Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institute for Computer Science
Department VI
Autonomous Intelligent Systems

Rheinische Friedrich-Wilhelms-Universität Bonn

Master Thesis

Stochastic Transformer for Prediction of Multiple Futures

Author:
Nathan Corral

First Examiner:
Prof. Dr. Sven Behnke

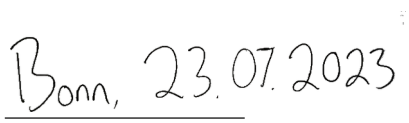
Second Examiner:
Prof. Dr. Christian Bauckhage

Advisor:
M. Sc. Angel Villar-Corrales

Date: July 23, 2023

Declaration

I hereby declare that I am the sole author of this thesis and that none other than the specified sources and aids have been used. Passages and figures quoted from other works have been marked with appropriate mention of the source.



Place, Date

Signature

Abstract

Video prediction is a challenging computer vision task that has recently attracted attention due to its applicability for a large variety of problems, such as autonomous driving, human-robot collaboration, or representation learning. However, making accurate predictions from video is a challenging task due to the inherent uncertainty, stochasticity, and complexity of real life environments. Several existing works use recurrent neural networks (RNNs) equipped with stochastic modules in order to capture video dynamics into latent variables and predict multiple futures. While these methods achieve impressive results, they are restricted by the limited ability of RNNs to learn long-range and complex dependencies. In this work, inspired by recent methods used in dialogue response generation, we develop a novel transformer predictor module able to model long-term dependencies, while still maintaining multiple valid predictions of the future. Our proposed transformer learns a prior distribution of the underlying uncertainty of the data, and combines latent variables sampled from this distribution with predicted features conditioned on the observed video frames in order to generate multiple possible futures. In experiments on Stochastic/Deterministic Moving-MNIST, KTH, and the human poses of Humans 3.6M, we show the efficient predictive power of transformers and the advantages of maintaining a stochastic interpretation of the future.

Contents

1 Introduction

Anticipation of what will happen next is an essential metric for intelligence. Humans regularly rely not only on this ability, but the ability of others to do the same. A random actor at a traffic light is a dangerous situation, due in part to the strong, collective assumption that they will behave within a set of pre-determined actions. In this thesis, we attempt to model the distribution of possible future outcomes in a situation-agnostic way. To this end, we present the STochastic TRansformer module (STTR), which learns to predict a distribution over possible futures.

Uncertainty is an underlying aspect of future prediction, as multiple futures can stem from the same seed [10] [5]. Inspired by [10], we learn a distribution of possible futures that can be sampled to make a stochastic prediction. A key advantage of stochastic over deterministic models is that as uncertainty grows, deterministic models will converge to the average of all possible future outcomes [51]. This makes the predictions of deterministic models blurry in the case where two outcomes are equally likely. Our stochastic model avoids this and continues to generate sharp and varied predictions in cases where multiple futures could exist.

Furthermore, maintaining a single, deterministic representation of the future can lead to penalizing valid guesses where multiple futures could be considered as logical continuations of the present. Consider the case of a person attempting to catch a train in Figure 1.1. Two valid predictions could be that they successfully board the train or that they miss it an are eventually forced to hail a cab. During training, our STTR selects the true future using non-causal knowledge and and also learns the distribution of possible outcomes. During evaluation, we follow [10] and [17] by taking multiple samples from this learned distribution and selecting the highest performer with respect to the ground truth.

We apply our model to two different settings. In Video Prediction, the STTR is evaluated over the perceived realism of the prediction, according to commonly used image metrics. In Human Pose Prediction, we predict the continuation of motion in the human skeleton.

The main contributions of this thesis are as follows:

1. We present the STTR, a transformer-based prediction model.

Figure 1.1: Uncertainty of the future can lead to two equally valid predictions, despite collapse into a single outcome. [19]

2. We show that our STTR is able to capture the uncertainty of the future contained in a distribution. Taking samples from this distribution makes both variable and realistic predictions.

3. Evaluation of our STTR shows very good performance when compared against strong, stochastic baselines.

This thesis continues in the following structure:

- *Chapter 2: Theoretical Background*: Summarize popular architectures in deep learning and the building blocks used in our model.
- *Chapter 3: Related Work*: Discuss the previous state-of-the-art work in the field of Video Prediction and Human Pose Prediction.
- *Chapter 4: Method*: Introduce our Stochastic Transformer and implementation details.
- *Chapter 5: Evaluation*: Comparisons against other models and several qualitative samples.
- *Chapter 6: Outlook and Conclusion*: Present our opinions on future direction and a conclusion of the thesis.

2 Theoretical Background

This sections represents an incomplete introduction to Deep Learning. Many methods are superficial summarizations of the extensive work developed and refined over the last decades. It should be included as to give high-level background information for full understanding of the thesis.

2.1 Stochastic Gradient Decent

A simple building block of Deep Learning is represented as a function f with parameters θ. It computes the output as follows:

$$\hat{y} = f_\theta(x). \tag{2.1}$$

To begin with, the parameters θ are initialized randomly and the output $\hat{y}$ is white noise. To get a more logical output, we need a meaningful set of parameters θ.

The data x we operate on is often very structured, for example an image. We say a very good set of parameters θ will extract features from x. Examples of features in an image are: the location, inside/outside, number of people, orientation of the ground plane, ...

To transition from random initialization of θ to a meaningful set of parameters, we optimize over θ. This optimization procedure needs:

- A desired output y_i for the each given input x_i. This composes a dataset of $\mathcal{K}$ input-output pairs $(x_i, y_i)\ \forall\ i \in 1..\mathcal{K}$
- A loss function L which ranks how good our prediction $\hat{y}_i$ is compared to the ground truth value y_i.

So long as the loss function L and learned function f are differentiable with respect to θ, we can update the parameters θ using gradient descent.

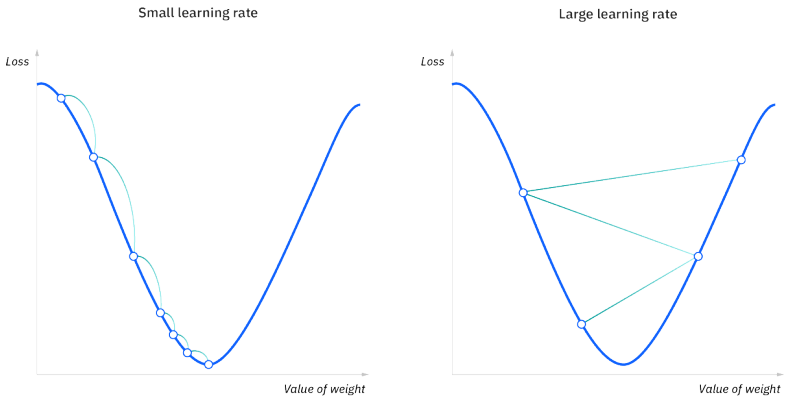


Figure 2.1: Effect of Learning Rate (LR) on the optimization process extracted from [1]. On the left the LR is too small and takes a long time to converge. On the right, the LR is too high and does not converge to a local minimum.

