises and clusters by placing so-called 'stamps' on the network, as required by our unsupervised object localisation task. We illustrate an overview of the architecture in Figure 3.2.

The network consists of an *encoder* that encodes the input image, and a *decoder* that reconstructs the input image as good as possible. The encoder consists of several sequential convolution layers and pooling layers to extract useful features. The decoder works in two steps: (1) the network predicts the coordinates of the objects in the *selection-and-localisation layer* (SL-layer), described in Section 3.4, and (2) the network places stamps on these predicted locations in the *stamp layer* (Section 3.3).

These stamps in the stamp layer are automatically learned by backpropagation. The network can only use a limited number of stamps, so, to reduce the loss, the network must stamp the same or similar objects with the same stamp. As multiple objects get stamped by the same stamp, this stamp represents a cluster of objects, resulting in an automatic clustering of the objects of the input image.

For every new input image, we obtain: (1) the coordinates of each object and (2) which stamp should be placed on these coordinates. Thus, we get localisation and clustering of each object; unsupervised multi-class object localisation.

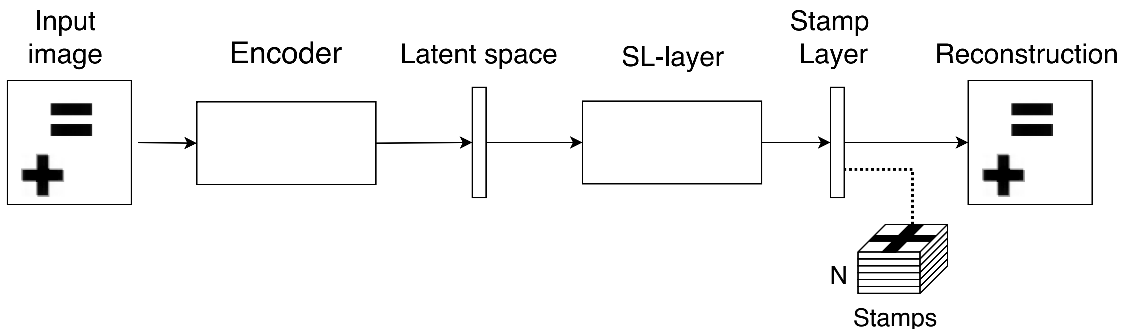


Figure 3.2: An overview of the architecture of StampNet. StampNet is an autoencoder that consists of an encoder and a decoder. The encoder encodes the input image to a compressed representation using several convolutional layers. The decoder consists of the SL-layer and the stamp layer, which create the reconstruction of the input image. The SL-layer localises each object and selects the stamp of the stamp layer, which are placed on an empty canvas to reproduce the input image as close as possible.

3.3 Stamp layer

3.3.1 Stamping in 2D

When discovering and localising objects in images, we need to be able to learn what type of objects there are in the input images. As we are looking for similar patterns or objects repeated on all images, we need a way of learning these prototypical objects.

A convolutional layer convolves the kernel of the layer over the input image. The kernel typically contains a shape; the output of the convolutional layer is a heatmap of where this shape occurs. This also means that convolutional layers can extract shapes that occur in the image automatically.

If we consider a normal convolutional autoencoder with convolutional layers in the encoder and deconvolutional layers in the decoder, we can extract these shapes without the need for any labels. However, if we constrain the decoder in such a way that the deconvolutional layer is only able to store the complete object, we would be able to extract complete prototypical objects automatically.

The stamp layer does this by performing a transposed convolutional operation (Section 2.1.1) over a two-dimensional one-hot tensor. When a kernel gets convolved over a one-hot tensor, the kernel gets rotated and placed on the location where the one-hot tensor equals one, as we illustrate in Figure 3.3. It is similar to a stamp that gets stamped on a particular location. Hence, we call this layer the *stamp layer* and the kernel the *stamps*.

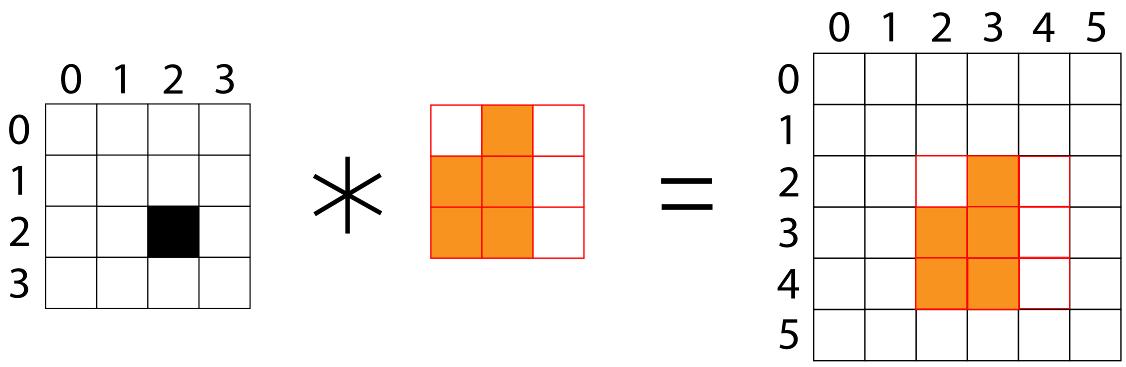


Figure 3.3: A transposed convolutional operation over a one-hot tensor. The kernel ω (middle) contains a shape and the one-hot tensor I_{xy} (left) indicates the coordinate where the shape will be placed, in this case $(2, 2)$. Result: the upper-left corner of the shape is placed at the same coordinate $(2, 2)$ in the output image.

We show that this holds formally. Consider a 2D one-hot tensor as input I_{xy} where $I_{x^*y^*} = 1$ and the rest is 0. Let ω_{xy} be a kernel of size $\psi_x \times \psi_y$. that contains our object (where its size $\psi_x \times \psi_y$ is defined as noted in Section 3.1). The formula for our output image O_{xy} is:

$$O_{xy} = I * \omega = \sum_{u,t} I_{vt}\omega_{x-u,y-v}$$

We know that $I_{uv} = 0$ for all values of u and v except when $u = x^*$ and $v = y^*$, thus:

$$\begin{aligned} O_{xy} &= \omega_{x-x^*,y-y^*} \\ O_{x+x^*,y+y^*} &= \omega_{xy} \end{aligned}$$

Therefore, we get kernel $\omega_{x,y}$ on the output image at coordinate (x^*, y^*). More precisely, the kernel will be placed between coordinates (x^*, y^*) and $(x^* + \psi_x - 1, y^* + \psi_y - 1)$ as the kernel has a size of (ψ_x, ψ_y).

When $0, 0 \leq x^*, y^* \leq \phi_x - \psi_y + 1, \phi_y - \psi_y + 1$, we get an output tensor O_{xy} of size $\phi_x \times \phi_y$ (as the size of the kernel gets added to the size of the position tensor). Thus, our position tensor I_{xy} should have a size of $(\phi_x - \psi_y + 1) \times (\phi_y - \psi_y + 1)$.

As a side note, many neural network libraries such as Keras or Tensorflow implement the cross-correlation operation instead of the convolution operation. In a convolutional operation, kernel ω first gets flipped over both axes before being convolved over the input image. Other than that, they are equivalent. The convolutional operation has some nice mathematical properties, we use it in our derivation as it avoids another kernel flipping at the end. However, in practice, the stamps get trained and stored flipped because of the cross-correlation use. The cross-correlation operation is denoted as $I \star \omega$.

3.3.2 Stamping in 3D

Now we expand this concept to three dimensions. Let us define the kernel of the stamp layer as ω. The stamp layer is a transposed convolutional layer; each feature map of this layer is a stamp. When there are multiple feature maps, we obtain multiple stamps stacked on top of each other, resulting in a 3D-tensor ω of size $\psi_x \times \psi_y \times N$, where N is the total number of stamps in the stamp layer.

We convolve this kernel over a three-dimensional one-hot tensor SL_{xyz}, where only $(x^*, y^*, s^*) = 1$ for stamp s^* and coordinate (x^*, y^*). As we only convolve over the x and y dimension, our output will be a two-dimensional image. Only a single location in the stamp dimension of the one-hot tensor will contain a 1 value, thus only one out of the N stamps can be output; all other stamps will be multiplied by 0. This way, the network can select which stamp it wants to output. For a single stamp, we have the following formula:

$$O_{xy} = \mathrm{SL} * \omega = \sum_{z=1}^{N} \left(\sum_{u,v} \mathrm{SL}_{u,v,z}\, \omega_{x-u,y-v,z} \right)$$

Only when $u = x^*$, $v = y^*$ and $z = s^*$, we have that $\mathrm{SL}_{u,v,z} = 1$, therefore:

$$O_{xy} = \omega_{x-x^*,y-y^*,s^*} \tag{3.1}$$

$$O_{x+x^*,y+y^*} = \omega_{xys^*} \tag{3.2}$$

Accordingly, given one-hot tensor SL_{xyz} where $(x^*, y^*, s^*) = 1$, we output stamp s^* at coordinate (x^*, y^*). We can broad this to multiple values where $(x^*, y^*, s^*) = 1$ as well, in the case when there are multiple objects in an image. For each object $1 \leq j \leq m$, let us $(x^{(j)}, y^{(j)}, s^{(j)}) = 1$ in our SL-tensor SL_{xyz}. We can factorise our SL-tensor for each object:

$$\begin{aligned} O_{xy} &= \mathrm{SL} * \omega \\ &= \sum_{j=1}^{m} SL^{(j)} * \omega \\ &= \sum_{j=1}^{m} O_{xy}^{(j)} \end{aligned}$$

Where we define $O_{xy}^{(j)}$ as the output of the convolution for each object separately. For each object $1 \leq j \leq m$, we can use the formula for a single object we derived earlier in Equation (3.1), resulting in:

$$\begin{aligned} O_{xy} &= \sum_{j=1}^{m} O_{xy}^{(j)} \\ &= \sum_{j=1}^{m} \omega_{x-x^{(j)},y-y^{(j)},s^{(j)}} \end{aligned}$$

Which is equivalent to placing stamp $s^{(j)}$ at location $(x^{(j)}, y^{(j)})$ for each stamp $1 \leq j \leq m$. Thus, there is a sum in this equation. When stamps overlap, their values are summed up. Should two stamps be partially overlapped, and thus sum their values, we clip the output to avoid exceeding the maximum image value. Moreover, we constrain the kernel ω to be non-negative and smaller than the maximum image value.

Finally, there is a limit of N stamps stored in the stamp layer, so the network has to select one of these stamps for each object. To minimise the loss, the network selects the stamp that is the closest to the object. Using gradient descent, the selected stamp gets updated to look more like the stamped object. Thus, this stamp itself acts as a prototypical object.

In summary, we use an autoencoder to perform unsupervised learning. We constrain the decoder to only store full objects in the kernel of the final convolutional layer, the stamp layer. The network places these stamps on an empty canvas to reconstruct the input image, given that the input is the sum of 3D one-hot tensors for each stamp we want to place. As the network is only able to select a limited number of stamps, it has to select the stamp that is the closest to the object. This creates a clustering of objects as each stamp represents a set of objects that looks similar to the stamp.

3.4 Selection-and-localisation layer

3.4.1 Gumbel-Softmax

The selection-and-localisation layer (SL-layer) predicts the coordinates (localisation) and selects the stamp used (selection). As mentioned in Section 3.3, when the output of the SL-layer is a one-hot tensor with $(x, y, s) = 1$, stamp s gets placed at location (x, y).

