Fall 12-5-2019

# Amodal Instance Segmentation and Multi-Object Tracking with Deep Pixel Embedding

Yanfeng Liu
*University of Nebraska - Lincoln*, yanfeng.liu@huskers.unl.edu

# AMODAL INSTANCE SEGMENTATION AND MULTI-OBJECT TRACKING WITH DEEP PIXEL EMBEDDING

by

Yanfeng Liu

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Electrical Engineering

Under the Supervision of Professors Eric T. Psota and Lance C. Pérez

Lincoln, Nebraska

December, 2019

AMODAL INSTANCE SEGMENTATION AND MULTI-OBJECT TRACKING WITH DEEP PIXEL EMBEDDING

Yanfeng Liu, M.S.

University of Nebraska, 2019

Advisor: Eric T. Psota and Lance C. Pérez

This thesis extends upon the representational output of semantic instance segmentation by explicitly including both visible and occluded parts. A fully convolutional network is trained to produce consistent pixel-level embedding across two layers such that, when clustered, the results convey the full spatial extent and depth ordering of each instance. Results demonstrate that the network can accurately estimate complete masks in the presence of occlusion and outperform leading top-down bounding-box approaches.

The model is further extended to produce consistent pixel-level embeddings across two consecutive image frames from a video to simultaneously perform amodal instance segmentation and multi-object tracking. No post-processing trackers or Hungarian Algorithm is needed to perform multi-object tracking. The advantages and disadvantages of such a bounding-box-free approach are studied thoroughly. Experiments show that the proposed method outperforms the state-of-the-art bounding-box based approach on tracking animated moving objects.

## ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my advisors, Dr. Eric Psota and Dr. Lance Pérez for mentoring me and encouraging me during my time at the University of Nebraska-Lincoln. Their invaluable philosophies in research and life have made my graduate experience productive and enjoyable. I would like to extend my sincere thanks to Dr. Stephen Scott for being my committee member. I would like to thank the senior graduate students in my lab, Jay Carlson and Mateusz Mittek, for their encouragement and discussions about research ideas. Finally, I would like to thank my family and friends who have always supported me throughout my time at the University – none of this would have been possible without them.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Computer vision is a field that focuses on a high-level algorithmic understanding of digital images. It strives to intelligently describe everything in an image, such as the localization/classification/action of an object, the relationship between objects, and the 3D structure of the scene. Since videos can be treated as a collection of images, they are included in the consideration of computer vision. With videos, more complex spatiotemporal tasks emerge, such as the re-identification of the same objects across multiple images, multi-object tracking, structure from motion, and odometry estimation.

The field of computer vision has gained increasing popularity after the huge success of deep learning since AlexNet was proposed in 2012 [1]. Fig 1 shows the architecture of AlexNet. Back then GPUs (Graphics Processing Unit) were not optimized for neural networks yet and AlexNet was too big to fit on one GPU so it was distributed to multiple GPUs. The applications of deep learning have now flourished in many domains, such as object detection [4] and tracking [22], semantic segmentation [23], instance segmentation [24], image synthesis [25], and video synthesis [26].

Deep learning is a sub-field of machine learning that utilizes the effective modeling capability of deep neural networks in many tasks, including computer vision, natural language understanding, reinforcement learning, etc. It serves as a powerful tool that automates fea-

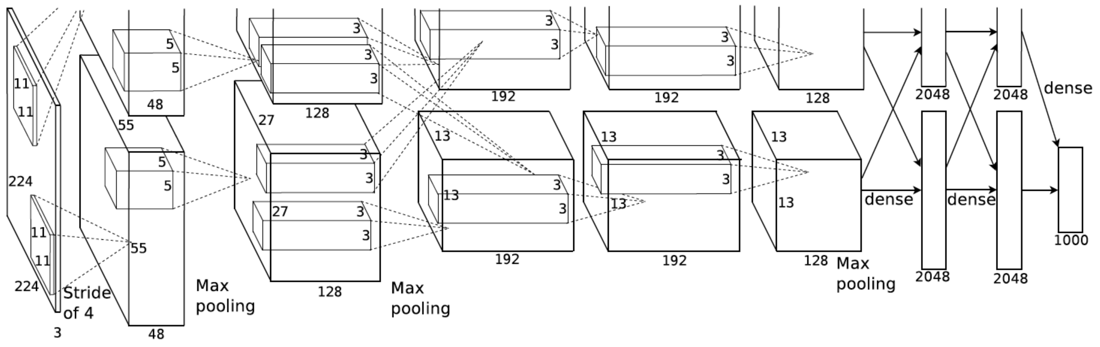


Figure 1: Architecture of AlexNet. Image from [1].

ture extraction and relationship reasoning. The weights are initialized randomly without prior knowledge of the problem. Given the input data, the model predicts the output and corrects its weights against the ground truth through a derivative-based technique called back-propagation. The network is expected to gradually formalize filters that are capable of modelling increasingly complex relationships as the number of layers and weights go up.

Before deep learning, a lot of algorithms were elaborately and manually designed by researchers. It was a very time-consuming process and often limited to specific environments and applications. The features extracted from human-designed algorithms have both the blessing and the curse of interpretability. On one hand, people understand exactly what the features are trying to capture, but, on the other hand, there could be untapped features that are potentially useful but not directly understandable to human beings. In overly complex problems, it is also unclear how to capture the correct features needed for a generalized solution. Deep learning, in contrast, is a one-size-for-all solution that learns features in an iterative process. Researchers now instead focus on the problem formulation and weight optimization aspects of the problem-solving.

In general, there are three classes of learning paradigms in machine learning: supervised learning, semi-supervised learning, and unsupervised learning. In supervised learning, every training example is presented as a pair of input and expected output, or ground truth; in semi-supervised learning, a small set of labelled data and a larger set of unlabelled data are provided; in unsupervised learning, the input is provided without the ground truth, which should be instead inferred from the input itself.

Deep learning thrives in all three learning paradigms. Most vision tasks mentioned above can be formulated as any of the three paradigms. Classic supervised learning is the most obvious route, but it requires a lot of human effort to provide the labelled data. When labelled data is too expensive or impractical to obtain due to the limitations of human labor, it often becomes the bottleneck of the performance. Semi-supervised learning has shown promising results where the model learns useful feature extraction from unlabelled data through techniques such as auto-encoding [27] before fine-tuning on labelled data. Unsupervised learning is the most challenging to design but also the most promising. Once formulated, an unsupervised model can essentially use cheaply obtained training data and benefit from a diverse set of data distributions.

Given the advancement in tools and algorithms in recent years, computer vision researchers have become progressively more interested in a precise, pixel-wise understanding of images. From crude to precise, image understanding tasks can be ordered as: image classification – bounding box localization and classification – semantic segmentation – semantic instance segmentation – panoptic segmentation, each providing more information than the last.

Image classification assigns a class label to an entire image. For example, if an image contains a truck as the salient object, it should be assigned the "truck" label regardless of its exact appearance or background. Figure 2 shows a classic dataset, CIFAR-10 [28], for image classification. In image classification, the challenge is to model the variety of a category of objects. For example, trucks should all have wheels and large space for cargo. In recent years, neural networks have achieved super-human performance (defined roughly as lower than 5.1% top-5 error rate) on certain datasets of image classification [11, 29], such as ImageNet [7].

Bounding box localization and classification, also known as object detection, takes image classification one step further. Its goal is to produce the exact location of the object of interest. By drawing a bounding box around the object, it also opens up the possibility of labelling multiple objects at the same time. For example, in Figure 3, there are multiple human faces in the image. The algorithm draws a bounding box around each face and classifies it as "face." Object detection is an active research area with the support from large datasets such as Microsoft COCO, PASCAL, and Labeled Faces in the Wild [30, 31, 32]. Object detectors generally fall into two categories, one-stage detectors such as [33, 34],

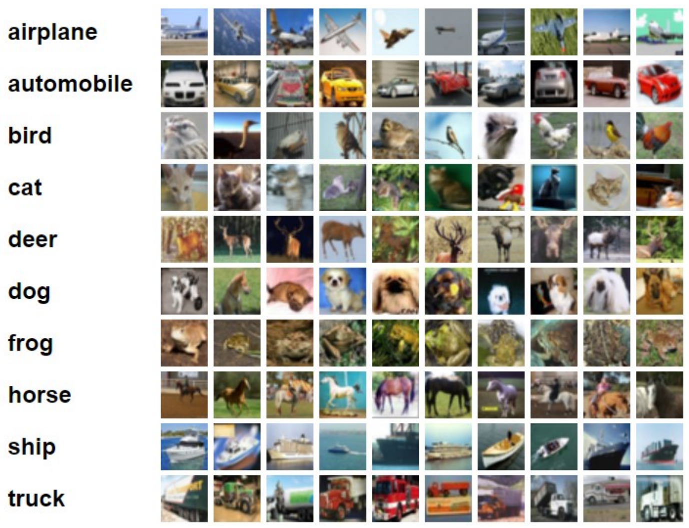


Figure 2: Image classification from CIFAR-10. Image from http://bit.ly/33Mspyw

and two-state detectors such as [35, 36, 37]. The difference is one-stage detectors output a fixed number of detections on a grid while two-stage detectors propose an arbitrary number bounding boxes and filter them in post-processing.

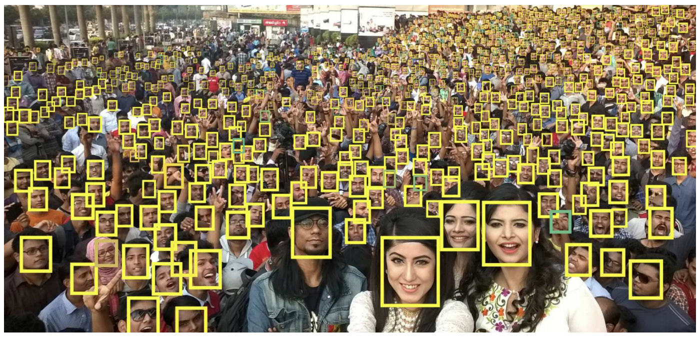


Figure 3: Face detection on multiple people [2]

Semantic segmentation takes image understanding to another level by assigning a class label to each pixel, achieving localization and classification at the finest granularity. For example, in Figure 4, the algorithm labels each pixel that belongs to the "person" class as red, drivable road as purple, sidewalk as pink, road signs as yellow, and street lights as orange. The actual colors do not matter as long as they are consistent for the same class. Modern semantic segmentation methods mostly rely on fully-convolutional neural networks [15, 14]. Datasets like CityScape and KITTI [3, 38] provide large amounts of driver-view training data, mostly for autonomous driving research purposes.

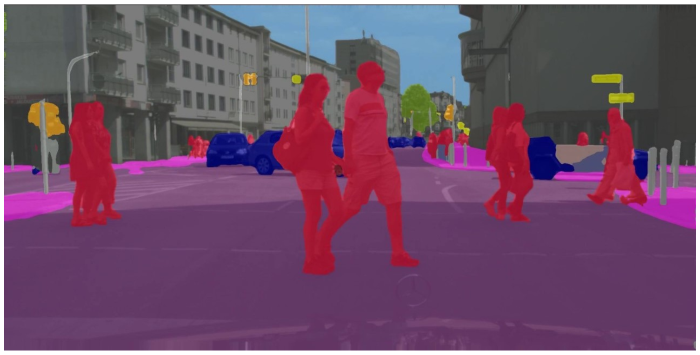


Figure 4: Semantic segmentation on a street view in the CityScape dataset [3]

Instance segmentation looks beyond the semantic classification of the pixel and assigns a unique identity to each visible object within the same class. For example, in Figure 5, the pixels that belong to chairs are correctly identified and separated from all other objects in the scene, plus each chair has a distinct identity (illustrated with a unique color) consistent within its boundary. Some methods build on top of object detection and estimates the object mask within the bounding box [4, 39, 40], while the others take a bottom-up approach that assigns properties to pixels and cluster them later into instance masks [41, 42, 43].

Panoptic segmentation combines semantic segmentation and instance segmentation into one task. The countable things are treated as instances where as the uncountable “stuff” is treated as semantic segmentation. It aims to assign a class and/or identity to every pixel in the image. It was first proposed by Alexander Kirillov et al. in 2018 [44], which also

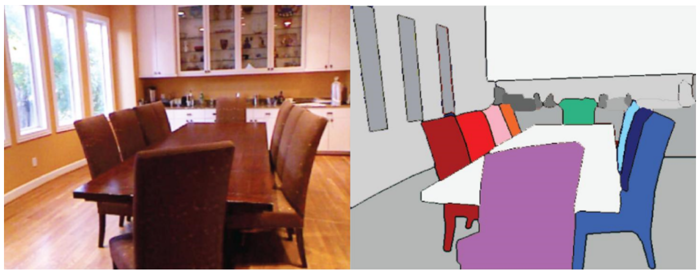


Figure 5: Semantic instance segmentation on chairs. Image from `http://bit.ly/2DRMRTW`

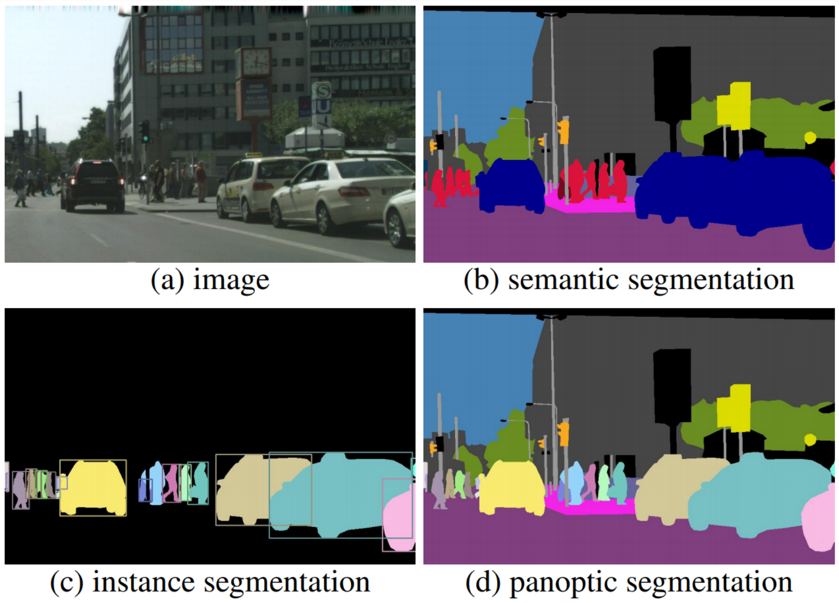

(a) image
(b) semantic segmentation
(c) instance segmentation
(d) panoptic segmentation

Figure 6: A side-by-side comparison of different segmentation tasks. Image from `http://bit.ly/34Xjuvm`

introduced a new Microsoft COCO Panoptic dataset for the task. Others quickly followed up on improvements [45, 46, 47, 48, 49]. Figure 6 compares the relationship between semantic segmentation, instance segmentation, and panoptic segmentation.

Even though panoptic segmentation provides classification and identity for each pixel in the image, it is by no means the end of all pixel-level tasks. So far the focus has been on the visible portion of the image, but human beings have much stronger perceptive capability in the sense that they are also able to infer what is occluded. We don't simply assume an object is cut in half just because we can only see half of it. We imagine where the rest of it is and what it looks like based on what we see. Amodal segmentation fulfills this need by explicitly estimating the complete mask based on the visible part. This is particularly useful in tasks such as multi-object detection and tracking because, ideally, objects should not be lost when briefly occluded.

Instance segmentation, either modal or amodal, traditionally has two general branches of techniques, top-down and bottom-up, as mentioned previously. The top-down approach, much like a two-state object detector, relies heavily on generating bounding boxes around objects. The instance mask is then estimated strictly within the boundary of the box, as shown in Figure 7.

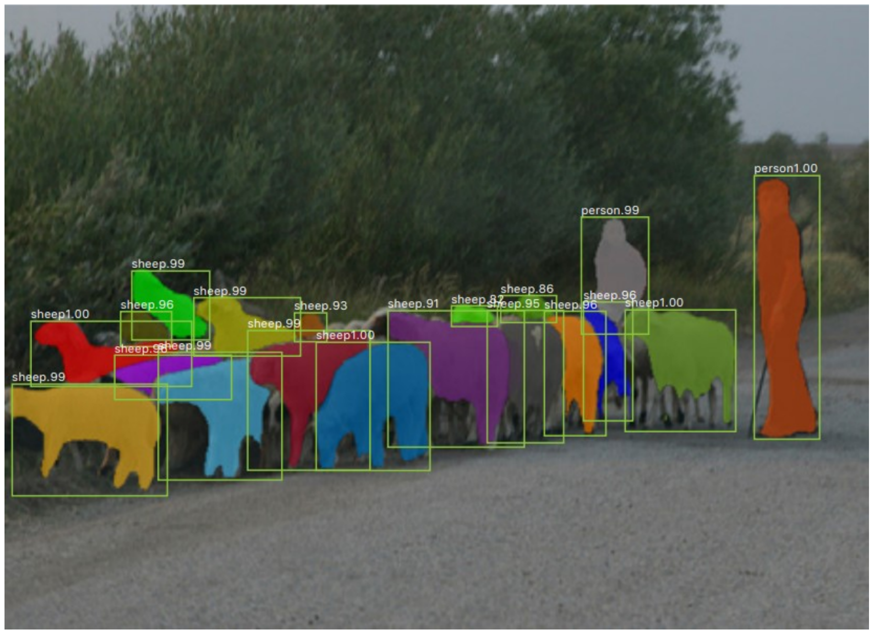


Figure 7: Sample instance segmentation result from Mask R-CNN [4]

In cluttered scenes where a lot of objects are overlapping, it becomes challenging to generate bounding boxes for each object due to non-maximum suppression (NMS). NMS is a necessary step to filter out overlapping bounding boxes. It achieves this filtering typically by calculating the intersection over union (IOU) between candidate boxes. IOU is designed to measure the extent of overlapping between two shapes. NMS sorts all bounding boxes by their confidence scores. When two boxes have an IOU over a pre-determined threshold, NMS discards the one with a lower confidence score and keeps the other. Figure 8 illustrates the definition of IOU.

Current bounding box proposal methods typically generate many more boxes than the actual number of objects, in the hopes that a post-processing algorithm will only keep the candidates that have both the highest confidence and the best overall fit for each object. Figure 9 demonstrates the effect of such post-processing. The problem with this approach is that, when objects are cluttered and overlapping, a lot of valid bounding boxes are treated

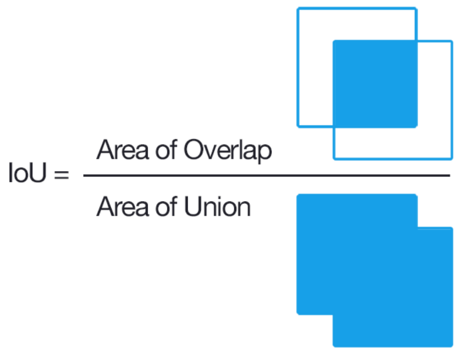


Figure 8: Intersection over union. Image from `bit.ly/33RC9Yq`

as false-positives and discarded. Figure 10 shows such a failure case. Several modifications such as Soft NMS and Softer NMS are proposed to mitigate this issue [50, 51]. They improve benchmark scores but do not address the fundamental problem of bounding boxes – bounding boxes can repeatedly cover the same region multiple times and they intentionally do so to systematically cover important visual clues. Methods like [35] even explicitly allow ~2k bounding box proposals in the intermediate step to cover the image as much as possible. As long as bounding boxes are still part of the algorithm, this problem will persist.

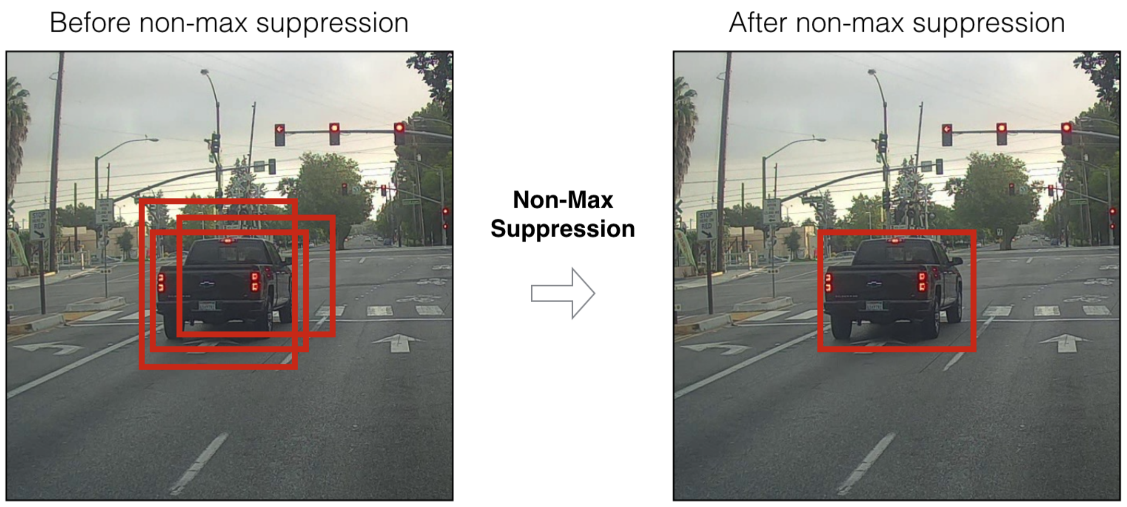


Figure 9: The effect of non-max suppression. Image from `bit.ly/2YjNe38`

Another issue with bounding boxes is that, when the algorithm tries to generate a mask

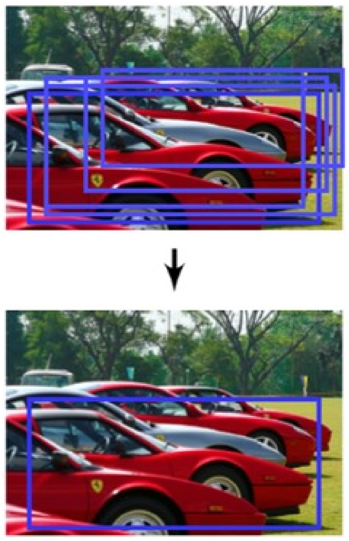


Figure 10: A failure case of non-max suppression. Image from `bit.ly/355TkXi`

for the salient object of the detected class and two objects are both visible in the bounding box, it can be confusing to decide which one is the targeted object. Technically, both of them are, so the algorithm ends up attempting to cover both objects with a strange shape. Examples of this type of error are shown in later chapters.

Other than the tasks mentioned above, there are other areas in computer vision that take advantage of videos instead of static images. Videos consist of multiple frames of highly correlated content. The fact that neighboring frames are taken within temporal and spatial proximity provides extra information.

Optical flow is usually calculated to visualize pixel movement between consecutive video frames and help track object movement, as shown in Figure 11. Traditionally this is done through a differential method called the Lucas-Kanade method [52, 53]. Modern methods rely heavily on deep learning to estimate optical flow through either supervised learning [54, 55] or unsupervised learning [56]. Optical flow can also be jointly learned with other tasks such as depth estimation [57] and segmentation [58].

Human key-points estimation is usually done jointly with human detection [4] to provide extra information to be used in pose estimation [59, 60] or fun applications such as synthesized dancing videos [61] or motion capture. Figure 12 demonstrates the task of

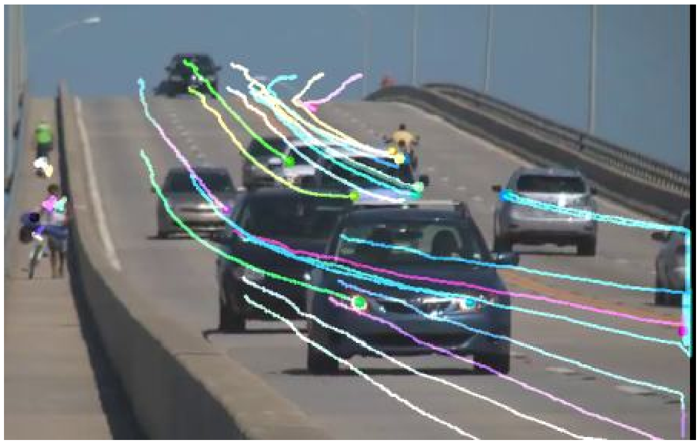


Figure 11: Optical flow, tracked through multiple frames and visualized on the last frame. Image from `http://bit.ly/2PjjDTk`

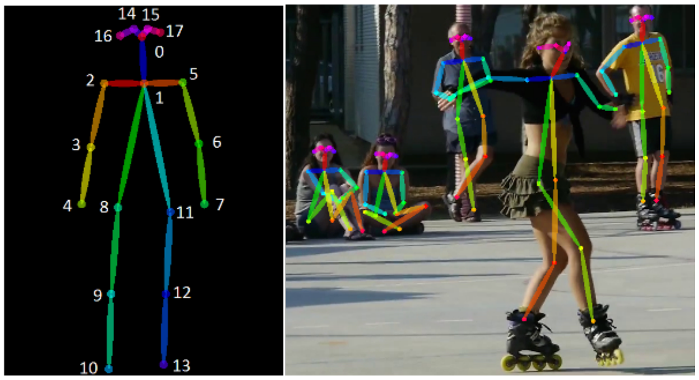


Figure 12: Human key-points estimation [5]

human key-points estimation.

Three dimensional reconstruction is another area that leverages the power of videos. The concept is that a camera moves continuously in a 3D space, taking 2D images periodically. The appearances of objects warp based on their shape and distant to the camera. Structure from motion extracts feature points from frames and re-projects them into the 3D scene in order to reproduce the structure [62]. Stereo matching tackles the problems differently. Stereo matching assumes that two cameras are side-by-side and calculates the depth at every pixel by measuring how much each pair of corresponding pixels shifts between the left and right image [63]. Figure 13 and 14 illustrate the two methods.