Gradient Descent

Gradient Descent searches for the optimal parameters θ^* which minimize:

$$\arg_\theta \min \sum_{i=0}^{\mathcal{K}} L(f_\theta(x_i), y_i). \tag{2.2}$$

It does so by first calculating the gradient with respect to θ. This gradient is the slope of the loss function L at the current values of θ. By taking a step in the opposite (downward) direction of the slope, we get closer to a minimum value of L. Taking a step is equivalent to updating the current values of θ:

$$\theta := \theta - \alpha \sum_{i}^{\mathcal{K}} \frac{\partial}{\partial \theta} L(f_\theta(x_i), y_i), \tag{2.3}$$

where α (the learning rate) controls the size of the step. Figure 2.1 gives a comparison of how different learning rates effect the value of the loss. The left of 2.1 shows how too small of a learning rate will require many update steps before settling at the local minimum. On the right shows how too large of a learning rate can lead to poor optimization. At the last point on the right figure, an additional step would overshoot the local minimum, and lead to a higher loss. An appropriate learning rate selection is key to a successful optimization.

Updating the Neural Network requires taking a gradient of the loss w.r.t. θ (i.e. the term $\frac{\partial}{\partial \theta} L(f_\theta(x), y)$ in Eq 2.3). The difficulty is that the loss function L depends on a sub-function f_θ. Fortunately, we can use the chain rule of calculus

which states:

$$\text{if } z = f(y) = f(g(x)), \text{ then } \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}. \tag{2.4}$$

The chain rule makes computation of the gradient possible. As f_θ can be composed of other sub-functions, we apply the chain rule for every module that composes f_θ.

Stochastic Gradient Descent

Datasets can be very large, making computation over all $\mathcal{K}$ elements a memory bottleneck. Additionally, the loss surface is complex and high-dimensional, meaning the optimal step depends not only on the gradient direction, but also the step size. Stochastic Gradient Descent makes an approximation of the gradient.

We select a mini-batch size B, such that $B \ll \mathcal{K}$. We then select B random pairs (x_i, y_i) from the dataset and compute the gradient (following Eq 2.3) using this reduced subset. In addition to being much more computationally efficient, this adds randomness to the gradients which can help avoid begin stuck in a local minimum.

2.2 Neural Network Common Modules

In this section, we present some common Neural Network (NN) building blocks for f_θ. These are chained together to form neural networks, using the chain rule to calculate the gradient.

2.2.1 Multi-layer Perceptron

A common building block for NNs is the Linear or Fully Connected (FC) Layer, which is diagrammed in Figure 2.2. Every output is a weighted combination of every input, with a bias that learns a constant shift of the output. Using matrix multiplication, this layer computes:

$$\hat{y} = Wx + b \tag{2.5}$$

where $W \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ are the learned parameters θ of the layer, denoted as the weights and bias, respectively.

The Mulit-Layer Perceptron (MLP) comes from cascading h Linear Layers. All layers between the input and output are named *hidden* layers. A hidden layer with more neurons than either the input or output is considered and inverse bottleneck.

This definition of an MLP is incomplete. The composition of linear transformations states that multiple linear transforms can be combined together into a single, linear transform. And so a linear layer with enough parameters would have equal expressive power compared to an MLP. This leads us to the next section, where we discuss how activations are inserted between NN Layers to add non-linearity to the output.

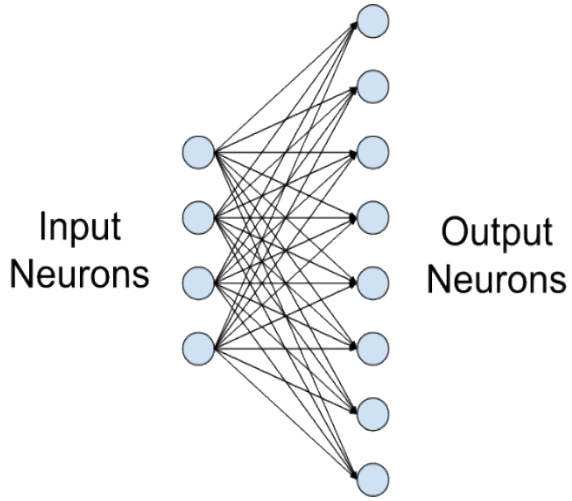


Figure 2.2: Fully Connected Layer. Each output neuron is connected to every input neuron. The 'bias' is omitted. Figure extracted from [2]

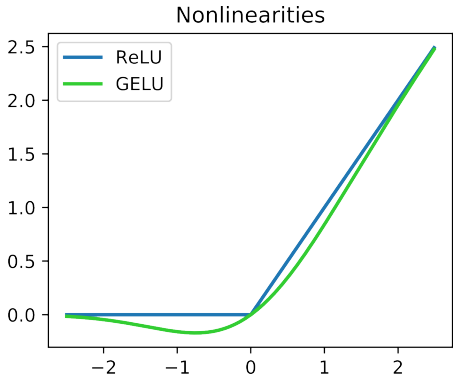


Figure 2.3: Plot showing the ReLU and GELU non-linear activation functions. Figure extracted from [3].

2.2.2 NonLinear Activation Functions

Non-linearity is an essential property of a universal function approximator [9]. The Universal Approximation Theorem states that a MLP with a single hidden layer and a sufficient number of neurons can approximate any continuous function, given the appropriate activation function. This highlights the expressive power of NNs, and the importance of a non-linear activation function.

Activation functions are typically placed on the output of each layers. A simple and popular non-linear activation function is the Rectified Linear Unit (ReLU) [31] [24], which follows the rule:

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.6}$$

The ReLU has no effect on the gradient when $x \geq 0$, and sets the gradient to zero otherwise. This can be problematical when, for example, a layer outputs a large negative number. Without gradients, it can be difficult to recover from the undesired state.

This leads to the introduction of Gaussian Error Linear Units (GELU) [25] shown in Figure 2.3. By adding a small slope when $x \leq 0$, the network is less failure prone.

Many other activations exist and are a good ablation study. For our networks, we use GELU in the Transformers and ReLU otherwise.

2.2.3 Convolutions

Convolutional Neural Networks (CNNs) [18] are a type of deep learning model that has revolutionized computer vision tasks such as image classification [31], object detection [20], and image segmentation [6]. CNNs are inspired by the concept of cross-correlation, which is a signal processing operation for computing the similarity between two signals.

Convolutions are applied to the input data using learnable filters, known as kernels. This is shown in Figure 2.4. These filters are small matrices that are slid across the input data, computing element-wise multiplications and summing the results. The 1D Convolution is formulated for each output j and kernel size K:

$$\hat{y}_j = b_j + \sum_{k=-K/2}^{K/2} x_{j-k} w_k, \tag{2.7}$$

where we also add a learnable bias b.

This process allows the network to capture local patterns and spatial dependencies within the data. The output of a convolutional layer is often referred to as a feature map, which represents the presence of specific features at different spatial locations.

The power of CNNs lies in stacked convolutions, to increase the receptive field of each point in the feature map. The receptive field is the region of the original input which has an effect on a single point on the feature map, and grows as convolutions are stacked. CNNs a popular choice in computer vision, achieving state-of-the-art performance in many image-based tasks.