One way to output such tensor is to create a prediction value for each coordinate and stamp. We then perform the arg max function over these logits to obtain a one-hot tensor. This function, however, results in a gradient of 0 for all non-max values, making it unfit for backpropagation as all the parameters behind the non-max values are not updated.

We provide a different solution: we model the coordinate and object-selector as categorical distributions. When we sample from these distributions, we get a one-hot tensor with the predicted stamp and coordinates. The *Gumbel-Softmax* is a differentiable sampling of a smoothly deformed categorical distribution. We explain the Gumbel-Softmax in detail in Section 2.1.2.

Let X, Y and S be the categorical distribution for the x-coordinate, the y-coordinate and stamp type s respectively. We use their marginal distributions and not their joint distributions as this reduces the sample space significantly. Let us sample the sample x, y coordinate and stamp s separately and combine their results in a joint probability distribution:

$$\mathrm{SL}_{xyz} = \Pr[X = x] \Pr[Y = y] \Pr[S = z]$$

Where each of these samplings is calculated according to the softmax. For example, let us calculate the probability that the x-coordinate equals i:

$$\Pr[X = i] = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{x=0}^{\phi_x - \psi_x + 1} \exp((\log(\pi_x) + g_x)/\tau)}$$

Where π_i is the probability logit for the i-th coordinate (which will automatically be estimated using backpropagation) and $g_i \sim \mathrm{Gumbel}(0, 1)$ is noise sampled from the Gumbel distribution. This function also uses temperature τ that allows gradients to flow through during training and gradually anneals the temperature to become a sampling from a categorical distribution [18]. We illustrate the effect of the temperature variable in Figure 2.5.

The network combines the three Gumbel-Softmax outputs by performing, for each of the m objects in the image, the tensor product of the coordinates and the stamp selector probability vector. We obtain m individual selection-and-localisation tensors (SL tensors), as they contain information on the position and type of each shape. In formulae, the individual SL-tensor reads, for all $1 \leq j \leq m$:

$$\mathrm{SL}^{(j)}_{xyz} = \Pr(X^{(j)} = x) \Pr(Y^{(j)} = y) \Pr(S^{(j)} = z)$$

Once every shape has been selected and localised, we combine them into the global selection-and-localisation tensor by summing over j:

$$\mathrm{SL}_{xyz} = \sum_{j=1}^{m} = \mathrm{SL}^{(j)}_{xyz}$$

Which is exactly the SL-tensor we used in Section 3.3 for placing the predicted stamps at the predicted coordinates. We show a depiction of this calculation in Figure 3.4.

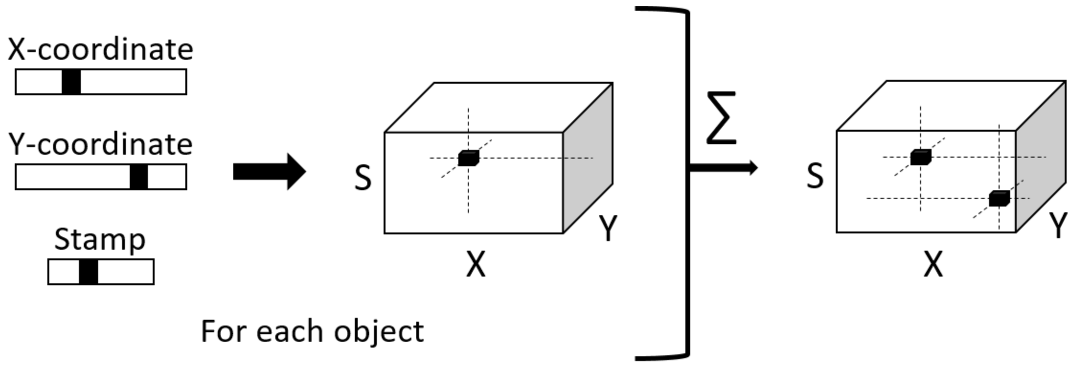


Figure 3.4: The input and output of the SL-layer. On the left, we have three categorical random variables X, Y and S, which output a prediction of stamp s at coordinate (x, y). These are combined using the tensor product for each object in the image and all these predictions are combined into a single tensor on the right. In the stamp layer, the predicted stamps s gets placed at the predicted coordinates (x, y).

3.5 Encoder

The encoder extracts useful features from the input image to create the encoding, which is eventually used to create the discrete latent space of the SL-layer. Since we work with images, we use convolution layers to extract spatial features from the input image, similar to VGG [38]. We use the Leaky ReLU activation functions to avoid the vanishing gradient problems of sigmoid [16] and the dying ReLU problem of a regular ReLU activation function.

After each convolutional layer, the network contains a dropout layer to avoid overfitting [40]. Moreover, the network contains a batch normalisation step to improve the stability and speed of the training [17]. In our current task, it is especially effective, as otherwise the network gets stuck in a local minimum (by, for example, only using a single stamp). Finally, we use max-pooling to reduce the size of the image so we can obtain the encoding.

3.6 Complete architecture

We show the complete architecture in Figure 3.5. StampNet is an autoencoder that trains under the constraint of minimising the Euclidean distance between the input and the output. It consists of an encoder that creates the encoding of the input image and a decoder that puts stamps on the image to reproduce the input image by predicting the coordinates with the SL-layer (Section 3.4) and putting stamps on these predicted coordinates in the stamp layer (Section 3.3).

In short, for each object in every image, we obtain a coordinate prediction (x, y) as well as the stamp s used for this object. When we look at our problem formulation in Section 3.1, the objective was: (1) to predict the location of each object and (2) to cluster each object into clusters. For each object, we use the predicted (x, y) for the localisation objective and use stamp s for the clustering of objects.

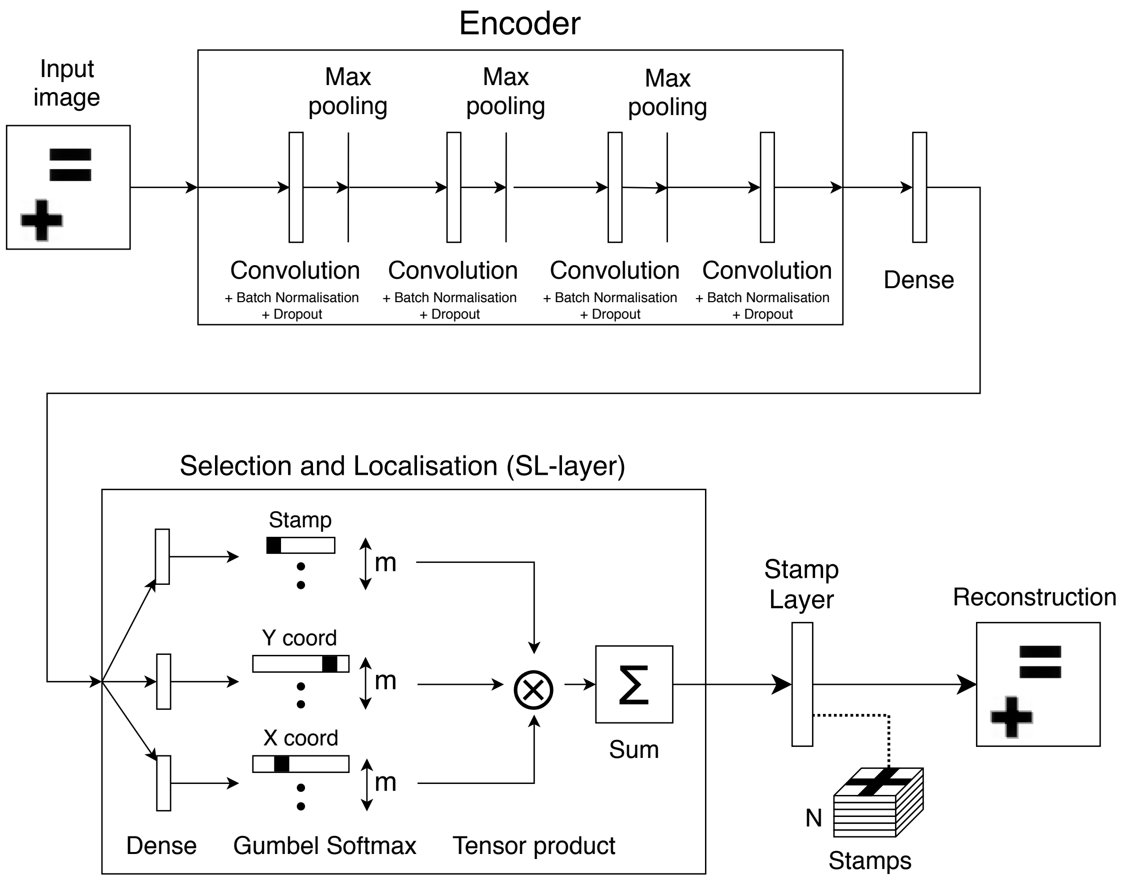


Figure 3.5: The complete network architecture of StampNet. The encoder extracts spatial features of the input image into an encoding, using a set of convolutional layers and pooling layers. The decoder works in two steps: first, the SL-layer predicts the coordinates and stamp type of each object in the input image. Then, the selected stamps are placed according to the predicted positions to generate the output image. The stamps are learned automatically using backpropagation.

Chapter 4

Results

In unsupervised multi-class object discovery, we have datasets consisting of images with one or more objects. These objects have to be discovered and extracted for each image. In this chapter, we evaluate the performance of StampNet on several of these datasets.

However, as the multi-class version of unsupervised object discovery is a new task, there do not yet exist standardised datasets for this problem. Therefore, we create a new dataset with simple shapes to illustrate a simple version of the problem (Section 4.2). Moreover, we augment the existing datasets Translated MNIST (T-MNIST) and Cluttered-Translated MNIST (CT-MNIST) [28] by adding more objects to each image of the dataset in Section 4.3. Finally, we demonstrate an application for StampNet (Section 4.4).

At the start of each of these sections, we explain the details of each of the datasets. We explain the method of testing in Section 4.1, where we go into detail of the measures and the parameters of the model.

Dataset	**CorLoc**	**IoU**	**Purity**
Simple Shapes ($m = 1$)	0.9999	0.9718	0.9928
Simple Shapes ($m = 2$)	0.9828	0.9500	0.9564
T-MNIST ($m = 1$)	0.9983	0.8925	0.7891
T-MNIST ($m = 2$)	0.9729	0.8537	0.5277
CT-MNIST ($m = 1$)	0.9972	0.8912	0.8113
CT-MNIST ($m = 2$)	0.9545	0.8394	0.6149
Pedestrian Tracking ($m = 3$)	0.7816[a]	0.6308[a]	0.9597

[a]Calculated for localizing the pedestrian, not the walls.

Table 4.1: Experiment results of StampNet on all datasets. We report three metrics, CorLoc, IoU, and Purity, to take into account the score for object discovery, object localisation and object clustering respectively (see Section 4.1 for more information about these metrics). The CorLoc and IoU metric for the pedestrian dataset is slightly different from the other datasets, as only the ground-truth values for the pedestrian are known.

4.1 Method of testing

The network has been trained on a training set and evaluated on a separate test set. We use an annealing schedule of $\tau = \max(0.2, 7.0 \cdot \exp(-0.01t))$ for the Gumbel-Softmax, updated every epoch, similar to the schedule used in the Gumbel-Softmax paper [18]. For the results, we use a temperature of $\tau = 0.001$ to enforce a one-hot choice of the stamps. We list the remaining hyperparameters in Table B.1.