Among current research areas mentioned above, amodal instance segmentation and multi-object tracking are the two areas that strive for the highest level of scene understanding. One describes the complete shapes of objects regardless of their visibility, the other preserves the identities of multiple objects, given their detections, across video frames. This thesis focuses on these two areas and proposes a unified method for both. The proposed method is intuitive yet powerful. It is easy to construct, with very few parameters to manually tune. Specifically, the contributions of this thesis include:

- Investigate the limitations of bounding boxes in highly crowded scenes for both instance segmentation and tracking;
- Propose a bounding-box-free method for amodal instance segmentation based on pixel embeddings;
- Extend the proposed method into a multi-object tracking method without motion modelling or complex identity matching post-processing;
- Compare the behaviors and trade-off between the proposed models and state-of-the-art models in the literature to highlight current issues in segmentation and tracking.

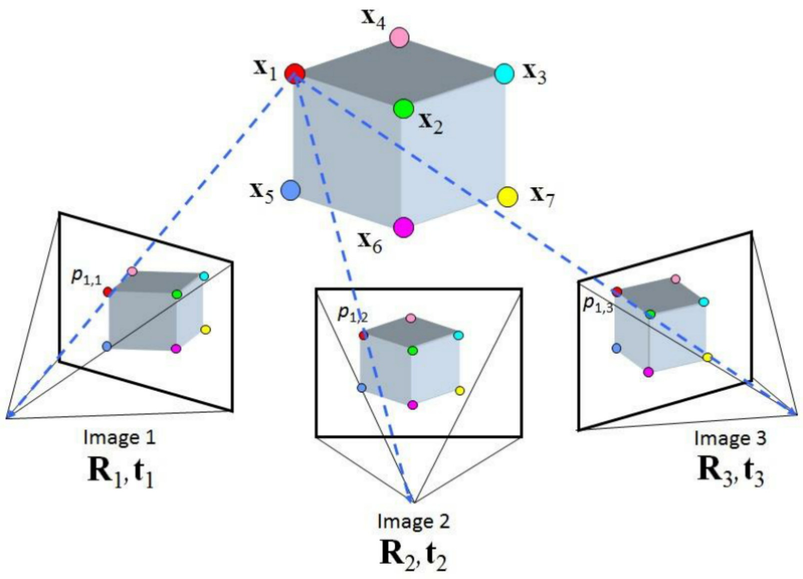


Figure 13: Structure from motion. Image from `http://bit.ly/2YjQsUk`

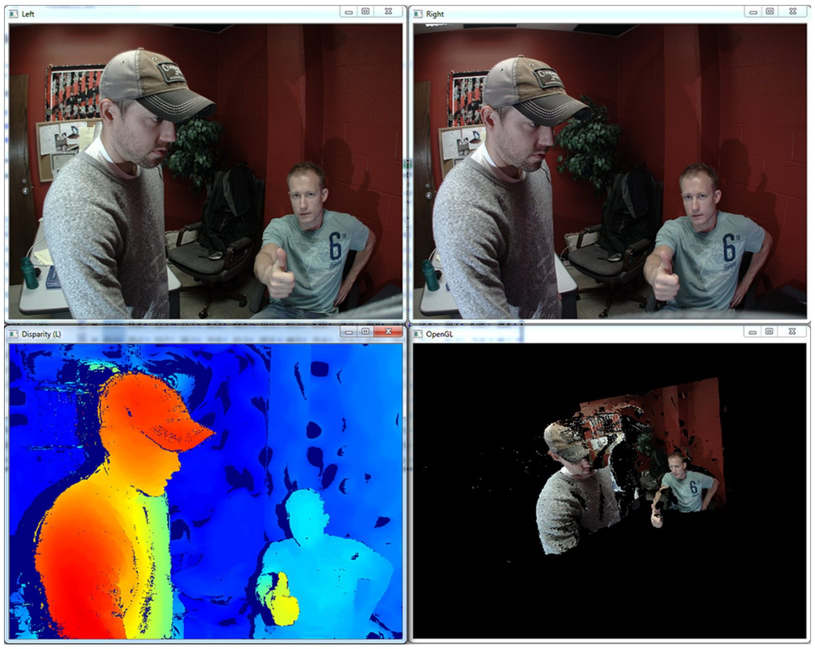


Figure 14: Stereo Matching. Top: left and right view. Bottom: color-coded depth map and 3D reconstructed scene. Image from `http://bit.ly/2LoAfrS`

# 2 Related Work

## 2.1 Deep Learning and Computer Vision

### 2.1.1 Components of Deep Learning

Even though the new wave of deep learning interest only started in 2010s, the core components and fundamentals of deep learning are not young. This section reviews the development of deep learning from the last century and introduces the core concepts and techniques along with their history.

The foundational concept of perceptron [64] was first introduced in 1957. Frank Rosenblatt proposed it as a binary classification algorithm that has the following rule:

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0, \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $w$ and $x$ are vectors of real values of the same length, and $b$ is the bias. Today, the form of $w \cdot x + b$ is still a fundamental computing rule in a fully connected network. The difference is that neural networks now are much more flexible than simple binary classification. Each $f(x)$ and the non-linear activation function after it are together considered a "neuron". A generic architecture for fully connected neural networks is shown in Figure 15. In this case, the network has one input layer, two hidden layers, and an output layer. The dimensions of inputs and outputs are pre-defined by the problem, but there is a lot of flexibility for how many hidden layers and how many neurons per layer a network can have. Generally, the capacity of network increases as the network gets more complicated.

What is not shown in Figure 15 is the activation function. There is an activation function for every layer of the neural network. The function can have many designs to choose from, but it needs to be differentiable and nonlinear. The differentiability allows backpropagation, thus training, of the neural network. The nonlinearity provides the network with the ability

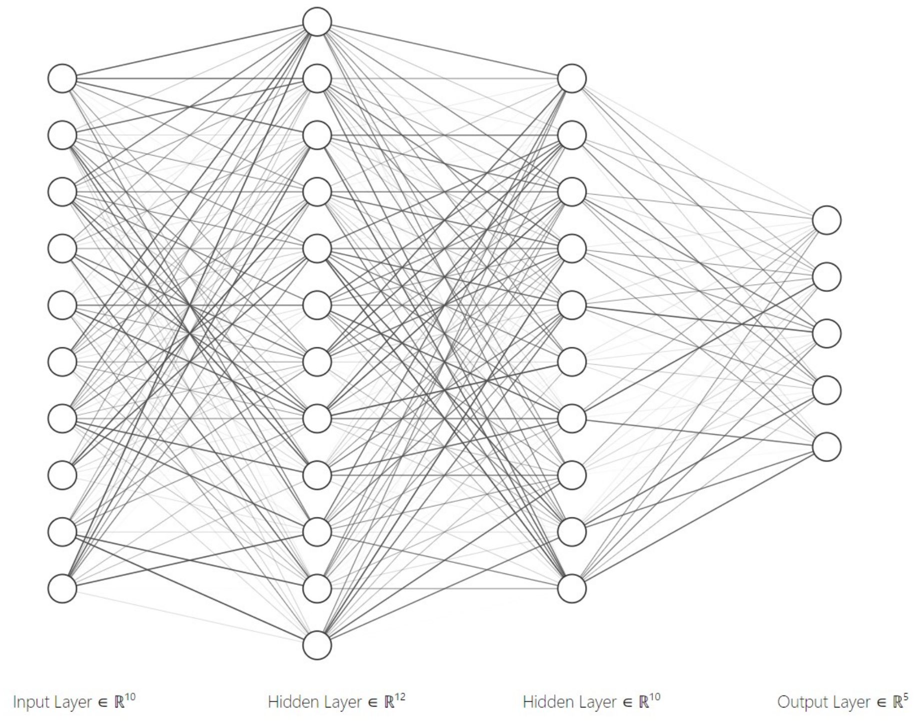


Figure 15: A generic architecture of fully connected networks. Image rendered by `http://bit.ly/2s28oa8`

to model nonlinear relationships in data, which is what makes neural networks powerful. If there is no nonlinear activation function, then the entire fully connected network can be reduced to one matrix multiplication. In other words, there is no reason to have multiple layers without nonlinear activation functions.

Consider a two-layer fully connected network without activation functions. It can be represented as

$$f(x) = A_1(A_0x + b_0) + b_1 ,$$

which can be simplified as

$$f(x) = A_1A_0x + A_1b_0 + b_1 .$$

Let $A = A_1 A_0$ and $b = A_1 b_0 + b_1$, then we have

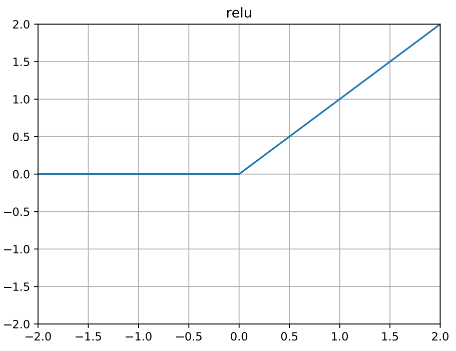
$$f(x) = Ax + b\,,$$

which is the form of a one-layer fully connected network.

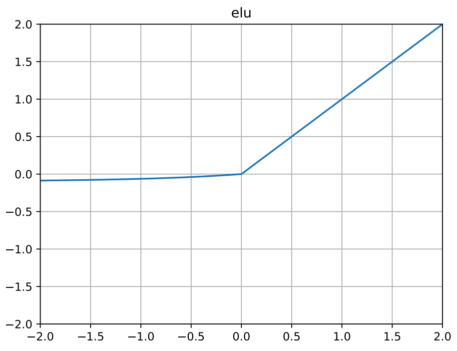
Common choices of nonlinear activation functions include ReLU (Rectified Linear Unit) [65], LeakyReLU [66], ELU (Exponential Linear Unit) [67], Sigmoid, Hyperbolic Tangent (tanh), and softplus [68]. Figure 16 shows each function listed above. Their equations are as the following:

$$\begin{aligned}
\text{ReLU}(x) &= \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise} \end{cases}, \\
\text{ELU}(x) &= \begin{cases} x & \text{if } x > 0, \\ \alpha(e^x - 1) & \text{otherwise} \end{cases}, \\
\text{LeakyReLU}(x) &= \begin{cases} x & \text{if } x > 0, \\ \alpha x & \text{otherwise} \end{cases}, \\
\text{Sigmoid}(x) &= \frac{1}{1 + e^( - x)}\,, \\
\text{tanh}(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}}\,, \\
\text{softplu}s(x) &= \text{ln}(1 + e^x)\,.
\end{aligned}$$

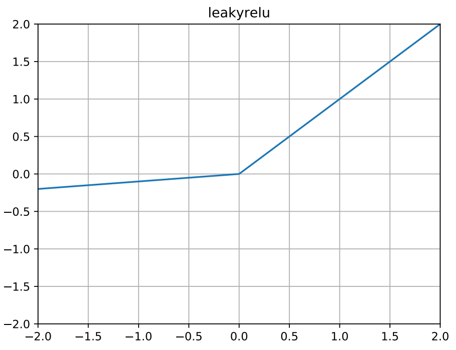
Fully connected networks work well with one dimensional data, sometimes two dimensional data if it is flattened to be one-dimensional but, in general, fully connected networks struggle to work well on two dimensions and beyond due to its input format limitations. By design, the input of a fully connected network is a one-dimensional vector of fixed size. Inputs of different shapes have to be cropped or padded, depending on whether they have more numbers or fewer. In contrast, convolutional neural networks solve this problem by



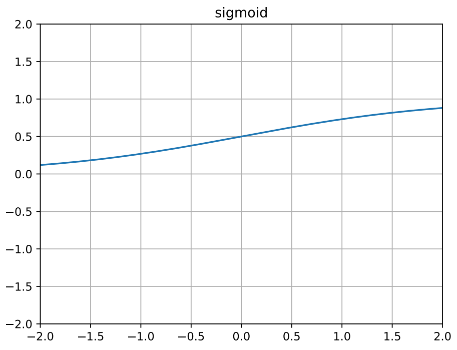
(a)



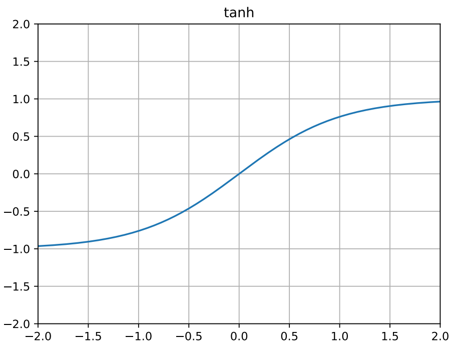
(b)



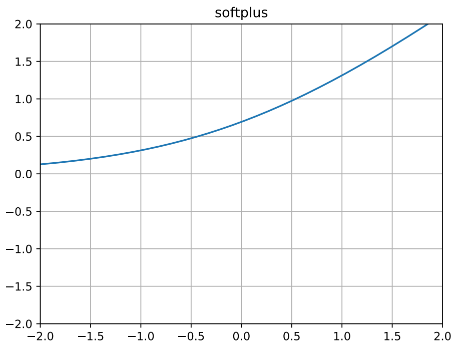
(c)



(d)



(e)



(f)

Figure 16: Plot of various activation functions. Image rendered by the author.

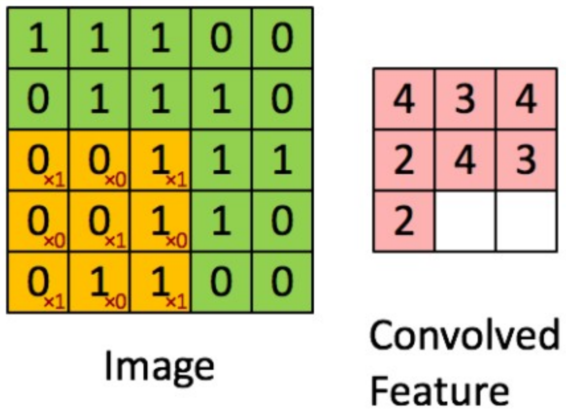


Figure 17: Process of a convolutional filter convolving with the input. Note how the output location corresponds to the location of the input. Image from `http://bit.ly/2PkeSsz`

utilizing small convolutional filters to convolve with the input. The input can be one, two, three, or higher dimensional. The most commonly used are two and three dimensional convolution filters. They are usually of size $(3 \times 3)$, $(5 \times 5)$, or $(7 \times 7)$. Bigger sizes are theoretically possible but rarely used. The size of each spatial dimension also needs to be an odd number in order to preserve the shape of the input. Figure 17 demonstrates the process of a two dimensional convolution. At the borders of the input image, the designer of the algorithm has a choice between zero-padding the border or leaving it as-is. Zero-padding will cause the output of convolution to have the same shape as the input, whereas leaving it as-is will cause the output to have a smaller shape. In the case of Figure 17, there is no zero-padding, so the input image changes from $5 \times 5$ to $3 \times 3$

The convolutional layer has many variants. One of them is the atrous convolutional layer. It allows the layer to have explicit control over the output resolution by changing how “sparse” the atrous convolution filters are. Another variant is the convolution transpose layer. It increases the output resolution compared to the input in contrast to decreasing it in normal convolution layers. This is achieved by flipping the order of the input and the filter.

To train a neural network, whether it is convolutional or fully connected, a core technique called Stochastic Gradient Descent (SGD) was introduced. In 1951 and 1952, two

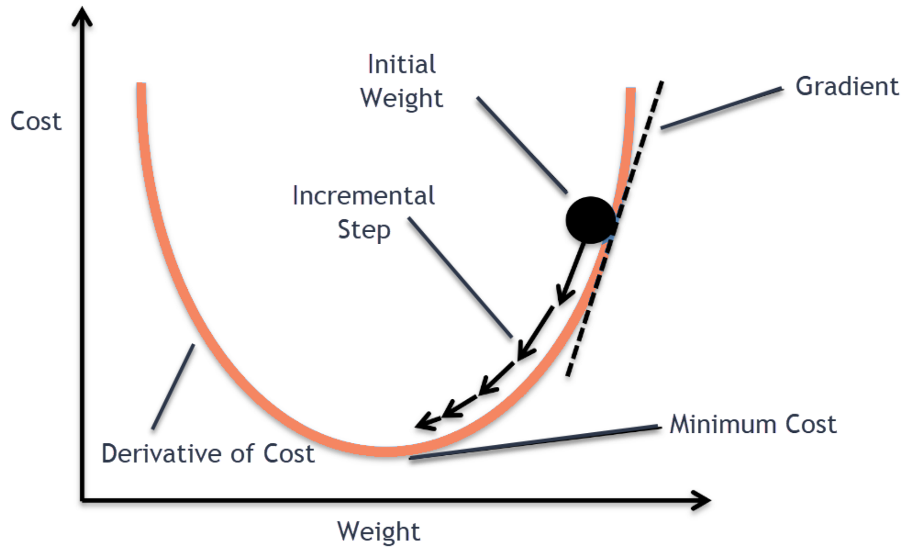


Figure 18: Procedure of Stochastic Gradient Descent. Image from `http://bit.ly/2PkLSRP`

papers were published that were considered the foundational works of SGD. It is an iterative optimization algorithm to minimize/maximize the value of a function. It works in the following stages: (1) start at a random location on the function value plane with random parameters; (2) find the gradient with respect to the parameters to decide the direction for its next step; (3) update by taking a step of predefined step size in that direction; (4) repeat step (2) and (3). Figure 18 shows the stages of SGD.

SGD requires the gradients of the function to be known. It is done through a technique called automatic differentiation (autodiff). In 1970, the earliest form of general automatic differentiation was developed by Seppo Linnainmaa [69, 70], even though it was not called "backpropagation" at the time. Modern deep learning frameworks use the reverse mode automatic differentiation for calculating gradients. The procedure is best shown through a toy example.

Assume we want to calculate the gradient for

$$f(x_1, x_2, x_3) = \alpha x_1 x_2 + \beta x_3^2$$

with respect to $x_1, x_2, x_3$ where $\alpha$ and $\beta$ are constants. In other words, we are interested in $\frac{df}{dx_1}, \frac{df}{dx_2}, \frac{df}{dx_3}$. The algorithm breaks $f$ down to basic units whose the gradient functions are known and clearly defined. In this case,

$$\begin{aligned} y_1 &= x_1 x_2\,, \\ y_2 &= x_3^2\,, \\ y_3 &= \alpha y_1\,, \\ y_4 &= \beta y_2\,, \\ y_5 &= y_3 + y_4\,. \end{aligned}$$

It then uses the commonly known Chain Rule to iteratively find the gradients with respect to the target parameters. As a reminder, the Chain Rule states that, if there are three variables $a, b, c$ where $a$ depends $b$ and $b$ depends on $c$, and we are interested in the derivative of $a$ with respect to $c$, then it can be calculated as

$$\frac{da}{dc} = \frac{da}{db} \cdot \frac{db}{dc}\,.$$

In the case of the example, gradients for each intermediate step can be easily obtained as the following:

$$\frac{dy_5}{dy_3} = 1, \frac{dy_5}{dy_4} = 1, \frac{dy_3}{dy_1} = \alpha, \frac{dy_4}{dy_2} = \beta, \frac{dy_1}{dx_1} = x_2, \frac{dy_1}{dx_2} = x_1, \frac{dy_2}{dx_3} = 2x_3\,.$$

Since $f = y_5$, we can then $\frac{df}{dx_1}, \frac{df}{dx_2}, \frac{df}{dx_2}$ as:

$$\begin{aligned}
\frac{df}{dx_1} &= \frac{df}{dy_3} \cdot \frac{dy_3}{dy_1} \cdot \frac{dy_1}{dx_1} = 1 \cdot \alpha \cdot x_2 \,, \\
\frac{df}{dx_2} &= \frac{df}{dy_3} \cdot \frac{dy_3}{dy_1} \cdot \frac{dy_1}{dx_2} = 1 \cdot \alpha \cdot x_1 \,, \\
\frac{df}{dx_3} &= \frac{df}{dy_4} \cdot \frac{dy_4}{dy_2} \cdot \frac{dy_2}{dx_3} = 1 \cdot \beta \cdot 2x_3 \,.
\end{aligned}$$

The exact values of these gradients can then be easily calculated when constant $\alpha, \beta$ and the values of $x_1, x_2, x_3$ are plugged in.

Many optimization algorithms proposed over the years. Among them, the most famous ones include Adamax [71], Adam [71, 72], RMSProp, AdaGrad [73], Adadelta [74], and Nadam [75]. They propose extensions on top of the original SGD by introducing ideas such as momentum, sliding window average of gradients, and adaptive learning rate. The choice of optimizer, together with learning rate, can be a deciding factor in machine learning. Therefore it should be carefully tested through a search algorithm.

In 1997, long-short term memory (LSTM) recurrent neural network was invented by Sepp Hochreiter and Jürgen Schmidhuber [76]. LSTM is still one of the core modules in modern research of recurrent neural networks (RNN). To be able to appreciate LSTM, one first needs to understand the vanilla RNN design. RNNs are the backbone of natural language processing. They have the architectural advantage of being able to handle sequential data naturally. Language is one of the two manifestations of intelligence, together with visual understanding. Therefore, being able to understand human languages, translating them to one another, rephrasing with similar meanings attracted huge research interest in the community. Previously neural networks have been a state-less model, meaning that the content and ordering of previous inputs and outputs have no effect on the next inputs or outputs. Being able to retain information in a way that models human memory is desirable in sequential data processing. The recurrent neural network was designed to solve this

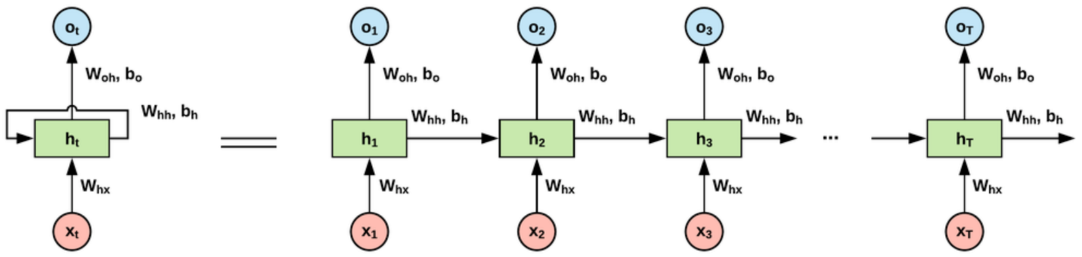


Figure 19: A generic architecture of recurrent neural networks. Image from http://bit.ly/2rUkou5

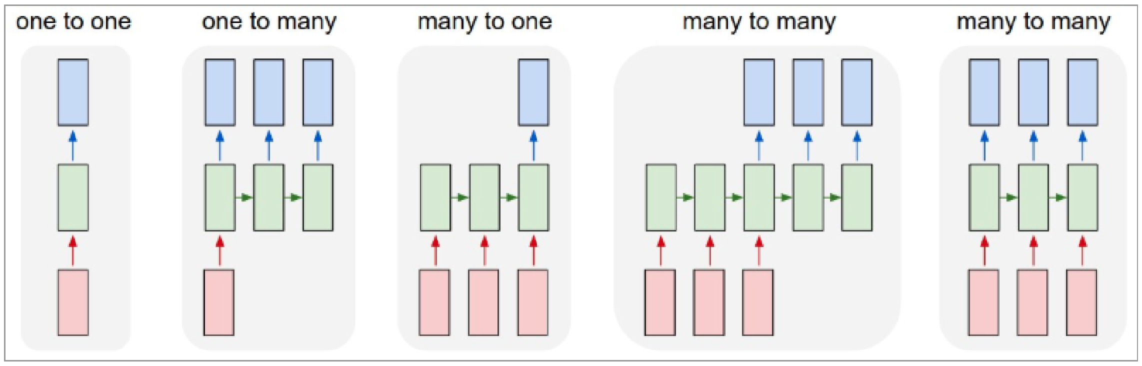


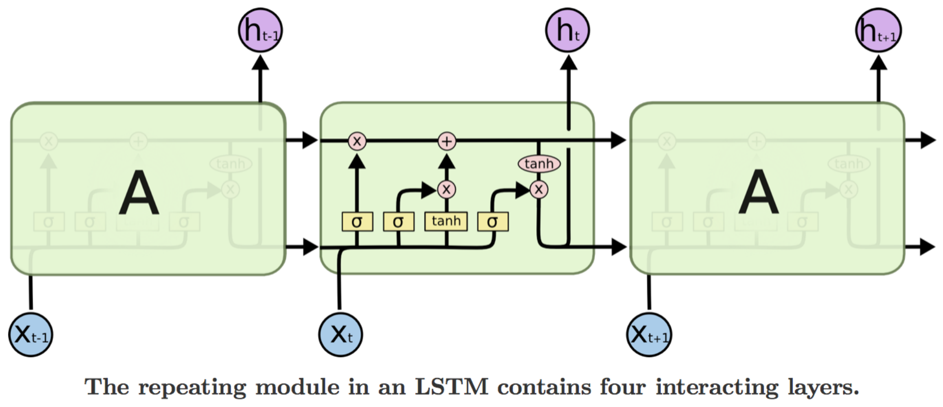
Figure 20: Flexible input/output for recurrent neural networks. Image from http://bit.ly/2LqwsKw

particular problem. Figure 19 shows a generic architecture of a RNN.

In the figure, there is a hidden state, denoted as $h_t$ for time step $t$. The hidden state is affected by the previous inputs so that it produces different outputs of the same input $x_{t+1}$ in time step $t + 1$ given different previous inputs from time step 0 to $t$. The right hand side of the figure is the setup unrolled in time. This architecture is very flexible in its input/output arrangement. It can be one-to-one, one-to-many, many-to-one, many-to-many, with arbitrary input and output length. Figure 20 illustrates these input/output arrangements.

The computation of RNN is as follows. Let $x_t$ be input at time $t$, $h_t$ be hidden state at time $t$. The internal state is updated using both the previous hidden state and the input:

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t) \, ,$$



h t-1
h t
h t+1
A
A
tanh
σ
σ
tanh
σ
X t-1
X t
X t+1
The repeating module in an LSTM contains four interacting layers.

Figure 21: Architecture of LSTM recurrent neural networks. Image from `http://bit.ly/2rUkou5`

and then the output is:

$$y = W_{hy} h_t \, ,$$

so one RNN unit keeps three parameter matrices: $W_{hh}, W_{xh}, W_{hy}$.