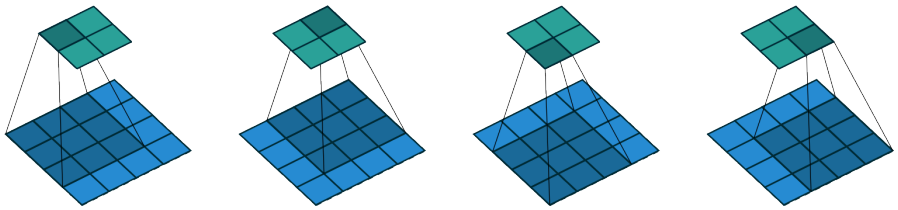


Figure 2.4: Convolving a 3×3 Kernel over a 4×4 input. Figure taken from [13]

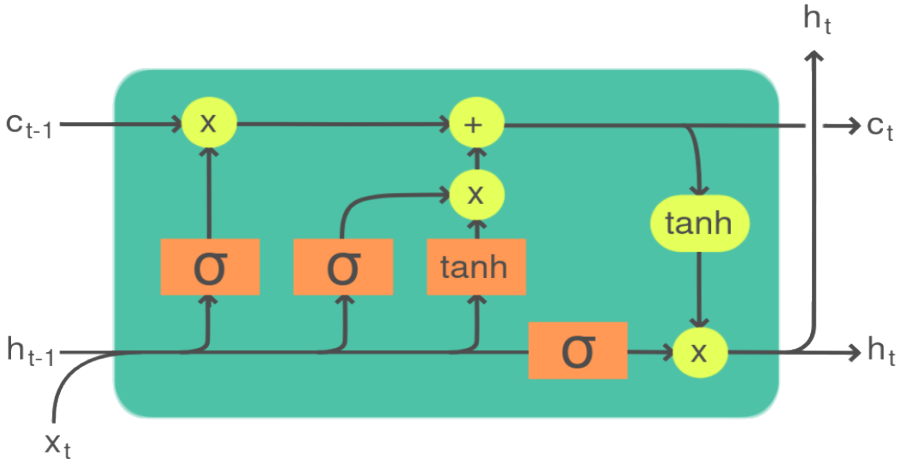


Figure 2.5: The LSTM operates by passing two outputs to itself in the next step. The first output, a cell state c_t, keeps long term information that may be useful at later timesteps. The second output is the short term memory which forms the prediction (h_t) at the current timestep.

2.2.4 Recurrent Neural Networks

So far, the modules we have discussed operate without memory. For each input, we produce an output with no recollection of what came previously. This motivates the Recurrent Neural Network (RNN). RNNs are characterized by their recurrent connection, which loops from their output back to their input. They use this connection to maintain temporally significant information across timesteps.

The key characteristic of an RNN is its recurrent nature, where the output of a previous step is fed back as input to the network at the next step. This loop enables the network to maintain a form of memory and capture relevant information from the past, to use with future predictions. RNNs are a natural choice for handling temporally dependent data.

An advanced variant of the RNN is the Long-Short-Term Memory cell (LSTM) [44] shown in Figure 2.5. The LSTM was specifically designed to address the challenges of capturing long-term dependencies in sequential data. Unlike traditional RNNs, LSTMs introduce internal gates for merging new information into its stored

cell state. The input gate allows addition of new information, the forget gate gives the option to remove information from the cell state, and the output gate controls the prediction of that timestep.

By regulating the flow of information through these gates, LSTMs provide better retention of long-term information than naive RNNs. This allows them to make accurate predictions even across long sequences. The LSTM a powerful extension to the RNN, and is used as a baseline in our work for integration of temporal information.

2.2.5 Transformers

Transformers [52] are attention-based neural networks that have dominated in the field of Natural Language Processing (NLP) [11].

They offer three powerful advantages over LSTMs:

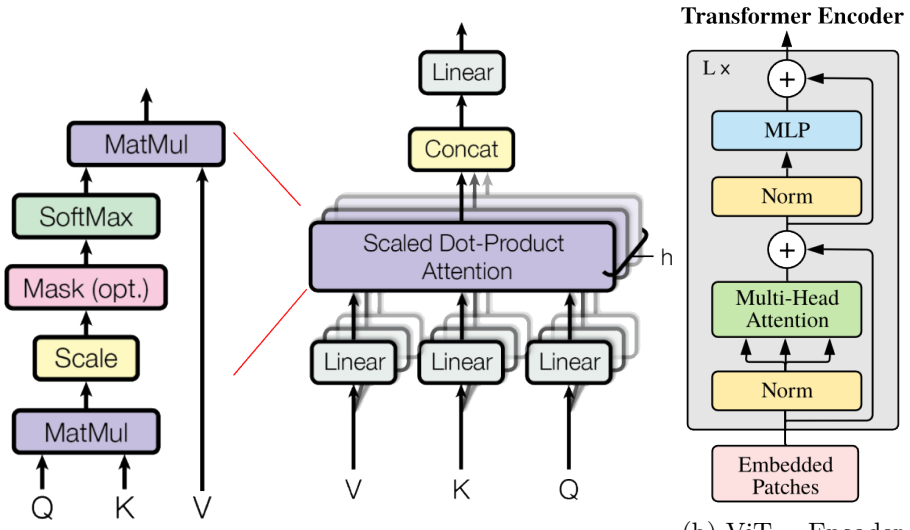
1. **Parallel Processing:** Instead of operating on data in a step-by-step sequential way, transformers process all inputs in a single step.

2. **Long-Range Dependencies:** Because of this parallel operation, transforms can directly reference any previous timestep. RNNs must encode and store information to be retained, causing problems if the future has a long dependency on the past.

3. **Adaptability:** Transformers are modular in nature, and can handle any sequence given to them. This allows us to explore using transformers on both spatial and temporal domains.

Self Attention

The core function a transformer performs is self-attention, which is shown to the left of Figure 2.6a. The objective for self-attention is to merge information from all other inputs denoted as tokens. These are used in update of the current token. We first derive a Query, Key, and Value from each token by linearly projecting each token with three different linear mappings. We use the Query and Key to create a set of attention scores, with each token gaining an attention score for itself and all other tokens. These scores are normalized by softmax, and control the merge of information. The merge is done through matrix multiplication between the attention matrix and the Values. As stated earlier, the self-attention operates on all tokens in parallel. This means, that the last token gets mixed with information from the first token, and vice-versa. A mask is optionally added to prevent the



(a) Multi-head Attention taken from [52]

(b) ViT Encoder taken from [12]

Figure 2.6: The ViT Encoder (b) performes Multi-Head Attention (a) over a sequence of input tokens

reverse connection, i.e. tokens can only attend to themselves and previous tokens, thus enforcing causality.

A point of implementation, is that we scale self attention relative to the feature dimension of the token. If a single token has shape d_k, we multiply by $\frac{1}{\sqrt{d_k}}$ before performing softmax. The reasoning for this is that as d_k grows, the dot product between the Key and the Query will also increase [52]. In large magnitudes, the softmax can be pushed into regions where it has extremely small gradients. The scale helps stabilize training in the self-attention block.

The overall equation for attention is given by:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}_{\mathbf{K}}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}. \tag{2.8}$$

Where in self-attention, the $\mathbf{Q}$, $\mathbf{V}$, and $\mathbf{K}$ (Query, Value, and Key respectively) are all different linear projections of the inputs.

