Future Object Segmentation for Complex Correlated Motions

Student: Pierre Eugene Valassakis
Supervisor: Prof. Gabriel Brostow

MSc Machine Learning

September 2018

This report is submitted as part requirement for the MSc Degree in Machine Learning at University College London. It is substantially the result of my own work except where explicitly indicated in the text.

Department of Computer Science

University College London

Abstract

Predicting the future is an important component of true artificial intelligence, allowing systems to anticipate events and act accordingly. In computer vision the task is generally formulated as inferring a future representation from past images or frames. Practical applications also span a wide range, with models capable of predicting the future being a natural fit in contexts such as robotics or autonomous driving. In its richest form, the task consists of rendering full RGB images from past frames. This has proven a substantial challenge however, and many alternative formulations attempt to make predictions in a simpler and more constrained space. A recent thread that has gained considerable momentum consists of predicting future semantic masks. We tackle this problem under the light of object segmentation.

We develop models and create datasets that are fit to the challenge, and investigate their behaviour to gain deeper insights. We first introduce a new data bank, CorrelatedMotions. We argue it is more suited to the task of future image segmentation than Cityscapes, which is currently often used for benchmarking. We also develop a model for inference, which includes a bespoke architecture that we name SpatialNet. We evaluate this model on CorrelatedMotions, and show it overall outperforms simpler baselines. Finally, we delve deeper into our model through a series of experiments. We extend the scope of our task to future predictions in variable times, and work towards a deeper understanding on the mechanics responsible for the behaviours we observe.

The code for this project can be found in the following repository:
https://github.com/eugval/Motion_Prediction

Acknowledgements

To my supervisor, Prof. Gabriel Brostow, for his support, advice and help.

To Stephan Garbin, for his advice and support.

To James R. Watson, for assisting with the data gathering by starring in the Football dataset.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Predicting the future is a crucial part of intelligence, and allows humans to anticipate for events to come, delay gratification and make decisions accordingly [94, 62]. It is therefore natural that the artificial intelligence community has taken interest to the task. Systems capable of reliably anticipating future events could indeed have profound ramifications in many applications in the field [95, 18, 57]. Robots capable of foresight might start being reliable every day companions; and autonomous cars would be able to anticipate collisions and react accordingly [95, 18, 57, 45, 56, 100]. At a lower level, model-based reinforcement learning would benefit from a robust and accurate model, capable of anticipating the real environment's behaviour [95].

In the computer vision community, predicting the future takes the form of anticipating what is to come given a video or image. Some approaches attempt to predict actions given a scene [86, 38, 53, 39, 98]. Others try to infer the movement in the image through optical flow [71, 100, 99]. Arguably the richest form of such prediction would render the full future image in RGB space [101, 58, 81, 93, 48]. This task has been considerably challenging however [93, 101, 58], and in [57] Luc *et al.* argue that predicting directly at the semantic level is more successful than first obtaining a full rendering of the future and then understanding this prediction. In fact, this paper sparked the recent approach of predicting the future as an image segmentation [45, 8, 63, 56]. Such predictions take the form of a semantic map [45, 8, 63, 57], where each pixel in the predicted future is assigned one of many classes, or an instance segmentation map [56], where each instance in these classes can be identified individually.

In our approach, we subscribe to the idea of predicting the future in the semantic level. Nonetheless, we differentiate ourselves from other works in the future image segmentation space by treating the task as an object segmentation problem. In other words, we select a single instance within a target frame in a video, and aim to predict its dense segmentation mask at a time in the future. This has the advantage of being conceptually cleaner than previous methods. With semantic segmentation, there is no way to differentiate between different objects in the same class, which is an issue for applications requiring actions with respect to instances [56]. Furthermore, to the best of our knowledge, attempts in the in

the instance segmentation space lack the ability to link the instances temporarily [56]. In other words, an additional tracking-by-detection [83, 7, 104] post-processing system would need to be applied to the predictions to link them frame-by-frame. Our approach works seamlessly, as it consists of selecting a particular object in the input and predicting its future mask in the output. As an interesting note, a particular inspiration in crafting our approach was the idea of video synthesis [42, 87, 23, 97, 47]. This is a more niche application than the ones previously mentioned. In the imagined scenario, a scene would be provided, along with splices of different agents. Sliding a particular splice into the scene, an interesting application would then predict how these agents can behave, generating frames in the process. In doing so, being able to select a single agent and predict its behaviour is important, which is the main idea our model builds around.

1.2 Objective and Contributions

As briefly mentioned, we aim to construct a model that is capable of future object segmentation. That is, given a series of input frames and the segmented masks of a single agent in these frames, we want to predict the segmentation mask of this agent in a future frame.

Naturally following from this objective, our contributions are four-fold:

1. We introduce a new data bank, CorrelatedMotions, that includes several datasets. We argue that these are better suited for benchmarking in the task of future image segmentation than the currently used Cityscapes [14, 16] dataset.

2. We cast the task of future image segmentation under the light of object segmentation, with a single instance selected for prediction. To the best of our knowledge, we are the first to investigate this task with this approach.

3. We develop a bespoke architecture construct which we name SpatialNet, and which is obtained by baking Spatial Transformer Networks [44] into a U-Net [85] backbone.

4. We explore predicting different times in the future using a latent variable. This has been attempted in different contexts [22, 99], but to the best of our knowledge we are the first to apply such a technique in the context of future image segmentation.

1.3 Outline

The remaining of this discussion is divided into six chapters, each covering a particular topic in regard to our investigation. These chapters are meant to be read sequentially, and information on previous sections is often required to follow current argumentation. Here, we present a brief outline of what each of these following chapters entails.

In Chapter 2 we present the background related to our investigation in two main parts. In the first section, we review the methods, the tools, and the broader context that relate our approach. In the second section,

we review the literature closest to our task, namely future prediction in computer vision. In doing so, we give a brief context overview of (1) future prediction through semantics, and (2) future prediction in RGB space; and then delve deeper into papers appertaining to future image segmentation.

In Chapter 3 we present our data bank, CorrelatedMotions. As such, we (1) justify the need to have a new data bank in the space of future image segmentation, (2) present the methods we use for capturing our data and generating our ground truths, and (3) present statistics and examples from our datasets.

In Chapter 4 we delve into our model in detail. Precisely, we present the architecture we use for our predictions, as well as our loss functions, evaluation metrics and implementation details.

In Chapter 5 we evaluate our model, testing its performance on our datasets. In order to do so we compare it against two baselines, that we obtain using a simpler model and a non-parametric simple output. We finally analyse our results and investigate the behaviour of the model on different data categories.

In Chapter 6 we perform a series of experiments in order to expand the scope of our model and further delve into and understand its behaviours and their roots.

Finally, in Chapter 7 we conclude our investigation through a discussion summarising our approach and findings, and presenting possible future work to follow our investigation.

Chapter 2

Background

In this Chapter, we explore the background related to our work. We divide this discussion into two main sections. The first section is concerned with the methods, tools and broader context that our approach uses or falls into. The second section presents the literature most related to our task, namely future prediction from image and video inputs.

2.1 Methods

To begin our discussion, we present in this section an overview of the methods, techniques and broader context most relevant to our work. We mainly focus on (1) deep learning techniques that are related to our models, and that can help justify our design choices and (2) research themes our project falls into. As such, we start by briefly going over convolutional networks, follow with a discussion on image segmentation, and end with a review of the Spatial Transformer Networks, a paper by Jaderberg *et al.* [44] that is central to our approach.

2.1.1 Convolutional Networks

In 2012, Convolutional Neural Networks (CNNs) grabbed the attention of the computer vision community with the performance of the AlexNet architecture on the ImageNet Classification challenge [51, 17, 49, 90]. They have been of ever-growing popularity since, and are now often at the heart of any recent competitive model relating to vision tasks [25, 24, 26]. As such, it is befitting to start with a brief mention of convolutional layers, and of popular architectures and techniques that are employed in conjunction with them. The *convolution* [1], '$*$', [31, 49, 20] is the defining operation of such models and can be written as

$$S(i,j) = (I * K) = \sum_m \sum_n I(i+m, j+n) K(m,n). \tag{2.1}$$

We follow the notation of [31], where I represents an input tensor (multidimensional array) to the operation, K another tensor typically referred to as a *kernel*, or a *filter* [65, 31] and S the output tensor of the operation.

[1] This is actually a widely accepted misnomer, the rigorous term being 'cross-correlation' [31].

In a typical 2D convolutional layer, this operation is repeated as a sliding window across the first two dimensions of the input tensor to produce the output tensor, referred to as the output *feature map* [31]. An additional bias term is also generally added to each output $S(i, j)$ [49]. Optionally, practical convolutional layers also support padding of the inputs, different stride values of the sliding window, and a dilation factor for the convolution [75, 68, 49, 20]. Convolutions defined this way naturally produce outputs with a different dimensionality from their inputs. Precisely, for a $H_{in} \times W_{in}$ input tensor, a padding value p, a $k \times k$ kernel, a dilation factor d and a stride value s, the dimensionality of the output is [75, 68]

$$H_{out} = \frac{H_{in} + 2p - d(k-1) - 1}{s} + 1 \tag{2.2}$$

$$W_{out} = \frac{W_{in} + 2p - d(k-1) - 1}{s} + 1. \tag{2.3}$$

In CNNs, convolutional layers are typically used in conjunction with pooling layers [31, 49, 78, 68], which are used to reduce the dimensionality of the output features and increase the *receptive field* of the network. In order to conserve dimensionality, as we will see in section 2.1.2, network architectures typically use a sequence of convolutional-pooling layers followed by a sequence of *deconvolutional* (or *transpose-convolutional*) layers [85, 55, 19]. The latter, as opposed to the former, typically increase the dimensionality of the input tensor [20, 76, 68, 88]. As the name suggests, this is achieved by transposing the kernel of the operation while in its matrix multiplication formulation, hence resulting in an operation where the outputs are larger than the inputs [88]. In a transpose convolution with a $H_{in} \times W_{in}$ input tensor, a padding value p, a $k \times k$ kernel, and a stride value s, the dimensionality of the output is [76, 68]

$$H_{out} = H_{in}s - 2p + k \tag{2.4}$$

$$W_{out} = W_{in}s - 2p + k. \tag{2.5}$$

Finally, the great successes of popular architectures such as VGG [89], Resnet [37], GoogleNet [96] or DenseNet [40] can be partially attributed to general advances and insights gained by the wider Deep Learning community. These include the use of dropout [92], batch normalisation (batch norm.) [43], as well as better performing initialisations [30, 52, 36], and non-linearities [64, 12]. It is interesting to draw further attention into the particular topic of skip connections, as it is a feature we heavily utilise and that has brought notable improvements to the performance of models specialised in semantic segmentation. Skip connections generally take the form of either pointwise addition or concatenation of previous layers to later layers in a a deep network architecture [34, 85]. It has been argued that doing so improves gradient flow (combating vanishing gradients), and helps the network ignore intermediate layers, hence improving both performance and reducing training time [34, 19, 85, 37, 66]. It is also possible to distinguish between short and long connections (to follow the nomenclature of [19]). The short category refers to connections akin to the ones introduced by Resnet [37], that allowed really deep architectures to be successfully trained and that are now standard in any significantly deep architecture [34]. The long category on the other hand broadly refers to connecting the features on the contracting part of a resolution-preserving architecture to their counterparts in the expanding part of the network. This can

take many forms, but has been shown to improve performance, as it allows the network to increase its receptive field significantly without sacrificing information due to down-pooling [85, 55, 19].

2.1.2 Image Segmentation

As one of the overarching themes of our project, it important to give some context information on image segmentation. We will start by giving an overview of the field, describing its main subtasks and methods. We will then take a much closer look into (1) U-Net [85], which is the base model for all our architectures, (2) The differentiable Intersection over Union (IoU) loss for image segmentation [80] which are also utilising, and (3) Mask-RCNN [35], a powerfull architecture that we are using for creating our data ground truths (c.f. section 3).

2.1.2.1 Overview

Segmentation is a very mature subject in computer vision, with its origins in the 1970's [105]. Extensively studied, there is an enormous amount of literature available in the subject, which we could not hope to give justice to [105, 33, 73, 27, 106, 69]. As such, we will be giving a brief, conceptual overview of some modern interesting avenues explored in the field, presenting some representative works and redirecting the reader to more comprehensive reviews. Our exposition is mainly based on three review papers [105, 33, 27], which get into much more detail on all the covered themes. We will also not be covering non deep-learning related techniques, with standard textbooks [73] and review papers [106, 69] being a more appropriate starting point for the reader to dive into the subject.

The main goal of image segmentation is to give a semantic label to each pixel of an image [105, 33, 27]. As mentioned by Garcia-Garcia *et al.* in [27], this can be seen as an extension to image classification that follows the coarse-to-fine inference trend: instead of classifying the image as a whole, its constituent parts are targeted. It is an important step in achieving a detailed semantic understanding of a scene, with obvious benefits to fields such as robotics or autonomous driving [105, 27, 56]. Within that broad category, more subtle sub-tasks can be identified, that vary in their level of difficulty and the amount of information they convey about a scene. Foreground/background extraction, or *object segmentation* [80], is one possibility, where the goal is to separate the 'foreground' (interest pixels from an object/category) with the rest of the image [105, 80]. Semantic segmentation is another where each pixel is classified as belonging into one of several categories [105, 27, 56]. An important point is that pixels belonging to the same category but to different objects would be indistinguishable by such methods [56, 27]. A natural step up then is that of instance segmentation, which combines object detection and semantic segmentation, and where each pixel is assigned to a particular object (or instance) and class label [56, 27, 35, 84]. Finally, one can go even further where the constituent parts of each instance start getting identified individually [27].

Over the years numerous methods and techniques have been explored for achieving the above-mentioned goals. Three particularly interesting ones in the context of deep learning are (1) Region-based segmentation, (2) Weakly supervised segmentation, and (3) Fully Convolutional Network (FCN) - based ap-

proaches [105, 33, 27, 55]. The first category is closely related to the detection framework, as it revolves around the idea of classifying different patches, or regions of interest (RoIs), within an image and then using that information to produce a segmentation [33, 10, 35, 29]. Interesting papers using deep learning techniques in this area include [35, 29]. Weakly supervised learning attempts to tackle the issue of the difficulty of obtaining carefully crafted dense labels typically required for training [105, 33]. Specifically, methods falling under this category attempt to retrieve the segmentation map by using coarse or image-level labels such as bounding boxes or image global classifications [105, 33]. Finally, variants of FCNs, an acronym initiated by the seminal work of Long *et al.* [55], are possibly the most explored methods in recent years. The basic premise is that these architectures make use of convolutional and deconvolutional layers in order to produce dense outputs that include spatial information [27]. This often translates to an encoder-decoder architecture where the encoder part successively reduces resolution via pooling layers and the decoder part upsamples the resulting features via deconvolutions [5, 55, 85, 27]. One of the challenges of semantic segmentation that becomes apparent with such models is the competing nature of local vs global information: If a large succession of pooling layers is used to get access to global information, fine grained local information is inevitably lost and vice-versa [27, 33]. Furthermore, local information is important in order to be able to produce sharp and precise boundaries, while global information is needed for context information [27, 33]. In order to overcome this hurdle, many methods have been proposed, including skip connections [5, 85], multi-scale architectures [58, 21], and post processing via Conditional Random Fields (CRFs) [11, 27, 33].

2.1.2.2 U-Net

U-Net, by Ronneberger *et al.* [85], is a great example of a FCN-based network and using skip connections to help semantic segmentation. It is the base for all of our models, and hence in this section we will review its architecture in some detail, which is best visualised with Fig. 2.1 below.

At its heart, U-Net is split in an encoder part that decreases the resolution of successive features and increases their receptive field, and a decoder that scales them back up to the original resolution. At each resolution scale, U-Net uses two convolutional layers with 3×3 kernels, followed by a ReLu [64] non-linearity. In the encoder part of the network max-pooling [31] layers are used in order to half the resolution of the input, and this is followed by doubling the amount of feature channels. In the decoder part of the network, deconvolutional layers are used in order to double the resolution and half the number of feature channels. In order to combat the loss of fine-grained local information, U-Net makes clever use of skip connections, concatenating the features of the encoder part of the network to the features in the decoder part with the same resolution. Finally, in order to correct for any miss-mach in dimentionality during the concatenation process, cropping of the encoder features is used [85].

2.1.2.3 Differentiable IoU loss

The Intersection over Union (IoU) is often a standard metric for judging the segmentation quality of an image [57, 56, 85, 55, 11, 5, 80]. As the name indicates, and guided by [46, 80], we can define the IoU

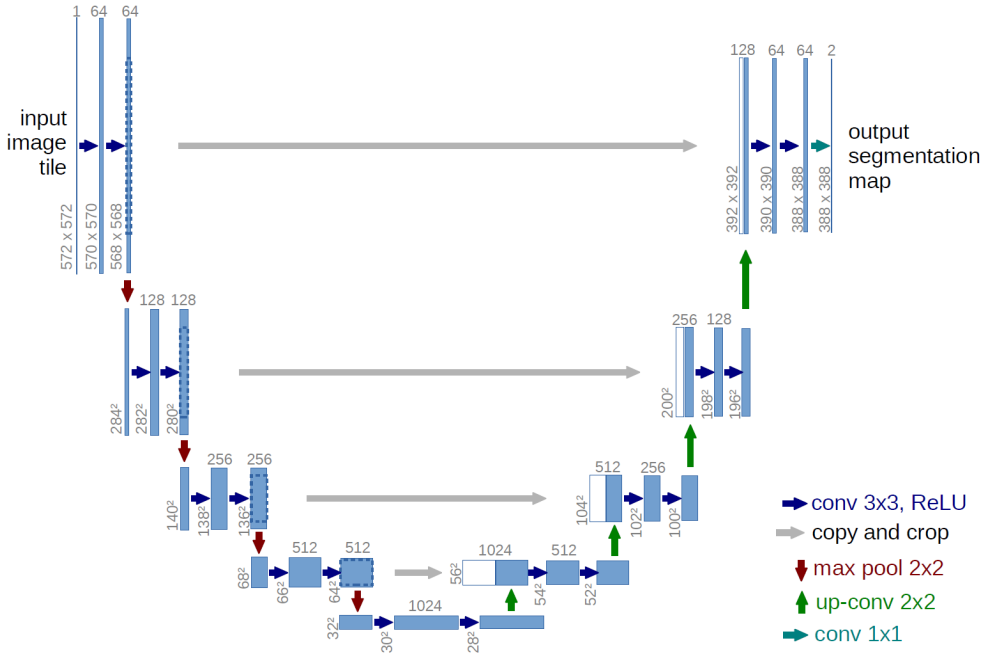


Figure 2.1: The U-Net architecture. Figure taken from [85].

as follows. Let M_1 and M_2 be two binary masks indicating the locations of the pixels of a segmented object or class in an image, then [46, 80]

$$IoU = \frac{\sum[M_1 \odot M_2]}{\sum[M_1 + M_2 - M_1 \odot M_2]}. \tag{2.6}$$

Where $\odot$ is the hadamard product [61] between M_1 and M_2, the addition is point-wise and the sum is over all the pixels in the mask.

In their work [80], Rahman *et al.* argue that in the task of object segmentation as defined in section 2.1.2.1, which our project falls into, optimising the IoU metric directly is better than using some generic, global loss. The conceptual argument they present stems from the fact that the number of pixels belonging to the foreground is usually substantially smaller than their background counterparts. As such, they argue that this class imbalance could hurt performance if combined with a loss function that would optimise global accuracy. The example they present is a dataset with 90% of the pixels belonging to the background, where a system optimising for accuracy could get a performance 90% by only predicting background pixels. They also verify their hypothesis experimentally, where they show that using a direct optimisation of IoU generally beats the same network optimised through a softmax loss on various datasets.

Finally, [80] defines a differentiable loss function for optimising the IoU directly. The challenge is that for computing the IoU, the network outputs (one for each pixel) need to be collapsed to a 0 or 1 class, indicating whether the pixel belongs to the segment or not. Rahman *et al.* propose an approximate IoU alternative. Following their notational conventions, let V be the set of all pixels in all images, X the corresponding set of sigmoid network outputs, with X_v the output of the network for pixel v, and Y the

set of ground truth assignments for each pixel with $Y_v \in \{0, 1\}$. Then they define the IoU approximation as [80]

$$IoU = \frac{I(X)}{U(X)}, \tag{2.7}$$

$$I(X) = \sum_{v \in V} X_v * Y_v, \tag{2.8}$$

$$U(X) = \sum_{v \in V} X_v + Y_v - X_v * Y_v, \tag{2.9}$$

and the IoU loss as [80],

$$L_{IoU} = 1 - IoU. \tag{2.10}$$

This operation is then entirely differentiable and can be back-propagated, with the gradient being [80]

$$\frac{\partial L_{IoU}}{\partial X_v} = \begin{cases} \frac{-1}{U(X)}, & \text{if } Y_v = 1 \\ \frac{I(X)}{U(X)^2}, & \text{otherwise.} \end{cases} \tag{2.11}$$

2.1.2.4 Mask-RCNN

Mask-RCNN [35] is the current holder of state-of-the-art for the instance segmentation problem, and is the architecture we use in order to generate our dataset. In order to truly gain a deep understanding of the intricate mechanics comprising this framework, we invite the reader to further explore the succession of papers that resulted in Mask-RCNN, namely those introducing R-CNN [29], Fast-RCNN [28], Faster-RCNN [82], and finally Mask-RCNN [35]. In this section, we will present a broad overview of the Mask-RCNN architecture, following the summaries presented in [67, 4, 56].

