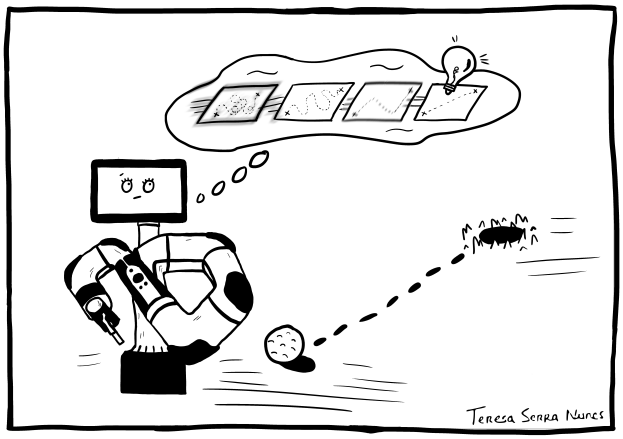
TÉCNICO LISBOA



Action-conditioned disentanglement of agent and objects for video prediction in robotic tasks

Manuel Serra Nunes

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Dr. Plinio Moreno López

Prof. José Alberto Rosado dos Santos-Victor

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Dr. Plinio Moreno López
Members of the Committee: Prof. Manuel Cabido Peres Lopes

October 2020

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First, I would like to express my gratitude to my supervisors Prof. Plinio Moreno and Prof. José Santos-Victor for the precious advice and encouragement, as well as for making me feel welcome at Vislab, where I had the opportunity to be part of a group driven by curiosity and the determination for spreading scientific knowledge.

I would also like to express my deep gratitude to Atabak for tirelessly mentoring me during the past two years, for his incredible availability - even in the last few days of his own thesis - and for his friendship. Yet, above all else, for believing in me and always setting the bar high. Without him I would have learned and achieved a small fraction of what I have.

To the friends I made along my journey at Técnico a thank you for all the unforgettable moments. A big, big thank you to my girlfriend Pipa for her companionship and love, through all the highs and lows. To my family, mom, dad, sisters, brothers and grandpa who have always supported me and are the source of my curiosity and enthusiasm for learning.

Finally, I dedicate this work to avó Celeste, for being the foundation of the person I am.

Abstract

Based on past experience, our brain is capable of anticipating the imminent future, simplifying routine tasks such as reading or driving. Yet, prediction may have an even more significant role in human faculties, as suggested by the growing support for prediction based theories of the brain that connect perception, learning and prediction. If these are correct, studying the prediction of sensory signals may turn out to be an important stepping stone on the path towards intelligent systems.

Predicting visual input, in the form of video frames, is maybe the most obvious way of materializing these ideas. The effort for designing video prediction models has intensified in the last five years, yet current solutions are still hampered by the rapid increase in difficulty that comes with the size of the frames and the prediction horizon. Two ways of mitigating these problems are to disentangle sources of information and to add the future actions of an agent as an extra input. Furthermore, an action conditioned model may allow a robot to imagine possible futures that result of executing different available action sequences and to select the most convenient course of action. Naturally, in this type of framework the ability of the model to grasp the implications of each action is of critical importance.

With this in consideration, this work can be summarized in the following contributions: (i) we propose a new method that evaluates video prediction models based on their ability to guide action decisions; (ii) we design an autoencoder model that separates agent information from information of objects, using knowledge of future actions and the inherent structure of video data; and (iii) we investigate whether separately predicting object information, conditioned on the actions, can improve the state of the art in video prediction.

Keywords

Video Prediction, Disentangled representations, Robotics, Benchmarking, Predictive Coding, Deep Learning

Resumo

Tendo por base a sua experiência, o cérebro humano é capaz de anticipar futuros iminentes e assim simplificar tarefas rotineiras como ler ou conduzir. Ainda assim, o papel desta capacidade de previsão nas faculdades humanas poderá ser de uma ainda maior prepoderância, tal como sugerido pelo crescente apoio que teorias do cérebro humano que relacionam perceção, aprendizagem e previsão têm ganho. Se, de facto, estas ideias estiverem corretas, o estudo da previsão de inputs sensoriais poderá revelar-se um importante passo no desenvolvimento de sistemas inteligentes.

Prever inputs visuais, na forma de frames de vídeo, é talvez o modo mais evidente de concretizar estas ideias. Nos últimos cinco anos, o esforço para desenvolver modelos de previsão de vídeo tem vindo a crescer mas, ainda assim, as soluções existentes continuam a ter dificuldade em ultrapassar o rápido aumento de complexidade trazido pela dimensão dos frames e pelo horizonte de previsão. Opções para mitigar este problema incluem a dissociação de fontes de informação e o uso das futuras ações de um agente como um input extra. Uma outra vantagem de um modelo condicionado pelas ações é o facto de possibilitar que um robot imagine os possíveis futuros que resultam da execução de diferentes sequências de ação. É de notar, no entanto, que neste tipo de aplicação a escolha de um modelo que interprete corretamente as consequências de cada ação ganha especial importância.

Tendo isto em consideração, este trabalho pode ser sumarizado pelas seguintes contribuições: (i) a proposta de um novo método que avalia modelos de previsão de vídeo com base na sua capacidade para guiar tomadas de decisão; (ii) o design de um modelo autoencoder que separa informação relativa ao agente de informação sobre objetos, baseado no conhecimento das futuras ações do agente e na estrutura inerente a um vídeo; (iii) a investigação se a previsão em separado da informação de objetos, condicionada às ações, permite melhorar o estado da arte em previsão de vídeo.

Palavras Chave

Previsão de Vídeo, Representações desassociadas, Robótica, Benchmarking, Codificação Preditiva, Aprendizagem Profunda

Contents

List of Figures

List of Tables

Abbreviations

PP	Predictive Processing
NN	Neural Network
CNN	Convolutional Neural Network
VAE	Variational Auto Encoder
GAN	Generative Adversarial Network
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
ReLU	rectified linear unit
PCA	Principal Component Analysis
ICA	Independent Component Analysis
GB	Guided Backpropagation
GA	Gradient Ascent
KL	Kullback-Leibler
MAE	Mean Absolute Error
PSNR	Peak Signal to Noise Ratio
SSIM	Structural Similarity
FVD	Fréchet Video Distance
QoE	Quality of Experience
MOS	Mean Opinion Score
LPIPS	Learned Perceptual Image Patch Similarity
VP	Video Prediction
SV2P	Stochastic Variational Video Prediction
DNA	Dynamic Neural Advection
CDNA	Convolutional Dynamic Neural Advection
SAVP	Stochastic Adversarial Video Prediction
SVG-LP	Stochastic Video Generation Learned Prior
MCnet	Motion-Content Network
DRNET	Disentangled-Representation Net
ADR	Action-conditioned Disentangled Representations
ADR-AO	Action-conditioned Disentangled Representations - Agent Only
BAIR	Berkeley Artificial Intelligence Research Lab
MPC	Model Predictive Control

1

Introduction

1.1 Motivation

For the last decade, research in robotics and computer vision has been shaped by the availability of ever growing datasets which, allied with increasingly powerful computational resources, have allowed the proliferation of data driven algorithms. These have seen applications in an array of visual tasks that include image classification [1], captioning [2], object detection [3] and segmentation [4] and also image colorization [5] and synthesis [6].

A characteristic shared between all the mentioned tasks is the fact that they operate over static images. In recent years, however, video data has steadily gained relevance to the point where 82 years worth of new content were uploaded to YouTube every day in 2019 [7] and estimations indicate that by 2022 video will account for 82% of internet traffic, up from 75% in 2017 [8]. But the incentive for investing in research for video applications is not limited to the amount of available data. Video is arguably the best data to describe and capture the environment of which an agent is a part of, as demonstrated by human physiology, where vision is the most significant sensory input for our behavior and understanding of the world. A video of a dynamic scene, captured by a static camera, has information, at each frame, of the spatial disposition of the objects and entities therein, as well as their appearance. When unfolded in time, for several steps, these frames carry information about the dynamics of the environment: how objects transform in space, how they influence each other's movements and even how they can change shape (for example a human running). This is invaluable information otherwise very difficult to encode in a rich and comprehensive way. But what makes video data interesting is also what makes it complex and difficult to process. All this information comes at a cost, presenting itself in the form of high dimensionality and complex interactions between pixels.

The specific task of video prediction, however, has the characteristic of being a self-supervised problem (see section 1.2), meaning that a supervisory signal can be obtained from the input sequence itself, by holding out the frames that are supposed to be predicted. This way, and as further explored

in this dissertation, an agent interacting with the environment and observing the outcomes can in theory acquire an unlimited amount of data, and use the free supervision it provides to continuously improve a video prediction model.

The importance of such a prediction task may seem relatively shallow, at first. Applications of video prediction include weather nowcasting, and future semantic segmentation. But the ability to predict video carries a much deeper significance: an animal that is capable of predicting the imminent future as it chases prey, a baby that cries expecting his parents to show up or an autonomous system that turns the wheel of a speeding vehicle just enough to keep it within anticipated bounds must have previously learned a model of the inner workings of their world, which relates current state and action to expected future state. Hence, the task of learning to predict video overlaps with the process of learning about the world, a notion that is the main motivation behind this thesis and that is explored throughout it.

1.2 Problem Definition and Objectives

The problem of video prediction is fairly simple to state: we intend to design a model that, given some recently observed context frames $[\boldsymbol{x}^{t-c} : \boldsymbol{x}^{t}]$, is capable of generating N future frames $[\boldsymbol{x}^{t+1} : \boldsymbol{x}^{t+N}]$. Yet, predicting video frames is not an easy task as by nature it is a very high dimensional problem. Not only is each pixel in the input a dimension that the video prediction model must interpret, but each pixel of the output is also a dimension in which the model must make a decision. Furthermore, real world environments are characterized by rich, complex and uncertain dynamics which are very difficult to model.

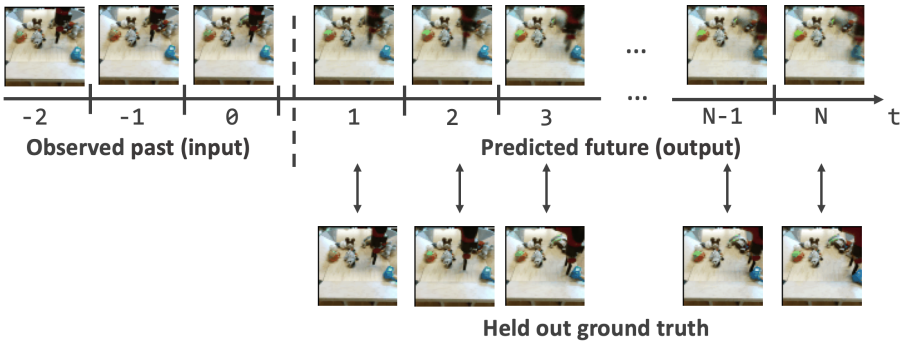


Figure 1.1: Video prediction is a self-supervised problem in which the first few frames of a video can be used as input while subsequent frames are held out to be used as ground truth in the computation of the loss term.

Not everything is bad though - because video prediction is a sequential problem, it is naturally formulated in a self-supervised manner. This means that for each video sequence in a dataset, a few frames at the beginning can be used as input to the model, while the rest of the sequence is held out and can serve as a free supervisory signal to be compared with the model's predictions (see Fig. 1.1). This is an invaluable property of this problem, as it avoids the need for expensive labeled datasets and enables an agent to actually learn from its own experience, possibly even in an online fashion

while it interacts with the environment.

Additionally, in this work we focus on the specific problem within video prediction in which there is access to the actions an agent will perform. When this is the case, we refer to the problem as action-conditioned video prediction. This sub-problem is particularly interesting in the field of robotics since, as detailed in chapter 3, it allows the imagination of different possible futures given different candidate action sequences, a strategy that can be used in planning tasks. This however, raises the question of how to evaluate video prediction models and select the best one to be used in robotic planning scenarios. In the literature, the answer to this question has mostly been based on metrics designed to approximate human perception of quality. In our first research question, however, we ask how to do it in an action oriented way, from the standpoint of a robot. This is discussed is chapter 3, along with our solution to the problem, which is a new evaluation metric based on the ability of an inference network to recognize the agent's actions from the predicted frames.

A second question raised by the availability of the actions is whether these, along with the sequential structure of the video, can be used to disentangle (separate) the sources of visual information related with the agent from those related with external objects present in the scene. The importance of this question is described in chapter 4. Our solution involves the proposition of a new autoencoder model - ADR-AO - that successfully solves the problem. Finally, we intend to explore how having disentangled representations of the agent's actions and of the objects in the scene can be beneficial for a video prediction model. These research questions are summarized in Fig. 1.2.

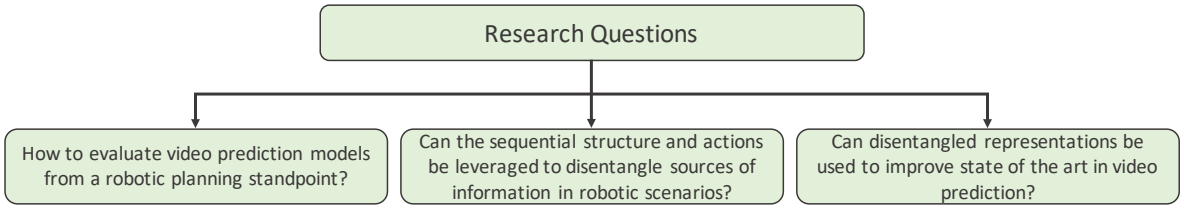


Figure 1.2: Proposed research questions.

1.3 Document Overview

The present document is composed of five chapters. Chapter 2 starts by introducing both the concept of representation learning and the deep learning approaches used throughout this work, with emphasis on the close connection between the two. The main approaches to video prediction are also presented in this chapter. Chapter 3 analyses the impact of knowing the commanded actions on the predictions and proposes a new metric for evaluating video prediction models from a robotic standpoint. In chapter 4 the concept of disentangled representations is introduced, along with our proposed model for video prediction. Finally, the results are presented in chapter 5 and the final remarks on the research questions and conclusions in chapter 6. This thesis is organized such that each new chapter builds on top of the concepts discussed in the previous ones. The common thread

running through this work is an evolving perspective on the relationship between learning, perception and prediction, which constitutes the basis of our argument for the importance of research in video prediction.

2

From human behaviour to video prediction

This chapter starts by introducing the concept of representation learning and its importance for both humans and machines. This notion is also reflected in the deep learning models presented next, which are of crucial importance in the field of video prediction. Finally, before a detailed overview of the state of the art solutions for video prediction, a parallelism is made between this problem and prediction based theories of the human brain, motivating the discussion for the next chapters.

2.1 Learning Representations

A crucial step in problem solving - as we are taught from a young age - is to find a point of view that offers a new, simpler angle on how to approach the task at hand. For example, dividing 210 by 6 is a trivial problem using long division, however, it becomes incredibly more difficult if instead the question is presented as dividing the Roman numerals CCX by VI [9]. Most people would start by finding a better representation for the information and convert the numbers from Roman to Arabic notation, allowing the use of long division. In general, a good representation for available information is one that makes subsequent tasks easier [9].

It turns out that such a notion may be rooted not only in the way we are encouraged to think by our professors but also in human nature and, in particular, in the way we convert incoming sensory signals - such as light or sound - into worldly concepts that are perceived and understood.

Starting in the second half of the twentieth century, the growing maturity of microelectrode technology enabled neurobiologists to track the activity of single neurons, bringing about new insights into how the brain processes sensory information. In particular, unexpected responses resulting from stimulating a neuron's receptive field, led to the discovery that receptive fields are organized in center and surround regions (Fig. 2.1). Within such organization, two types of behavior are found: *on-center* cells fire when their receptive field is stimulated in the center region and are inhibited when the stimulus is located in the surround, while *off-center* cells display the opposite activity [10]. Such indication that neurons in

the visual system respond optimally to transitions or boundaries in light led to the concept of a feature detector visual system, demonstrated for the first time in frogs, thanks to their capacity to detect flies for prey, which is enabled by on-center and off-center neurons.

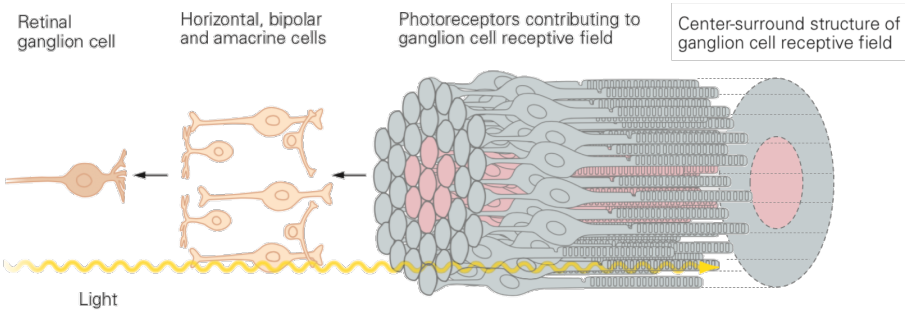


Figure 2.1: Center-surround receptive field of a ganglion cell. Source: [10].

Later studies by Hubel and Wiesel led to the observation of orientation selective neurons in cats (see Fig. 2.2) and to the subsequent discovery of complex cells, whose response varies not only with a line's orientation but also with the temporal movement of the stimulus [11]. A third type of cell, termed hypercomplex was later found to additionally be sensitive to the length of the line of stimulus, responding optimally to shorter lines - a property that could in principle allow them to serve as *corner detectors* [12].

These discoveries gave traction to the feature detector visual system hypothesis and lead Hubel and Wiesel to support the idea that processing in the visual system has an hierarchical nature: early in the visual pathway, neurons respond to simple patterns of stimulus to their receptive field and, as information flows to subsequent levels, neuron activations become more and more selective, eventually responding only to complex objects such as human faces [10]. These ideas were also influential in later work by Marr [13].

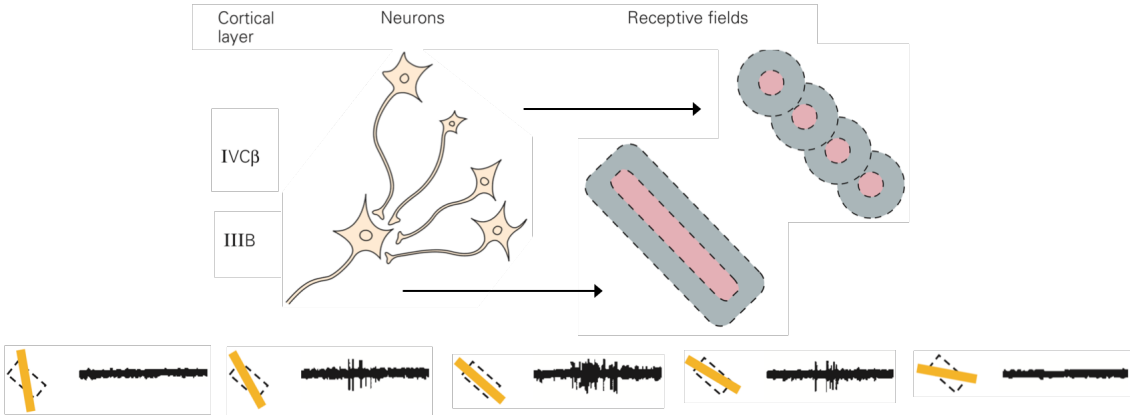


Figure 2.2: An example of an edge detector neuron in the $\mathrm{III}B$ cortical layer. This is caused by the four, diagonally aligned on-center neurons that constitute its receptive field. In the bottom are example firing rates (in black) of the neuron when a stimulus with different orientations (in yellow) is received by the receptive field. Source: [10].

The question that ensues such an hypothesis is how far the hierarchy goes? The existence of the so called *grandmother cells* - neurons that are only activated by the presence of our grandmother -

seems impractical, as it implies there being a cell to represent any given face and even different angles of the same face. Instead, it is now believed that not every single object or concept is represented by a specific neuron. Rather, representations in the brain are most likely *distributed codes* [10], meaning that each object is associated with a pattern of activations. Once again, the importance of finding the best representation to solve a task becomes apparent: all the visual information necessary to recognize an observed object is present right away at the retina, however the task can only be solved once it is made easier, by decomposing information into simple features which can then be aggregated into a concept or perception by identifying the pattern of activations.

Adding to the importance of the notion of good representations is the fact that, just as for humans, the problem solving ability of computers is heavily affected by how information is represented. For instance, the 8-queens problem - place eight queens on a chess board such that no queen can attack another - can be solved by a search algorithm, given some formulation of states and operators [14]. The most straightforward solution would be to define that a state is any arrangement of 0 to 8 queens on the board and that adding a queen to any empty square is the operator. This would result in $64 \times 63 \times 62 \times \cdots \times 57 = 1.4 \times 10^{14}$ possible states. An alternative solution would be to prevent the choice of squares which are already attacked and to determine that queens should be placed one per column, from left to right. Such a formulation reduces the search space to 2057 possible states.

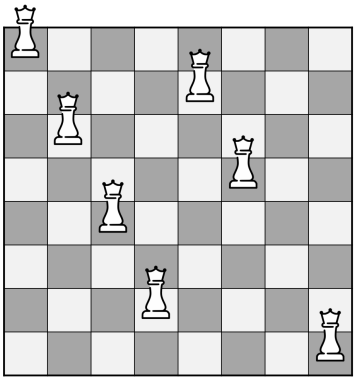


Figure 2.3: A near solution to the 8-queens problem: place eight queens on a chess board such that no queen can attack another.

Formulation 1:

- **States**: any arrangement of 0 to 8 queens on the board.
- **Operators**: placing a queen in any empty square.

Formulation 2:

- **States**: All possible arrangements of 0 to 8 queens, one per column, with no queen attacking another.
- **Operators**: Add a queen to any square in the leftmost empty column such that it is not attacked by any other queen.

Lastly, in the intersection between human inspired systems and computational science, the concept that finding a good representation for a problem should make subsequent tasks easier has been influential in research on supervised and unsupervised deep learning algorithms, such as neural networks and autoencoders. These are further presented in section 2.2 and constitute the main tool for solving the problems proposed in this work which, as will be clarified throughout this thesis, revolve themselves around the challenge of finding the best representation possible for video observations.

2.2 Neural Networks

Even though only recently have the computational means and large available datasets made neural networks a widespread framework, the concepts in which they are based upon have long been around. Gradient descent for solving optimization problems was first used by Cauchy in 1847 [15] as a way of solving the algebraic equations that describe the orbit of astronomical bodies. Several years later, in the 1940s the first perceptron models started to be used as linear function approximators and later, during the 1960s, 1970s and 1980s, the development of backpropagation with independent contributions from researchers such as, Linnainmaa [16], LeCun [17], Parker [18] and Rumelhart [19] allowed the extension of these models to multilayer neural networks, which eventually became a popular machine learning technique for learning non-linear functions. Since then, the concepts behind neural networks have essentially remained the same, with some innovations such as the use of rectified linear units as activation functions, as well as new regularization methods including Dropout and Batch Normalization that have pushed the limits of what can be achieved with this tool [9].

In the next section, the most fundamental form of a neural network is briefly introduced, followed by some advances it inspired, in particular Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and autoencoders, which nowadays are of central importance in the field of video prediction.

2.2.1 Feedforward Neural Networks

In its essence, a neural network is a layered composition of simple functions arranged in structures called neurons, that maps an input data sample to the answer being sought. Such *input-to-target* mapping is learned by optimizing the error of the network's output with gradient descent, on examples coming from a training dataset in which both input and target are known. The key for a neural network to be of practical relevance, however, is that it should be able to *generalize*, that is, achieve good performance not only on the data examples observed during training, but also on previously unseen data, coming from a test dataset.

A simple fully connected network is exemplified in Fig. 2.4(a), where an input in $\mathbb{R}^3$, is mapped to an output in $\mathbb{R}^2$, using two hidden layers, or feature maps, which are vectors in $\mathbb{R}^4$. Each unit of these two vectors is called a neuron, illustrated in detail in Fig. 2.4(b), and its value is obtained by first applying a linear transformation to its inputs

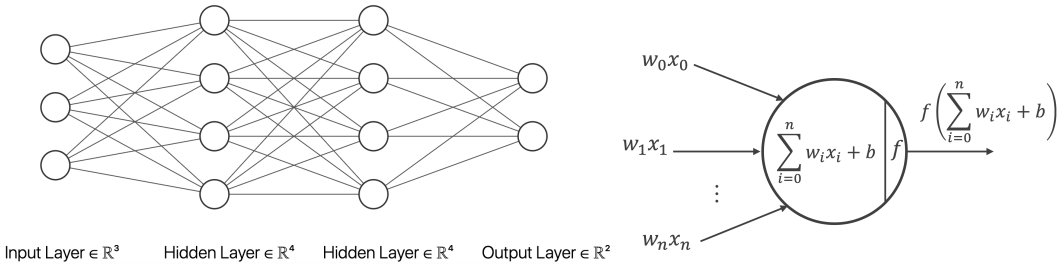
$$\sum_{i=0}^{n} w_i x_i + b, \tag{2.1}$$

where each weight value w_i is a parameter that is adjusted during training, allowing the network to give different credit to each of the feature map values x_i, coming from the previous layer. Were a neuron's output value simply determined by a linear function, the hypothesis space $\mathcal{H}$ of neural networks - *i.e.*, the space of input-output mappings that are searched during optimization - would be restricted to linear functions. However, the use of a non-linear activation function f at the output of

each neuron

$$f\left(\sum_{i=0}^{n} w_i x_i + b\right), \tag{2.2}$$

allows the hypothesis space to be extended to non-linear transformations, a key property in neural networks. Typical choices for activation functions include the sigmoid function, hyperbolic tangent and the rectified linear unit (ReLU). For the last hidden layer of a network the activation is usually a softmax or a linear function, respectively used in problems such as classification and regression.



(a) Neural Network **(b)** Neuron

Figure 2.4: A three layer, fully connected neural network and its base neuron unit applying a linear transformation to the different inputs followed by an activation function.

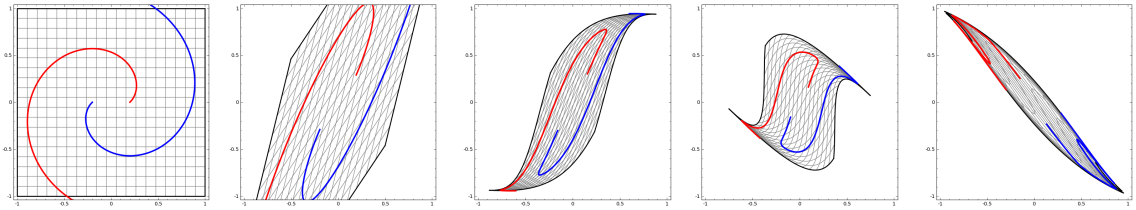
Also controlling $\mathcal{H}$ is the size of the network, which is one of the main contributing factors to an important balance in neural network design: *underfitting vs. overfitting*. Deeper networks, with more neurons have a greater representational capacity, however, due to optimization bottlenecks or the size of the available datasets, a network with too much capacity tends to perform well on the training data and yet produce a big generalization error, a problem known as *overfitting*. On the other hand, a model with low capacity is likely to *underfit*, *i.e.*, it won't fit either the train or test datasets. Several methods have been devised in the field of machine learning in an attempt to enable optimal balance between these two problems. These are typically focused in preventing overfitting, thus assuming that the network is given enough representational power to avoid underfitting.

One of the most common methods to prevent overfitting is adjusting model capacity using regularization. Regularization methods induce a preference for solutions that are simpler, as long as the error with such solution is not significantly worse than other solutions. Simpler models are typically the ones that assign less overall credit to the features coming from the previous layer, as that results in lower model capacity. For this reason, a straightforward way of signalling the preference for simpler models is to add an extra term to the loss function, penalizing the squared norm of the weight matrices $\mathbf{W}$. Alternative regularization methods include the Dropout algorithm [20], in which neurons and their connections are randomly dropped during training, or the use of Early Stopping, which causes the training loop to break once the train loss and validation loss start to diverge.

Given this brief and somewhat taxing overview, it is perhaps more important in the scope of this work to seek an intuitive understanding of the inner workings of neural networks, in an attempt to make less use of them as a black box solution, and more as a tool that displays some logical behaviour.

In particular, it is important for the purpose of this work to understand neural networks from the standpoint of representation learning.

Yoshua Bengio, one of the fathers of deep learning, defines neural networks as "a particular kind of representation learning procedure that discovers multiple levels of representation, with higher-level features representing more abstract aspects of the data" [21], meaning that each layer of a neural network is working to find a description of the data it receives that makes it easier for the next layer to succeed in its task. However, all layers in the network with exception of the last one are assigned with this same objective, resulting in descriptions of the data becoming more and more abstract in an attempt to make it easier for the last layer to solve its task, which may be something such as linearly classifying the original data.



(a) (b) (c) (d) (e)

Figure 2.5: A classification problem in which a simple network is asked to classify a given point as one of two entangled classes. At each layer - (a) through (e) - the network learns increasingly more abstract representations that allow a linear classifier at the output layer to correctly separate the data. Adapted from Christian Olah's blog [22].

To understand why such an effort may be necessary to solve a classification task, it is helpful to look at a low dimensional example such as linearly separating space in two classes, based on a set of known data points [22]. From Fig. 2.5(a) it is easy to conclude that from the initial representation of the data, the output layer would fail to correctly classify every point in space as “red” or “blue”. However, because each layer has the ability to scale, translate and deform the input according the its activation function - in this example an hyperbolic tangent - a new representation is learned at each level (Figures 2.5(a) to 2.5(e)), gradually making it easier for the classifier to perform well. By the final hidden layer (Fig. 2.5(e)), the original data is hardly recognizable but it has become trivial to linearly separate the space in two classes.