The design works better than fully connected networks on sequential data, but it still has its problems. Researchers found that this design does not handle long-term dependencies in sequential data, such as language. In the original design, $h$ was responsible for both (1) remembering particular information at a certain point in time and (2) carrying long term persistent information. LSTM mitigates this problem by explicitly introducing two memory states, cell state $C$ and hidden state $h$, and extra gates to decide whether some information should be remembered or forgotten. Figure 21 illustrates the internal structure of a LSTM unit.

The first Sigma gate is the forget gate $f_t$

$$f_t = Sigmoid(W_f[h_{t-1}, x_t] + b_f)$$

where $[,]$ means concatenation.

The second and third gate work together to decide what which values in the input $x$

should be used to update the hidden state and how:

$$\begin{aligned} i_t &= Sigmoid(W_i[h_{t-1}, x_t] + b_i \\ \widetilde{C}_t &= tanh(W_C[h_{t-1}, x_t] + b_C)\,. \end{aligned}$$

The cell state $C_t$ is then updated by

$$C_t = f_t C_{t-1} + i_t \widetilde{C}_t\,.$$

The last part of LSTM is to produce hidden state $h_t$:

$$\begin{aligned} o_t &= Sigmoid(W_o[h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t)\,. \end{aligned}$$

The cell state $C$ and hidden state $h$ can be think of the long and short term memory, respectively. $C$ is preserved between time steps and only updated when necessary. The update contains multiplication and addition, which help both gradient backpropagation and preserving information. $h$ is derived from $C$ at each time step and may be adjusted drastically at any step.

### 2.1.2 Major Datasets

In 1998, Yann LeCun and his team released MNIST [6], a dataset of $32 \times 32$ greyscale images of hand-written digits from 0 to 9. It is commonly known today as a quick and easy-to-learn dataset for neural network sanity check. Figure 22 shows samples from the MNIST dataset.

In 2010, ImageNet was launched by Fei Fei Li's team. With more than 14 million images from 20000 classes, it was the biggest dataset of image classification at the time,

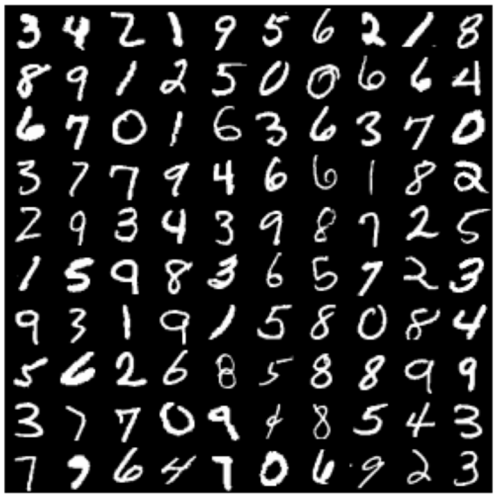


Figure 22: Sample data from MNIST dataset [6]

Figure 23: Sample data from ImageNet dataset [7]

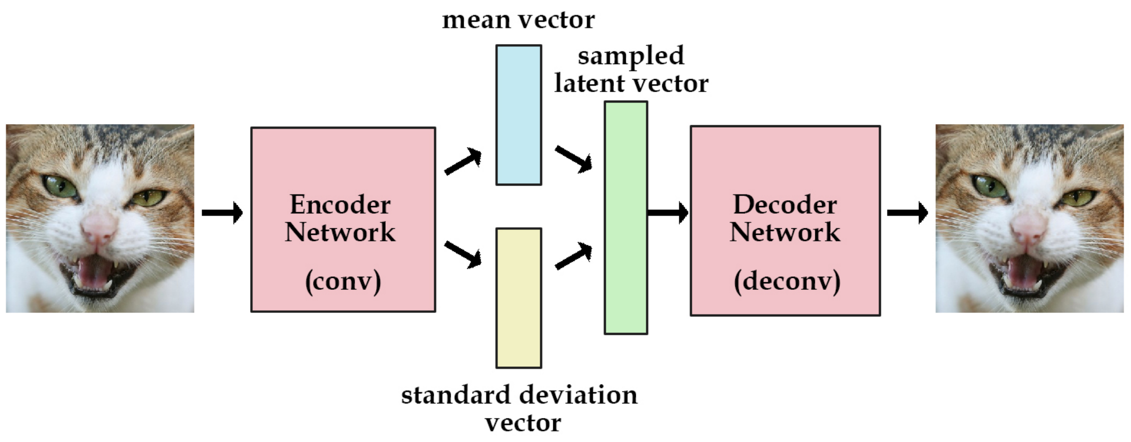


Figure 24: Workflow diagram for VAE. Image from `http://bit.ly/2PkLS4b`

and it inspired and challenged many researchers to focus on training on large amounts of data and modeling diverse environments. Pre-training on ImageNet is still a common technique for many computer vision tasks. Figure 23 shows samples from the ImageNet dataset.

### 2.1.3 Generative Models

In 1985, the autoencoder [77] was invented by Rumelhart et al. to utilize neural networks for self-supervised representational learning. The goal of an autoencoder is to take data $x$ as input and produce output $x_0$ as output such that $x = x_0$. The loss function can be as simple as mean squared error (MSE). The network is forced to learn internal representations of the data. If the intermediate representation is forced to be lower dimensional than the data, then the network needs to learn efficient and robust representations. The encoder and decoder can then be used as data compression tools and the latent space representation can be seen as the encoded data.

In 2013, Variational Autoencoder (VAE) [27] was introduced by Kingma et al.. It is based on autoencoder but the model is meant to be generative. In other words, the output does not have to rely on a similar input. To do that, VAE puts constraints on the latent space that it needs to be close to a Gaussian distribution, regardless of the original data distribution. It does not matter if the data to be generated is cat images or stock prices. The encoder outputs a mean vector and a variance vector to form an $N$-dimensional Gaussian distribution. The decoder then samples from this distribution and tries to reconstruct the input. Figure 24 illustrates this workflow.

Note that the sample is random therefore not guaranteed to be the same even when the input is the same. This forces the entire Gaussian distribution to be representative of the original data. When a VAE is successfully trained, any sample from the distribution should produce something that looks realistic. A property of such a network is that by gradually changing the sample point, the output also changes gradually, meaning that there are no sudden jumps. Figure 25 illustrates this continuous representation learned by VAEs on MNIST. In the middle row, as the two dimensional noise changes along their axes, the reconstructed image also continuously changes from 6 to 2, then to 0, 4, 9, and 7. Note how each number is realistic and slightly different from its neighbors.

To train a VAE, the loss function also needs to be adjusted from that of autoencoders. A simple reconstruction loss is no longer good enough. There also needs to be a distribution loss so that the distribution of the internal representation is close to the true prior distribution, which is assumed to be Gaussian. Let $z$ be the hidden variable that generates observed data $x$, and let $p(z|x)$ represents the true distribution of $z$ given $x$. We approximate $p(z|x)$ with a trainable distribution $q(z|x)$. The exact loss function is then formulated as:

$$E_{q(z|x)} logp(x|z) - KL(q(z|x) \| p(z|x)) \, , \tag{2}$$

where the first term measures reconstruction loss, and the second term measures difference between the estimated distribution and true distribution through KL divergence.

The implementation of the sampling layer requires a reparameterization trick. A normal sampling layer does not allow gradients to backpropagate due to its randomness. The trick

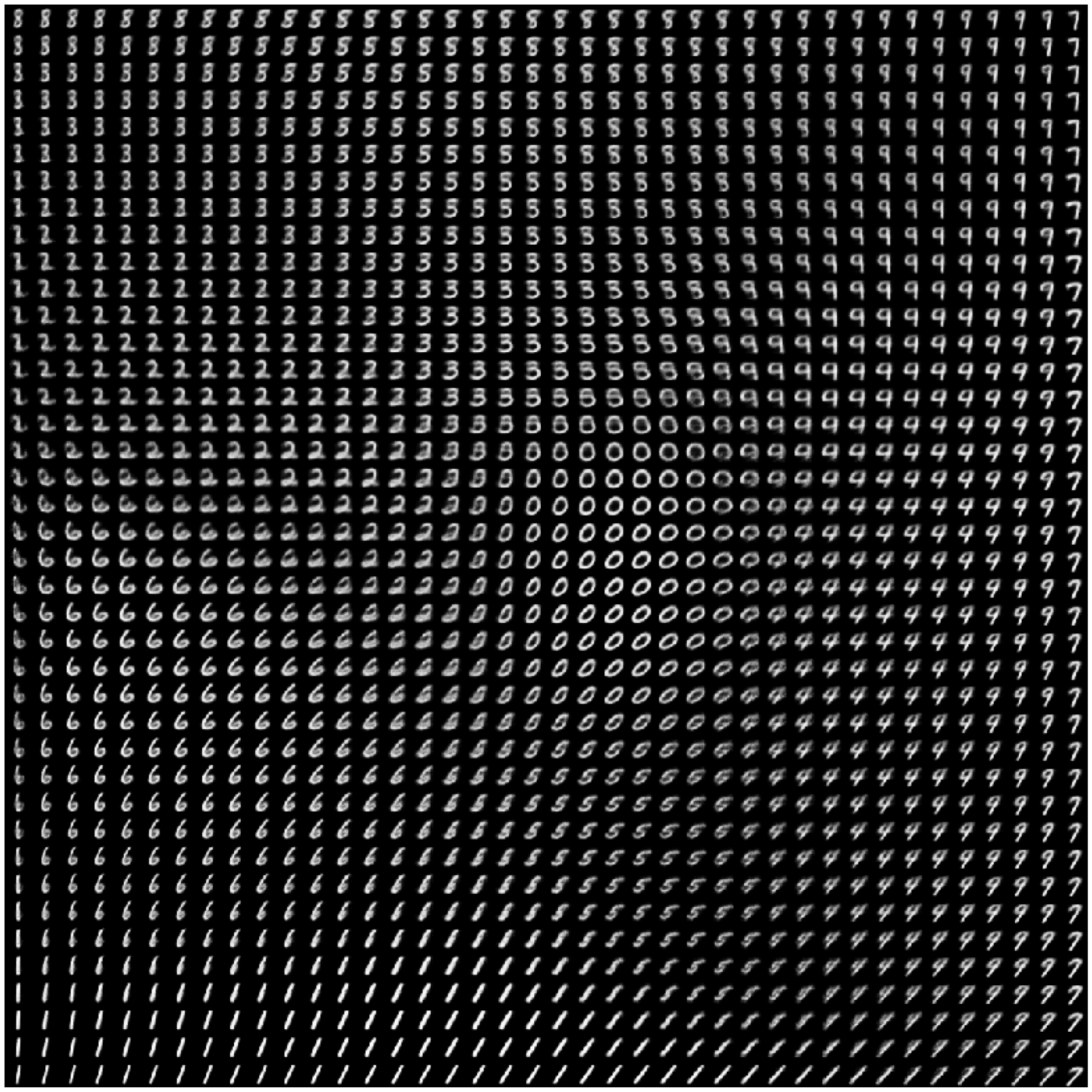


Figure 25: Results of continuous sampling from VAE. Image from `http://bit.ly/2YkLeaQ`

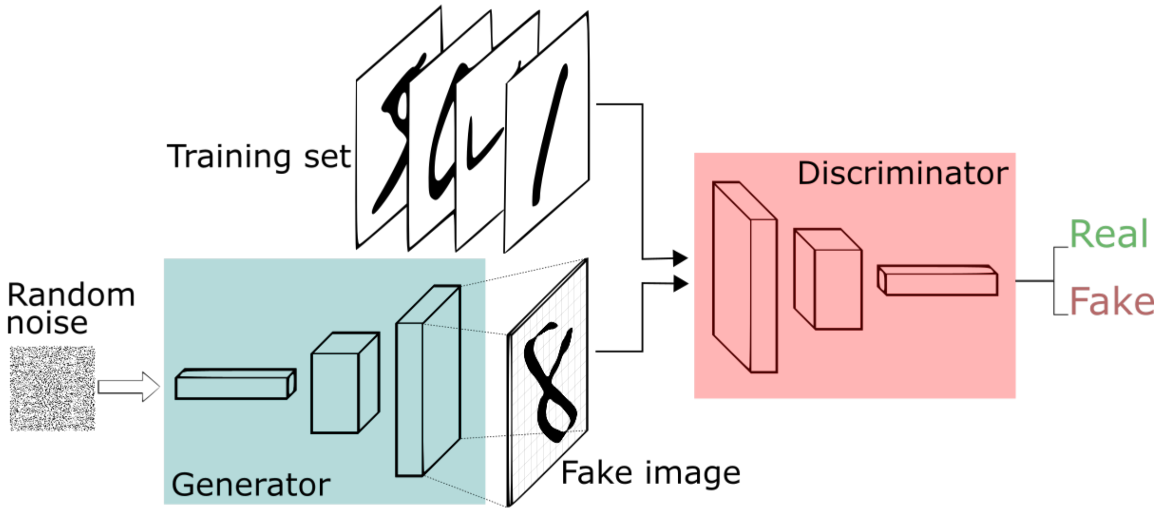


Figure 26: Workflow diagram for GAN. Image from `http://bit.ly/2YnloCZ`

is to set up the previous layer to explicitly produce a mean $\mu$ and a variance $\sigma$. They are then combined together through

$$x = \mu + \epsilon \cdot \sigma$$

where $\epsilon$ is a randomly sampled Gaussian noise with 0 mean and unit variance. This way the Gaussian sampling layer allows gradients to flow back through $\mu$ and $\sigma$, which tie the layers together.

In 2014, Goodfellow et al. invented Generative Adversarial Network (GAN) [78]. It is considered one of the most influential inventions in deep learning, as it inspired countless research ideas and displayed the powerful creativity in neural networks. GANs are similiar to VAEs in the way that they are both generative models. The idea is to set up two networks, one serves the role of discriminator $D$, the other serves the role of generator $G$. The generator generates target output from a sample from a noise distribution, and the discriminator decides whether it is real or fake. This workflow is illustrated in Figure 26.

By providing desired outcome as real data and treating the generator's outcome as fake data, the discriminator gets progressively better at separating those two sets. The discriminator in turn forces the generator to come up with progressively realistic imitation of the real data. The desired end goal is to have a generator that perfectly fools the discriminator by producing results no different than the provided set. In [78], the generator tries to minimize the loss function while the discriminator tries to maximize it. Let $x$ be real data and $z$ be sampled noise from the distribution. The loss function is then defined as:

$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))] \,. \tag{3}$$

In training, there is a trick that the authors recommended. In the GAN paper the authors mentioned that the minimax loss can make the network stuck in early stages, so they recommend changing the loss function for the generator from minimizing

$$log(1 - D(G(z)))$$

to maximizing

$$log(D(G(Z))) \,.$$

Later in the famous Wasserstein GAN paper [79], Arjovsky et al. propose to modify the loss function so that $D$ no longer performs binary classification of “real” vs “fake”. Instead, $D$ outputs a real value number per image so that the number for real data is bigger than for fake data. This new metric is inspired by the Earth Mover Distance. This loosens up the restriction that $D$ has to give a probability. As a result, equation 3 becomes less confusing and training is more stable. $D$ now maximizes

$$D(x) - D(G(z))$$

where as $G$ maximizes

$$D(G(z))$$

Many applications and improvements of GANs were proposed after its invention. GauGAN [8] converts simple doodling into realistic beautiful pictures, as shown in Figure 27.

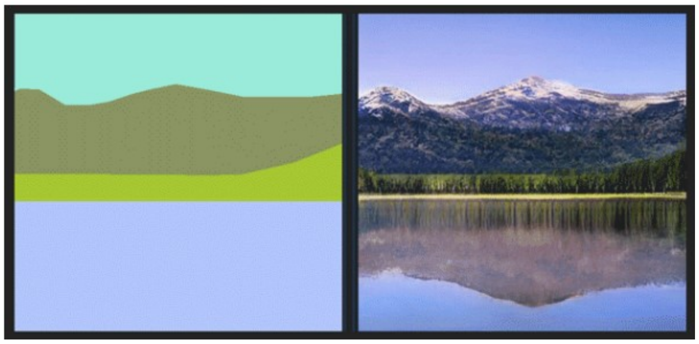


Figure 27: GauGAN. Note how the reflection matches the shape of the mountain and the color of the sky [8]

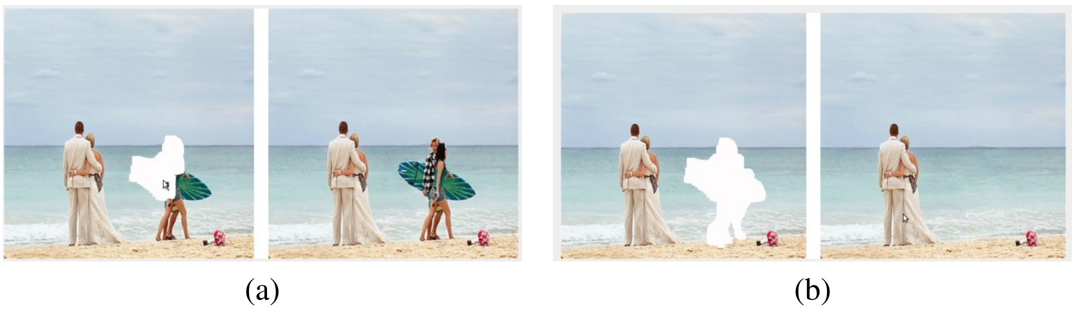


Figure 28: Image In-Painting. (a) shows the processing of masking void areas; (b) shows the completed mask and repaired image [9]

Image in-painting [9] completes pictures with user-defined void areas in them, as shown in Figure 28. BigGAN [10] achieves ultra-realistic image generation with GAN, as shown in Figure 29.

### 2.1.4 Common Architectures

In 2015, Residual Neural Network (ResNet) [11] was proposed by Kaiming He et al. It introduced the novel idea of skip connections between layers to facilitate gradient flow and mitigate the vanishing gradients problem when training deep neural networks. With the help of the skip connections, the authors were able to train a network as deep as one thousand layers. ResNet was the first neural network to achieve 3.7% error rate on ImageNet, better than the human performance of 5%. Figure 30 illustrates the skip connections in ResNet.

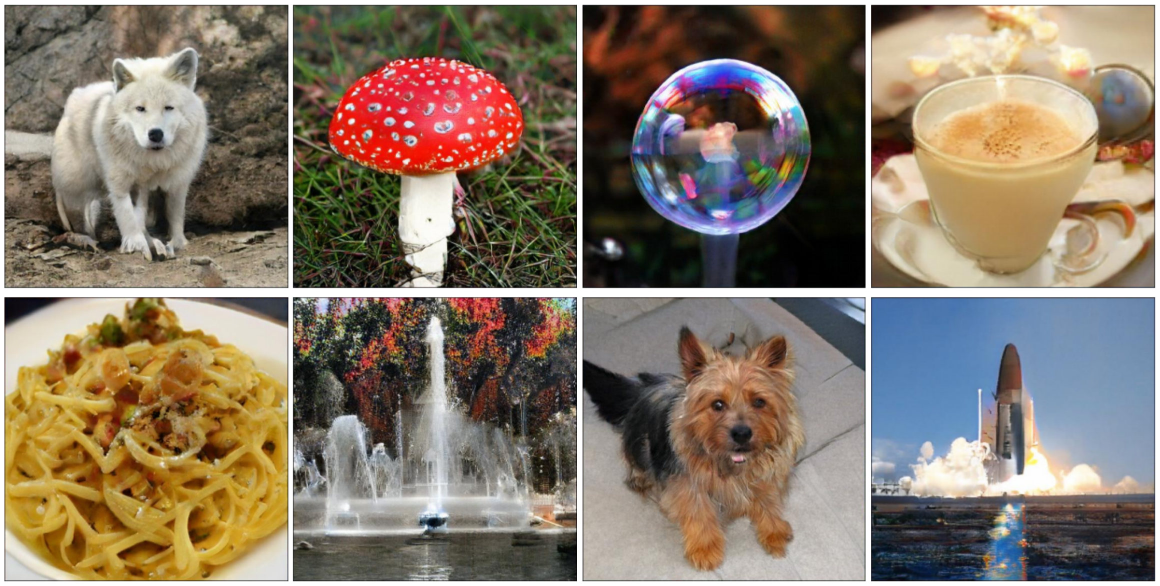


Figure 29: Realistic images generated by BigGAN [10]. Yes they are all fake.

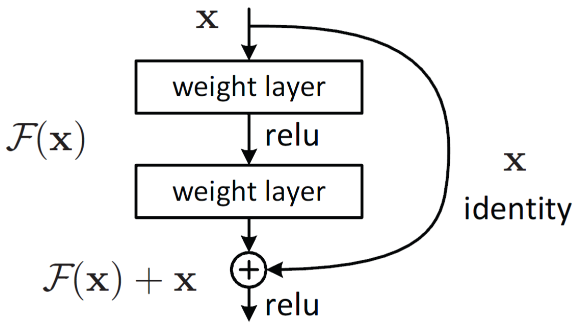


Figure 30: Connection in ResNet [11]

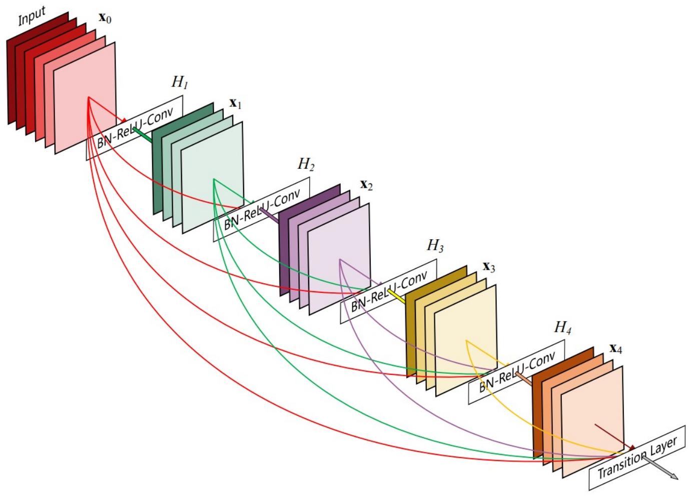


Figure 31: Connection in DenseNet [12]

The idea of using shortcut connections to facilitate gradient flows and boost performance were further explored in 2017 and 2019. Huang et al. [12] proposed to connect every convolutional layer with every other convolutional layer in the same "dense block" to form dense skip connections, as shown in Figure 31. Xie et al. took the idea to the extreme and randomly wired networks in many possible ways [13], as shown in Figure 32. Both show improvements in performance compared to sequentially layered networks.

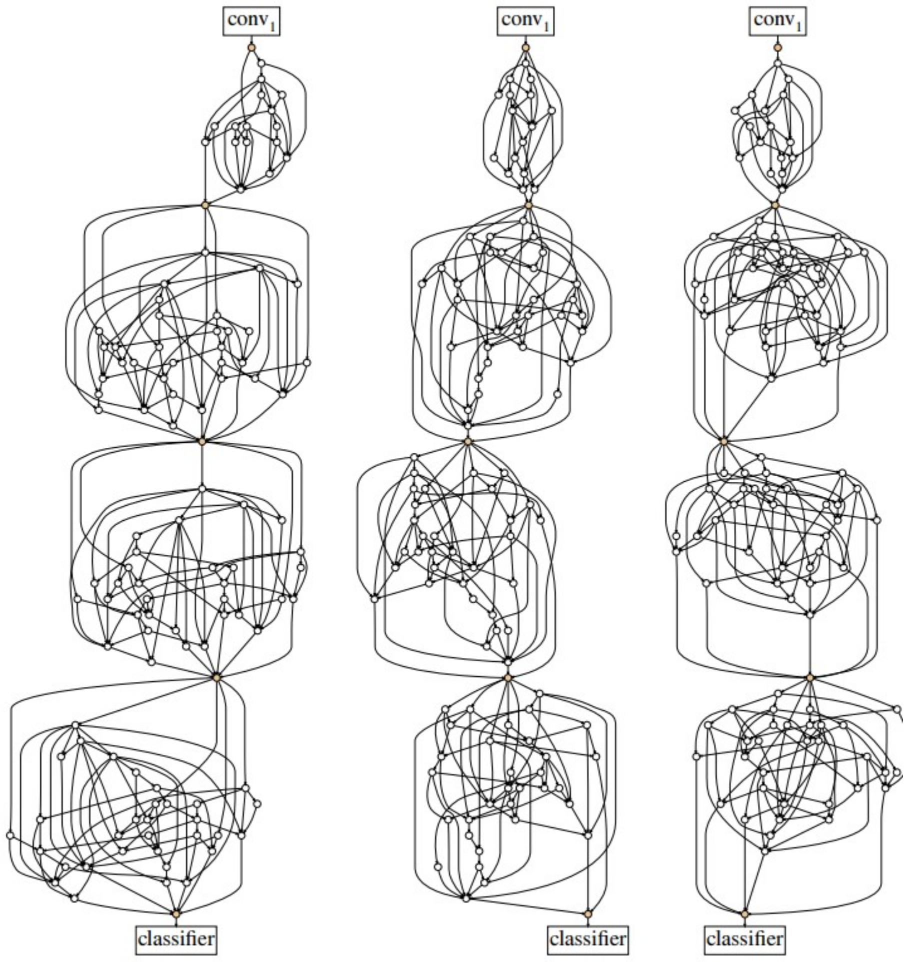


Figure 32: Connection in randomly wired neural networks [13]

## 2.2 Segmentation

Semantic segmentation allows systems to interpret image content in both the spatial and categorical domain. Semantic segmentation separates pixels of one class from the rest of the image. It is the first step towards segmentation level image understanding. When an image contains multiple disjoint segments of the same category, the segments can easily be separated into unique instances.

The research progress in getting high-quality semantic segmentation results inspired many architectures and ideas. SegNet [14] proposes max-unpooling layer to perform up-sampling the same way downsampling is performed at the previous max-pooling layer. This is a novel way to recover information. Figure 33 shows the architecture of SegNet. Long et al. propose fully convolutional network (FCN) [15] to perform semantic segmentation on arbitrary-size image by taking away the fully connected layers that are traditionally at the end of the network for classification. Figure 34 shows the architecture of FCN.