At its core, Mask-RCNN can be seen as composed of two separate structures: (1) The convolutional backbone and (2) The network (or detection) head [56, 4, 35]. The purpose of the backbone is to extract convolutional features from the input image. The exact nature of the backbone can vary, with standard convolutional architectures [4, 35] (such as Resnet [37]) or Feature Pyramid Networks (FPNs) [54, 35] being common choices [4, 35, 56]. These features are then fed into the detection head, which can itself be decomposed into two stages [56]. The first stage is the Region Proposal Network (RPN), which finds Regions of Interest (RoIs), i.e. regions in the image that are likely to contain an object. The second and final stage uses these RoIs in order to (1) predict a class labels, (2) refine their bounding box coordinates and (3) output a segmentation for the enclosed object [4, 56]. The main innovations of Mask-RCNN compared to its predecessor, Faster-RCNN, are (1) adding a method to avoid slight miss-alignments when extracting the RoI features for the final stage and (2) adding a branch to the network for producing the instance segmentations [67, 35].

2.1.3 Spatial Transfomer Networks

At last, in this section we will take a close look at the spatial transformer network, which lies at the heart of our solution. Proposed by Jaderberg *et al.* [44], the spatial transformer network can learn an explicit spatial transformation applied to a set of features in some CNN architecture. This transformation can be more or less restrictive, and hence depending on the number of free learnable parameters it can represent a simple Euclidean transformation or a full Homography mapping [73]. The module is furthermore differentiable, and hence can be incorporated in a deep architecture with the ability to backpropagate through it [44].

As illustrated in Fig. 2.2, the spatial transformer module is applied independently to each depth channel of a feature set U to produce and new transformed set V. It can be decomposed into three main parts: The localisation network (L.N.) , the grid generator and the sampler [44]. For notational convenience that will become apparent in the relevant sections, in our models we refer to the combination of the grid generator and the sampler as the transformer unit (T.U.).

The purpose of the localisation network is to learn the set θ of parameters of the spatial transformation to be applied. It takes the form of any network with convolutional and dense layers, with the constraint that the output layer needs to have a number of units equal to the number of free parameters in the target transformation. For instance, in 2D, an Affine transformation would have 6 parameters, while a simple translation would only have 2 [73]. It is important to note then that this network makes the transformation applied conditional upon the input feature map [44].

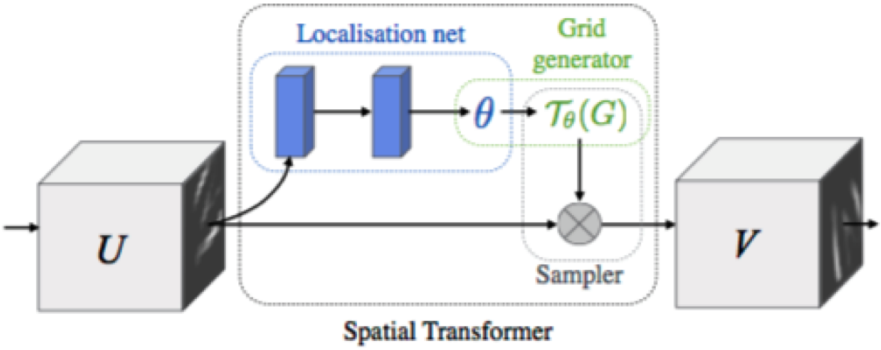


Figure 2.2: Illustration of the Spatial Transformer Network. Figure taken from [44].

In order to make networks with spatial transformer modules trainable end to end through backpropagation, the transformation is applied in a differentiable way. In fact, this is the purpose of the grid generator and the sampler, and an illustration of the mechanism can be seen in Fig. 2.3. Qualitatively, the trick is to sample from the input feature map the grid of points that map to the spatial locations of the output feature map [44]. Quantitatively, and following the notational conventions form [44], we have that the following holds [44],

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = T_\theta(x_i, y_i) = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{11} & \theta_{11} \\ \theta_{11} & \theta_{11} & \theta_{11} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}. \tag{2.12}$$

Where (x_i^s, y_i^s) and (x_i^t, y_i^t) represent the spatial coordinates of the input and output feature maps respectively, T_θ represents the parametric transformation applied, and θ_k represent the parameters learned in the localisation network [44]. As the name suggests, the purpose of the grid generator is then to generate a set $\{x_i^s, y_i^s\}_i$ of coordinates in the input features corresponding the the set of coordinates $\{x_i^t, y_i^t\}$ in the output features. Using these coordinates, the sampler finally can use any differentiable sampling technique to map the feature values at (x_i^s, y_i^s) to (x_i^t, y_i^t) [44].

In [44], Jaderberg *et al.* mainly advocate spatial transformer modules as helping with (1) translational, scale and roational invariance in classification, (2) localisation of certain types of features in images and (3) networks requiring a spatial attention mechanism. In our work, we use the spatial transformer modules in the application of predicting future mask segmentations. In fact, as described in section 4.1, we believe that these modules will help us encode motion patterns from our data, and empower a more accurate prediction for the future segmentations.

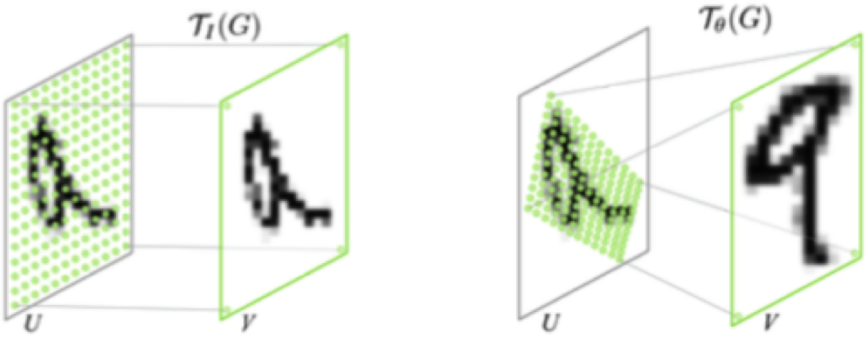


Figure 2.3: Illustration of the grid sampling mechanism in the spatial transformer module. The left image illustrates an identity mapping while the right one shows some transformation with parameters θ. Figure taken from [44].

2.2 Related Work

Predicting the future in a computer vision setting is a challenging task, and many different approaches have been attempted over the years. We have identified three main avenues that we believe are worth mentioning, namely (1) predicting the future using semantics or an abstracted space such as optical flow, (2) using direct RGB prediction in image space, and (3) using image segmentation techniques for predicting future segmentations. Our work falls into the third category, which was in part developed in parallel to our work. In this section we will have a closer look into some interesting examples in each

of these categories, with greater emphasis on the third, and mostly considering works that are related to recent deep learning techniques.

2.2.1 Future Prediction through Semantics

In this first line of investigation two threads appear to be more commonly explored, namely (1) predicting optical flow (a dense vector field at each pixel indicating its motion) and (2) predicting future actions.

2.2.1.1 Optical Flow

To the best of our knowledge, the seminal work on predicting future optical flow was conducted by Pintea *et al.* [71]. More precisely, they use a single static image along with a Random Forest (RF) in order to predict optical flow at each pixel, their conjecture being that appearance provides sufficient clues for predicting corresponding motion. An additional challenge they had to overcome is that the motion space is continuous, and hence they had to adapt the RF method to deal with a regression setting. Finally, they illustrate a set of possible applications to this motion prediction task, namely (1) outlier detection (2) action recognition, and (3) motion saliency.

Following [71], Walker *et al.* [100] tackle the problem using CNNs, and test their method in a harder dataset. In order to facilitate the task, they also cast it as a classification problem, where each pixel can be assigned to one of 40 possible optical flow vectors. As a follow-up, Walker *et al.* [99] expand the scope of their work in order to account for the uncertain nature of possible motions given a static image. In order to do so they use the Variational Autoencoder (VAE) framework, and sampling from a latent variable in order to produce different possible predictions given an input.

Similarly to many other works [57, 56, 45, 8, 63, 86, 38], and in contrast with the above-mentioned methods, we use several frames of a video as input to our models. We believe this is justified as (1) having more frames would provide richer motion cues than a single image, (2) we conjecture that having such cues would allow us to predict more complex motions, and (3) in contexts such as robotics, autonomous driving and video synthesis, we assume that video data would be naturally available.

2.2.1.2 Action Prediction

In the action prediction line of research, the main goal is to predict what action is going to be taken next by the subject of a video sequence [86, 38, 53, 39, 98]. In one of the early works on the subject [86], Ryoo formulates the problem of recognising activities before they are fully observed, and proposes a modified bag-of-words method as the solution. Following up, Hoai and Torre propose a Support Vector Machine (SVM) based method to tackle the challenge [38]. More recently, Lan *et al.* [53] relax the constraint of multiple inputs representing the early part of an action, and test their solution on more realistic data. A different take was also taken by Huang *et al.* [39], which consider action recognition in a human interaction setting, where by observing the actions of one person the aim is to predict and simulate what its counterpart will do. Finally, a deep learning approach was adopted by Vondrick *et al.* [98]. In this work, a neural network architecture is utilised in order to predict the future frame representation from the

current frame. This representation can then be used for action recognition in the future, and the method has the advantage of being able to leverage large amounts of unlabelled data.

2.2.2 Future RGB prediction

The task of directly predicting RGB values in pixel space is another major avenue that has been explored. The basic premise itself is conceptually simple: given an input of a few video frames, generate a possible future [101, 93, 81, 58, 48]. The corresponding task nonetheless is very complex, as it requires modelling the correct structure, dynamics and rendering [101] for a given scene, and then regressing a sample from the resulting distribution of possible future frames in the highly unconstrained space of RGB values [101, 58, 48].

In one of the early papers on this problem, Ranzato *et al.* [81] borrowed techniques from the language modelling literature in order to create baselines for next frame generation and frame filling. Further work by Srivastava *et al.* [93] explores LSTM based models that aim to capture good video representations in an unsupervised way. In [58], Mathieu *et al.* attempt to tackle the blurriness problem observed in previous works by using a multi-scale network and moving away from the Mean Squared Error (MSE) loss in favour of adversarial training and an image gradient difference loss. More recently, Kalchbrenner *et al.* [48] chose a more probabilistic approach, considering the factorisation of the joint distribution of the pixels through video pixel networks. Finally, Walker et al. [101] argue that they can facilitate the task by disentangling the modelling of structure, dynamics and rendering, and hence using a smaller dimensional space for their predictions. In order to do so they choose to make future predictions in pose space, before using those in order to render a frame prediction.

Forecasting in RGB space from video data is a difficult task, and there is still a lot of room for improvement in the area [93, 101, 58, 48]. Although it would contain very rich information about the future events that are being predicted, it is reasonable to argue that such level of detail is not always necessary. In the context of autonomous driving for instance, it is intuitive that the useful prediction is where the different actors in the scene will be in the future, not necessarily what their exact appearance is [57]. This is also consistent with the appearance of the more constrained task of future segmentation in this space [57, 56, 63, 45, 8], which we describe in section 2.2.3 below.

2.2.3 Future segmentation prediction

Casting the task of future video prediction as a segmentation problem is a very recent approach, and was partially developed in parallel to our work. To the best of our knowledge it was pioneered by [57], where Luc *et al.* used a multi-scale network [58] to predict a dense semantic segmentation map for a future frame, using past frames as inputs. Their base models predict one frame into the future, and they experiment with two ways for extending this into longer term predictions, as illustrated in Fig. 2.4. The first one (top) consists of using an autoregressive framework, where output predictions are recursively applied as inputs, with backpropagation through time [103] used in order to fine tune the models and correct for potential error accumulations. The second one (bottom) uses a multiple output design

in order to predict different future times simultaneously from a single input. They also adventure in predicting different combinations of RGB values and segmentation masks as inputs and outputs for their models. Their experiments indicate that (1) it can be quite effective to directly predict the image segmentations [57, 56], and (2) the autoregressive method was more effective when predicting segmentations but the simultaneous predictions model gave better results in RGB space [57].

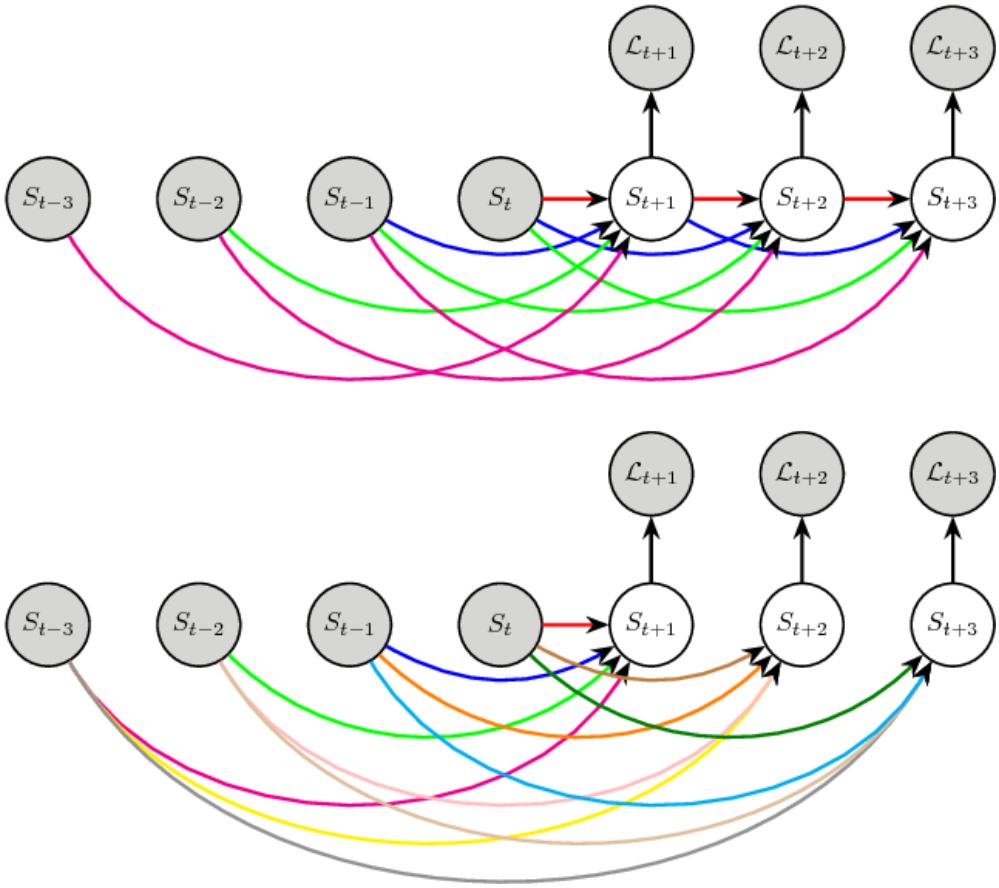


Figure 2.4: Illustration of the autoregressive (top) and simultaneous - or "batch" [57]- (bottom) ways for making longer term predictions in [57]. In the autoregressive case, prediction outputs are used as inputs for the next prediction [57]. Figure taken from [57].

In order to improve from [57], Jin et al. [45] combined learning optical flow and image segmentations into one model, simultaneously predicting both and using each to bolster the performance of the other. More precisely, as illustrated in Fig. 2.5, Jin *et al.* used two modules, one for optical flow prediction and one for semantic image segmentation. In turn, (1) the optical flow module outputs are fed into the semantic segmentation module in order to help the future segmentation, and (2) the obtained segmentations are used to divide the image into moving, static, and other objects in order to aid with the flow estimation. Similarly to [57], this work predicts a dense semantic segmentation map one frame into the future, and is applied recursively with backpropagation through time in order to make longer term predictions.

Finally, in the space of future semantic segmentation, two very recent releases are [8] and [63]. In [63], Nabavi *et al.* use convolutional LSTMs and bi-directional LSTMs for task, and argue those are effective

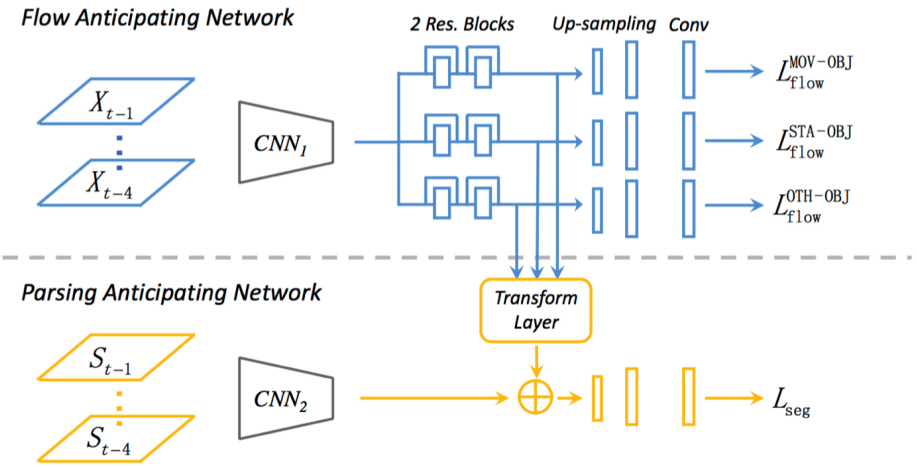


Figure 2.5: Illustration of the model used in [45] for simultaneous optical flow and semantic segmentation prediction. Figure taken from [45].

models. In [8], Bhattacharyya *et al.* tackle the full Bayesian formulation, proposing an importance sampling method in the variational framework. For longer term predictions, [8] uses a recursive formulation from one step ahead outputs while the method used in [63] is not clear.

Most related to our work, Luc *et al.* [56] recently proposed a method that integrates future segmentation in the Mask-RCNN pipeline. Elaborating on their work in [57], they address the issue of instance segmentation as opposed to semantic segmentation. A challenge that had to be overcome is that the number of instances from each class changes frame to frame. To tackle this, as illustrated in Fig. 2.6, Luc *et al.* make the predictions in the feature space within the Mask-RCNN pipeline. More precisely, they insert an independent (from each other) Feature to Feature network at each resolution layer l (F_2F_l) of the FPN backbone of the Mask-RCNN architecture. These F_2F_l networks maintain resolution, take as an input the FPN features at layer l from a few past frames, and aim to predict the features at layer l of the next frame. Similarly to their previous work, they apply the model recursively in order to make longer term predictions; using an autoregressive framework and backpropagation through time. It may be important to note that due to memory constraints, it was only possible to train the lowest resolution layer offline. As such the other layers had to be initialised to those trained values and fine-tuned during inference. A possible drawback of this system is that it is inherently tied to the Mask-RCNN architecture. Although currently state-of-the-art, if a stronger segmentation architecture gets released it is not a necessity that the methods presented in this paper will be applicable to it. Our approach does not suffer from this, and would only naturally improve performance-wise with better segmentation and tracking architectures generating better training ground truths. Finally, it is important to note that our work might prove complementary to the framework proposed in [56]. In fact, as mentioned in section 7.2, an interesting future work to explore would be the effect of integrating of our SpatialNet networks as the F_2F_l models in the above formulation.

Our work differs from the above-mentioned investigations in three principal ways: First, we propose a

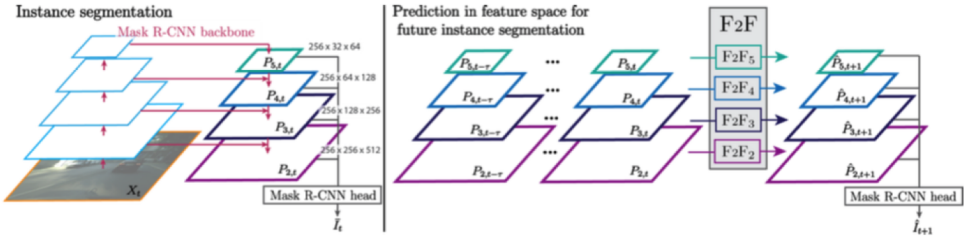


Figure 2.6: Illustration of the Mask_RCNN architecture (left) and the F_2F_l construct used in [56] for the future segmentation task (right). Figure taken from [56].

new data bank, CorreletedMotions, which we argue is more appropriate for this task of future segmentation prediction. Second, we treat the task under the light of object segmentation, where inference is made for a specific object in the scene, specifically designated in the inputs. This is similar to the instance segmentation proposed in [56], but conceptually cleaner: In [56], an RGB image is fed into their model, and future predictions for all the present objects can be made at different time-steps. However, there is no way to systematically link instances between those predictions. In other words, cars A, B and C get predicted future masks in times t+1, t+2 and t+3, but there is no way to systematically detect which mask is which between t+1,t+2 and t+3, or to link the masks back to specific RGB objects at time t. An additional system of tracking-by-detection [83, 7, 104] would need to be applied. As opposed to this our method works seamlessly: car A is designated at t, and we obtain its segmentation at a future time. Third, we investigate using a latent variable for accounting for different prediction time-steps. Although attempted in different contexts [22, 99], to the best of our knowledge we are the first to apply such a technique to future image segmentation. As mentioned above, all other methods use a recursive formulation, with the exception of [57] that experiments with multiple simultaneous outputs and [63] which is unclear in its approach.