Of course most problems of interest such as speech recognition or image processing are of much higher complexity and live in high dimensional spaces which we cannot visualize. Yet, the same principle still holds: a deep neural network learns multiple levels of representation that reflect the semantic content of the data, dissociated from low-level features like waveforms or pixels [21], and that allow the final layer of the network to succeed in its task.

In recent years, the increasing appeal of neural networks has lead to new benchmarks being set in

different problems, as well as to new solutions and products in fields as diverse as finance, medicine, automotive and translation. Even though such developments are of incredible value, neural network based algorithms are still far from constituting a definitive solution as challenges such as the sophistication of optimization methods, the scalability of computations and the explainability of the algorithms continue to represent bottlenecks on what can be achieved. A difficulty that is particularly onerous in image processing and in video prediction is the very high dimensionality of both the input data the model must process and the output sequence it must generate, which can easily reach the order of the hundreds of thousands of pixels. Driven by this problem, a significant research effort has been made on finding data efficient methods that can leverage the structure of the data to reduce the dimensionality burden. CNNs and RNNs, described in sections 2.2.2, and 2.2.3 are of especial interest in video prediction as they are optimized to deal with image data and sequential data, respectively.

2.2.2 Convolutional Neural Networks

Convolutional neural networks are a kind of neural network designed to solve problems in which the data is structured in a grid, making them a prime candidate model for image and video frame processing. The name convolutional stems from the fact that these networks use the convolution of the layer input with the weights (organized in a filter structure also called kernel), instead of the matrix multiplication used in fully connected networks (see Eq. 2.2).

This modification is not only motivated by the improvement in data efficiency, but also by the notion that most meaningful features in an image - such as edges and corners - are sparse interactions between pixels in the same local region, a fact that supports an architecture that disregards the relationship between pixels of the image that are distant from each other in favor of pixel dependencies in a short neighborhood around each pixel [23].

This type of behaviour can be achieved with a 2D convolution operation ((2.3)), which corresponds to sliding a flipped filter w of size k over every pixel (u, v) of the input image I, taking a dot product at every step and producing a feature map S, as exemplified in Fig. 2.6. Because the filter cannot be slid over the input's edges, the operation produces an output of smaller size than the input, which depends on the image size N, filter size k and on the stride s with which the sliding window advances. Optionally, the user can add zero padding of size p to the input, resulting in an output size given by equation (2.4). Instead of learning a matrix W of weights modelling the interaction between every possible pair of pixels (see (2.1)), the network is now learning the weights of the filters (Fig. 2.6), with these modelling the local interactions between pixels.

Each layer is typically convolved with several filters, producing the same number of stacked feature maps, which are then passed as input to the next layer. The reason for using multiple filters is that each one is looking for a specific feature, for example, a layer looking for edges could have a filter approximating the horizontal derivative to find vertical edges and another filter approximating the vertical derivative to find horizontal edges. Just like for fully connected Neural Networks (NNs), each

$$S(u,v) = \sum_{m\in(-k,k)} \sum_{n\in(-k,k)} I(m,n)w(u-m,v-n) \quad (2.3)$$

$$Output\ size = \frac{N-k+2p}{s} + 1 \quad (2.4)$$

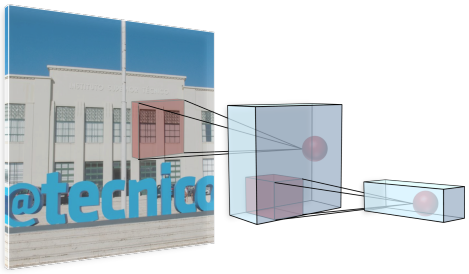


Figure 2.6: Two dimensional convolution operation over an image. The filter (in red) is convolved over all spatial locations, producing an output feature map (in blue).

convolutional layer is followed by a non linear activation function, whose output is maximum when the kernel finds a part of the image that corresponds to the feature it is looking for (e.g. vertical edge).

In trying to solve a problem at the last layer, the convolutional filters of the network learn to extract the most relevant features of the input image, hinting that visualizing the weights may provide insight on the representation learning capabilities of the network. Doing so, reveals that learnt filters have an hierarchical relationship with the depth of the network: at upper levels they are activated by (*i.e.* are looking for) generic, low level features of the image, such as edges, while at deeper levels it is the more abstract aspects of the image that cause the filters to be activated [24].

This property often leads filters at upper layers to converge to Gabor functions of variable center, orientation, Gaussian scale and spatial frequency of brightness [9] (Fig. 2.7(a)), which allows each filter to respond to specific brightness patterns in an image. In the case of convolutional filters at deeper levels of the network, which are looking for abstract features, simply visualizing the weights provides no insight on what the filter is looking for. Instead, we can use techniques such as Guided Backpropagation (GB) [24] to find out the pixels in the input image to which a neuron responds the most or Gradient Ascent (GA) [25] to generate a synthetic image that activates a given neuron the strongest. Example visualizations obtained using GB and GA can be seen in Fig. 2.7(b), which demonstrate that, at deeper levels, the receptive field of the CNNs neurons encompasses high level entities such as frogs or cats, and not just any edge or corner.

In the case of CNNs, motivation for this behaviour and hierarchical structure goes even further and garners parallelism with the anatomy of the visual system in animals, described in section 2.1, which may explain their remarkable image processing capabilities. A loose parallelism between CNNs and receptive fields in the visual system can be made, in which the convolutional filters capturing local dependencies between pixels work as the receptive field of a neuron, producing a response that is passed to a higher level layer of the network. Illustrations of the CNN and eye structure in Figures 2.6 and 2.1 respectively convey this parallelism, with the receptive field in the periphery of the retina resembling the red convolutional filters and with the summarization of information being present in both the signal transmission to the ganglion cells and the hierarchical structure of CNNs.

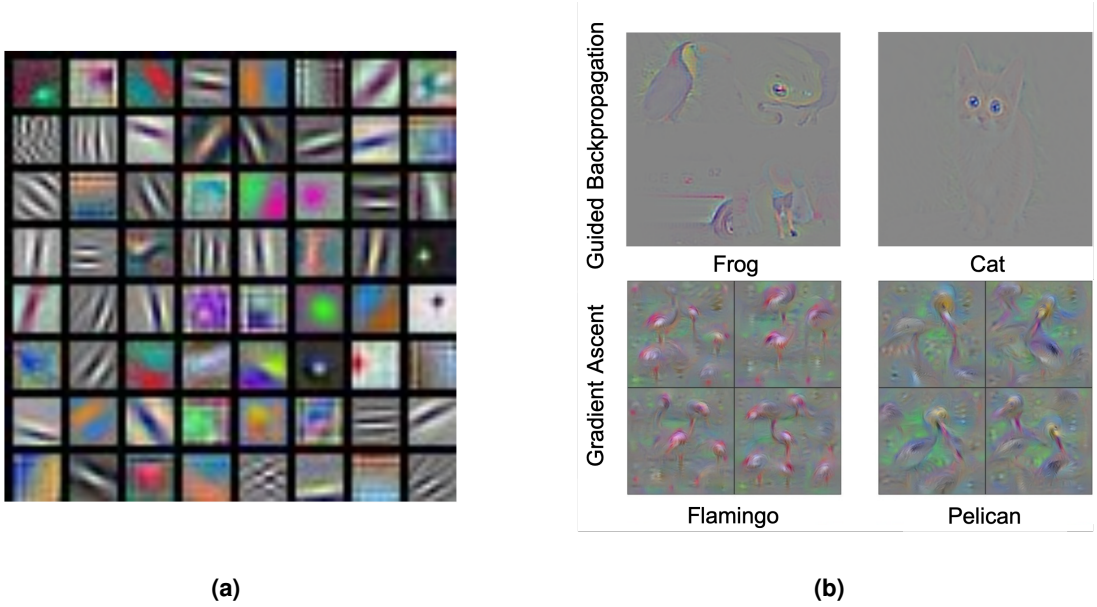


Figure 2.7: Visualization of representations learned by CNNs for layers at different depths. (a) Filters learned at the first layer of AlexNet, resembling different variations of Gabor functions. Source [23]. (b) Top: visualization of the image patches causing stronger activations at the 12^{th} layer of a CNN [24], obtained using GB. Bottom: inputs preferred by layer $fc8$ of AlexNet, for the flamingo and pelican classes of ImageNet [25], obtained using GA.

Despite having become the standard method for processing image data in a variety of subjects, the fact that the output of a CNN is solely based on the most recent input and not on the history of observed inputs constitutes a limitation in problems like video prediction, in which the information contained in the sequence of frames is crucial for a good performance. This is an intuitive fact when thinking of how humans read a piece of text, translate a spoken sentence or watch a video. Our knowledge of what has happened does not start over after each word or each frame, instead there is some kind of persistence that allows us to have a deep understanding of the sequence of events [26], a quality that is introduced in Recurrent Neural Networks.

2.2.3 Recurrent Neural Networks

A possibility for solving the problem of dealing with sequential information in a naive way would be to simply concatenate the past n elements of a sequence - a video or written phrase - and pass it as a single input to the network, in order to predict the next element in the sequence. Such an approach, however, would not only result in a significant increase in the dimensionality of the problem but also require the definition of the time scale of past observations needed to solve it. For example, we can imagine an hypothetical model trained to identify the murderer in Agatha Christie's Poirot television episodes. This model would need the ability to keep track of the relevant scattered clues, which may be revealed at any given point in time. Even setting a high number of past frames for input would never guarantee success, suggesting that a better solution is for the model to have selective memory of past events.

Recurrent neural networks (RNNs) exploit sequential information by having a recurrent cell (Fig. 2.8(a)) that continually receives as input new elements $\boldsymbol{x}_t$ of a sequence. Within this cell, there is an internal feedback loop - the hidden state $\boldsymbol{h}$ - which works as a memory, allowing past information to persist. An intuitive way of thinking of this recurrent behaviour is as a series of simple neural networks taking different inputs and passing information about what they have seen to the next element (Fig. 2.8(b).

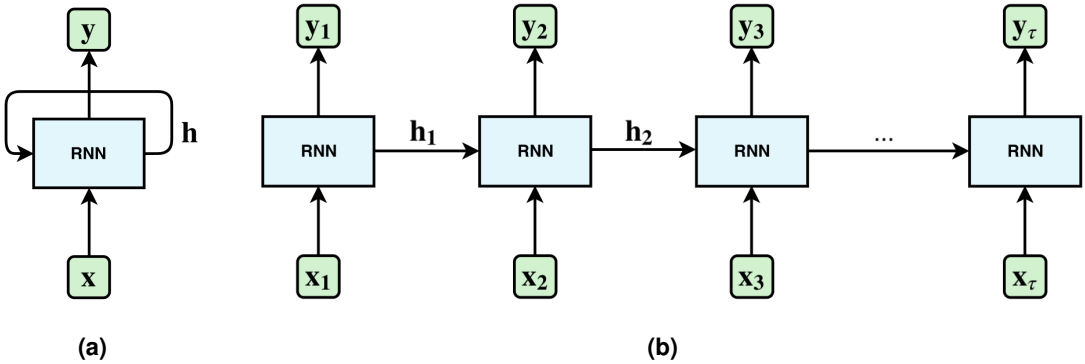


Figure 2.8: High level architecture of a recurrent neural network: (a) recurrence of the RNN cell; (b) unfolded visualization of the recurrence.

The recurrence relation is described by (2.5), where f_W is a function with parameters W which are the same for every time-step

$$\boldsymbol{h}_t = f_W(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t). \tag{2.5}$$

For a standard RNN cell (presented in Fig. 2.9(a)) f_W is

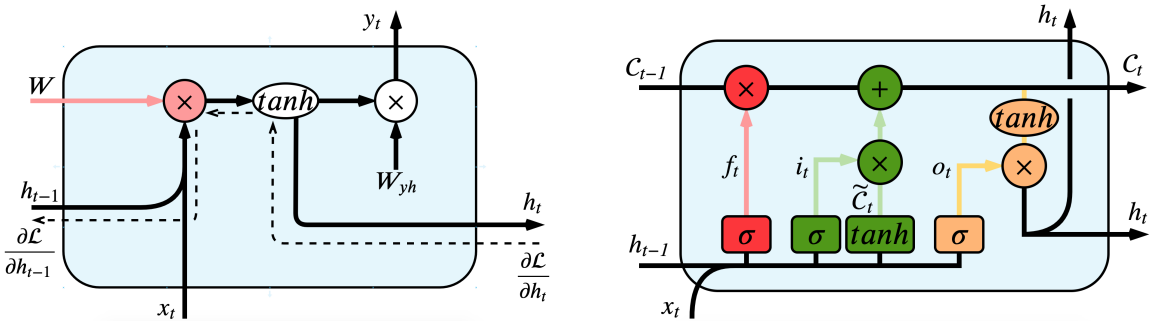
$$\boldsymbol{h}_t = \tanh\left(\begin{pmatrix} W_{hh} & W_{xh} \end{pmatrix}\begin{pmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{x}_t \end{pmatrix}\right) = \tanh\left(W\begin{pmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{x}_t \end{pmatrix}\right) \tag{2.6}$$

and the output is given by

$$\boldsymbol{y}_t = W_{yh}\boldsymbol{h}_t. \tag{2.7}$$

At training time, the loss is computed between each output $\boldsymbol{y}_t$ and the corresponding target $\boldsymbol{t}_t$, with targets for sequential problems often being very cheap to obtain, as they can be held out from the input sequence itself.

Having obtained the loss term, we're interested in computing the derivative of the loss with respect to the weights $\frac{\partial \mathcal{L}}{\partial W}$, so that these can be updated. Using backpropagation, each RNN cell in Fig. 2.8(b) can be seen as a gate in a computational graph, which receives $\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t}$ and computes $\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_{t-1}}$, as shown in Fig. 2.9(a). There is however a problem with the backward flow of gradient in these cells: in a multiplication gate, the gradient is scaled by the factor of the other input of the gate, which in this case is W^T (red in Fig. 2.9(a)). This means that at every time-step of a sequence, the gradient is multiplied by the same factor. Gradients multiplied by factors smaller than 1 will converge to zero as they approach earlier cells, preventing significant weight updates and compromising the ability of the model to learn long term dependencies in the data.



(a) **(b)**

Figure 2.9: (a) Standard RNN cell defined in equation (2.6). The input and hidden state are stacked and multiplied by a weight matrix W, followed by a hyperbolic tangent operation, giving the next hidden state $\boldsymbol{h}_t$. To obtain the output, $\boldsymbol{h}_t$ is multiplied by a second weight matrix W_{yh}. (b) LSTM cell with forget gate in red, input gate in green and output gate in yellow controlling the flow of information in the cell state. Adapted from [26].

With the intention of preventing this problem, Hochreiter and Schmidhuber proposed the Long Short Term Memory (LSTM) in 1997 [27]. LSTM cells make use of 4 internal neural network layers to control the stream of information in what are now two memory vector states: the cell state $\boldsymbol{c}_t$ and hidden state $\boldsymbol{h}_t$. Each of these 4 layers is responsible for a different task in managing the states and are of the form of equation (2.2), meaning each one has an associated activation function, bias and weight matrix (2.8), shared across all time steps.

$$\begin{pmatrix} \boldsymbol{f}_t \\ \boldsymbol{i}_t \\ \tilde{\boldsymbol{C}}_t \\ \boldsymbol{o}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ tanh \\ \sigma \end{pmatrix} W \begin{pmatrix} \boldsymbol{h}_{t-1} \\ \boldsymbol{c}_t \end{pmatrix} = \begin{cases} \boldsymbol{f}_t = \sigma\big(W_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f\big) \\ \boldsymbol{i}_t = \sigma\big(W_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i\big) \\ \tilde{\boldsymbol{c}}_t = tanh\big(W_C \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_C\big) \\ \boldsymbol{o}_t = \sigma\big(W_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o\big) \end{cases} \tag{2.8}$$

Three of the layers - $\boldsymbol{f}_t$, $\boldsymbol{i}_t$ and $\boldsymbol{o}_t$ - have their outputs squashed to the $[0:1]$ range by a sigmoid function, which makes them work as gatekeepers for information. An element of $\boldsymbol{c}_t$ or $\boldsymbol{h}_t$ that is multiplied by a value close to 0 will be removed from the memory whereas an element multiplied by a value close to 1 is almost entirely preserved. With this property in mind, the role of each layer in the information flow can be understood by how it interacts with the memory states.

The first layer, $\boldsymbol{f}_t$, is traditionally designated as *forget gate*. The outputs of $\boldsymbol{f}_t$ are directly multiplied elementwise with $\boldsymbol{c}_{t-1}$ (red blocks in Fig. 2.9(b)), determining which values of $\boldsymbol{c}_{t-1}$ will be forgotten (multiplication by 0) and which will be kept (multiplication by 1). Next, to decide what information of the current observations $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$ is relevant to be added to $\boldsymbol{c}_t$, candidate values $\tilde{\boldsymbol{c}}_t$ in the $[-1:1]$ range are obtained and filtered with $\boldsymbol{i}_t$, the *input gate*. The resulting vector is directly added to the cell state, forcing these two layers to learn what to add to memory (green blocks in Fig. 2.9(b)). At this point, the new cell state $\boldsymbol{c}_t$ has been obtained, according to 2.9

$$\boldsymbol{c}_t = \boldsymbol{f}_t * \boldsymbol{c}_{t-1} + \boldsymbol{i}_t * \tilde{\boldsymbol{c}}_t. \tag{2.9}$$

Finally, an hyperbolic tangent function brings the values of the cell state to the $[-1:1]$ range, which

are then selected by the *output gate*, $\boldsymbol{o}_t$, to form the new hidden state (yellow blocks in Fig. 2.9(b)), according to 2.10. Both $\boldsymbol{c}_t$ and $\boldsymbol{h}_t$ are then passed as feedback to the next temporal step.

$$\boldsymbol{h}_t = \boldsymbol{o}_t * tanh(\boldsymbol{c}_t). \tag{2.10}$$

This architecture represents a significant increase in complexity when compared to the simple RNN, but more than for how it controls the stream of information in the memory state, this structure is attractive for how it allows the gradient to be transmitted during backpropagation without causing vanishing or exploding gradients.

Even though some problems such as video prediction and translation are inherently sequential, some other tasks, that don't seem to have temporal correlation at first sight, may also benefit from a sequential representation and the features learned by an RNN. In [28], Ha and Eck take the problem of image generation - which would tipycaly be modeled as a group of pixels - and use an RNN based model named Sketch-RNN to instead learn the task as a sequence of short strokes of a pencil. The model is trained on a dataset of hand-drawn sketches in which the strokes are represented by the directions in which to move, when to lift up the pencil and when to stop. Figure 2.10 shows how from an initial line drawn by a user, the model can generate different versions of a given object, one line at a time, implying that the concept of such objects must be represented in the memory of the recurrent cells of the model. Adding to the effectiveness of the recurrent units of Sketch-RNN, is the fact that it uses a special type of architecture named autoencoder, which is designed to learn the characteristics of the data that should be included in the representation.

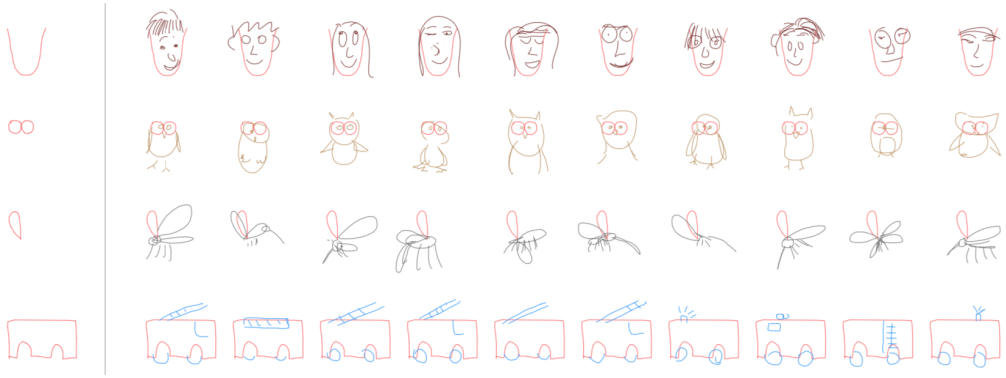


Figure 2.10: Incomplete sketches (red lines) finished by Sketch-RNN in different possible ways. See https://magenta.tensorflow.org/assets/sketch_rnn_demo/multi_predict.html for more.

2.2.4 Variational Autoencoders

Autoencoders are an unsupervised approach to learning a low dimensional feature representation of the factors of variation in the training data. In other words, the aim is to obtain a hidden vector whose values specify the characteristics necessary to distinguish examples in the data. For example, if we had a dataset composed of dog pictures, it would be useful to have a vector describing the main characteristics of each (and any) dog, such as the length of the fur, the shape of the snout or the length of the tail.

The autoencoder approach to solving this problem is based on learning an identity mapping while imposing an information bottleneck at an intermediate representation. Taking the picture of a dog as example, we start by using a function f - the encoder - to map the picture $\boldsymbol{x}$ to a low dimensional, intermediate vector $\boldsymbol{h} = f(\boldsymbol{x})$. A second function g - the decoder - takes $\boldsymbol{h}$ as input and maps it to the original high dimensional space of the picture, obtaining $\hat{\boldsymbol{x}} = g(\boldsymbol{h})$. If we further impose that the reconstruction $\hat{\boldsymbol{x}}$ should be as similar as possible to the original image $\boldsymbol{x}$, by optimizing $\ell_2(\boldsymbol{x}, \hat{\boldsymbol{x}})$, the small size of $\boldsymbol{h}$ will force the encoder to prioritize conveying the main characteristics describing the data, while the decoder will learn to closely reconstruct the data from those characteristics only.

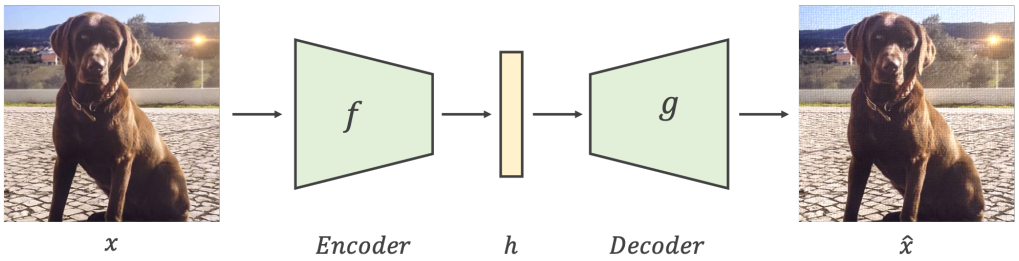


Figure 2.11: Base autoencoder architecture.

This architecture can be implemented using any of the previously described types of neural network as encoder and decoder, with the choice depending on the type of data at hand. Also, autoencoders are trained without supervision, since the answer they should output is the input itself, allowing direct computation of the loss term.

Accompanying the growing pervasiveness of Neural Networks, the basic form of autoencoder has seen new variations aimed at improving the learned representations. A type of autoencoder that has been particularly relevant in video prediction models is the Variational Auto Encoder (VAE) [29]. Variational autoencoders are a probabilistic interpretation of the generic autoencoders in which the deterministic encoder and decoder functions are generalized to probabilistic distributions $p_{encoder}(\boldsymbol{h}|\boldsymbol{x})$ and $p_{decoder}(\hat{\boldsymbol{x}}|\boldsymbol{h})$. The result of this interpretation is the model directly modelling the distribution that generated the input data, leading to better hidden representations and to the possibility of using the decoder as a generative model, from which new data points can be sampled.

With the goal being to estimate the underlying distribution $p_{data}(\boldsymbol{x})$ that generated the training data, we start by defining a graphical model defined in Fig. 2.12 and equations (2.11)) and (2.12)), that produces new data points, given an input latent vector $\boldsymbol{h}$, that is assumed to encode semantically meaningful information.

Figure 2.12: Graphical model of the latent vector model.

Joint distribution:

$$p_\theta(\boldsymbol{x}, \boldsymbol{h}) = p_\theta(\boldsymbol{x}|\boldsymbol{h})p_\theta(\boldsymbol{h}) \tag{2.11}$$

Marginal distribution over $\boldsymbol{x}$:

$$p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}|\boldsymbol{h})p_\theta(\boldsymbol{h})d\boldsymbol{h} \tag{2.12}$$

To train this model, we want to make the marginal distribution $p(\boldsymbol{x})$ as similar as possible to $p_{data}(\boldsymbol{x})$.

We start by defining the decoder $p(\boldsymbol{x}|\boldsymbol{h})$ as a multivariate Gaussian distribution whose parameters - because of the complexity of the mapping from the latent vector to the high dimensional output - are given by a neural network. The problem is now to learn the parameters θ that maximize the likelihood of obtaining a reconstruction that comes from the same distribution of the training data.

The maximization of this likelihood is however intractable as it would require the computation of $p_\theta(\boldsymbol{x}|\boldsymbol{h})$ for all possible values of $\boldsymbol{h}$. In VAEs, this problem is overcome by finding an approximation of the posterior $p_\theta(\boldsymbol{h}|\boldsymbol{x})$, which relates with $p_\theta(\boldsymbol{x})$ by

$$p_\theta(\boldsymbol{h}|\boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x}|\boldsymbol{h})p_\theta(\boldsymbol{h})}{p_\theta(\boldsymbol{x})}. \tag{2.13}$$

This approximation is achieved by defining an encoder $q_\phi(\boldsymbol{h}|\boldsymbol{x})$ as an inference network which will allow us to set a tractable lower bound on the data likelihood $p_\theta(\boldsymbol{x})$ that can be optimized. To rewrite $p_\theta(\boldsymbol{x})$ we start by noticing that it is constant with respect to $\boldsymbol{h}$ and therefore the same as its expectation with respect to $\boldsymbol{h}$:

$$log\, p_\theta(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big[log\, p_\theta(\boldsymbol{x})\big]. \tag{2.14}$$

Then, by applying the Bayes Rule as in equation (2.13), we obtain

$$\begin{aligned} log\, p_\theta(\boldsymbol{x}) &= \mathbb{E}_{\boldsymbol{h}\sim q_\phi(\boldsymbol{h}|\boldsymbol{x})}\left[log\, \frac{p_\theta(\boldsymbol{x}|\boldsymbol{h})p_\theta(\boldsymbol{h})}{p_\theta(\boldsymbol{h}|\boldsymbol{x})}\right] \\ &= \mathbb{E}_{\boldsymbol{h}\sim q_\phi(\boldsymbol{h}|\boldsymbol{x})}\left[log\, \frac{p_\theta(\boldsymbol{x}|\boldsymbol{h})p_\theta(\boldsymbol{h})}{p_\theta(\boldsymbol{h}|\boldsymbol{x})}\frac{q_\phi(\boldsymbol{h}|\boldsymbol{x})}{q_\phi(\boldsymbol{h}|\boldsymbol{x})}\right] \\ &= \mathbb{E}_{\boldsymbol{h}}\big[log\, p_\theta(\boldsymbol{x}|\boldsymbol{h})\big] - \mathbb{E}_{\boldsymbol{h}}\left[log\, \frac{q_\phi(\boldsymbol{h}|\boldsymbol{x})}{p_\theta(\boldsymbol{h})}\right] + \mathbb{E}_{\boldsymbol{h}}\left[log\, \frac{q_\phi(\boldsymbol{h}|\boldsymbol{x})}{p_\theta(\boldsymbol{h}|\boldsymbol{x})}\right] \end{aligned} \tag{2.15}$$

The second and third term of the last step in (2.15) correspond to the Kullback-Leibler (KL) Divergence of two distributions: $\mathrm{D}_{KL}(p||q) = \mathbb{E}\left[log\, \frac{p}{q}\right] \geq 0$. Hence we arrive at

$$log\, p_\theta(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{h}}\big[log\, p_\theta(\boldsymbol{x}|\boldsymbol{h})\big] - \mathrm{D}_{KL}(q_\phi(\boldsymbol{h}|\boldsymbol{x})||p_\theta(\boldsymbol{h})) + \mathrm{D}_{KL}(q_\phi(\boldsymbol{h}|\boldsymbol{x})||p_\theta(\boldsymbol{h}|\boldsymbol{x})). \tag{2.16}$$

The first term of (2.16) can be estimated through sampling of the decoder network. This sampling is differentiable with the reparameterization trick from [29] and can be optimized with gradient descent. The second term is the KL divergence between two distributions that have been defined to be Gaussians for which there is a closed form solution that can be maximized. The third term however, uses $p_\theta(\boldsymbol{h}|\boldsymbol{x})$ which is an unknown distribution. Nevertheless, as the term is the KL divergence between two distributions it will always be greater or equal to 0. In the end, we arrive at a lower bound (2.17) for the likelihood of $\boldsymbol{x}$ (Variational Lower Bound or ELBO) which is differentiable and can be optimized.

$$log\, p_\theta(\boldsymbol{x}) \geq \mathcal{L}(\boldsymbol{x}, \theta, \phi) = \mathbb{E}_{\boldsymbol{h}}\big[log\, p_\theta(\boldsymbol{x}|\boldsymbol{h})\big] - \mathrm{D}_{KL}(q_\phi(\boldsymbol{h}|\boldsymbol{x})||p_\theta(\boldsymbol{h})) \tag{2.17}$$

Interpreting the first term on the right hand side of (2.17) we see that for each $\boldsymbol{h}$ sampled from $q_\phi(\boldsymbol{h}|\boldsymbol{x})$ we take the expectation of the data being reconstructed. This term is therefore representing the reconstruction, or the decoder, and we wish it to be high so that the data can be correctly recovered.