In each image, there are one or more objects together with their bounding box of size $\psi_x \times \psi_y$. We test the performance of the task on three tasks: object localisation, object discovery and clustering performance of the extracted stamps. For each of these tasks, we use a metric to score the network to get an idea of how well the network performs unsupervised object discovery.

4.1.1 Object localisation

For object localisation, we use the Intersection over Union (IoU) measure [11]. The IoU measures the overlap between two different objects; a higher IoU means that the two objects overlap more. We use it to measure how well our predicted bounding box overlaps with the ground-truth bounding box of an object. The formula for IoU is:

$$\text{IoU}(i) = \frac{area(B^{(i)}) \cap area(B_p^{(i)})}{area(B^{(i)}) \cup area(B_p^{(i)})}$$

Where $B_p^{(i)}$ and $B^{(i)}$ are the predicted bounding boxes and the ground truth bounding boxes respectively for all objects of image i and $area(B)$ is the area bounded by B. These bounding boxes consist of the bounding box of each object separately:

$$area(B^{(i)}) = \bigcup_{j=1}^{m} area(B^{(i,j)})$$

And:

$$area(B_p^{(i)}) = \bigcup_{j=1}^{m} area(B_p^{(i,j)})$$

We combine the areas of the bounding boxes before calculating the IoU (instead of calculating the IoU for each object separately), because it avoids having to assign which predicted bounding box belongs to which true bounding box.

We average the IoU of each image separately to obtain the total average IoU:

$$\text{IoU} = \frac{1}{n}\sum_{i=1}^{n} \text{IoU}(i)$$

4.1.2 Object discovery

We consider an object discovered when the IoU is greater or equal than 0.5. The ratio of all objects discovered is called CorLoc [5] and we use this measure for object discovery. CorLoc is often used in literature as a measure for object discovery [5, 29, 19, 41].

We calculate the CorLoc measure as follows: first, we assign the predicted bounding boxes to the closest true bounding boxes in a greedy manner. Let $B_P^{i,j}$ be the reassigned index for image i and object j. In turn, we calculate the IoU of each object and their respective predicted bounding box:

$$\text{IoU}(i,j) = \frac{area(B^{(i,j)}) \cap area(B_p^{(i,j)})}{area(B^{(i,j)}) \cup area(B_p^{(i,j)})}$$

We use this to calculate the CorLoc of each object:

$$\text{CorLoc}(i,j) \begin{cases} 1 & \text{if IoU}(i,j) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

The total CorLoc is the average of the CorLoc of all images and objects:

$$\text{CorLoc} = \frac{1}{nm} \sum_{i,j} \text{CorLoc}(i, j)$$

4.1.3 Object clustering

To measure how well the model can differentiate between classes, we use a measure called clustering purity [46]. We consider each of the predicted stamp as a cluster and we want to find out how 'pure' our clusters are. In other words, suppose we label each cluster perfectly, what accuracy would we be able to obtain. Generally, a larger purity corresponds to a better clustering.

For each observation, we predict the stamp s it belongs to. These stamps form a clustering of the data, each observation belonging to the predicted stamp, thus we obtain clusters s. We use this to calculate the purity of a cluster s by first discerning the label that results in the maximum number of correct labellings:

$$\text{MaxCorrect}(s) = \max_{1 \leq l \leq M} \nu_s^l$$

Where ν is the number of samples in cluster s that belong to ground-truth class l. We sum this up for all our clusters and normalise it to obtain our true clustering purity:

$$\text{Purity} = \frac{1}{n} \sum_{s=1}^{N} \text{MaxCorrect}(s)$$

As a small side note, another common measure for clustering is Normalized Mutual Information (NMI) that calculates the mutual information between the clustering and the true labelling. However, because the mutual information is normalized, the NMI decreases when there are more clusters. We use more stamps than classes as we are not interested in perfect clustering, but instead are more interested which objects are inside the training data. This is also more realistic, as in a real dataset the number of classes might be unknown. Hence we use clustering purity. An added advantage of using clustering purity is that we can compare it to accuracy, unlike NMI.

4.2 Simple Shapes

4.2.1 Data

The Simple Shapes dataset uses $M = 5$ different simple shapes: a 'triangle', a rotated 'equal', a 'plus', an 'equal' and a 'slash', which we show these in Figure 4.1. The pixel values of these shapes are set to 1 while the rest of the image equals 0.

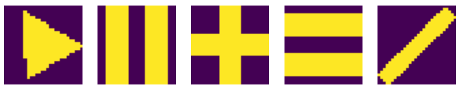


Figure 4.1: The different shapes of the Simple Shapes dataset.

We randomly place $m = 2$ shapes of size 28×28 on an empty canvas of size 84×84. We consider 50000 generated training samples and 10000 generated test samples, a fixed amount of samples similar to the size of MNIST. The stamp layer contains $N = 10$ stamps, greater than the available $M = 5$ different classes.

4.2.2 Results

In Figure 4.2, we report the $N = 10$ stamps learned by the network. We observe that these shapes are nearly identical to the shapes used to generate the dataset, see Figure 4.1. From this observation, we conclude that the network can extract the simple objects of the input images and store them in the stamps of the network.

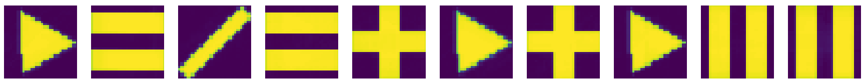


Figure 4.2: The stamps learned by the network in the simple shapes dataset for $m = 2$. These stamps are stored in the stamp layer (rotated) and placed on an empty canvas by the network to reproduce the input image. Note that these stamps look very similar to the stamps of the ground truth in Figure 4.1.

We detail sample prediction outputs in Figure 4.3. In these samples, the network localises the shapes and assigns the correct stamp to each shape. Even when the shapes overlap, the network can predict the correct shape and stamp.

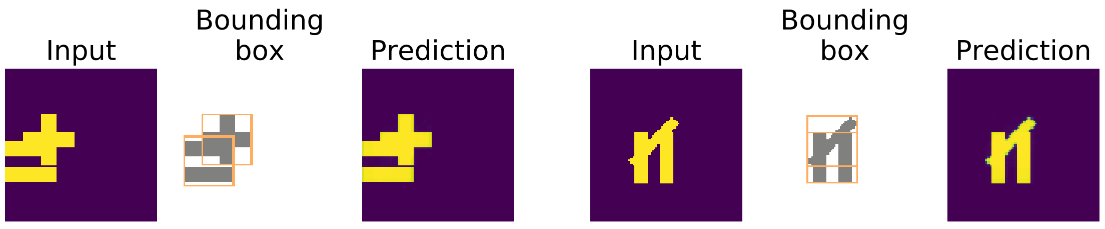


Figure 4.3: Sample predictions from the Simple Shapes dataset for $m = 2$. These observations contain overlapping shapes and we learn that the network can detect these shapes, bound them and put the correct stamp on their locations.

In Table 4.1, we quantify the CorLoc, IoU and purity evaluated over the test set. For $m = 1$, the CorLoc of 0.9999 indicates that almost all of the objects in the test set has been discovered by the network. Moreover, we achieve high scores in IoU and purity on this dataset.

The scores are slightly lower with $m = 2$ shapes on the canvas. The remaining errors are the result of two problems: (1) the network assigns one stamp at the coordinate of the other stamp and (2) the network assigns an inaccurate stamp to an object. We show an example of the latter problem in Figure 4.4. However, we still have a purity and CorLoc of over 0.95, so these problems occur rarely.

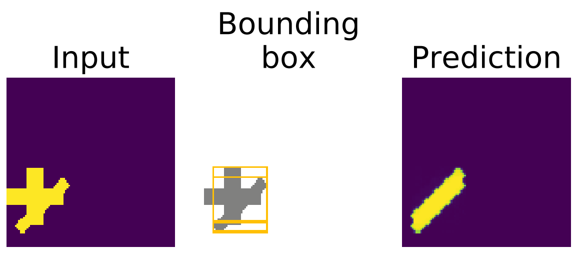


Figure 4.4: A sample where the network assigns a wrong object class to an object. While the network should put a 'plus' stamp, it instead stamps the image with the 'slash' stamp.

To get a better idea of the clustering purity, we plot the usages of the stamps for each sample in Figure 4.5. We observe that, while not uniformly, all stamps are used during testing. Interestingly, while the two stamps of the 'plus' class are used around the same amount of times, it is not the case for the two 'equals' classes. This depends on how the network optimises the stamp selection for each class. Both the case of having a single stamp for one class or multiple stamps for one class are equally valid solutions, as the stamps or clusters themselves are still 'pure', i.e. only for that particular class. We observe that the network optimises to both solutions.

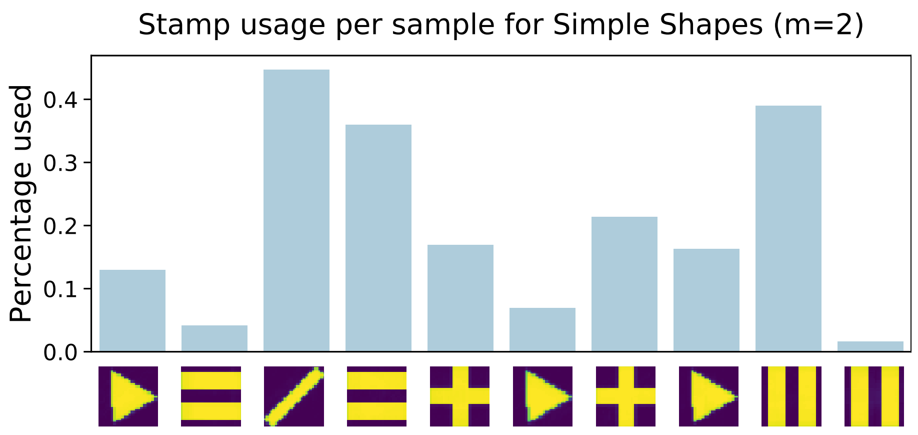


Figure 4.5: The stamp usage per sample for the Simple Shapes dataset ($m = 2$). Two of the learned stamps are used to reproduce the input image.

4.3 Translated and Cluttered Translated MNIST

4.3.1 Data

Both Translated MNIST (T-MNIST) and Cluttered Translated MNIST (CT-MNIST) are variations on the MNIST handwritten digit dataset [22]. We create the T-MNIST by translating the digit randomly on an empty canvas. In CT-MNIST, we add an extra step of background clutter in the form of smaller MNIST digits [28].

In T-MNIST, MNIST digits are uniformly placed on an empty canvas of size 84×84. These MNIST digits have size 28×28. In CT-MNIST, these MNIST digits are uniformly placed on an empty canvas of size 100×100 instead, to best compare our results with existing literature. For CT-MNIST, we additionally add clutter by uniformly placing 8 smaller clutter digits of size 8×8 to the dataset.

In both cases, we generate 60000 training samples and 10000 test samples. We test for $m = 1$ and $m = 2$ digits on the canvas with $N = 40$ stamps.

Network	**Accuracy**
RAM [28]	0.927
DRAW [15]	0.966
RNN-SPN [39]	0.985
DCN [2]	0.986

Table 4.2: Supervised classification CT-MNIST for $M = 1$ digit reported in other literature. For comparison, the clustering purity of our unsupervised model is 0.811 for this task.

4.3.2 Results

We observe that the network learns different prototypical MNIST digits for both T-MNIST and CT-MNIST, see Figure 4.6 and Figure 4.7 respectively. These stamps capture a variety of different shapes and numbers in the dataset. However, some of these digits look like a combination of MNIST digits, as we see in Figure 4.6; some stamps look like a combination between a 9 and an 8. We believe that this stems from the loss function; since the 8 and 9 look similar and have a large overlap, the loss is relatively low when a combination digit is stamped on the location. This reduces the clustering purity as a single stamp is used for two classes.