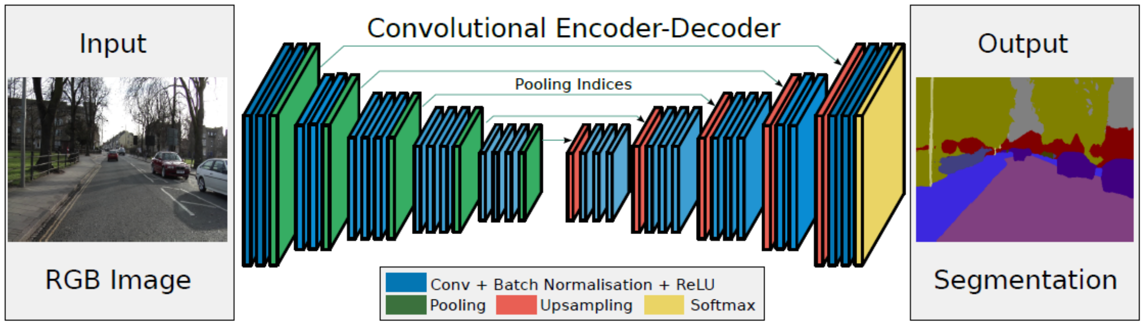


Figure 33: Architecture of SegNet [14]

The DeepLab family of DeepLabV1, DeepLabV2, DeepLabV3, DeepLabV3+ [80, 81, 82, 23] of semantic segmentation models propose several novel ideas and improvements over the existing methods. Among the techniques used and popularized are atrous convolution, fully connected conditional random field (CRF), and atrous spatial pyramid pooling.

Atrous convolution works by introducing gaps to the traditional convolution so that the receptive field of a convolution filter can be manually controlled without increasing size or parameters. Atrous spatial pyramid pooling processes input at different atrous dilation rate

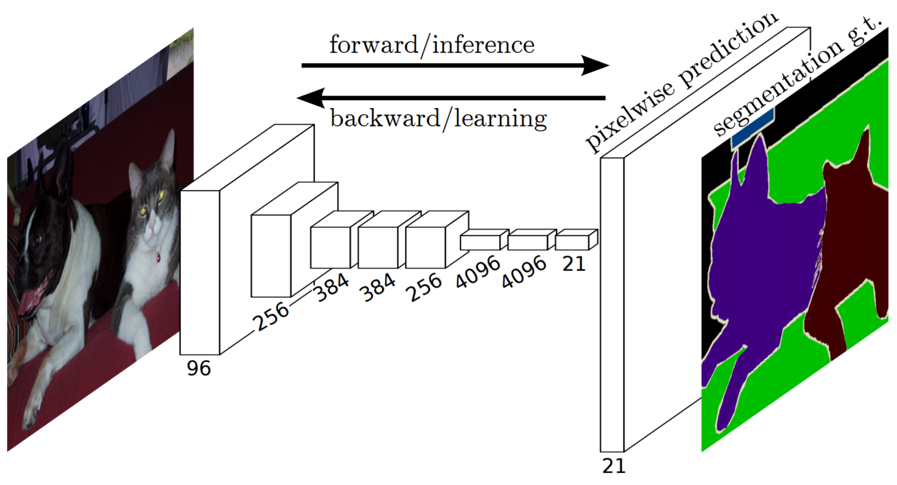


Figure 34: Architecture of FCN [15]

and saves the output features at different scales. This helps detecting objects of different sizes. CRF is part of the post-processing step, thus making the method not end-to-end trainable. CRF is later dropped in DeepLabV3 and DeepLabV3+.

In reality, objects in scenes are rarely separated out perfectly without interference from other objects and/or the environment. Usually, the result of semantic segmentation can only be interpreted as a collection of ambiguous, inseparable blobs. Figure 35 (a, b, c) illustrates the limitations of semantic segmentation in cluttered scenes. With only pixels of the same class highlighted, it is impossible to separate out individual pedestrians. The best one can do is connected components, but it fails as soon as two people share border.

Instance segmentation extends semantic segmentation by distinguishing between objects of the same class. Figure 35 (d) illustrates the result of instance segmentation. Each person has a unique color/identity assigned to him/her. This way, different instances of the same class can be distinctly separated.

However, as some people or objects move in front of others, they unavoidably occlude part of others' appearance. Therefore, each object's full spatial occupancy and depth ordering — two properties that humans instinctively estimate — are not represented in the output. Full spatial occupancy means the entire mask of the object, regardless of how much of it is occluded by others in front of it. Depth ordering means how it ranks in terms of the

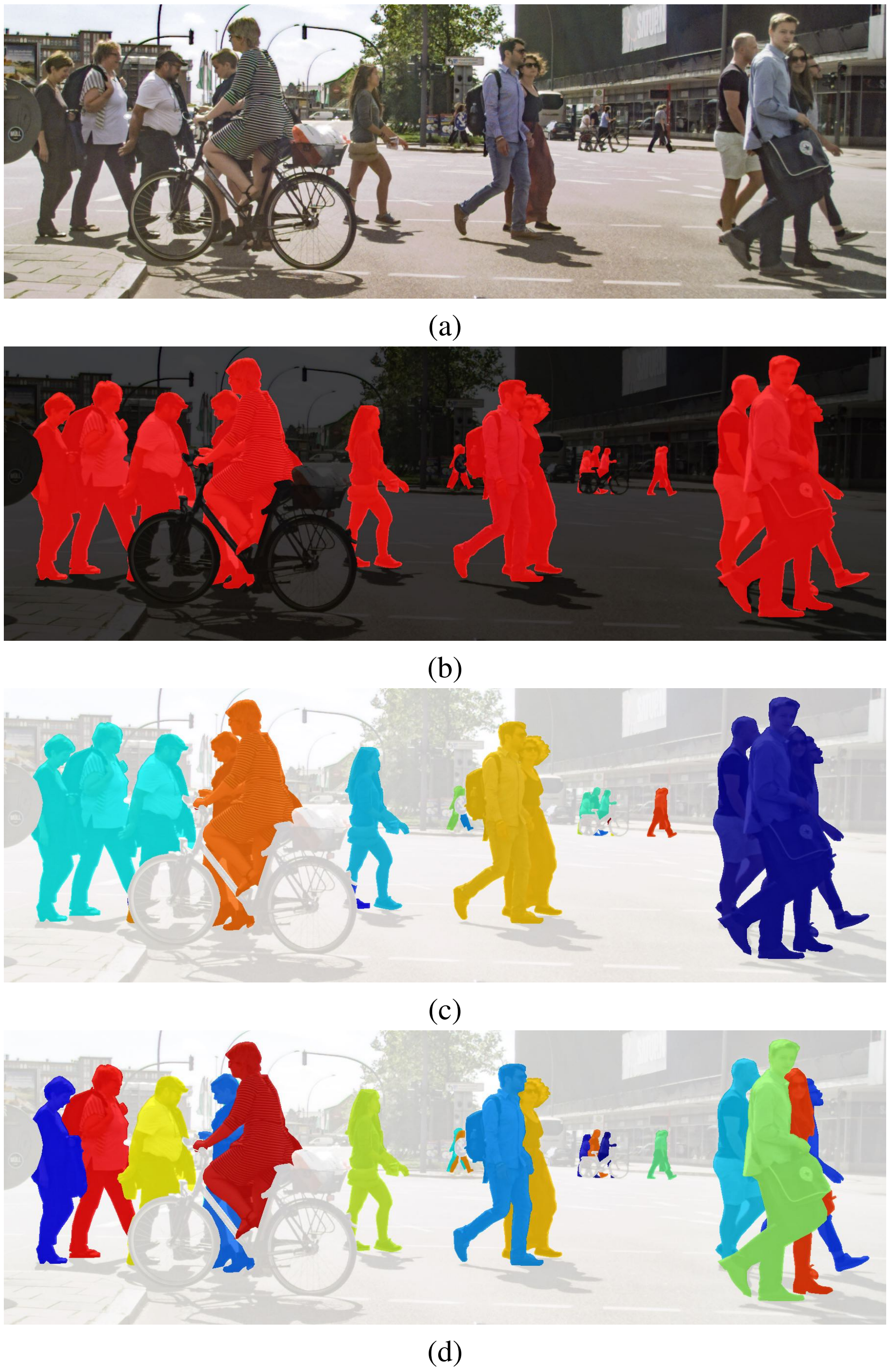


Figure 35: Semantic segmentation and instance-level segmentation of people (Cityscapes dataset [3], Hamburg image #036527): (a) original image; (b) semantic person segmentation; (c) grouping via connected components; (d) person instance segmentation.

number of objects in front of it. If there is no object in front it, it should be ranked layer 1. The object behind it should be ranked layer 2. The process goes on until every object has a depth ordering.

Traditional instance segmentation does not consider full spatial occupancy or depth ordering. In contrast, when occluded regions are taken into consideration, it is referred to as amodal segmentation [83]. The normal instance segmentation that does not consider the occlusion is referred to as modal segmentation.

To successfully segment occluded regions, the method not only needs to know where occlusions happen, but also the shape of unseen object parts relative to what is observed. If the objects are non-rigid, there can be multiple plausible solutions. Imagine a hand behind a curtain. It could be making any shape without people knowing. Any valid shape it can make is technically a version of the ground truth. On top of the inherent difficulty of the task, the lack of amodal ground truth makes it difficult to develop and evaluate new methods. Li and Malik [83] composited training data from PASCAL VOC 2012 [84] by overlaying foreground masks. However, the resulting images are unnatural, with unrealistic lighting and object scales.

Zhu et al. [85] introduced the COCO Amodal dataset, consisting of thousands of amodal masks approximated by human annotators. It should be pointed out that the "ground truth" is never truly the ground truth and only estimated by human annotators. It is close to the ground truth, but the real spatial layout was unavoidably lost when the original providers of the modal version of the dataset decided to not record it. Ehsani et al. [16] introduced a synthetic dataset "DYCE" consisting of images rendered from various indoor graphics models at different angles. Unfortunately, fundamental flaws exist in the ground truth for both datasets. COCO Amodal uses inconsistent rules between annotators regarding ambiguous issues such as whether shadows should be considered part of the mask. In DYCE, there are systematic rendering bugs found in the ground truth masks, and the annotation appears unprofessional with typos and synonyms in the labels such as "coffeemachine" and “coffemachine” coexisting as distinct classes. Figure 36 shows incorrectly rendered ground truth masks for the category “wall”. Table 1 shows the typos and synonyms in class names ground truth. The list is not exhaustive. Due to these reasons, the only two commonly known admodal instance segmentation datasets were practically unusable at the time of writing.

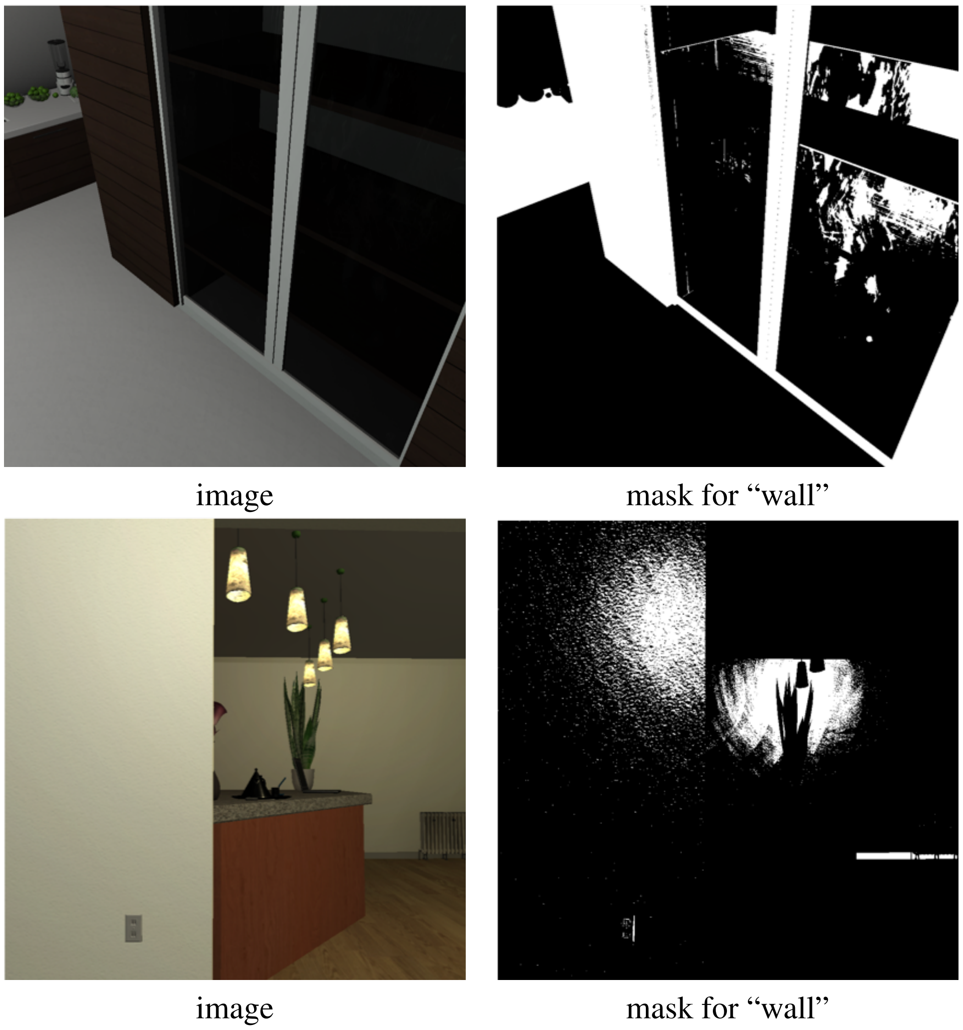


Figure 36: DYCE [16] image and ground truth mask for “wall”; it appears that the mask for light reflected off the wall is provided instead

Due to the reasons mentioned above, the methods in this thesis are trained and tested on our own synthetic dataset. The details of this dataset can be found in section 3.

Fortunately, in this year’s CVPR, two amodal instance segmentation datasets were re-

Table 1: Typos and synonyms in class names ground truth in DYCE

| |
|---|
| walls, wall |
| rug, carpet |
| mug, cup |
| diningtable, table |
| tray, plate |
| coffemachine, coffeemachine, coffemaker |
| couch, sofa |
| painting, picture |
| island, cabinet |
| lamp, light |

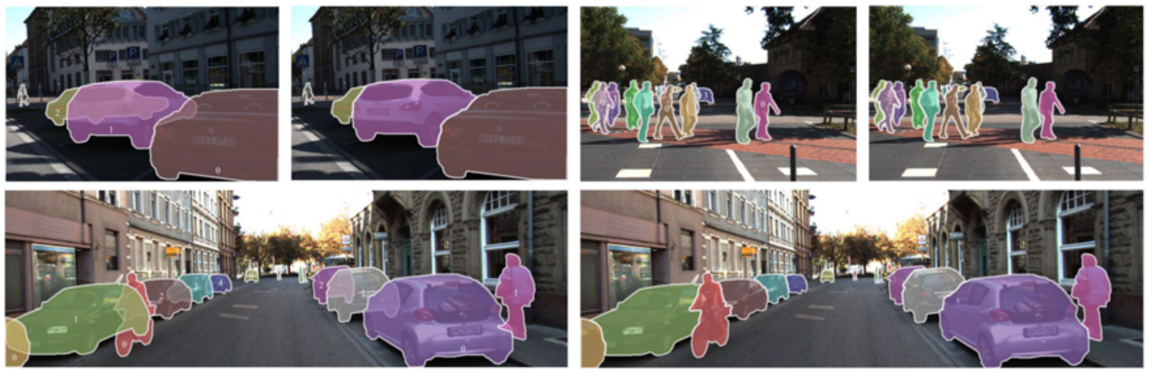


Figure 37: KINS [17]

leased. They are called KINS and SAI-VOS. One is further annotated on top of KITTI, the other is rendered using GTA V game engine. We hope that these two datasets will inspire more research in the area. Figure 37 and Figure 38 show example data from these datasets.

Because ground truth instance labels are permutation invariant, the common approach of training deep fully-convolutional networks (FCNs) to detect and segment objects faces the dilemma of an ambiguous target [15]. In semantic segmentation, each class label can be formatted as a one-hot encoding, such that the difference between two one-hot encoding vectors can be measured with categorical cross-entropy. Labels for instances can be represented as integers in the ground truth save file, but not for the loss function. Integer instance labels unavoidably mean the distance between instance 1 and instance 2 is closer than that between instance 1 and 3. Instance segmentation cannot use one-hot encoding either because the number of total instances is unknown, unlike semantic segmentation where the

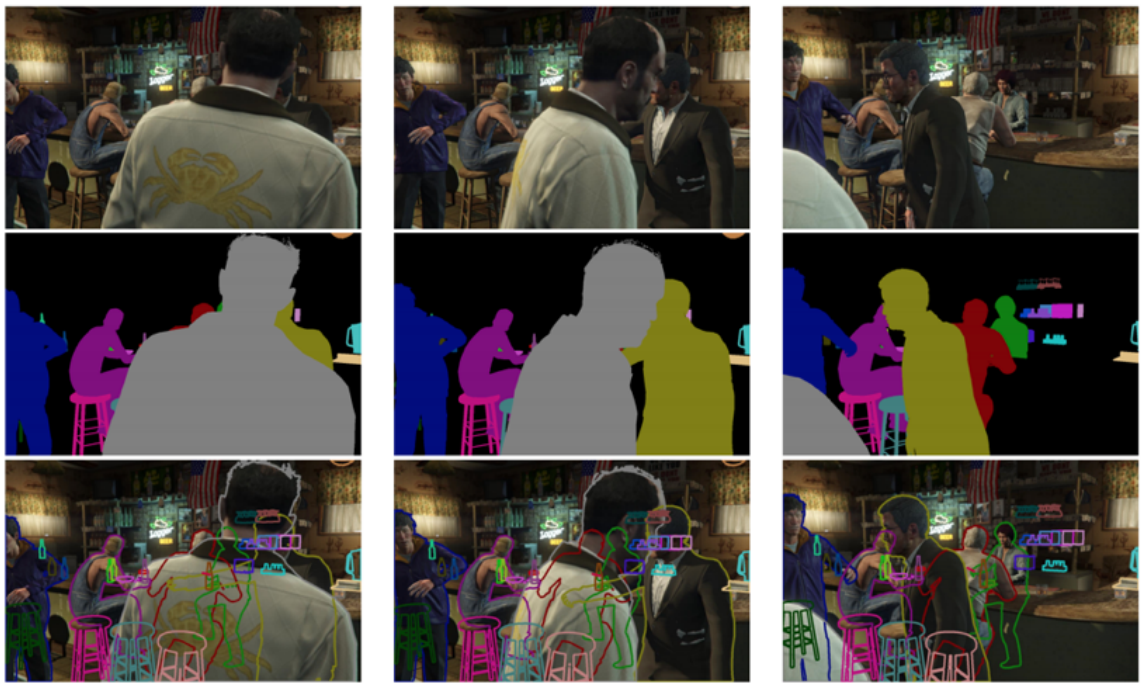


Figure 38: SAIL-VOS [18]

classes to classify are predefined.

There are generally two categories of approaches used to achieve instance segmentation. Top-down methods begin by finding the regions (often bounding boxes) that contain each instance, and then performing pixel-wise segmentation of the dominant instance within that region. For example, Mask R-CNN [4] extends Faster R-CNN [37] by adding a branch for segmentation mask prediction in parallel with the other branches (bounding boxes and classification). Li et al. [86] proposed an alternative that uses location-sensitive fully convolutional networks to partition bounding boxes into 3$\times$3 grids, and then evaluates the likelihood that each partition contains the correct part relative to the other partitions. Top-down methods bypass the ground truth instance label problem because each instance is isolated in the bounding box. There are no other masks that it needs to be distinguished from because the bounding box already does that.

The second category, bottom-up methods, begin by assigning attributes to pixels and then clustering them into instances. Examples include those that use pixel embedding to move the high-level detection stage to the end of the process [43, 41]. Fathi et al., [43] adopt this principle by training a network to evaluate pairwise pixel similarity. They train a separate model to generate seed points that represent the typicality of a pixel compared to other pixels in the area. Brabandere et al., [41] proposes a discriminative loss function to train pixel embeddings such that they are close within the same instance but far apart for different instances. The bottom-up methods solve the instance label problem by using clusters. Each cluster needs to be far enough away from other clusters, therefore eliminating the need for an integer-label-based metric.

The above methods propose a surjective mapping from pixels to instances. However, it is worth considering if this is an optimal representation of semantic instance segmentation. Computer vision often aims to reverse-engineer scenes from images/video, and an assignment of all visible parts to a single membership is an incomplete descriptor. In contrast, the full segmentation masks and relative depth ordering prior to image projection provides a more complete descriptor.

To this end, Yang et al. [87, 88] estimate layer ordering as part of instance segmentation and introduce a learned predictor based on relative detection scores, position on the ground plane, and size. They acknowledge the benefits of full spatial segmentations of visible and occluded parts, but their method focuses on the benefits of depth ordering for instance grouping. Chen et al. [89] attempt to fill occluded regions by selecting similar non-occluded exemplar templates from a library; this improves instance segmentation of visible pixels. Uhrig et al. [90] propose to consider explicit depth ordering estimation for instance segmentation. Their method exploits ground truth depth information, but it does not attempt to recover occluded segments. While each of these methods uses the concept of occlusions to improve instance segmentation, none of them explicitly targets the full spatial extents and depth ordering of instances.

Li and Malik [83] use an iterative approach to gradually predict the amodal masks based on the bounding box and classification produced by an object detector. They compute the visible mask and iteratively expand upon it to produce the amodal mask and bounding box. Ehsani et al. [16] propose a GAN-based model to produce both the segmentation and the appearance of the occluded regions, assuming that foreground segmentation is already pre-calculated by other methods. However, their method focuses on crops with one salient object.

Amodal segmentation remains as a challenging task and very few studies and datasets exist. Current methods either focus on a special case of the general problem or extend upon top-down approaches. This thesis proposes an alternative bottom-up approach and examines some challenges associated with amodal segmentation.

## 2.3 Multi-Object Tracking

Multi-object tracking (MOT) has been a popular research area for a long time. The goal of MOT is to track all objects in a video. This requires the position and identity of objects to be preserved. The position is usually described by a bounding box in the format of $(top, left, bottom, right)$ or a centroid in the format of $(x, y)$. The identity of the object is annotated as an object ID . The ID could be either an integer or a string, as long as it is unique.

Due to the strong demand of surveillance and autonomous driving, most multi-object tracking datasets contain only pedestrians. PETS 2009 [91] is an old dataset primarily targeting surveillance videos; MOTChallenge [92] is a large MOT dataset with data collected and released in 2015, 2016, and 2017. Tasks mostly include multi-people tracking, but more subsets such as 3D tracking and sports analysis are also coming soon; PoseTrack [93] contains human pose estimation and human pose tracking data, marking a shift towards more detailed human-oriented tracking tasks; NVIDIA AICity Challenge [94] contains multi-object detection and tracking data from surveillance cameras together with other tasks, such as vehicle speed estimation; Path Track [19] is a recently published dataset with diverse scenes from ball room to stadium; KITTI dataset [38] contains videos collected from car-mounted cameras, mostly from European cities. MOTS (Multi-Object Tracking and Segmentation) [21] is a 2019 dataset with both pedestrian tracking and segmentation data.

Despite the recent rise in popularity, multi-object tracking has been following a typical procedure pipeline for a long time. Figure 39 shows the workflow of a general MOT algorithm. A MOT algorithm can be roughly divided into the detection stage and the association stage. In the detection stage, an object detector is applied to each frame of the video and produces object locations, sometimes together with extra descriptors such as color histogram features or appearance embeddings. Kalman filter [95], proposed in 1960, is a common tool in estimating and refining the object location predictions and often plays

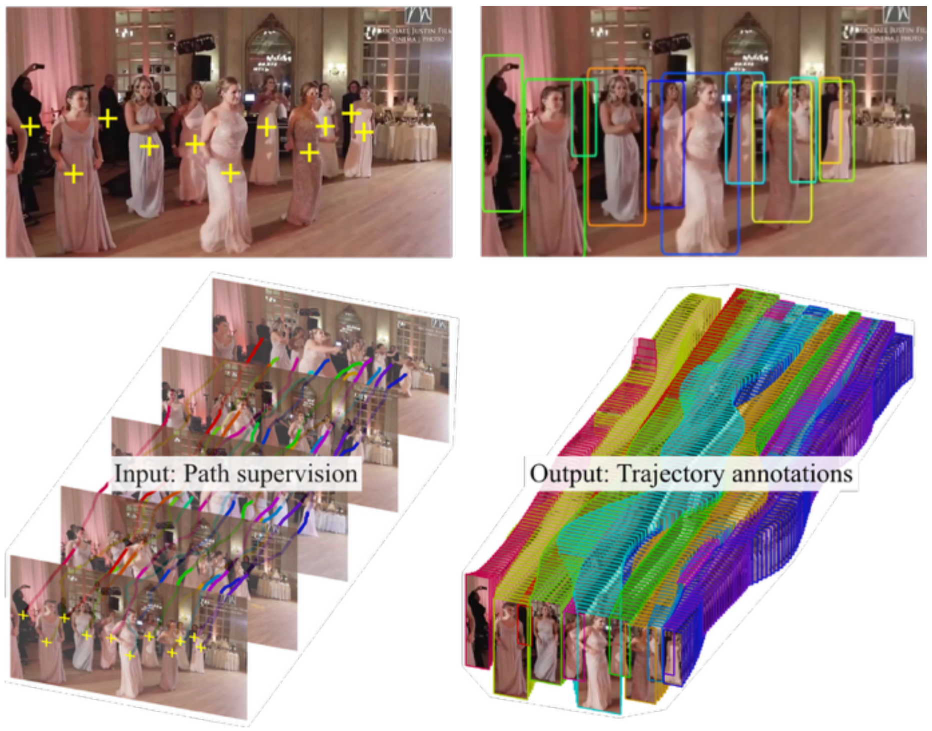


Figure 39: MOT workflow [19]

a part in state-of-the-art methods. In the association stage, the tracking algorithm associates the detections in the current frame with the detections in the previous frames. Offline methods also have access to future data so they can perform two-way associations. The unmatched detections are initiated as new tracklets, and the undetected objects that are previously tracked will be marked lost after a period of time. The famous Hungarian Algorithm [96] is usually used in the association stage. Most MOT algorithms do not deviate too much from this protocol. Most of the work is done to improve the speed or performance of the stages.