The second term is a measure of how similar the approximate posterior is to the prior so, by maximizing (2.17), we're asking $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to be similar to the prior, which was previously defined to be a simple Gaussian distribution. The second term is thus serving as regularization for the encoder network. It is important to notice that $p_\theta(\boldsymbol{x})$ is a fixed distribution over the training data, which we are trying to model. Therefore, by maximizing the ELBO, we minimize the third term of equation (2.16). Minimizing this term results in training the encoder network, as it represents how well $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ is approximated.

To have some insight on the type of low dimensional representation VAEs learn, we can use a trained model to generate four initial data points, place them in the corners of a grid and interpolate the dimensions of the latent vector to generate new outputs with intermediate characteristics. This is possible because training the encoder and the generator network together incentivizes the model to learn a predictable coordinate system for the latent vector - a common ground that both encoder and decoder can interpret - leading some of its axis of variation to be associated with particular characteristics of the training data. As an example, using Sketch-RNN described in section 2.2.3, which uses recurrent encoder and decoder networks, it's possible to generate four initial sketches, conditioned on initial drawings provided by a human (see Fig. 2.13(a)). Then, the axis of the hidden vector can be interpolated to generate the rest of the grid. In the example of Fig. 2.13, the network is trained on sketches of both cats and pigs and interpolating between different latent vectors reveals how dimensions encode features such as the presence of a body or the shape of the head and ears.

Figure 2.13: Interpolation of the axis of the latent vector starting at four different input sketches and revealing how some dimensions are interpretable and associated with particular characteristics of the training data. (a) Human inputs and initial reconstructions, placed in the corners; (b) Population of a grid by interpolating from the initial sketches.

2.3 The Predictive Brain

Up to this point, discussion has centered around systems dedicated to feature detection, such as the human brain and neural networks, and how they can learn representations of the observed data that facilitate subsequent tasks that include perception, classification or image generation. Feature detection is not, however, the sole perspective on how systems learn representations. As discussed in this section, an alternative proposition of particular interest to this thesis is the possibility that representations are learned in a prediction based fashion.

Following up on the discoveries of Hubel and Wiesel in the 1960s (see section 2.1) and inspired by predictive coding techniques used in signal processing [30] that leverage the spatial correlation of natural images to approximate each pixel as a combination of surrounding pixels, a new interpretation of the antagonistic center-surround properties of the receptive field of neurons in the retina was proposed by Srinivasan *et al*. [31]. In his view, the signal at the center of the receptive field would be estimated by a weighted average of the signals measured in the surround and only the error between the two transmitted to subsequent levels in the visual pathway. Previous studies on the firing conditions of center-surround neurons in which applying the same stimulus in both regions of the receptive field produced little response would thus be explained by the center being easily predicted from the surround stimulus, resulting in no error signal. On the other hand, when the signals in both regions differ, a weighted average of the surround stimulus would fail to approximate the center, causing the neuron to transmit the error by firing.

Under this view, the early stages of visual processing would not be engaging in feature detection, rather they would be working both to reduce redundancy in the transmitted signals by only relaying information that is not easily predicted from spatial and temporal correlations in visual scenes and to keep transmitted signals in a small dynamic range, that can be transmitted with greater accuracy by neurons that have a limited firing rate.

Later work by Rao and Ballard [32] extended the notion of predictive coding in the retina to the visual cortex, in an hierarchical architecture. They proposed that at each level of processing there would be two types of neuron populations: representation neurons that, based on an internal world model, try to predict activity at the lower level and error neurons, computing the mismatch between the prediction coming from the upper level via feedback connections and the actual neuronal response at that level. Forward connections would then convey the prediction error to the upper layer for the world model to be updated, progressively allowing better estimates of future sensory inputs. This mechanism can be seen as each level of the hierarchy trying to suppress the information at the lower level that can be easily predicted, only allowing surprising deviations from the world's regularity to flow to higher levels.

Nowadays, the ideas of Rao and Ballard have matured into a theory of brain functioning and perception usually referred to as Hierarchical Predictive Coding or Predictive Processing (PP). This theory supports the idea that brains are predictive engines that are continuously trying to anticipate the struc-

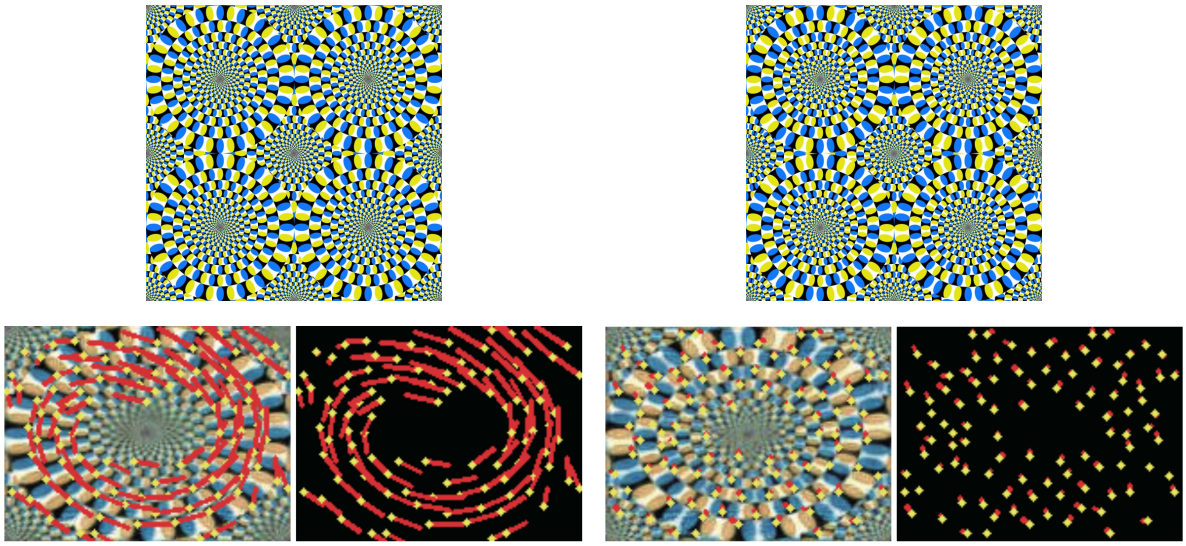
ture of the incoming sensory signal based on past experience, while the actual incoming observation is used as a target to determine the prediction error. Predictions can thus be corrected and progressively become more accurate. Under the light of PP, a visual scene is not perceived once a specific pattern of neuron activations is recognized by the brain – like in the feature detector model of section 2.1 - but rather when the brain's prediction successfully matches the incoming signal, which may only happen after online corrections to the initial guess.

Predictive Processing may seem rather intuitive once we realize that many of our day-to-day activities rely on expectations of what is going to happen – not only on basic levels, like anticipating a rapidly evolving visual scene to track a moving object or expecting to feel cold as we dive into a pool but also on more complex levels like anticipating someone's behavior due to social norms, or successfully reading a msisepelld wrod without even noticing [33]. Likewise, it is common for humans to feel strangeness when its expectation is met with surprise, like a change in our favorite music.

For the Predictive Processing theory to be feasible, however, an additional wrinkle must be taken into account and that is prediction driven learning. Traditional human education is based on the concept of teacher learning, i.e., on the presence of a supervisory signal indicating what the subject should be doing or concluding given the current input. This kind of learning, however, is at odds with the way humans learn about the world in the earlier stages of their lives, which is characterized by observation, experimentation and trial-and-error. These constitute a way of interacting with the world and learning what to expect at any given situation while having in our senses a self-supervisory signal that is free, rich and continuously available for comparison. This allows the optimization of a predictive model which in turn leads to an ever improving grasp of the surrounding world [34]: a baby that is capable of anticipating what is about to happen when her mom opens the fridge or predicting what she will see and hear as a stack of block toys starts leaning in one direction has already collected a lot of knowledge about the world and how to act within it.

It is important to notice, however, that despite growing evidence and support for a predictive theory of the brain, there is still much discussion and research to be made on how exactly PP could be implemented. A particularly compelling method for learning about the feasibility of such a system is computational modeling. An example of this is PredNet [35], a neural network based implementation of the Predictive Coding of Rao and Ballard that is, in essence, a video prediction model. Later work hypothesized that if PredNet was indeed a reasonable model of how the brain processes visual information, then it should perceive movement in the same way humans do, not only in actual moving but also on static visual illusions like the classical rotating snakes illusion [36]. To test this hypothesis, PredNet was trained on natural scenes to allow it to learn the spatiotemporal rules of the world and at test time was given 20 frames of the rotating snakes illusion - which is simply the same image repeated 20 times - and asked to predict the next few frames. Surprisingly, the frames predicted by PredNet on the illusion revealed motion that could be measured with optical-flow while, on a control version modified for humans not to perceive movement, this predicted motion was absent (see Fig. 2.14).

Despite not constituting proof that the human brain is engaging in predictive behavior, these results are an indication that a Predictive Coding inspired model and the human brain may perceive the world in the same way and can serve as motivation for further studies on biologically inspired models of video prediction.



(a) Illusion **(b)** Non-illusion

Figure 2.14: PredNet predictions on rotating snakes illusion. (a) the original illusion (top) is given as input and the resulting predictions display motion (bottom) which indicates that the network perceives the movement in the illusion, just as humans. (b) When a control image without illusory motion is given as input the network's predictions no longer show movement. Source [36].

2.4 State of the Art in Video Prediction

As explored in section 2.3, the early stages of human learning are defined by continuous observation and experimentation of how the world works. This is a thought that, according to the words of Alan Turing, may bring us closer to solving the quest of reproducing the human mind:

Instead of trying to produce a program to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain.

- Alan Turing, 1950 [37]

An important stepping stone on the path towards intelligent agents is thus providing them with the ability to explore the environment and to learn from interaction. An equally important notion that may lead to cracking this challenge, as suggested by Srivastava *et al*. [38] is that, if based on an observed sequence of visual queues an agent has the ability to predict the imminent future, then it must have acquired a representation of the spatial and temporal dynamics of the world. Predicting future video frames is perhaps the most straightforward materialization of this idea as the better a

system can make predictions about future observations, the better the acquired feature representation must be [39]. For example, a robot that is able to predict that a falling stick will become occluded by a box, must understand where the trajectory of the stick will take it, be capable of perceiving depth and also of recognizing that the box in the foreground is opaque.

Early work in anticipation of visual information includes sensori-motor networks [40], which emulate the interaction between the visual and motor systems in organisms to predict future visual stimulus. In [41], sensori-motor networks are applied to small image patches to predict the next time step's stimulus. However, when the problem is extended to more generic settings involving observations of a complete scene and longer temporal sequences, more complex models become necessary, making deep neural networks the predominant solution for video prediction in the last few years. Examples include a combination of CNNs and RNNs termed ConvLSTM [42] and applied in weather nowcasting and the work of Luc *et al*. [43] on semantic segmentation of future video frames for autonomous vehicles. In another direction, influential work by Mathieu *et al*. [39] focused on improving the quality of the generated video by experimenting with different loss functions as an alternative to ℓ_2, which is hypothesized to cause blurry predictions.

Despite significant advances in the last few years, video prediction remains an open problem, with predictions still being made on low resolution images (usually 64×64) and on relatively short temporal scales. Today, video prediction models can, for the most part, be classified in one of two types of approach: pixel motion models and latent variable models, which are explored in greater detail in sections 2.4.1 and 2.4.3. Additionally, models can be either deterministic or stochastic, *i.e*. conditioned on random variables, as described in section 2.4.2 and can be conditioned on an agent's actions, which is shown in red in Figs. 2.15 and 2.17 and discussed in more detail in chapters 3 and 4.

2.4.1 Pixel Motion Models

One of the most meaningful contributions to video prediction was perhaps the introduction of the concept of pixel motion by Finn *et al*. [44], Xue *et al*. [45] and De Brabandere *et al*. [46], which liberates the system from having to predict every pixel from scratch by modelling pixel motion from previous images and applying it to the most recent observation.

The fundamental idea behind pixel motion methods for prediction of video frames is that if the motion of pixels or objects present in the image is known, then it can be applied as a transformation to the most recently observed frame, in order to obtain the future position of the objects, while the background can be extrapolated from the previous frames. The main challenge with this approach lies in modeling the motion of the pixels and combining the predictions for each pixel/object into a single predicted image. The advantages are evident, though: by focusing prediction on the moving pixels, the dimensionality of the problem is greatly reduced. Also, since the model is based on the prediction of independent pixel movement and does not rely on any *a priori* physical model of the world, it has the ability to generalize what is learned during training to novel objects, never seen

before.

In Finn's work, two different solutions to the problem of modelling pixel motion are proposed (see Fig. 2.15). The first one, Dynamic Neural Advection (DNA), predicts a distribution over locations in the previous frame for each pixel in the new frame. This is done under a regularizing assumption that the movement is continuous and therefore the pixel will end up in a neighboring region of its original position, further reducing the dimensionality of the problem. Hence, the new position of the pixel is determined as a weighted average of the neighboring pixel values in the previous frame. In practice, this is equivalent to a two dimensional convolution between the most recent frame and the predicted motion kernel.

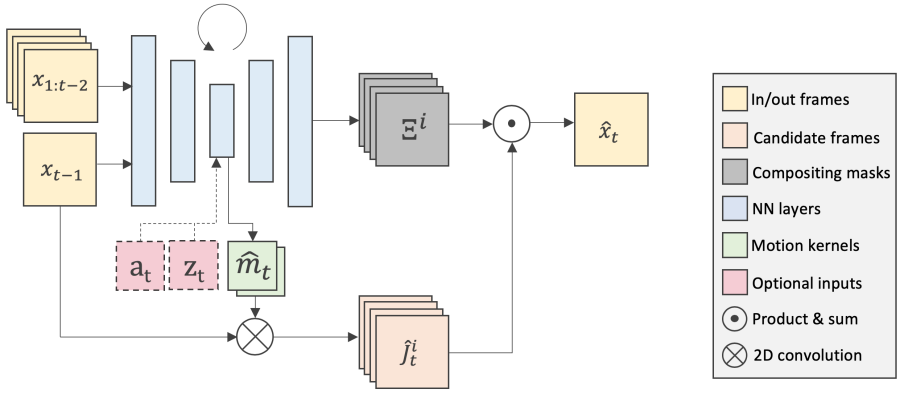


Figure 2.15: Generic pixel motion model.

The second way of modeling motion is the Convolutional Dynamic Neural Advection (CDNA), which adds the assumption that the motion predicted in a region of the image can be applied to other regions in the same image, introducing the notion that patches that are part of the same rigid object will move together and can share the same transformation. In this regard, multiple distributions over locations are predicted for the frame, with each corresponding motion kernel $\hat{m}$ being applied to the entirety of the previous image $x_{t-1}(u, v)$, producing several candidate predicted frames $\hat{J}_t(u, v)$:

$$\hat{J}_t(u, v) = \sum_{k \in (-k,k)} \sum_{l \in (-k,k)} \hat{m}(k, l) x_{t-1}(u - k, v - l) \tag{2.18}$$

This way, a stack of predicted images $\hat{J}_t(u, v)$ is obtained and it is therefore necessary to define the final value for each pixel, from all the candidates. The solution proposed in [44] is to also predict a compositing mask Ξ indicating the contribution of each pixel in each candidate predicted image $\hat{J}_t(u, v)$, to the final predicted image $\hat{x}_t$, that is obtained by an elementwise multiplication of the stack of predicted images and the mask:

$$\hat{x}_t^{(i)} = \sum_c \hat{J}_t^{(c)} \odot \Xi^c, \tag{2.19}$$

where c denotes the channel of the mask and stack of images. It is worth noting that the sum over each channel of the mask must sum to 1.

The object centric interpretation of CDNA is useful in two ways: on the one hand it reduces the number of motions to be predicted; on the other it facilitates learning object interactions in a planning

setting. An aspect common to the described pixel motion methods is that, by taking the expectation over different possible origins of the pixel being predicted, the model ends up averaging the values of all possible pixels, a result that will never contain the true value of the pixel (see Fig. 2.16). Instead, to comply with real world environments, the model should ideally sample from a learned distribution that assigns different likelihoods to each possible origin of the pixel being predicted.

2.4.2 Stochastic Video Prediction Models

A solution to the problem presented in 2.4.1 is proposed by Babaeizadeh *et al*. [47] in which the CDNA architecture is conditioned on a set of random variables (z_t inputs of Fig. 2.15). The main insight in this work is that the averaging of possible values for a pixel is originated by unobservable aspects of the world, such as the weight of an object or its drag coefficient on which the model is forced to give its best guess. This is illustrated by the toy problem of Fig. 2.16, in which a body with random shape and color moves in one of eight possible directions, also randomly selected. When a Video Prediction (VP) model such as CDNA is asked to predict the next few frames from a single input frame, it has access to the shape and color of the body but not to the direction of movement, which is a hidden aspect of the world. In these conditions, a deterministic model minimizing the mean squared error - such as CDNA - ends up predicting an average of all possible outcomes.

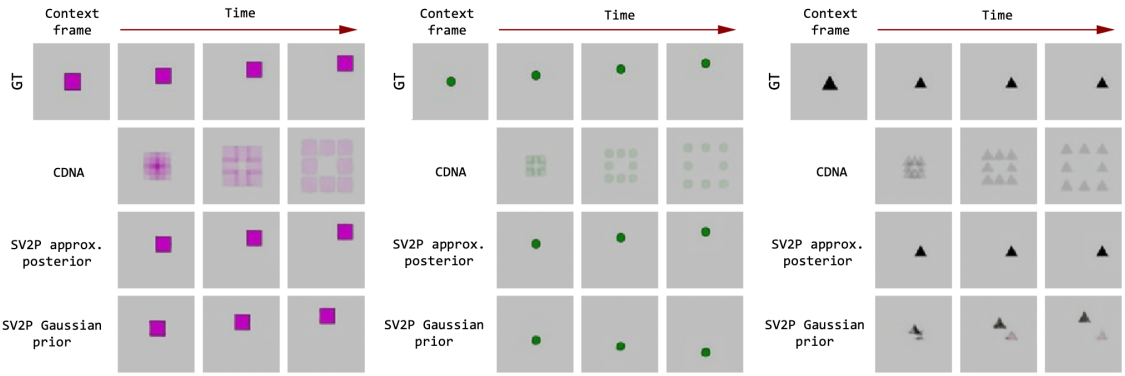


Figure 2.16: The importance of stochasticity in video prediction. While deterministic models such as CDNA are forced to average different possibilities in order to reduce prediction error, stochastic models like SV2P rely on random variables to make a decision for one of different possible futures. Better seen online at https://web.ist.utl.pt/ist181063/vp_examples/pixel_motion_models/sv2p/stochasticity_example/. Adapted from [47].

The Stochastic Variational Video Prediction (SV2P) model of Babaeizadeh *et al*. solves this issue by introducing a convolutional inference network $q_\phi(\boldsymbol{z}|\boldsymbol{x}_{0:T})$ - or encoder - that approximates the true posterior $p_\phi(\boldsymbol{z}|\boldsymbol{x}_{0:T})$ and outputs the parameters of a Gaussian distribution $\mathcal{N}(\mu_\phi(\boldsymbol{x}_{0:T}), \sigma_\phi(\boldsymbol{x}_{0:T}))$ from which a set of latent variables $\boldsymbol{z}$ is sampled. The architecture of CDNA, conditioned on the latent variables, is used to implement the generative model from which predictions can be sampled:

$$p(\boldsymbol{x}_{c:T}|\boldsymbol{x}_{0:c-1}, \boldsymbol{z}) = \prod_{t=c}^{T} p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}, \boldsymbol{z}). \tag{2.20}$$

To learn the parameters θ and ϕ of the generative model and the inference network, the variational lower bound used to train VAEs can be optimized, forcing the model to learn to interpret the hid-

den aspects of the environment from the latent variables. This can be seen in the two bottom rows of Fig. 2.16, where z conditions the model into choosing a direction of movement. If the learned inference network could be used at test time, the correct direction could always be predicted (third row of Fig. 2.16). However, at test time the model has no access to the future frames that serve as input to the inference network so the latent variables are sampled from a Gaussian prior, loosing expressiveness and leading to the model choosing a wrong (but still realistic) direction (fourth row of Fig. 2.16).

2.4.3 Latent Variable Models

Rather than focusing on the pixel motion approach of only modelling movement and applying it to the most recent frame, some authors have opted for pixel based approaches that use latent variables to reason about all the information contained in the video [38] [48]. Even though this may seem an unpromising solution to video prediction, different interpretations of the base architecture, described in this section and in chapter 4, can lead to improved results over pixel motion models.

Latent variable models are implemented with autoencoders (see Fig. 2.17) and start by obtaining a low dimensional feature representation $\boldsymbol{h}$ for each observed frame. Typically, a Recurrent Neural Network is then used to project this representation into the future, taking into account the past sequence of $\boldsymbol{h}$ vectors and finally, for each predicted $\hat{\boldsymbol{h}}$ that the RNN outputs, a decoder network is used to construct a future frame.

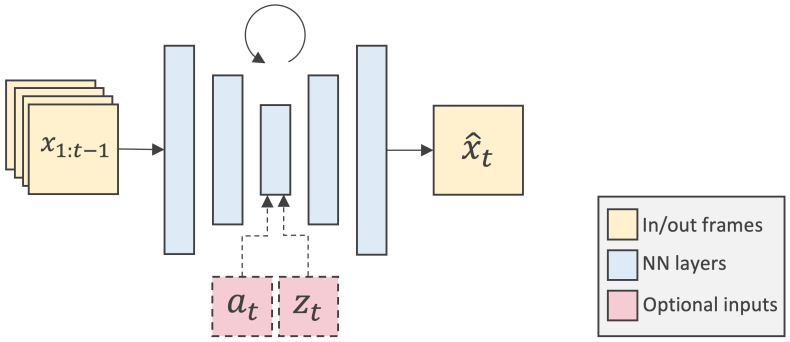


Figure 2.17: Example architecture of a simple latent variable video prediction model: a recurrent autoencoder model, optionally conditioned on actions $\boldsymbol{a}_t$ and on random variables $\boldsymbol{z}_t$.

This type of architecture poses two main difficulties: on the one hand the encoder is tasked with selecting the essential factors of variation of the scene and not only expressing them using a small set of real numbers, but also in a way that the decoder can interpret. On the other hand - and unlike pixel motion models - the decoder has to directly generate every pixel of the predicted frames.

Still, if the model learns to produce good, low dimensional representations of the data, it is possible to obtain crisp predictions for much longer into the future. This is because, as long as the $\boldsymbol{h}$ vectors are consistent in time and the decoder can interpret them, every new frame will be constructed in the same conditions, preventing the accumulation of errors. This is unlike pixel motion models, which

rely on the actual pixel values from the most recent frame (whether it was observed or predicted) to construct the new one, meaning that errors can easily compound and lead to blurry predictions.

This feature of latent variable models expresses itself in the works of Oh *et al*. [49] and Denton & Fergus [50]. In the first of these works, an action-conditioned model based on a convolutional autoencoder is proposed and the predicted frames are used as input to a Reinforcement Learning algorithm that learns how to play Atari games, making action decisions up to 100 steps into the future. In another direction, Denton proposes a stochastic model - SVG-LP - which generates future frames from a latent vector that is a combination of both a deterministic estimate and samples from a learned prior distribution that models unobservable aspects of the environment on which the model must make a decision. As before, these are aspects that may affect future outcomes in unexpected ways, such as the spin of a falling ball that will cause it to bounce in an unknown direction once it hits the floor. Even though the stochastic nature of SVG-LP often makes its predictions diverge from the actual observed events, the model is capable of generating sharp, realistic looking frames of natural scenes for more than 100 time steps into the future. An example of SVG-LP's predictions on BAIR dataset is shown in Fig. 2.18, compared against SV2P's predictions.

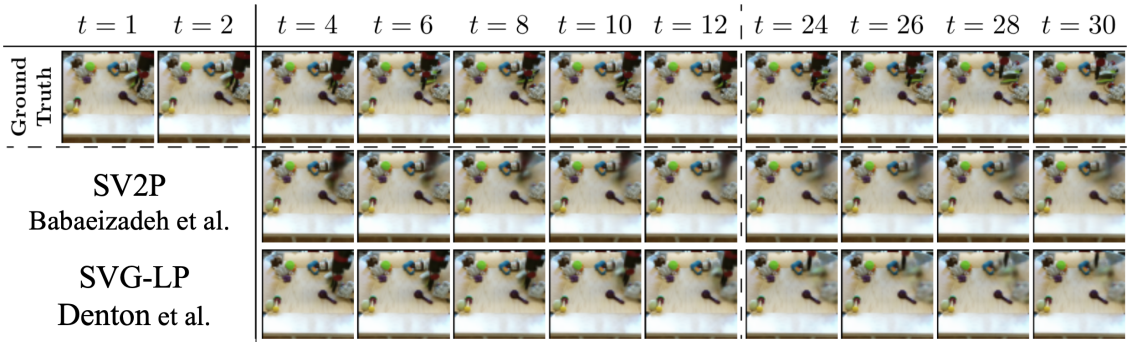


Figure 2.18: Example video predictions from two different models on BAIR dataset. SV2P is a stochastic pixel motion model, while SVG-LP is a stochastic latent variable model. Both models were given two input frames and trained to predict up to time step 12 (dashed line). Better seen online at https://web.ist.utl.pt/ist181063/vp_examples/pixel_motion_models/sv2p and https://web.ist.utl.pt/ist181063/vp_examples/latent_variable_models/svg/. Source [51].

Finally, a second benefit of latent variable models is the possibility of incentivizing the low dimensional representation to disentangle (*i.e*. separate) the factors of variation of the data, creating interpretable representations and allowing the extrapolation of different components of the video separately. This is explored in chapter 4 as a possible solution for reducing the dimensionality burden of the problem, which remains one of the main obstacles in video prediction. Furthermore, the introduction of action information may be one of the possible solutions for obtaining disentangled representations in robotic tasks while, as previously hinted, it may allow an agent to reason about possible interactions with the environment before making action decisions, a topic that is explored in detail in chapter3.

3

Adding actions to the mix

In the previous chapter the discussion focused on the relationship between perception, learning and prediction and how video prediction may bridge predictive theories of the human brain and intelligent machines. But as described in the first section of this chapter the discussion cannot be complete without taking the role of actions in consideration. Furthermore, knowledge of the actions allows the use of video prediction in robotic planning tasks, which we detail in the second section. Finally, this leads the discussion to our first contribution, which is detailed in the last section of the chapter.

3.1 Active Prediction

In section 2.3 the predictive nature of the brain and how it may permeate both perception and learning in most activities of our daily lives was discussed. During a tennis match, for example, a player can produce serves of up to 210 km/h, meaning that the ball traverses the court in an impressive 400 ms. Even though most people would believe they saw the actual movement of the ball, the average 200 ms of latency between the arrival of visual information at the retina and its propagation to the visual cortex mean that the ball has already moved 12 metres by the time a player sees it at a given position. This implies that the perceived movement of the ball is in fact caused by the brain engaging in some kind of predictive processing. Moreover, for a tennis player to be able to execute the fast, precise movement necessary to return the serve, sensori anticipation must somehow be connected with action decision and the motor cortex.

This relationship between perception and action is at the core of Karl Friston's Free Energy Principle [52], which provides a probabilistic account of the Predictive Coding theory while generalizing to the motor system the idea that the brain is committed to performing prediction error (surprise) minimization. Specifically, Friston argues that biological organisms are defined by their natural tendency to resist disorder [53], *i.e.*, by seeking states that allow them to remain within physiological bounds. In doing that, agents associate beneficial states with low entropy (or uncertainty) and work to mini-

mize surprise or, in other words, the likelihood of undesirable states. The surprise of sampling some data cannot, however, be directly obtained but can nevertheless be optimized by Free Energy minimization, which is an upper bound on surprise. Free Energy is an information theory measure which is a function of a probabilistic representation of the causes of sensori observations and the actual perceived sensations. This interpretation entails that, while in the perspective offered in section 2.3 updating the internal model of the world was the single mechanism available for reducing the error between sensory predictions and actual observations, action is now regarded as a complementary way of reducing such an error, by selectively acting upon the world in ways that result in desirable observations and bring the internal model's predictions about.

Despite concrete evidence in support of the Free Energy Principle still being limited, recent work provides empirical evidence in its favor. With the intent of studying how the motor system reacts to body illusions that cause an information mismatch between different sensory sources, an experimental setup is proposed in [54] where virtual reality is used to deceive the visual system of human subjects by showing a virtual arm positioned in a place that does not correspond to the true, resting, position of the subject's arm (see Fig. 3.1). This setup creates a conflict between visual and proprioceptive information, therefore inducing an error between the expected and perceived body pose. Interestingly, in the experiment horizontal forces were measured by a sensor attached to the users' hand, revealing involuntary movement directed at the position of the virtually observed arm. These observations hint that the brain may be trying to minimize the surprise caused by the sensori mismatch by acting to bring about expected states, which is in line with the hypothesis of predictive error minimization.

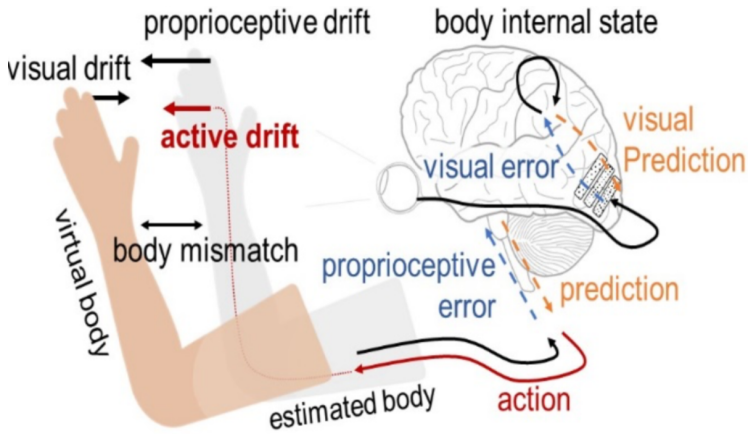


Figure 3.1: Visual and proprioceptive error about the expected position of the arm is induced by showing a virtual displaced arm to the subject and hiding its actual arm. In an attempt to inhibit this error, the brain issues actions that lead to an active drift of the arm towards the artificial observed arm. Perceptual proprioceptive drift is caused by the visual error mismatch while a perceptual visual drift has also been shown to exist, towards the real arm. Source [54].