Chapter 3

Dataset

In this Chapter we present our data bank, CorrelatedMotions, that is comprised of several datasets showcasing two main scenes with correlated motions. We argue that this data bank is superior than the currently used Cityscapes [16, 14] for the task of future image segmentation, describe how it was obtained, and showcase statistics and example datapoints.

3.1 Motivation and Overview

In section 2.2, we introduced the works relating to the task of future prediction through video data. In the recent branch of future image segmentation (section 2.2.3), Cityscapes [16, 14] seems to have prevailed as the consensus benchmark dataset. It is currently comprised of 5000 images with fine (precise and dense) annotations and 20 000 images with coarse (rough and sparse) labelling. These images are in fact the 20^{th} frame out of 30 frame sequences making up $1.8s$ video snippets. As is illustrated in Fig. 3.1, these video snippets are captured in a first person driving perspective. That is, the data is captured by driving around in different cities at different times of the year, with a camera mounted on the car. We argue that this dataset is not appropriate for the task of predicting the future motions of different agents from a video snippet. The main issue with Cityscapes is the capturing perspective. Driving a car around cities creates a tunnel view of the world that skews and overwhelms the motions of agents in a scene: Looking at video examples of the dataset [15, 72], it is apparent that one could do pretty well by learning the pattern of motion of the camera and the effect it has on the scene, especially on shorter term predictions. As such, we propose here a new dataset, which we name CorrelatedMotions, and which does not suffer from this predicament.

In fact, our dataset stems from the line of thought that a truly intelligent system should be able to identify the interplay between different agents in a scene with respect to themselves, not just with respect to the camera. Consequently, it should be able to pick up (1) the physics of the world (cars do not sink into the ground, inertia prohibits brusque motion changes), and (2) the correlations between the motions of different agents in the scene (a car should not run over pedestrians, it can only move on the road, it should stop at a stop sign). With the intention to explicitly capture such complex scenes and correlated

(a) Example of a fine-annotated image.

(b) Example of a coarse-annotated image.

Figure 3.1: Examples from the Cityscapes dataset [16, 14, 15]. Taken from [15], the webpage also contains more, as well as video examples.

motions, we have used two different settings for the data: (1) Predicting the motions of cars across a pedestrian crossing, and (2) Predicting the motion of two people as they pass a football to each other. In the first setting, a capable inference system would need to identify and encode the rules of the road and the crossing. In other words, cars need to stop and wait when pedestrians are ready to cross, or continue at consistent speed when this is not the case. In the second case, a model needs to learn the motion correlations inherent to the game being played. In our particular setting, the most important one lies between the motion of the ball and the motion of the players, which have to move in order to intercept it.

Our data bank contains raw data, referring to the sampled frames from the video footage, and processed data, referring to the preprocessed, refined datapoints directly usable by our models. Unless otherwise specified, a "datapoint" in our context will hereafter represent a structure with the following:

- A set of 3 *input* RGB images, $\{X_{t-4}, X_{t-2}, X_t\}$ at frames $\{t-4, t-2, t\}$, or equivalently times $\{t-0.4s, t-0.2s, t\}$, which we will henceforth universally denote as $\{t-4, t-2, t\}$.
- A set of 3 *input* binary masks segmenting a single agent, $\{M_{t-4}, M_{t-2}, M_t\}$.
- A set of 3 centroid coordinates for the binary masks, $\{\mathbf{c_{t-4}}, \mathbf{c_{t-2}}, \mathbf{c_t}\}$.
- A future mask used as our *label*, M_{t+5} or M_{t+10}, depending on the context of the prediction: medium term -$0.5s$-, or long term -$1.0s$-.
- A centroid coordinate for the future mask, $\mathbf{c_{t+5}}$ or $\mathbf{c_{t+10}}$.

Furthermore, the data fall into the Football set, the Crossing set and the Large Crossing set, with the processed data being further split between the following categories:

- Football (FBL), with 9036 training examples and 1002 test examples for predictions of 0.5s ahead.
- Long Term Football (FBLLT), with 8790 training examples and 976 test examples for predictions of 1.0s ahead.
- Crossing (CRS), with 13212 training examples and 1466 test examples for predictions of 0.5s ahead.

- Long Term Crossing (CRSLT), with 10343 training examples and 1149 test examples for predictions of 1.0s ahead.
- Large Crossing (LCRS), with 27760 training examples and 3082 test examples for predictions of 0.5s ahead.
- Long Term Large Crossing (LCRSLT), with 22714 training examples and 2521 test examples for predictions of 1.0s ahead.

In the following sections we give detailed explanations of these categories, what they represent and how they were obtained. We furthermore explore statistics for these and any insights that can be gained from those, as well as showcase some representative examples of datapoints.

3.2 Data Generation Methods

3.2.1 Capturing the data

For all the data, we use a Sony RX0 [91] to capture the original footage. Example frames are showcased in section 3.3.2. The football scene is captured from a single camera pose at $100fps$ and 1280×720 resolution and amounts to $9m\ 50s$ of footage. The crossing scene consists of two viewpoints, one for each traffic lane of a double-sided road, filmed at $50fps$ and 1920×1080 resolution. As described below, the first viewpoint amounts to $13m\ 57s$ of footage and is used for the Crossing set while both viewpoints add up to $26m\ 24s$, and are used together for the Large Crossing set.

After obtaining the footage, we sample frames as JPEG images to create the raw data. As illustrated in table 3.1, we sample at 10fps and at a resolution of 960×540. For the Crossing set one viewpoint is used while both viewpoints are combined for the Large Crossing one. As a result of these manipulations, 5414, 8367 and 15836 images are obtained for the Football, Crossing and Large Crossing data respectively.

Table 3.1: Metadata regarding the raw data from the CorrelatedMotions data bank.

Data	Number of frames	Resolution	Frame rate	Number of viewpoints
Football	5414	960×540	10	1
Crossing	8367	960×540	10	1
Large Crossing	15836	960×540	10	2

3.2.2 Getting the Ground Truths

Supervisory signals for our models consist of tracked segmentations for the classes of concern. This can be visualised with the examples in section 3.3.2. In other words, for each object in the clips we need a mask segmentation tracked across different frames. In order to obtain this, we implement a succession

of (1) detecting and segmenting all object instances in the clips, (2) discarding irrelevant segmentations, and (3) tracking the remaining detections across frames.

For the first step of this pipeline, we use the Mask-RCNN [35] implementation from [59], which we run for all our frames. In the second step we use heuristics in order to get rid of less relevant masks. Regarding the Football data, only person detections with confidence above 99% are kept in an effort to isolate the two players in the scene (c.f. Fig 3.3). Additional positional heuristics are also used in order to further isolate the players from other detections. For the Crossing data, only car masks with confidences above 80% are kept. We further discard the bottom 50% of masks in terms of size in order to minimise the number of parked car detections, which would otherwise overwhelm our dataset with trivial examples. Finally, inspired by the SORT tracker [7] and its implementation from [6], we implement a tracking-by-detection mechanism across frames. Specifically, framing the setup as an optimal assignment problem [9] and using the IoU of masks between sequential frames, we solve the tracking using the Hungarian method [7, 9, 6]. At the end of this process each mask has an ID assignment which allows us to track the object across frames, and everything is stored in an hdf5 file format [3] using the h5py python package [13]. Failures in this pipeline introduce noise in our data, some of which is inevitable. Nonetheless, as described in 3.2.3, sampling for the final datapoints also alleviates the issue somewhat further, as only objects with a consistent tracking history across the required time-frame are designated for use in our models.

3.2.3 Preprocessing

The raw data and ground truths in the Football, Crossing and Large Crossing sets, as obtained by the above methods, provide a lot of flexibility and can be used in a variety of settings. In order to use them in our models however, we further clean up and categorise the data through a series of preprocessing steps.

First, all images and masks are resised to a resolution of 455×256, in order to reduce memory overhead, speed up subsequent processes and reduce the dimensionality to values closer to what is used in our models. Second, centroid positions are obtained and stored for all of the masks, by computing the mean x and y mask position. Third, sets of consistent datapoints, as defined in section 3.1, are sampled. In order to do so, the data are scanned for single-ID masks across the required time-steps. In particular, times $\{t-4, t-2, t, t+5\}$ are used to create the Football, Crossing, and Large Crossing sets; and times $\{t-4, t-2, t, t+10\}$ are used to create the Long Term Football, Long Term Crossing, and Long Term Large Crossing sets (c.f. section 3.1). Finally, train/test splits are created for these datasets, using roughly 10% of the data for testing. For the test set sampling, the 90^{th} percentile frame is found and used as a split frame, with all subsequent ones belonging to the test set.

3.3 Data Illustrations and Statistics

3.3.1 Dataset statistics

In order to gain insight into the distributions of our data, we recorded different metrics across the sets, with the results being depicted in table 3.2 and Fig. 3.2. Specifically, we calculated the mask IoU (mIoU), the bounding box IoU (bbIoU), and the centroid displacement in pixels (c.f. section 4.3 for a detailed definition), between the input mask at time t, and the label mask at time $t+5$ or $t+10$. Table 3.2 depicts the mean values for these metrics, as well as their standard deviation across the different sets. Fig. 3.2 shows the detailed histogram distributions of the number of examples versus the value of each metric, and for each dataset.

Table 3.2: Number of examples in each of the datasets and across the train/test split. This table also depicts the mean mask IoU (mmIoU), mean bounding box IoU (mbbIoU) and the mean centroid displacement (mDis) , give or take one standard deviation, between the input and the ground truth label for these sets. The sets in question are described in section 3.1, and consist for the Football (FBL), the Long Term Football (FBLLT), the Crossing (CRS), the Long Term Crossing (CRSLT), the Large Crossing (LCRS), and the Long Term Large Crossing (LCRSLT) datasets.

	Training set				Test set			
data categories	# examples	mmIoU	mbbIoU	mDis (px)	# examples	mmIoU	mbbIoU	mDis (px)
FBL	9036	0.48 ± 0.3	0.57 ± 0.3	6.8 ± 10.2	1002	0.48 ± 0.3	0.59 ± 0.3	6.5 ± 11.7
FBLLT	8790	0.36 ± 0.3	0.45 ± 0.3	12.6 ± 19.7	976	0.35 ± 0.3	0.45 ± 0.3	12.2 ± 22.5
CRS	13212	0.55 ± 0.3	0.59 ± 0.3	20.7 ± 27.0	1466	0.55 ± 0.3	0.58 ± 0.3	19.4 ± 25.3
CRSLT	10343	43.9 ± 0.4	0.47 ± 0.4	40.3 ± 51.3	1149	0.42 ± 0.4	0.45 ± 0.4	39.1 ± 48.3
LCRS	27760	0.57 ± 0.3	0.61 ± 0.3	17.9 ± 24.2	3082	0.56 ± 0.3	0.6 ± 0.3	17.9 ± 23.9
LCRSLT	22714	0.46 ± 0.4	0.50 ± 0.4	34.0 ± 45.9	2521	0.44 ± 0.4	0.47 ± 0.4	35.0 ± 45.7

In reading the statistics in table 3.2 and Fig. 3.2, it is important to keep in mind that large centroid displacements correspond to small values of IoU, and vice-versa. Moreover, all the displacement units are in pixels, IoU values are unitless, and an IoU value of 0 could represent any displacement value large enough for the two masks/bounding boxes not to overlap.

Some notable observations can be made directly from table 3.2. First, the long term prediction datasets have less datapoints than their medium term counterparts. This makes sense considering the generation process, as it is less likely that a consistent ID will persist across a 14 frame span than a 9 frame span. Second, there is a quite high variance in the data in terms of movement, which all three metrics represent in slightly different form. Third, mask IoU values are consistently smaller than bounding box IoUs, which makes sense as mask IoUs are more easily affected by shape deformations.

Nonetheless, the histograms in Fig. 3.2 give a more rounded picture of the distribution of our data, and

allow us to draw some more fine grained conclusions. For instance, we can see from the figures that our test set distributions reflect their training counterparts quite well. It is also appears that the IoU distributions are generally multimodal, with a peak towards high overlap of the input and future masks and a peak towards no overlap. Unsurprisingly, long term predictions skew this distribution towards the no-overlap end, as objects have double the time to move away from their initial positions. Finally, we see that small centroid displacements can account for a wide range of IoU values, as most of the data is skewed towards to initial peak in the centroid displacement space.

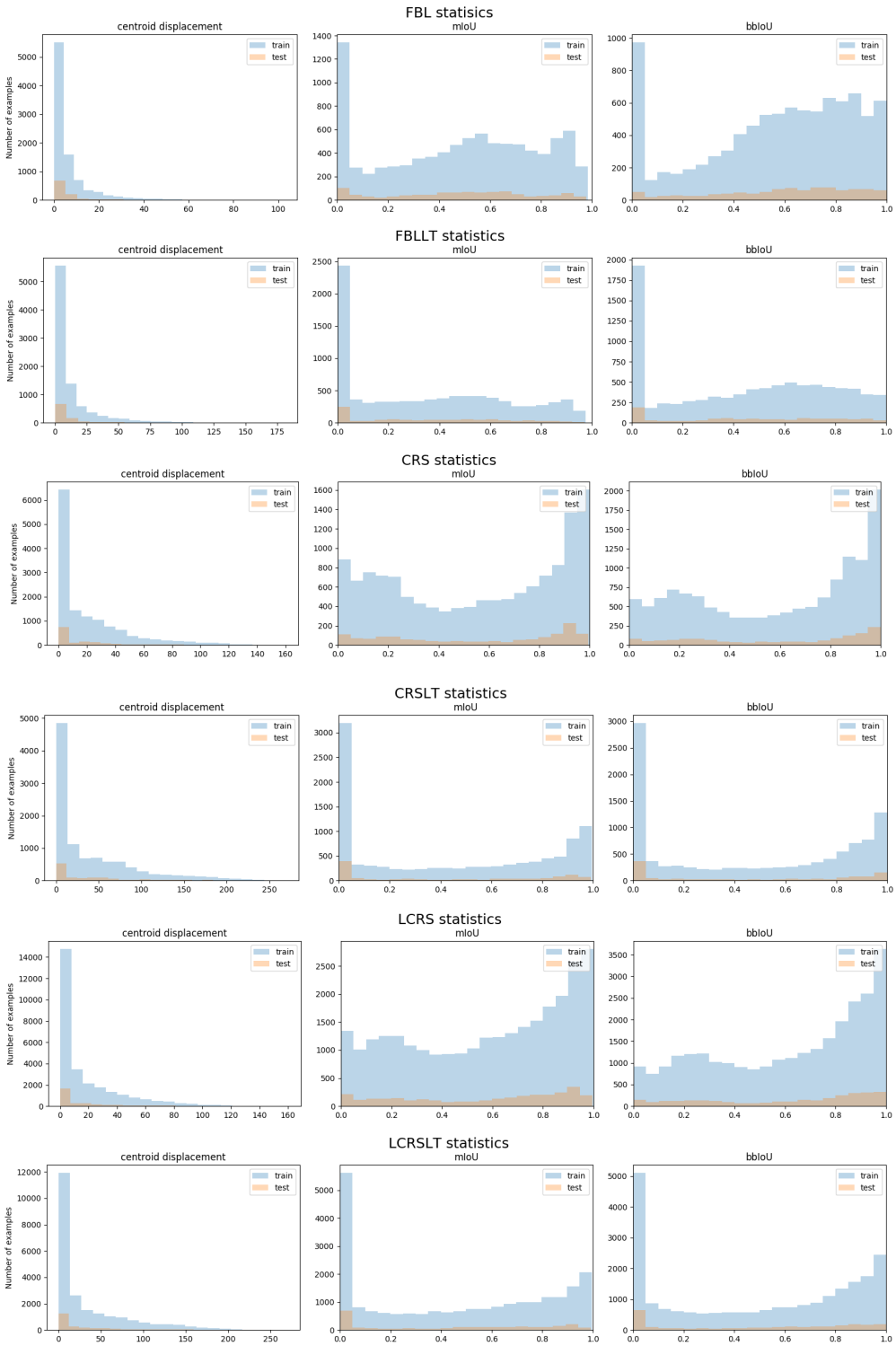


Figure 3.2: Dataset statistics: Histograms of the number of examples for different values of mask IoU (mIoU), bounding box IoU (bbIoU) and centroid displacement (in pixels) between the inputs and ground truths and across our datasets, namely Football (FBL), Long Term Football (FBLLT), Crossing (CRS), Long Term Crossing (CRSLT), Large Crossing (LCRS), and Long Term Large Crossing (LCRSLT).

3.3.2 Example Datapoints

In this section we display some typical examples of the Football, Crossing and Large Crossing sets. Their long-term prediction counterparts are omitted, as no significant difference would be noticeable. It is furthermore important to note that in all of Fig. 3.3, 3.5, and 3.4, only the input masks and images at time t are shown, while our models typically also receive their $t-4$ and $t-2$ counterparts. In Fig. 3.3, two examples are shown, one for each player. In Fig. 3.4 also two examples are shown, one for each camera viewpoint. Finally, in Fig. 3.5 three examples are shown, representing possible car motions in the dataset.

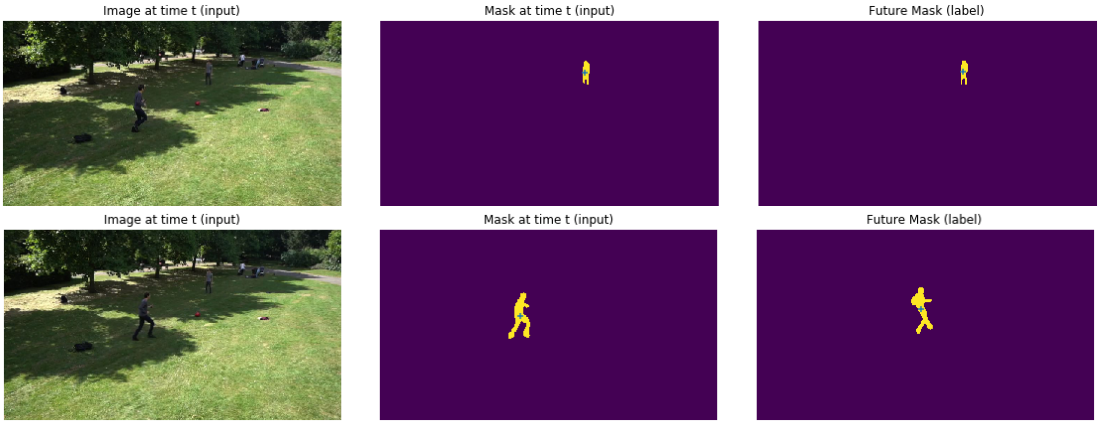


Figure 3.3: Typical datapoint images and masks from the Football set. Only the inputs at time t are shown, and the label is at $t+5$. Our models typically take additional inputs at $\{t-2, t-4\}$.

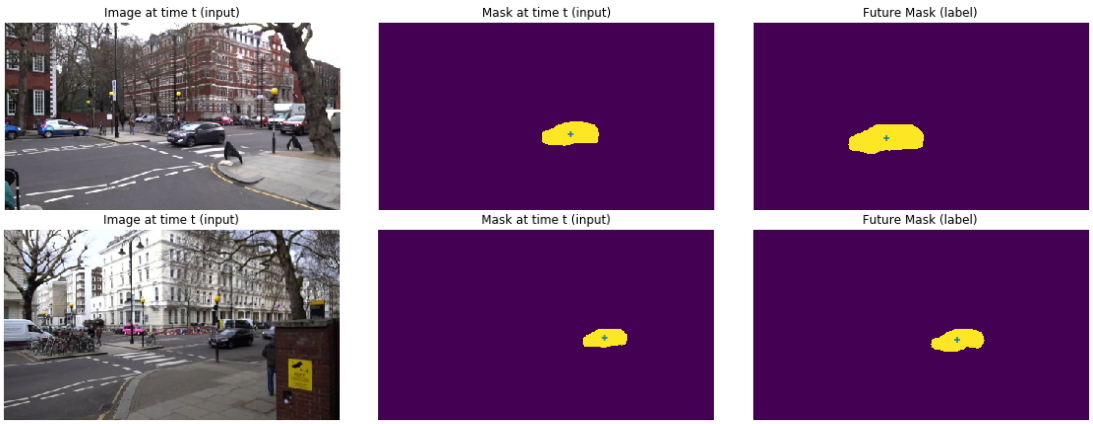


Figure 3.4: Typical datapoint images and masks from the Large Crossing set. Only the inputs at time t are shown, and masks are at $t+5$. Our models typically take additional inputs at $\{t-2, t-4\}$.