We show sample predictions in Figure 4.8 and Figure 4.9 for T-MNIST and CT-MNIST respectively. The network can recognise overlapping shapes and can reproduce these images by stamping the correct stamp on their locations. In case of the cluttered version, the network is unable to reproduce the clutter, however, stamping the large MNIST digits still results in the least error.

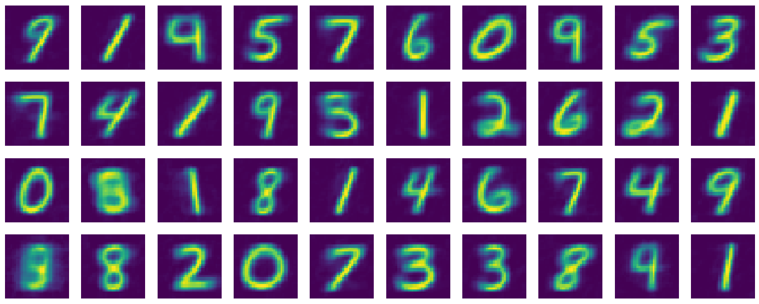


Figure 4.6: The stamps learned from the T-MNIST dataset when $m = 2$. These stamps capture a variety of different shapes of the MNIST dataset. Most of these digits can be classified as they look like digits.

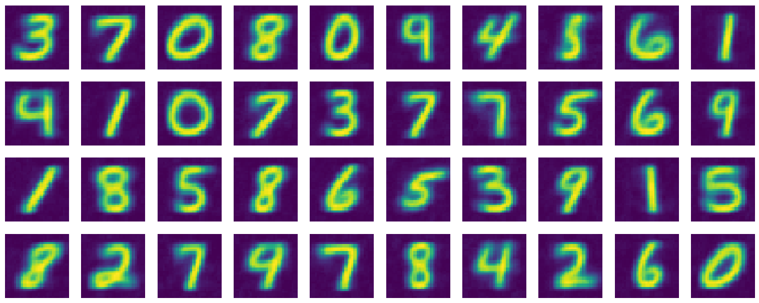


Figure 4.7: The stamps learned from the CT-MNIST dataset when $m = 2$. These stamps capture a variety of different shapes of the MNIST dataset. These stamps are notably thicker than the stamps learned from the T-MNIST dataset in Figure 4.6.

The network discovers most MNIST digits, as the CorLoc measure (Table 4.1) indicate. Even when there are $m = 2$ digits on a cluttered canvas, we observe that the network discovers over 95% of the digits. The added clutter results in a slight drop of localisation (Table 4.1) and an increase in purity.

We note a comparison of supervised classification of CT-MNIST for $M = 1$ digit in Table 4.2. We observe that without supervision, StampNet performance comes near these supervised alternatives using purity as the measure for comparison, i.e. when all stamps are labelled correctly. Note that these networks only classify a single digit, while our model can discover multiple digits in each image.

To explain the increase in purity, we look at the figures that denote the stamp usage of T-MNIST (Figure 4.10) and CT-MNIST (Figure 4.11). We see that T-MNIST mainly uses a single, general stamp that is a combination of many different MNIST digits, while the case of CT-MNIST is much more uniform. By using a single stamp more often, this stamps represents more shapes and more classes, reducing the clustering purity. We hypothesise that the extra clutter acts as a regularisation, forcing the network to use more stamps to get a more accurate reconstruction.

Finally, we visualise the training of CT-MNIST in Appendix A. The network initially learns rough blocks as stamps and finds the rough location of a single digit. As training continues, the stamps start to look more like MNIST digit and the location gets more narrowed down. We see that the network first finds the first digit, after which it finds the second digit.

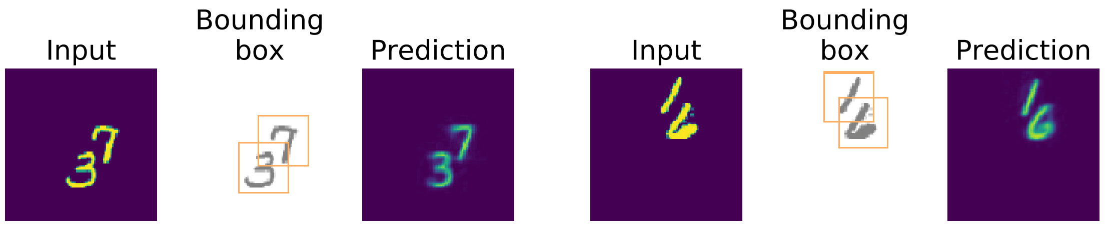


Figure 4.8: Sample predictions from the T-MNIST dataset for $m = 2$.

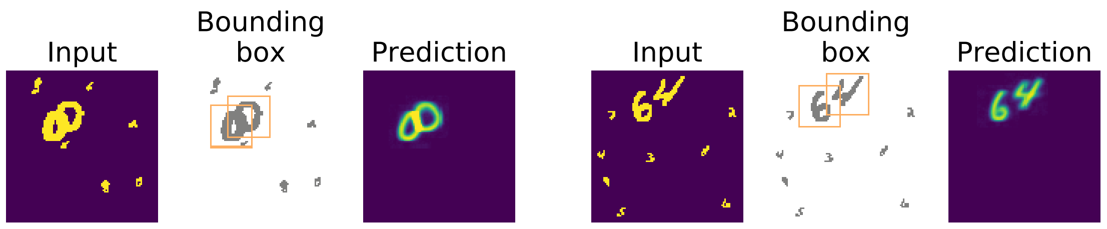


Figure 4.9: Sample predictions from the CT-MNIST dataset for $m = 2$.

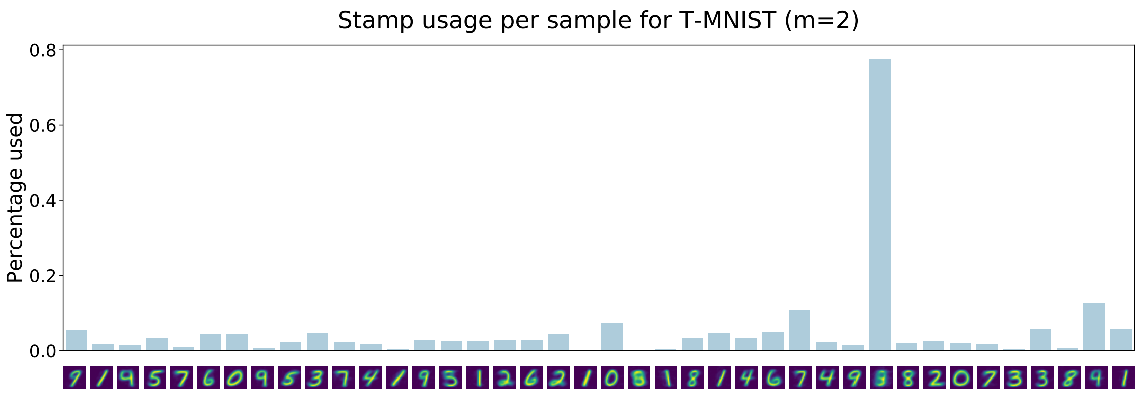


Figure 4.10: The stamp usage per sample for the T-MNIST dataset ($m = 2$). Two of these learned stamps are used to reproduce the input image. The high usage of a single stamp is notable, resulting in a reduced clustering performance as said stamp represents multiple classes.

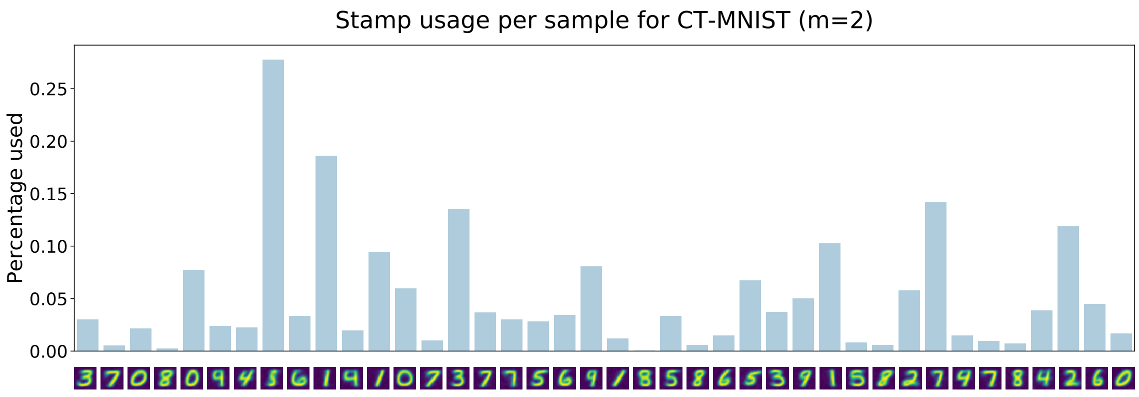


Figure 4.11: The stamp usage per sample for the CT-MNIST dataset ($m = 2$). Two of the learned stamps are used to reproduce the input image. Compared to the stamps usages of T-MNIST in Figure 4.6, these usages are more uniform.

4.4 Pedestrian localisation in overhead depth images

4.4.1 Data

Overhead depth maps are an increasingly popular tool to perform high accuracy pedestrian tracking for studying the dynamics of human crowds in real-life venues (e.g. [37, 4, 8]). Overhead depth maps come in the form of greyscale images where the value of each pixel corresponds to the distance from the camera. In overhead depth view pedestrians have similar "ovoid" shape, which is different from that of walls, objects and so on. The characteristics of this dataset make it very suitable for the network's object discovery capability.

We consider here a reduced depth dataset from the real-life crowd tracking experiment [7], annotated with bounding boxes (image size: 80×80, bounding boxes size 40×40. See the sample on the left side of Figure 4.14, which displays a pedestrian on the left side and a wall on the right side. In the dataset, there are only images with either a wall on the right side or the left side.

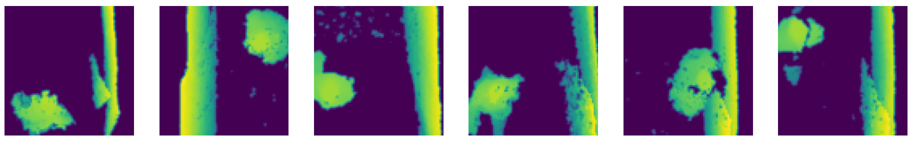


Figure 4.12: Sample observations of the Pedestrian dataset, a dataset containing overhead depth maps of pedestrians. The 'ovoid' shapes in the image are the pedestrians and the vertical lines are the walls. These walls are canvasses that are affected by the wind, thus they are not constant in each image.

We test the network considering $N = 40$ stamps of size 40×40 and $m = 3$ stamps simultaneous on the image. We only evaluate the discovery and localisation of the pedestrian stamp and not the walls, as the ground truth bounding boxes for the walls is non-existent and the main goal is to detect the pedestrian. Still, we will test the capability of the network to differentiate between pedestrians and the wall on the side. For evaluating all these metrics, we assign the stamp that is closest to the bounding box of the pedestrian.

4.4.2 Results

We illustrate samples of the results in Figure 4.14. In both cases, we observe that the network places a single stamp on the pedestrian and two stamp on the walls. The stamps extracted by the network Figure 4.13 show that different objects are successfully captured by the network, as we can differentiate between the pedestrians and walls object.