In the last decade, Friston's ideas have steadily gained support and popularity and other authors have extended his views to other facets of human condition. Of particular interest to this work is a predictive view of cognitive control proposed by Pezzulo [55]. Other than just an active online (overt) loop of sensory acquisition and error correction in short time scales, Pezzulo suggests the existence of an off-line (covert) loop that allows the execution of imaginary actions and the unrolling

of their consequences into a long-term future. Combined with theories of sensorimotor control such as Affordance Competition [56], this framework allows the mental rehearsal of multiple competing courses of action in parallel and the progressive elimination of each possibility until a decision has been made, effectively enabling long-term action planning.

If these ideas are correct, perception, prediction and action are inextricably connected and should therefore be studied as a single system of active prediction, rather than separately. Under this light, the rest of this work focuses on video prediction conditioned on action information - a topic that was previously mentioned in section 2.4 - with special focus on models designed for robotic planning tasks, as the ones described in the next section.

3.2 Prediction Based Planning

If, in fact, human condition has evolved in the direction of exploiting the capabilities of sensory prediction for determining its own actions, then so should artificial prediction models. This notion motivates what is perhaps the most important application of video prediction: action planning. The relevance of this task lies on the fact that it is the missing piece that allows to close the loop on the active, prediction based framework we've been discussing: acting upon the environment allows an agent to observe the inner workings of its world, which in turn elicits better sensory prediction, enabling further, more complex interactions.

In this type of tasks, video prediction models serve as forward models, *i.e.*, as a function that, given the current state and action, outputs the next state. This is explored in the work of Ha *et al*. [57], with the introduction of an architecture that learns a policy for solving OpenAI Gym [58] reinforcement learning problems using an encoder of observed video frames and a MDN-RNN to predict future visual codes, given current and past observations and executed actions.

In a robotic context, Agrawal *et al*. [59] learn a model of intuitive physics through the interplay between a forward and an inverse model. Given an initial and goal state - for example, the image of an object in an initial position on a table, and a similar image in which the object was moved to a desired position (see Fig. 3.2(a)) - an inverse model should be able to output the actions whose execution takes the object from one state to the other. However, in real world scenarios, different possible courses of action could result in the desired effect, making such an inverse model difficult to learn in practice. At the same time, one of the main obstacles in learning forward models is the dimensionality burden associated with predicting every pixel. Nonetheless, as Agrawal notices, in most practical scenarios an abstract description of the environment - such as object pose and motion - is more useful than the actual value of each pixel. Hence, he proposes to learn an abstract, low dimensional, feature representation of the scene (see Fig. 3.2(a)), in which the inverse model can easily determine the action that leads from one representation to the other while the forward model can predict the state that results of executing a specific action, without having to produce every single pixel. Learning

this feature representation is possible because training the inverse model serves as a constraint, determining that it must be possible to infer the executed action from the feature representation of two images.

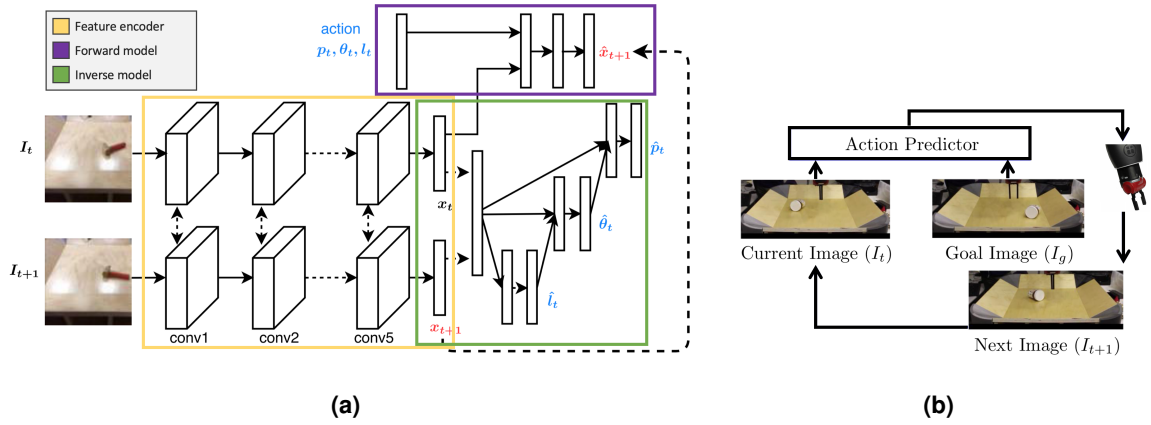


Figure 3.2: (a) Model architecture highlighting the feature encoder and the inverse and forward model taking feature vectors as input. A poke action is represented by a triplet of point to poke (p_t), a direction (θ_t) and a length (l_t). **(b)** Greedy planner loop, using the inverse model as action predictor and a Baxter gripper to execute the actions. Adapted from [59].

To confirm the validity of this model, a Baxter robot was tasked with poking different objects placed on top of a table, from an initial position to a goal position. Having an image of both states, the inverse model can be used to infer an action that will bring the object closer to the goal (see Fig. 3.2(b)). By executing this action and continuously repeating the process, the robot successfully achieved its goal in most situations, even when shown previously unseen objects.

In spite of the good results achieved by this architecture, it represents a greedy planner, as actions are only planned one step ahead in the direction that reduces the distance to the goal the most. Such an approach can result in sub-optimal solutions in which the goal is achieved but not in an efficient way - for example, by pushing the object in zig-zags instead of straight ahead. Alternatively, this problem should be mitigated by choosing an approach in which each decision takes into account both its current and future consequences.

An example of such a strategy is the cognitive planning framework discussed in section 3.1 based on testing multiple possible courses of action and unrolling their consequences into the future which has also been adapted for robotic tasks. One of the first works in this direction is the Visual Model Predictive Control (MPC), proposed by Finn *et al*. [60] and based on the CDNA video prediction architecture (see section 2.4.1). In this experimental setup, a user selects an initial pixel that is part of an object, to be pushed to a goal location by a robotic arm (see Fig. 3.3). To arrive at the goal, the model starts by sampling $\mathcal{M}$ action sequences from a uniform distribution Q_j. These, along with past observations of the scene, are fed to the video prediction model, in order to determine the action sequence that maximizes the likelihood of the designated pixel moving to the goal position in the future. Because a uniform distribution is not the best model for the actions the robot should execute, a multivariate Gaussian distribution Q_{j+1} is fitted to the $\mathcal{K}$ action sequences with best probability of

success and the process of sampling actions and predicting their future is repeated a fixed number of times (see algorithm 1). Once this iterative process is finished, the action with best chances of success is selected for execution, a new observation is collected and the whole process repeats itself, until the designated pixel has reached the goal position.

Algorithm 1: Visual MPC

inputs: predictive model $\mathcal{M}$, designated pixel. $\{d_0\}$, pixel goal position $\{g\}$
for *t=1,...,T* **do**
  Initialize Q_1 with uniform distribution.
  for *j=1,...,J_t* **do**
    Sample M action sequences $\mathbf{a}^{(m)}_{t:t+H-1}$ from Q_j.
    Use model $\mathcal{M}$ to compute the distributions over future pixel locations $\mathbf{P}^{(m)}_{t+H-1}$.
    Fit multivariate Gaussian distribution Q_{j+1} to K samples with highest probability of success $\mathbf{P}^{(m)}_{t+H1}$.
  Execute best action $\mathbf{a}^*_t$ with highest probability of success.
  Observe new image I_{t+1}.
  Set next designated pixel location d_{t+1} using optical flow computed from image observations $I_{t:t+1}$, and d_t.

Figure 3.3: A robotic arm using Visual MPC to push objects from an initial position (in green) to a designated goal position (in red). Source [60].

The Visual MPC model succeeds in learning how to make basic manipulations of objects - even previously unseen ones - from raw images while requiring minimal engineering. It is also an example of how prediction and action can be combined to produce continuous self-improvement, even though the framework is yet to be demonstrated on less controlled environments. To reach that stage, it will be essential to design video prediction models that allow to learn a comprehensive model of the world and, perhaps even more importantly, that are capable of generalizing that model to previously unknown environments, with a brief adaptation phase. Moreover, as will be discussed in the next section, the ability to choose the best possible video prediction model for planning will also be a determining factor in achieving such a milestone.

3.3 Evaluating Video Prediction Models

Given the central role of video prediction models in the predictive planning algorithms described in section 3.2, it is only natural that the ability of these methods to select the best possible action in a given situation is very dependent on how well the video prediction model can anticipate future observations. Having a metric that can rank video prediction models based on how well they perform as a forward model is therefore of fundamental importance.

As will be described in section 3.3.1, most state of the art work in video prediction measures the performance of the models using metrics designed to reflect human perception of quality. While these metrics might be useful for applications that facilitate human decisions such as precipitation nowcasting [42], we argue that they are not necessarily adequate in action oriented robotic applications such as planning, where the quality of the video prediction model should be measured by how well it can guide action decisions from the predicted frames. Inspired by this notion, in section 3.3.2 we propose a new, simple metric for ranking video prediction models from a robotic standpoint.

3.3.1 The human standpoint: PSNR, SSIM, LPIPS & FVD

Recent work on video prediction has, for the most part, used quality metrics which focus on measuring image quality from a human perception perspective. These are metrics traditionally used to assess video compression algorithms as well as transmission quality. Image and video quality or Quality of Experience (QoE) is a very subjective concept, which depends not only on the data fidelity of the reconstructed image but also on the interests, past experience and expectations of the viewer, as well as the setup conditions such as the lighting, distance and screen [61]. Given the nature of the QoE, its standard measure is the Mean Opinion Score (MOS), which is the average quality rating, given by a sample of viewers. However, because surveying users is expensive, difficult to scale and can't provide real time evaluations, there has been extensive research on objective metrics - such as PSNR and SSIM - that approximate the MOS.

Peak Signal to Noise Ratio (PSNR) is maybe the most popular quality metric in image and video processing, often serving as a benchmark for novel, more complex, metrics. It is a simple logarithmic measure of the mean squared error and therefore compares the images pixel by pixel, not taking into account the content. This pixel level nature of PSNR is its main drawback, as it leads to pathological cases (see Fig. 3.4 in which it fails at approximating human judgement [61].

(a) Original **(b)** Contrast suppressed **(c)** JPEG compressed **(d)** Blurred

Figure 3.4: Three different distortions of the Goldhill image. Despite distortion (b) having better quality, all 3 distortions score the same in terms of PSNR. On the other hand, SSIM respectively assigns the scores 0.9622, 0.6451 and 0.6430, clearly separating distortion (b). Source [62].

An alternative metric that addresses this problem is the **Structural Similarity (SSIM) Index**, proposed by Wang *et al*. [62], which starts from the principle that signals that are close in space have strong

dependencies between each other and that the human visual system is highly adapted for extracting this structural information, therefore proposing a method for comparing corresponding image patches. In the work of Wang, structural information is considered to be the attributes of the image that reflect the structure of objects, apart from mean intensity and contrast and is defined as the correlation coefficient between two image patches. SSIM indices are calculated using a sliding window of size 8×8 which produces an index map. An index is 1 if the structures of the two images within each patch are exactly the same and the final SSIM score corresponds to the average of the index map which is in the range $[-1, 1]$. Despite its improved performance over PSNR - as exemplified in Fig. 3.4 - SSIM is still sensitive to scale and geometric distortions [63], meaning that a small rotation or translation between objects in the images being compared can be enough for the score to diverge from human opinion.

With the difficulty of creating hand engineered metrics for modelling the ambiguity of human perception and answering questions such as: *is a red circle more similar to a red square or a blue circle?* The community turned to neural networks and the data representations that they construct when trained on large scale datasets. In particular, Zhang *et al*. [64] proposed the **Learned Perceptual Image Patch Similarity (LPIPS)**, a frame level method in which two images being compared are given as input to a deep convolutional neural network such as VGG [65] trained on ImageNet [66], in order to obtain deep feature embeddings of the images, from different layers. These are then normalized and scaled and the ℓ_2 loss taken, with the final score being the average ℓ_2 distance between embeddings across the spatial dimension and across all layers. The fundamental insight behind this method is that representations that are useful in tasks such as semantic classification in natural scenes, are also representations whose Euclidean distance is predictive of human quality perception.

Despite the remarkable results of LPIPS, it is still a frame level method and therefore doesn't take into account sequential information such as temporal coherence. The best option for evaluating predicted frames from a human standpoint would be a video level metric as is the case of the **Fréchet Video Distance (FVD)**, proposed by Unterthiner *et al*. in [67]. FVD proposes to measure the similarity between two sample videos as the distance between the distribution that generated the original data P_R and the distribution defined by the prediction model P_P. However, because neither of the probability density functions is known, a procedure similar to the one adopted by LPIPS is used to approximate them: a $3D$ CNN is trained to recognize actions in a large-scale dataset of natural videos involving humans, with action recognition being viewed as a temporal extension of the image classification task of LPIPS, used to obtain feature representations of the videos. P_R and P_P are then approximated by fitting multivariate Gaussian distributions to the activations of the CNN. Finally, the FVD of the approximated distributions P'_R and P'_R is obtained as the Fréchet distance [68] between the two.

3.3.2 The robotic standpoint: action inference as a new metric

Despite all the metrics described in section 3.3.1 being widely used in state of the art video prediction research, we argue that they are not necessarily adequate in action oriented applications such as robotic planning. Instead we propose a new angle on the problem where the key idea is that the quality of the video prediction model should be measured by how well it can guide an agent in deciding its actions from the predicted frames.

We start by assuming that the better the dynamics representation of the agent is at encoding action features, the better it will be for planning actions based on the expected outcome. Under this assumption, the problem turns into evaluating how well a VP model is encoding action features and assigning it a score based on such evaluation. With this in mind, we hypothesise that the ability to observe a sequence of predicted frames and infer the executed actions should be an indication that the VP model is correctly encoding action features. To better illustrate this idea, first consider a failure case: if the VP model generates a sequence of predicted frames that do not correspond to the actions executed by the robot, then no action inference model can recognize the correct set of actions from the predicted images, resulting in a low action inference score. On the other hand, if the VP model understands the consequences of the input actions, then the frames it predicts should correctly reflect the action and its consequences, allowing an inference model to recognize the actual executed actions and attain a high score.

To put the idea in practice, we first train a simple CNN to infer the actions executed between every two frames of predicted videos. Specifically, the network receives a pair of frames concatenated along the channels dimension, as illustrated in Fig. 3.5, for the action that led from one frame to the other to be inferred. We choose an eight layer CNN with filter size 5 and number of filters starting at 128 and being reduced by half at each layer, with exception of the last layer, where the number of filters corresponds to the dimensionality of the action representation $|\boldsymbol{a}|$. With exception of layers 4 and 6, where stride is 1, we use stride 2 with zero padding, meaning that after each layer the size of the feature map is reduced by half, until in the output it is $1 \times 1 \times |\boldsymbol{a}|$. It is important to emphasize, however, that the focus is not in finding the architecture that provides the best possible action inference from the predicted frames, but rather to use a common action inference model in frames predicted by different VP models, which allows the predictions to be compared in terms of how well action features are encoded.

The actions $\boldsymbol{a}$ are assumed to be continuous and multidimensional, to be representative of most robotic control action-spaces. In the case of BAIR Dataset, for example, actions can be represented as gripper displacements δx and δy in the xy plane. The goal of the inference model is then to recognize the true gripper movement in these two axis (exemplified in Fig.3.6) for every time step.

Because action dynamics should not change over time, model parameters are shared across all time steps of a sequence, as indicated by the dashed arrows in Fig. 3.5. Also, while a RNN would typically have been useful for learning the sequence of executed actions, we chose to input a window of two

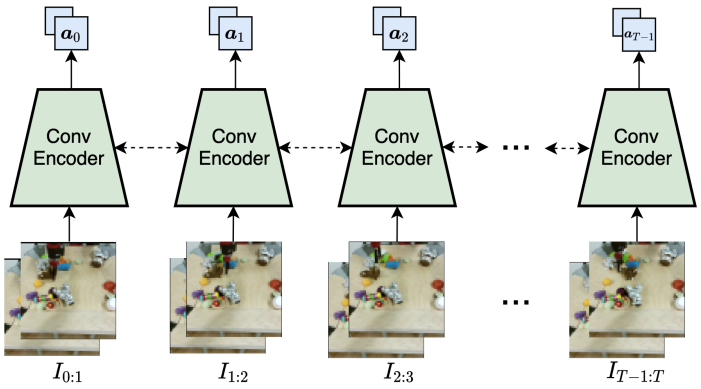


Figure 3.5: Action inference network unrolled in time. At each time step the network receives a pair of frames and outputs a multidimensional recognized action. Dashed lines indicate that the weights of the network are shared for all time steps.

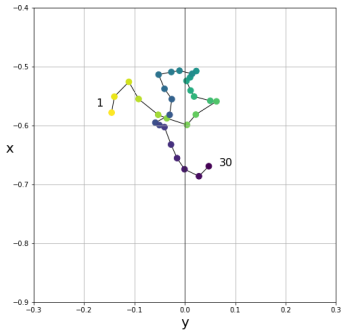


Figure 3.6: Bird's eye view of the a gripper trajectory in BAIR dataset.

frames at a time, cutting off any temporal correlation between actions. This forces the inference model to identify actions from the pixels instead of focusing on learning the temporal action distribution. The option for a window size of two frames is due to the fact that in the selected dataset (see section 5.1.1) the robot's actions are randomly updated every two frames. For datasets with different conditions, however, the window size parameter can control the temporal information received by the network without shifting the attention of the model from the frames, and it is expected that bigger windows should result in better action inference.

In order to compare how the proposed metric correlates with PSNR, SSIM and FVD, we select a group of VP models from prior work to be tested using our metric. By comparing model performance under metrics designed to predict human quality perception with our metric, designed to assess the capabilities of the model to guide action decisions, we intend to answer the question *"Does a good video prediction from a human perspective correspond to a good video prediction from the standpoint of a robot making decisions?"*. This is an interesting question considering that a change in model ranking, when compared to PSNR or SSIM, may not only influence the choice of the VP in an action planning experiment but also indicate that the best representation for a robot to make a decision may not look like anything a human may recognize and inspire new lines of research such as optimizing for losses other than ground truth similarity.

Having delineated a solution for the first question proposed in section 1.2, the focus is now turned to the two remaining research questions, which ultimately aim at designing a video prediction model directed at the kind of planning tasks described in this chapter. Hence, one of the goals of the approaches detailed in chapter 4 is for the proposed solution to achieve the best performance among the tested models in our action inference metric.

4

Separating sources of information: disentangled video prediction

With the intent of exploring solutions for reducing the dimensionality burden of video prediction, this chapter starts with an introduction to the concept of disentangled representations. A video prediction model - DRNET - that takes advantage of this notion is then presented. This serves as a baseline for our proposed video prediction model ADR-VP, which is composed of three parts, described in the following sections.

4.1 Disentangled Representations

The notion of learning representations of the data that facilitate the realization of subsequent tasks has been one of the central themes in this discussion (see sections 2.1 and 2.2.1), with especial consideration given to autoencoder architectures (section 2.2.4) that impose constraints on the dimensionality and statistical structure of the hidden representation to discover the underlying factors generating the data. Typically, these factors of variation are described as being a set of sources interacting in complex ways. Ideally, these sources could be *disentangled* and varied independently of one another to produce a change in the associated characteristics of the observed data [69]. For example, the image of a dog can be roughly generated from parameters that are sensitive to the color and length of the fur, the size of the dog, and the length of his muzzle, and changing any of those parameters results in a different dog image being generated. In practice however, obtaining a representation in which the underlying causes of the data are factorized in this fashion is a difficult task, currently seen as one of the challenges in AI with most potential to cause significant advances in the field. This is because, as pointed out in [21], an independent understanding of each factor of variation in the data translates in a significantly easier solution to any subsequent task, such as classification or prediction, making disentanglement the most robust approach for representation learning. Furthermore, a model that is capable of learning disentangled representations of the observed data is well suited for generalization

tasks like transfer learning or zero-shot learning [70].

Blind Source Separation: Independent Component Analysis

Early work on separating the underlying causes of observed data focused on methods for solving the linear sub-problem, in which the observed signal is a linear combination of the different sources. Despite having applications in different types of Blind Source Separation problems, the most famous application of these methods is the *Cocktail Party Problem*, that consists in separating the superimposed voice signals of different people speaking at the same time. Starting in the mid 1990's, the main solution to this problem was Independent Component Analysis (ICA). Considered an extension of Principal Component Analysis (PCA) [71], ICA looks to find a linear representation in which data components are, as much as possible, statistically independent [72]. Therefore, unlike PCA which tries to find uncorrelated components of the data, ICA looks for independent components, starting from two assumptions: first, that the data generating sources - for example the superimposed voices - are statistically independent and, second, that each component comes from a non-Gaussian distribution [72].

Encouraging factorial distributions

In the last decade, unsupervised architectures such as the VAE (see section 2.2.4) have shown the ability to learn disentangled representations of the data while relaxing some of ICA's assumptions and, most importantly, while extending applications to problems in which sources of variation present non-linear interactions - as is the case with most real-world data. As studied by Higgins *et al*. [73], the disentanglement capabilities of VAEs stem from the KL (regularization) term of its objective, which encourages the learned posterior $q(\boldsymbol{h}|\boldsymbol{x})$ to be close to the Gaussian prior $p(\boldsymbol{h}) \sim \mathcal{N}(0, I)$ (see section 2.2.4). Because the prior's covariance is set to the identity, it constitutes a factorial distribution which in practice results in an incentive for the dimensions of the latent vector $\boldsymbol{h}$ to be independent. With this in mind, Higgins proposed β-*VAE*, a simple variation of the base model in which a weighting factor β is added to the KL divergence term, allowing disentanglement to be further incentivized by increasing the weight of the KL term:

$$\frac{1}{N}\sum_{i=1}^{N}\Big[\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{x}^{(i)})}[log\, p(\boldsymbol{x}^{(i)}|\boldsymbol{h})] - \beta KL(q(\boldsymbol{h}|\boldsymbol{x}^{(i)})||p(\boldsymbol{h}))\Big]. \tag{4.1}$$

A shortcoming of this approach, however, is that increasing the weight of the regularization term to improve disentanglement implies a lesser emphasis on reconstruction, leading to a trade-off between the two.

Following up on Higgins' work, Kim *et al*. [74] proposed a solution to the trade-off problem. They start by re-writing the KL divergence term of the VAE objective as

$$\mathbb{E}_{p_{data}(\boldsymbol{x})}[KL(q(\boldsymbol{h}|\boldsymbol{x})||p(\boldsymbol{h}))] = I(\boldsymbol{x};\boldsymbol{h}) + KL(q(\boldsymbol{h})||p(\boldsymbol{h})), \tag{4.2}$$

where $I(\boldsymbol{x};\boldsymbol{h})$ is the mutual information between $\boldsymbol{x}$ and $\boldsymbol{h}$. Minimizing the $KL(q(\boldsymbol{h})||p(\boldsymbol{h}))$ term with higher values of β brings $q(\boldsymbol{h})$ closer to the factorial prior, while minimizing the mutual information reduces the amount of information about the data point $\boldsymbol{x}$ that is expressed in $\boldsymbol{h}$, handicapping the model's capacity for reconstruction. In order to directly penalize high KL values in (4.2), Kim proposed the *FactorVAE*, which adds a third term to the VAE objective:

$$\frac{1}{N}\sum_{i=1}^{N}\left[\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{x}^{(i)})}[log\, p(\boldsymbol{x}^{(i)}|\boldsymbol{h})] - KL(q(\boldsymbol{h}|\boldsymbol{x}^{(i)})||p(\boldsymbol{h}))\right] - \gamma KL(q(\boldsymbol{h})||\bar{q}(\boldsymbol{h})), \tag{4.3}$$

where $\bar{q}(\boldsymbol{z}) = \prod_{j=1}^{d} q(z_j)$ and the third term represents the *Total Correlation* [75], which is a measure of dependence between random variables. This formulation allows the weight on the second term to be kept as 1, which prevents the mutual information from being penalized while separately encouraging independence in the dimensions of $\boldsymbol{h}$. Figure 4.1 shows *FactorVAE*'s ability to recognize different interpretable factors of variation in the data and representing them independently of one another.

Figure 4.1: Latent vector interpolation in *FactorVAE*. The model identifies interpretable factors of variation - such as fringe length - that can be independently controlled. Source [74].

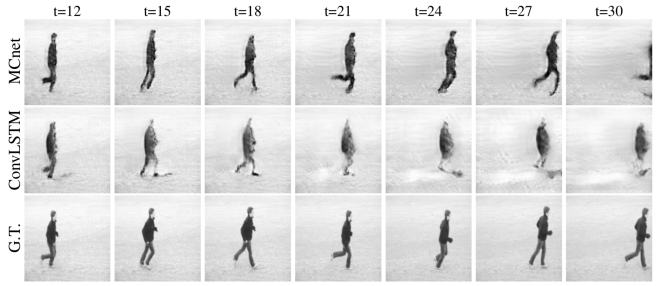


Figure 4.2: Future frame predictions obtained with MCnet by continuously feeding the predicted frame $\hat{\mathbf{x}}_{\mathbf{t+1}}$ as a new input on KTH dataset. First row: MCnet's predictions; Second row: ConvLSTM baseline's predictions; Third row: ground truth video. Source [76].

From structural biases

Approaches like the β-*VAE* and *FactorVAE* show impressive results while preserving the base autoencoder architecture that provides the models with the necessary flexibility to be applied in a wide range of problems. Yet, some applications have an intrinsic structure that gives *a priori* information about the interacting causes, which can be leveraged to facilitate disentanglement without compromising on supervision. This insight is particularly relevant in sequential data, which usually features both time variant and invariant information or hierarchical relations in its structure. Speech utterances, for example, present sequence-level attributes such as the speaker's fundamental frequency and volume and segment-level attributes affected by the phonetic content, such as spectral contours and formant duration, allowing each type of attribute to be explicitly modeled in separate [77].

In video sequences, efforts have focused on disentangling content (e.g. background and object appearance) from the changing, dynamic information that is revealed as observations unfold. Approaches include the Motion-Content Network (MCnet) [76], which separates the two sources of in-

formation by encoding them in two different streams: the content encoder receives the most recently observed frame x_t as input, which informs about the appearance but not the motion, as it is a single static frame. On the other hand, the motion encoder takes the difference between consecutive frames x_t and x_{t-1} at different time steps, therefore having access to the changes in the scene but not to the static content. From the motion and content representations, the model then reconstructs the next frame $\hat{x}_{t+1}$, which can be fed back as input to keep predicting future frames.

In a related approach for disentangling content and motion, MoCoGan [78], starts by determining that the content representation z_C should remain roughly the same throughout a video, and therefore the same content vector can be used to generate all of the new frames. On the other hand, an RNN is used to map a sequence of random variables $[\epsilon^1, ..., \epsilon^k]$ into a sequence of motion codes $[z_M^1, ..., z_M^k]$, which along with z_C are given as input to a time agnostic decoder, that outputs each frame $\hat{x}^t$ from the inputs $\{z_C, z_M^t\}$. To force the separation of information sources, a Generative Adversarial Network (GAN) like approach is adopted, in which a frame-level discriminator tries to determine whether a frame is sampled from a real video or is generated, and a video-level discriminator is trained to determine if a full video sequence is real or generated. Training the whole architecture in an adversarial game in which the generative model tries to fool the discriminators, pushes the former to produce realistic looking frames while at the same time its only sub-module with access to the temporal dimension - the RNN - is incentivized to learn motion sequences that reflect realistic dynamics.

In developing the video prediction model proposed in this thesis, it was important to choose a baseline model to build off of. Despite the good results that both MoCoGan and MCnet present, both models have characteristics that could prove disadvantageous. On the one hand MoCoGan's architecture is fairly complex, making use of a generative model and two discriminators, which are typically difficult to train in an adversarial approach. Furthermore, MoCoGan was designed for video generation, which is a different task than video prediction. On the other hand, despite being more simple, MCnet predicts video by feeding back the most recently predicted frame as input instead of continuously predicting in feature space which, as described in section 2.4.3, can lead to error accumulation, resulting in a faster degradation of the generated frames as predictions go further into the future. For these reasons, the adopted baseline follows the work of Denton and Birodkar [79], which is described in detail in the next section.

4.2 DRNET: a baseline for video prediction

In the same vein as the works described in section 4.1, Denton's approach to disentangled representation based video prediction [79] leverages the sequential coherence in videos to isolate content and pose (motion) information. Their proposed model Disentangled-Representation Net (DRNET) is a solution that is both elegant and capable of producing realistic looking frames hundreds of time steps into the future, as it takes advantage of the separate representations for data efficiency and makes predictions in feature space.

As was the case in MCnet, DRNET's architecture uses two separate encoding streams for content and pose. The first important insight is that from the content representation $\boldsymbol{h}_c$ of frame $\boldsymbol{x}^t$ and the pose representation $\boldsymbol{h}_p$ of a future frame $\boldsymbol{x}^{t+k}$ it should be possible to generate the corresponding future frame $\boldsymbol{x}^{t+k}$. The most important observation, however, is what ultimately allows the separation between content and pose. Denton notices that:

1. Content, *i.e.* the appearance of the objects and background, remains broadly the same throughout a given video sequence, even though it may change from video to video;
2. A pose representation, *i.e.* the position of objects and how they move, changes during the course of a video, but the way it is encoded should be the same from video to video.