Multi-Head Self Attention

The attention mechanism, while powerful, has a weakness. For each token there is only one attention map computed, bottlenecking the capacity to attend over different features. This is solved through Multi-Head self-Attention (MHA), shown to the right in Figure 2.6a. We chunk the input into h heads before passing it to

the linear encoding of the Key, Value, and Query. The input is re-formed after self-attention is performed on each head.

Positional Encoding

The transformer operates on all tokens, but has no way to distinguish their order. For example, we want the sequence of images from a video to be ordered according to their timestep. We learn an embedding for how the transformer should apply this order. It gets added to the tokens once, before being sent to the first transformer block.

ViT Encoder

A common transformer block is the ViT Encoder [12] shown in Figure 2.6b. The inputs and outputs to this block have the same shape, and are a list of tokens. MHA is performed over each token in the list, with a residual connection added at the output. An recurrent MLP forms an inverse-bottleneck on the d_k dimension of each token. These blocks are repeated L times.

We use the ViT Encoder heavily in our work and refer to it also as a Deterministic Transformer.

3 Related Work

In this section, we present background work done in our two evaluation tasks. The first, Video Prediction, uses a sequence of context frames from a video to predict the continuing frames. Much prior work has been done in video prediction and we defer to [40] for a more exhaustive background. We will then introduce the second task, Human Pose Prediction. This task is very similar to Video Prediction, except that images are replaced with a human skeleton. The objective is to continue the action of the human skeleton from the context motion.

3.1 Video Prediction

Video prediction is a self-supervised learning task of predicting future video frames conditioned on previous frames. It is self-supervised, because learning is done without manual annotation of the desired targets. Data takes the form of a sequence of images, and at each prediction timestep, we are asked for a pixel-wise forecast of the subsequent frame. In addition to the high-dimensional spatial dependency of image predictions, videos add temporal dependencies. Motion present in the conditioned sequence should be continued on to the prediction. This is an intractably hard problem, as motions deviate and outside elements enter the camera's field of view. As a result of the difficulty of this problem, networks which have strong performance in Video Prediction also tend to have strong performance in downstream tasks [54].

This is due to the representative power of networks trained on Video Prediction. Features are aggregated both spatially and temporally to create diverse spatio-temporal knowledge. In [54], the authors train on over 600 hours of publicly available television shows from Youtube. The features learned in their network are later used in action forecasting. Seo et al. [46] perform action-free pre-training on Video Prediction. They later add action conditioning and fine-tune there model with Reinforcement Learning. Other applications for Video Prediction include precipitation prediction [48], autonomous driving [26], and robotics [15]. Many domains leverage the power of self-supervised representation learning on video data. Below we review the most popular approaches to Video Prediction.

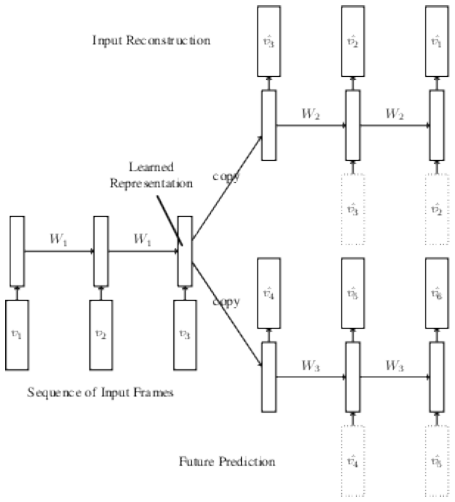


Figure 3.1: Figure from [51]. The labeled "Learned Representation" is a compressed encoding of all context frames. Using two different LSTMs, the top branch decodes this back into the original context frames while the bottom branch predicts the continuation of the video.

Traditional approaches relied on using domain specific engineering to extract salient information from videos. Optical flow, for example, is the displacement of pixels between two consecutive images in a video. Patraucean et al. [42] used an optical flow prediction module to capture motion changes in the video. Further work by Li et al. [35] use optical flow to predict multiple motions conditioned on a single frame. Concurrent work is being done on video prediction in the frequency domain. In the work by Farazi et al. [14], they formulate the translation between two images as a phase difference in the frequency domain. The Transform Model operates on this phase difference to predict the transformation for the next frame. These works use the difference between consecutive frames, either through optical flow or in the frequency domain, to simplify the learning objective for video prediction.

Early video prediction models were designed to make a single, deterministic approximation of the future. Srivastava et al [51] developed an LSTM-based model for deterministic continuation of the video, which is shown in Figure 3.1. The authors first make an encoded representation of the entire input sequence using an LSTM. This representation is used as a seed for frame-by-frame prediction of the rest of the video. They argue that prediction of the video continuation causes the encoder LSTM to learn good features, by forcing it to extrapolate the objects and motions present in the input sequence. Shi et al. extends on LSTM-based methods by introducing ConvLSTMs [47], which maintain spatial structure in the predictor.

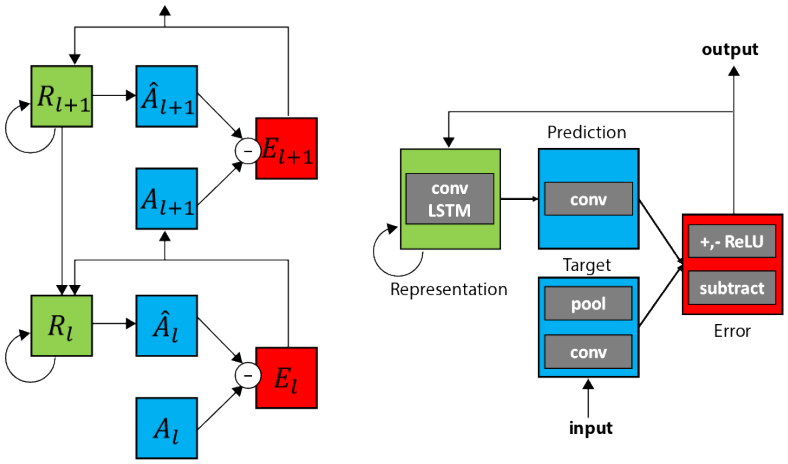


Figure 3.2: PredNet Architecture from [37]. Hierarchical design is shown on the left with bottom-up connections coming from the error signal "E". Top-down connections come from the recurrent network "R". The right shows specific modules used when PredNet is applied to Video Prediction.

Hierarchical models became popular with Lotter et al. [37]. The authors of this work introduced the Predictive Coding Network (PredNet), a hierarchical model with both bottom-up and top-down connections (shown Figure 3.2). At each layer, PredNet tries to predict what the input will be at the next timestep. Error between the prediction and input is propagated up the hierarchy of the network. The Recurrent Representation layer models temporal information and is passed down the hierarchy of the network. Many additional works have followed with use hierarchical designs [53].

Despite advances, these methods suffer from being fully deterministic, which only allows them to predict one possible future. This hurts their ability to generate sharp, long-term predictions [10] [5]. Mathieu et al. attempts to solve this problem by using a generator-discriminator architecture [39]. They design a discriminator to determine if the last frame in a sequence of frames is from the video prediction model or the ground truth data. However, the drawback to generator/discriminator models is that they are difficult to train [21] and are susceptible to mode collapse, which can remove variance in future predictions.