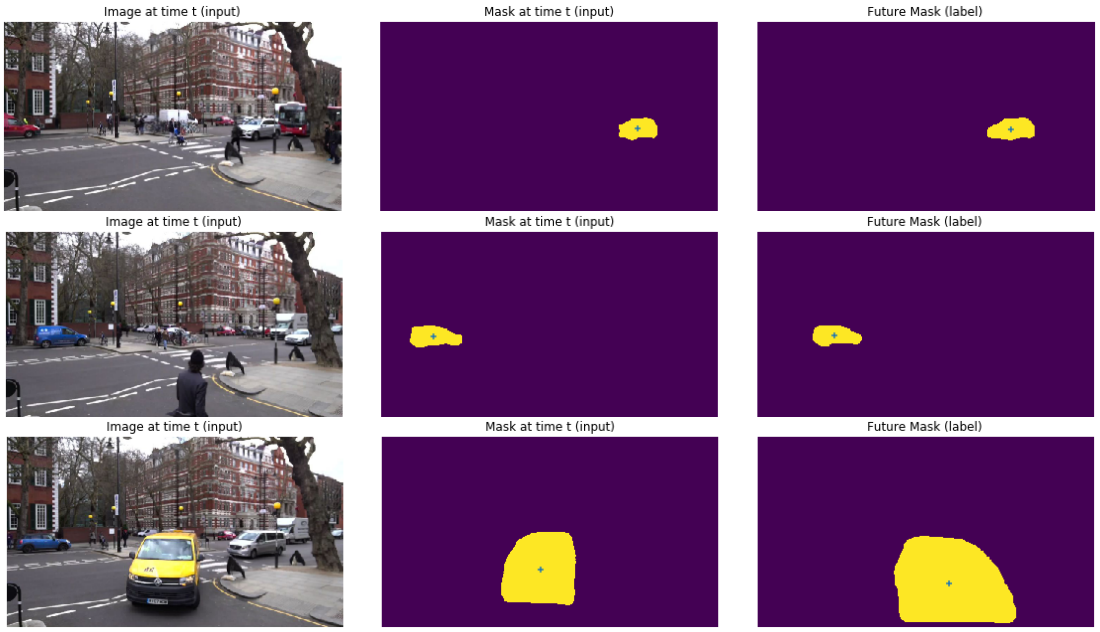


Figure 3.5: Typical datapoint images and masks from the Crossing set. Only the inputs at time t are shown, and the label is at $t + 5$. Our models typically take additional inputs at $\{t - 2, t - 4\}$.

Chapter 4

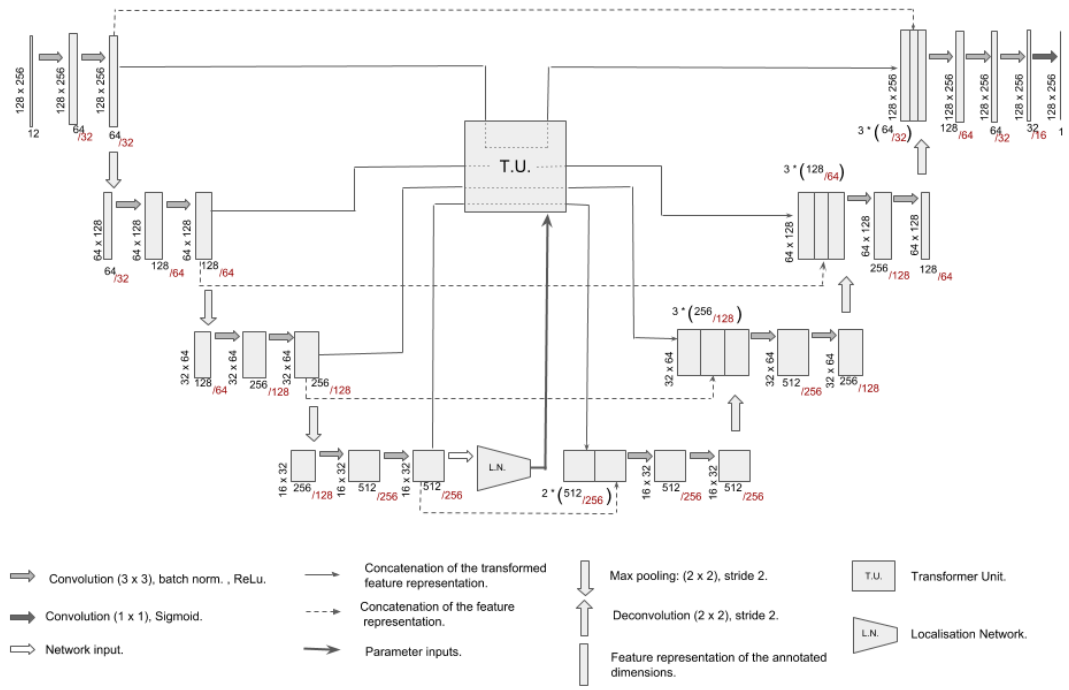
Prediction Model

This chapter focuses on describing the different components of the model we use in order to make our predictions. In particular, we delve into the details of our architecture, loss function, evaluation metrics and implementation that are recurrent across our experiments, justifying our choices along the way.

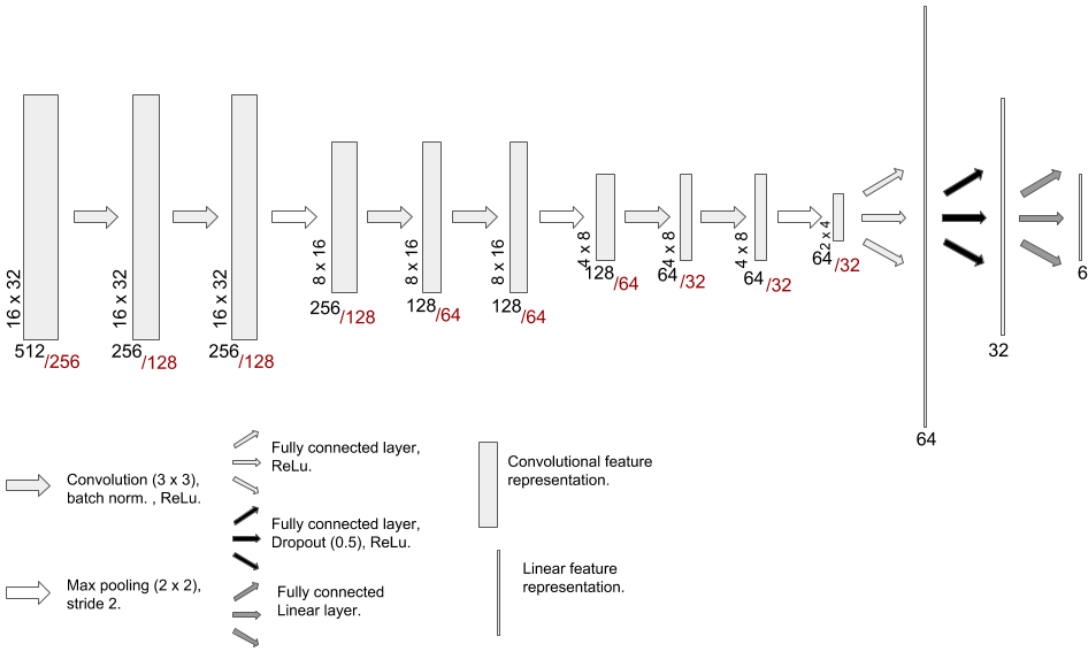
4.1 Architecture

Framing our task in the image segmentation framework allows us to utilise and build off of the available tools in the segmentation literature in creating our model. As described in sections 2.1.1 and 2.1.2, skip connections and fully convolutional architectures have had great success in the segmentation task by balancing the availability of global and local information. In order to utilise these principles effectively, we choose to base our architecture off U-Net [85], which has a simple yet effective configuration that is easily extensible. In extending this model we note that in "traditional" segmentation the target instance and resulting mask are static and overlapping. In our context, the predicted mask is often required at a different spatial location, where the receptive field of fine-grained features might not have access to representations of the object to segment. In other words we make the following hypothesis: Features of the instance to segment are often not available at its future location where the segmentation is required, therefore hurting the ability of a segmentation network to do its task. We further conjecture that we can fix this issue by incorporating a spatial transformer module (c.f. section 2.1.3 and [44]) into the architecture, therefore giving the network the option to shift features as necessary. Our final configuration, which we name SpatialNet, is depicted in Fig. 4.1, which we next describe in detail.

Unless otherwise specified, inputs to our models are 12 channel tensors made up of 3 RGB images and 3 Masks designating a specific object at frames $t-4, t-2, t$. Outputs are the predicted segmentation of the designated object at a specific time in the future. This is illustrated in Fig. 3.3, 3.4, and 3.5. Following the principles of U-Net, described in section 2.1.2.2, our architecture can be conceptually divided in an encoder (contracting) block and a decoder (expanding) block. In the contracting part of the network there is a sequence of convolutional and max pooling layers, gradually decreasing the resolution of the feature representations. Each resolution level is composed of two convolution - batch norm. - Relu blocks, with (3×3) kernels and appropriate padding in order to preserve resolution (c.f. section 2.1.1). Note that this is different from U-Net's approach, where resolution is lost at every convolution. The max pooling



(a) The SpatialNet architecture.



(b) The architecture of the Localisation Network (c.f section 2.1.3 and [44]) within SpatialNet.

Figure 4.1: Model architecture. The red subscript annotation designates an alternative configuration.

layers all have (2×2) kernels with a stride of 2, hence halving the values of spatial dimensions. Every pooling operation is also followed by a doubling of the number of channels in the feature representations. Moreover we used two main configurations, differing by the number of channels in each layer, that are represented in Fig. 4.1 though the red subscript annotation. In the expanding part of the network, we

gradually increase the resolution using (2×2) deconvolutional layers with stride 2. Each deconvolution is again followed by two convolution - batch norm. - Relu blocks, with (3×3) kernels and padding as appropriate in order to preserve resolution. At the lowest level, the feature representation is fed to a localisation network (c.f. section 2.1.3 and [44]), which is depicted in Fig. 4.1b. This network regresses 6 parameters corresponding to an affine transformation through a series of convolutional, pooling and fully connected layers. All convolutions use (3×3) kernels, batch norm., ReLus, and padding to preserve resolution. Pooling layers use (2×2) max pooling with stride 2. Dropout is also used where indicated in the figure, and the last layer has a linear activation. The regressed parameters are then fed to a transformer unit, described in section 2.1.3 and [44]. Finally, the encoder and decoder modules are connected at each resolution layer by concatenating the up-sampled features in the decoder part with (1) their counterpart in the encoder and (2) their counterpart in the encoder after the regressed affine transformation is applied through the transformer unit.

4.2 Loss Functions

For training some of our models we use a combination of the differentiable IoU loss presented in section 2.1.2.3 and a centroid distance loss that we define here. Our main hypothesis is that training would be helped by adding an extra penalty term that would penalise even more heavily motion discrepancies. In other words, by further encouraging the predicted and ground truth centroids to be on top of each other, we are explicitly prioritising getting the motion right first. In making this we were inspired by [99] and [45], which also decomposed their loss into different components of interest. Overall, we get $L_{total} = L_{IoU} + \alpha L_{dist}$, where L_{IoU} is the IoU loss defined in section 2.1.2.3, α is a dynamic scaling factor that keeps both components at the same order of magnitude, and L_{dist} is the distance loss described below.

As we mentioned above, the main idea of L_{dist} is to force the centre of mass of the predicted and ground truth masks to be on top of each other, with the ground truth centroid being another supervisory signal available to us in training time (c.f. chapter 3). In order to do so the position of the predicted centroid in pixel coordinates is found, and then the squared Euclidean distance [32] is taken with the ground truth. Formally, we have the following.

Let M_0 be the model output map of sigmoid activations, such that

$$M_0 = \begin{bmatrix} m_{11} & m_{12} & m_{13} & \dots & x_{1n} \\ m_{21} & m_{22} & m_{23} & \dots & m_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{d1} & m_{d2} & m_{d3} & \dots & m_{ln} \end{bmatrix}$$

Then consider the flattened vector $M = [m_{11}, m_{12} \dots m_{21} \dots m_{ln}]^T$, the position vector in pixel coordinates $\mathbf{p}_{i \times j} = \begin{bmatrix} x_{1,i} \\ x_{2,j} \end{bmatrix}$ such that m_{11} is the output value at position $\mathbf{p}_{1 \times 1}$, the position matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_{1 \times 1} & \mathbf{p}_{1 \times 2} & \dots & \mathbf{p}_{l \times n} \end{bmatrix},$$

and the ground truth centroid coordinates $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$.
Then, the predicted centroid coordinates are given by $\mathbf{p_{pred}} = M'\mathbf{P}$, where M' is the normalised array $M' = M/\sum_{i,j=1,1}^{l,n} m_{i,j}$, and the distance loss is defined as

$$L_{dist} = ||\mathbf{y} - \mathbf{P}M'||^2. \tag{4.1}$$

Furthermore this loss is differentiable, and we can give its gradient with respect to its inputs, M, as necessary for backpropagation.

$$\frac{\partial||\mathbf{y} - \mathbf{P}M'||^2}{\partial m_{i,j}} = \frac{\partial(\mathbf{y} - \mathbf{P}M')^T(\mathbf{y} - \mathbf{P}M')}{\partial m_{i,j}} \tag{4.2}$$

$$= \frac{\partial(\mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{P}M' + M'^T\mathbf{P}^T\mathbf{P}M')}{\partial m_{i,j}} \tag{4.3}$$

$$= \frac{2m_{i,j}}{\sum_{l,s} m_{l,s}}\left[-\mathbf{y}^T\mathbf{p_{i\times j}} + \frac{(\mathbf{y}^T + \mathbf{p_{i\times j}^T})\mathbf{P}M}{\sum_{l,s} m_{l,s}} - \frac{M^T\mathbf{P}^T\mathbf{P}M}{(\sum_{l,s} m_{l,s})^2}\right]. \tag{4.4}$$

In practice, we use Pytorch's [74, 68] predefined operations in order to construct the loss, and backporpagation happens automatically through the implementation of these modules.

4.3 Evaluation Metrics

In order to investigate the performance of our models we have three principal metrics, which were also mentioned in section 3.3: (1) The mask IoU between the prediction of our model and the ground truth mask, (2) the corresponding boundin box IoU, and (3) the mask centroid distance between the ground truth and the prediction. Although all three metrics showcase similar information, namely how much the model was able to capture the motion and shape of the designated agent, differences in their properties allow us to draw deeper insights through their simultaneous use.

The centroid displacement is an absolute measure, in pixels, of the difference in position between the centroids of the predicted and true masks. It mirrors the distance loss described in section 4.2, and is calculated the same way, except for the final scaling. The mask IoU (mIoU) is a direct measure of the correctness of our prediction. Calculated as shown in section 2.1.2.3, a mask IoU of 1 would indicate a perfect prediction from our model. The bounding box IoU (bbIoU), as the name indicates, is the IoU measure obtained after we convert masks into bounding boxes. In order to do so we use the largest and smallest values of the mask in each dimension to create an encompassing rectangle. While the mask IoU treats the shape deformation and motion of an object equally, the bounding box IoU strips away some of the shape information and mostly focuses on how well the motion is predicted. Moreover, considering both measures simultaneously allows us to draw deeper insights:

- If both the bounding box IoU and the mask IoU are low, it is a clear indication that the prediction is of poor quality, and vice-versa.
- If the mask IoU is low while the bounding box IOU is high, this indicates that the model managed to learn the movement pattern and relative scale the agent in motion, but has failed to capture its shape deformation.

- If the bounding box IoU is low while the mask IoU is high, this is an indication that most of the prediction is correct, but there is dramatic failure where some mass is predicted at completely unreasonable locations. This is illustrated in Fig. 4.2.

Figure 4.2: A case where the bounding box IoU will be lower than the mask IoU: Most of the mass is located at the correct place (top left), but there is a component at a widely different location (mid right).

4.4 Implementation details

In implementing our network, we use the Pytorch deep learning framework [74, 68]. Unless otherwise specified, inputs and and labels are both re-sized to a resolution of 128×256 before being fed to the net. We train the model during a maximum of 65 epochs, using a batch size of 32, using He initialisation [36], and the Adam [50, 77, 68] optimiser with a learning rate of 0.001. In order to increase the variability of our data, we make use of data augmentation [102, 70] during training. Specifically, we perform random cropping, horizontal flipping, and rotation of the inputs. We perform the cropping in a way as to preserve the aspect ratio of the images, by cropping two or four adjacent edges as a time. Rotation is capped at $3°$, and flipping is triggered with probability 0.5. We also implement an early stopping [31] mechanism to combat overfitting if necessary, and dropout is set to 0.5 whenever used.

Chapter 5

Principal Results

This chapter is dedicated to presenting the main results for our investigation. Starting with a broad overview of all our findings and baselines, we then focus on the principal result, comparing our model with baseline alternatives.

5.1 Overview

Our approach can be distilled down to a main experiment aimed at comparing our model to baselines, and a set of introspection experiments aimed at (1) expanding the scope of our model, and (2) investigating its behaviour under various scenarios. In this section we give a summary of each experiment, redirecting the reader to the relevant sections, and present an overview of all our findings in table 5.1.

Our primary result (c.f. section 5.2) focuses on quantitatively and qualitatively assessing the performance of our model. In this process, we compare our approach to baselines that consist of outputting the input mask, and using a simple U-Net [85] variant. Assessing against several of our datasets, we find that our model consistently outperforms both (in all but one case, c.f. section 5.2), as is also apparent in table 5.1.

In the experiment of section 6.1, we investigate expanding the scope of our model to multiple time predictions. In order to do so we use a latent variable in the input that indicates the future time to predict (c.f section 6.1.1).In sections 6.2 and 6.3, we test our model's behaviour when we add more quantity and variability to our data, and when we only make use of masks for the inputs. We find that using mask inputs hurts the performance of the model, while using additional data increases it. Finally, in section 6.4, we investigate the effect of using Resnet [37] features pre-trained on ImageNet [17] as inputs to our SpatialNet.

5.2 Primary Result

In this section we investigate the performance of our model, comparing it against two simpler baselines. In doing so we utilise the Football and Crossing datasets for both 0.5s and 1s ahead predictions, and we assess quantitative and qualitative results on the corresponding test sets (c.f. section 3.2.3).

Table 5.1: Overview Table of the *mean bounding box IoU* between predictions and ground truths. *SpatialNet*Δt indicates the SpatialNet configuration with multiple time predictions. *SpatialNetM* refers to SpatialNet with only mask inputs. *SpatialNetL* is the SpatialNet model trained on the Large Crossing data. *SpatialNetR* designates the model with SpatialNet operating on the pre-trained Resnet feature encodings.

	Crossing data		Football data		
Model	0.5s pedictions	1s pedictions	0.5s pedictions	1s pedictions	Relevant section
Input Baseline	58.5 %	46.6 %	56.5 %	44.9 %	5.2
U-Net Baseline	79.2 %	65.5 %	65.3 %	51.5 %	5.2
SpatialNet	80.4 %	68.6 %	67.8 %	51.9 %	5.2
SpatialNetΔt	77.0 %		60.0 %		6.1
SpatialNetM	79.2 %	66.7 %	66.9 %	51.0 %	6.2
SpatialNetL	82.5 %	74.3 %	N/A	N/A	6.3
SpatialNetR	80.2 %	N/A	66.8 %	N/A	6.4

5.2.1 Setup

5.2.1.1 Models and Datasets

As described in detail in chapter 4, our model consists of the SpatialNet architecture trained with the $L_{IoU} + \alpha L_{dist}$ loss combination. We assess it's performance against four data configurations, namely the Football and Crossing sets with either 0.5s or 1s ahead predictions. An important note is that in the context of the Football data we showcase the shallower SpatialNet performance (red channel annotations in Fig. 4.1a), while when referring to Crosssing data the deeper SpatialNet is used. The reason for this are early experiment indications of a better performance for these configurations. We believe that the underlying cause might be the larger volume and higher variability of the Crossing data, but we differ further investigation to future work (c.f. section 7.2).

5.2.1.2 Baselines

We compare our model against two baseline metrics: (1) using the input mask as the prediction, and (2) using a variation of the U-Net for inference. For our simpler baseline then we take the input mask at time t as the output mask, and compare it the ground truth at $t + 5$ or $t + 10$. This gives the performance of the trivial solution, which any "intelligent" system would need to beat. As a more competitive baseline, we use a slight variant of the U-Net. In fact in implementing this model we are inspired by the implementations in [41] and [60], and hence use the specifications from [85] with the following differences:

1. Padding is used where appropriate in order to match the spatial dimensions of the expanding and contracting part of U-Net, negating the need for cropping before concatenation (c.f. section 2.1.2.2),

2. A shallower overall network is used, with the resolution of the inputs only being reduced three times, and the maximum feature channel depth attained being 512 (c.f. Fig. 2.1),
3. The output map consists of a single channel, giving the mask of the future instance being segmented.

We train this "shallow" U-Net using the IoU loss and the specifications detailed in section 4.4.