This means that identity information captured by $\boldsymbol{h}_c$ at time t should be roughly the same as the information captured at time $t+k$, since $\boldsymbol{h}_c$ should carry no time varying information. On the other hand, because $\boldsymbol{h}_p$ is ideally a generic representation of pose and agnostic to content information, it should be indistinguishable between different videos. These constraints are imposed through different terms in the objective function.

The first term imposes that, as mentioned above, the model must be able to reconstruct a future frame at time $t+k$ from the content representation at time t and a pose representation at time $t+k$:

$$\mathcal{L}_{reconstruction}(E_c, E_p, D) = ||D(\boldsymbol{h}_c^t, \boldsymbol{h}_p^{t+k}) - \boldsymbol{x}^{t+k}||_2^2, \tag{4.4}$$

where $\boldsymbol{h}_c$ and $\boldsymbol{h}_p$ are respectively a function of the content encoder E_c and the pose encoder E_p and D is the decoder that maps low dimensional representations to predicted frames. This forces the pose representation to contain the necessary information for the predicted frame to reflect the content transformations that occur between time step t and $t+k$.

To implement the constraint that the content representation should be time invariant, a second term is added, penalizing deviations between content representations obtained with different frames of the same video:

$$\mathcal{L}_{similarity}(E_c) = ||E_c(\boldsymbol{x}^t) - E_c(\boldsymbol{x}^{t+k})||_2^2. \tag{4.5}$$

The third part of the loss function establishes the constraint that pose vectors should be indistinguishable between videos and independent of object identity. For that, the pose encoder E_p competes with a discriminator $Disc$ in minimizing their respective adversarial losses. At each training iteration the discriminator receives pairs of pose representations that may belong to the same video at different time steps $\{\boldsymbol{h}_{p,i}^t, \boldsymbol{h}_{p,i}^{t+k}\}$ or may have been extracted from different videos $\{\boldsymbol{h}_{p,i}^t, \boldsymbol{h}_{p,j}^{t+k}\}$. For each pair, the discriminator must make a decision on the likelihood of the representations belonging to the same video or not, using a cross-entropy loss:

$$-\mathcal{L}_{adversarial}(Disc) = log(Disc(E_p(\boldsymbol{x}_i^t), E_p(\boldsymbol{x}_i^{t+k}))) + log(1 - Disc(E_p(\boldsymbol{x}_i^t), E_p(\boldsymbol{x}_j^{t+k}))) \tag{4.6}$$

At the same time, the pose encoder E_p optimizes its part of the adversarial loss by trying to prevent the discriminator from correctly classifying pose representation pairs. To win the adversarial game,

the pose encoder must learn how to remove all video specific information from the representation while producing an $\boldsymbol{h}_p$ vector that's generic enough that it could represent the pose for different video sequences. As before, this is achieved with a cross-entropy loss:

$$-\mathcal{L}_{adversarial}(E_p) = \frac{1}{2}log(Disc(E_p(\boldsymbol{x}_i^t), E_p(\boldsymbol{x}_i^{t+k}))) + \frac{1}{2}log(1 - Disc(E_p(\boldsymbol{x}_i^t), E_p(\boldsymbol{x}_i^{t+k}))) \quad (4.7)$$

which has its minimum when the Discriminator outputs 0.5 for any pair of frames, which corresponds to maximum entropy.

The final training objective for learning the disentangled content and pose representations is:

$$\mathcal{L} = \mathcal{L}_{reconstruction}(E_c, E_p, D) + \alpha\mathcal{L}_{similarity}(E_c) + \beta(\mathcal{L}_{adversarial}(E_p) + \mathcal{L}_{adversarial}(Disc)). \quad (4.8)$$

The model described up to this point learns disentangled representations, but is not capable of predicting future video frames as it has no perception of time. For that, an LSTM network is added. As described in Fig. 4.3, the LSTM takes as input two concatenated content and pose vectors. Because the content representation is assumed to have few changes over time, the same $\boldsymbol{h}_c$ vector is reused for every time step, while the pose vector is updated. At every step, the LSTM outputs a pose vector one time step ahead of the one it received as input. This way, after a few context steps for initialization of the LSTMs internal state, the predicted $\boldsymbol{h}_p$ vectors can be fed back as input, allowing the *hallucination* of future vectors and prediction in feature space (see Fig. 4.3). Finally, at each time step, the initial content vector along with the corresponding pose vector are given as input to the decoder, which generates the predicted frame. The LSTM is trained in a separate stage than the rest of the model, with its objective being the ℓ_2 loss between predicted pose vectors $\tilde{\boldsymbol{h}}_p^{t+k}$ and pose vectors coming from the pose encoder $\boldsymbol{h}_p^{t+k}$.

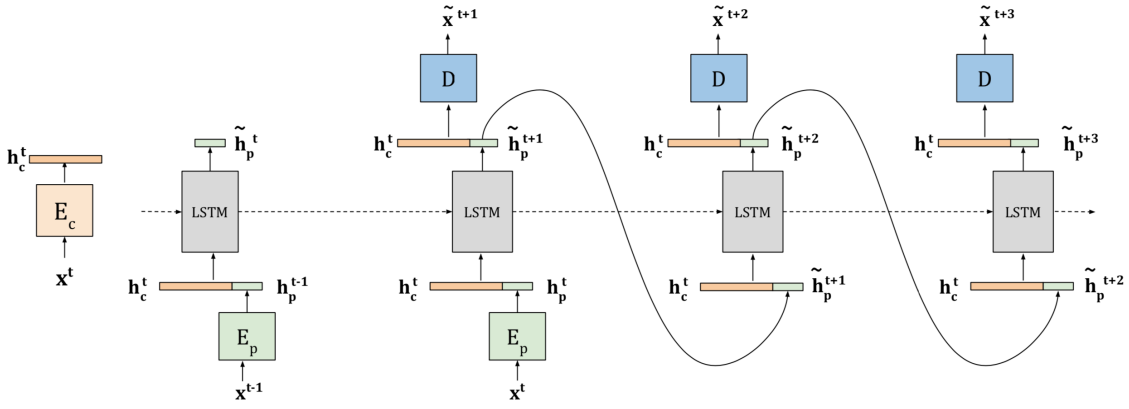


Figure 4.3: Video prediction with DRNET. A single content vector is obtained and reused for every time step. After some observed frames are given as context for initialization of the LSTM's internal state, pose vectors are fed back as input for prediction into the future. Source [79].

By being able to reuse the same content vector and predicting only the smaller pose representation, DRNET assures it maintains data efficiency. Also, predicting in feature space allows every frame to be decoded in the same conditions, preventing errors from compounding and allowing predictions to go further into the future. These characteristics of DRNET's architecture reflect themselves in the results presented in Fig. 4.4.

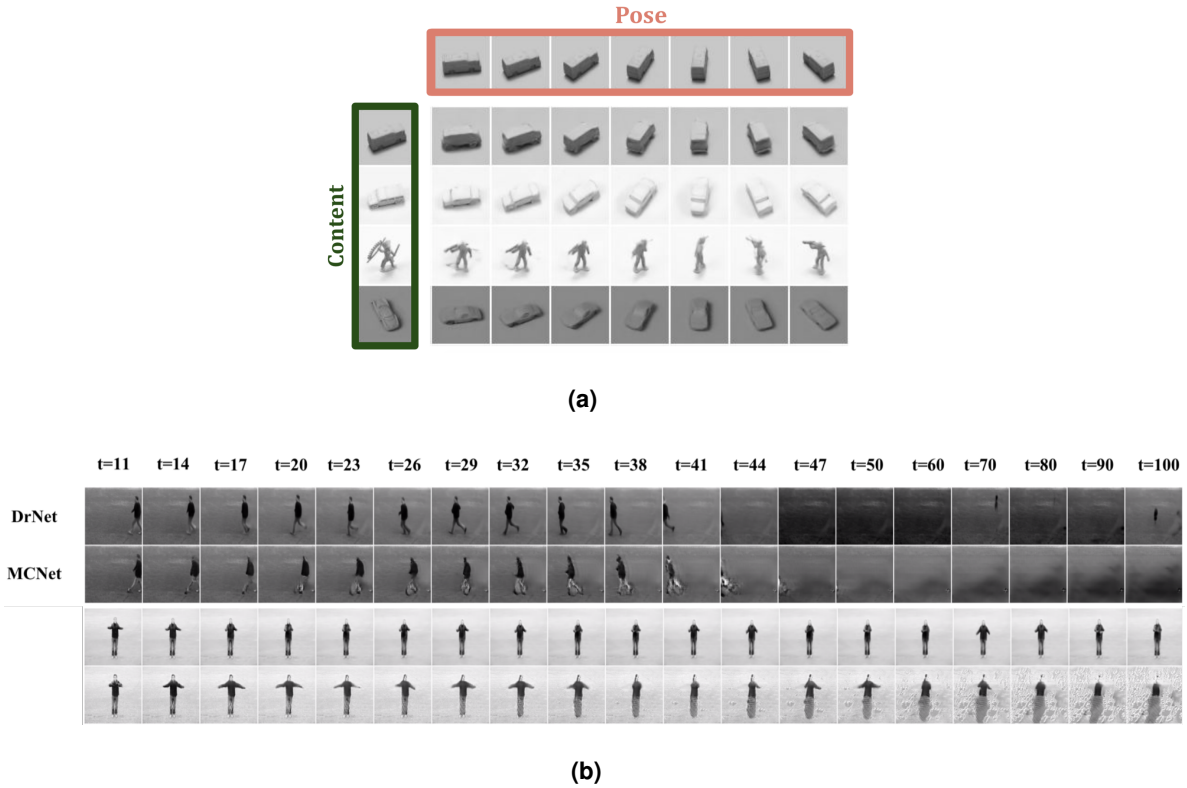


Figure 4.4: Results obtained with DRNET. (a) Demonstration of content and pose disentanglement. Each image in the grid was generated using the corresponding representation on the content column and pose row, allowing to produce images of different objects in new poses. (b) Video prediction with DRNET on the KTH dataset and MCnet baseline (discussed in section 4.1). While DRNET is able to keep producing sharp images for long periods, errors in MCnet quickly compound to produce blurry images. Note: better seen online at https://web.ist.utl.pt/ist181063/vp_examples/drnet_results/. Source [79]).

4.3 ADR: Action-conditioned Disentangled Representations

Having established the importance of disentangling representations for efficient video prediction and the far reaching impact of properly taking advantage of structural biases in the data, a final, pivotal piece of information can now be used for benefit of the learned representations and predictions and that is the commanded actions in a robotic setting. Unlike the video prediction problems seen in the previous section, robotics is fundamentally an action oriented field. Actions are, as discussed in section 3.1 and studied under the concept of affordances [80], what connects agents with the entities of their environment and what allows them to learn how to interact with them to pursue desirable states. In video prediction of robotic settings, actions represent a source of information that is directly controlled and that guides the temporal evolution of the remaining sources of information that generate the video. Furthermore, if developments in robotics are to take a page off of human behaviour, then the inextricable relationship between action, perception and prediction discussed in sections 2.3 and 3.1 is, in my opinion, a research direction that may bridge the gap between humans and robots.

Backing up this belief are some of the most common - and often overlooked - characteristics of human perception such as the reason why humans can't tickle themselves. Having studied this question, Blakemore, Wolpert and Frith [81] suggest that sensory inputs that can be precisely predicted are suppressed and therefore difficult to discern, an idea that is in line with Predictive Coding theory. In the specific case of tickling, the self induced movements of the hands - to which our brain has direct access - can be very precisely predicted, removing all the surprise and inhibiting the corresponding sensations [34]. Having someone else do the tickling - or even tickling ourselves with a feather - introduces the necessary unpredictability in the movements for the sensation to be felt. If this idea is correct, the human brain appears to disentangle its own, self generated movement - which is easy to predict - from the movement of external factors, such as objects and other agents. This way, information can be selected, making external factors more salient and easier to model. Similar phenomenons in nature have been the subject of considerable research, with particular focus on some species of fish capable of sensing and generating electric fields. Mormyrid fish, also called elephantfish, rely on this ability for navigation, communication and for sensing small electric fields generated by invertebrates, allowing the detection of prey in low visibility waters. Doing this, however, requires the distinction of the self generated, high amplitude pulses used for navigation and communication from the low amplitude signals associated with prey. When looking for prey, the former are thus suppressed by internally generated predictions of the outcomes of self behavior, allowing the sensori processing to be fully dedicated to external inputs [82].

These observations and studies, along with the active prediction theories of human perception, reveal the potential benefits of shifting attention from predictable or controllable aspects of the environment to the external and more uncertain sensori queues. It raises the question of whether they can be leveraged in robotic tasks, leading to the main motivation behind the video prediction model proposed in this section. This, is based on learning a mapping between commanded actions and the expected

visual input of an agent's own pose, a problem that is the reverse of previous studies on learning by imitation [83] where, after translating an observed gesture to one's self perspective, the visual perception of the gesture is mapped to a motor command. In our approach, the obtained self pose can then be suppressed, in order to allow better modelling of external signals such as object movement. Despite the existence of previous work on action-conditioned video prediction, to the best of our knowledge no previous work has explored actions for disentangling sources of information.

4.3.1 ADR-AO: Disentangling agent movement

Building off of DRNET's architecture and the previously presented idea that self induced movement can be well predicted and should be suppressed, we ask the question:

Can the movement of a robot be reconstructed up to time step $t+k$ *from the content information observed at time step* t *and the actions commanded to the robot between* t *and* $t+k$*?*

Solving this challenge requires a model that is capable of both identifying which parts of an image are related to the robot and of transforming that information according to the specified action. As in Denton's DRNET, we start by setting the constraint that, for the most part, the content of a video remains the same during the same video sequence, but can change from video to video. Hence, we adopt DRNET's similarity loss term, that encourages the content representation $\boldsymbol{h}_c$ to be the same when built from frames of the same video, which has the effect of making it time-invariant:

$$\mathcal{L}_{similarity}(E_c) = ||E_c(\boldsymbol{x}^{t-c:t}) - E_c(\boldsymbol{x}^{t+m-c:t+m})||_2^2, \tag{4.9}$$

where E_c represents the content encoder, $\boldsymbol{x}^t$ the video frame at time step t and m is a time gap chosen at random. We consider time step t the present moment while $t-c$ is the first frame in the video and with $c+1$ being the number of past frames given to the content encoder as context.

While movement in DRNET was modeled by the pose representation $\boldsymbol{h}_p$, which required an adversarial loss term to remove any content information, we have a simplified task at this stage because we're only concerned with the agent's own movement, which is determined by the commanded actions $\boldsymbol{a} = [\boldsymbol{a}^0, ..., \boldsymbol{a}^t, ..., \boldsymbol{a}^{t+k}]$, with $\boldsymbol{a}^t \in \mathbb{R}^{|\boldsymbol{a}|}$. Using actions has the benefit of content information being separated from movement information by default. This contrasts with DRNET's pose vector, that is built from an observed frame that contains both content and pose information, therefore requiring separation. So, to answer the initial question only one more loss term is needed, imposing that the model should be able to reconstruct the future frame $\boldsymbol{x}^{t+k}$ from the content representation at time t and the action representation $\boldsymbol{h}_a$ at time $t+k$:

$$\mathcal{L}_{reconstruction}(E_c, A, D) = ||D(\boldsymbol{h}_c^{t-c:t}, \boldsymbol{h}_a^{t+k}) - \boldsymbol{x}^{t+k}||_2^2, \tag{4.10}$$

with k being another random time gap, $\boldsymbol{h}_a^{t+k}$ being the output of the action encoder $\boldsymbol{h}_a^{t+k} = A(\boldsymbol{a}^{t-c:t+k})$ and D being the decoder that generates the predicted frame $\hat{\boldsymbol{x}}^{t+k}$ from the content and action representations.

The complete training objective is then

$$\mathcal{L} = \mathcal{L}_{reconstruction}(E_c, A, D) + \alpha \mathcal{L}_{similarity}(E_c) \tag{4.11}$$

and the model architecture, which we refer to as Action-conditioned Disentangled Representations - Agent Only (ADR-AO), is summarized in Fig. 4.5. As described in section 5.1.1, the action representation for BAIR dataset has dimensionality $|a| = 7$ and is the concatenation of the commanded Cartesian position of the robot's gripper and the commanded joint angles, including an extra value for the gripper state (open or closed). Despite its extreme simplicity, this autoencoder model succeeds both at disentangling self movement from content information and at reconstructing the future pose of the robotic arm only from the appearance at an initial time t and the future actions, as demonstrated in the results section 5.3.

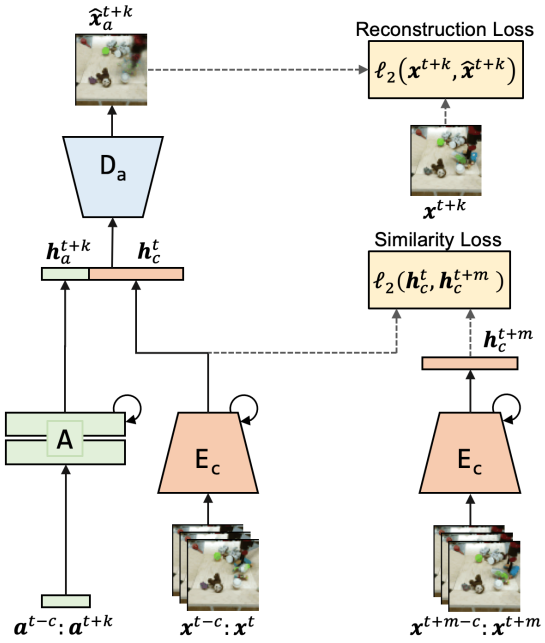


Figure 4.5: Architecture of ADR-AO model.

Stochastic version

As discussed in section 2.4.2, real world environments are characterized by complex dynamics that often depend on unobservable and unpredictable factors. When a limited amount of inputs is given to the model to encourage disentanglement, as is the case of the model presented in the previous section, this effect can be intensified. Under these conditions, a model that is trying to minimize reconstruction error using an ℓ_2 objective can end up producing an average of the different possible outcomes, instead of predicting the most likely scenario. A patent example observed in the model of Fig. 4.5 is concerned with the difficulty of perceiving depth from a single $2D$ observation of the scene. Even though the trajectory and scale of the robotic arm can be well predicted from the actions when it moves in the depth dimension, it is difficult for the model to perceive whether the objects on the table will occlude the arm or the other way around, an effect that is illustrated in the middle row of Fig. 4.6.

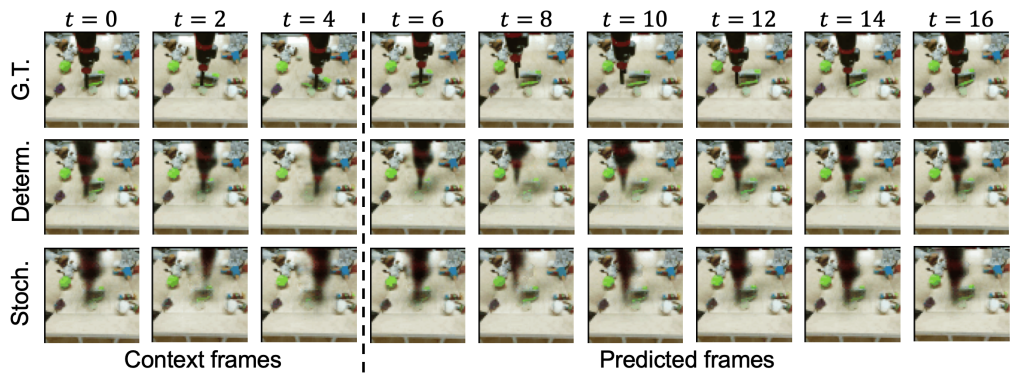


Figure 4.6: An example of the importance of stochasticity for modelling unobservable aspects of the world: from the $2D$ image the model cannot perceive depth. To minimize the error of predicting whether the arm passes in front or behind the green stapler, the deterministic model of Fig. 4.5 averages the two modes, resulting in a blurry prediction in which the stapler momentarily disappears. On the other hand, the stochastic model of Fig. 4.7 can make a decision on the most likely mode, placing the arm in front of the stapler and preserving most of the stapler's shape. Note: better seen online at https://web.ist.utl.pt/ist181063/vp_examples/stochasticity_experiment/.

To solve this issue, we extend the model to learn a set of latent variables $\boldsymbol{z}_a$ that the model can associate with the unobservable factors, allowing it to decide on one of the possible outcomes for these factors. This is achieved with the help of a distribution $q_\phi(\boldsymbol{z}_a^t|\boldsymbol{a}^{0:t+k}, \boldsymbol{x}^t)$, that is trained as an inference network both with a reconstruction term and with the regularizing constraint that it should remain close to a prior distribution $p(\boldsymbol{z}_a)$ chosen to be a Gaussian $\mathcal{N}(0, \boldsymbol{I})$, as in the VAE objective. The final extended objective function becomes

$$\mathcal{L} = \mathcal{L}_{reconstruction}(E_c, A, D) + \alpha\mathcal{L}_{similarity}(E_c) + \beta \mathrm{D}_{KL}(q_\phi(\boldsymbol{z}_a|\boldsymbol{a}^{0:t+k})||p(\boldsymbol{z}_a)) \tag{4.12}$$

To condition the inference network on the sequence of executed actions, we use an LSTM that at each time step receives the content representation $\boldsymbol{h}_c$ of an initial frame concatenated with the corresponding time step's action representation $\boldsymbol{h}_a^{t+k}$. The LSTM outputs the parameters of a multivariate Gaussian distribution from which $\boldsymbol{z}_a$ is sampled: $\boldsymbol{z}_a^{t+k} \sim \mathcal{N}(\boldsymbol{\mu}_\phi^{t+k}, \boldsymbol{\sigma}_\phi^{t+k})$. Finally the decoder reconstructs each frame $\hat{\boldsymbol{x}}^{\boldsymbol{t+k}}$ from the hidden vector composed of the concatenation of $[\boldsymbol{h}_c^t, \boldsymbol{h}_a^{t+k}, \boldsymbol{z}_a^{t+k}]$. This extended architecture is summarized in Fig. 4.7.

It is worth noting that unlike common practice in stochastic video prediction models [47] [50], where at test time the latent variables must be sampled from the prior $p(\boldsymbol{z}_a)$, in our model the approximate posterior distribution $q_\phi(\boldsymbol{z}_a^t|\boldsymbol{a}^{0:t+k}, \boldsymbol{x}^t)$ can be used at test time. This is because other models are usually conditioned on future frames, to which there is no access at test time while in our case both the most recently observed frame $\boldsymbol{x}^t$ and the future commanded actions on which the approximate posterior is conditioned are known. This allows our model to use the latent variables to model hidden factors of variation without creating random futures that diverge from the truth, as is the case in [50].

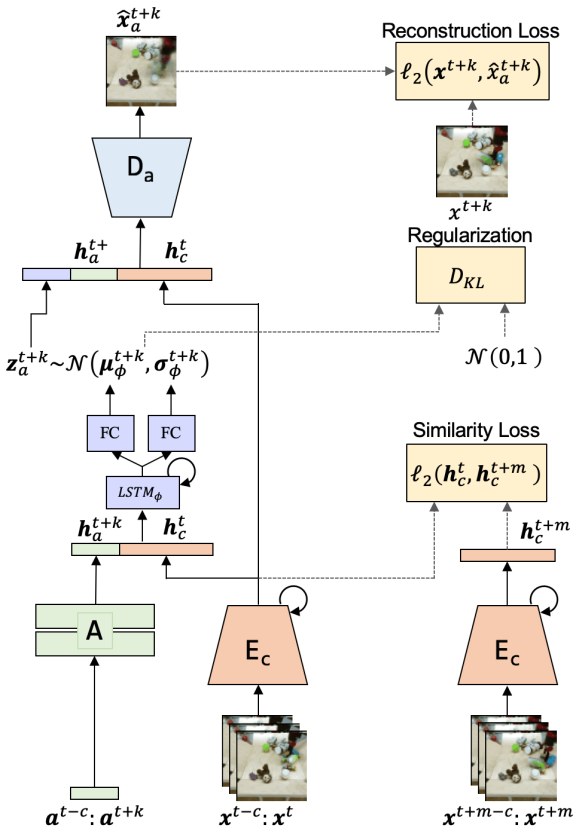


Figure 4.7: Stochastic ADR-AO.

Architecture and training details

The content encoder is a 5 layer CNN with kernel size 4 and stride 2, which reduces the size of the feature map by two at each layer. The first layer has 64 filters, and each layer has double the filters of the previous one with exception of the last layer which has $|\boldsymbol{h}_c|$ filters. This means that after the first convolutional layer the $64 \times 64 \times 3$ image is compressed to a $32 \times 32 \times 64$ feature map and after the last layer a $1 \times 1 \times |\boldsymbol{h}_c|$ encoding is obtained. At training time, a fixed number of context frames is used to condition the content encoder. To extract temporal information from these frames, the 3^{rd} and 5^{th} layers are ConvLSTM layers. The option for a recurrent content encoder is due to the fact that big objects in the scene can occlude smaller objects, which won't be seen if a single frame is given as input. Once the bigger object moves, this causes the model to focus on learning to inpaint the previously occluded background which besides not being the intention, gives rise to blurry patches in the reconstructed images. This differs from the architecture of DRNET that always conditions the content encoder on a single frame, however the authors test the model on datasets with simple or plain backgrounds like KTH and moving MNIST, where it is easy for the model to inpaint occluded portions of the background.

The decoder network is a mirrored version of the content encoder without the recurrent layers and with common convolutions replaced by transpose convolutions that double the size of the feature map. The size of the kernels and stride remain at 4 and 2, respectively, which prevents checkered artifacts

in the output image, as described in [84]. As in DRNET, we add skip connections from E_c to D in a UNet [85] fashion, which allows better gradient flow in the backward pass and facilitates learning. Finally, the action encoder is a simple, three layer neural network and the LSTM has two layers with 256 units, a linear embedding layer at the input and is followed by two parallel fully connected layers for outputting the means and variances that parameterize the multivariate Gaussian distribution.

For experiments with BAIR dataset, we set $|\boldsymbol{h}_c| = 128$, $|\boldsymbol{h}_a| = 16$ and $|\boldsymbol{z}_a| = 10$. It is worth noting that architectural parameters such as the number of layers and different sizes (kernel, representation vectors, units) are application dependent and may require tuning. However, the model is generic to tasks with first person action information, as long as the base modules are preserved.

We train with $\alpha = 1$ and $\beta = 1 \times 10^{-6}$ using the ADAM optimizer [86] and cyclical learning rates [87] between 1×10^{-5} and 8×10^{-5}, which provides better generalization error.

4.3.2 Disentangling object information

In section 4.3, we saw how living organisms can benefit from the ability to suppress easily predictable, self generated sensori inputs to instead focus on external sources of sensation, with the examples of self tickling and of some fish species that use electrical signals to detect prey. Analogously, we argue that a video prediction system used in a robotic context should focus on the external, difficult to predict, consequences of the robot's actions, rather than on the foreseeable self movements of the robot. This shift in attention can be made possible with the help of the ADR-AO model proposed in the previous section: to do so, we start by recalling that the ADR-AO is trained to only be aware of the planned motion of the agent, with its predictions showing objects that end up getting displaced in the ground truth video being ignored and left in their original position (see fig. 4.8). This means that the error between ground truth frames $\boldsymbol{x}^t$ and frames generated with ADR-AO $\hat{\boldsymbol{x}}_a^t$, *i.e.*,

$$\boldsymbol{x}_{err}^t = \boldsymbol{x}^t - \hat{\boldsymbol{x}}_a^t, \tag{4.13}$$

produces an image dominated by information of the objects that moved during the video, while the background and the movement of the agent - which have already been predicted - get suppressed, as illustrated in Fig. 4.8. In practice, this cue on the object information can then be leveraged by a model to learn a representation for the objects that ADR-AO fails to perceive, in a mechanism that draws parallelism with the forward propagation of yet unexplained error in Predictive Coding theory. Importantly, this information is obtained in a fully self-supervised way, without the need for data pre-processing or human annotation to obtain object locations and identities, as is common in most state of the art work [88] [89].