Stochastic models have demonstrated success in generating both sharp images and realistic variation in possible futures [17]. An early work in stochastic modeling by [56] used cross-convolutions to predict multiple possible futures given a single input frame.

Additional work has been done to separate the temporal information contained in a video from the spatial content. Franceschi et al. [17] introduce a Latent Residual Dynamic Model (SRVP). They condition the generation of each frame

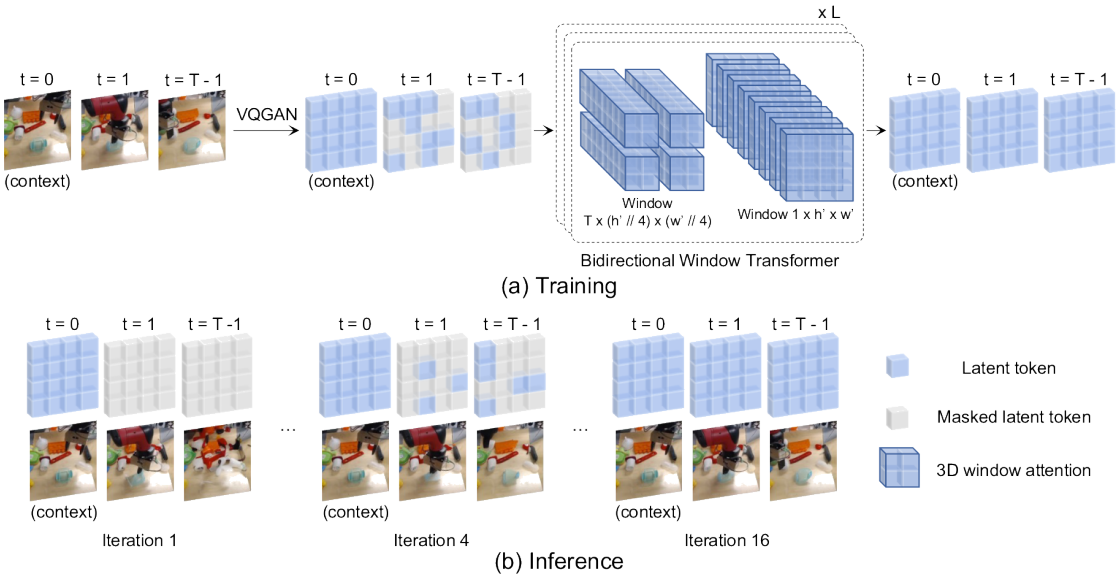


Figure 3.3: MaskViT Architecture and Training/Inference Procedure from [23]. (a) Training is performed by prediction of all masked tokens in a single step. Inference is performed in iterations with each iteration saving a fixed number of unmasked predictions (picked by confidence scores).

only on the current state of the predictor and a static background variable w. The internal state of the model is updated deterministically from the previous state. Each state is used to generate a stochastic sample which gets incorporated into the next state. This decouples the deterministic movements in the video from the variability of an uncertain future.

Success in Large Language Models has inspired others to unify successful methodologies across domains. In the case of Video Prediction, Gupta et al. [23] introduce the MaskViT shown in Figure 3.3. The MaskViT is a transformer-based predictor architecture which operates on the spatio-temporal domain. Images are encoded by a single-frame encoder to give a latent discretization of the input. These form tokens for the predictor, with each image given $h \times w$ discrete tokens. Using full attention on the spatio-temporal domain is prohibitively expensive, due to long prediction horizons. As an example, predicting 30 frames from a 16x16 feature maps requires attention over 7680 tokens. The MaskViT restricts attention to two separate windows through a Bidirectional Window Transformer. The first (spatial) window attends only to tokens in the current timestep. The second (Spatiotemporal) window performs 3D attention over decreased spatial dimensions. The windows are applied sequentially, to gain both local and global interactions in a single block.

Figure 3.4: 21 Joint Human Skeleton from the thesis [7]

Training and Inference is done on the MaskViT without auto-regression. Training masks a variable percentage of the predicted tokens and forces the Transformer to recreate the prediction from partial knowledge in one step. During Inference, all predicted tokens are masked and we slowly unmask them over multiple iterations. At a single iteration, the Bidirectional Window Transformer predicts all masked tokens, but only a fixed number (those with the highest confidence) are kept. The entire predicted sequence is unmasked by accumulating saved tokens at each iteration.

Taking inspiration from these related works, we designed our Stochastic Transformer (STTR). An addition source of inspiration comes from the Variational Transformer designed by [36] which is used in Dialogue Response Generation. We also implement and test an LSTM version of our model, which provides the background of our training procedure and is taken from Denton et al. [10]. We formally present this model, our architecture, comparison, and methodology in the next chapter.

3.2 Human Pose Prediction

The task of Human Pose Prediction is very similar in structure to Video Prediction. Instead of images, however, we are given and asked to predict the 3D joint positions of a single human skeleton. Figure 3.4 shows an example of the skeletal data structure.

In the same vein as Video Prediction, we are given a time series of context skeletons and tasked to continue the action performed by the skeleton. We now present a few background works that have shown success in this task.

Fragkiadaki et al. introduce the Encoder-Recurrent-Decoder (ERD) [16], an architecture framework for this task. The ERD consists of three parts and is shown in Figure 3.5. The Encoder takes a complete human skeleton and produces

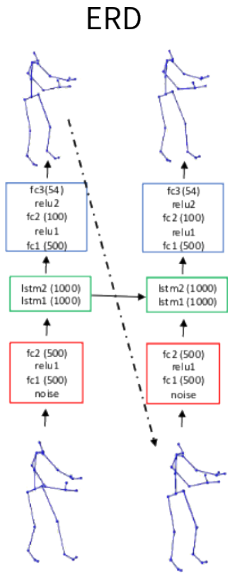


Figure 3.5: The ERD model [16] is an auto-regressive encoder-predictor-decoder architecture using a deterministic LSTM as a predictor.

a encoding, which summarizes the key features of the skeleton. The encoding is passed to the Recurrent layer, an RNN. Along with the current encoding, the RNN also receives the hidden state from the previous timestep, which conditions the prediction of the next step on all previous inputs. The RNN outputs an encoded prediction for the next timestep, that gets passed to the Decoder. The Decoder transforms the prediction into a human skeleton.

The ERD operates auto-regressively, meaning it feeds its prediction of the last timestep back into itself for prediction of the next timestep. This can cause the accumulation of errors in the predictions. A very strong insight from Martinez et al. is to reduce the complexity of the problem by predicting only the velocity of the skeleton, instead of the skeleton itself [38]. Since velocities are usually small and centered at zero, they are natural to express by deep learning architectures. This is achieved very simply by adding a recurrent connection from the current to the next prediction.