5.2.1.3 Quantitative evaluations

In order to quantify the performance of our model and baselines, we use the metrics described in section 4.3. In fact, we calculate the mean mask IoU, the mean bounding box IoU and the mean centroid distance for the test set predictions of each of the models, and report them in table 5.2. In order to investigate further, we break down the test examples into three categories:

1. *no movement*, examples where the input and the future ground truth have a bounding box IoU above 90%,
2. *moderate movement* examples, where the input and future bounding boxes overlap at 40% to 60%,
3. *high movement* examples, where the input and the future bounding boxes do not overlap.

We run the shallow U-Net baseline and the SpatialNet on these subsets, and report the histograms of the number of datapoints falling into different bounding box IoU value ranges. We found that the corresponding mask IoU histograms are largely redundant, and we omit them from the analysis. For completeness, these are shown in appendix A.

5.2.1.4 Qualitative evaluations

In our qualitative evaluation, we first showcase a couple of "detailed" datapoint predictions in Fig. 5.3 and 5.4 in order to illustrate the exact input/output relationships for our model. In these diagrams we show (1) the input image and mask at time t of a test set datapoint, (2) the full scale 256×455 ground truth, (3) the 128×256 re-scaled label used by our network, (4) the 128×256 raw sigmoid output, (5) the 256×455 upsampled and thresholded prediction, and finally (6) an amalgamation of the input (blue), output (red), and label (green) masks as well as the corresponding label and output centroids. Regarding (6), it is also important to note that overlapping colours would combine, and hence a yellow region indicates an overlap between the model's prediction and the ground truth, a purple region an overlap between the prediction and the input, and a white region an overlap between all three masks.

We further illustrate several of our predictions in Fig. 5.5. In these diagrams, we superimpose an RGB image at time t with the future segmentations of several object instances in that image. In other words, each mask represents our prediction of where the object will be in the image in the future. Taking this idea further we have created two videos from the test sets that we make public online[1][2]. We note that in the videos all the masks are brought down to the same colour, and that they only concern 0.5s ahead predictions.

Finally, to further investigate the behaviour of our model, we perform a stress test by making "synthetic" datapoints (c.f Fig. 5.6). From the Crossing test set, we select two sequences where a car proceeds

[1]Football dataset predictions: `https://www.youtube.com/watch?v=45lCqqchYjM&frags=pl\%2Cwn`

[2]Crossing dataset predictions: `https://www.youtube.com/watch?v=4Q7eRbBr6GM&frags=pl\%2Cwn`

uninhibited straight through the pedestrian crossing. From these sequences, we select a set of frames at $\{t-4, t-2, t\}$, just before the car reaches the crossing lines. As a control experiment, we use those to create suitable inputs to our model and perform the inference. Then, we take other sequences with people walking across and compose them into the selected frames, in a way that would block the passage of the car. We create new datapoints out of these manufactured frames and perform inference again, comparing the results with the uninhibited case. In other words, we are testing whether the model would pick up on the pedestrians crossing, and then altering its prediction accordingly.

5.2.2 Results and Analysis

5.2.2.1 Quantitative results

As mentioned in section 5.2.1.3, table 5.2 shows the mean mask IoU, bounding box IoU and centroid distance for each dataset/model combination. It becomes apparent that SpatialNet consistently outperforms our baselines, with an exception for the mask IoU metric on the 1s ahead Football predictions. In fact, in this dataset the performance of the shallow U-Net and the SpatialNet are very close, more so than for any other data configuration. It is not entirely clear at this stage why this occurs, and a detailed investigation the cause is left to future work. Nonetheless, we can conjecture that the future on this dataset is the most uncertain, and we do not have the training volume and variation necessary in order to support the potentially extra capacity SpatialNet would offer.

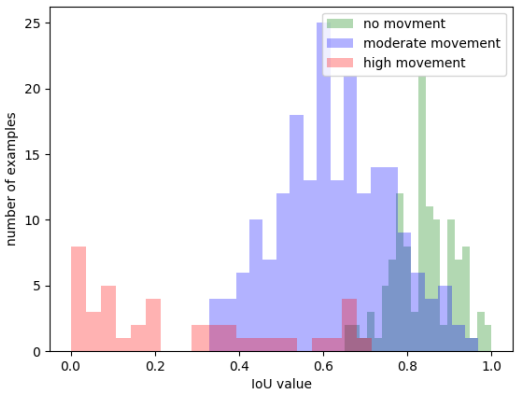
Taking a closer look at table 5.2 allows us to make several more observations. Globally, all our evaluation metrics are consistent with each other. The bounding box IoU is higher than the mask IoU across the board, which aligns with the intuition that predicting the exact shape of an object is harder. It also indicates that there are no significantly many cases where most of the prediction is correct but with some mass predicted at wild locations (c.f. section 4.3 and Fig. 4.2). Furthermore, the relative performances follow the same patterns for all metrics, except in the 1s Football case. In other words, a decrease in mean centroid distance between any two configurations indicates an increase in bounding box IoU and an increase in mask IoU. Supported by this observation, we mainly focus on the bounding box IoU for some of our further enquiries.

Another set of observations is that (1) the long term -1s- predictions and (2) the Football data consistently achieve lower scores. This makes sense conceptually, as (1) longer term predictions are inherently harder, as the number of plausible futures given the input increases, and (2) people pose deformation has many more degrees of freedom than any shape change that could be seen on cars.

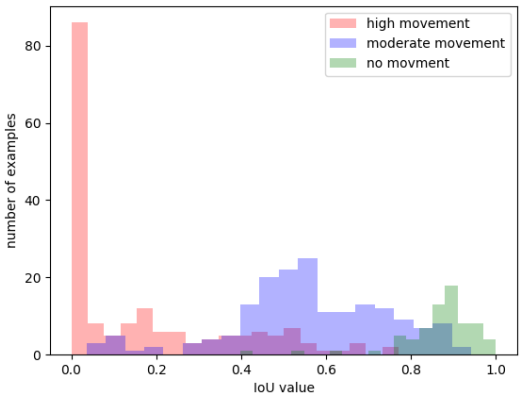
Finally, we see that there is a substantial performance jump when comparing the input baseline to U-Net or SpatialNet, while the performance increase between the latter two is less significant. In fact, looking at the mean bounding box IoU in table 5.2, the average jump in performance between the input baseline and another model is 14.4%, while the jump between U-Net and SpatialNet is 1.8%. In an attempt to gain further insight on this behaviour, we split our test data into high, moderate and no movement examples, and plot the histograms for the distribution of bounding box IoU values in Fig. 5.1 and 5.2.

Table 5.2: Primary results overview: Comparing SpantialNet to simpler baselines on the different datasets. Metrics are taken between the predictions and ground truths. Perfect scores consist of an IoU of 100% and a centroid distance of 0.

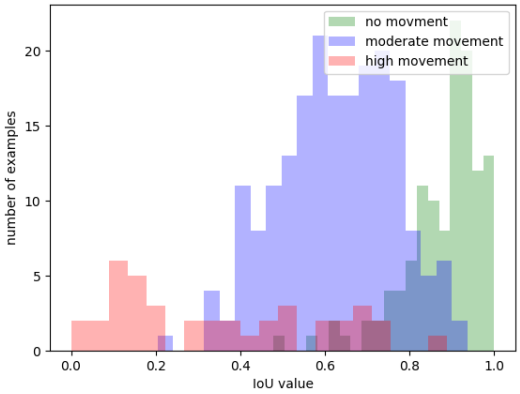
Dataset	Future time	Model	mean mask IoU	mean bounding box IoU	mean centroid distance
Football	0.5s	Input Baseline	48.0 %	58.9 %	6.55px
		Unet	55.5 %	65.3 %	4.9px
		SpatialNet	**58.1 %**	**67.8 %**	**4.0px**
	1s	Input Mirroring	35.1 %	45.4 %	12.2px
		Unet	**42.3 %**	51.5 %	9.0px
		SpatialNet	42.1 %	**51.9 %**	**8.6px**
Crossing	0.5s	Input Mirroring	54.9 %	58.5 %	19.4px
		Unet	76.9 %	79.2 %	4.4px
		SpatialNet	**78.2 %**	**80.4 %**	**3.6px**
	1s	Input Mirroring	42.0 %	44.6 %	39.13px
		Unet	64.4 %	65.5 %	11.8px
		SpatialNet	**66.5 %**	**68.5 %**	**9.0px**



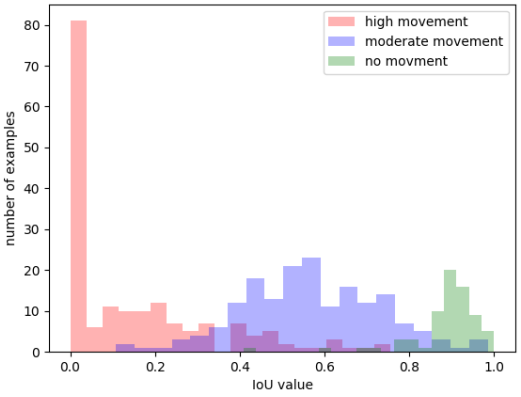
(a) Shallow U-Net, 0.5s ahead predictions.



(b) Shallow U-Net, 1s ahead predictions.

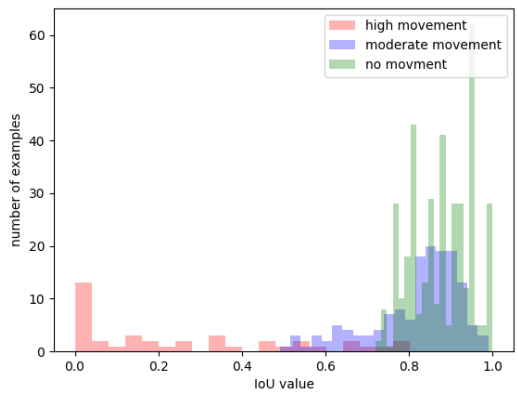


(c) SpatialNet, 0.5s ahead predictions.

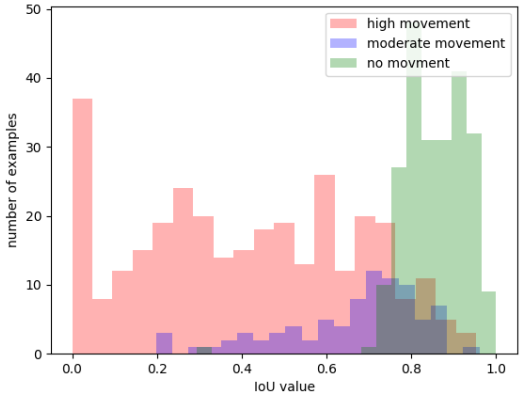


(d) SpatialNet, 1s ahead predictions.

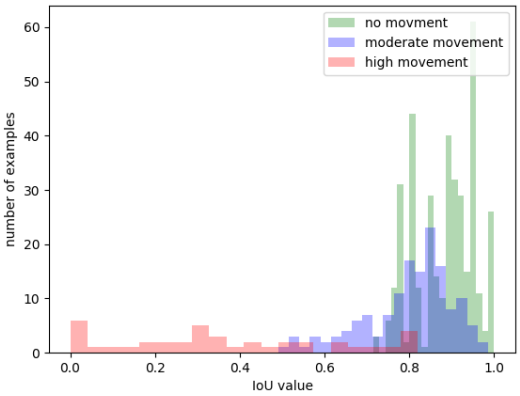
Figure 5.1: Histograms of the number of examples vs the bounding box IoU between the prediction and ground truth on high, moderate and no movement test examples on the Football dataset.



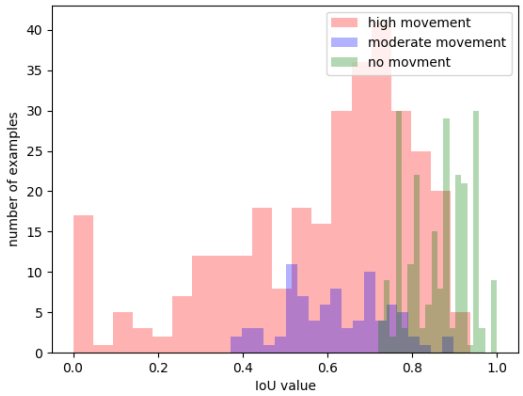
(a) Shallow U-Net, 0.5s ahead predictions.



(b) Shallow U-Net, 1s ahead predictions.



(c) SpatialNet, 0.5s ahead predictions.



(d) SpatialNet, 1s ahead predictions.

Figure 5.2: Histograms of the number of examples vs the bounding box IoU between the prediction and ground truth on high, moderate and no movement test examples on the Crossing dataset.

Fig. 5.1 and 5.2 first reveal that the performance of both models is highly correlated to the level of movement of the underlying instance and the prediction accuracy of inference models. Overall the clear trend is that the higher the movement, the lower the prediction accuracy. In other words, it is easier to predict the future location of an object staying still than an object moving, which is fairly intuitive.

Closer examination can sometimes reveal interesting patterns, such as an interplay between the performance of the model in the different movement categories. This becomes apparent in the 1s Crossing predictions for example. In histograms 5.2b and 5.2d, it is clear that SpatialNet performs much better in the high movement category, but if anything there is a hint of deterioration on the other categories. Additionally, this might begin to explain the high initial jump in performance from the input predicting baseline, followed by the much harder gains between more complex models. Precisely, the initial jump seems to come from a model learning to predict well low or moderate movement instances. Later improvements require learning to accurately predict higher levels of movement, while maintaining the high level of accuracy on the low ones, which is intuitively a harder task. Finally, we note that even in cases where the prediction accuracy might not be ideal, it does not necessarily mean that the model's prediction is wildly unreasonable qualitatively. This is best illustrated with the examples we showcase in section 5.2.1.4.

5.2.2.2 Qualitative results

Fig. 5.3 and 5.4 illustrate the inputs and outputs of our SpatialNet for two datapoints in the Football and Crossing dataset, respectively. In Fig. 5.4 we can see the car prediction is quite accurate, with the yellow region representing an overlap between the model's output and the ground truth prediction. In Fig. 5.3 we showcase an interesting and often observed phenomenon, particularly on the Football set, where the the prediction output trails off behind the actual ground truth. This gives predictions that qualitatively are fairly good, even though quantitatively they might miss the mark.

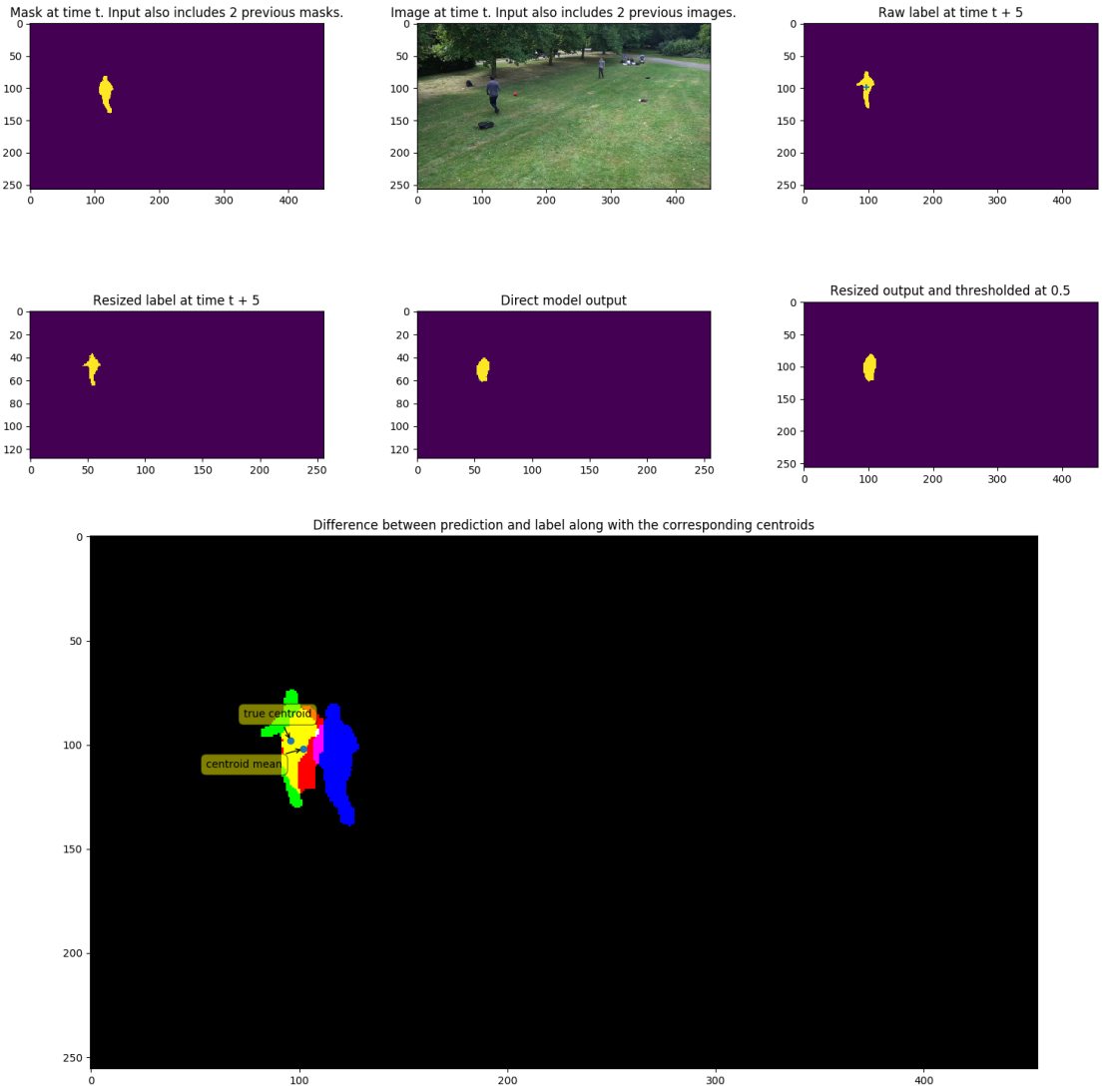


Figure 5.3: Detailed qualitative illustration of the inputs and outputs for a test set datapoint on the Football dataset. From left to right, top to bottom, the figures illustrate (1) The imput mask at time t, (2) the input image at time t, (3) the raw full resolution ground truth, (4) the ground truth resised to the model dimensions, (5) the model's direct sigmoid output, (6) the resized and thresholded output, and (7) a superposition of the input mask (blue), the output mask (red), the ground truth (green), and the locations of the ground truth and predicted centroids. Note that for (7) RGB colours combine (c.f. section 5.2.1.4).