To measure the performance of MOT algorithms in benchmarks and competitions, many metrics are proposed. A good metrics system should be able to measure the proposed method's ability to precisely locate objects and consistently maintain their identities. It should also be intuitive to understand and easy to use, requiring as few parameters to choose as possible. Among the most commonly used are MOTA (Multi-Object Tracking Accuracy), MOTP (Multi-Object Tracking Precision), and IDF1 (Identity F1 Score). Both

MOTA and MOTP are proposed by Bernadin and Stiefelhagen in [97] as part of the proposed CLEAR MOT metrics system. Their proposed system fits the description of a good MOT metrics system above with the following rules: (1) all correspondences between a ground truth object and a hypothesis can only be considered when their distance is within a threshold, regardless of how this distance is computed; (2) when the tracker makes a mistake by switching the identities of two objects that it is tracking, the metrics system only punishes the results once, regardless of when the identity switch happens. This ensures that the switch has a fixed punishment unrelated to the timing. An alternative metrics design that calculates the accumulated distance error between a ground truth object and a hypothesis would be unfair because it favors late mistakes rather than early mistake due to the error buildup; (3) when two object proposals are both within the distance threshold of a previous tracked object, the one with the same identity as the previous one will be favored, even if it has a longer distance, because identity consistency is more important than accurate localization.

The MOTP score is calculated as the following:

$$MOTP = \frac{\sum_{i,j} d_t^i}{\sum_t c_t}, \tag{4}$$

where $d_t^i$ is the distance between a ground truth object and its matched hypothesis, and $c_t$ is the number of matches in frame $t$.

The MOTA score is calculated as the following:

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t}, \tag{5}$$

where $m_t$ is the number of misses, $fp_t$ is the number of false positives, and $mme_t$ is the number of mismatches.

MOTP measures the average distance between hypothesis and object. MOTA accounts for all object configuration errors made in the tracking process. It is therefore more commonly used than MOTP. This is also because, once again, maintaining consistent identities is more important than coming up with precise bounding boxes or centroids.

IDF1 measures the ratio of correctly identified detections over the average number of ground-truth and computed detections. It is proposed by Ristani et al. in [98]. The authors argue that it is more important to measure how well a tracker can correctly determine who is where at all times than how often it makes incorrect decisions, which is what CLEAR MOT measures. The calculation of IDF1 is a bit more involved than the other two, as described below:

$$IDFN = \sum_{\tau \in AT} \sum_{t \in T_\tau} m(\tau, \gamma_m(\tau), t, \Delta) \tag{6}$$

$$IDFP = \sum_{\tau \in AC} \sum_{t \in T_\gamma} m(\tau_m(\gamma), \gamma, t, \Delta) \tag{7}$$

$$IDTP = \sum_{\tau \in AT} len(\tau) - IDFN = \sum_{\tau \in AC} len(\gamma) - IDFP \tag{8}$$

$$IDF_1 = \frac{2IDTP}{2IDTP + IDFP + IDFN}, \tag{9}$$

where $\tau$ is a true trajectory, $\gamma$ is a computed trajectory, $AT$ is true identities, $AC$ is computed identities, $m(\cdot)$ is misses, $t$ is time, and $\Delta$ is the intersection threshold of two detection boxes.

IDF1 provides a balanced measure of precision and recall. According to the authors, the new proposed metrics system achieves the following properties similar to CLEAR MOT: (1) a correct match is one-to-one; (2) the matching is the most favorable to the tracker; (3) errors of any type are penalized in the same currency. Moreover, it also handles overlapping and disjoint fields of view in exactly the same way. This is a missing feature from other previous methods, but sometimes less useful. For example, in this focused area of this thesis work, there are no multi-camera setups that require re-identification, therefore no disjoint or overlapping fields of view. Nevertheless, IDF1 serves as a robust and intuitive measurement in evaluation part of this thesis work.

The rich collection of people datasets in turn enabled many research works in people-oriented multi-object tracking. With the rise of deep learning, most methods are now using it in one aspect or more.

DeepMOT [99] makes the observation that most tracking algorithms rely on the Hungarian Algorithm, which is non-differentiable. Differentiability is a key feature of a problem that can be solved with deep learning. If the assignment algorithm can only be treated as a post-processing step, that means the tracking algorithm cannot be directly optimized based on the MOTA and MOTP metrics. The authors therefore propose a differentiable assignment algorithm that allows their tracking algorithm to be optimized end-to-end.

Bergmann et al. [100] approach the tracking problem in a surprisingly simple yet effective way. They modify a Faster R-CNN to predict the bounding boxes of currently tracked objects in the next frame, on top of the regular detection results from the original Faster R-CNN, which are the bounding box locations and classifications for the current frame. This simple setup is coupled with a constant speed motion model and a Siamese network for appearance embedding. The input is formulated as a stacked array of frames in order to provide spatiotemporal information. The authors urge fellow researchers to focus on more challenging aspects of multi-object tracking because they discover that fancy and complex models do not significantly outperform their no-bell-and-whistle Faster R-CNN model (Tracktor++) on current datasets. If tracking-by-detection can be just as good, the authors ask, what is the real benefit of dedicated tracking methods? A brief investigation of success and failure cases for their method reveals that all existing methods tend to fail on occluded objects and small objects.

Wang et al. [20] proposes a model that integrates bounding box detection and visual embedding into a shared model. They argue that a shared model saves computation and allows multiple tasks to be jointly optimized. They exploit several recently proposed techniques to improve performance on top of their novel design. First, the authors utilizes an improved version [101] of triplet loss [102] to help pulling apart embeddings from all instances of different identities in a batch, instead of just a positive pair and a negative pair. Second, the authors uses the idea from [103] to make the weights of multiple loss terms trainable so that they are better balanced. They train a YOLO-like one-stage model to simultaneously output the location, classification, and embedding per grid location. In post-processing, each object's visual embedding is updated with a weighted sum of previous and current embedding, and its location adjusted with a Kalman filter. The tracker then performs the Hungarian Algorithm twice, once based on appearance similarity, the other based on location overlap.

Voigtlaende et al. [21] proposes a model (TrackR-CNN) to simultaneously perform detection, tracking and segmentation for multiple objects, together with a corresponding dataset that they annotated. They make the observation that bounding-box based performance has been saturating, based on recent tracking evaluations on common benchmarks. They point out that bounding box-level annotations are sometimes too coarse, which this thesis agrees with and investigates. The problem is that the salient object is rarely the only relevant element inside the bounding box. Other objects of the same or different category often appear partially within the bounding box as well. A visual descriptor will then fail to extract useful information due to competition, unless a mask is also provided. The authors therefore extend Mask R-CNN, which already provided instance segmentation and bounding box detection, with a head for 3D convolution to process temporal information and link object identities over time.

It can be argued that these algorithms can be generic enough to be migrated to other domains, such as cars or other categories. However, as the field advances, more algorithms are starting to tailor their tricks towards pedestrian-only applications. Wang et al. specify the anchor boxes in the detector to be exclusively 3:1 aspect ratio since humans are usually standing upright [20]. This assumption does not hold for all objects; Tang et al. leverage techniques from person re-identification to help improve tracking performance [104].

The skewed focus on pedestrian tracking brings extra limitation to existing MOT methods because not all objects can be differentiated based on appearance. For example, animals look very similar to each other because they do not wear clothes; cells are almost identical-looking because they are split from a common cell. Despite the visual similarity, humans still succeed in tracking multiple objects. We hypothesize that this is partially due to our perception of continuous movements. In other words, as long as the objects are moving continuously in a video with a reasonable frame rate, people can follow the objects by extracting the spatiotemporal information even without the help of discriminative appearances.

# 3 Amodal Instance Segmentation Using Deep Pixel Embedding

## 3.1 Motivation

Methods relying on bounding box selection are inherently limited by *a priori* region selection. When instances of similar size overlap with one another, the region selection phase often experiences one of three types of error: 1) due to non-maximum suppression, the algorithm ignores the bounding box of the occluded instance, as shown in Figure 40 (b); 2) the instance will be represented as a collection of separate partial instances with different labels, as shown in Figure 40 (c); 3) the segmentation model gets confused about which object should be segmented, so it segments many objects within the bounding box and produces a mixed mask.

The issues mentioned above indicate that bounding box is a low-quality descriptor of object orientation, shape, location, and depth ordering. To solve these issues, bounding box needs to be moved from the procedure and the workflow needs to be reworked from bottom up. Masks, on the other hand, naturally described the orientation, shape, precise location, and depth ordering of objects. To produce fine-grain masks, the network needs to produce pixel-level features based on a larger context region instead of the region defined by bounding boxes.

We propose a fully-convolutional, end-to-end trainable approach that jointly estimates

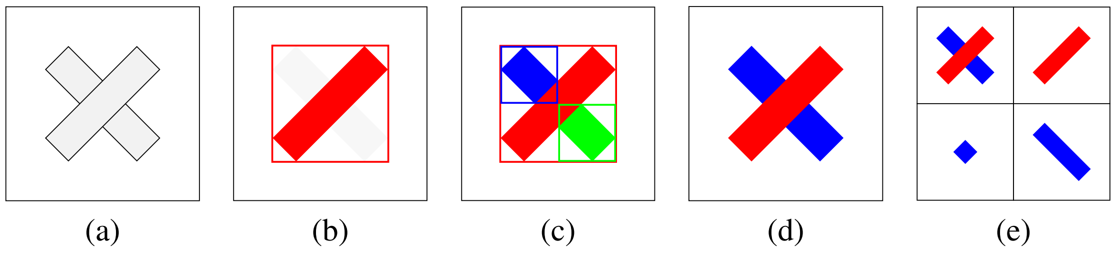


Figure 40: Instance segmentation output scenarios for two overlapping rectangles. Image rendered by the author.

the presence of occlusion and provides consistent instance labeling across foreground and occluded regions. The method is evaluated on an easily configurable synthetic dataset consisting of various types of shapes with occlusions with precisely known amodal masks. Results demonstrate that the method is capable of accurately estimating layered spatial occupancy and outperforming a state-of-the-art top-down alternative.

## 3.2 Method

The goal of the proposed method is to produce full instance masks for each segmented object as long as part of the object is visible in the image. To circumvent the limitations of DYCE and COCO Amodal datasets, a synthetically generated dataset of shapes is used. The advantages of this set include 1) full control of scene complexity; 2) access to precise ground truth; and 3) rigid shapes where the ground truth is often unique given partial observations.

The dataset has three classes of shapes: triangles, rectangles, and circles. All shapes have a fixed size, but their locations, orientations, and depth orderings are randomized. Shapes have the same color as the background, only distinguished by their black outlines, so that the network cannot cheat by simply detecting color or intensity. This representation forces the network to rely on outlines and be aware of large regions for context.

For training and evaluation, the ground truth masks are arranged in the following manner: foreground semantic classification, occlusion semantic classification, foreground instance labels, and occlusion instance labels. See Fig. 41 for a sample of training data. The proposed method generates the following four outputs: 1) *foreground multi-class semantic segmentation*, that labels pixels as background, foreground, or occluded; 2) *occlusion multi-class semantic segmentation*, that labels occluded pixels as one of the classes; 3) *foreground embedding*, used to cluster foreground pixels into instances; 4) *occlusion embeddings*, that are consistent with visible instances. It is worth noting that occluded pixels are defined as those where one instance occludes another instance. While the method does

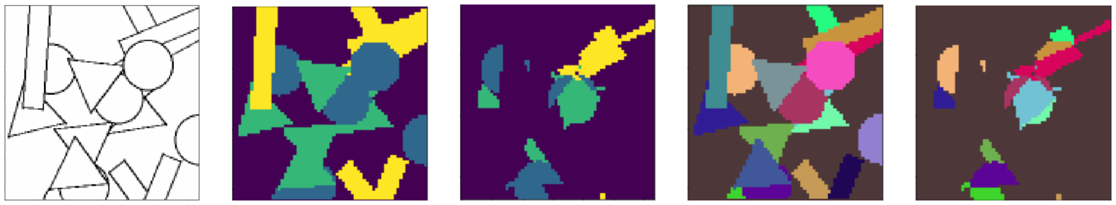


Figure 41: Sample training data. From left to right: input image, foreground class mask, occlusion class mask, foreground instance mask, occlusion instance mask.

not consider occlusions caused by background objects, this could trivially be added as another class to the output.

To train instance embedding across both the foreground and occluded regions, the method uses a variation of the discriminative loss function introduced in [41] . Consider an input image $\mathcal{I}$ and a pair of embedding outputs $\mathcal{E}_f$ and $\mathcal{E}_o$ that contain embedding vectors for the foreground region and occluded region, respectively. The embedding outputs are matrices with the same spatial dimensions (rows and columns) as the input image, where the number of channels $C$ represents the dimensionality of each pixel's embedding. The goal of the network is to map each foreground pixel $p \in \mathcal{I}$ to a $C$-dimensional embedding vector $\mathcal{E}_f(p)$ and each occluded pixel $p \in \mathcal{I}$ to a $C$-dimensional embedding vector $\mathcal{E}_o(p)$ such that embedding vectors for pixels belonging to the same instance are close together in the $C$-dimensional space and embedding vectors for different instances are far apart.

The overall loss consists of three terms: variance $l_{var}$, distance $l_{dst}$, and regularization $l_{reg}$. Let $K$ be the total number of classes, $N_k$ be the number of instances of class $k$, $N$ be the number of ground truth instances and let $\mathcal{R}_f^n \subseteq \mathcal{I}$ and $\mathcal{R}_o^n \subseteq \mathcal{I}$ denote the set of foreground and occluded pixels for instance $n \in \{1, \ldots, N\}$, respectively. Also, let $\mu_n$ be the average embedding vector of all pixels in both the foreground and occlusion embeddings for instance $n$. The variance term and distance term are defined as

$$l_{var} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{|\mathcal{R}_f^n|+|\mathcal{R}_o^n|}\left(\sum_{p\in\mathcal{R}_f^n}\left[||\mu_n - \mathcal{E}_f(p)|| - d_{var}\right]_+^2 + \sum_{p\in\mathcal{R}_o(n)}\left[||\mu_n - \mathcal{E}_o(p)|| - d_{var}\right]_+^2\right) \tag{10}$$

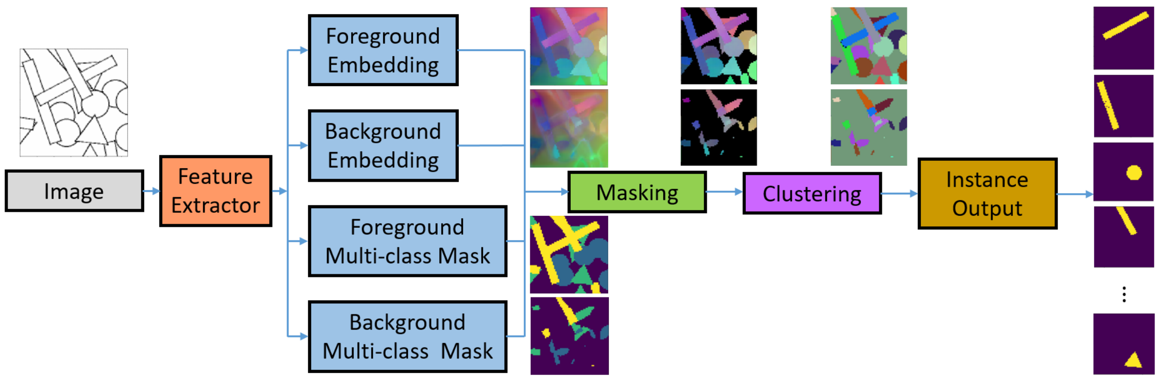


Figure 42: Proposed segmentation architecture.

and

$$l_{dst} = \sum_{k=1}^{K} \frac{1}{N_k(N_k-1)} \sum_{n=1}^{N_k} \sum_{\substack{m=1 \\ m \neq n}}^{N_k} [2d_{dst} - ||\mu_n - \mu_m||]_+^2, \tag{11}$$

where $[a]_+ = max(a, 0)$ is the hinge loss, and $|| \cdot ||$ is L1 distance. Constants $d_{var}$ and $d_{dst}$ are the margins for the variance and distance term. Effectively, $l_{var}$ penalizes pixels that belong to the same instance but are farther than $d_{var}$ apart in the embedding space, and $l_{dst}$ penalizes cluster centers that represent different instances but are closer than $d_{dst}$. The regularization term

$$l_{reg} = \frac{1}{N} \sum_{n=1}^{N} ||\mu_n|| \tag{12}$$

prevents the network from minimizing $l_{dst}$ by simple embedding amplification. Finally, the network is trained to minimize $L_{total} = \alpha \cdot l_{var} + \beta \cdot l_{dst} + \gamma \cdot l_{reg}$. Fig. 42 presents an example of the method applied to shapes.

The network uses a pre-trained feature extractor and produces a depth-concatenation of four outputs. The four modules share the output from the feature extractor. The method clusters the network's pixel embeddings into instances using the algorithm presented in [41]. A random unlabeled pixel is selected and the embeddings around it within $v_{var}$ are grouped together. The mean embedding of this group is used for the next round of grouping until convergence.

For post-processing, the algorithm starts by randomly picking a non-labelled pixel and labels all pixels with an embedding within the similarity threshold as the same instance. The algorithm then randomly selects another pixel and repeats this process until all pixels are labelled. The threshold used here is the same value as the threshold used in training, which is $d_{var}$.

## 3.3 Implementation Details

The proposed method uses DeeplabV3+ [23] with an Xception backbone as the feature extractor. Its final upsampling and logit layers are removed and the 256-dimensional output is used as features. Input size is set to $256 \times 256$ and the output size is $64 \times 64$. For the loss function, $\alpha = \beta = \gamma = 1, d_{var} = 0.5, d_{dst} = 1.5$. Embedding dimension $C = 6$ and the mean shift threshold for clustering is 1.5, which is consistent with $d_{dst}$. The embedding module consists of 256, 256, 128, and $C$ convolution filters, with RELU activations.

For comparison, Mask R-CNN is selected as a representative model of top-down approaches for baseline due to the success and popularity of the R-CNN family among object detection and segmentation architectures. It is an architecture originally designed only for foreground instance segmentation, but it can be easily modified to perform amodal instance segmentation by fine-tuning on amodal ground truth. Its weights are pre-trained on MS COCO. Its output is upsampled from the original $m \times m$ mask to the corresponding bounding box size and put in the context of the whole image. The final output is resized to $64 \times 64$ to be directly comparable with our method.

All models are trained for 100 epochs in Tensorflow with Adam optimization (learning rate = 1e-4 and batch size = 2). Training examples are generated at runtime with the same initial random seed. Each epoch has 1000 images.

## 3.4 Experiments

The embedding model and Mask R-CNN are both evaluated on 1000 shapes images randomly generated with 6 instances per class. Results in Table 2 show that our model outperforms Mask R-CNN on AP, $\text{AP}_{50}$, $\text{AR}_{\text{None}}$, and $\text{AR}_{\text{Partial}}$. Mask R-CNN achieves better results on $\text{AP}_{75}$ when non-max suppression threshold $t = 0.7$, and $\text{AR}_{100}$, $\text{AR}_{\text{Heavy}}$ when $t = 0.9$, but it cannot retain high performance with a single value of $t$. The same performance pattern repeats when the number of instances per class is increased from 6 to 12.

As part of an ablation study, the semantic segmentation prediction for the model is replaced with ground truth. With 6 instances per class, AP increases from 0.7673 to 0.7959 and $\text{AR}_{100}$ increases from 0.7933 to 0.8300 when using ground truth; with 12 instances per class, AP increases from 0.5262 to 0.5419 and $\text{AR}_{100}$ increases from 0.5697 to 0.6013. These marginal improvements suggest that semantic segmentation is not the primary bottleneck of performance.

In the second part of the ablation study, the instance clustering result is replaced by ground truth. This way, the only source of error is insufficient layers of masks to accommodate the full complexity of occlusions. As Table 4 shows, the performance gains diminishing improvement as the model allows more layers to represent occlusion. Data

Table 2: Performance of models on 6 instances per class with different NMS threshold $t$

| $N$ | Model | AP | $\text{AP}_{50}$ | $\text{AP}_{75}$ | $\text{AR}_{100}$ | $\text{AP}_{\text{None}}$ | $\text{AR}_{\text{Partial}}$ | $\text{AP}_{\text{Heavy}}$ |
|---|---|---|---|---|---|---|---|---|
| 6 | Ours | **0.7673** | **0.9091** | 0.7800 | 0.7933 | **0.9983** | **0.9637** | 0.6190 |
| | $\text{MRCNN}_{t=0.1}$ | 0.5553 | 0.6526 | 0.6249 | 0.5823 | 0.7916 | 0.6986 | 0.4373 |
| | $\text{MRCNN}_{t=0.3}$ | 0.6781 | 0.8010 | 0.7706 | 0.7132 | 0.8765 | 0.8417 | 0.5898 |
| | $\text{MRCNN}_{t=0.5}$ | 0.7238 | 0.8701 | 0.8187 | 0.7620 | 0.9017 | 0.8768 | 0.6562 |
| | $\text{MRCNN}_{t=0.7}$ | 0.7332 | 0.8860 | **0.8323** | 0.7766 | 0.9112 | 0.8822 | 0.6748 |
| | $\text{MRCNN}_{t=0.9}$ | 0.6109 | 0.7334 | 0.6800 | **0.8010** | 0.9240 | 0.8978 | **0.6998** |

Table 3: Performance of models on 12 instances per class with different NMS threshold $t$

| $N$ | Model | AP | $\text{AP}_{50}$ | $\text{AP}_{75}$ | $\text{AR}_{100}$ | $\text{AP}_{\text{None}}$ | $\text{AR}_{\text{Partial}}$ | $\text{AP}_{\text{Heavy}}$ |
|---|---|---|---|---|---|---|---|---|
| 12 | Ours | **0.5262** | **0.7099** | 0.5192 | 0.5697 | **0.9958** | **0.9499** | 0.4049 |
| | $\text{MRCNN}_{t=0.1}$ | 0.3564 | 0.4350 | 0.4068 | 0.3730 | 0.7247 | 0.6220 | 0.2562 |
| | $\text{MRCNN}_{t=0.3}$ | 0.4641 | 0.5827 | 0.5312 | 0.4887 | 0.8226 | 0.7678 | 0.3715 |
| | $\text{MRCNN}_{t=0.5}$ | 0.5127 | 0.6416 | 0.5874 | 0.5399 | 0.8618 | 0.8019 | 0.4290 |
| | $\text{MRCNN}_{t=0.7}$ | 0.5203 | 0.6570 | **0.5926** | 0.5533 | 0.8722 | 0.8110 | 0.4424 |
| | $\text{MRCNN}_{t=0.9}$ | 0.4022 | 0.5253 | 0.4450 | **0.5856** | 0.8913 | 0.8326 | **0.4702** |

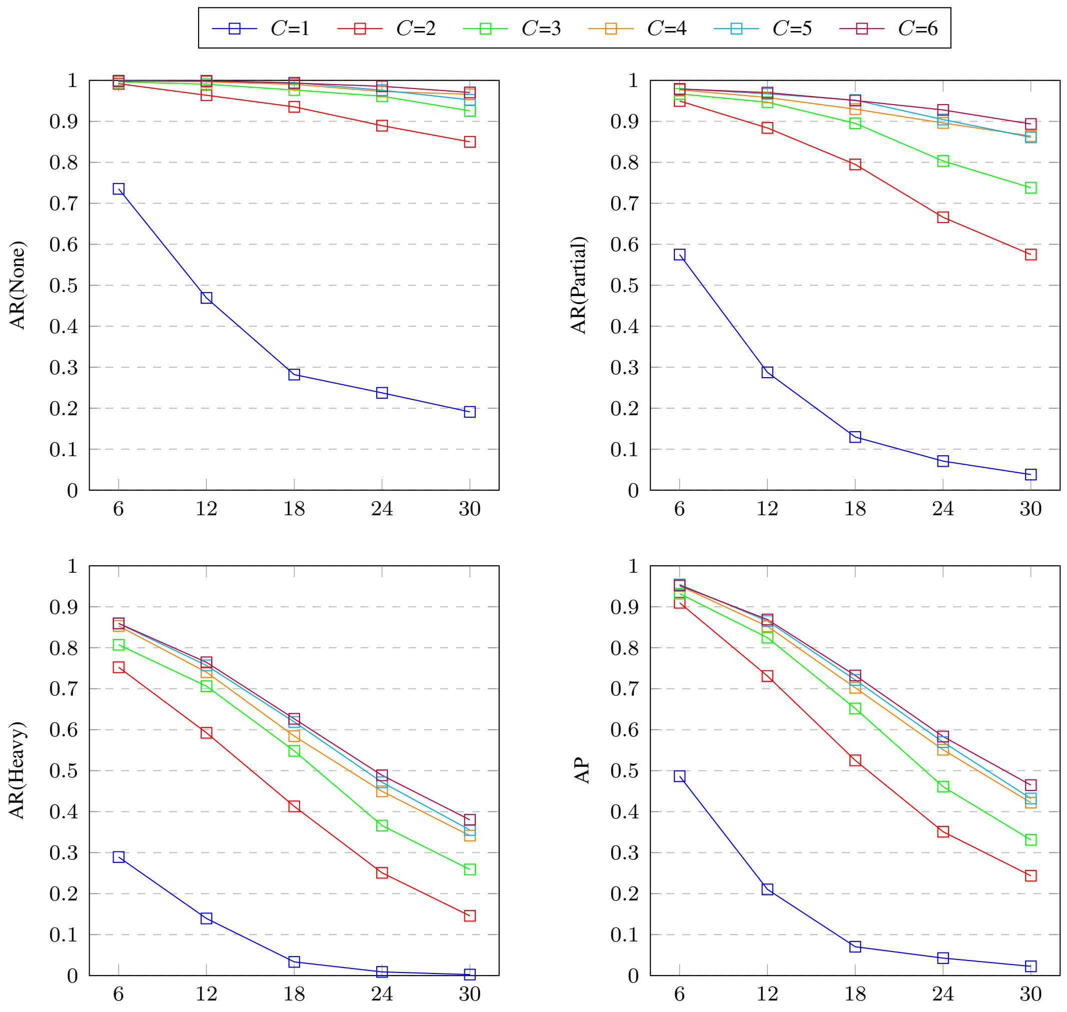
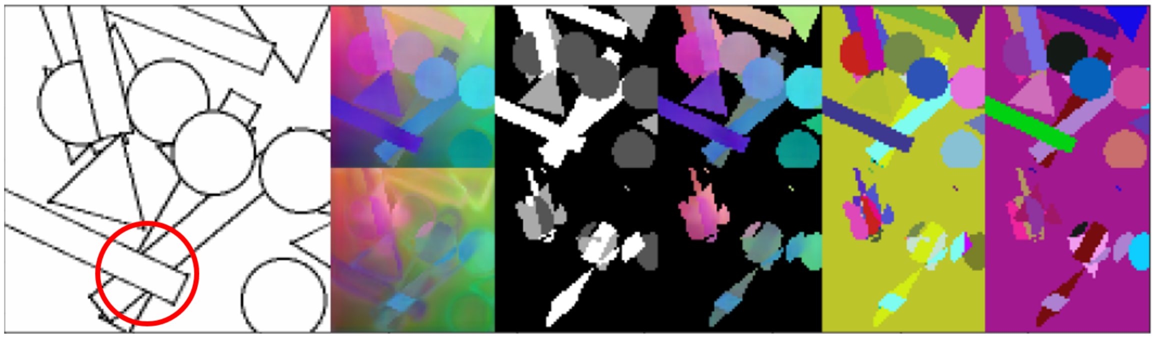


Figure 43: Performance for different number of instances per class and different embedding dimensions $C$. x-axis is number of instances, y-axis is different metrics, and colored lines represent models with different $C$.

suggests that three or more layers of mask output in the embedding model instead of just “foreground and occlusion” could improve performance, depending on the complexity of the application.