Further work by Gui et al. moves towards improving the realism of predictions. They propose a training procedure with two discriminators to regularize the loss [22]. As shown in Figure 3.6, the first of these is the fidelity discriminator. The fidelity discriminator identifies if a sample was produced by the model or if it came from the dataset. By feeding it only the prediction of new frames, the fidelity discriminator trains the model to generate realistic motion as seen in the dataset. The second of these is the continuity discriminator. The continuity discriminator checks if the context concatenated to the prediction still resembles continuous clips

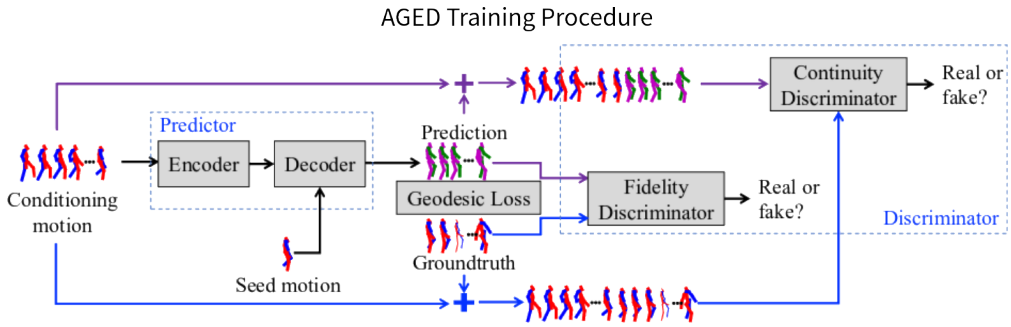


Figure 3.6: The AGED Training procedure uses two discriminators [22]. The first regularizes fidelity of the prediction by comparing only the against predicted poses. The second regularizes continuity in the continuation of the context by pre-appending the context to the prediction.

from the dataset. Their Adversarial Geometry-aware Encoder Decoder (AGED) model provides a standard baseline used in Human Pose Prediction.

4 Method

One application of our method is the Video Prediction task. Given N context frames $C_1, ..., C_N$ from a video, we are tasked to continue this sequence. Each frame is an image with $H \times W$ pixels. By conditioning on the context, we predict the next M frames $\hat{X}_1, ...\hat{X}_M$ and compare these to the continuing ground truth video $X_1, ...X_M$

We investigate multiple models for video prediction, and propose a novel module, STTR, which is shown in Figure 4.1. Our STTR predictor model generates a sample for one potential future using a learned distribution. This sample gets decoded into an image $\hat{X}_t$ that contains one possibility for the continuation of the context sequence.

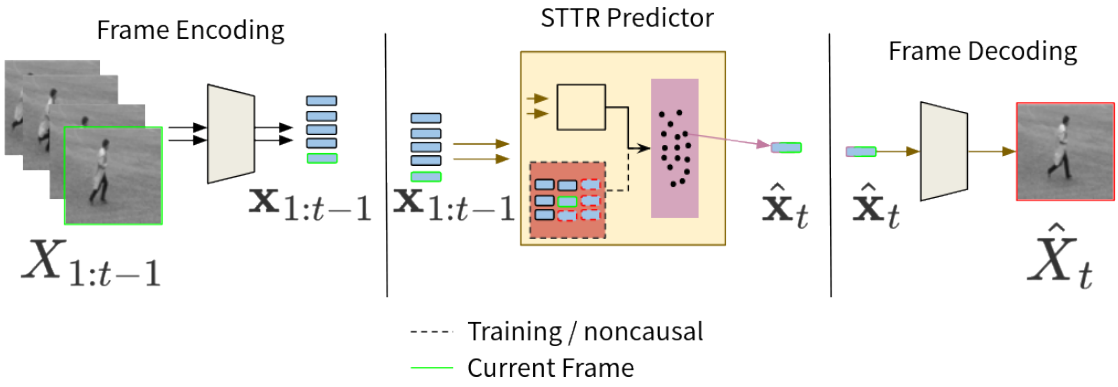


Figure 4.1: Stochastic sampling of the next frame using our STTR Predictor.

In Section 4.1, we introduce four different prediction models, including our proposed STTR. This follows with Section 4.2, where we apply these predictors to the Video Prediction task. In Section 4.3, we formulate Human Pose Prediction, highlighting similarities and differences to Video Prediction. Using Human Pose Prediction as an example, we propose a variation of our model that uses separate attention over the temporal and spatial domains. Finally, in Section 4.4 we discuss implementation details and training procedures

4.1 Predictor Models

The role of the predictor is to forecast features to future time steps. For the current timestep, we encode an image into a vector representation:

$$\mathbf{x}_{t-1} = \text{Enc}(X_{t-1}). \qquad \text{Figure 4.3} \qquad (4.1)$$

This gets passed to the predictor. The predictor has already seen every input leading up to the current, and uses this to predict the next features:

$$\hat{\mathbf{x}}_t = \text{Pred}(\mathbf{x}_{1:t-1}). \qquad \text{Figure 4.2} \qquad (4.2)$$

In the last step, the prediction is passed to the Decoder. The Decoder transforms the features into the predicted image:

$$\hat{X}_t = \text{Dec}(\hat{\mathbf{x}}_t). \qquad \text{Figure 4.4} \qquad (4.3)$$

This motivates the introduction of the Predictor. It is the only block that has access to past information, and the only block to make a prediction of the future.

We say that this predictor operates causally. If time was laid out on a timeline, the causal predictor could only see to the left of the current position. Our work also involves a non-causal predictor, that can see the whole timeline. We continue this section with the introduction of our four predictor models.

Determinstic LSTM

We begin with a simple deterministic LSTM (Det LSTM) predictor show in figure (Figure 4.2 a). The Det LSTM starts by warming up an uninitialized cell state. It is fed each context vector $C_{1:N-1}$ with the output of the LSTM discarded. This initializes the cell state with the content seen so far. At the last context frame C_N, we use the hidden state of the LSTM to predict the first frame $\hat{\mathbf{x}}_1$.

We feed each prediction into the Decoder to generate an image from the current prediction. In an auto-regressive way, we re-encode our predicted image to form the features of the next timestep. We update our timestep $t - t \leftarrow t$, and repeat the process until we have M Images $\hat{X}_{1:M}$.

SVG

In the paper Stochastic Video Generation by Denton et al. [10], the authors present their novel SVG model, which incorporates stochastic sampling into an