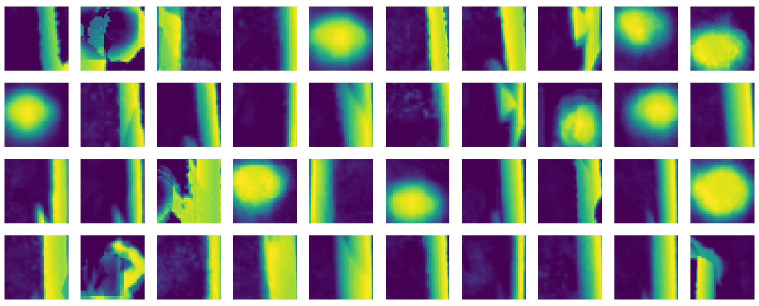


Figure 4.13: The stamps learned from the Pedestrian dataset when $m = 3$. We recognise different objects within these stamps; the 'ovoid' objects represent a pedestrian whereas the vertical line represents a wall. A few of the stamps contain noise, although we see in Figure 4.15 that these are not used in the predictions.

We present the measures in Table 4.1. The high clustering purity of 0.9597 indicates that the network is capable to differentiate between pedestrians and walls. The results for the discovery and localisation of the pedestrian are lower, however. The CorLoc of 0.7816 indicates that the network can find the pedestrian only in around 78% of the images. The IoU is also lower compared to the results of the other results, such as CT-MNIST.

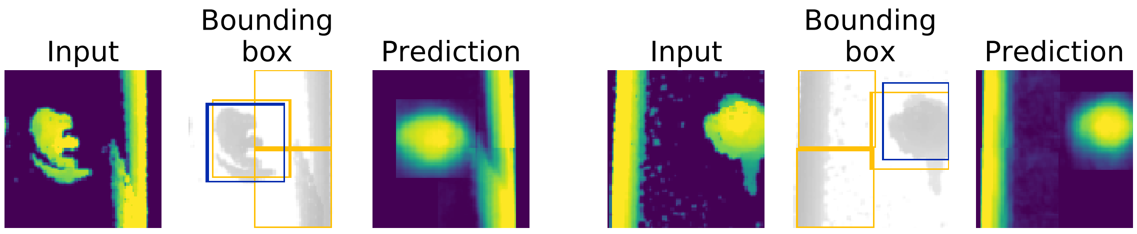


Figure 4.14: Sample predictions from the Pedestrian dataset for $m = 3$. The true bounding box is coloured blue and the predicted bounding boxes in orange. We observe that the network can reconstruct the image by placing stamps for the pedestrian and the wall on their respective locations.

The lower measures are notable compared to the other datasets in this thesis. We argue that there are two reasons for this: firstly, the quality of the dataset is lower than the perfectly generated MNIST or shape dataset. The bounding boxes of the pedestrian sometimes go over the edges, resulting in a smaller ground-truth bounding box, making it much more difficult to obtain an IoU of over 0.5 for the CorLoc measure. Moreover, the ground truth data is not a perfect bounding box either. We can see both of these issues at the right observation of Figure 4.14, where the blue bounding box does not capture the pedestrian perfectly. Secondly, the dataset contains real-world images and thus contains a lot of noise. This could lower the capability of the network to localise the correct objects.

We illustrate the stamp usages of the network in Figure 4.13. For the reconstruction of the pedestrian, a single stamp is mainly used by the network. However, to reproduce the various different types of walls in the dataset, the network uses different shapes of the walls.

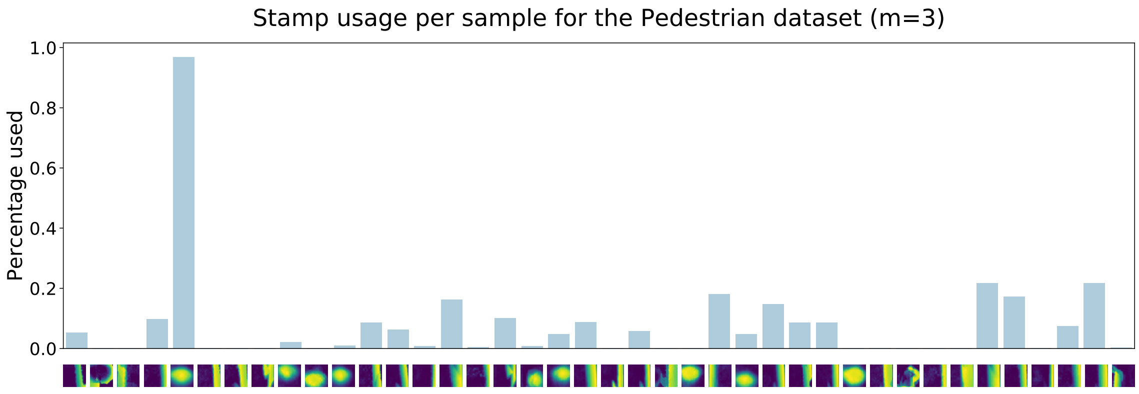


Figure 4.15: The stamp usage per sample for the CT-MNIST dataset ($m = 2$). We observe that a single pedestrian stamp is used very often for the reproduction of the pedestrian, whereas the stamps used to reproduce the walls are more varied.

Chapter 5

Conclusion

In this thesis, we have introduced StampNet to localise multiple objects from multiple classes in an unsupervised manner. StampNet is a neural network that is capable of discovering, localising and clustering multiple objects simultaneously on images.

We accomplish this task by incorporating the predictions of the bounding boxes in an autoencoder, removing the need for labels. We achieve this by placing the kernel of a convolutional layer, called stamps, at location coordinates predicted by the SL-layer, a discrete latent space (see Figure 3.5 for more detail). The stamps are learned automatically through backpropagation and represent prototypical objects of the image.

Following the second goal stated in Section 1.3, we have introduced new datasets to test for unsupervised multi-class object discovery. We created a simple dataset, the Simple Shapes dataset, that tests the ability to localise and differentiate basic shapes (Section 4.2). Additionally, we generated a more complex dataset based on the MNIST digits, namely T-MNIST and CT-MNIST, where we have MNIST digits randomly placed on an image, without and with clutter in the background. Finally, we provided measures for each of the subtasks we identified, namely object discovery, object localisation and object detection.

We also provided an application of StampNet in the form of pedestrian localisation in overhead depth images in Section 4.4, as aimed for by our third goal. The network finds and localises both the pedestrian and walls in these overhead depth images. We can differentiate between the pedestrians and walls quite well in Figure 4.13.

The results of StampNet on the Simple Shapes and both MNIST datasets show that our model can perform multi-class object discovery for multiple simple objects (goal 1). We observe that the network is even capable of detecting and localising multiple overlapping objects, as we see in Figure 4.9. As noted in Table 4.1, the network can discover most objects. However, the clustering purity is much lower for these tasks, which we attribute in part due to the unbalanced use of certain stamps (for T-MNIST-2 in particular Figure 4.10). Still, compared to other supervised methods on the CT-MNIST-1 dataset in Table 4.2, StampNet performed pretty well for an unsupervised method. Thus, while StampNet can localise and discover these objects, the clustering is not ideal when there are multiple MNIST digits in the image.

We believe that this network can be useful on the task of pedestrian localisation in overhead depth images. While the network is only able to localise the pedestrian around 78% of the time, in reality, the data comes from a video feed. Thus, we can combine StampNet with statistical methods to create a more consistent localisation, as localisation between images close in time should be similar. Moreover, due to the high clustering purity of the network, all stamps that are considered a pedestrian are, most likely, a pedestrian, which is useful for pedestrian the detection task.

Finally, we can view StampNet in a different light; we can consider StampNet as an algorithm similar to k-means clustering. While k-means and StampNet differ greatly, StampNet divides the dataset into k clusters based on minimising the MSE measure, similar to k-means clustering. However, StampNet adds functionality by being able to: (1) use convolutional shapes in deciding the clusters, (2) able to localise these clusters and (3) able to cluster multiple objects simultaneously. Thus, we can see StampNet that can perform a k-means clustering with neural networks while having additional functionality.

5.1 Limitations and future work

The main limitation of StampNet is also the first set of constraint as part of the research scope. We only detected relatively simple objects of fixed sizes, as this made it easier to obtain a similarity measure between objects. The measure of overlap has been used in the loss function of the neural network, per-pixel Euclidean distance. This limits the use of our network in real-life image considerably.

There are several paths for future work on this topic. First of all, the network sometimes uses a single specific stamp too often resulting in a reduced accuracy (Figure 4.10). Future work can look into ways to make the stamp usages more uniform to increase the performance of the network.

Another idea to improve the performance of the network is to use the fact that the predicted coordinates have spatial properties. Instead of the categorical distribution, we could predict the coordinates using the multivariate distribution which might reduce the complexity space of the random variable.

Moreover, we could find new applications for our current work in the field of clustering. StampNet can perform clustering without any localisation. However, combining clustering and localisation could perhaps result in a better clustering. For example, in the case of the Omniglot dataset, the clustering could improve by creating stamps smaller than the actual image, making the clusters more invariant to translations.

Finally, we can look into more complicated stamps to avoid the main limitation of this work. If we could place more complicated stamps, the number of practical applications would raise considerably. We recommend looking into making the stamps dependent on the input, allowing more combinations of the stamps.

Bibliography

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2189–2202, 2012. 9

[2] Amjad Almahairi, Nicolas Ballas, Tim Cooijmans, Yin Zheng, Hugo Larochelle, and Aaron Courville. Dynamic capacity networks. In *International Conference on Machine Learning*, pages 2549–2558, 2016. 24

[3] Hakan Bilen and Andrea Vedaldi. Weakly supervised deep detection networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2846–2854, 2016. 9

[4] D. Brščić, T. Kanda, T. Ikeda, and T. Miyashita. Person tracking in large public spaces using 3-d range sensors. *IEEE Transaction on Human-Machine Systems*, 43(6):522–534, 2013. 27

[5] Minsu Cho, Suha Kwak, Cordelia Schmid, and Jean Ponce. Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1201–1210, 2015. 9, 20

[6] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Weakly supervised object localization with multi-fold multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 39(1):189–203, 2017. 9

[7] Alessandro Corbetta, Werner Kroneman, Maurice Donners, Antal Haans, Philip Ross, Marius Trouwborst, Sander Van de Wijdeven, Martijn Hultermans, Dragan Sekulovski, Fedosja van der Heijden, et al. A large-scale real-life crowd steering experiment via arrow-like stimuli. *arXiv preprint arXiv:1806.09801*, 2018. 27

[8] Alessandro Corbetta, Jasper A. Meeusen, Chung-min Lee, Roberto Benzi, and Federico Toschi. Physics-based modeling and data representation of pairwise interactions among pedestrians. *Physical Review E*, 98:062310, Dec 2018. 27

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[10] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997. 9

[11] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 20

[12] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017. 8

[13] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 8

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 8

[15] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015. 24

[16] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998. 16

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 16

[18] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 4, 7, 15, 19

[19] Armand Joulin, Kevin Tang, and Li Fei-Fei. Efficient image and video co-localization with frank-wolfe algorithm. In *European Conference on Computer Vision*, pages 253–268. Springer, 2014. 9, 20

[20] Vadim Kantorov, Maxime Oquab, Minsu Cho, and Ivan Laptev. Contextlocnet: Context-aware deep network models for weakly supervised localization. In *European Conference on Computer Vision*, pages 350–365. Springer, 2016. 9