ADR: Model Description

Having a separate cue on object information, we can now train an autoencoder to obtain a low dimensional representation of the objects $\boldsymbol{h}_o$. We aim at reconstructing a video frame at time step $t + k$

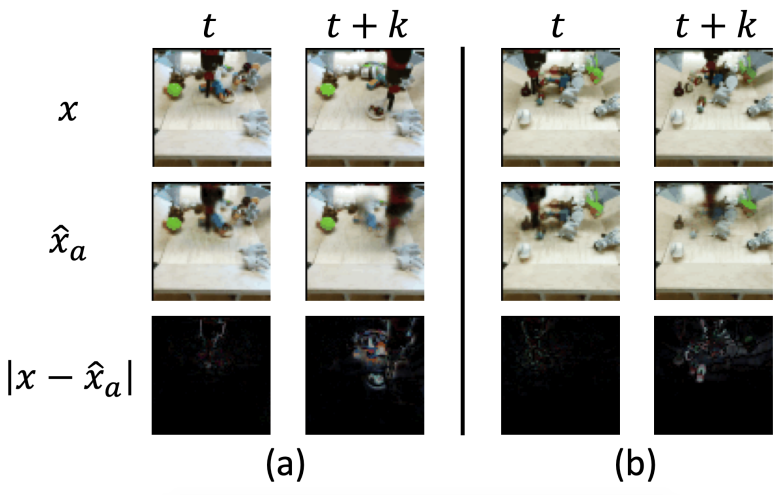


Figure 4.8: Top row: ground truth frames of two different examples of BAIR dataset in which the robotic arm displaces objects. **Middle row**: predictions with ADR-AO, where the movement of the arm is well predicted but the displaced objects are left in their initial position. **Bottom row**: the error between the two rows is dominated by object information (left hand side example), with both the negative and the new object position being visible (right hand side example). Better seen online, with zoom at https://web.ist.utl.pt/ist181063/vp_examples/error_images/.

from the content information of $c+1$ context frames $\boldsymbol{x}_{t-c:t}$, and a sequence of actions $\boldsymbol{a}_{t-c:t+k}$ to be performed by the agent. While the goal might be the same, the approach differs in the task of the autoencoder: whereas for ADR-AO the decoder reconstructs its best approximation of $\boldsymbol{x}_{t+k}$, to learn the feature representation of the objects the new autoencoder is now trained to reconstruct the error $\boldsymbol{x}_{err}$. As such, the future frame $\boldsymbol{x}_{t+k}$ is also needed as an extra input, to obtain the error image. If, ideally, the reconstruction $\hat{\boldsymbol{x}}_{err}$ is sufficiently close to the original error image, the future frame $\hat{\boldsymbol{x}}_{pred}^{t+k}$ can be obtained as

$$\hat{\boldsymbol{x}}_{pred}^{t+k} = \hat{\boldsymbol{x}}_{a}^{t+k} + \hat{\boldsymbol{x}}_{err}^{t+k}. \tag{4.14}$$

In practice, however, we verify that better results are obtained if $\boldsymbol{x}_{err}$ is split into its positive and negative components

$$\boldsymbol{x}_{err} = \boldsymbol{x}_{err+} - \boldsymbol{x}_{err-} = max\{0, \boldsymbol{x}^t - \hat{\boldsymbol{x}}_a^t\} - max\{0, \hat{\boldsymbol{x}}_a^t - \boldsymbol{x}^t\}, \tag{4.15}$$

with each one being reconstructed separately, as represented in Fig. 4.9.

Because at this stage ADR-AO is considered to have been pre-trained, we reuse its action A and content E_c encoders to obtain low dimensional feature representations of the actions $\boldsymbol{h}_a$ and of the content $\boldsymbol{h}_c$, which aid the decoder D_o in the reconstruction of the error image. Yet, the trainable parameters of both E_c and A are frozen, and only E_o and D_o are trained. This allows content representations to remain agnostic to temporal features, as previously imposed by the similarity loss term of ADR-AO, while action representations carry no information about the appearance of the scene, since they are obtained solely with the actions $\boldsymbol{a}_{t-c:t+n}$ as input.

The model of Fig. 4.9(a) - which we call Action-conditioned Disentangled Representations (ADR) - is trained with three different reconstruction loss terms, as illustrated in Fig. 4.9(b). The first term determines that the obtained frame $\hat{\boldsymbol{x}}_{pred}$ should be as close as possible to the corresponding ground

truth frame, in terms of ℓ_2 distance. This forces the encoder E_o to generate $\boldsymbol{h}_o$ vectors that are a good summary of the information present in $\boldsymbol{x}_{err+}$ and $\boldsymbol{x}_{err-}$ and that allow the decoder to produce a good recovery of the input. The second and third loss terms are concerned with the positive and negative components of the error reconstruction and encourage their ℓ_2 distance to the input error images to be as small as possible. The need for these two extra loss terms was verified empirically, as they encourage sparsity in the decoder's output and provide an additional cue for the reconstruction, preventing the model from overfitting to the training data. The complete loss function is presented in equation (4.16):

$$\mathcal{L}(E_o, D_o) = ||\boldsymbol{x}^{t+k} - \hat{\boldsymbol{x}}^{t+k}_{pred})||_2^2 + \alpha(||\boldsymbol{x}^{t+k}_{err+} - \hat{\boldsymbol{x}}^{t+k}_{err+})||_2^2 + ||\boldsymbol{x}^{t+k}_{err-} - \hat{\boldsymbol{x}}^{t+k}_{err-})||_2^2) \tag{4.16}$$

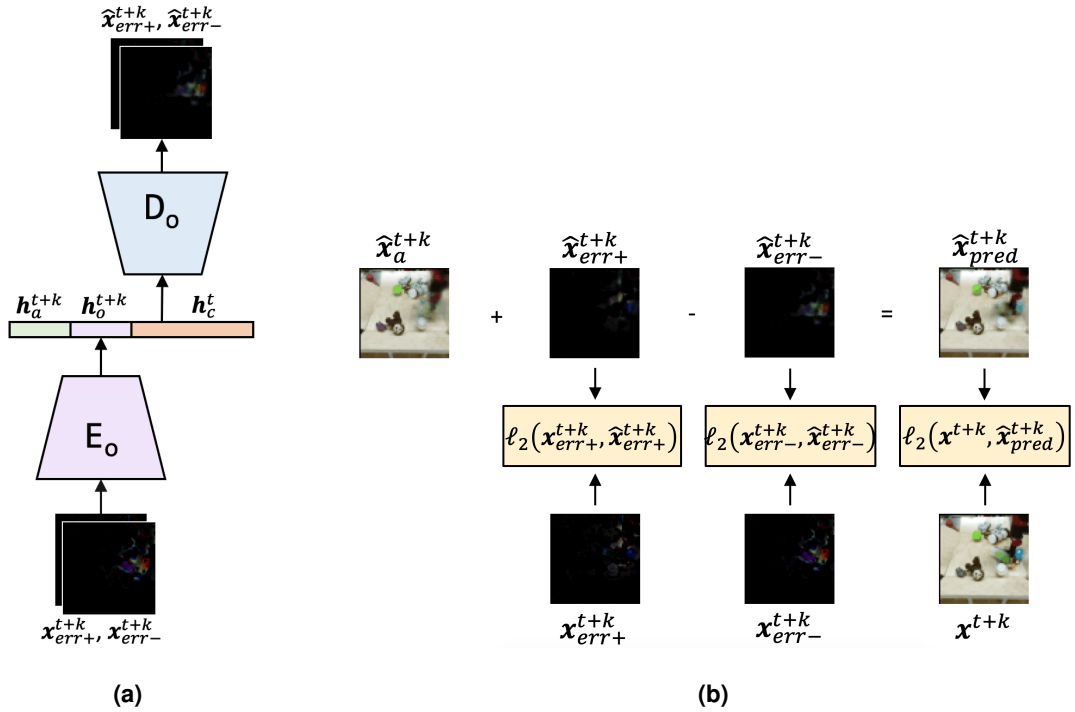


Figure 4.9: (a) ADR architecture. The $\boldsymbol{h}_a$ and $\boldsymbol{h}_c$ vectors are obtained from the pre-trained action and content encoders of ADR-AO, whose weights are frozen and that are omitted for simplicity. (b) Procedure to obtain the final reconstructed frame from the error images, along with the loss terms used for optimizing ADR.

Architecture details

To implement the autoencoder described in the previous section an architecture similar to the one used in ADR-AO was chosen, starting with both the trained action and content encoders being directly reused. Similarly to E_c, the object encoder E_o is a 5 layer CNN with kernel size 4, stride 2 and zero padding, which again reduces the size of the feature map in half at each layer. As before, the first layer uses 64 convolutional filters, a number that is doubled at each layer, with exception of the output layer, where it corresponds to $|\boldsymbol{h}_o|$. Each layer of the object encoder has a Leaky ReLU [90] activation preceded by a Batch Normalization [91] operation that accelerates the training of deep models while also providing regularization. In contrast to E_c, the object encoder has no recurrent layers, meaning

its input concerns only one temporal step. Another input related difference is the fact that E_o receives two images concatenated in the channels dimension, one for the positive error and another for the negative. Similarly, the output of the decoder D_o has now 6 channels, split into positive and negative error. As before, it is a mirrored version of the object encoder and receives skip connections at each layer coming from E_c, in the style of UNet.

For the BAIR dataset, we choose $|\boldsymbol{h}_o| = 128$ and the weighting parameter of the loss function $\alpha = 1$. As before, the network is trained using the ADAM optimizer and cycling learning rates, this time between 2×10^{-5} and 8×10^{-5}.

Training details

The training of the ADR model was significantly more challenging than the training of ADR-AO, particularly in terms of obtaining a good generalization error. This starts with the option for training the models separately with ADR building on top of the pre-trained E_c and A networks. This has the implication of the generalization error of ADR-AO being carried over to the new stage of training. It is therefore important, when training ADR-AO, to stop as soon as the training and validation error start to diverge.

Even though the verification of this condition allows the training of ADR to start on favorable terms, the model still shows some propensity to overfit. As such, a series of steps were taken to provide additional regularization. The decision that had the most impact on the generalization error was the modification of an original architecture in which the model outputted a single error image for both the positive and negative components and was only optimized to reconstruct the complete frame, according to equation 4.14. Instead, by splitting the reconstruction error into positive and negative components and optimizing each one separately, the model is given additional guidance on its task while preserving a requirement for sparse outputs in which most pixels are 0 (black), resulting in increased regularization.

Additionally, the best strategy for regularizing deep neural networks is to train on as much data as possible [9]. It is therefore common practice to augment the number of examples in a dataset by applying translations and rotations to images or injecting some noise. In our particular application, a simple way of increasing the number of training examples the model is confronted with is to select the context frames randomly within each video sequence in the dataset, instead of always using frames at the beginning of the sequence. This is useful because the tendency is for the norm of the error images to be at its minimum for the most recently observed frame and to increase with the temporal distance the context frame, as more objects get displaced. This is true both if, for example, 5 frames are observed and the future 10 frames are predicted, and if 5 frames are observed and the previous 10 are predicted. Hence, placing the context frames at random time steps produces greater diversity in the error images the model is trained to reconstruct and helps with its regularization. Another method used to regularize the model is the training with cyclical learning rates while experiments with weight

decay were also made, though they did not improve the generalization error.

A second challenge in training ADR is to reduce the training loss as much as possible. The most impactful hyperparameter in the training loss is, naturally, the size of the hidden representation $|\boldsymbol{h}_o|$. The bigger the size, the lower the reconstruction error but also the more relaxed is the autoencoder's constraint that forces the model to find meaningful factors of variation to describe object data. A compromise must therefore be made which, for our problem, is settled at $|\boldsymbol{h}_o| = 128$. To further reduce the reconstruction error, different options were tried as input for E_o: simply inputing the frame to be reconstructed $\boldsymbol{x}^{t+k}$, inputting the difference between $\boldsymbol{x}^{t+k}$ and $\hat{\boldsymbol{x}}_a^{t+k}$, as well the two concatenated in the channels dimension and, finally, inputting the difference image separated in positive and negative components. The choice for the latter option was purely based on it being the solution that offered the best results.

Variations and Discussion

Even though, as presented in section 5.3, the proposed ADR architecture showed the ability to model the residual information coming from ADR-AO, there are some relevant considerations left. Even though the proposed goal is to separate self controlled information from information concerning external objects, this cannot, in essence, be perfectly achieved. On the one hand, ADR-AO is optimized to reconstruct the whole frame, which includes information of objects that may have moved. This means that inevitably it will have some understanding - even if very limited - of object information. On the other hand, and most importantly, objects do not move independently of an agent's own movement. Therefore, there is an inherent correlation between information of moving objects and the agent's action. Furthermore, because ADR-AO doesn't perfectly predict the movement of the agent, there is some signal related to self information that is passed in the error image to the object autoencoder (as can be seen by the contour of the robotic arm in Fig. 4.8).

To mitigate this problem we decide to provide the action representation $\boldsymbol{h}_a$ to the object decoder (as seen in Fig. 4.9(a)). The idea is that it may allow D_o to use $\boldsymbol{h}_a$ to reconstruct the part of the error related to the agent while leaving $\boldsymbol{h}_o$ available for modelling the objects. To encourage further disentanglement between self and object information an additional adversarial method was experimented. Inspired by the adversarial loss term of DRNET, a discriminator receives a pair of representations with 50% chance of coming from the same video, at the same time step and 50% chance of coming from different videos at the same time step. This pair of representations is composed by one object representation and one action representation and the discriminator is trained to determine whether the pair belongs to the same video or not. The generative model, in this case E_o is then trained to output $\boldsymbol{h}_o$ vectors that prevent the discriminator from correctly classifying the representation pairs. Ideally, this would encourage E_o to produce object representations that have no action information (and therefore cannot be associated with either an action of the same video or an action of a different video). Despite some experiments using this adversarial game as an extra loss term for training the object model, its

impact on disentanglement is hard to quantify (for reasons described in chapter 5). Furthermore, the fact that an extra constraint on $\boldsymbol{h}_o$ leads to a worse reconstruction error lead to the decision not to use this solution in the final model architecture, even though future experiments a possible direction in future work. Lastly, and similarly to what was described for ADR-AO, modelling unobservable factors of variation in the data should be an important addition to the object autoencoder.

4.3.3 ADR-VP: Disentangled video prediction

In the two previous sections we proposed a model capable of learning separate representations for the agent's actions and for the movement of the objects. While ADR-AO only requires knowledge of the ground truth frames at training time for computing the loss function, the complete model, ADR, also makes use of the ground truth for obtaining the error images that serve as input to the E_o module. While this poses no problem when the task is limited to reconstruction, in a practical video prediction problem the ground truth frames would not be available, meaning that only ADR-AO could be directly used for video prediction, with the caveat that it does not consider object information.

To predict video using ADR, a new module that can acquire knowledge of the temporal evolution of the object representations $\boldsymbol{h}_o$ needs to be added. For that, we opt for the use of an LSTM, as illustrated in the diagram of Fig. 4.10. At each time step t, the LSTM receives a complete representation of the current frame, which is the concatenation of the content, action and object representations, and is asked to output the object representation for the next time step $(t+1)$. During the temporal period corresponding to the context frames (two time steps in Fig. 4.10), the $\boldsymbol{h}_o$ vector received by the LSTM can be obtained from the observed frames, allowing the initialization of the hidden state. However, once the model runs out of context frames, the predicted vectors $\hat{\boldsymbol{h}}_o$ must be fed back to the model, effectively allowing it to unroll imagined futures. Finally, to obtain the predicted error images that allow the construction of $\hat{\boldsymbol{x}}_{pred}$, D_o is used at each time step, as shown in Fig. 4.10. We refer to this updated framework as Action-conditioned Disentangled Representations - Video Prediction (ADR-VP).

An important benefit of having disentangled sources of information is highlighted by this architecture: because content vectors are trained to be constant in time and action representations are obtained from commanded actions to which the model has access (even future ones), the LSTM can focus on only predicting information related to the objects, reducing the complexity of its task. Finally, it is important to note that, to predict the object representation for the next time step, the LSTM should receive both the current and the next action vectors as input. This allows the LSTM to predict the movement of the objects conditioned on a known imminent movement of the agent, whereas otherwise it would have to guess object movement based on an unknown trajectory of the arm.

Training details and Discussion

To train the kind of model described in the previous section, two strategies are typically used: teacher forcing [9] or scheduled sampling [92]. The first approach is characterized by removing the feedback

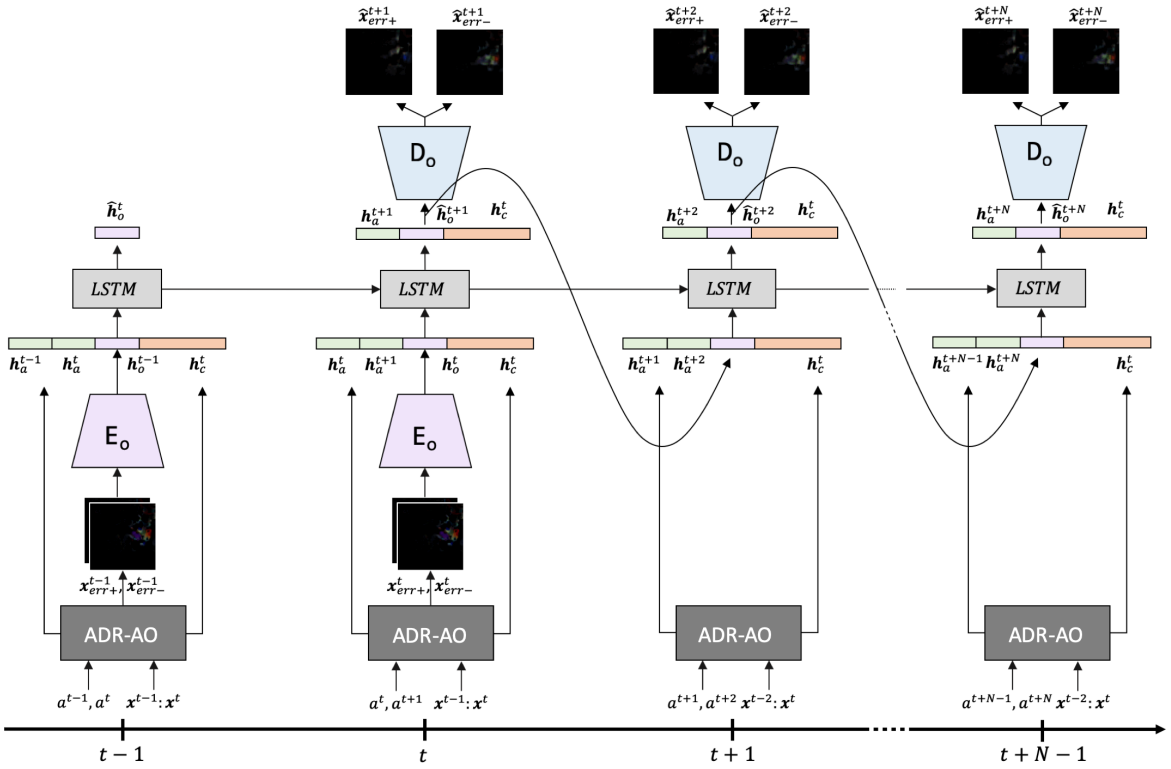


Figure 4.10: Proposed architecture for prediction of future video frames. The context frames are used to initialize the hidden state of the LSTM. When the current time step is reached, the predicted $\hat{\boldsymbol{h}}_o$ vectors are fed back to the model, allowing the imagination of future frames. Importantly, content and action vectors do not need to be predicted as they can be obtained from the context frames and future commanded actions, respectively.

of the LSTMs output, instead using the ground truth frames to generate the input regardless of the time step (as shown if Fig. 4.11). At the same time, the output of the network is optimized to be close to the ground truth vector, which in our application is obtained from E_o. This approach has the benefit of assuring that, during training, error is not accumulated as time advances, assuring a balanced error signal at all steps. At the same time, during backpropagation, the flow of gradient is interrupted at the input of the LSTM, preventing the updates at earlier steps from being dependent on all subsequent time steps. The main disadvantage of teacher forcing lies in the fact that at test time the network's predictions must be used as feedback. If these are not sufficiently close to the structure of $\boldsymbol{h}_o$ vectors seen during training, errors can accumulate and result in poor prediction performance.

In trying to mitigate this challenge, scheduled sampling is a mixed training approach between teacher forcing and plain output feedback. At each time step, the input to the LSTM is randomly chosen between the ground truth vector, and the vector coming from the output of the previous step. In the first training iterations, the training behavior is similar to teacher forcing as the input's probability is heavily skewed towards the ground truth vector. As training progresses, however, the chance of sampling a vector coming from the network steadily increases, allowing the network to progressively adapt to the type of input it will see at test time.

Despite having tested both solutions for training ADR-VP, we were not able to train the model to a point in which it successfully predicts object movement. There may be several factors contributing to this

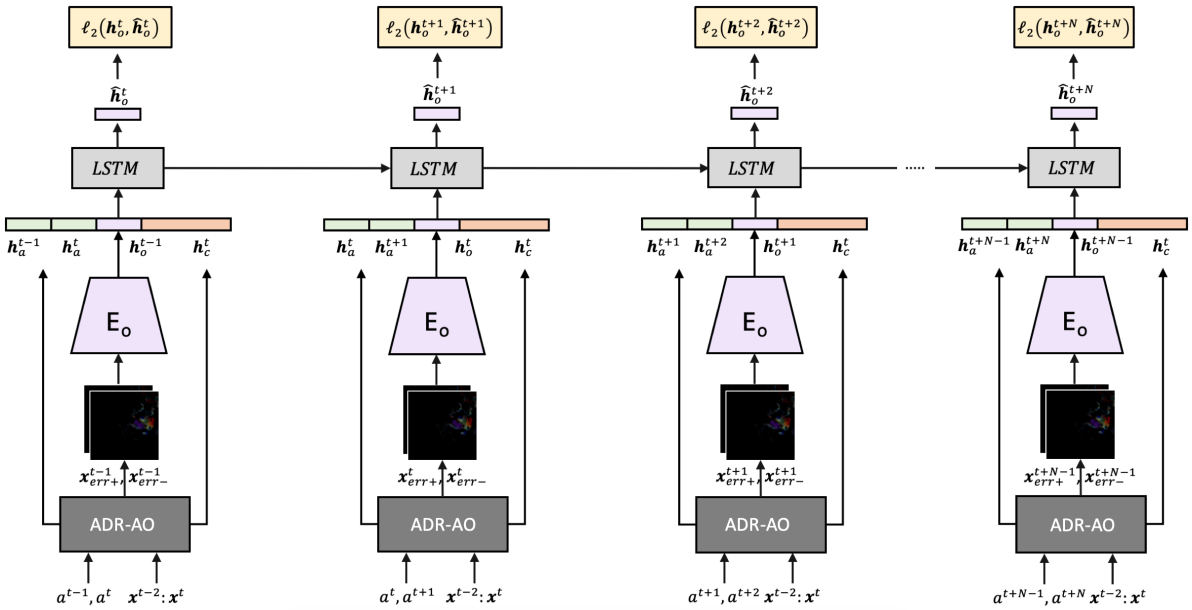


Figure 4.11: Teacher forcing training of ADR-VP, with the loss terms in yellow.

result, in particular the dataset and the architecture. BAIRs robot push dataset has the particularity of the robotic arm moving randomly around the container, which possibly impacts the training of our model. As mentioned in section 5.1.1, the random movement causes video sequences in the dataset to often have no interaction between the robotic arm and the objects contained in the box. Furthermore, even in sequences in which objects are displaced, the movement is typically only seen in a subset of the frames, with other frames showing no object movement. This implies an imbalance in the dataset in terms of the number of frames in which objects move, versus the number of frames in which they are static, possibly leading the network to learn that predicting no movement minimizes the expected loss. In comparison, our chosen baseline, DRNET, which also trains an LSTM for video prediction with teacher forcing, is tested on the KTH dataset (see Fig. 4.4) and succeeds in predicting video frames. This dataset, however, is composed of videos of human actions such as walking and waving the arms up and down, where movement is consistent and present from frame to frame, which possibly avoids the aforementioned issue.

The other effect of the random actions of BAIRs robotic arm is unpredictability. Even though the uncertainty in the arm's trajectory is offset by the knowledge of the actions that will be commanded in the future, this kind of movement may still have some reflection in the consistency of the movement of the objects. In particular, because the arm's trajectory can change from frame to frame, so can the objects' trajectory, cutting of the temporal correlation that would allow the LSTM to produce good predictions.

Still, there have been video prediction models tested on BAIR dataset, such as CDNA, SV2P and Stochastic Video Generation Learned Prior (SVG-LP), described in section 2.4. However, as demonstrated in the results chapter, quantitative and qualitative results obtained with these models show that the only one that successfully predicts object movement is SVG-LP, with the others simply blur-

ring them out. Even so, SVG-LP is a stochastic action-free model, leading most of its predictions to diverge from the actual observed future. In contrast, the fact that ADR-VP is conditioned on the actions, constrains it to only predict futures that are close to the truth, which possibly results in increased difficulty during training.

Finally, the reason for the poor predictive capabilities of ADR-VP may be related with its architecture and the complexity of the the 3 stage training procedure. This not only makes the training process difficult, due to increased run times and number of parameters to be tuned but also increases the chances of error accumulation from one stage to the other. An example of this problem is given in section 5.3.2. Another factor introducing complexity is the dimensionality of $\boldsymbol{h}_o$. While DRNET uses the same scheme to predict $\boldsymbol{h}_p$ vectors of size 24, we set the d$|\boldsymbol{h}_o|$ to 128. This represents a considerable increase in complexity which we try to account for by increasing the number of trainable parameters in the LSTM.

5

Results

Having laid out our approaches for solving the three proposed research questions we now present the experiments that were conducted to test them. This chapter starts with a presentation of the selected datasets, followed by a comparison between different video prediction models, both in the existing metrics and in our novel one. Results for the proposed models for disentanglement are then presented and, lastly, an analysis of their video prediction capabilities is made, in comparison with existing models.

5.1 Datasets

5.1.1 BAIR Dataset

Acquired at the Berkeley Artificial Intelligence Research Lab (BAIR), the BAIR robot push dataset [93] is a benchmark dataset used in most video prediction research work [47] [48] [50] [51] [94]. BAIR's video sequences consist in a Sawyer robotic arm randomly moving and pushing a wide range of objects within a confined table (see Fig. 5.1). These include everyday life objects such as a baseball, a stapler, a teddy bear or a Nutella jar. Each of the 43520 video samples in the dataset has 30 RGB frames of size 64×64 and, despite the table and background being fixed, lighting conditions and the objects often change. Also, frames are obtained from a fixed camera position that corresponds to the first person point of view of the robot.

Besides the video frames, the dataset also provides the actions commanded to the robot at each time step, as well as the gripper state. Commanded actions are represented as a $4D$ vector indicating the joint angles and the gripper position (open or closed). These are updated at half the rate of the video frames, *i.e.*, for every two frames a new action is commanded. Still, in frames where the action is not updated, the previous action is executed, meaning that the gripper is continuously changing its state, which is considered to be the $3D$ Cartesian position.

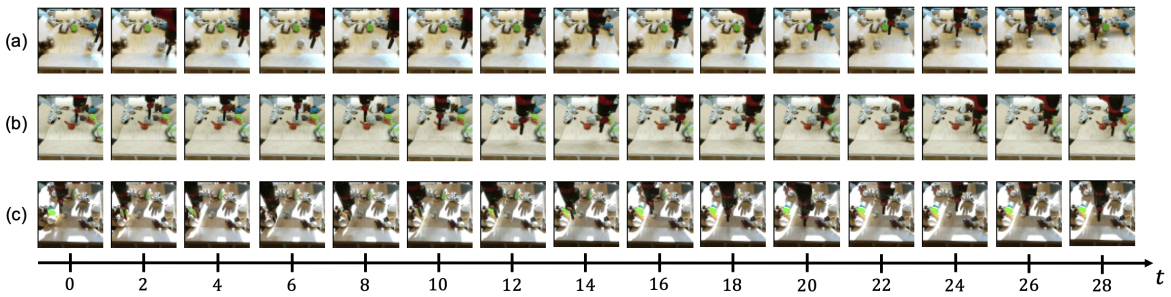


Figure 5.1: Three samples of the BAIR dataset. (a) The robotic arm pushes a baseball from right to left; (b) No object in the box is touched by the arm; (c) In some samples lighting conditions vary. Best seen online with zoom at https://web.ist.utl.pt/ist181063/vp_examples/bair_dataset_examples/

A characteristic of the BAIR dataset is that the robot's actions are randomly selected. This means that a model that is trying to infer or predict the actions won't simply be able to learn their distribution. However, in robotic video prediction tasks, actions are considered an input - even if they are future actions - so they can still useful to guide predictions of future observations. Another consequence of the random nature of the actions is that in a considerable amount of video samples, the robot does not touch any of the objects, making it more difficult for a model of object dynamics to be learned.

Another characteristic of the BAIR dataset that has particular effect on our results is the fact that actions are only updated every two frames. Even though the gripper's Cartesian position still changes at every time step, the variance of displacements in x and y is higher on time steps in which joint velocities are updated. This aspect of the data is depicted in Fig. 5.2, where the gripper displacements between each pair of frames of the test set is scattered for the y axis, revealing an alternating standard deviation. In practice, this alternating nature affects the action inference network described in section 3.3.2 as it does not experience all types of action the same way, therefore becoming better fit to some situations than others. For this reason, results are presented separately for odd and even steps.

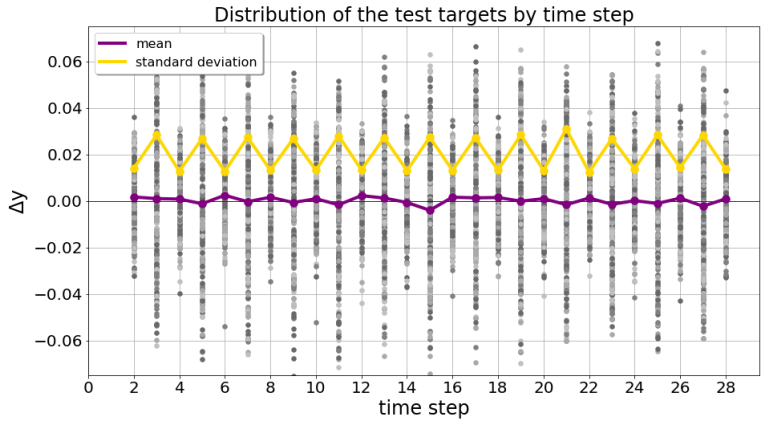


Figure 5.2: Alternating variance of the gripper displacement in the y axis for even and odd timesteps.