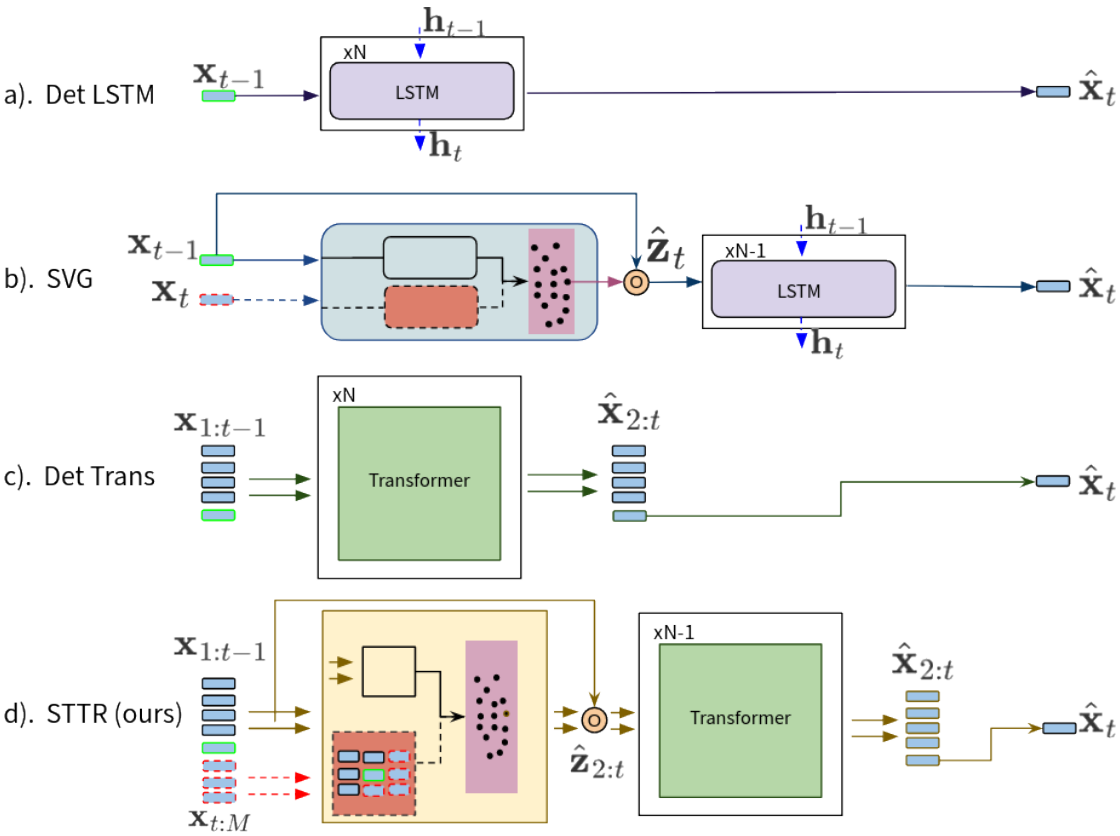


Figure 4.2: Depiction of the four prediction modules. The Det. LSTM. (a.) and Det. Trans. (c.) make a single, deterministic guess of the future. SVG [10] (b.) and Our STTR (d.) take a stochastic sample of the future learned by a noncausal teacher.

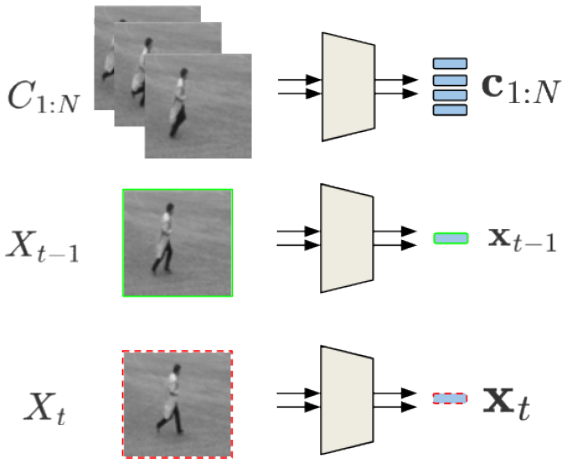


Figure 4.3: The Encoder compresses each image to a vector.

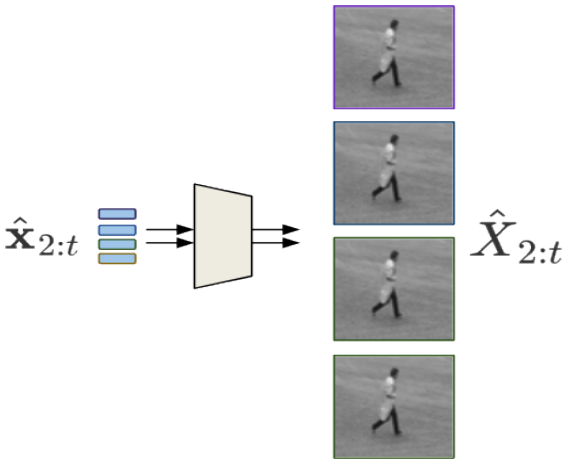


Figure 4.4: The Decoder extracts an image from a vector.

LSTM-based architecture. The SVG model enhances the LSTM by introducing a stochastic block.

A sub module for the SVG runs non-causally, and helps to guide future predictions. We can only predict a single future at a time, selecting a latent sample $\hat{\mathbf{z}}_t$ to convey that prediction. The non-causal branch learns to use this latent sample in the most efficient way, as it has already seen the future. This is used to train a causal branch, which tries to replicate the non-causal prediction. As it is an impossible task to guess the future, the causal branch must make predictions with enough variance such that the non-causal prediction is present in the distribution of guesses.

In the SVG architecture, the causal and non-causal branch are both LSTMs, φ and ϕ respectively. Each LSTM predicts the parameters of a distribution, from where the latent sample is taken. The causal $LSTM_\varphi$ is known as the prior, as it estimates $p(\mathbf{z})$ which is the distribution containing all possible futures. In reality, we do not know this distribution of all possible futures and the prior branch predicts $p(\hat{\mathbf{z}}_t|\mathbf{x}_{1:t-1})$. The non-causal $LSTM_\varphi$ predicts parameters for the posterior, an estimation $q_\varphi(\hat{\mathbf{z}}_t|\mathbf{x}_{1:t})$ of the true future.

We do this by running the posterior LSTM one timestep ahead of the prior. At the current timestep $t-1$, each LSTM predicts the parameters for a Normal distribution:

$$\mu_\varphi, \sigma_\varphi = \text{LSTM}_\varphi(\mathbf{x}_{1:t-1}), \tag{4.4}$$

$$\mu_\phi, \sigma_\phi = \text{LSTM}_\phi(\mathbf{x}_{1:t}). \tag{4.5}$$

During training, we sample the latent vector from the posterior. To prevent this sample from simply copying the entire content of the target frame, we constrain the posterior distribution to be close to the priors. This is done through the KL Divergence [32]:

$$D_{KL}(q_\varphi(\hat{\mathbf{z}}_t|\mathbf{x}_{1:t}) \parallel p(\hat{\mathbf{z}}_t|\mathbf{x}_{1:t-1})). \tag{4.6}$$

This enforces that the latent vector $\hat{\mathbf{z}}_t$ captures information not present in the previous frames.

The rest of the SVG focuses on abstracting unnecessary information out of the latent sample. We concatenate the current frame encoding by

$$\hat{\mathbf{z}}_t = \text{cat}(\mathbf{x}_{t-1}, \hat{\mathbf{z}}_t), \tag{4.7}$$

so that the latent sample does not need to contain any image specific information. The prediction is then post-processed so the latent prediction is mixed with the

content of the previous image. Post-processing is done by the second Det-LSTM in Figure 4.2 (b).

During inference, we discard the non-causal posterior, and instead take many samples from the prior. We use the sample with the best performance when compared to the ground truth.

Deterministic Transformer

Before we explain how we apply this LSTM stochastic sampling technique to a Transformer architecture, we first introduce a Deterministic Transformer (Det Trans) predictor (Figure 4.2 c).