[21] Gunhee Kim and Antonio Torralba. Unsupervised detection of regions of interest using iterative link analysis. In *Advances in neural information processing systems*, pages 961–969, 2009. 1, 9

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4, 24

[23] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 8

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 8, 9

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 5

[26] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016. 7

[27] Santiago Manen, Matthieu Guillaumin, and Luc Van Gool. Prime object proposals with randomized prim's algorithm. In *Proceedings of the IEEE international conference on computer vision*, pages 2536–2543, 2013. 9

[28] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014. 19, 24

[29] Kazuhiko Murasaki, Yukinobu Taniguchi, and Tetsuya Kinebuchi. Unsupervised multi-class object discovery by spherical clustering of deep features. *ITE Transactions on Media Technology and Applications*, 7(1):2–10, 2019. 9, 20

[30] Megha Pandey and Svetlana Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *2011 International Conference on Computer Vision*, pages 1307–1314. IEEE, 2011. 9

[31] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 8, 9

[32] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 8, 9

[33] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1, 8, 9

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 8, 9

[35] Michael Rubinstein, Armand Joulin, Johannes Kopf, and Ce Liu. Unsupervised joint object discovery and segmentation in internet images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1939–1946, 2013. 9

[36] Olga Russakovsky, Yuanqing Lin, Kai Yu, and Li Fei-Fei. Object-centric spatial pooling for image classification. In *European conference on computer vision*, pages 1–15. Springer, 2012. 9

[37] S. Seer, N. Brändle, and C. Ratti. Kinects and human kinetics: A new approach for studying pedestrian behavior. *Transportation Research part C: Emerging technologies*, 48:212–228, 2014. 27

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 16

[39] Søren Kaae Sønderby, Casper Kaae Sønderby, Lars Maaløe, and Ole Winther. Recurrent spatial transformer networks. *arXiv preprint arXiv:1509.05329*, 2015. 24

[40] 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*, 15(1):1929–1958, 2014. 16

[41] Kevin Tang, Armand Joulin, Li-Jia Li, and Li Fei-Fei. Co-localization in real-world images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1464–1471, 2014. 1, 9, 20

[42] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013. 9

[43] Sara Vicente, Carsten Rother, and Vladimir Kolmogorov. Object cosegmentation. In *CVPR 2011*, pages 2217–2224. IEEE, 2011. 9

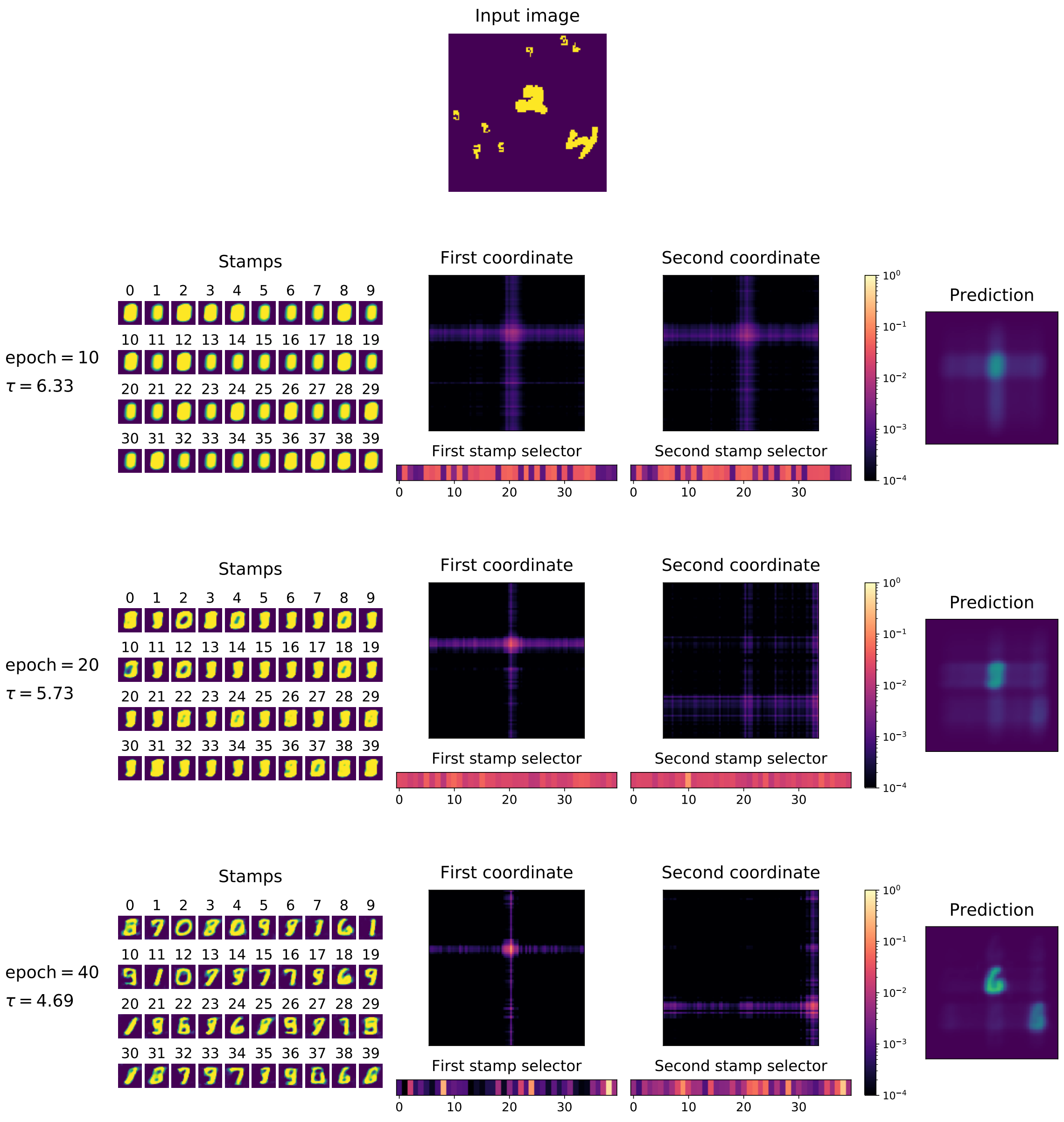
[44] Joost Visser, Alessandro Corbetta, Vlado Menkovski, and Federico Toschi. Stampnet: unsupervised multi-class object discovery. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2951–2955. IEEE, 2019. 4

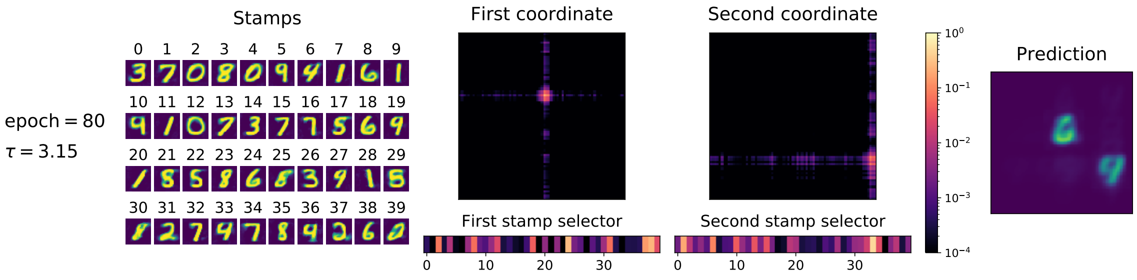
[45] Fan Wang, Qixing Huang, Maks Ovsjanikov, and Leonidas J Guibas. Unsupervised multi-class joint image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3142–3149, 2014. 9

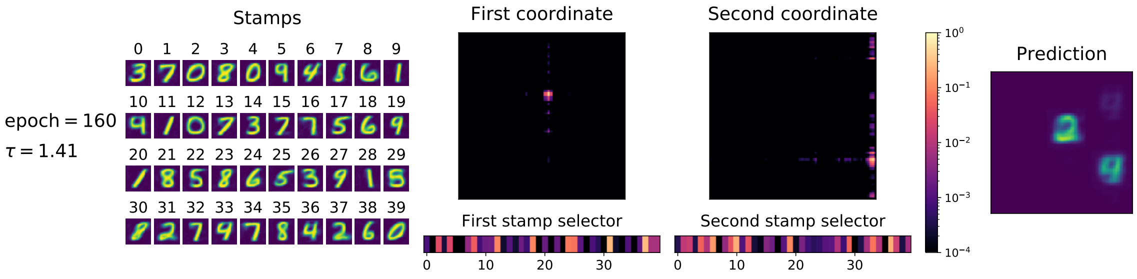
[46] Zhirong Yang, Tele Hao, Onur Dikmen, Xi Chen, and Erkki Oja. Clustering by nonnegative matrix factorization using graph random walk. In *Advances in Neural Information Processing Systems*, pages 1079–1087, 2012. 21

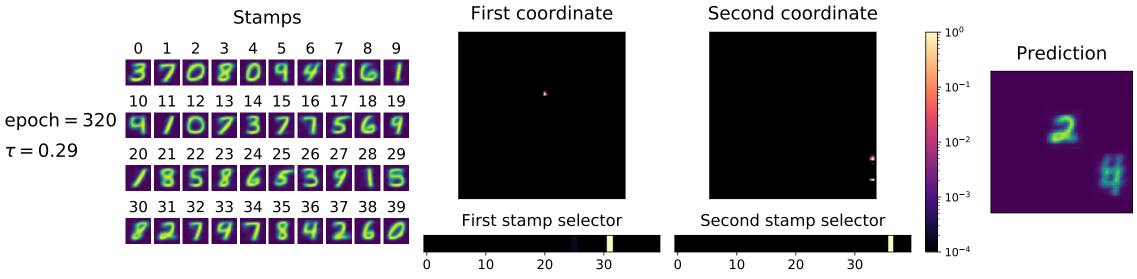
Appendix A

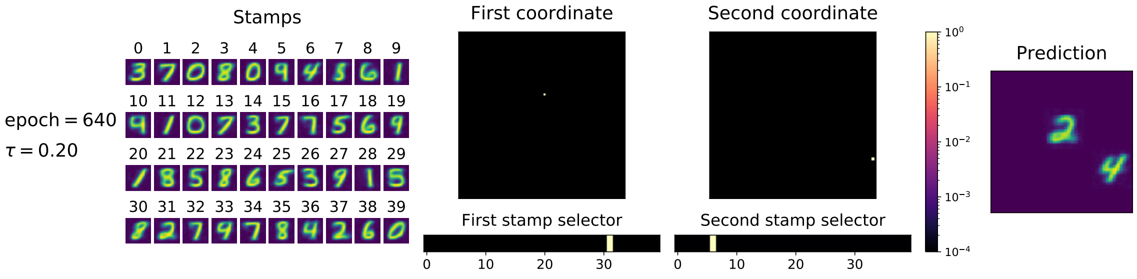
Training visualisation




epoch = 80
τ = 3.15
Stamps
First coordinate
Second coordinate
First stamp selector
Second stamp selector
Prediction


epoch = 160
τ = 1.41
Stamps
First coordinate
Second coordinate
First stamp selector
Second stamp selector
Prediction


epoch = 320
τ = 0.29
Stamps
First coordinate
Second coordinate
First stamp selector
Second stamp selector
Prediction


epoch = 640
τ = 0.20
Stamps
First coordinate
Second coordinate
First stamp selector
Second stamp selector
Prediction

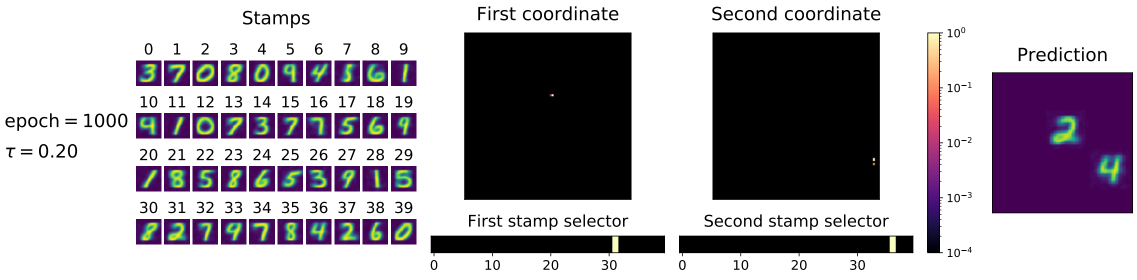


Figure A.1: A three-page illustration of how the network trains over epochs for a single input image of the CT-MNIST dataset. We see that initially, after 10 epochs, the network learns ‘blobs’ as stamps. These blobs are used to find a rough location of the first object using both placeable stamps. The stamp selector shows that almost all stamps are used to determine this. After 40 epochs, we observe that the network locates the second object and the stamps itself start to look more like MNIST digits. The network starts to narrow down both locations while continuing the training, as well as reducing the number of stamps used to create the object. Finally, we see only a single coordinate and stamp for each object at the end of the training.