The embedding model is also trained and evaluated on shapes datasets with different number of instances per class in order to study its embedding capacity. Since embeddings for shapes of different classes are allowed to be similar, the class is limited to rectangles in this study. Fig. 43 shows that the performance drops when there are more instances of the same class. This happens for two reasons: first, more instances introduce more


(a) left to right: image, embedding, semantic segmentation mask, masked embedding, labels, ground truth. Top is foreground, and bottom is occlusion

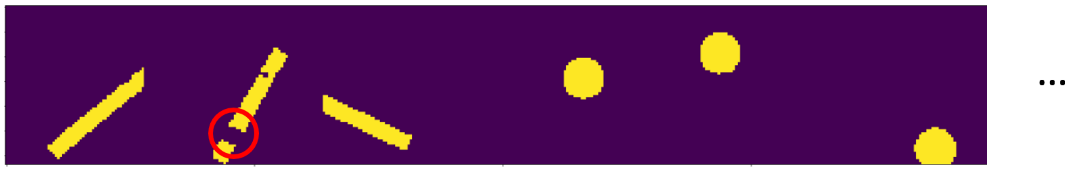

(b) individual instance masks

Figure 44: Failure cases for embedding model. Red circle indicates a triple stack.

occlusions, which increases the chance of more than two objects stacking together; second, distinguishing between more instances within the same class requires the model to have higher embedding capacity. Fig. 43 also shows that increasing the dimension results in diminishing but consistent improvement on performance. This is because the loss function encourages embeddings of the same instance to be close to one another, and embeddings of different instances to be far away. The penalty on the magnitude of embeddings makes this goal hard to achieve in low dimensions. In higher dimensions it is much easier to find another embedding that is both close to the origin and far enough from other embeddings.

The structure of the embedding model allows easier expansion into three or more layers of masks. Two layers shows its limits when objects are heavily occluded. For example, it is impossible to get correct results in Fig. 44 because there are three objects stacked within the red circle. The model can recover two of them at best. This is the most typical failure case for the proposed embedding model. By evaluating masks constructed from different number of layers of ground truth, Table 2 and Table 3 show that more layers of masks will lead to better results and that when the number of layers is held constant, performance on

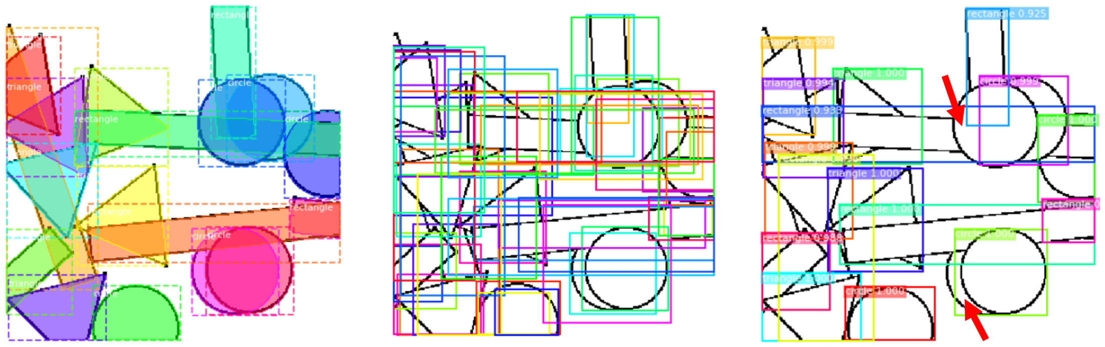


Figure 45: First failure case for Mask R-CNN. Left to right: ground truth, unfiltered proposals, filtered proposals

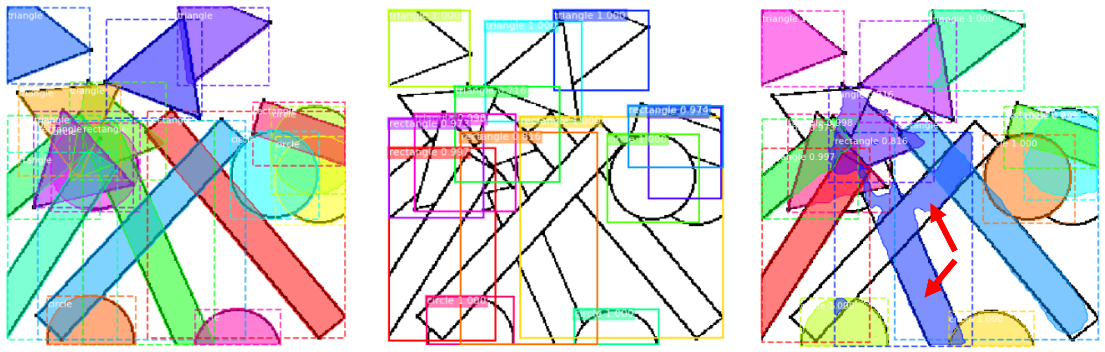


Figure 46: Second failure case for Mask R-CNN. Left to right: ground truth, filtered proposals, final results

heavily occluded regions gets worse because more instances are stacked.

Mask R-CNN has two typical failure cases, as shown in Fig. 45 and Fig. 46. In the first case, the region proposal network generates the correct bounding boxes for the indicated circles, but some get filtered out during the non-maximum suppression stage. In the second case, both indicated rectangles fit within one bounding box and the mask generator is confused about which one is the salient object.

This points to a fundamental difference between the two types of approaches. Top-down, bounding-box-based approaches have the ability to look at a region multiple times and potentially generate a complete mask each time. However, this also acts as a double-edged sword when multiple objects could appear in the same bounding box and compete for the mask if they are the same class. Spamming bounding boxes will also cause many

Table 4: Performance of instance masks constructed from different number of layers of ground truth masks. $N$ is the number of instances per class.

| $N$ | Layers | AP | $AP_{50}$ | $AP_{75}$ | $AR_{100}$ | $AP_{\text{None}}$ | $AR_{\text{Partial}}$ | $AP_{\text{Heavy}}$ |
|---|---|---|---|---|---|---|---|---|
| 6 | 2 | 0.8584 | 0.9604 | 0.8614 | 0.8627 | 1.0000 | 0.9927 | 0.7408 |
| | 3 | 0.9703 | 0.9901 | 0.9802 | 0.9744 | 1.0000 | 0.9997 | 0.9509 |
| 12 | 2 | 0.6594 | 0.8317 | 0.6436 | 0.6611 | 1.0000 | 0.9910 | 0.5251 |
| | 3 | 0.8703 | 0.9703 | 0.8812 | 0.8729 | 1.0000 | 0.9993 | 0.8206 |
| | 4 | 0.9584 | 0.9901 | 0.9703 | 0.9629 | 1.0000 | 0.9999 | 0.9475 |

false positives. The embedding model, on the other hand, is a bottom-up, bounding-box-free approach. Each pixel can only have one final embedding per layer, which will then be clustered into an instance label or ignored as background. The trade-off is that, situations like Fig. 45 and Fig. 46 can be avoided, but, when number of layers of ground truth mask in the training data is insufficient, the best possible performance will have a relatively low upper bound.

Figure 47 shows some of the foreground embedding and clustering results. The embeddings are reduced to three dimensions using principal component analysis and then mapped to between 0 and 1 for RGB display. For instance segmentation, each label is assigned a random color for uniqueness. Semantic segmentation is displayed in greyscale. Integer 1, 2, 3 are used to indicate class 1, 2, 3 in the labelling, which correspond to circle, triangle, and rectangle.

## 3.5 Discussion

The proposed method pushes the boundaries of a deep network's understanding of images by training it to estimate segmentation masks for unseen parts, and to associate them with visible instances. Experiments show that this bottom-up approach outperforms Mask R-CNN, a typical architecture for instance segmentation, by addressing the fundamental flaw in top-down approaches: the inability of bounding boxes to precisely capture the spatial relationship between cluttered objects. While the method only considers two-layer occlusion scenarios, the network structure can be easily modified to handle an arbitrary number of

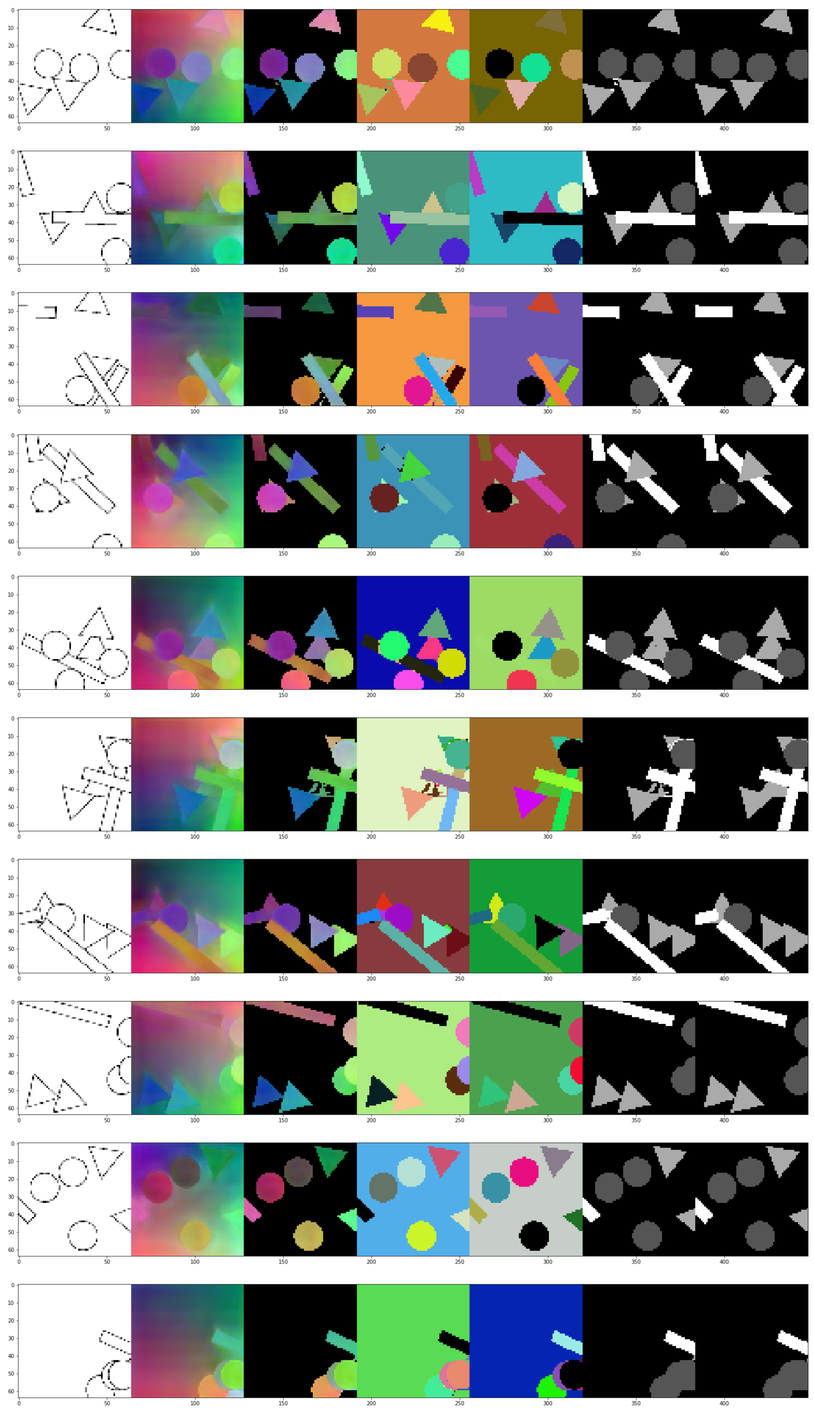


Figure 47: Sample results from the segmentation embedding model. From left to right: image, embedding, embedding masked by semantic segmentation, instance clustering, instance ground truth, semantic segmentation prediction, semantic segmentation ground truth

object types arranged in three or more layers.

Because it is difficult to obtain accurate annotations of occluded parts, the proposed method instead uses a synthetic dataset for training. We hope this and other recent works will motivate the creation of large datasets with ground truth masks representing the full spatial occupancy of occluded instances.

# 4 Multi-Object Tracking Using Deep Pixel Embedding

## 4.1 Motivation

The field of multi-object tracking has traditionally used bounding boxes as the dominant choice of location representation. As we showed in Section 3, there are many problems with this choice.

First, bounding boxes are not precise. They are upright rectangles that surround the object and its environment. The exact shape and area of the object is unknown from this simple description. Second, bounding boxes do not handle occlusion and heavily overlapping situations very well. By design, bounding boxes are passed through non-maximum suppression or one of its variant versions, to get rid of duplicate detections. Unfortunately, this process also gets rid of correct detections that are heavily overlapping with each other.

Second, segmentation performed based on the enclosed image inside the bounding box is inaccurate. Even when the segmentation algorithm is well designed and well trained, it still unavoidably fails when it is unclear which object inside the bounding box is supposed to be segmented.

These two issues carry over to bounding box-based tracking as well. Tracking heavily relies on a good object detector. When objects heavily occlude each other, as commonly seen in crowded pedestrian walking scenes, bounding boxes often either get lost or switched

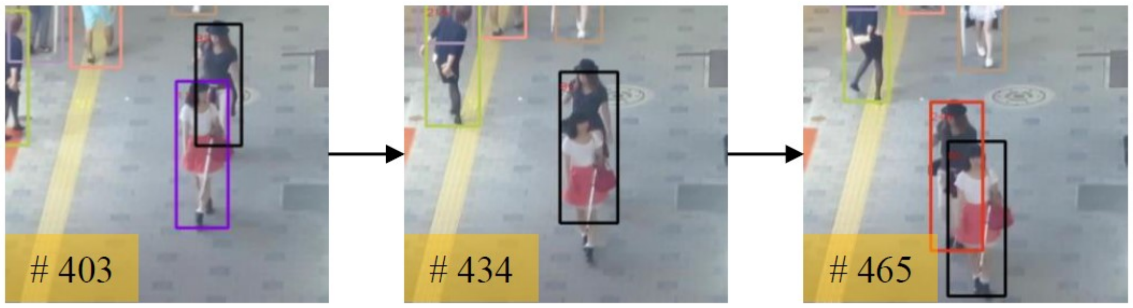


Figure 48: Identity switches to the other object that also appears in the same bounding box. Image from [20]

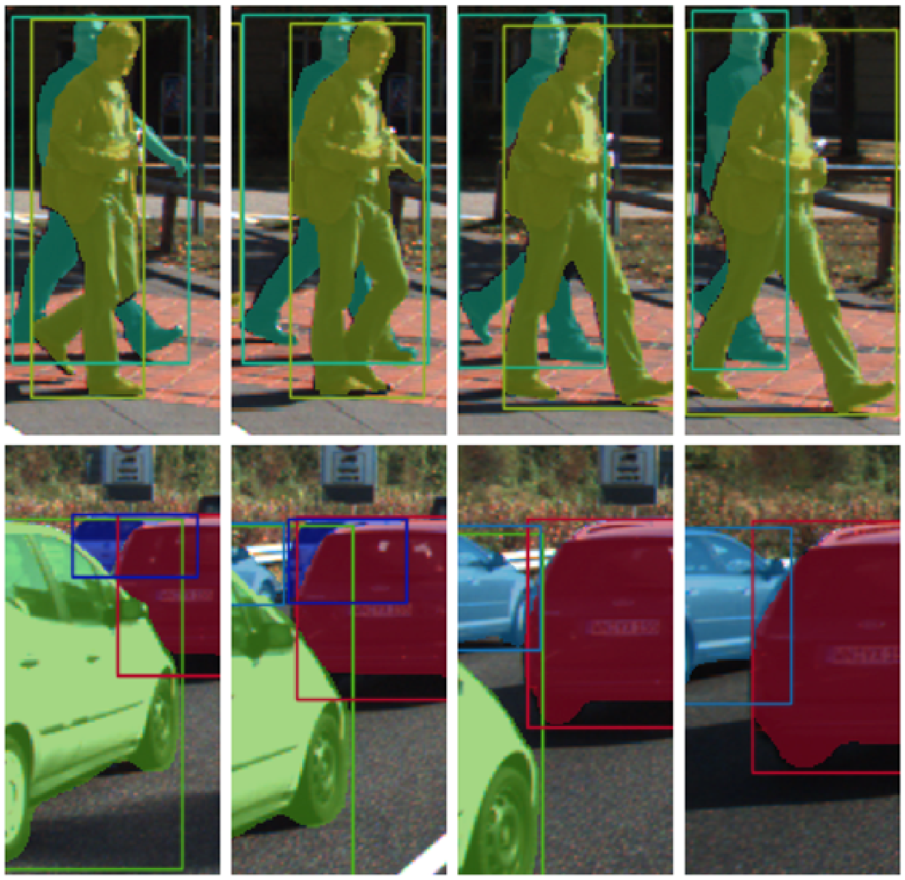


Figure 49: Issues with bounding boxes compared to masks. Image from [21]

to some other object. Switch happens mainly due to two reasons under this design: (1) the bounding boxes are so close to each other that the assignment cost of the incorrect one is the same as the correct one or lower; (2) when two objects appear within the same bounding box, the appearance encoding becomes a mixed balance of the two objects. As the appearance encoding drifts away, it gets associated with the incorrect object. Figure 48 from [20] illustrates this scenario.

Figure 49 also demonstrates these two issues. The bounding boxes in the top enclose two objects that are heavily overlapping, but the masks are able to capture the outline differences with pixel-level precision. In the bottom row, the bounding boxes meant to capture the salient object end up enclosing more area of the secondary object due to the shapes of the objects. Both problems can be solved by using masks instead.

## 4.2 Method

Due to the issues mentioned above, this thesis proposes a bounding box-free, segmentation-based tracking method. It is an extension of the amodal instance segmentation method introduced in chapter 3. It takes advantage of the extra pixel-level information given by the mask to produce more precise localization. By enforcing pixel embedding consistency between the amodal instance mask across two consecutive frames, tracking is simultaneously done via segmentation with no extra post-processing for data association or motion modelling.

To accommodate the spatiotemporal information in the input, the solution turns out to be surprisingly simple. We channel-wise concatenate two consecutive frames from the video. This is inspired by many papers on video-based tasks that use a similar input formatting [55, 105]. Alternative design choices include a convolutional LSTM module [106] in the middle of the network, or a 3D convolutional module [107] after the input layer. As before, a DeeplabV3+ model is used as the shared feature extractor for different heads. Other than the previously introduced heads that produce foreground embedding, background embedding, foreground semantic segmentation, and background segmentation for image 1, we add four more heads that perform the same tasks for image 2. Figure 50 illustrates the model architecture.

To the best of our knowledge, there is no amodal segmentation-based multi-object tracking dataset. The MOTS dataset proposed in [21] comes close. They do provide segmentation masks together with bounding boxes and identity information but unfortunately it is not amodal. In CVPR 2019 two amodal segmentation datasets [108, 109] are proposed but not published at the time of writing. One is synthesized using a GTA V engine, the other annotated based on an existing dataset. They are also not for tracking, meaning there are only random images but they do not form a continuous video. We use the same method from section 3 to animate moving shapes. The shapes have three categories: circle, triangle, and rectangle. They have a randomly initiated position and velocity. When they reach

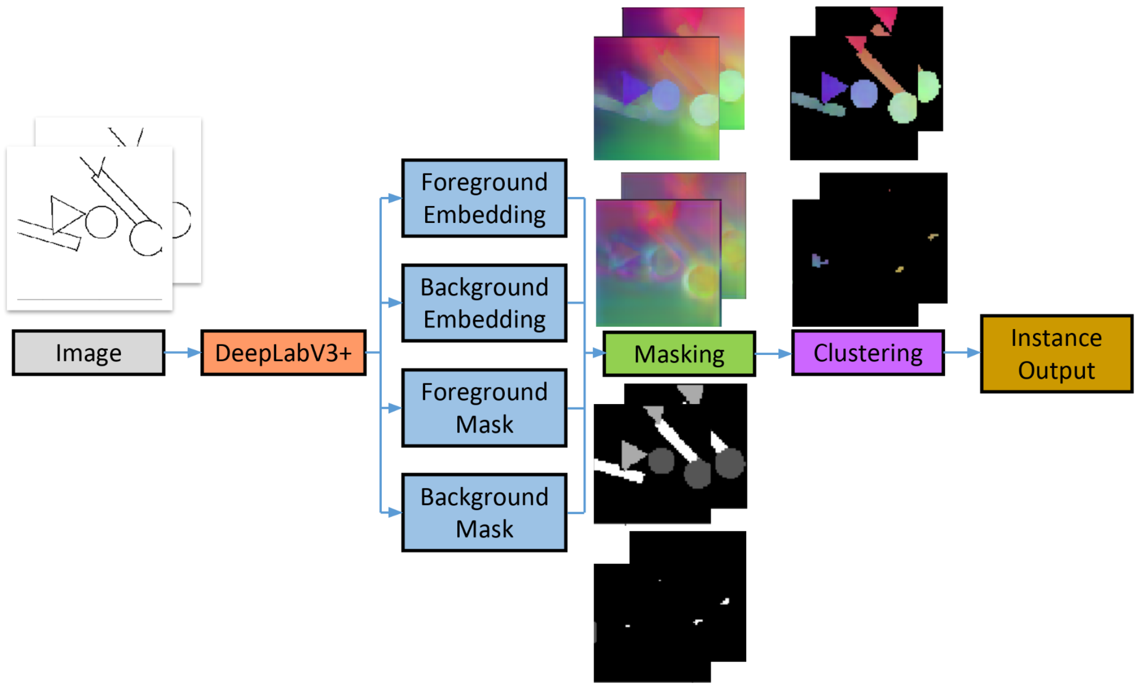


Figure 50: Proposed tracking architecture.

the boundary of the image, they bounce back without loss of speed. Each shape is on a separate depth layer, so no two shapes will collide with each other. Figure 51 illustrates an example of consecutive frames from this synthetic dataset. The ground truth information used for training has eight parts, four per image. For each image, it contains the (1) foreground instance identity mask, (2) background instance identity mask, (3) foreground semantic segmentation mask, and (4) background semantic segmentation mask.

The loss function used to train for the amodal instance segmentation task is similar to the one used in section 3. The image-pair version of it takes the embeddings in both images, pushes them together if they are from the same identity, or pulls them apart if they are from different identities.