5.1.2 Google Push Dataset

Google's Push Dataset provides a similar environment to the one seen in BAIR dataset: a robotic arm pushing objects placed inside a container. There are, however, some differences that make this dataset a good complement to BAIR. Even though the broad task is the same, Google's dataset makes use of a different robotic arm with a different gripper, and new reference for the commanded actions. Also, in contrast with BAIR dataset, the actions are not random. Instead, the arm is commanded to perform pushes to the middle or sweeps to the outside. Other differences include the camera angle, objects present in the container, lighting and general aspect of the background. These disparities make the Google Push Dataset a good candidate for studying whether the proposed models can generalize and adapt to new situations and to verify that their design is not tuned to a particular situation. In total, the dataset is composed of 59000 examples that make for over 1.5 million frames.

5.2 Evaluating Video Prediction from a robotic standpoint

With the goal of providing the first method for benchmarking video prediction models from the standpoint of a robot using predictions to make action decisions, we proposed a new evaluation metric in section 3.5. In our solution, we argued that a video prediction model is as good for planning tasks, as the ability of an action inference model to infer the true actions from the predicted frames. As discussed in section 3.3, this idea contrasts with state of the art metrics, such as PSNR, SSIM and FVD, which are designed to measure human perception of quality.

5.2.1 Experimental Setup

To test whether our proposed action inference metric provides new insights that are useful for selecting the best video prediction model to be used in a planning task - like the one presented in section 3.2 - we select a set of state of the art video prediction models to be compared in both types of metric. The selection of the tested VP models was made with the intent of covering the main approaches to video prediction, which opens up the possibility of identifying the most significant features of a video prediction model used in a robotic planning context. Additionally, especial focus was given to action conditioned models. With that in mind, the selected models were 1) CDNA: a deterministic model based on pixel-motion [44], 2) SAVP: which also models pixel motion but introduces variational and adversarial terms to the loss, to try to improve prediction quality and account for the variability in the environment [51], 3) a variant of SAVP in which the adversarial term is suppressed, 4) SV2P: an extension of CDNA conditioned on stochastic variables [47], 5) and finally we test SVG-LP: the stochastic, action-free model of [50].

All tested VP models were pre-trained by the respective authors with exception of CDNA, which was trained on over 200000 iterations, using scheduled sampling [92]. At training time, models receive

2 context frames and actions (with the exception of SVG-LP which is action-free and therefore only receives the frames) and predict video up to time step 12, with each prediction being fed back as input for the next time step.

After training, a forward pass is made over the entire training set and the generated predictions, this time generalizing until step 30, are saved as a new dataset for subsequent training of the action inference model. Having a dataset of predictions for each VP model, the action inference network is trained on the 28 frame long predictions. In our experiments, we define the actions being inferred as the displacements Δx and Δy of the robot's gripper along the x and y axis, between every two time steps. The ground truth targets for the actions are directly extracted from the BAIR dataset gripper state sequences by subtracting consecutive temporal positions for both axis. This results in an action sequence of length 27 for each 28-frame predicted video.

5.2.2 Quantitative Comparison

We start by evaluating the selected group of VP models on some of the traditionally used metrics described in section 3.3 and on the recently proposed FVD. As opposed to the methodology adopted by some of the previous work [50], [51] in which 100 possible futures are sampled and the best score out of the group is reported, we choose to sample a single time, in order to better approximate the conditions of a robot planning actions. This approach has especial impact on action-free models like SVG-LP, that are exposed to greater uncertainty. Regarding the action-conditioned models, the results displayed in Fig. 5.3 are in line with previous reports, indicating that models have better performance when conditioned on both actions and stochastic variables, as is the case with SAVP-VAE and SV2P. On the other hand, the addition of an adversarial loss term seems to affect performance negatively, which reflects on SAVP having a lower PSNR/SSIM score than a deterministic model like CDNA, despite the high visual appeal of the predicted frames.

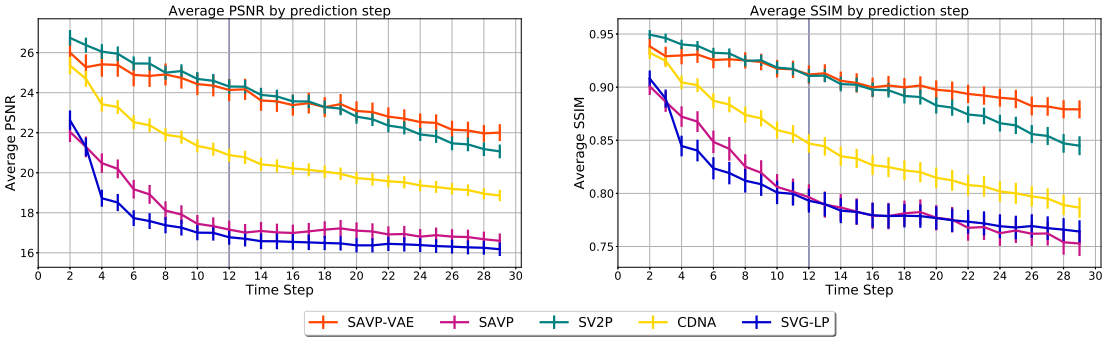


Figure 5.3: Average PSNR and SSIM over the test set with 95% confidence interval. The bold vertical line indicates the training horizon. Results were reproduced with modification from [44], [47], [50], [51].

We compute the FVD values for the test set predictions and present them in table 5.1 using batches of size 32 and discard the two context frames to only consider the predictions of length 28. As indicated by the authors of FVD, the metric is highly dependent on the batch size, as the larger it is, the better the approximation of the distributions P_R and P_P will be (see section 3.3). It is therefore important

that the batch size is kept constant when comparing across models. In our experiments, FVD was computed using batches of size 32 which, despite producing higher values when compared to the original paper [67] (where batch size was 256), preserves model rankings.

For each VP model's predictions dataset, the action inference model that produces the best validation score during training is selected. To measure how well it can identify the executed actions, we compute the R^2 goodness-of-fit metric which in our particular case represents the percentage of change in ground truth action variance that is explained by the inferred actions. A model that perfectly identifies the executed actions will have a score of 1.0 whereas a model that simply outputs the mean Δx and Δy will have a score of 0.0. It is worth noting that while R^2 may not be a strong enough statistic for comparing different regression models, the focus of this work is to assess the predictions made by different VP models, using the same training regime for the inference model. In our experiments R^2 is computed along the 256 test examples for each time step and the evolution of the metric over time is reported in Fig. 5.4. The Mean Absolute Error (MAE) is also presented in Fig. 5.5, computed in the same way as R^2, for each time step.

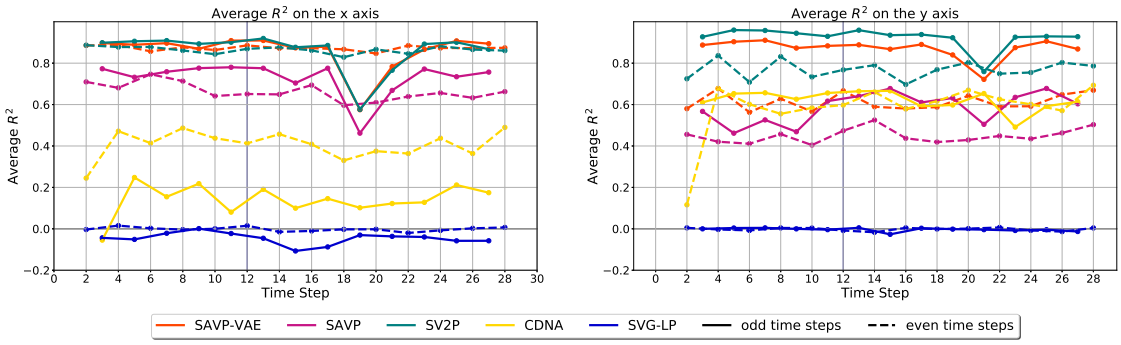


Figure 5.4: R^2 results over time for predictions made by different VP models. Odd and even time steps are presented separately.

The most immediate characteristic in the temporal evolution of action inference that arises from an initial analysis of figures 5.4 and 5.5 is that the temporal downgrade artefact in performance observed in PSNR and SSIM is not manifested in the capacity of the model to recognize the actions, with exception of results for CDNA. This quality of the metric stems from the fact that the parameters of the inference model are shared across all time steps, a choice based on the fact that action dynamics do not change over time and therefore VP models should have a consistent action encoding for all time steps. For this reason, a VP model that encodes actions in a consistent manner should allow the inference network to better learn how to recognize actions and will therefore display stable R^2 and MAE values across time, as is verified for SAVP and SAVP-VAE. On the other hand, because video predictions made by CDNA have changing dynamics, starting with good resolution and transitioning to blurry images as time advances, it is difficult for the action inference model to learn to identify actions.

The performance of the action inference on predictions made by different models indicates, based on figures 5.4 and 5.5 and on table 5.1, that the model that is better encoding action features and would therefore be the most suited in robotic planning problems is SV2P, closely followed by SAVP-VAE, im-

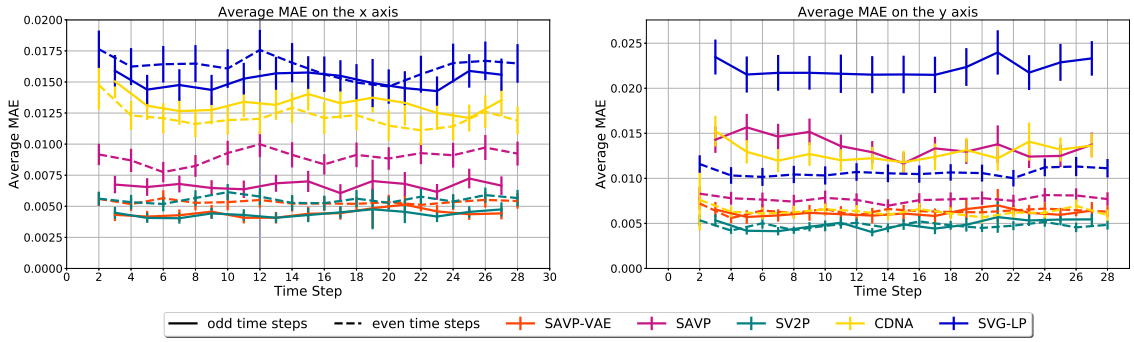


Figure 5.5: Average MAE results with 95% confidence interval for predictions made by different VP models. Odd and even time steps are presented separately.

plying that conditioning on stochastic variables is beneficial but the introduction of the adversarial loss for better image quality removes attention from optimal encoding of action features. These models even outperform the ground truth oracle, supporting the argument that the stochastic variables should be accounting for non observable aspects of the scene and that some blurring of the background may actually help the inference network focus on the action features. On the other hand, the action-free SVG-LP model has an R^2 value of approximately 0 and an MAE value of 0.163, which corresponds to the variance of the data. This indicates, as observed in section 5.2.3, that the inference model is unable to identify the actions and limits itself to predicting a constant average. The reason for this result is the fact that an action-free stochastic model from which a single prediction is sampled, may produce a future that is different from the ground truth, causing recognized actions to not match the targets and preventing the model from learning a meaningful mapping during training.

In general, and as reported by [67], PSNR and SSIM present a very high correlation as both of them are based on frame by frame comparisons with the original data. Furthermore, because most VP models use an ℓ_2 term in the loss function, these are biased metrics. We also verify that multiple ranking changes occur between our proposed score and FVD, including SV2P scoring the best in action recognition while having an FVD value close to that of SVG-LP, which for being action-free has the lowest score under our metric. These results show that the ability to recognize actions from predicted images doesn't necessarily correlate with previously proposed metrics and that action inference may offer a valuable perspective for choosing the best model in a planning scenario.

Table 5.1: FVD, R^2 and MAE values for each VP model.

Model	FVD Value	R2 Score	MAE Value
CDNA	943.5	0.4339	0.0111
SAVP	738.3	0.5953	0.0092
SAVP-VAE	**409.8**	0.8427	0.0056
SV2P	691.1	**0.8719**	**0.0049**
SVG-LP	728.2	-0.0582	0.0160
Oracle	0.0	0.8203	0.0058

5.2.3 Qualitative Comparison

To better understand how the R^2 and MAE results reflect on action inference in practice, examples of inferred action sequences are showed against the ground truth (dashed black line) in Figs. 5.6 and 5.7. We find the best and worst R^2 score on action inference from any VP model and then plot the inferred actions from that video sequence for all models. The best performing model in the selected examples was SAVP-VAE, while the worst was SVG-LP. A visual interpretation of the extent to which the inference network is successful at recognizing the actions is in conformance with the results of 5.2.2, with the SAVP-VAE and SV2P models performing the best, followed by SAVP and CDNA.

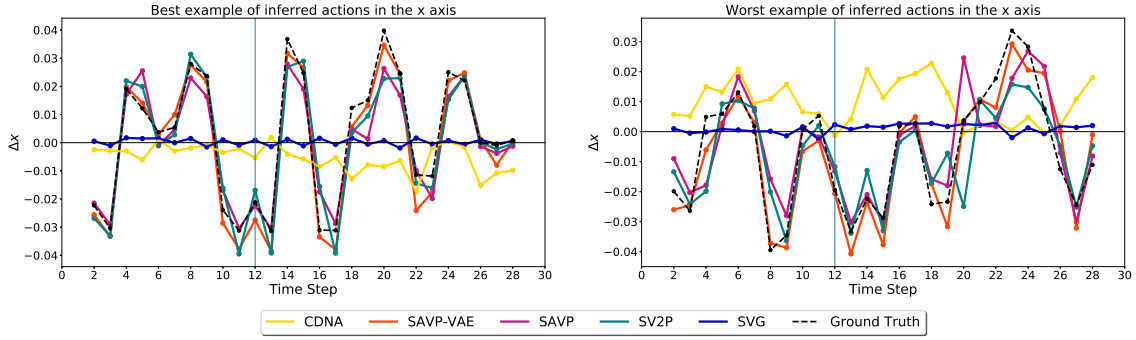


Figure 5.6: Best and worst examples of inferred Δx.

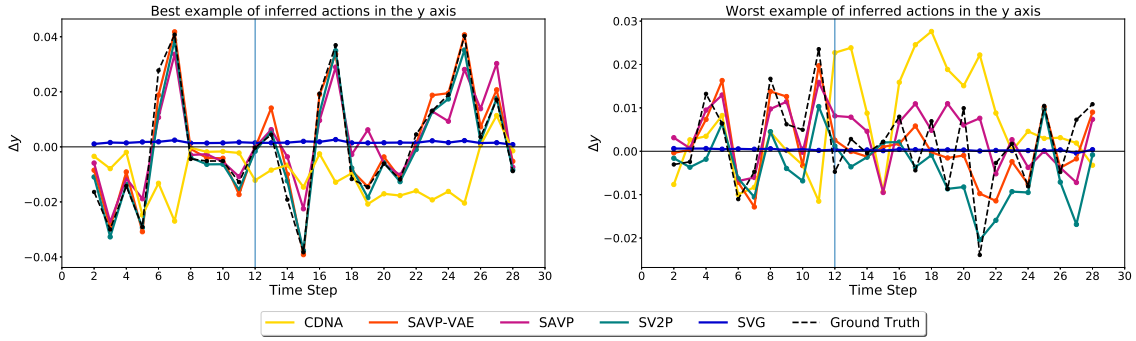


Figure 5.7: Best and worst examples of inferred Δy.

5.3 Disentangling agent information from external objects

5.3.1 Agent information

In analysing how the proposed model fares in natural scenarios like the one of BAIR dataset, we start by addressing its disentanglement capabilities. Here, the intention is to understand if ADR-AO is capable of distinguishing the body of the agent (in this case the robotic arm) from other bodies in the environment. As detailed in section 4.3, this must be achieved using only the content information present in an initial frame and the sequence of actions (or positions) in which the agent will be at each time step.

Most existing metrics for assessing disentanglement are designed for information theoretic approaches to the problem, where the learned factors of variation encoded by each dimension of the hidden representation are encouraged to be uncorrelated (see section 4.1). These metrics attempt to measure properties of the hidden representation that include informativeness, independence and interpretability [73] [95]. In contrast, the disentanglement approach followed in this work relies on inductive biases related to the structure of the data, in particular its sequential nature, imposing no constraints on the hidden representation. Hence, the available quantitative metrics for evaluating disentanglement are not applicable to our proposed model. Instead, we must rely on a qualitative assessment, knowing that the learned representations should be (1) informative about the reconstructed frame in the sense that it allows a correct recovery of the content and agent pose, (2) flexible enough to sample different possible (and possibly unseen) scenarios in a planning task, and (3) generalizable to new domains with a short adaptation period. Different experiments were implemented to verify these conditions, with the results being presented in the following sections.

Can the model recognize self information?

We start by evaluating the reconstruction capabilities of ADR-AO. Because the intention is to learn representations for self (agent) information and content, an ideal result on BAIR dataset would be for these to allow the reconstruction of the future frame $\boldsymbol{x}^{t+k}$ with the robotic arm correctly positioned and all the objects in their original (most recently observed) pose. Figure 5.8 shows two predicted sequences obtained using ADR-AO, where 5 frames were given as input and the next 15 predicted. The model succeeds in isolating visual information that is part of the agent (in this case the robotic arm), as demonstrated by its ability to correctly translate the arm, despite some loss in sharpness. Furthermore, the fact that the predicted movement matches the ground truth trajectory of the arm means that the model is successfully learning a mapping between commanded actions and the resulting arm position in the visual field.

A shortcoming of the proposed approach for disentangling content and self information from the moved objects is the fact that the reconstruction term of the loss encourages the model to reconstruct the whole future frame, including the objects that were displaced. Even though the limited information contained in the inputs provided to the model prevents it from consistently learning to predict the movement of the objects, it nevertheless predicts the existence of some change in the pixels that are part of these objects. Specifically, the model often blurs objects which predictably would have moved as a result of the movement of the robotic arm, in an attempt to reduce the reconstruction error in these image patches.

As detailed in section 4.3, we try to mitigate this effect by introducing stochastic variables. Despite these resulting in some improvement, they are not a complete solution to the problem, which is left for future work. Nevertheless, it is important to note that the agent's actions and object movement are inherently correlated and therefore can't be completely disentangled from one another. We believe in

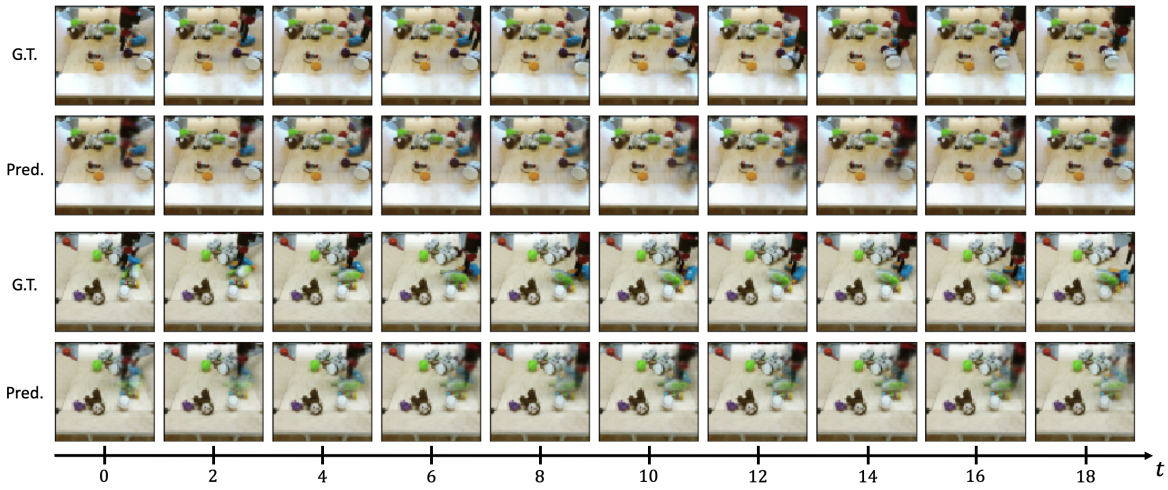


Figure 5.8: ADR-AO predictions on BAIR dataset. While the movement of the arm is well predicted, displaced objects are left in their original position. Better seen online at https://web.ist.utl.pt/ist181063/vp_examples/adr_ao.

fact, that due to this correlation, it should be possible for ADR-AO to gain a consistent understanding of the object movements that result from the robot's actions, provided the existence of enough data and model capacity. In practice however, this is not the case and, as intended, the error between the ground truth future frames and the predictions ends up being dominated by information of the objects that were displaced by the agent.

Can the model generate new accurate futures?

With the intent of the designed model being its use in robotic planning tasks, where different possible action sequences and their corresponding future can be sampled, it is important to determine whether it is capable of generating previously unseen agent movements. To do so, we train ADR-AO using only the Cartesian position of the gripper as input actions (before the arm's joint angles were also used), allowing artificial and previously unseen arm movements to be handcrafted by setting a sequence of positions where the arm should move to. This experiment is illustrated in Fig. 5.9, where the original actions of a video sequence of the dataset are replaced by a new, infinity shaped trajectory. The model is able to generate a future that matches the new trajectory while maintaining the original background. This confirms that, if the object movement can later be modelled, ADR-AO may be used for planning tasks.

Is the representation of self generalizable to new environments?

Finally, we investigate whether the representations learned on BAIR dataset are generalizable to new environments, given some short adaptation. For that, ADR-AO pre-trained on BAIR dataset is fine-tuned on 800 examples of the Google Push Dataset, which despite constituting a related scenario, has a different robotic arm, gripper, objects, camera angle and reference for the joint angles and gripper position. As stated in the literature [70], disentangled representations should facilitate tasks such as

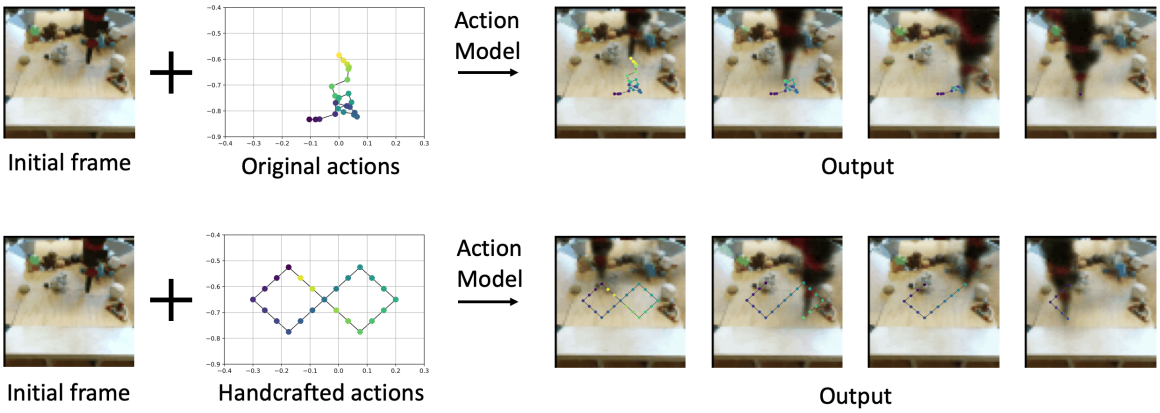


Figure 5.9: Prediction with handcrafted action sequence, allowing mental imagination of previously unseen movements. Better seen online at https://web.ist.utl.pt/ist181063/vp_examples/disent_self_movement/

transfer and few-shot learning, so a good performance on Google Push Dataset based on such few adaptation examples should be a positive indication of the content and self information sources being separated.

The ability to adapt to the Google Push Dataset is illustrated in Fig. 5.10, where three different sequences and the corresponding predictions are presented. From a first glance at the sequences it is apparent that the model succeeds in reconstructing the background with remarkable detail, despite the limited number of observations of the environment and the objects it contains. On the other hand, the visual information of the robotic arm is well predicted in terms of trajectory but somewhat blurred. This is an expected result given that the content suffers little change from the most recently observed frame and the decoder has access to skip connections that allow a better flow of information. In comparison with BAIR, the robotic arm has movements of greater amplitude which are difficult to map from the commanded action, given such a reduced number of examples. This results in greater uncertainty and in the averaging of possible positions, causing the blurred effect. Furthermore, as illustrated in the third row of Fig. 5.10, when the movements of Google's robotic arm diverge too much from the experience collected with the BAIR dataset - such as the articulation of the last joint before the gripper - the reconstruction error is noticeably higher. Nevertheless, the model is capable of identifying the new arm and gripper and displays some success in adapting the action representation to the reference frame, with the broad movements being well predicted in spite of the loss of precision when compared to predictions in BAIR dataset.

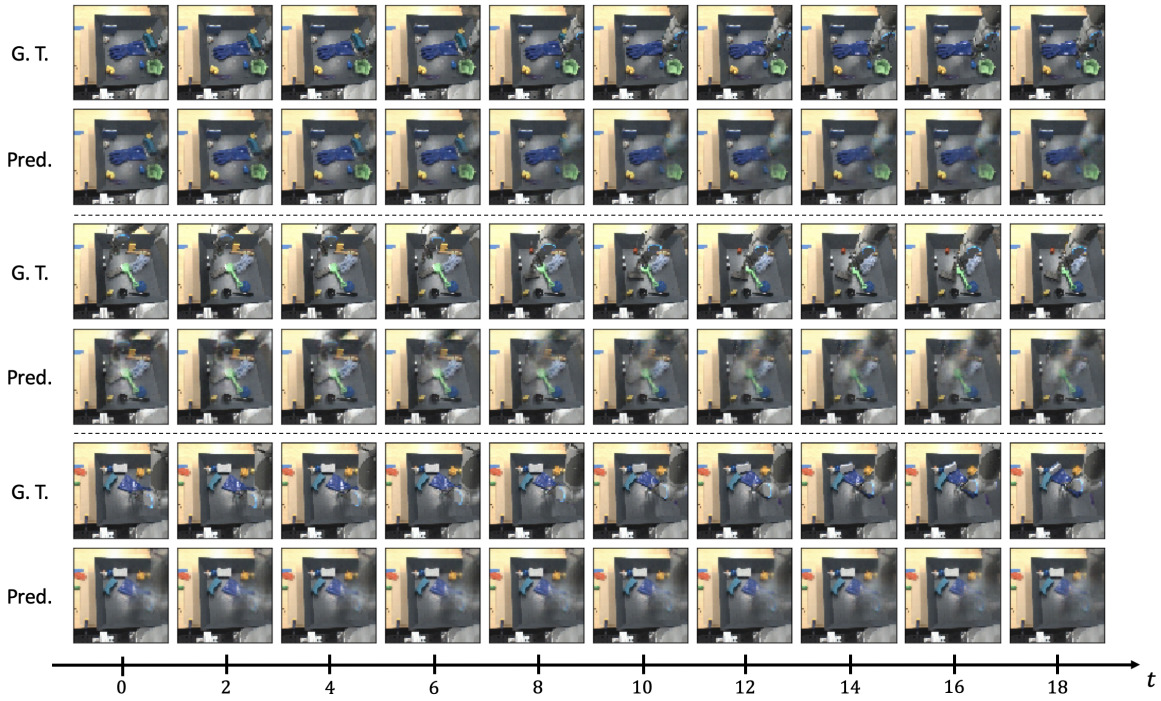


Figure 5.10: Generalization to Google Push Dataset. The first two rows illustrate the ability of the model to recognize a different arm and broadly adapt to a new reference frame for the actions and camera angle. On the other hand, as exemplified in the third row, actions that diverge too much from the experience collected in BAIR dataset result in greater error. Better seen online at https://web.ist.utl.pt/ist181063/vp_examples/google_push/

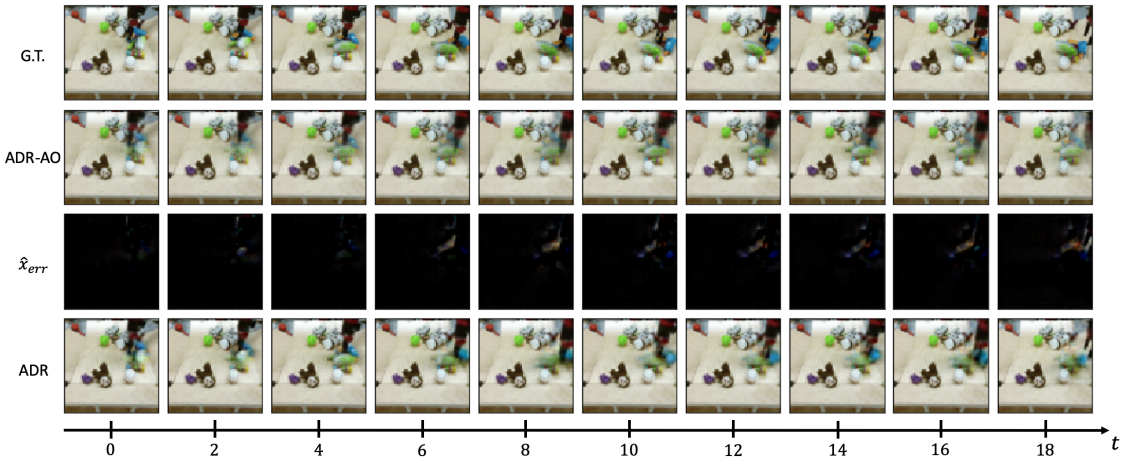
5.3.2 Object information

Having modelled the movement of the agent we now turn to the objects in the scene. As described in section 4.3.2, the separate object representation is obtained as a consequence of ADR-AO's prediction error, which we consider to be dominated by object information, therefore requiring no further steps for disentanglement. As such we restrict the analysis to a qualitative assessment of the reconstruction ability of ADR.