Appendix B

Hyperparameters

Hyperparameter	**Value**
epochs	1000
steps_per_epoch	128
batch_size	32
dropout_rate	0.2
conv_kernel_size	$(3, 3)$
tau_init	7.0
anneal_rate	0.005
min_temperature	0.2

Table B.1: The hyperparameters we use for training the StampNet model.

Appendix C

Paper version

The following pages contain the paper version of this research, which has been published in 2019 in the IEEE International Conference on Image Processing (ICIP).

STAMPNET: UNSUPERVISED MULTI-CLASS OBJECT DISCOVERY

Joost Visser *Alessandro Corbetta* *Vlado Menkovski* *Federico Toschi*

Eindhoven University of Technology

ABSTRACT

Unsupervised object discovery in images involves uncovering recurring patterns that define objects and discriminates them against the background. This is more challenging than image clustering as the size and the location of the objects are not known: this adds additional degrees of freedom and increases the problem complexity. In this work, we propose StampNet, a novel autoencoding neural network that localizes shapes (objects) over a simple background in images and categorizes them simultaneously. StampNet consists of a discrete latent space that is used to categorize objects and to determine the location of the objects. The object categories are formed during the training, resulting in the discovery of a fixed set of objects. We present a set of experiments that demonstrate that StampNet is able to localize and cluster multiple overlapping shapes with varying complexity including the digits from the MNIST dataset. We also present an application of StampNet in the localization of pedestrians in overhead depth-maps.

Index Terms— object discovery, unsupervised learning, image localization, image clustering

1. INTRODUCTION

Discovery and localization of objects is an important task in computer vision and image analysis. There is a significant amount of existing methods that successfully address the challenge of object localization when objects are explicitly annotated with labels and bounding boxes. Deep convolutional neural networks such as YOLO [1] and Faster R-CNN [2] demonstrated significant success. Annotations, however, often require major human effort and thus come with significant costs.

In contrast, object discovery deals with localization in absence of annotations. This means finding and clustering recurring patterns that define the objects. Previous work on unsupervised localization typically addresses one object class in multiple images, which is referred to as object co-localization [3, 4]. However, unsupervised localization of multiple classes of objects remains a significant challenge. This problem is further adds complexity as we cannot assume that only a single object is present in the image. In other words, the model needs to simultaneously discover the objects (or form categories) and learn to perform localization.

As the object size and alignment is not predetermined the complexity of the clustering of the objects grows significantly with the degrees of freedom added by the localization. Analogously, the localization is difficult because during training objects categories are not predetermined. An additional complication is that the objects can overlap, which further increases the difficulty of clustering the objects.

To address these challenges we propose a novel autoencoding neural network architecture, StampNet, that discovers objects and localizes them simultaneously. StampNet has two characterizing features: first, it has a latent space consisting of discrete random variables that encode the cluster assignment and its location. Second, it has a final layer consisting of a fixed number of convolutional filters (*stamps*) that encode the discovered objects. The size and number of these filters determine the maximum size and maximum number of objects to be discovered respectively. We refer to this layer as a *stamp layer* and hence the name of the network StampNet.

This paper is structured as follows: in Section 2, we report on the state-of-the-art on unsupervised object discovery. In Section 3, we introduce the StampNet architecture. In Section 4, we show the results of StampNet on four datasets. The discussion Section 5 closes the paper.

2. RELATED WORKS

Various studies have been performed on topics closely related to unsupervised object discovery. In co-localization and co-segmentation, the goal is to extract the position of common objects between images, using bounding boxes and segmentation respectively. Many of these studies obtain good results, but simplify the problem by assuming a single object class [3–5].

Recent studies have worked on the more difficult problem of multi-class co-localization [6] and co-segmentation [7]. Cho et al. [7] use an off-the-shelf region proposal system to form a set of candidate bounding boxes and match these across images. Wang et al. [6] use functional maps to model partial similarity across images, but they assume that the image set contains two classes of objects or very similar objects.

To the best of our knowledge, only one study has focussed on our task of multi-class object discovery: tackling simultaneous localization and classification without supervision. Murasaki et al. [8] extract deep features using a pre-trained neural network and clusters these features. However, their model is limited to a single single object in each image.

3. STAMPNET ARCHITECTURE

3.1. Architecture overview

In this section, we describe the architecture of StampNet (see sketch in Figure 1). StampNet is an autoencoder, i.e. a neural network which outputs a reconstruction of an input image based on an internal representation of the input. This representation is formed by the encoder under the training constraint of minimising Euclidean distance between the input and the output.

The encoder follows a VGG-like [9] structure with stacks of convolutional layers (with Leaky ReLU activations [10]) followed by a max pooling layer (Batch Normalization [11] and Dropout [12] are used during training).

The decoder, the peculiarity of StampNet, works in two stages: first, it predicts the coordinates and stamp type for each shape in the input image (Selection-and-localization layer); second, through the stamp layer, the selected stamps are placed according to the predicted positions to generate the output image. In the following subsection, we introduce the notation employed and we detail the selection-and-localization and the stamp layers.

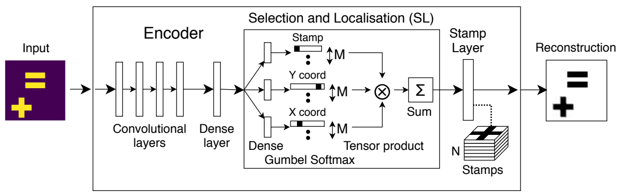


Fig. 1: The StampNet architecture consists of an encoder (convolutional neural network), selection-and-localization layer and the stamp layer. The encoder produces a map from image space to the latent space. The selection-and-localization layer maps the latent representation to an activation map for the stamp layer (Section 3.3). Finally, the stamp layer produces the reconstruction of the image based on the activation map (Section 3.4).

3.2. Notation

We consider an input image of pixel size $\phi_x \times \phi_y$ containing $m \leq M$ shapes indexed by the variable η (i.e. $1 \leq \eta \leq M$, where the upper-bound M to the number of shapes is predefined). We assume each shape to fit within one of N stamps of pixel size $\psi_x \times \psi_y$. The set of stamps is stored in a collection of N convolutional kernels of size $\psi_x \times \psi_y$, that form, as whole, a tensor Ω of size $N \times \psi_x \times \psi_y$. We finally name X and Y the random variables respectively associated to the x and y coordinate of a shape and S the categorical random variable defining the stamp type s. Thus, it holds $0 \leq x, y \leq \phi_x - \psi_x + 1, \phi_y - \psi_y + 1$ and $0 \leq s \leq N - 1$.

3.3. Selection-and-localization layer

The selection-and-localization layer (SL-layer) predicts the coordinates (localization) and selects the clustered object to use (selection). As objects are clustered in the kernel of the stamp layer, these need to be convolved over the predicted location and only the selected object should be convolved. This happens when the output of the SL-layer is a one-hot tensor, (see Equation 2).

Therefore, we want to output a one-hot tensor at the predicted coordinates and predicted stamp. One way to output such tensor is to use a max function. This function, however, results in a gradient of 0 for all non-max values, making it unfitting for backpropagation. We provide a different solution: we model the coordinates and object-selector as categorical distributions. When we sample from these distributions, we get a one-hot tensor with the predicted stamp and coordinates. The *gumbel softmax* [13] (also known as the *concrete distribution* [14]) is a differentiable sampling of a smoothly deformed categorical distribution.

In the SL-layer, we first predict the probability distribution of the x coordinate, y coordinate and the stamp type s. Then, we calculate a sampling using the gumbel softmax function. This function uses temperature τ that allows gradients to flow through during training and gradually anneals the temperature to become a sampling from a categorical distribution [13].

The network combines the three gumbel softmax output by performing, for each of the M shapes, the tensor product of the coordinates and of the stamp selector probability vectors. We obtain M individual selection-and-localization tensors (SL tensors), as they contain information on the position and type of each shape. In formulae, the individual SL tensor reads for all $1 \leq \eta \leq M$:

$$SL^{(\eta)}_{ijk} = P(X^{(\eta)} = i)P(Y^{(\eta)} = j)P(S^{(\eta)} = k) \qquad (1)$$

Once every shape has been selected and localized, we combine them into the global selection-and-localization tensor summing on η:

$$\mathrm{SL}_{ijk} = \sum_{\eta=1}^{M} SL^{(\eta)}_{ijk}.$$

3.4. The stamp layer

The output O of the network, i.e. the reconstruction of the input, is provided by the stamp layer, which performs a convolution operation between the global SL tensor and the stamp tensor Ω. We write $O = \mathrm{SL} * \Omega$, which in components satisfies:

$$O_{i^*j^*} = \sum_{k=1}^{N} \left(\sum_{ij} \mathrm{SL}_{ijk}\, \Omega_{(i^*-i)(j^*-j)k} \right) \tag{2}$$

We constrain the kernel values in Ω to be non-negative and smaller than the maximum image value with training-time clipping. Moreover, should two stamps be partially overlapped, and thus sum their values, the output is further clipped to avoid exceeding the maximum image value.

4. RESULTS

We evaluate the performance of StampNet on several datasets. The first dataset contains five clearly distinguishable shapes, followed by more complex shapes of MNIST. The final dataset contains noisy overhead images of pedestrians. The performance of the network is evaluated on two tasks: (1) discovery and localization of each shape and (2) clustering performance of the extracted stamps. We evaluate the first task with CorLoc [7] and Intersection over Union (IoU) [15]. For the second task, we use clustering purity [16] to measure how well the model can differentiate between classes.

The network has been trained on a train set and evaluated on a separate test set. We use an annealing schedule of $\tau = \max(0.2, 7.0 \cdot \exp(-0.01 \cdot t))$, updated every epoch for the gumbel softmax. For the results, we use a temperature of $\tau = 0.01$ to enforce a one-hot choice.

4.1. Simple Shapes dataset

The Simple Shapes dataset uses five different simple shapes: a 'plus', an 'equal', a rotated 'equal', a 'slash', and a 'triangle'. We randomly place $M = 2$ shapes of size 28×28 on an empty canvas of size 84×84. We consider 50000 generated training samples and 10000 generated test samples. The stamp layer contains $N = 10$ stamps.