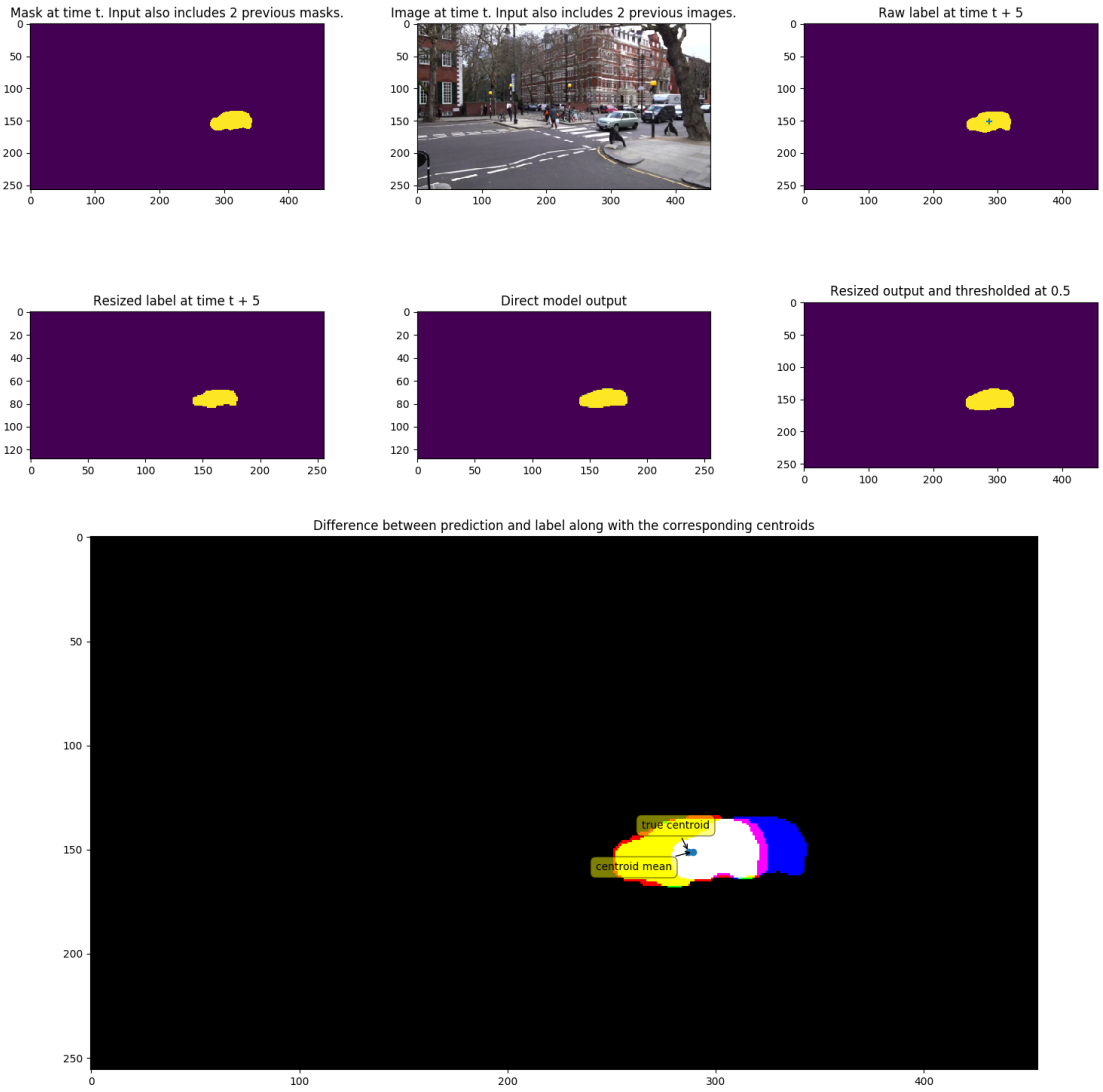
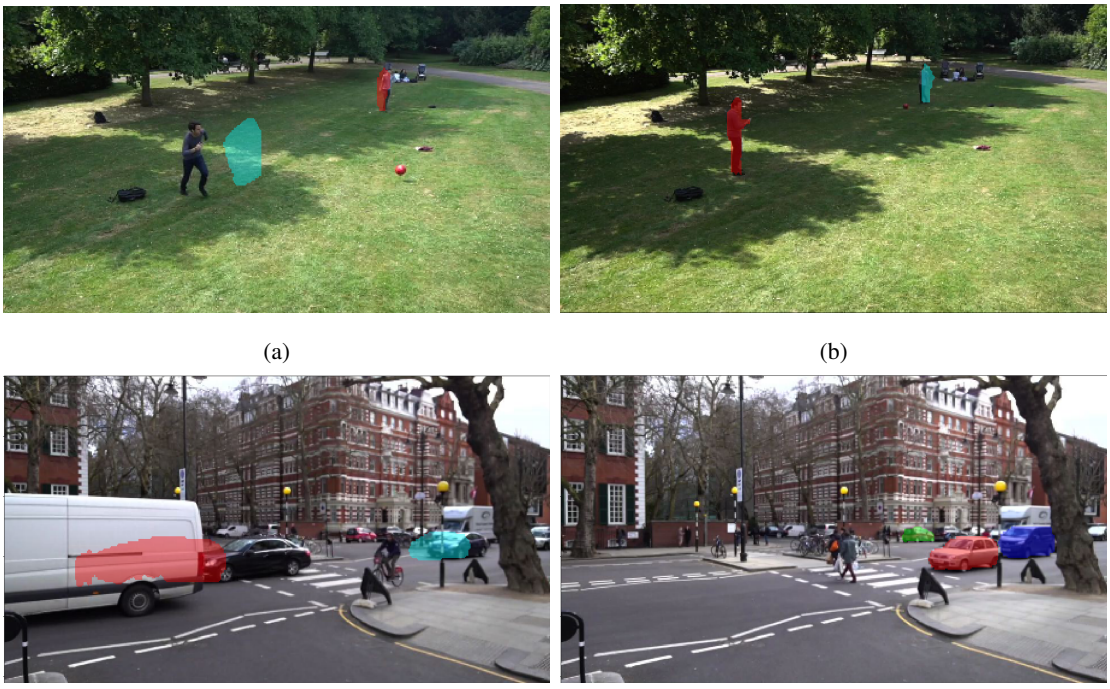


Figure 5.4: Detailed qualitative illustration of the inputs and outputs for a test set datapoint on the Crossing dataset. From left to right, top to bottom, the figures illustrate (1) The imput mask at time t, (2) the input image at time t, (3) the raw full resolution ground truth, (4) the ground truth resised to the model dimensions, (5) the model's direct sigmoid output, (6) the resized and thresholded output and (7) a superposition of the input mask (blue), the output mask (red), the ground truth (green), and the locations of the ground truth and predicted centroids. Note that for (7) RGB colours combine (c.f. section 5.2.1.4).

In Fig. 5.5 we take a different approach, where we overlay the masks of all the predictions for a given frame on top of the corresponding RGB image. This diagram also allows us to illustrate some phenomena we discussed in section 5.2.1.3. To start with, we note that the predictions on the instances that are not in movement are much more precise than when heavy movement occurs. This is consistent with our previous observation that high movement examples obtain lower scores. We further observe that this phenomenon is much more pronounced for the Football data than for the crossing data. This might partially explain why the Crossing dataset performance scores are consistently higher. Moreover, conceptually this makes sense, as human pose changes significantly with time, while the shape of the car stays largely the same, except for perspective effects.



(a) (b)

(c) (d)

Figure 5.5: Representative examples of predictions made with SpatialNet on the 0.5s ahead Football (top) and Crossing (bottom) datasets. Note that predictions on different objects are represented by different coloured masks.

Finally, as explained in section 5.2.1.4, we examine the behaviour of our model by taking scenes where a car passes uninhibited through the crossing, and splicing in pedestrians to see whether the prediction will change. This is illustrated in Fig. 5.6, where the top row represents the uninhibited scene while the bottom row gives the predictions on the manufactured datapoints. As we can see from the figure, the model fails to pick up on the difference and virtually the same prediction is returned. A thorough investigation of the causes are left to future work. Nevertheless, here we can provide several hypotheses on what might be creating this behaviour. A simple explanation could come from the quality of the splicing. Due to time constraints and available resources, we were not able to create high quality synthetic data. The pedestrians splices are fairly irregular, with occasionally missing parts, and we can also notice some ghosting in the image. Although it seems unlikely, such unrealistic RGB signals might cause the model to fall back information obtained mostly by the masks, which would explain the similar behaviour. Another explanation might be that the model overly relies on the information provided by the mask inputs in general, and we investigate this avenue further in section 6.2. Lastly, and perhaps most likely, this behaviour might stem from the fact that we are forcing a situation that does not naturally occur in the data, and hence that the model has not been exposed to during training. In other words, in the input images the car is moving as if there was no pedestrian on the crossing. If pedestrians are about to cross, cars start to decelerate long before, and the motion cues in the image are those from stopping. Here, we are presenting the model with a situation where all the motion cues point to an uninhibited crossing,

while splicing pedestrians to block the way. In reality, this would correspond to the situation where a distracted driver would fail to notice pedestrians until the last instant, and would be forced to take a hard break action. No such situations are occurring in our training set, making this manufactured example an outlier. In this scenario, it is not that the model has failed to learn to accurately predict the future given the training set, but rather that the training set is inadequate for that case. Although it would be quite hard to find near-accident scenes to train explicitly for this, in section 6.3 we investigate what happens when we increase the volume and variability of the data. We find that this significantly improves the performance of our model, which is at least consistent with the conjecture here.

Figure 5.6: Stress test with manufactured test examples. The top row shows the input image and uninhibited predictions from datapoints that are naturally occurring in the dataset. The bottom images are created from those on top by incorporating a splice of pedestrians on the crossing. Predictions are then made from those manufactured datapoints.

Chapter 6

Introspection Experiments

In this chapter we explore several avenues that aim to expand the scope of our model, investigate its behaviour in different situations and test alternative architecture constructs. A complete set of experiments would also warrant a detailed comparison with the other works in the space [57, 56, 8, 45, 63], and which we reviewed in section 2.2.3. However, due to time and resource constrains we choose to focus on experiments revealing the inner workings of our solution, while leaving such investigations to future work (c.f. section 7.2). As such, in this chapter we delve into (1) Expanding the scope of our model to multiple time predictions, (2) Testing the effect of the RGB input channels on our predictions, (3) Testing SpatialNet's behaviour if we increase the number and variability of available data, and (4) Investigating the use of Resnet [37] feature representations.

6.1 Experiment 1: Multiple time predictions

6.1.1 Motivation and Setup

In this first experiment we expand the scope of our task to incorporate multiple time predictions. In other words, we want to use our model for predicting segmentations at a variable time into the future. In order to do so, we propose using a latent variable as an extra channel in the inputs that would indicate the future time of the prediction. This technique and other similar approaches can already be found in the literature [22, 99]. Nevertheless, to the best of our knowledge, we are the first to investigate such a method for predicting different timesteps in the context of future image segmentation. As described in section 2.2.3, most approaches consist of an autoregressive framework, where short term predictions are used recursively as inputs for a longer term prediction. The exception to this is [63], where the approach is unclear, but where we suspect that either the same recursive framework was used, or independent networks were trained for the different times, akin to our method so far.

Expanding the scope of our model to multiple time predictions is conceptually simple using a latent variable. We create a new channel on the input, which has a single value in all positions, and that indicates the future time to predict. During training with this new configuration, the ground truth would then match the time indicated by the latent variable. In an ideal setting where we would have ground truths available for each future frame, this latent variable could be an integer indicating how many frames into the future to predict. As we only have ground truths for 5 and 10 frames ahead however, we restrict

ourselves to having a binary latent variable, with a channel of zeros indicating a prediction $0.5s$ ahead and a channel of ones indicating a prediction $1s$ ahead.

Excluding the use of the latent variable and corresponding ground truth, the rest of the setup remains mostly as described in section 5.2.1. We still use both the Football and Crossing datasets, the same evaluation metrics, and the same qualitative visualisations. Nonetheless, we now compare our new results with the values previously obtained by SpatialNet, which is illustrated in table 6.1. We also note that three values are given for the previous model, one for $0.5s$, one for $1s$ predictions, and one taking the average of the two.

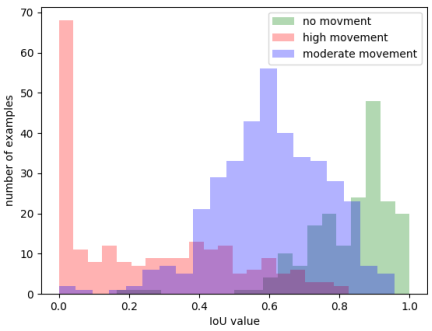
6.1.2 Results and Analysis

The first thing that becomes directly apparent from table 6.1 is that the performance of the model on the harder problem of multiple time prediction is better than the average of the individual performances. In other words, SpatialNet trained on both timesteps performs better than two separate SpatialNet models trained on each timestep independently. This is consistent across all metrics and for both datasets. We interpret this as a sign of reducing overfitting to the training set: By training on the harder problem we practically double the training examples. This also increases the variability of the data, and perhaps forces the model to learn better discriminating features. Naturally, this result begs for exploring training with more varied data, an experiment that we present in section 6.3.

Table 6.1: Mean mask IoU (mmIoU), mean bounding box IoU (mbbIoU) and mean centroid distance (mDis) between the predictions and ground truths for the SpatialNet and SpatialNet with multiple time predictions (SpatialNetΔt) on the test sets of the Crossing and Football data. Perfect scores are 100% IoU and 0px distance.

		Crossing data			Football data		
Metric	Model	0.5s ahead	1s ahead	Average	0.5s ahead	1s ahead	Average
mmIoU	SpatialNet	78.2 %	66.5 %	72.35 %	58.1 %	42.1 %	45.1 %
	SpatialNetΔt		75.3 %			51.2 %	
mbbIoU	SpatialNet	80.4 %	68.6 %	74.5 %	67.8 %	51.9 %	59.85 %
	SpatialNetΔt		77.0 %			60.0 %	
mDis	SpatialNet	3.6px	9.0px	6.3px	4.0px	8.6px	6.3px
	SpatialNetΔt		5.6px			5.4px	

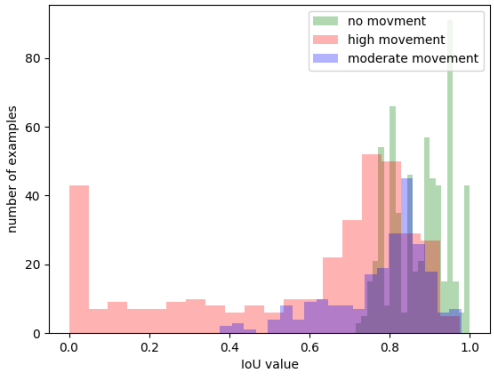
It is also interesting to look at the histograms of the number of test examples versus the bounding box IoU for each movement category, as described in section 5.2.1. Those are illustrated in Fig. 6.1a and 6.2a. Both exhibit the expected behaviour, consistent with our previous results. Finally, Fig. 6.1b and 6.2b show qualitative predictions at both timesteps for the present instances, and we can note how the shape deformation tends to be less precise in the long term inferences.



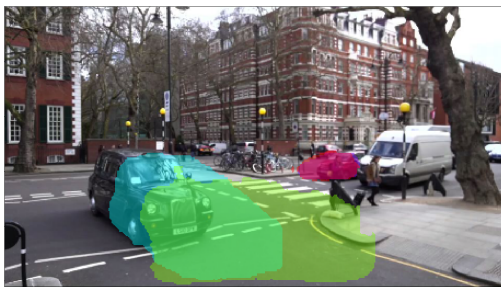
(a) Histogram.

(b) Qualitative plot.

Figure 6.1: Supporting diagrams for the performance assessment of the multiple time predictions model on the Football dataset. On the left, we show the histogram of the bounding box IoU between predictions and ground truths on the different movement categories on the test set. On the right, we show the qualitative plot of a random test example. Different time predictions are shown through different coloured masks.



(a) Histogram.



(b) Qualitative plot.

Figure 6.2: Supporting diagrams for the performance assessment of the multiple time predictions model on the Crossing dataset. On the left, we show the histogram of the bounding box IoU between predictions and ground truths on the different movement categories of the test set. On the right, we show the qualitative plot of a random test example. Different time predictions are shown through different coloured masks.

6.2 Experiment 2: Using Masks only

6.2.1 Motivation and Setup

We believe it is important to investigate the impact that the available RGB information is having to the prediction accuracy. Input masks already carry a great amount of information regarding the motion of the designated instance. For example, shape, current movement dynamics and deformation rate can all be reasonably inferred just looking at the masks. That said, context information is also very important in ultimately making "intelligent" predictions. A clear example in our context is when pedestrians are just past the crossing: The car might have been still, but the prediction should indicate the car to go forward. A system overly relying on masks would not be able to pick up this extra information, and would wrongfully predict the car to stay still. We also get a hint for this predicament in our findings during the stress test in section 5.2.2.2. All in all, it is not unreasonable to think that our model perhaps only chooses to use the simpler information available from the masks, and hence we choose to test this explicitly. In fact, even though the next two experiments (c.f. sections 6.3 and 6.4) are also indirectly related to this idea, in this section we directly test the impact of the RGB information. In other words, we compare the previous performance of our model with one subjected only to mask inputs.

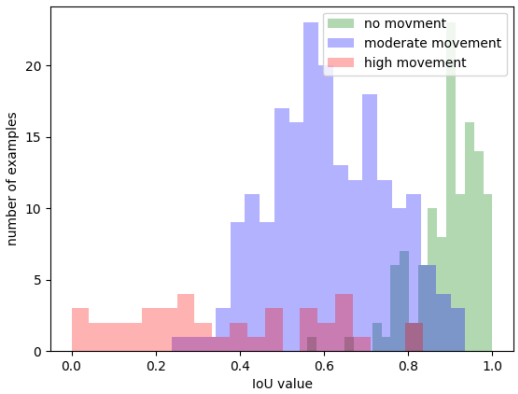
As such, we now have a 3 channel input tensor to SpatialNet, made of the masks at time $\{t-4, t-2, t\}$. The remaining of the setup is as described in section 5.2.1. In table 6.2, we compare the results for the mask-only configurations with their full-input counterparts. In Fig. 6.3 we further show a more detailed performance breakdown, with the distribution of bounding box IoUs for each movement category.

6.2.2 Results and Analysis

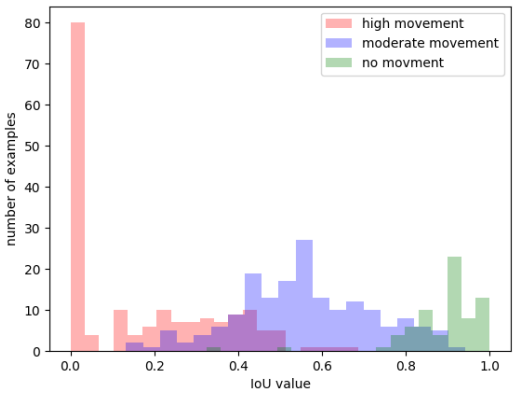
In table 6.2, we observe that by using the RGB as part of the input there is a slight increase in performance across the board. As such, we conclude that our model has learned to somewhat capture cues from the image context. The small increase however (about 1% on average), perhaps hints that there is further room for improvement in this domain. Other clues to support this are (1) the model's failure to alter its prediction during the stress test in section 5.2.2.2, and (2) the small increase in performance between SpatialNet and U-Net baseline (c.f. table 5.2). As we also discuss in section 7.2, our hypothesis is that as the volume and variability of the training test increases, the model learns to better use the vastly larger space of the RGB images and the overall accuracy increases. Nevertheless, there are many possible alternative hypotheses for this small observed increase in table 6.2. For instance, one possibility is that in the test sets not many examples are available where the RGB clues truly offer information that would alter a prediction made by only using the masks. This would keep the performance increase low, even though the model would have already learned to greatly use RGB cues when available. For completeness, and for direct comparison with the diagrams in section 5.2.2.1, we also present the histograms of the bounding box IoU on the different movement categories in Fig. 6.3.

Table 6.2: Mean mask IoU, mean bounding box IoU and mean centroid distance between predictions and ground truths for the SpatialNet and SpatialNet with mask only inputs (SpatialNetM) on the test sets of the Crossing and Football data.

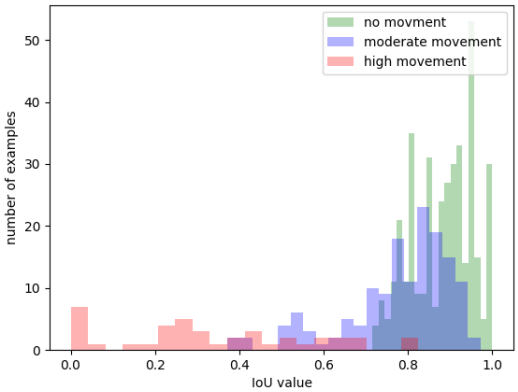
Dataset	Future time	Model	mean mask IoU	mean bounding box IoU	mean centroid distance
Football	0.5s	SpatialNet	58.1 %	67.8 %	4.0px
		SpatialNetM	57.5 %	66.9 %	4.1px
	1s	SpatialNet	42.1 %	51.9 %	8.6px
		SpatialNetM	41.4 %	51.0 %	9.4px
Crossing	0.5s	SpatialNet	78.2 %	80.4 %	3.6px
		SpatialNetM	77.2 %	79.2 %	3.8px
	1s	SpatialNet	66.5 %	68.5 %	9.0px
		SpatialNetM	64.7 %	66.7 %	10.4px



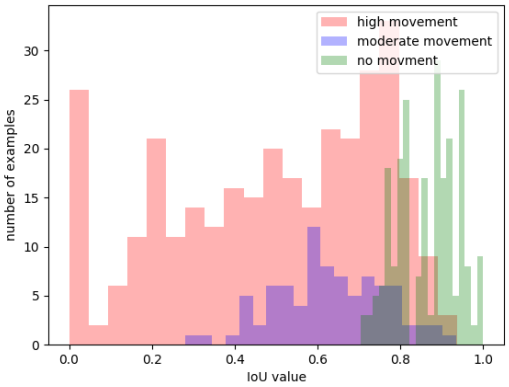
(a) Football dataset, 0.5s ahead predictions



(b) Football dataset, 1s ahead predictions



(c) Crossing dataset, 0.5s ahead predictions



(d) Crossing dataset, 1s ahead predictions.

Figure 6.3: Histograms of the number of examples vs the bounding box IoU between predictions and ground truths obtained by the SpatialNet with mask-only inputs. The histograms are divided between high, moderate and no movement test examples from the different sets.

6.3 Experiment 3: Adding more data.

6.3.1 Motivation and Setup

In this experiment, we are investigating the behaviour of our model when subjected to a larger volume and variation of data. In order to do so, we use the Large Crossing dataset, described in Chapter 3. This is important in the quest of identifying the limiting factors on the performance of our model, as it can help us identify the effect of the data on the results we observe.

Our setup here is as described in section 5.2.2.2, with the exception of using the Large Crossing dataset. Because this set contains footage of the crossing from both lanes, not only it increases the number of frames available for training, but the variability of the data too. This is illustrated in Fig. 6.5, where we show one frame for each viewpoint. We further hypothesise that this will help the model better isolate relevant features, improving its generalisation capacity and overall accuracy. In our quantitative analysis, we use both 0.5s and 1s predictions, and compare the performance obtained with its counterpart from the small Crossing dataset.