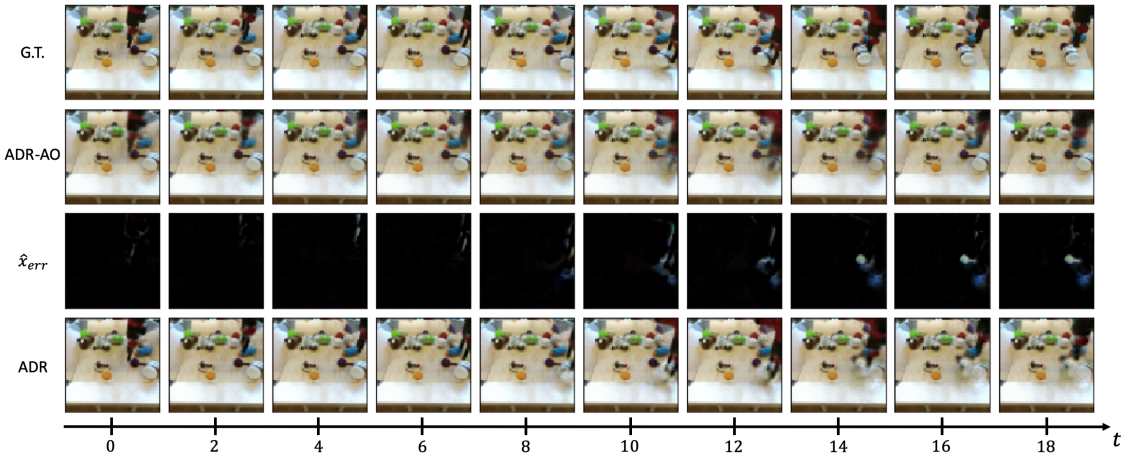
In general, we observe that the model succeeds in capturing the object information conveyed in the error signal, enough for the true movement of the objects to be perceived in a frame by frame reconstruction. This is illustrated by the first example of Fig. 5.11, where a group of objects is pushed and their movement, which was absent in the output of ADR-AO, is now apparent. There is however a significant amount of blurriness in the moved objects, which is likely introduced by ADR-AO as an attempt to minimize the reconstruction loss in areas around objects, an effect that was described in section 4.3.2. This is supported by the example of Fig. 5.11(a). Another aspect worth highlighting is how ADR can sometimes show some difficulty in modelling the negative change of the object, *i.e.*, in removing the object from its initial position. This is exemplified in Fig. 5.11(b), where the latter frame reconstructions show that the Nutella jar leaves a ghost like shade in its original position. Additional examples can be seen online in the *url* of Fig. 5.11.

These results demonstrate that the broad movement of the objects is well reconstructed, which rep-

resents a first step towards object information being well modelled - one of the main faults of existing video prediction models. Nevertheless, the aforementioned challenges show that there is still room for improvement and better object reconstruction should result in increased consistency in the $\boldsymbol{h}_o$ vector, which in turn should facilitate the training of ADR-VP.



(a)



(b)

Figure 5.11: Reconstruction with ADR. Better seen online at https://web.ist.utl.pt/ist181063/vp_examples/adr.

5.4 Video Prediction

To evaluate the video prediction capabilities of ADR we test the model both on visual quality metrics (PSNR, SSIM and FVD) and in our proposed action inference metric. Because the results for the available state of the art models have been analysed in section 5.2, we now opt to present only the results for the three best models previously found (CDNA, SAVP-VAE and SV2P). Additionally, we show the performance of both the deterministic and stochastic versions of ADR-AO and of ADR trained on top of the stochastic ADR-AO. As before, for these experiments all models were trained

to predict frames up to time step 12, using two context frames. The results presented were obtained by providing the same amount of context frames as in training but asking the models to predict until time step 30, in order to understand how they perform when the prediction horizon is extended. The results were obtained for BAIR dataset, on a test set of 256 examples.

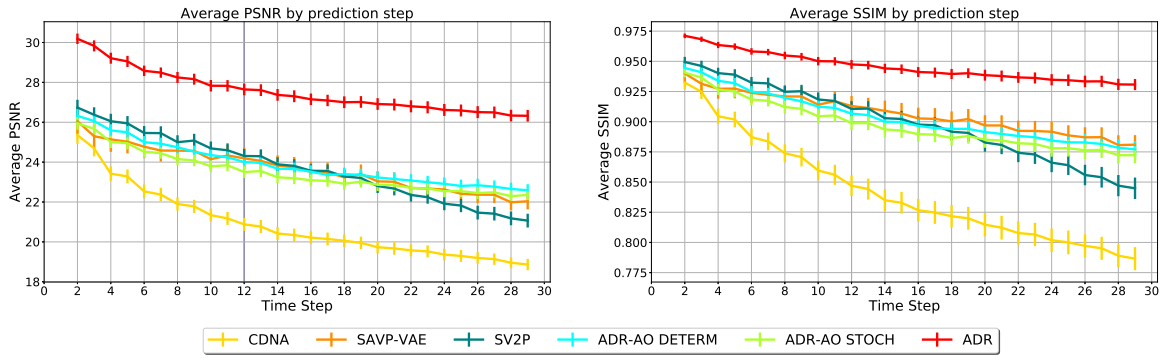


Figure 5.12: Comparison between the proposed models and existing VP models in terms of PSNR and SSIM. The vertical blue line indicates the training horizon.

Starting with an analysis of the models' performance in terms of PSNR and SSIM, presented in Fig. 5.12, there are two aspects to highlight. The first one is that both the deterministic and stochastic versions of ADR-AO obtain a performance that matches the performance of the two best video prediction models available in the literature (SAVP-VAE and SV2P). This is especially true in comparison with SV2P, which has a much more significant drop in performance as the prediction unfolds. On the other hand, SAVP-VAE shows a very high correlation with our ADR-AO, being slightly outperformed in terms of PSNR in the final steps, but also slightly outperforming in terms of SSIM in the final steps. On another note, we verify that the deterministic version of ADR-AO marginally outperforms the stochastic version. Because both of these metrics are very correlated with the ℓ_2 loss, this is a result that was expected as, during training, the stochastic version has an extra loss term to optimize for, removing focus from the ℓ_2 reconstruction term. On the other hand, the deterministic ADR-AO is allowed to minimize the reconstruction loss as much as it can, even if - as seen in Fig. 4.6 - this results in added blurriness and objects disappearing.

It is important to consider that unlike the other video prediction models, ADR-AO is explicitly designed not to consider object information, making these two models the floor of the performance that our complete model can achieve. Furthermore, the fact that in spite of this characteristic, ADR-AO still achieves state of the art results is a confirmation that existing video prediction models fail at conveying the future of object information and a validation of the importance of our research questions and proposal.

The second aspect of Fig. 5.12 worth being highlighted is the added performance that comes with considering object information, as shown by ADR. Even though ADR is not in practice a video prediction model, as it requires access to ground truth video frames (as discussed in section 4.3.3), it still provides proof of the importance of modelling object movement and represents a ceiling on the performance that ADR-VP may achieve. As such, results for a working ADR-VP should be between

ADR-AO and ADR, prospecting that further work on ADR-VP can lead to an improvement of the state of the art in video prediction.

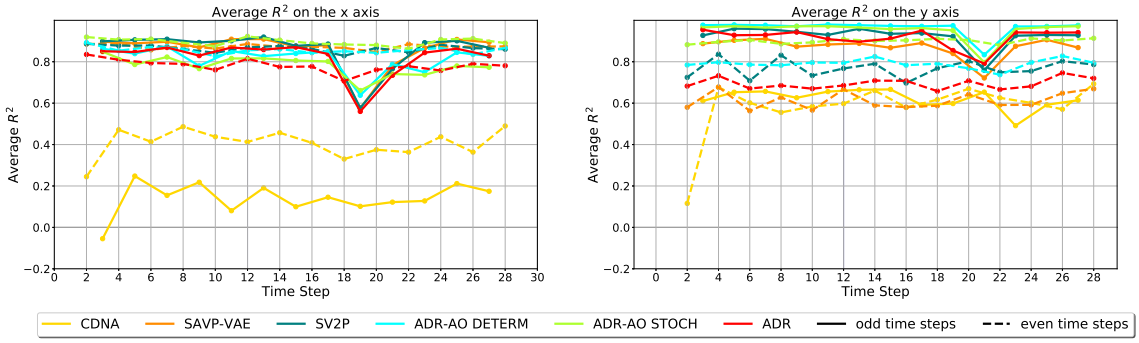


Figure 5.13: Average R^2 results for action inference on predictions by the proposed models. The vertical blue line indicates the training horizon.

As a second way of measuring model performance, we test how well the predicted frames allow an action inference model to recognize the agent's actions, with the results measured in terms of R^2 score and MAE and presented in Figs. 5.13 and 5.14 and in table 5.2. As discussed in section 5.2, this analysis is directed at evaluating the fitness of the video prediction models for robotic planning tasks.

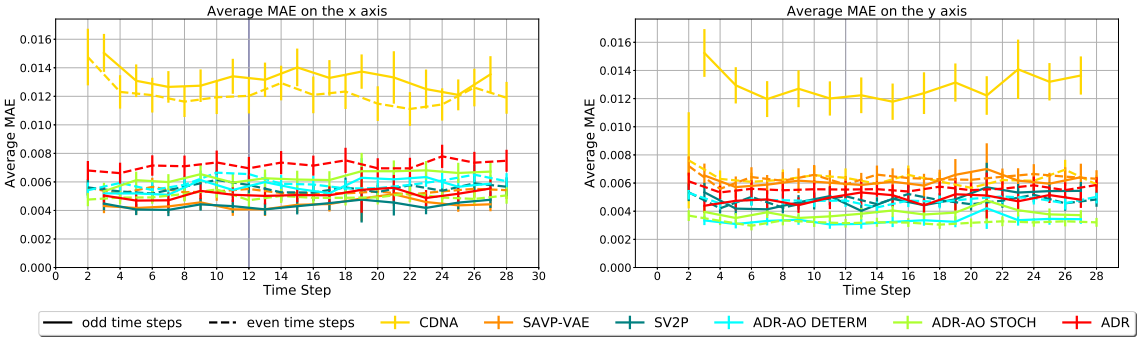


Figure 5.14: Average MAE results for action inference on predictions by the proposed models. The vertical blue line indicates the training horizon.

The results indicate that the best action recognition was achieved in frames predicted by the stochastic version of ADR-AO, closely followed by the deterministic ADR-AO and SV2P, with the separation between models being better perceived in the values of table 5.2. Interestingly, while in PSNR and SSIM the stochastic ADR-AO model did worse than the deterministic, it is now the opposite. This is a result that further demonstrates that simply minimizing the reconstruction error is not beneficial for downstream tasks such as classification and planning and that a prediction that is plausible is preferable to one that blurs out uncertain aspects of the scene. A final aspect to take into consideration is how ADR's predictions don't allow a better performance of the action inference model, when compared to the agent only versions. This suggests that keeping the background static while moving the arm is helpful for the inference model. A possible explanation for this result is that, because the model only has to correctly recognize the movement of the arm, the removal of other dynamic information allows it to focus on its task. This idea is further supported by the oracle results (where the action

Table 5.2: Model comparison in terms of FVD, R^2 and MAE.

Model	FVD Value	R2 Score	MAE Value
CDNA	943.5	0.4339	0.0111
SAVP	738.3	0.5953	0.0092
SAVP-VAE	**409.8**	0.8427	0.0056
SV2P	691.1	0.8719	0.0049
SVG-LP	728.2	-0.0582	0.0160
ADR-AO Determ.	793.0	0.8759	0.0049
ADR-AO Stoch.	857.2	**0.8837**	**0.0046**
ADR	801.1	0.8297	0.0057
Oracle	0.0	0.8203	0.0058

inference model was run on ground truth frames), which are both worse than the results achieved by some of the video prediction models and very close to those achieved by ADR.

6

Conclusions and Future Work

6.1 Conclusions

To summarize and discuss the work developed in this thesis we reflect on the questions proposed in the introduction.

In trying to solve the first question, related with the difficulty of assessing video prediction models from the standpoint of a robot planning actions, we proposed a novel metric. Based on an action inference network that tries to recognize the actions of the robot from the predicted frames, we were able to demonstrate the limitations of evaluating video prediction models solely based on their ability to generate realistic looking frames. Instead, we argue that the choice of video prediction models used in planning tasks should be based on complementary metrics, so that both visual quality and the ability of the model to understand the consequences of the robot's actions are considered. A shortcoming of our action-inference metric is that it only takes into account the actions of the robot, as these are the only ground truth queue that we had access to. To be completely generic, our approach should be extended to allow the verification of whether the result of the actions is not only well predicted in terms of their consequences on the position of the arm, but also in terms of their consequences on object positions.

In our second research question, we proposed to test the idea of whether the sequential structure of video and the agent's actions can be used to disentangle sources of information in video. Our proposed solution takes advantage of these inductive biases to separate visual information related with the agent, from information of the objects present in the scene. The experiments we conducted show the ability of our solution to separately process the movement of the agent and to obtain an unsupervised signal dominated by information of moving objects, which can be independently modelled. We also demonstrate the ability of the model to generate previously unseen sequences and to adapt to new environments, given a short learning period. Nevertheless, we consider that because of the inherent correlation between the agent's actions and object movement, these two sources of

information cannot be completely disentangled, which is a limitation of our approach. Moreover, our strategy for obtaining representations for content, actions and object does not encourage independence in the dimensions of the hidden vectors, which would bring the added benefit of interpretable representations.

As a last research question, we explored the possibility of improving state of the art in video prediction by building a model based on the previously obtained representations. In our approach, we try to shift attention from the agent's own movement, which is foreseeable, to the movement of the objects, which has greater uncertainty and is more difficult to model. Our solution includes two video prediction models: ADR-AO and ADR-VP. ADR-AO manages to match, and in some cases improve, the state of the art while explicitly disregarding object information. Such a result is a confirmation of the inability of existing models to predict object dynamics and highlights the importance of our proposal to focus on object information. Yet, by not considering the objects, ADR-AO is not of practical use in planning tasks. In those applications, ADR-VP should be the better choice. However, as described in section 4.3.3, we failed to train ADR-VP to the point where it succeeds in object prediction. While there may be different reasons for this result, we highlight the imbalance in dynamic vs. static frames in BAIR dataset and the complexity of the 3 stage training of our model as the most likely causes. Nevertheless, the results presented for ADR, which represent a ceiling on what may be achieved with ADR-VP, are an encouraging signal and serve as motivation for future work on fixing ADR-VP.

6.2 Future Work

There are several possible directions for continuing the work developed in this thesis. Starting with the proposed action inference metric which, as stated before, should be extended to also consider the objects in the scene. Doing that may require the introduction of better datasets that include states of the environment other than gripper position, in particular objects' positions and speeds, allowing the assessment of models based on more comprehensive states.

Regarding the ADR models, there is an array of possible improvements and extensions. The most immediate work direction is fixing ADR-VP, possibly with a simpler architecture, trainable end-to-end and testing it on more diverse large scale datasets such as RoboNet [96]. The proposed models should then be tested on planning tasks similar to the one proposed in [60], and compared to alternative video prediction models. This would allow both the study of the correlation between our action-inference metric with the success rate in planning tasks, and to determine whether our proposed solution for video prediction is in fact better suited for planning tasks.

In another direction, it would be interesting to extend the idea of disentangling agent and objects to new tasks. For example, in scenarios where the camera is attached to a moving robot, it would be interesting to use the robot's movement as commanded action. The model should then learn to discount the robot's own movement, correct for the evolving camera position and allow the identifica-

tion of moving bodies. Another possibility is the addition of third person agents, who can generate movement that is completely independent of the robot.

Finally, to continue the study of the role of prediction in the human brain, architectural modifications can also be considered. For instance, the extension to an hierarchical version of the model, where successive stages of error are modelled would be closer to theories of the brain such as predictive coding.

6.3 Closing remarks

The models and experiments described in this thesis were implemented using Keras [97] and Tensorflow [98] and developed on over 1400 hours of GPU time. Adding to the work described throughout this document, there are complementary material contributions that hopefully can help future research on the topic, we highlight the following:

- **ICRA publication**. The work on video prediction benchmarking was published as a conference paper in ICRA 2020, under the title "Action-conditioned Benchmarking of Robotic Video Prediction Models: a Comparative Study", which can be consulted at http://vislab.isr.ist.utl.pt/wp-content/uploads/2020/07/mserra-icra2020.pdf;
- **ICRA presentation**. With the conference being adapted to an online format, we prepared a video presentation of the paper which can be viewed at https://www.youtube.com/watch?v=MVCRLVg8vc8&t=1s;
- **Code for action inference metric**. In order for the proposed metric to be easily used to assess models that we did not consider or that may be published in the future, we make our code base public and include instructions on how to use it. It can be accessed at https://github.com/m-serra/action-inference-for-video-prediction-benchmarking;
- **ADR code**. We also make the code for the proposed disentangled video prediction model available at https://github.com/m-serra/adr;
- **Pre-trained models**. Both aforementioned repositories include pre-trained models for our results to be easily reproduced and compared with, in future publications.

Bibliography

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

[2] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[3] 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*, 2015, pp. 91–99.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[5] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European conference on computer vision*. Springer, 2016, pp. 649–666.

[6] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[7] Statista.com. (2019) Hours of video uploaded to youtube every minute as of may 2019. Accessed 3 Sep. 2020. [Online]. Available: https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/

[8] Cisco. (2018) Cisco predicts more ip traffic in the next five years than in the history of the internet. Accessed 3 Sep. 2020. [Online]. Available: https://newsroom.cisco.com/press-release-content?articleId=1955935

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[10] E. R. Kandel, J. H. Schwartz, T. M. Jessell, D. of Biochemistry, M. B. T. Jessell, S. Siegelbaum, and A. Hudspeth, *Principles of neural science*. McGraw-hill New York, 2000, vol. 4.

[11] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, p. 574, 1959.

[12] K. A. Martin, “A brief history of the “feature detector”,” *Cerebral cortex*, vol. 4, no. 1, pp. 1–7, 1994.

[13] D. Marr, *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 1982.

[14] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[15] A. Cauchy, “Méthode générale pour la résolution des systemes d’équations simultanées,” *Comp. Rend. Sci. Paris*, vol. 25, no. 1847, pp. 536–538, 1847.

[16] S. Linnainmaa, “The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors,” *Master’s Thesis (in Finnish), Univ. Helsinki*, 1970.

[17] Y. Le Cun, “Learning process in an asymmetric threshold network,” in *Disordered Systems and Biological Organization*, E. Bienenstock, F. F. Soulié, and G. Weisbuch, Eds. Springer Berlin Heidelberg, 1986.

[18] D. Parker, “Learning-logic (tech. rep. tr-47). cambridge, ma: Massachusetts institute of technology,” *Center for Computational Research in Economics and Management Science*, 1985.

[19] D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986.

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

[21] Y. Bengio, “Deep learning of representations: Looking forward,” in *International Conference on Statistical Language and Speech Processing*. Springer, 2013, pp. 1–37.

[22] C. Olah. Neural networks, manifolds, and topology – colah’s blog. Accessed 8 Jul. 2020. [Online]. Available: https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

[23] F.-F. Li, A. Karpathy, J. Johnson, and S. Yeung. (2017) Stanford CS231n: Convolutional neural networks for visual recognition. [Online]. Available: http://cs231n.stanford.edu/

[24] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net,” *International Conference on Learning Representations (ICLR)*, 2015.

[25] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, “Understanding neural networks through deep visualization,” *International Conference on Machine Learning (ICML)*, 2015.

[26] C. Olah. Understanding lstm networks – colah’s blog. Accessed 6 Jan. 2019. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

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

[28] D. Ha and D. Eck, “A neural representation of sketch drawings,” in *ICLR 2018*, 2018, 2018. [Online]. Available: https://openreview.net/pdf?id=Hy6GHpkCW

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

[30] M. W. Spratling, “A review of predictive coding algorithms,” *Brain and cognition*, vol. 112, pp. 92–97, 2017.

[31] M. V. Srinivasan, S. B. Laughlin, and A. Dubs, “Predictive coding: a fresh view of inhibition in the retina,” *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 216, no. 1205, pp. 427–459, 1982.

[32] R. P. Rao and D. H. Ballard, “Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects,” *Nature neuroscience*, vol. 2, no. 1, pp. 79–87, 1999.

[33] M. Davis. Aoccdrnig to a rscheearch at cmabrigde uinervtisy, it deosn’t mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. the rset can be a toatl mses and you can sitll raed it wouthit porbelm. tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe. Accessed 22 Jul. 2020. [Online]. Available: http://www.mrc-cbu.cam.ac.uk/people/matt.davis/cmabridge/

[34] A. Clark, *Surfing uncertainty: Prediction, action, and the embodied mind*. Oxford University Press, 2015.

[35] W. Lotter, G. Kreiman, and D. Cox, “Deep predictive coding networks for video prediction and unsupervised learning,” *arXiv preprint arXiv:1605.08104*, 2016.

[36] E. Watanabe, A. Kitaoka, K. Sakamoto, M. Yasugi, and K. Tanaka, “Illusory motion reproduced by deep neural networks trained for prediction,” *Frontiers in psychology*, vol. 9, p. 345, 2018.

[37] A. Turing, “Computing machinery and intelligence-am turing,” *Mind*, vol. 59, no. 236, p. 433, 1950.

[38] N. Srivastava, E. Mansimov, and R. Salakhudinov, “Unsupervised learning of video representations using lstms,” in *International Conference on Machine Learning (ICML)*, 2015, pp. 843–852.

[39] M. Mathieu, C. Couprie, and Y. LeCun, “Deep multi-scale video prediction beyond mean square error,” in *International Conference on Learning Representations (ICLR)*, 2015.

[40] D. M. Wolpert, J. Diedrichsen, and J. R. Flanagan, “Principles of sensorimotor learning,” *Nature Reviews Neuroscience*, vol. 12, no. 12, p. 739, 2011.

[41] R. Santos, R. Ferreira, Â. Cardoso, and A. Bernardino, “Sensori-motor networks vs neural networks for visual stimulus prediction,” in *4th International Conference on Development and Learning and on Epigenetic Robotics*. IEEE, 2014, pp. 287–292.

[42] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, “Convolutional lstm network: A machine learning approach for precipitation nowcasting,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 802–810.

[43] P. Luc, N. Neverova, C. Couprie, J. Verbeek, and Y. LeCun, “Predicting deeper into the future of semantic segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 648–657.

[44] C. Finn, I. J. Goodfellow, and S. Levine, “Unsupervised learning for physical interaction through video prediction,” *CoRR*, vol. abs/1605.07157, 2016. [Online]. Available: http://arxiv.org/abs/1605.07157

[45] T. Xue, J. Wu, K. Bouman, and B. Freeman, “Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks,” in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 91–99. [Online]. Available: http://papers.nips.cc/paper/6552-visual-dynamics-probabilistic-future-frame-synthesis-via-cross-convolutional-networks.pdf

[46] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool, “Dynamic filter networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 667–675.

[47] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine, “Stochastic variational video prediction,” *CoRR*, vol. abs/1710.11252, 2017. [Online]. Available: http://arxiv.org/abs/1710.11252

[48] L. Castrejon, N. Ballas, and A. Courville, “Improved conditional vrnns for video prediction,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7608–7617.

[49] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. P. Singh, “Action-conditional video prediction using deep networks in atari games,” *CoRR*, vol. abs/1507.08750, 2015. [Online]. Available: http://arxiv.org/abs/1507.08750

[50] E. Denton and R. Fergus, “Stochastic video generation with a learned prior,” *CoRR*, vol. abs/1802.07687, 2018. [Online]. Available: http://arxiv.org/abs/1802.07687

[51] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine, “Stochastic adversarial video prediction,” *arXiv preprint arXiv:1804.01523*, 2018.

[52] K. Friston, J. Kilner, and L. Harrison, “A free energy principle for the brain,” *Journal of Physiology-Paris*, vol. 100, no. 1-3, pp. 70–87, 2006.

[53] K. Friston, “The free-energy principle: a unified brain theory?” *Nature reviews neuroscience*, vol. 11, no. 2, pp. 127–138, 2010.

[54] P. Lanillos, S. Franklin, and D. W. Franklin, “The predictive brain in action: Involuntary actions reduce body prediction errors,” *bioRxiv*, 2020.

[55] G. Pezzulo, “An active inference view of cognitive control,” *Frontiers in psychology*, vol. 3, p. 478, 2012.

[56] P. Cisek and J. F. Kalaska, “Neural mechanisms for interacting with a world full of action choices,” *Annual review of neuroscience*, vol. 33, pp. 269–298, 2010.

[57] D. Ha and J. Schmidhuber, “Recurrent world models facilitate policy evolution,” in *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[58] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” 2016.

[59] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, “Learning to poke by poking: Experiential learning of intuitive physics,” in *Advances in neural information processing systems*, 2016, pp. 5074–5082.

[60] C. Finn and S. Levine, “Deep visual foresight for planning robot motion,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2786–2793.

[61] S. Winkler and P. Mohandas, “The evolution of video quality measurement: From psnr to hybrid metrics,” *IEEE transactions on Broadcasting*, vol. 54, no. 3, pp. 660–668, 2008.

[62] Z. Wang, L. Lu, and A. C. Bovik, “Video quality assessment based on structural distortion measurement,” *Signal processing: Image communication*, vol. 19, no. 2, pp. 121–132, 2004.

[63] M. P. Sampat, Z. Wang, S. Gupta, A. C. Bovik, and M. K. Markey, “Complex wavelet structural similarity: A new image similarity index,” *IEEE transactions on image processing*, vol. 18, no. 11, pp. 2385–2401, 2009.

[64] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

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

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

[67] T. Unterthiner, S. van Steenkiste, K. Kurach, R. Marinier, M. Michalski, and S. Gelly, “Towards accurate generative models of video: A new metric & challenges,” *arXiv preprint arXiv:1812.01717*, 2018.

[68] M. Fréchet, “Sur la distance de deux lois de probabilité,” *Comptes Rendus Hebdomadaires des Seances de L’Academie des Sciences*, vol. 244, no. 6, pp. 689–692, 1957.

[69] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[70] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem, “Challenging common assumptions in the unsupervised learning of disentangled representations,” in *International Conference on Machine Learning (ICML)*, 2019.

[71] A. Tharwat, “Independent component analysis: An introduction,” *Applied Computing and Informatics*, 2018.

[72] A. Hyvärinen and E. Oja, “Independent component analysis: algorithms and applications,” *Neural networks*, vol. 13, no. 4-5, pp. 411–430, 2000.

[73] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, “β-vae: Learning basic visual concepts with a constrained variational framework,” in *International Conference on Learning Representations (ICLR)*, 2017.

[74] H. Kim and A. Mnih, “Disentangling by factorising,” in *International Conference on Machine Learning (ICML)*, 2018.

[75] S. Watanabe, “Information theoretical analysis of multivariate correlation,” *IBM Journal of research and development*, vol. 4, no. 1, pp. 66–82, 1960.

[76] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee, “Decomposing motion and content for natural video sequence prediction,” *International Conference on Learning Representations (ICLR)*, 2017.

[77] W.-N. Hsu, Y. Zhang, and J. Glass, “Unsupervised learning of disentangled and interpretable representations from sequential data,” in *Advances in neural information processing systems*, 2017, pp. 1878–1889.

[78] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, “Mocogan: Decomposing motion and content for video generation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1526–1535.

[79] E. Denton and V. Birodkar, “Unsupervised learning of disentangled representations from video,” in *Advances in neural information processing systems*, 2017, pp. 4414–4423.

[80] A. Dehban, L. Jamone, A. R. Kampff, and J. Santos-Victor, “A deep probabilistic framework for heterogeneous self-supervised learning of affordances,” in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 476–483.

[81] S. J. Blakemore, C. D. Frith, and D. M. Wolpert, “Spatio-temporal prediction modulates the perception of self-produced stimuli,” *Journal of cognitive neuroscience*, vol. 11, no. 5, pp. 551–559, 1999.

[82] A. G. Enikolopov, L. Abbott, and N. B. Sawtell, “Internally generated predictions enhance neural and behavioral detection of sensory stimuli in an electric fish,” *Neuron*, vol. 99, no. 1, pp. 135–146, 2018.

[83] M. Lopes and J. Santos-Victor, “Visual learning by imitation with motor representations,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 3, pp. 438–449, 2005.

[84] A. Odena, V. Dumoulin, and C. Olah, “Deconvolution and checkerboard artifacts,” *Distill*, 2016, accessed 22 Aug. 2020. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

[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*. Springer, 2015, pp. 234–241.

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

[87] L. N. Smith, “Cyclical learning rates for training neural networks,” in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017, pp. 464–472.

[88] Y. Ye, D. Gandhi, A. Gupta, and S. Tulsiani, “Object-centric forward modeling for model predictive control,” in *Conference on Robot Learning*, 2020, pp. 100–109.

[89] M. Janner, S. Levine, W. T. Freeman, J. B. Tenenbaum, C. Finn, and J. Wu, “Reasoning about physical interactions with object-oriented prediction and planning,” *International Conference on Learning Representations (ICLR)*, 2019.

[90] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *International Conference on Machine Learning (ICML)*, 2013.

[91] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *Proceedings of Machine Learning Research*, vol. 37, pp. 448–456, 07–09 Jul 2015.

[92] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, “Scheduled sampling for sequence prediction with recurrent neural networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 1171–1179.

[93] F. Ebert, C. Finn, A. X. Lee, and S. Levine, "Self-supervised visual planning with temporal skip connections," *Conference on Robot Learning (CoRL)*, 2017.

[94] M. Kumar, M. Babaeizadeh, D. Erhan, C. Finn, S. Levine, L. Dinh, and D. Kingma, "Videoflow: A flow-based generative model for video," *International Conference on Learning Representations (ICLR)*, 2020.

[95] K. Do and T. Tran, "Theory and evaluation metrics for learning disentangled representations," *International Conference on Learning Representations (ICLR)*, 2020.

[96] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, "Robonet: Large-scale multi-robot learning," *arXiv preprint arXiv:1910.11215*, 2019.

[97] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[98] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/