In Figure 2a, we report the $N = 10$ stamps learned by the network. We observe that these stamps are nearly identical to the shapes used to generate the dataset. We detail sample predictions outputs in Figure 2b. In these samples, the network localizes overlapping shapes and assigns the correct stamp to each shape.

In Table 1, we quantify the IoU and purity of evaluated over the test set. We achieve high scores in IoU and purity on this dataset. The remaining errors are the result of the network assigning an inaccurate stamp to a shape, which primarily happens when shapes overlap.

4.2. Translated and Cluttered Translated MNIST

We evaluate the performance of our model on two MNIST datasets: (1) Translated MNIST (T-MNIST) and (2) Cluttered Translated MNIST (CT-MNIST) [17]. In T-MNIST, MNIST digits are uniformly placed on an empty canvas. CT-MNIST adds clutter to these images by uniformly placing 8 smaller clutter digits of size 8×8 to add noise to the dataset. We choose in T-MNIST a canvas size of 84×84 and in CT-MNIST a canvas size of 100×100 to best compare our results with existing literature. In both cases, we generate 60000 training samples and 10000 test samples. We test for $M = 1$ and $M = 2$ digits on the canvas with $N = 40$ stamps.

We observe that the network discovers different MNIST digits in Figure 2d and 2f. The stamps learned by the network (Figure 2c and 2e) resemble different digits of MNIST.

The network discovers most MNIST digits, as the CorLoc measures (Table 1) indicate. Even when there are $M = 2$ digits on a cluttered canvas, the network discovers over 95% of the digits. The added clutter results in a slight drop of localization (Table 1) and an increase in purity.

We note the results of others in the case of CT-MNIST ($M = 1$) in Table 2. We observe that without supervision, StampNet performance comes near these supervised alternatives (using purity as the measure for comparison, i.e when all stamps are labelled correctly). Note that these networks only classify a single digit, while our model can discover multiple digits in each image.

Table 1: Experiment results

Dataset	CorLoc	IoU	Purity
Simple Shapes ($M = 1$)	0.9999	0.9718	0.9928
Simple Shapes ($M = 2$)	0.9828	0.9500	0.9564
T-MNIST ($M = 1$)	0.9983	0.8925	0.7891
T-MNIST ($M = 2$)	0.9729	0.8537	0.5277
CT-MNIST ($M = 1$)	0.9972	0.8912	0.8113
CT-MNIST ($M = 2$)	0.9545	0.8394	0.6149
Pedestrian Tracking ($M = 3$)	0.7816[a]	0.6308[a]	0.9597

[a]Calculated for localizing the pedestrian, not the walls.

Table 2: Supervised classification CT-MNIST ($M = 1$)

Network	Accuracy
RAM [17]	0.927
DRAW [18]	0.966
RNN-SPN [19]	0.985
DCN [20]	0.986

4.3. Pedestrian localization in overhead images

Overhead depth maps are an increasingly popular tool to perform high accuracy pedestrian tracking for studying the dynamics of human crowds in real-life venues (e.g. [21]). Overhead depth maps come in form of grey scale images where the value of each pixel encodes its distance from the camera plane. In overhead depth view pedestrians have similar "ovoid" shape, which is different from that of walls, objects

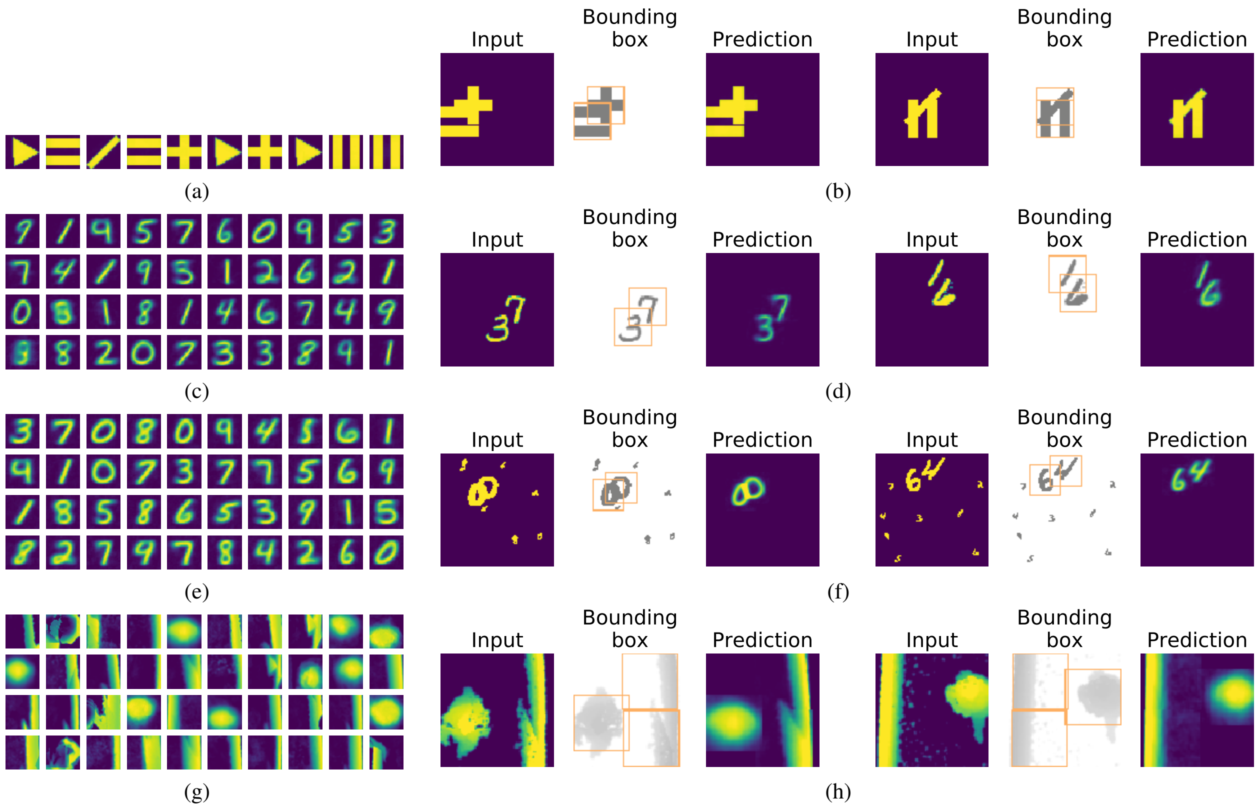


Fig. 2: The results of StampNet on four different datasets: Simple Shapes dataset (a,b), Translated MNIST (c,d), Cluttered Translated MNIST (e,f) and Pedestrian Tracking dataset (g,h). *Left:* (a,c,e,g) stamps learned by our model in the Stamp Layer (see Figure 1) *Right:* (b,d,f,h) samples of each dataset . The network predicts bounding boxes (orange) and places a stamp at these locations to reconstruct the input image as close as possible.

and so on. The characteristics of this dataset make it a very suitable for the StampNet's object discovery capability.

We consider here a reduced depth dataset from the real-life crowd tracking experiment [22], annotated with bounding boxes (image size: 80×80, bounding boxes size 40×40. See sample on the left side of Figure 2h, which displays a pedestrian on the left side and a wall on the right side).

We test StampNet considering $N = 40$ stamps of size 40×40 and evaluate both the accuracy of the localization and the capability of the network to differentiate between pedestrians and the wall on the side.

We illustrate samples of the results in Figure 2h and the measures in Table 1. In both cases, we observe that the network places one stamp on the pedestrian and two stamps on the walls. The stamps extracted by the network (Figure 2g) show that different objects are successfully captured by the network, and, as evidenced by the high clustering purity, the network is capable to differentiate between pedestrians and walls. Furthermore, we obtain a CurLoc value of 0.78 showing that we can localize with reasonable accuracy.

5. CONCLUSION

In this paper, we have introduced StampNet to localize multiple objects from multiple classes in an unsupervised manner. We accomplish this by incorporating the predictions of the bounding boxes into an autoencoder, removing the need of labels. We achieve this by placing the kernel of a convolutional layer on predicted location coordinates (Figure 1).

The results in Figure 2d and 2f show that StampNet is able to detect and localize overlapping MNIST digits without the need for any labels. Furthermore, the network clusters the shapes in the dataset as stamps (Figure 2e). We demonstrate an example of the value of this in an application of pedestrian tracking in overhead images.

Nevertheless, there are limitations to our model. The network is only able to place stamps of a fixed size. Furthermore, as we make use of the kernel of a convolutional layer, the network can only capture simple prototypical shapes.

Future work can look into generating more complex kernels for the stamp layer by making use of information from the input.

6. REFERENCES

[1] Joseph Redmon and Ali Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[3] Gunhee Kim and Antonio Torralba, "Unsupervised detection of regions of interest using iterative link analysis," in *Advances in neural information processing systems*, 2009, pp. 961–969.

[4] Kevin Tang, Armand Joulin, Li-Jia Li, and Li Fei-Fei, "Co-localization in real-world images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1464–1471.

[5] Michael Rubinstein, Armand Joulin, Johannes Kopf, and Ce Liu, "Unsupervised joint object discovery and segmentation in internet images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 1939–1946.

[6] Fan Wang, Qixing Huang, Maks Ovsjanikov, and Leonidas J Guibas, "Unsupervised multi-class joint image segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3142–3149.

[7] Minsu Cho, Suha Kwak, Cordelia Schmid, and Jean Ponce, "Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1201–1210.

[8] Kazuhiko Murasaki, Yukinobu Taniguchi, and Tetsuya Kinebuchi, "Unsupervised multi-class object discovery by spherical clustering of deep features," *ITE Transactions on Media Technology and Applications*, vol. 7, no. 1, pp. 2–10, 2019.

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

[10] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning*, 2013, vol. 30, p. 3.

[11] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[12] 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*, vol. 15, no. 1, pp. 1929–1958, 2014.

[13] Eric Jang, Shixiang Gu, and Ben Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[14] Chris J Maddison, Andriy Mnih, and Yee Whye Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.

[15] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[16] Zhirong Yang, Tele Hao, Onur Dikmen, Xi Chen, and Erkki Oja, "Clustering by nonnegative matrix factorization using graph random walk," in *Advances in Neural Information Processing Systems*, 2012, pp. 1079–1087.

[17] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al., "Recurrent models of visual attention," in *Advances in neural information processing systems*, 2014, pp. 2204–2212.

[18] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[19] Søren Kaae Sønderby, Casper Kaae Sønderby, Lars Maaløe, and Ole Winther, "Recurrent spatial transformer networks," *arXiv preprint arXiv:1509.05329*, 2015.

[20] Amjad Almahairi, Nicolas Ballas, Tim Cooijmans, Yin Zheng, Hugo Larochelle, and Aaron Courville, "Dynamic capacity networks," in *International Conference on Machine Learning*, 2016, pp. 2549–2558.

[21] S. Seer, N. Brändle, and C. Ratti, "Kinects and human kinetics: A new approach for studying pedestrian behavior," *Transportation Research part C: Emerging technologies*, vol. 48, pp. 212–228, 2014.

[22] Alessandro Corbetta, Werner Kroneman, Maurice Donners, Antal Haans, Philip Ross, Marius Trouwborst, Sander Van de Wijdeven, Martijn Hultermans, Dragan Sekulovski, Fedosja van der Heijden, et al., "A large-scale real-life crowd steering experiment via arrow-like stimuli," *arXiv preprint arXiv:1806.09801*, 2018.