6.3.2 Results and Analysis

Looking at table 6.3, it is immediately apparent that using the large dataset increases the performance of our model across the board. In fact, we achieve our highest score overall with 82.5% bounding box IoU for the 0.5s ahead predictions.

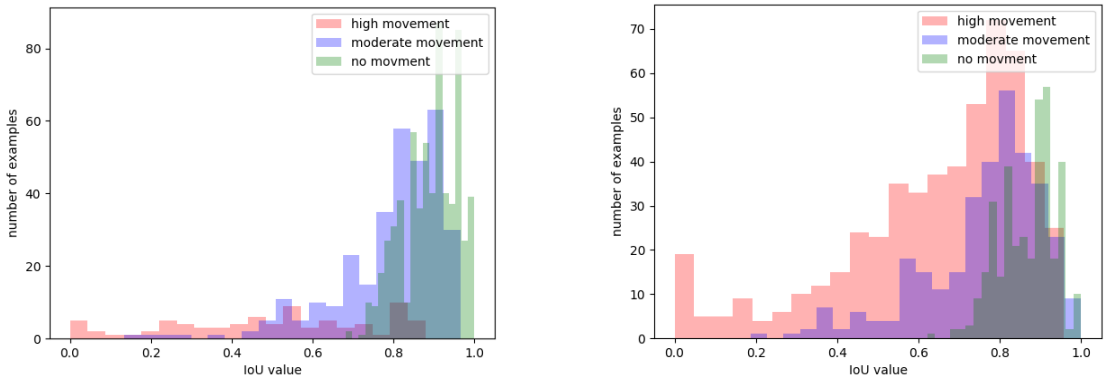
Table 6.3: Mean mask IoU, mean bounding box IoU and mean centroid distance between predictions and ground truths for SpatialNet (which uses the Crossing set) SpatialNetL (which uses the Large Crossing set).

Future time	Model	mean mask IoU	mean bounding box IoU	mean centroid distance
0.5s	SpatialNet	78.2 %	80.4 %	3.6px
	SpatialNetL	80.2 %	82.5 %	3.2px
1s	SpatialNet	66.5 %	68.5 %	9.0px
	SpatialNetL	71.3 %	74.3 %	6.9px

In Fig. 6.4 we show the performance histograms for the bounding box IoU, and in Fig. 6.5 we show some qualitative predictions from the two viewpoints of this larger dataset. Both figures are consistent with the large global performance obtained: most of the mass of the histogram density is towards high IoU scores, and the qualitative examples are convincing. Moreover, specifically for Fig. 6.5, we can see that the predictions for the cars in 6.5b seem to show a larger displacement for the 1s predictions, which is what we intuitively expect.

The results obtained here are in support of, or at least consistent with, our broad hypothesis that as the training data improves the model should learn to make better use of the available information, leading to a performance increase. In other words, our conjecture is that our model's performance is currently

limited by the quantity and variability of available data. The same thematic is explored in the next section, where we use a pre-trained feature extractor in order to create inputs for SpatialNet.



(a) 0.5s ahead prediction

(b) 1s ahead prediction

Figure 6.4: Histograms of the number of examples vs the bounding box IoU value between predictions and ground truths obtained by SpatialNet on the Large Crossing dataset. The histograms are divided between high, moderate and no movement test examples with 0.5 or 1 second ahead predictions.

(a) 0.5s ahead predictions

(b) 1s ahead predictions

Figure 6.5: Qualitative illustrations for the SpatialNet predictions on the Large Crossing dataset. Masks of different colours indicated predictions for different objects in the image.

6.4 Experiment 4: Using Resnet features

6.4.1 Motivation and Setup

We conduct a final experiment by altering our architecture to incorporate Resnet [37] feature maps pre-trained on ImageNet [17]. We conjecture that perhaps using pre-trained features would allow our model to better utilise RGB information, given the limited volume of our dataset. Our new configuration is illustrated in Fig. 6.6, which shows the complete architecture used for this experiment.

In this configuration, the three RGB images at times $\{t-4, t-2, t\}$ are independently passed through the first two blocks of Resnet-18 [37, 2, 79, 68], which downscales the resolution by a factor of 8. To keep some resolution at the end of the process, we also rescale our inputs to 256×512. In order to get the masks to the same dimensionality, a series of three convolutional and pooling layers is applied. The

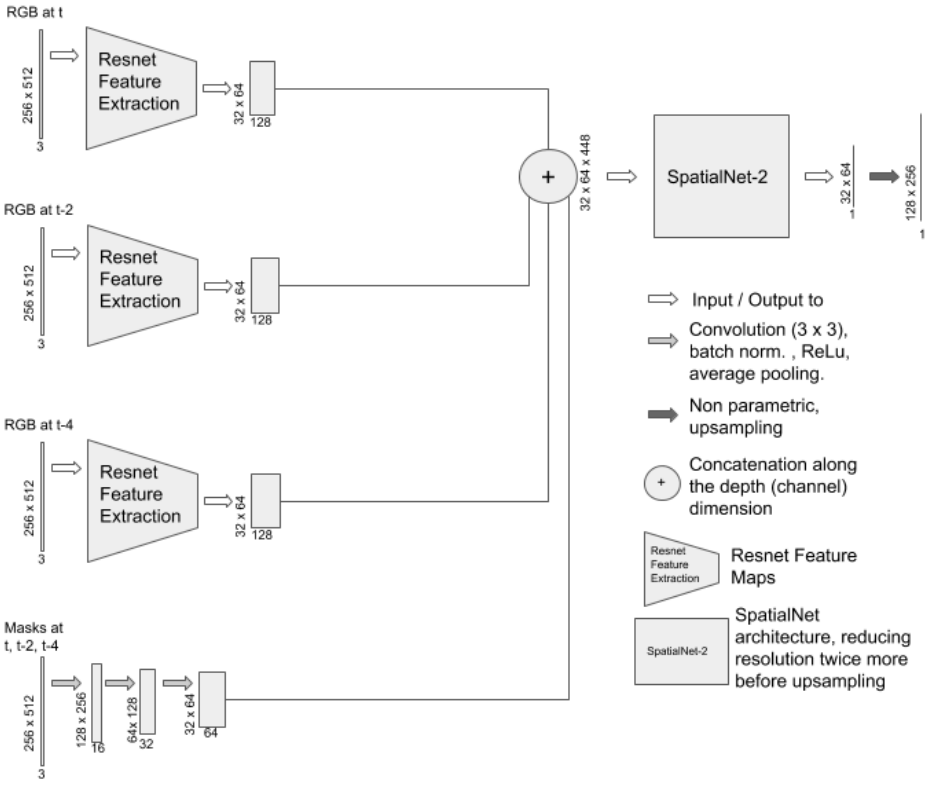


Figure 6.6: Illustration of the SpatialNet on Resnet [37] encoded features configuration.

convolutional layers have 3×3 kernels and a ReLu nonlinearity, and we also make use of batch normalisation and average pooling [31, 78, 68]. All the resulting feature representations are then concatenated to a $32 \times 64 \times 448$ tensor, which is used as an input to SpatialNet-2. SpatialNet-2 is a smaller variant of SpatialNet, where spatial dimensions are only reduced twice. The output of this network is a $32 \times 64 \times 1$ mask, which we upscale using non-parametric upsampling [1].

The remaining setup is similar to section 5.2.1, and we compare the performance of this new configuration with our previous SpatialNet. Table 6.4 summarises the results on the Crossing and Football datasets, and for on 0.5s ahead predictions only. Fig. 6.7 delves deeper into the performance of this model on the different movement categories.

6.4.2 Results and Analysis

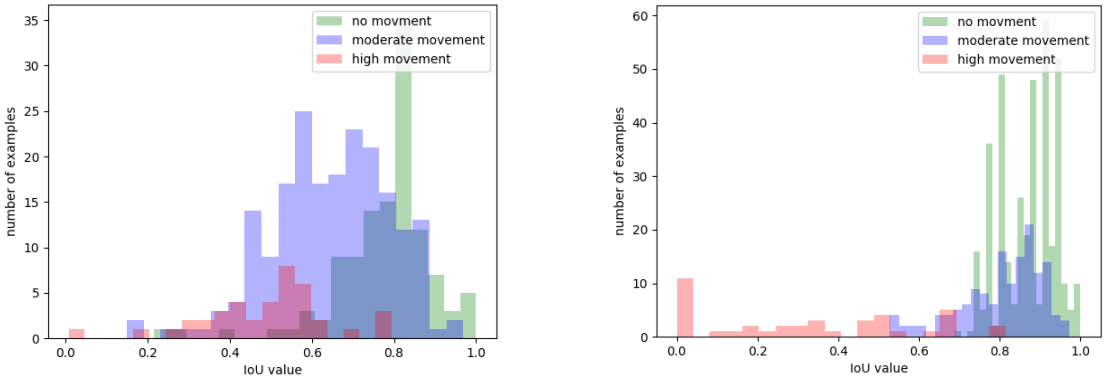
Table 6.4 reveals that using Resnet feature encodings gives results that are practically on par with the standalone SpatialNet model. Another interesting observation stems from Fig. 6.7a, where it appears that, at least on the Football dataset, using feature encodings slightly improves the performance on the high movement data (c.f Fig. 5.1c).

Although the results are not strong enough to draw any definitive conclusions, they illustrate that SpatialNet could be applied directly to feature representations. This is in support of our hypothesis that SpatialNet could be used as part of other frameworks, such as in [56], where it could be used as the F_2F_l networks. Explicit experimentation in this area is left to future work (c.f. section 7.2). Finally, it is important to note the vast amount of degrees of freedom in making this new architecture. It is not

unreasonable for example to think that the number pre-trained layers to incorporate or different ways of downsampling / upsampling the input/output masks might significantly change the performance obtained. Due to time and resource constraints however we are unable to perform such a hyper-parameter search, which then also left to future work (c.f. section 7.2).

Table 6.4: Mean mask IoU, mean bounding box IoU and mean centroid distance between predictions and ground truths for the SpatialNet and SpatialNet on pre-trained Resnet feature encodings (SpatialNetR) on the test sets of the Crossing and Football data with 0.5s ahead predictions.

Dataset	Model	mean mask IoU	mean bounding box IoU	mean centroid distance
Football	SpatialNet	58.1 %	67.8 %	4.0px
	SpatialNetR	58.3 %	66.8 %	3.5px
Crossing	SpatialNet	78.2 %	80.4 %	3.6px
	SpatialNeR	77.4 %	80.2 %	3.6px



(a) Football dataset.

(b) Crossing dataset.

Figure 6.7: Histograms of the number of examples vs the bounding box IoU between the predictions and ground truths obtained by SpatialNet on pre-trained Resnet feature encodings. The histograms are divided between high, moderate and no movement test examples with $0.5s$ ahead predictions.

Chapter 7

Conclusion

7.1 Discussion

In this investigation we treat the problem of future image segmentation using an object segmentation approach. As such, we consider each instance of the target class independently, predicting its segmentation mask at a given time in the future. To the best of our knowledge, this is a different approach from the other attempts in this area of future image segmentation, which have tackled the task using either a semantic or an instance segmentation formulation [57, 56, 63, 8, 45].

For treating the problem we also create a new data bank, which we argue is more appropriate for future image segmentation benchmarking than Cityscapes [16, 14], which was used so far by the few papers in the field. Our data bank, CorrelatedMotions, can be divided in three main subsets, namely (1) a Football set with 5414 frames of two players passing a football, (2) a Crossing set with 8367 frames of a pedestrian crossing scene, and (3) a 15836 frame Large Crossing set that complements the previous with extra footage of the opposite traffic lane. When obtaining segmentation ground truth masks, these three sets are further subdivided into two categories, containing ground truths for 0.5s and 1s ahead predictions respectively. In total, we obtain 6 datasets which we utilise for our subsequent experiments.

Our inference model revolves around an architecture which we call SpatialNet, and which is created by combining the U-Net [85] model with Spatial Transformer Networks [44]. In order to train SpatialNet we use a combination of the differentiable IoU loss [80] with a centroid distance loss between the positions of the mask centroids. In assessing the performance of our model, we use the IoU of the predicted masks with the ground truths, the IoU between the bounding boxes of those masks, and the distance in pixels between their centroids. We organise our approach in a primary experiment and several introspection follow-ups. In the primary experiment, we compare the performance of our model in the Crossing and Football datasets against two simpler baselines. We find that SpatialNet outperforms these in all but one metric from the Football dataset with 1s ahead predictions. We conjecture this stems from having insufficient data for this highly uncertain scenario, but we differ a rigorous investigation for future work. In our introspection experiments, we (1) extend the scope of our model to include multiple time predictions, (2) test the model's performance if only segmentation masks are used, (3) investigate the effect of increasing the volume and variation in the data, and (4) test Resnet [37] pre-trained feature

encodings to create inputs for SpatialNet.

Overall, considering the bounding box IoU metric, the highest average performance obtained is 82.5% on the Large Crossing dataset for 0.5s ahead predictions. Comparatively, on the smaller Crossing dataset, SpatialNet performs best with 80.4%, followed by the SpatialNet on Resnet encoded features with 80.2%. Only using masks as inputs achieves a performance of 79.2%, which is on par with the U-Net baseline. Finally, having multiple times predictions gets to 77.0%, and all of these results largely outperform the trivial baseline of predicting the input mask, which achieves 58.5%.

Further analysing our experiment results, we hypothesise that the current limitation of our model is the volume and variability of available data. We interpret our findings as repeatedly supporting that a more complex problem with more varied data would give comparatively better results: In experiment 1, using multiple time predictions gave a performance that is higher than the average of independently trained SpatialNet networks. In experiment 3, adding a larger quantity of data and a different viewpoint significantly increased the model's performance. Inspired by our observations in experiment 2, we further conjecture that the dynamic at play is that the current dataset is not varied enough for the model to fully capture the context information provided by the input images. In other words, we hypothesise that because the RGB space is much less constrained, having a small amount of data prompts the model to default to the much simpler mask inputs. This is also consistent with the observations in section 5.2.2 where SpatialNet, with arguably a higher capacity, only achieves small improvements over the simpler U-Net baseline.

7.2 Future Work

In this section we are going to explore possible future steps in this investigation, that would contribute in solidifying current findings, delving deeper into the inner workings of our model, and expanding the scope of our task.

The first step in continuing in our line of research would be to expand and further develop CorrelatedMotions, our data bank. In essence, more footage needs to be captured and our API needs to be upgraded to production-quality, so that our data bank can be open sourced and released to the public. In capturing more footage, the same themes of a simple football game and a pedestrian crossing would be kept, but more variability to the scenes needs to be added. For instance, with the Football dataset, variation can be added from using different backdrops, times of the day, players, football colour and viewing angles. For the Crossing dataset, different pedestrian crossings, weather conditions, and viewpoints could add the necessary variation. In the same spirit, ground truths for a wider range of future times could be generated. Having all this extra data would (1) allow us to release a competitive data bank to the public, (2) enable us to train models that do not overfit to a particular scene, and (3) help us in pushing and exploring the limits of our models in terms of performance.

In fact, with access to this larger data bank, we would proceed in further investigating our hypotheses for the limiting factors in our model's performance. As we saw in section 6.3, improving our data did make a significant contribution towards performance, and it is important to explore how far can this

trend go. In the same spirit, it is important to continue investigating the underlying mechanisms that justify the behaviour of our model. A first step is to verify our conjecture that the difference between a better performing model with its weaker counterpart is how well RGB information is incorporated, and that more variability in data is important in that regard. Possible experiments in that space include (1) training side-to-side models with explicitly more or less varied data, (2) masking key parts of the RGB inputs in those models and observing the resulting predictions, and (3) comparing the difference in performance improvement between mask-only and normal training under larger and smaller datasets.

Another important direction to explore is how does our model compare to the other approaches in the literature [57, 56, 63, 8, 45]. In order to thoroughly approach the task, we would need to (1) assess the performance of our model in the Cityscapes [14] dataset, and (2) assess the performance of relevant models in the CorrelatedMotions dataset. An important step in that process is to also make our model as competitive as possible through a full hyperparameter search. Due to resource constraints, this was not possible in our experiments, and hence the numbers reported might be sub-optimal. Furthermore, a hyper-parameter tuning process also makes more sense after capturing more data. Having these available would allow for a more meaningful validation workflow, which in turn would give us a better sense of our model's capacity to generalise. At last, an interesting experiment would be to test our SpatialNet architecture for the F_2F_l networks in [56], and assess whether it would help improving performance.

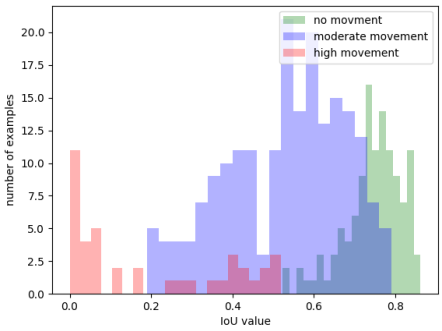
Finally, there are several extensions to our model and other introspection experiments that would be compelling to explore. Initially, after extending our data bank, we could train the multiple time prediction configuration for our model with several different time-steps. Instead of having just a binary latent variable indicating medium or long term predictions, we would include an integer variable with the number of frames in the future for the prediction. Next, we could investigate the effect of including more pre-processing information into the inference process. For instance, thanks to Mask-RCNN [35], we have available the segmented masks for different classes in the input frames. Investigating ways to exploit these either in order to bolster performance or to gain further understanding into the inner workings of our model is an interesting avenue to explore. Last but not least, we could extend the scope of our task to incorporate a probabilistic formulation, similarly to [8] for instance. We believe this is especially important for longer term predictions, where the future locations and shapes of the different agents in the scene becomes increasingly uncertain.

Appendix A

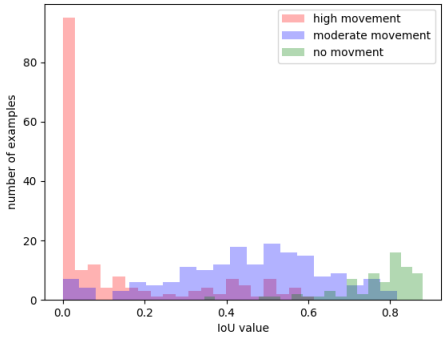
Additional Material

In this appendix we feature the plots showing the histograms of the number of examples versus the mask IoU metric. These plots all mirror the ones shown in the main part of the text, but show the mask IoU metric as opposed to the bounding box IoU. As we briefly mention in section 5.2.1.3, we have not found any significant way that these contribute to or alter our narrative, and hence are only shown here for completeness. We organise the appendix in sections mirroring our main narrative, in order to allow for an easy comparison.

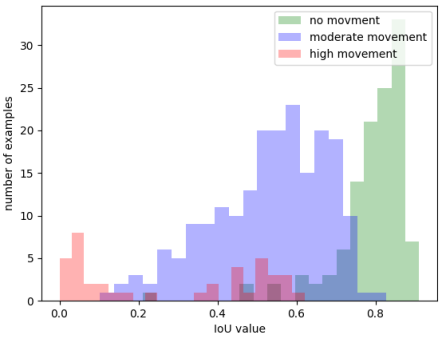
A.1 Primary Result



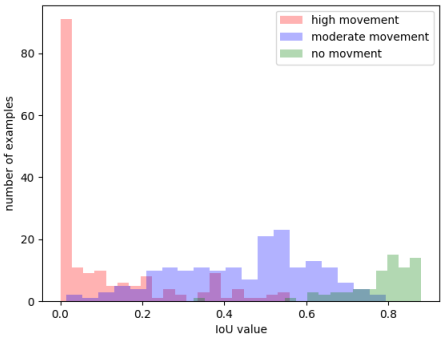
(a) Shallow U-net, 0.5s ahead predictions.



(b) Shallow U-net, 1s ahead predictions.

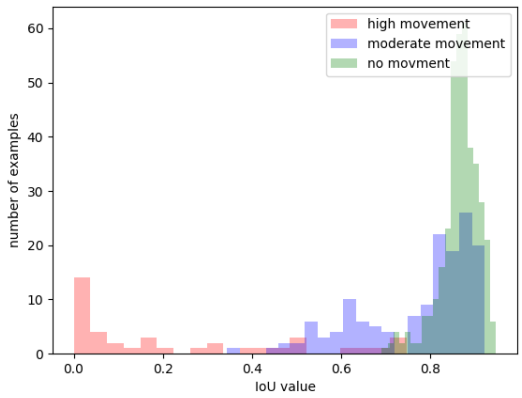


(c) SpatialNet, 0.5s ahead predictions.

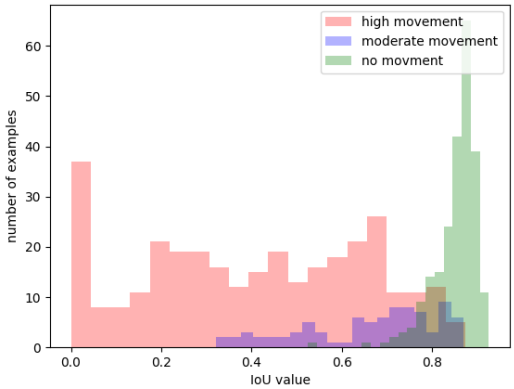


(d) SpatialNet, 1s ahead predictions.