Let $N$ be the number of ground truth instances and let $\mathcal{R}_1^n \subseteq \mathcal{I}_1$ and $\mathcal{R}_2^n \subseteq \mathcal{I}_2$ denote the set of pixels for instance $n \in \{1, \ldots, N\}$ for image 1 and 2, respectively. Also, let $\mu_n$ be the average embedding vector of all pixels in both the foreground and occlusion embeddings for instance $n$. While equation 11 and 12 stay the same, the variance loss term

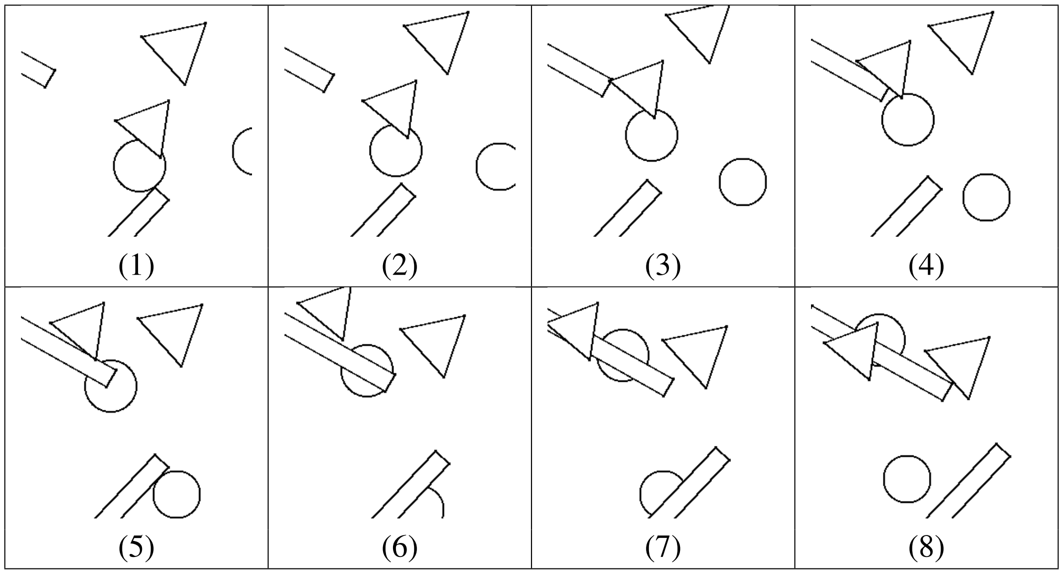


Figure 51: Sample sequence of moving shapes dataset

becomes:

$$l_{var} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{|\mathcal{R}_1^n| + |\mathcal{R}_2^n|} \left( \sum_{p \in \mathcal{R}_f^n} \left[ ||\mu_n - \mathcal{E}_f(p)|| - d_{var} \right]_+^2 + \sum_{p \in \mathcal{R}_o^n} \left[ ||\mu_n - \mathcal{E}_o(p)|| - d_{var} \right]_+^2 \right) \tag{13}$$

To perform instance segmentation and tracking, the pixel embeddings are clustered using the same post-processing algorithm described in section 3.2. The whole framework benefits from the simple and consistent setup that works very similarly compared to the segmentation version. By clustering pixels across two frames and across two layers, the model achieves simultaneous amodal instance segmentation and multi-object tracking.

Due to the format of the input, the model is technically only able to track two frames at a time. This means that if two pairs of images, $(I_0, I_1)$ and $(I_2, I_3)$, are fed into the network, then the model can only guarantee the identity consistency between $(I_0, I_1)$ or $(I_2, I_3)$, but not between $(I_1, I_2)$ or $(I_0, I_3)$. Therefore, the images are fed into with a time-step difference of 1. In other words, after the pair $(I_0, I_1)$ is processed, $I_0$ is shifted out and $I_2$ is shifted in. The next pair to be processed is thus $(I_1, I_2)$. This technique links all frames together and form long-term trajectories. Figure 52 illustrates this process.

The middle frame needs to be processed twice. Once as the second image in the first pair, then again as the first image in the second pair, so that it can link the identities of all three frames together through an intersection-over-union matching between the two versions of the mask for itself.

Experiments show that the model can produce consistent embeddings between the two frames but not across multiple frames when objects move too much. Figure 55 shows this phenomenon. Naively calculating embedding differences on a global scale and coming up with an average representation for a certain object will fail.

## 4.3 Implementation Details

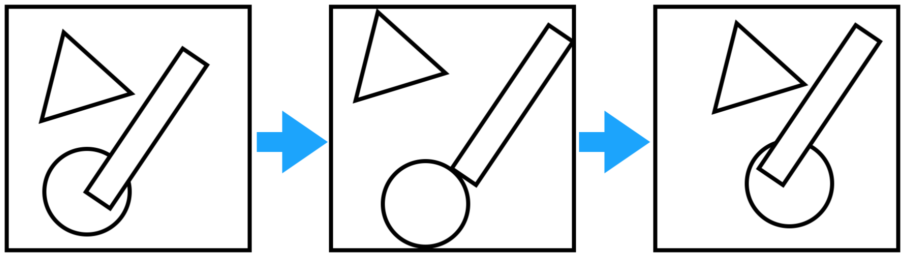
Similar to the amodal instance segmentation model proposed in section 3, the MOT model uses DeepLabV3+ as the shared feature extractor. The model adds four more heads to produce foreground instance embedding, background instance embedding, foreground semantic segmentation, and background segmentation for both images, therefore it has $2 \times 4 = 8$ sub-task heads. Previous research has shown that training multitasking model outperforms models trained on individual tasks independently [103]. By jointly training all these tasks together, the model is forced to learn more robust internal representations.

The loss function has two terms. One is categorical cross-entropy for the semantic segmentation, denoted as $L_{semantic}$, the other is the discriminative loss function for MOT, denoted as $L_{instance}$. The total loss is therefore

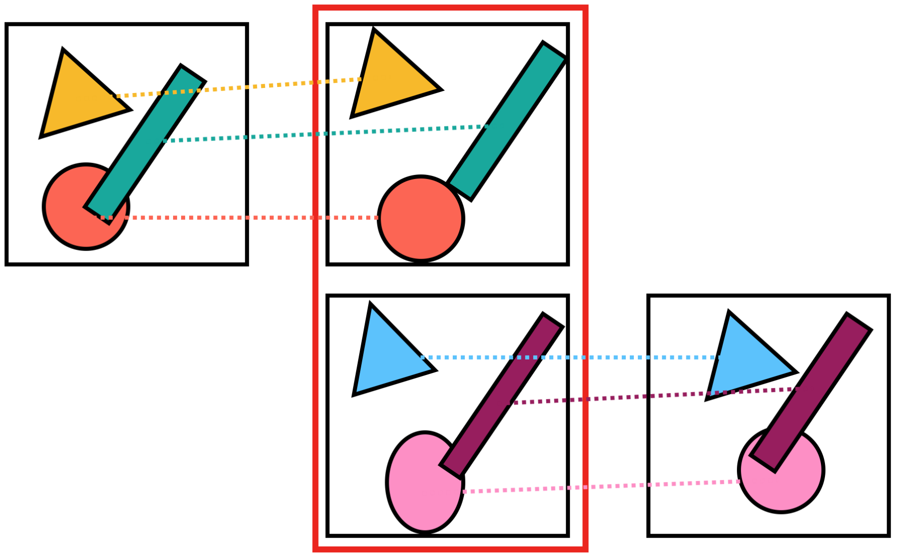
$$L = \alpha L_{semantic} + \beta L_{instance} \,.$$

$L_{instance}$ is modified to take in all the pixels across two frames and two layers together if they belong to the same instance according to the ground truth. The coefficients $\alpha$ and $\beta$ are both set to 1.

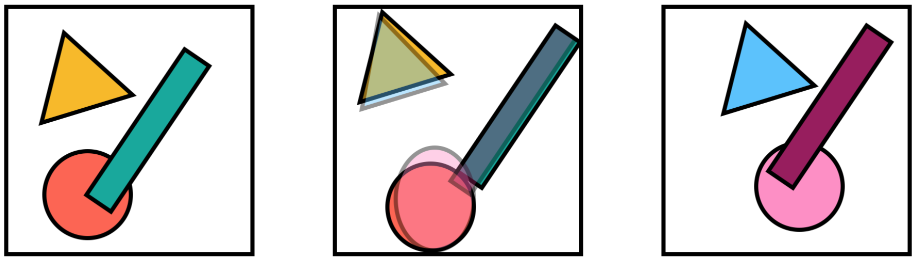
For comparison, the model from [20] is selected because it has top performance on common multi-object tracking benchmarks. Their Joint Detection and Embedding (JED)



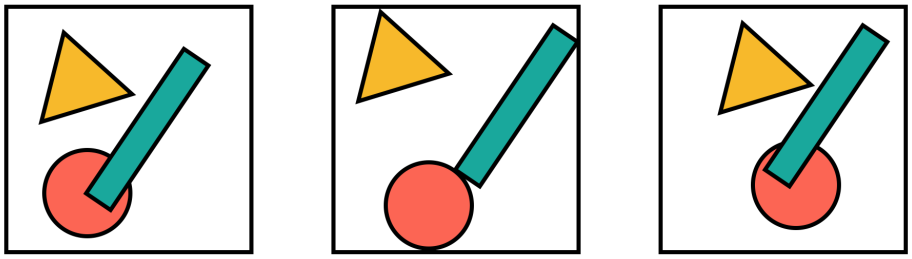
(a) 3 consecutive frames as input



(b) neighboring two frames can be linked by simply clustering the embeddings



(c) The middle frame plays the crucial role of linking identities together.



(d) All three frames are successfully linked together.

Figure 52: Multi-frame tracking process

model simultaneously outputs the bounding box, classification, and appearance embedding through a YOLO-like structure. They also take advantage of multi-scale features and produces predictions at each scale. The JDE model achieves real-time processing with 18.8 FPS and MOTA=64.6% compared to Faster R-CNN + QAN with a slightly higher MOTA=66.1% but only 6 FPS on MOT-16 test set. Figure 53 illustrates the architecture of the JDE model.

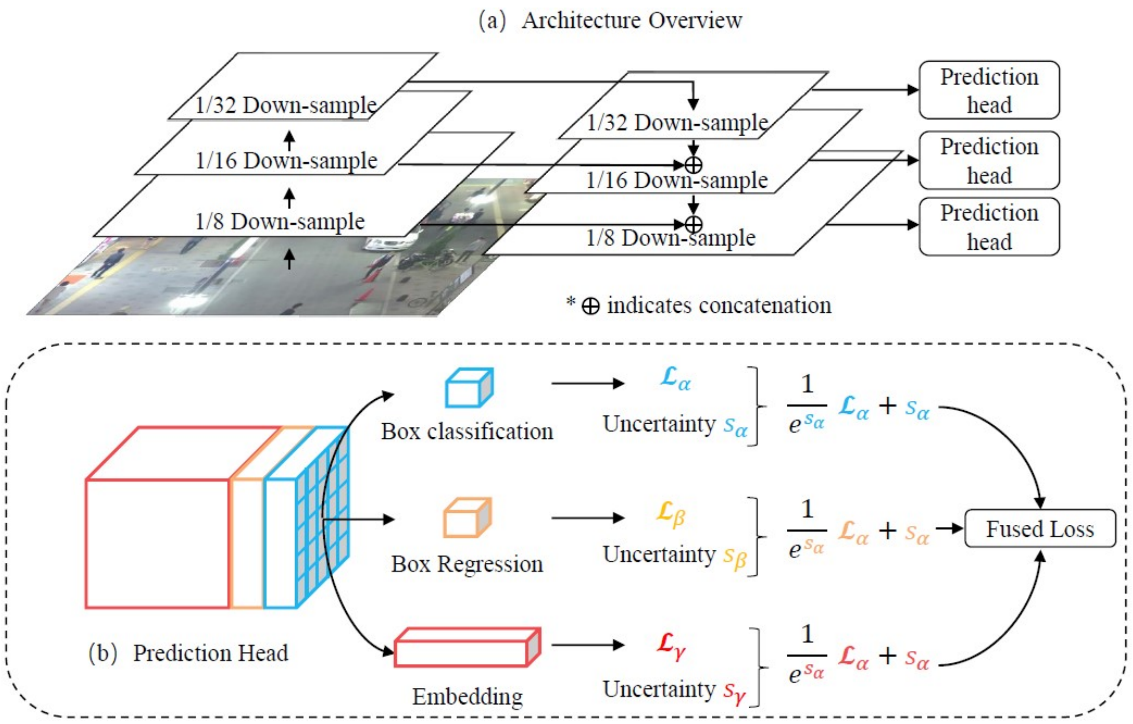


Figure 53: Architecture of JDE [20]

The JDE model is a classic example of bounding box based top-down MOT models. It predicts bounding boxes, classifies them, and produces appearance embeddings for each image region within the bounding boxes. The model is tailored towards pedestrian tracking because that is the only theme of multi-object tracking datasets at the moment. The model assumes that most objects will have a 3:1 aspect ratio because human beings are usually standing up-right. The model also assumes that each object looks different from other objects because each person is supposed to have a distinct choice of clothing. Interestingly, very few existing methods address the problem of multi-class multi-object tracking. This means most existing models assume that all objects being tracked belong to the same semantic class. In this case, it is pedestrians. To the best of our knowledge, there is also no existing method that address amodal multi-class multi-object tracking because there is simply no dataset for it. The JDE model is the closest and most promising model we can find in the literature in terms of performance and comparability to our research.

## 4.4 Experiments

Both the proposed embedding model and the JDE model are trained, validated, and tested on identical dataset settings. The train set consists of 500 sequences of moving shapes, 100 frames per sequence. The val and test set each has 5 sequences, 100 frames per sequence. The train, val, and test set are initialized with the random seed = 1, 2, 3 respectively for reproducibility purpose. This also ensures that the train, val, and test set do not overlap.

The embedding model uses Adam optimizer with the learning rate = $10^{-4}$, batch size = 1. The JDE model uses SGD optimize with the learning rate = $10^{-2}$, batch size = 1, which is the same as the setting in the original paper. Each model trains for 10 epochs, so the total number of training samples that each model sees is $500 \times 100 \times 10 = 500,000$.

For evaluation, we report IDF1 and MOTA metrics. The definition of these metrics can be found in section 2.3. IDF1 and MOTA are considered the two most common metrics to report on major benchmarks such as MOT-16. The original paper for JDE also reports these two metrics against other methods.

The performance of two models on each sequence is listed in detail in table 4.4. The overall performance is also calculated by averaging the scores on each sequence. Averaging is valid when the number of frames and identities per sequence is identical, which is the case in the synthetic dataset.

As shown in Table 4.4, our model outperforms the JDE model significantly. Figure 54 and Figure 55 show some of the tracking results from the test sequences.

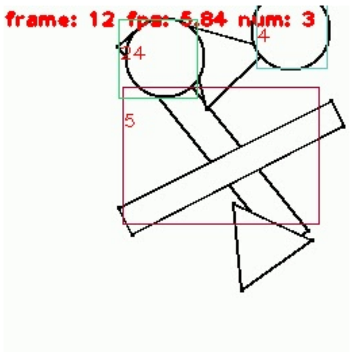
To further analyze the behavior and performance of the embedding tracking method,

Table 5: Performance of our proposed embedding model and the JDE model on 5 sequences of moving shapes

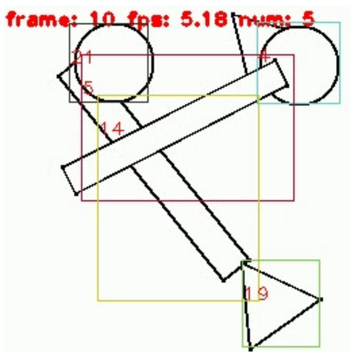
| Model | Sequence | IDF1 | MOTA |
|---|---|---|---|
| Ours | 1 | 33.8% | 67.0% |
| | 2 | 52.6% | 81.3% |
| | 3 | 68.1% | 88.7% |
| | 4 | 77.5% | 96.2% |
| | 5 | 31.0% | 63.0% |
| | Overall | **52.60%** | **79.24%** |
| JDE | 1 | 10.6% | 29.8% |
| | 2 | 15.1% | 30.1% |
| | 3 | 11.7% | 20.2% |
| | 4 | 12.5% | 40.3% |
| | 5 | 28.7% | 58.8% |
| | Overall | 15.72% | 35.84% |

Figure 56 and Figure 57 illustrate long sequences of tracking results produced by our MOT embedding model.

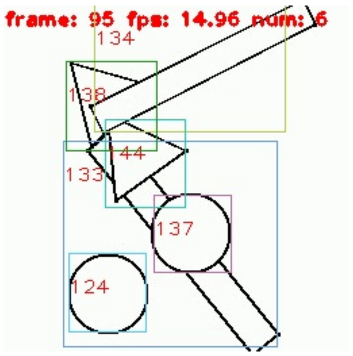
To push the limit of embedding based tracking, we also tentatively increased the difficulty of the synthesized shapes dataset by rotating each shape during the sequence by a random angular velocity. Figure 58 and figure 59 show qualitative results of our model on these sequences.



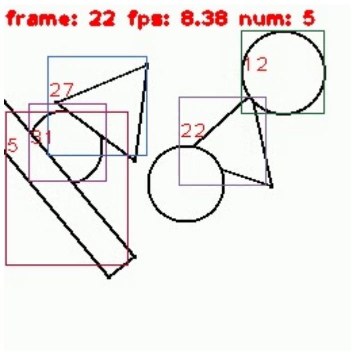
(a)



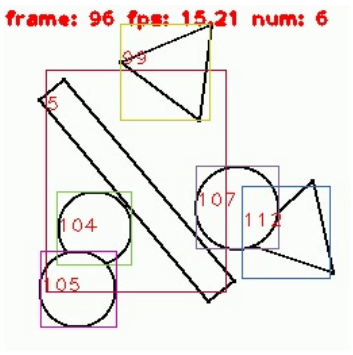
(b)



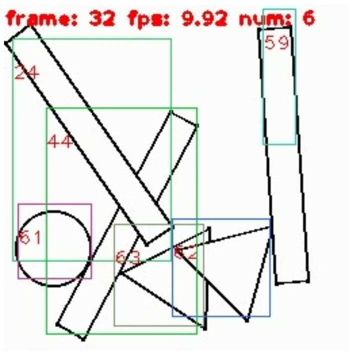
(c)



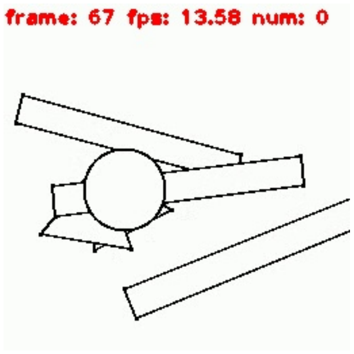
(d)



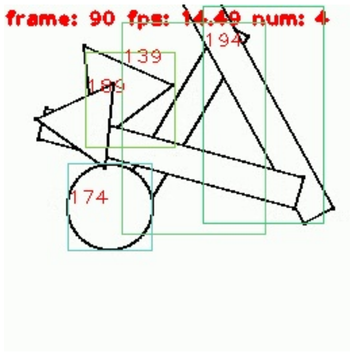
(e)



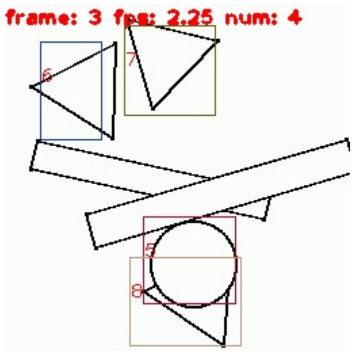
(f)



(h)



(i)



(j)

Figure 54: Sample results of the JDE embedding model on difference sequences.

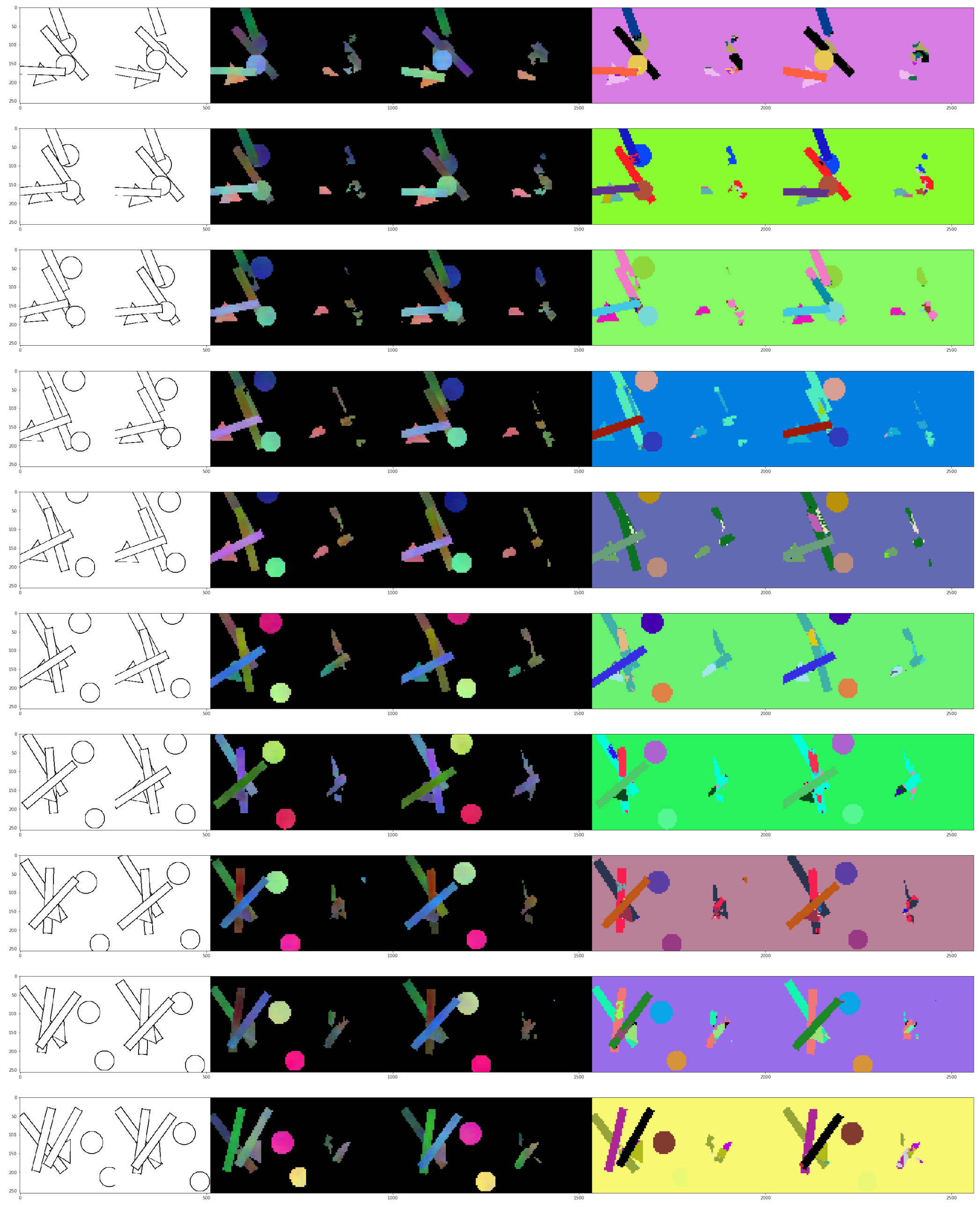


Figure 55: Sample results from the MOT embedding model. From left to right: image pair, masked foreground embedding and background embedding for image pair, foreground and background instance clustering for image pair

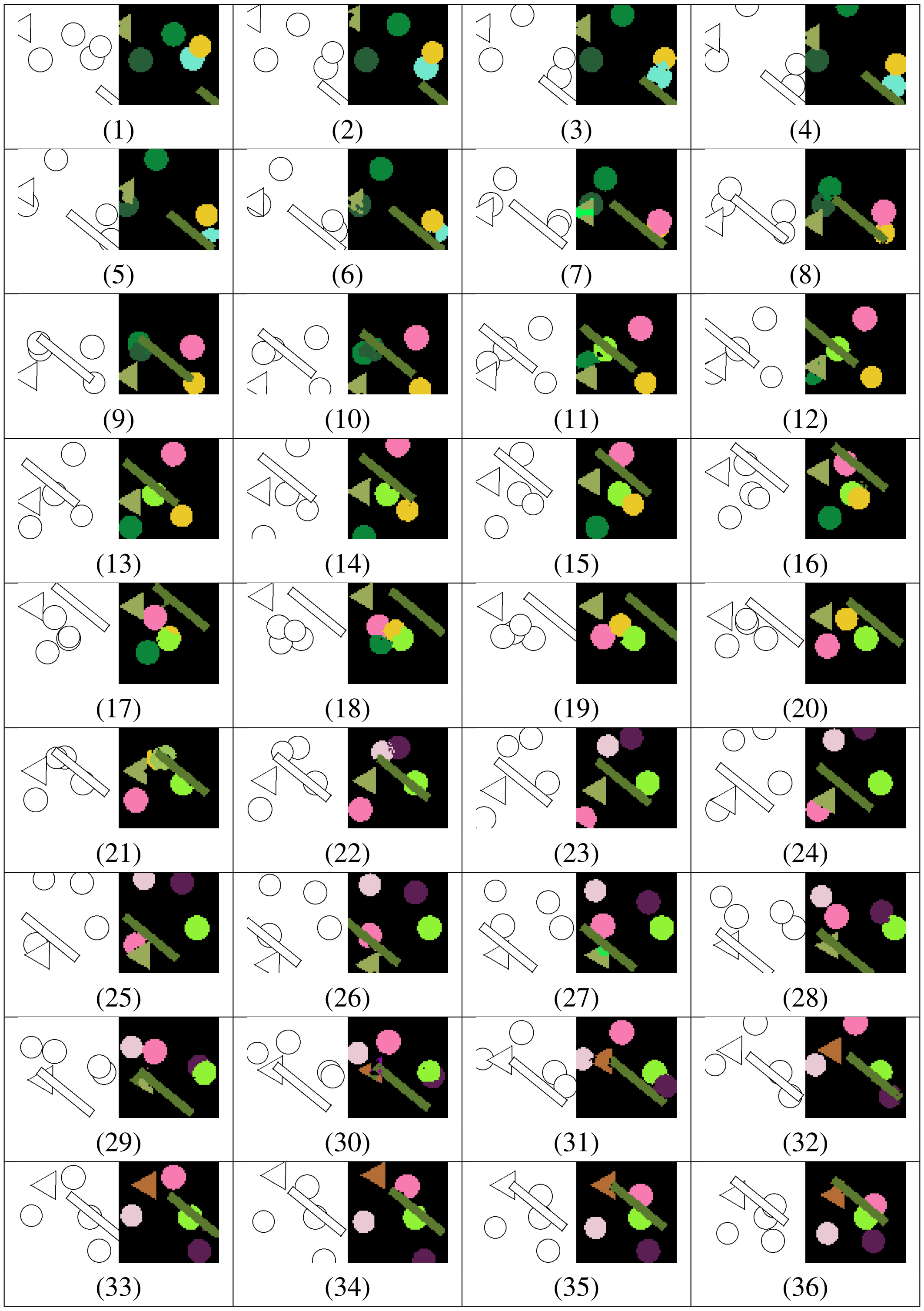


Figure 56: Sequence 1 of tracking results produced by our MOT embedding model

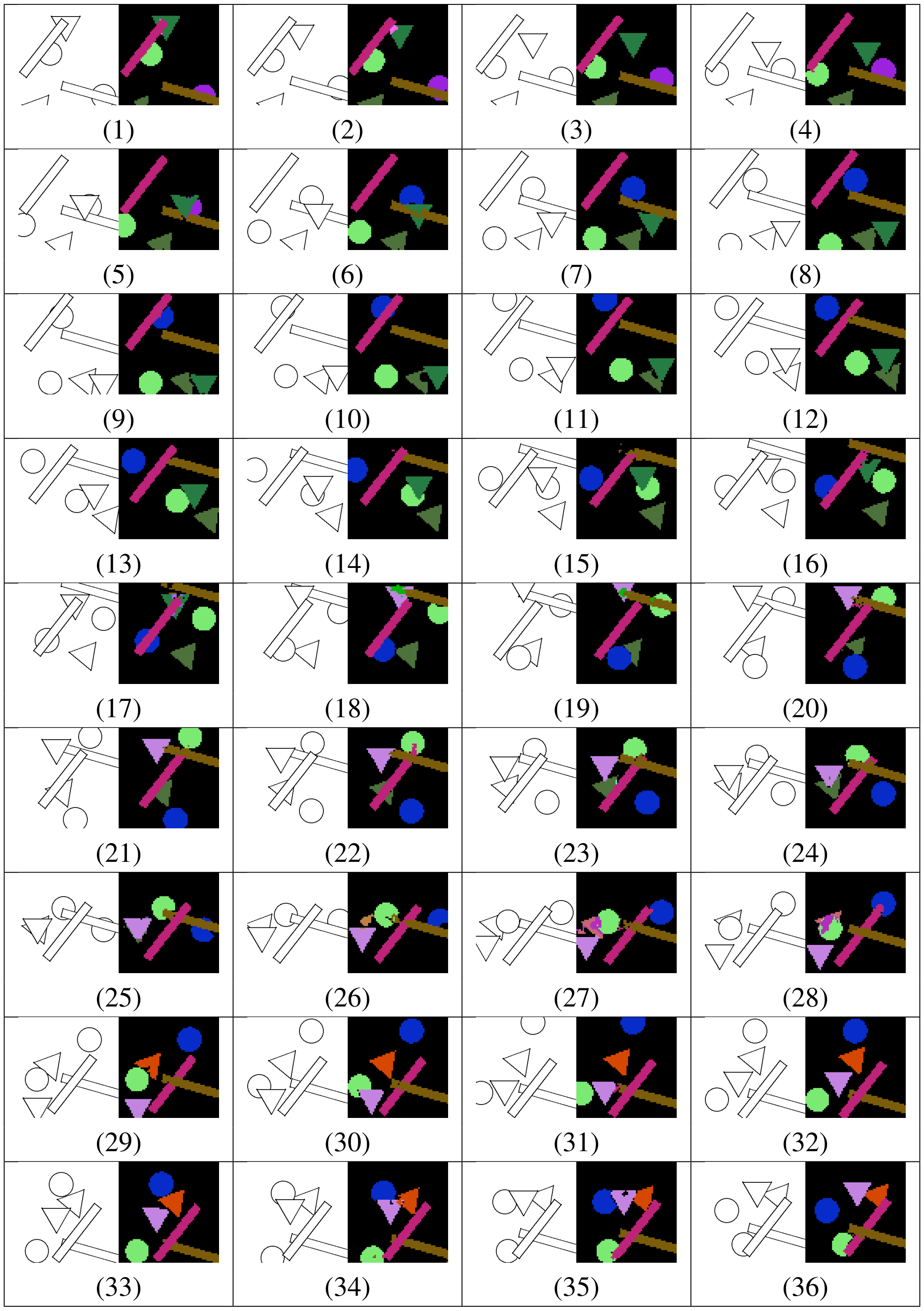


Figure 57: Sequence 2 of tracking results produced by our MOT embedding model

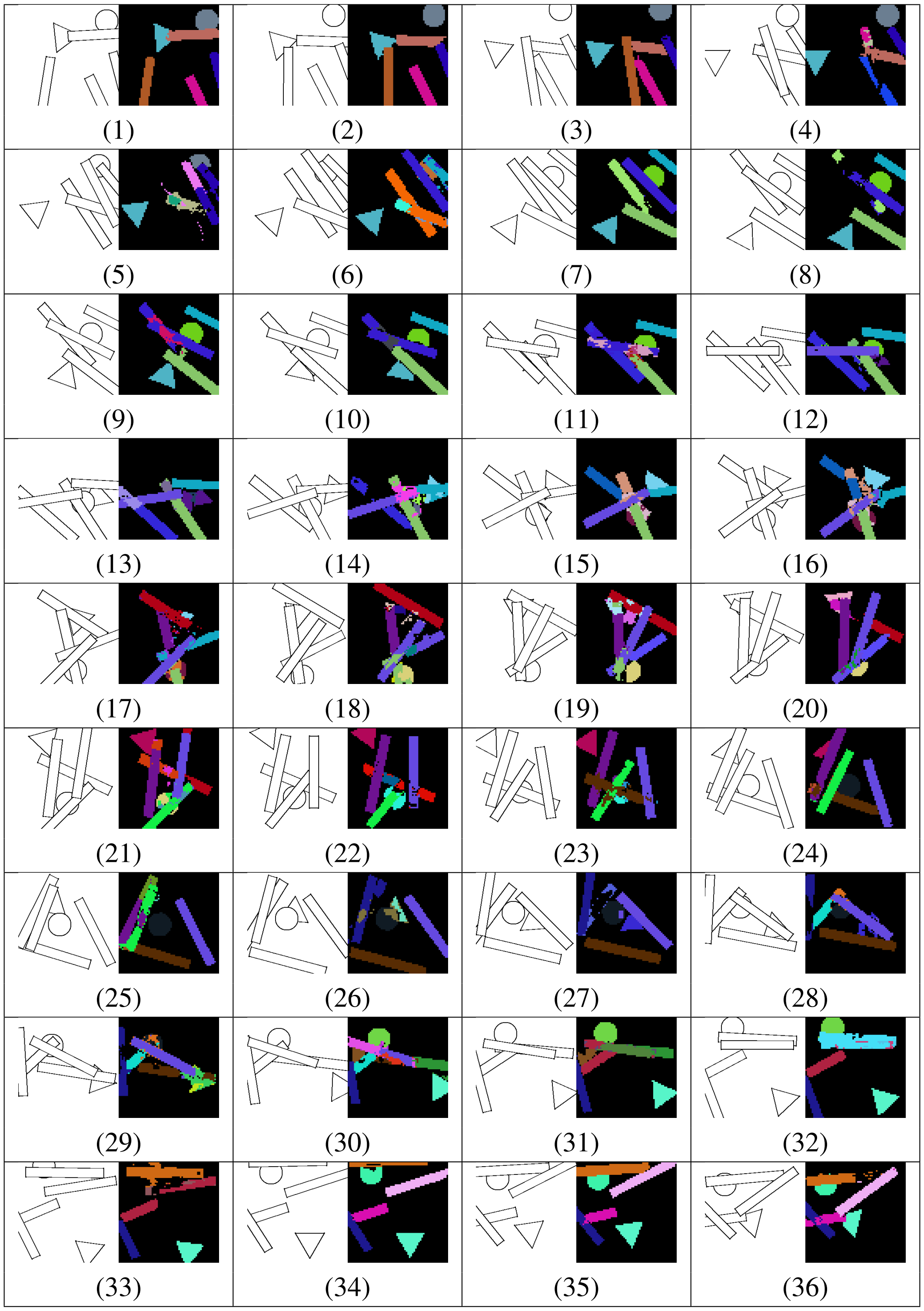


Figure 58: Qualitative results of the embedding based tracking model on rotate shapes, sequence 1.

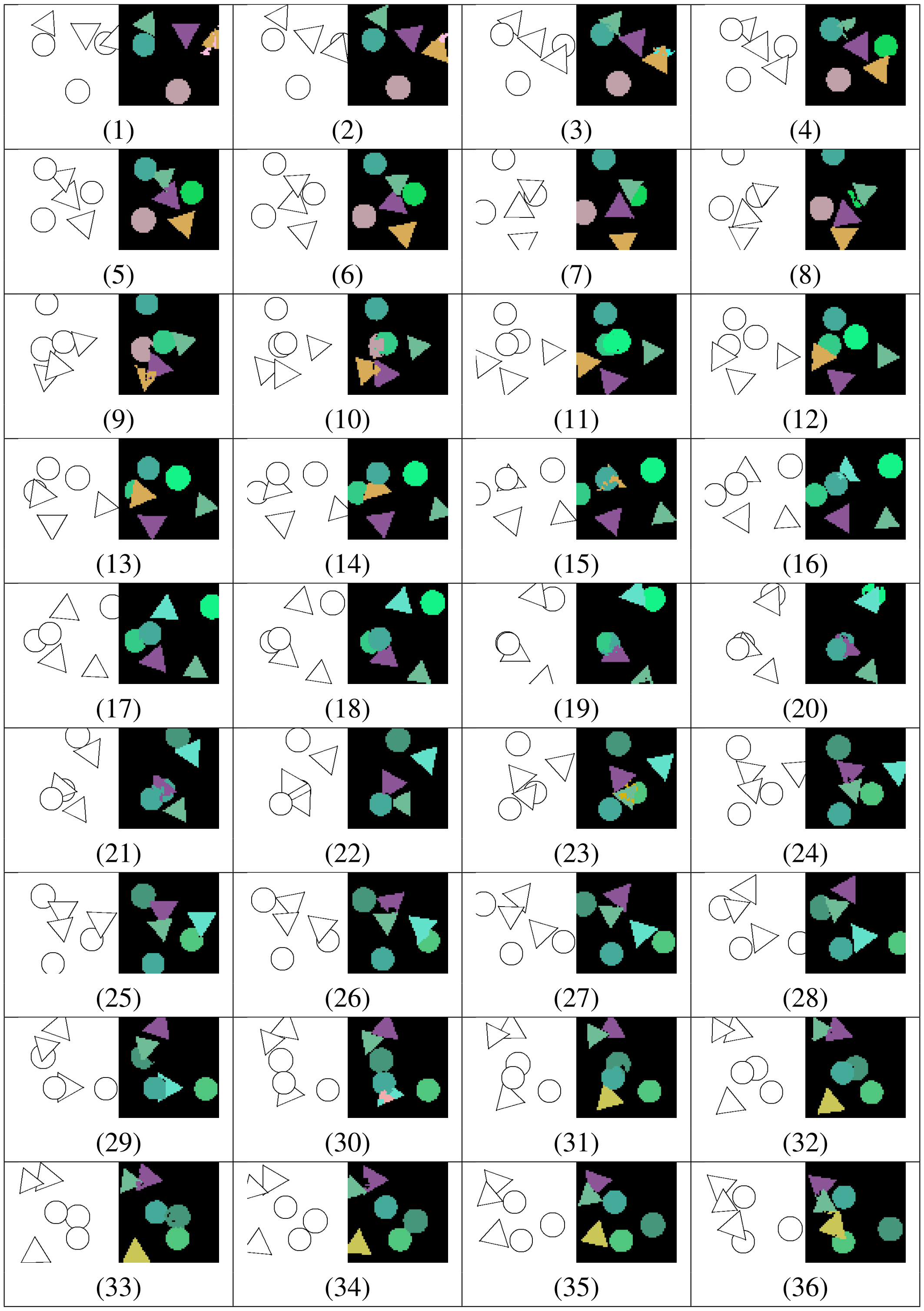


Figure 59: Qualitative results of the embedding based tracking model on rotate shapes, sequence 2.

## 4.5 Discussion

It can be observed from Figure 55 and Figure 54 that the embeddings are very good at detecting depth information and describe the exact spatial layout of shapes, whereas bounding boxes struggle with overlapping shapes. For example, in Figure 54 (a), bounding box 5 is on top of both sticks so the two sticks cannot be distinguished from bounding boxes. Also, the two bounding boxes are overlapping so much that one of them is eliminated by non-max suppression.

In the case of JDE models, there also seem to be three big issues with their approach. They will be discussed in depth in the next three paragraphs.

(1) The object detector is not very good at detecting objects of large scales, namely the sticks. This can be observed in Figure 54 (f)(h)(j). The sticks take 80% of the screen width or height, which seems to exceed the receptive field of the network. However, the anchors in their models are clearly defined to handle these scales. We suspect that it is because the sticks tend to have unusual aspect ratios when the other shapes occupy relatively square areas. The performance of the object detector seems very unstable. In some cases, all shapes are detected correctly, such as 54 (c)(e), which shows that the model is capable of handling objects at this scale; but in other cases, the model does so badly that it fails to detect all the objects, such as 54 (h).

(2) The appearance embedding module fails to capture meaningful embeddings for the shapes because it assumes that the shapes each have a unique look. Instead, the shapes all have black border and white color, on a white background. This makes the appearance embedding module struggle especially hard. As a result, it becomes very easy for the model to mix up the embeddings of different shapes. The tracker ends up switching identities and creating new identities constantly.

(3) The usage of Kalman filter is a bad choice. The JDE model holds the assumption that objects will leave the scene once they cross the border. The Kalman filter tracks the motion of objects and predicts that the objects will move of the frame given their current position

and velocity. However, because the boundaries are set up as barriers, the shapes follow the law of physics correctly and bounce back into the scene. It is indeed observed that the shapes get assigned new identities once they bounce back from the border, indicating that their current position is far away from their predicted position.

As for our proposed MOT embedding model, it can be observed that it gets the masks of the shapes mostly right, but when shapes overlap heavily with each other, sometimes there is confusion in the embedding about which shape belongs to which in the next frame, especially when similar looking shapes are clustered together. Due to the two-frame input format constraint, there is not enough temporal context to decide whether which way a certain shape is heading when there are similar ones around it. This is the major contributor to performance loss for our model.

This problem can be seen in Figure 56 (6)(7) where the old circle loses its identity and gets assigned as a new track. Remarkably, the orange circle does not lose its identity. It is correctly tracked from frame 1 to 21. Even though it is just barely visible in frame 7, its invisible part is still estimated correctly in the second layer to preserve its identity.

Figure 57 showcases another long sequence of tracking results. We can observe that the two sticks and two circles are tracked successfully throughout the sequence, with two new identities being incorrectly assigned on the triangles at frame 18 and 26.

# 5 Conclusion

This thesis explores the effectiveness and possibilities of deep pixel embeddings. It starts by examining the ineffectiveness of bounding boxes in heavily crowded scenes. The rectangular shape of bounding boxes makes them an awkward and cumbersome descriptor for fine-grain object information. This leads to the motivation for a method without bounding boxes at all. The idea of assigning a pixel embedding to each visible pixel for clustering, was originally used for modal instance segmentation. We take it further and develop the idea into amodal instance segmentation by explicitly allowing layers of masks to handle occlusion. The idea is taken further still into amodal multi-object tracking by enforcing embedding consistency between two frames and across both layers of masks. The framework is simple yet powerful. There is neither motion model nor data association, thus no parameters to tune. The amodal masks allow identity matching even when objects are hardly visible. The performance of the embedding models is evaluated on a synthetic shapes dataset and compared against state-of-the-art models in the literature. We hope this thesis inspires other researchers to further explore the potential of bounding box free, bottom-up methods for segmentation and tracking.

The effectiveness of embedding is promising. Through a synthetic shapes dataset with challenging object layouts and homogeneous colors, the embedding method shows its capability of modelling the surrounding context of objects and assigning distinctly separated embeddings that are uniform within object boundaries. It will be worthwhile to investigate the content of the embeddings and observe their relationship with the spatial location, appearance, and classification of the objects.

The embedding model is not perfect. There are still things to be improved upon. First, the model is limited by the number of masks explicitly constructed during the building stage of the neural network. If test scenes become more complicated and requires more layers, the theoretical upper-bound for the performance of the model will be limited. Future work should explore adaptive growth for the model. Second, the model cannot handle complete occlusion. This means if an object completely disappears for a frame or more, its identity is permanently lost because the key frame that links identities together will not find its mask. This disadvantage can be mitigated by the classic approach of assuming that the position of the object does not change during a time window when it is missing. The buffer time will loosen up the constraint on successfully detecting objects at every frame.

Given the current design of the loss function, the embedding seems to be correlated with the position of the objects. This makes sense since the network will take the shortest path to its objective whenever possible. One of the properties of this embedding scheme is that the embeddings change smoothly as the objects' locations change continuously, and it works well for instance segmentation. However, ideally the embeddings should capture more complicated features so that the embeddings for each instance is somewhat inherently different from other instances regardless of the location. This requires a more complicated loss function and different data formatting. Future work should explore this route.

Ultimately, the ideal embedding model should be able to output embeddings that not only distinctly and uniformly describe the outline of each instance, but also their classification, orientation, and location such that the entire scene can be perfectly reconstructed using only embeddings. This will require more complicated loss functions and potentially a change in the model architecture.

# References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[2] P. Hu and D. Ramanan, "Finding tiny faces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 951–959, 2017.

[3] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, pp. 2980–2988, IEEE, 2017.

[5] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: real-time multi-person 2d pose estimation using part affinity fields," *arXiv preprint arXiv:1812.08008*, 2018.

[6] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits, 1998," *URL http://yann. lecun. com/exdb/mnist*, vol. 10, p. 34, 1998.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[8] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2337–2346, 2019.

[9] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 85–100, 2018.

[10] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[12] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[13] S. Xie, A. Kirillov, R. Girshick, and K. He, "Exploring randomly wired neural networks for image recognition," *arXiv preprint arXiv:1904.01569*, 2019.

[14] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[16] K. Ehsani, R. Mottaghi, and A. Farhadi, "Segan: Segmenting and generating the invisible," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6144–6153, 2018.

[17] L. Qi, L. Jiang, S. Liu, X. Shen, and J. Jia, "Amodal instance segmentation with kins dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3014–3023, 2019.

[18] Y.-T. Hu, H.-S. Chen, K. Hui, J.-B. Huang, and A. G. Schwing, "Sail-vos: Semantic amodal instance level video object segmentation-a synthetic dataset and baselines," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3105–3115, 2019.

[19] S. Manen, M. Gygli, D. Dai, and L. Van Gool, "Pathtrack: Fast trajectory annotation with path supervision," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 290–299, 2017.

[20] Z. Wang, L. Zheng, Y. Liu, and S. Wang, "Towards real-time multi-object tracking," 2019.

[21] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "Mots: Multi-object tracking and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7942–7951, 2019.

[22] L. Chen, H. Ai, Z. Zhuang, and C. Shang, "Real-time multiple people tracking with deeply learned candidate selection and person re-identification.," in *ICME*, pp. 1–6, 2018.

[23] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 801–818, 2018.

[24] T.-J. Yang, M. D. Collins, Y. Zhu, J.-J. Hwang, T. Liu, X. Zhang, V. Sze, G. Papandreou, and L.-C. Chen, "Deeperlab: Single-shot image parser," *arXiv preprint arXiv:1902.05093*, 2019.

[25] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[26] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, "Synthesizing obama: learning lip sync from audio," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 95, 2017.

[27] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[28] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.

[29] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[31] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[32] G. B. Huang and E. Learned-Miller, "Labeled faces in the wild: Updates and new reporting procedures," *Dept. Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep*, pp. 14–003, 2014.

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[34] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[35] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[36] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[37] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[38] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[39] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, "Mask scoring r-cnn," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6409–6418, 2019.

[40] Z. Hayder, X. He, and M. Salzmann, "Boundary-aware instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5696–5704, 2017.

[41] B. De Brabandere, D. Neven, and L. Van Gool, "Semantic instance segmentation with a discriminative loss function," in *Deep Learning for Robotic Vision, workshop at CVPR 2017*, pp. 1–2, CVPR, 2017.

[42] S. Kong and C. C. Fowlkes, "Recurrent pixel embedding for instance grouping," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9018–9028, 2018.

[43] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy, "Semantic instance segmentation via deep metric learning," *arXiv preprint arXiv:1703.10277*, 2017.

[44] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9404–9413, 2019.

[45] Q. Li, A. Arnab, and P. H. Torr, "Weakly-and semi-supervised panoptic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 102–118, 2018.

[46] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun, "Upsnet: A unified panoptic segmentation network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8818–8826, 2019.

[47] D. de Geus, P. Meletis, and G. Dubbelman, "Panoptic segmentation with a joint semantic and instance segmentation network," *arXiv preprint arXiv:1809.02110*, 2018.

[48] Y. Li, X. Chen, Z. Zhu, L. Xie, G. Huang, D. Du, and X. Wang, "Attention-guided unified network for panoptic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7026–7035, 2019.

[49] H. Liu, C. Peng, C. Yu, J. Wang, X. Liu, G. Yu, and W. Jiang, "An end-to-end network for panoptic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6172–6181, 2019.

[50] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-nms–improving object detection with one line of code," in *Proceedings of the IEEE international conference on computer vision*, pp. 5561–5569, 2017.

[51] Y. He, X. Zhang, M. Savvides, and K. Kitani, "Softer-nms: Rethinking bounding box regression for accurate object detection," *arXiv preprint arXiv:1809.08545*, 2018.

[52] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.

[53] B. D. Lucas, "Generalized image matching by the method of differences.," 1986.

[54] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766, 2015.

[55] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2462–2470, 2017.

[56] P. Liu, M. Lyu, I. King, and J. Xu, "Selflow: Self-supervised learning of optical flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4571–4580, 2019.

[57] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, "Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras," *arXiv preprint arXiv:1904.04998*, 2019.

[58] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang, "Segflow: Joint learning for video object segmentation and optical flow," in *Proceedings of the IEEE international conference on computer vision*, pp. 686–695, 2017.

[59] R. Alp Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7297–7306, 2018.

[60] G. Moon, J. Y. Chang, and K. M. Lee, "Posefix: Model-agnostic general human pose refinement network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7773–7781, 2019.

[61] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros, "Everybody dance now," *arXiv preprint arXiv:1808.07371*, 2018.

[62] S. Ullman, "The interpretation of structure from motion," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 203, no. 1153, pp. 405–426, 1979.

[63] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[64] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[65] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[66] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, p. 3, 2013.

[67] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[68] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

[69] S. Linnainmaa, "The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors," *Master's Thesis (in Finnish), Univ. Helsinki*, pp. 6–7, 1970.

[70] S. Linnainmaa, "Taylor expansion of the accumulated rounding error," *BIT Numerical Mathematics*, vol. 16, no. 2, pp. 146–160, 1976.

[71] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[72] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *arXiv preprint arXiv:1904.09237*, 2019.

[73] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[74] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[75] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, 2013.

[76] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[77] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[78] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[79] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[80] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[81] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[82] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[83] K. Li and J. Malik, "Amodal instance segmentation," in *European Conference on Computer Vision*, pp. 677–693, Springer, 2016.

[84] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.

[85] Y. Zhu, Y. Tian, D. Metaxas, and P. Dollár, "Semantic amodal segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1464–1472, 2017.

[86] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2359–2367, 2017.

[87] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes, "Layered object detection for multi-class segmentation," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3113–3120, June 2010.

[88] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes, "Layered object models for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 1731–1743, Sept 2012.

[89] Y. T. Chen, X. Liu, and M. H. Yang, "Multi-instance object segmentation with occlusion handling," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3470–3478, June 2015.

[90] J. Uhrig, M. Cordts, U. Franke, and T. Brox, “Pixel-level encoding and depth layering for instance-level semantic labeling,” in *German Conference on Pattern Recognition*, pp. 14–25, Springer, 2016.

[91] J. Ferryman and A. Shahrokni, “Pets2009: Dataset and challenge,” in *2009 Twelfth IEEE international workshop on performance evaluation of tracking and surveillance*, pp. 1–6, IEEE, 2009.

[92] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, “Motchallenge 2015: Towards a benchmark for multi-target tracking,” *arXiv preprint arXiv:1504.01942*, 2015.

[93] U. Iqbal, A. Milan, and J. Gall, “Posetrack: Joint multi-person pose estimation and tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2011–2020, 2017.

[94] M. Naphade, M.-C. Chang, A. Sharma, D. C. Anastasiu, V. Jagarlamudi, P. Chakraborty, T. Huang, S. Wang, M.-Y. Liu, R. Chellappa, *et al.*, “The 2018 nvidia ai city challenge,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 53–60, 2018.

[95] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[96] H. W. Kuhn, “The hungarian method for the assignment problem,” *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[97] K. Bernardin and R. Stiefelhagen, “Evaluating multiple object tracking performance: the clear mot metrics,” *Journal on Image and Video Processing*, vol. 2008, p. 1, 2008.

[98] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, “Performance measures and a data set for multi-target, multi-camera tracking,” in *European Conference on Computer Vision*, pp. 17–35, Springer, 2016.

[99] Y. Xu, Y. Ban, X. Alameda-Pineda, and R. Horaud, “Deepmot: A differentiable framework for training multiple object trackers,” *arXiv preprint arXiv:1906.06618*, 2019.

[100] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, “Tracking without bells and whistles,” *arXiv preprint arXiv:1903.05625*, 2019.

[101] K. Sohn, “Improved deep metric learning with multi-class n-pair loss objective,” in *Advances in Neural Information Processing Systems*, pp. 1857–1865, 2016.

[102] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[103] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7482–7491, 2018.

[104] S. Tang, M. Andriluka, B. Andres, and B. Schiele, "Multiple people tracking by lifted multicut and person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3539–3548, 2017.

[105] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki, "Sfm-net: Learning of structure and motion from video," *arXiv preprint arXiv:1704.07804*, 2017.

[106] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, pp. 802–810, 2015.

[107] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[108] L. Qi, L. Jiang, S. Liu, X. Shen, and J. Jia, "Amodal instance segmentation with kins dataset," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[109] Y.-T. Hu, H.-S. Chen, K. Hui, J.-B. Huang, and A. G. Schwing, "Sail-vos: Semantic amodal instance level video object segmentation - a synthetic dataset and baselines," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.