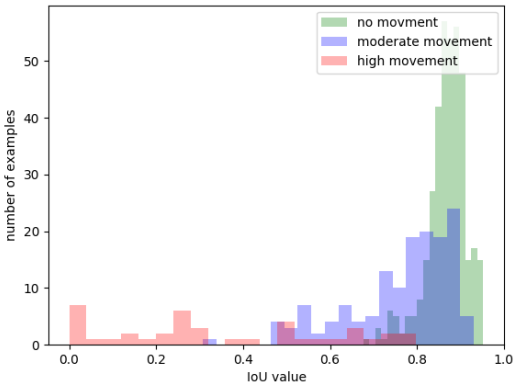
Figure A.1: Histograms of the number of examples vs the mask IoU between the prediction and ground truth on high, moderate and no movement test examples on the Football dataset.



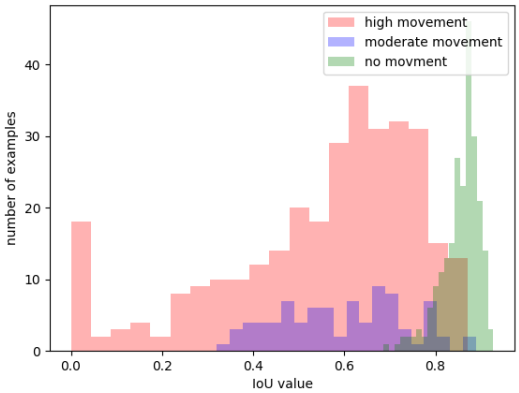
(a) Shallow U-net, 0.5s ahead predictions.



(b) Shallow U-net, 1s ahead predictions.



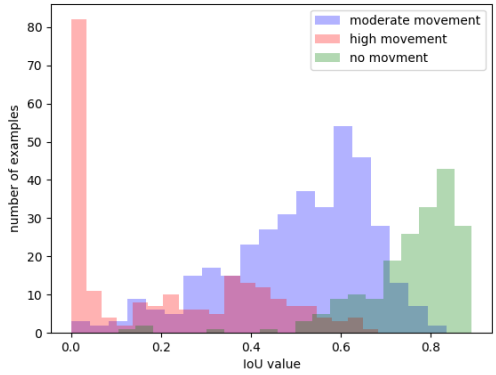
(c) SpatialNet, 0.5s ahead predictions.



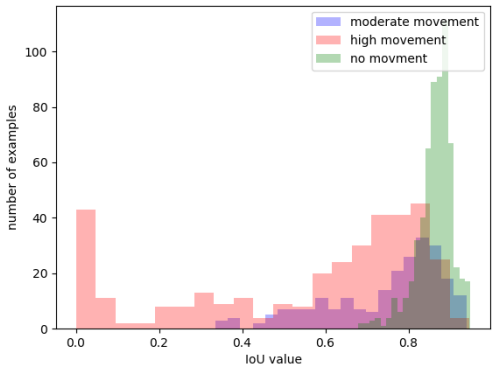
(d) SpatialNet, 1s ahead predictions.

Figure A.2: Histograms of the number of examples vs the mask IoU between the prediction and ground truth on high, moderate and no movement test examples on the Crossing dataset.

A.2 Experiment 1



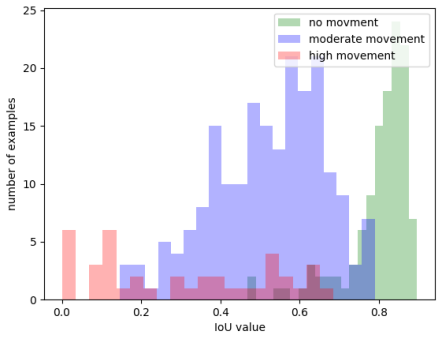
(a) Football dataset.



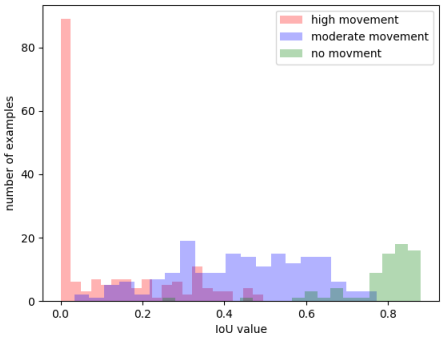
(b) Crossing dataset.

Figure A.3: Histograms of the mask IoU between predictions and ground truths on the different movement categories on the test sets, and for multiple time predictions.

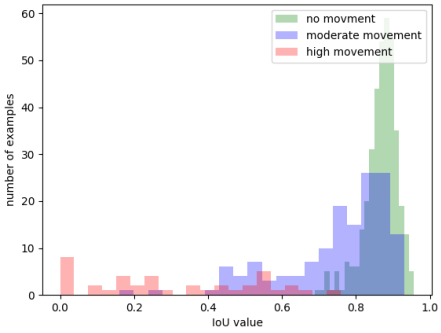
A.3 Experiment 2



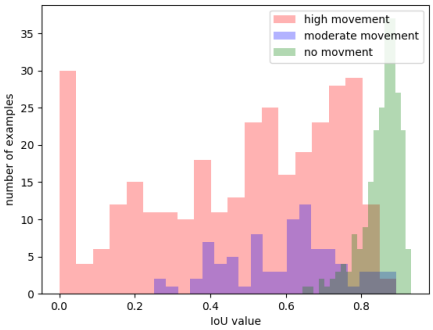
(a) Football dataset, 0.5s ahead predictions



(b) Football dataset, 1s ahead predictions



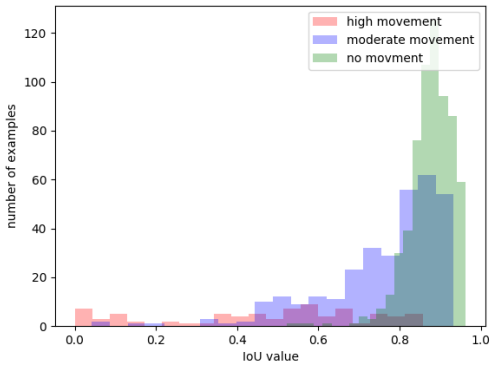
(c) Crossing dataset, 0.5s ahead predictions



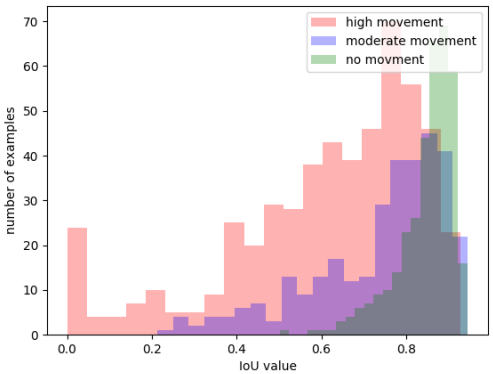
(d) Crossing dataset, 1s ahead predictions.

Figure A.4: Histograms of the number of examples vs the mask IoU between predictions and ground truths obtained by SpatialNet with mask-only inputs. The histograms are divided between high, moderate and no movement test examples from the different sets.

A.4 Experiment 3



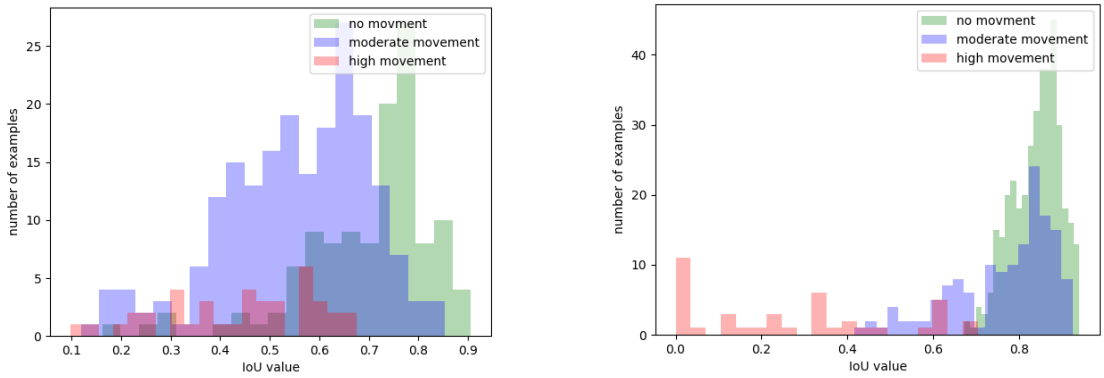
(a) 0.5s ahead prediction



(b) 1s ahead prediction

Figure A.5: Histograms of the number of examples vs the mask IoU between predictions and ground truths obtained by SpatialNet on the Large Crossing dataset. The histograms are divided between high, moderate and no movement test examples with 0.5 or 1 second ahead predictions.

A.5 Experiment 4



(a) Football dataset.

(b) Crossing dataset.

Figure A.6: Histograms of the number of examples vs the mask IoU between the predictions and ground truths obtained by SpatialNet on pre-trained Resnet feature encodings. The histograms are divided between high, moderate and no movement test examples with $0.5s$ ahead predictions.

Bibliography

[1] Cambridge in colour. digital image interpolation. http://www.cambridgeincolour.com/tutorials/image-interpolation.htm. [Accessed: 01 June 2018].

[2] pytorch/vision: Datasets, Transforms and Models specific to Computer Vision. Github, https://github.com/pytorch/vision. [Accessed: 20 June 2018].

[3] The HDF5 Group. HDF5, Documentation. http://portal.hdfgroup.org/display/HDF5/HDF5. [Accessed: 01 June 2018].

[4] W. Abdulla. Splash of Color: Instance Segmentation with Mask R-CNN and TensorFlow. Medium, https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46. [Accessed: 20 June 2018].

[5] V. Badrinarayanan, A. Handa, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*, 2015.

[6] A. Bewley. abewley/sort. Github, https://github.com/abewley/sort. [Accessed: 01 June 2018].

[7] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 3464–3468. IEEE, 2016.

[8] A. Bhattacharyya, M. Fritz, and B. Schiele. Bayesian prediction of future street scenes through importance sampling based optimization. *arXiv preprint arXiv:1806.06939*, 2018.

[9] D. Bruff. Harvard University, Math 20, Introduction to Linear Algebra and Multivariable Calculus,The Assignment Problem and The Hungarian Method. http://www.math.harvard.edu/archive/20_spring_05/handouts/assignment_overheads.pdf, 2005. [Accessed: 20 June 2018].

[10] H. Caesar, J. Uijlings, and V. Ferrari. Region-based semantic segmentation with end-to-end training. In *European Conference on Computer Vision*, pages 381–397. Springer, 2016.

[11] 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.

[12] 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.

[13] A. Collette and contributers. HDF5 for Python. http://www.h5py.org. [Accessed: 01 June 2018].

[14] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset. https://www.cityscapes-dataset.com. [Accessed: 01 June

2018].

[15] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. Cityscapes dataset examples. https://www.cityscapes-dataset.com/examples/#videos. [Accessed: 01 June 2018].

[16] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[18] A. Dosovitskiy and V. Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.

[19] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal. The importance of skip connections in biomedical image segmentation. In *Deep Learning and Data Labeling for Medical Applications*, pages 179–187. Springer, 2016.

[20] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[21] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[22] M. Firman, N. D. Campbell, L. Agapito, and G. J. Brostow. Diversenet: When one right answer is not enough. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5598–5607, 2018.

[23] M. Flagg, A. Nakazawa, Q. Zhang, S. B. Kang, Y. K. Ryu, I. Essa, and J. M. Rehg. Human video textures. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 199–206. ACM, 2009.

[24] C. V. Foundation. Cvpr 2017, open access repository. http://openaccess.thecvf.com/CVPR2017.py. [Accessed: 01 June 2018].

[25] C. V. Foundation. Cvpr 2018, open access repository. http://openaccess.thecvf.com/CVPR2018.py. [Accessed: 01 August 2018].

[26] C. V. Foundation. Iccv 2017, open access repository. http://openaccess.thecvf.com/ICCV2017.py. [Accessed: 01 June 2018].

[27] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.

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

[29] 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*, pages 580–587, 2014.

[30] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[31] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. pp. 224-270, 326-366, `http://www.deeplearningbook.org`.

[32] M. Greenacre and R. Primicerio. *Multivariate analysis of ecological data*. Fundacion BBVA, 2014.

[33] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew. A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval*, 7(2):87–93, 2018.

[34] K. He. Deep Residual Networks, Deep Learning Gets Way Deeper, 2016. [Accessed: 10 July 2018].

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

[36] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[37] 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*, pages 770–778, 2016.

[38] M. Hoai and F. De la Torre. Max-margin early event detectors. *International Journal of Computer Vision*, 107(2):191–202, 2014.

[39] D.-A. Huang and K. M. Kitani. Action-reaction: Forecasting the dynamics of human interaction. In *European Conference on Computer Vision*, pages 489–504. Springer, 2014.

[40] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017.

[41] J. Huang. jaxony/unet-pytorch: U-net implementation in pytorch: U-net implementation for pytorch based on https://arxiv.org/abs/1505.04597. Github, https://github.com/jaxony/unet-pytorch. [Accessed: 20 June 2018].

[42] C. Ilisescu, H. A. Kanaci, M. Romagnoli, N. D. Campbell, and G. J. Brostow. Responsive action-based video synthesis. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 6569–6580. ACM, 2017.

[43] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[44] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[45] X. Jin, H. Xiao, X. Shen, J. Yang, Z. Lin, Y. Chen, Z. Jie, J. Feng, and S. Yan. Predicting scene parsing and motion dynamics in the future. In *Advances in Neural Information Processing*

Systems, pages 6915–6924, 2017.

[46] J. Jordan. Evaluating image segmentation models. Medium, https://www.jeremyjordan.me/evaluating-image-segmentation-models/, 2018. [Accessed: 10 August 2018].

[47] N. Joshi, S. Mehta, S. Drucker, E. Stollnitz, H. Hoppe, M. Uyttendaele, and M. Cohen. Cliplets: juxtaposing still and dynamic imagery. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 251–260. ACM, 2012.

[48] N. Kalchbrenner, A. v. d. Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu. Video pixel networks. *arXiv preprint arXiv:1610.00527*, 2016.

[49] A. Karpathy. CS231 - Convolutional Neural Networks for Visual Recognition. Stanford University, http://cs231n.github.io/convolutional-networks/. [Accessed: 01 June 2018].

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

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

[52] S. K. Kumar. On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863*, 2017.

[53] T. Lan, T.-C. Chen, and S. Savarese. A hierarchical representation for future action prediction. In *European Conference on Computer Vision*, pages 689–704. Springer, 2014.

[54] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017.

[55] 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*, pages 3431–3440, 2015.

[56] P. Luc, C. Couprie, Y. Lecun, and J. Verbeek. Predicting future instance segmentations by forecasting convolutional features. *arXiv preprint arXiv:1803.11496*, 2018.

[57] P. Luc, N. Neverova, C. Couprie, J. Verbeek, and Y. LeCun. Predicting deeper into the future of semantic segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, 2017.

[58] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[59] Matterport. matterport/Mask˙RCNN: Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. Github, https://github.com/matterport/Mask_RCNN. [Accessed: 01 June 2018].

[60] Milesial. milesial/Pytorch-UNet: Pytorch implementation of the U-Net for image semantic segmentation, with dense CRF post-processing. Github, https://github.com/milesial/Pytorch-UNet. [Accessed: 20 June 2018].

[61] E. Million. The hadamard product. *Course Notes,*

http://buzzard.ups.edu/courses/2007spring/projects/million-paper.pdf, 3:6, 2007.

[62] W. Mischel. Processes in delay of gratification. In *Advances in experimental social psychology*, volume 7, pages 249–292. Elsevier, 1974.

[63] S. S. Nabavi, M. Rochan, and Y. Wang. Future semantic segmentation with convolutional lstm. *arXiv preprint arXiv:1807.07946*, 2018.

[64] 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)*, pages 807–814, 2010.

[65] M. A. Nielsen. Deep learning. *Determination Press*, 2015. http://neuralnetworksanddeeplearning.com, [Accessed: 01 June 2018].

[66] A. E. Orhan and X. Pitkow. Skip connections eliminate singularities. *arXiv preprint arXiv:1701.09175*, 2017.

[67] D. Parthasarathy. A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN. Medium, https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4. [Accessed: 10 August 2018].

[68] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[69] B. Peng, L. Zhang, and D. Zhang. A survey of graph theoretical approaches to image segmentation. *Pattern Recognition*, 46(3):1020–1038, 2013.

[70] L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[71] S. L. Pintea, J. C. van Gemert, and A. W. Smeulders. Déja vu: Motion prediction in static images. In *European Conference on Computer Vision*, pages 172–187. Springer, 2014.

[72] I. Preferred Networks. Experimental result of CityScape datasets (segmentation) 2016 version. Youtube, https://www.youtube.com/watch?v=1HJSMR6LW2gfrags=pl%2Cwn. [Accessed: 10 August 2018].

[73] S. J. Prince. *Computer vision: models, learning, and inference*. Cambridge University Press, 2012.

[74] Pytorch. Pytorch Documentation. https://pytorch.org/docs/stable/index.html. [Accessed: 01 June 2018].

[75] Pytorch. Pytorch Documentation, 2d convolution layer. https://pytorch.org/docs/stable/nn.html#conv2d. [Accessed: 01 June 2018].

[76] Pytorch. Pytorch Documentation, 2d transopose convolution layer. https://pytorch.org/docs/stable/nn.html#convtranspose2d. [Accessed: 01 June 2018].

[77] Pytorch. Pytorch Documentation, optimiser algorithms. https://pytorch.org/docs/stable/optim.html#algorithms. [Accessed: 01 June 2018].

[78] Pytorch. Pytorch Documentation, Pooling layers. https://pytorch.org/docs/stable/nn.html#pooling-layers. [Accessed: 01 June 2018].

[79] Pytorch. Torchvision documentation. https://pytorch.org/docs/stable/torchvision/index.html. [Accessed: 10 June 2018].

[80] M. A. Rahman and Y. Wang. Optimizing intersection-over-union in deep neural networks for image segmentation. In *International Symposium on Visual Computing*, pages 234–244. Springer, 2016.

[81] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014.

[82] 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*, pages 91–99, 2015.

[83] D. Riahi and G.-A. Bilodeau. Online multi-object tracking by detection based on generative appearance models. *Computer Vision and Image Understanding*, 152:88–102, 2016.

[84] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. In *European Conference on Computer Vision*, pages 312–329. Springer, 2016.

[85] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[86] M. S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1036–1043. IEEE, 2011.

[87] A. Schödl and I. A. Essa. Controlled animation of video sprites. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 121–127. ACM, 2002.

[88] N. Shibuya. Up-sampling with Transposed Convolution. Medium: Towards Data Science, https://towardsdatascience.com/up-sampling-with-transposed-convolution-9ae4f2df52d0. [Accessed: 10 August 2018].

[89] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[90] R. V. Singh. Imagenet winning cnn architectures–a review. github.io , http://rajatvikramsingh.github.io/media/DeepLearning˙ImageNetWinners.pdf. [Accessed: 01 June 2018].

[91] Sony. Rx0: Imaging of infinate possibility. https://www.sony.co.uk/electronics/RX0-series. [Accessed: 10 August 2018].

[92] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[93] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015.

[94] T. Suddendorf and M. C. Corballis. The evolution of foresight: What is mental time travel, and is it unique to humans? *Behavioral and brain sciences*, 30(3):299–313, 2007.

[95] R. S. Sutton, A. G. Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

[96] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[97] J. Tompkin, F. Pece, K. Subr, and J. Kautz. Towards moment imagery: Automatic cinemagraphs. In *2011 Conference for Visual Media Production*, pages 87–93. IEEE, 2011.

[98] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating the future by watching unlabeled video. *CoRR, abs/1504.08023*, 2, 2015.

[99] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision*, pages 835–851. Springer, 2016.

[100] J. Walker, A. Gupta, and M. Hebert. Dense optical flow prediction from a static image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2443–2451, 2015.

[101] J. Walker, K. Marino, A. Gupta, and M. Hebert. The pose knows: Video forecasting by generating pose futures. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 3352–3361. IEEE, 2017.

[102] X.-S. Wei. Must know tips for deep learning neural networks. KDnuggets, https://www.kdnuggets.com/2016/03/must-know-tips-deep-learning-part-1.html. Nanjing University, Accessed: 16 August 2018.

[103] P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.

[104] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *Image Processing (ICIP), 2017 IEEE International Conference on*, pages 3645–3649. IEEE, 2017.

[105] H. Yu, Z. Yang, L. Tan, Y. Wang, W. Sun, M. Sun, and Y. Tang. Methods and datasets on semantic segmentation: A review. *Neurocomputing*, 304:82–103, 2018.

[106] H. Zhu, F. Meng, J. Cai, and S. Lu. Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation. *Journal of Visual Communication and Image Representation*, 34:12–27, 2016.