Unlike a RNN, this ViT-based predictor has no internal state that gets passed between timesteps. Instead, it operates on all previous frames to predict the next. For the timestep $t = 1$, we use attention over the embedding of all context images $\mathbf{c}_{1:N}$ to predict the encoding for $\hat{\mathbf{x}}_1$. We abstract this context notation away, instead saying we predict $\hat{\mathbf{x}}_{2:t}$ using previous frame encodings $\mathbf{x}_{1:t-1}$ We discard redundant predictions, using only the last token for decoding of image $\hat{X}_t$ As with the previous predictors, this is fed back into the input in an auto-regressive prediction manner.

Stochastic Transformer Predictor

We introduce the Stochastic Transformer Predictor, the core module of our STTR architecture. We take inspiration from the Sequential Variational Transformer (SVT) proposed by Lin et al. [36]. This work uses non-causal, multi-head attention for diverse dialogue response. We now apply this method in a Predictor-based architecture.

Similar to the SVG, our STTR contains two branches; a causal prior, and a non-causal posterior. The non-causal branch is shown to the left in Figure 4.5. This branch performs MHA over every token in the video, and so it is able to mix future content into the latent sample. Unlike the SVG which only uses a single target frame in the prediction of its latent vector, we use all future frames. This allows us to enrich the latent space with a more informative future.

The causal branch is shown to the right in Figure 4.5. This branch predicts a distribution over possible futures using only current information. The parameters for this distribution are learned to approximate the posterior. Using KL Divergence to enforce this approximation, our STTR prior learns a distribution over possible futures.

Parameters μ and σ are used to form the Normal distribution, where the latent sample is drawn. Each branch predicts these parameters separately, through a sampling module. The MHA predicts a token for each frame, but we just use

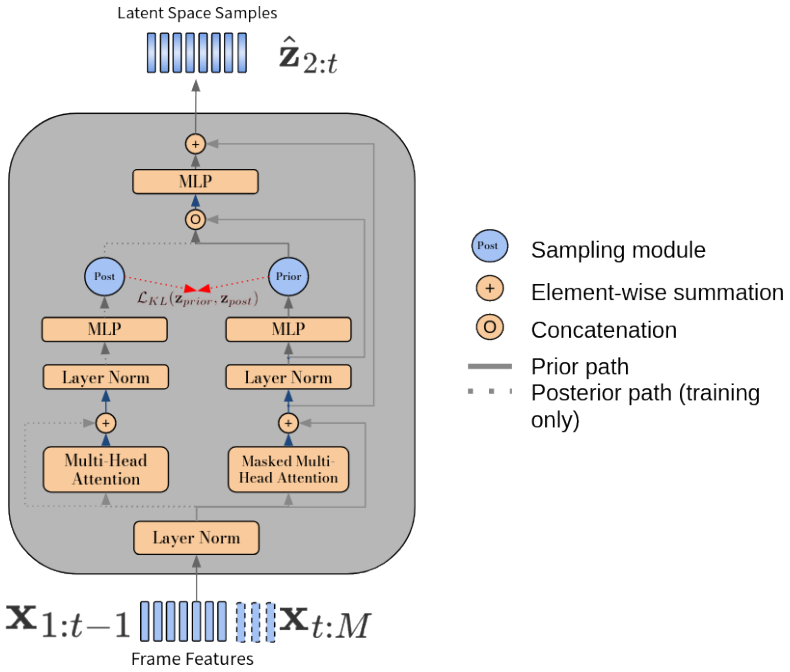


Figure 4.5: Our STTR Sampling block creates a latent vector $\hat{\mathbf{z}}_t$ carrying stochastic information about the next frame. The left (posterior) path is used only during training and samples from the noncausal distribution. The right (prior) path is used in training and evaluation to generate a sample of one possible future. We apply a KL Loss between the distributions to train the prior to approximate the posterior. This training enforces that the prior learn the distribution of multiple possible futures.

the token of the current frame as the input. The hyperbolic tangent restricts this to be within $[-1, 1]$. A linear layer projects it into a μ and σ. Using the reparameterization trick [30], we sample the distribution.

During inference, we discard the posterior. Since this is the case, we make residual connections from the prior. The first of these goes to concatenate the latent sample, which has been repeated for each token. The second forms the residual connection for the output of the block, which is commonly used to pass gradients in very deep neural networks [24].

The rest of the STTR is done by a Deterministic Transformer. Our prediction is fused with deeper features from the frame encodings.

4.2 Video Prediction

The main focus of our work is applied to Video Prediction. Given context images $C_{1:N}$, we want to predict the continuation of the sequence $X_{1:M}$. We keep the Encoder and Decoder the same, substituting various prediction architectures. The prediction of each models is decoded into images $\hat{X}_{1:M}$.

During training, we generate a supervisory signal through a frame-wise comparison between the ground truth and model predictions. Using Mean Squared Error (MSE) as the reconstruction loss gives:

$$\mathcal{L}_{Recon} = \frac{1}{M}\sum_{t=1}^{M}\|X_t - \hat{X}_t\|, \tag{4.8}$$

where $\|\cdot\|$ is the l_2 norm. This is averaged over every element in a minibatch.

The supervisory signal is backpropagated through the network to update the weights of each sub-module in an end-to-end manner. This is incomplete in the case of stochastic modules. Samples that are taking from the posterior branch are used during training and no samples from the prior branch are decoded. The causal branch uses a second supervisory signal, the KL Loss, to update its gradients.

The Kullback–Leibler Divergence (KLD), is an asymmetric loss used to to compare the difference between two probability distributions $p(x)$ and $q(x)$. Both the causal and non-causal branch predict a distribution, under which a sample is taken to predict the next frame. We use the KLD to enforce similarity between the distributions predicted in the causal and non-causal branches. This enforces that the prior branch matches the distribution output by the posterior, and is able to learn

a distribution over multiple possible futures. The KLD Loss term is:

$$\mathcal{L}_{KLD} = D_{KL}(q_\phi \| p_\varphi). \tag{4.9}$$

The trade-off for prioritizing the reconstruction loss over the distribution loss is controlled by a parameter β:

$$\mathcal{L} = \mathcal{L}_{Recon} + \beta \mathcal{L}_{KLD}. \tag{4.10}$$

When β is too small, the model ignores the distribution-based loss signal and focuses purely on reconstruction of the ground-truth. To achieve this, it over-uses the non-causal distribution prediction which then contains extremely precise information about the future. The causal branch can not hope to match the detail contained in the distribution, and fails to learn relevant features. During training the reconstruction loss will go to zero while the distribution-based loss skyrockets. During evaluation the model fails to make a salient prediction.

When β is too large, the model optimizes the distributions in the most direct way possible. It predicts a finite distribution, regardless the content of the video. This causes the KLD to drop to zero and the latent distribution does not contain any information about the future.

Through balance of the reconstruction and distribution-based losses, our STTR predictor is able to generate diverse and accurate predictions of the future. This is shown in Figure 4.6, with comparisons between the best, worst, random, and non-causal predictions from our model, all of which are plausible.

4.2.1 Encoder

The encoding procedure is shown in Figure 4.3.

Each image is encoded to a vector by a CNN At each layer in the CNN, the spatial dimension of the input is decreased and the feature dimension increased. At the final layer, we use global pooling over the spatial dimensions to output a one dimensional vector containing high level features from the content of the image.

4.2.2 Predictor

Each predictor is discussed in section 4.1. These are used to aggregate information over previous timesteps and make a prediction of the feature vector at the next timestep.

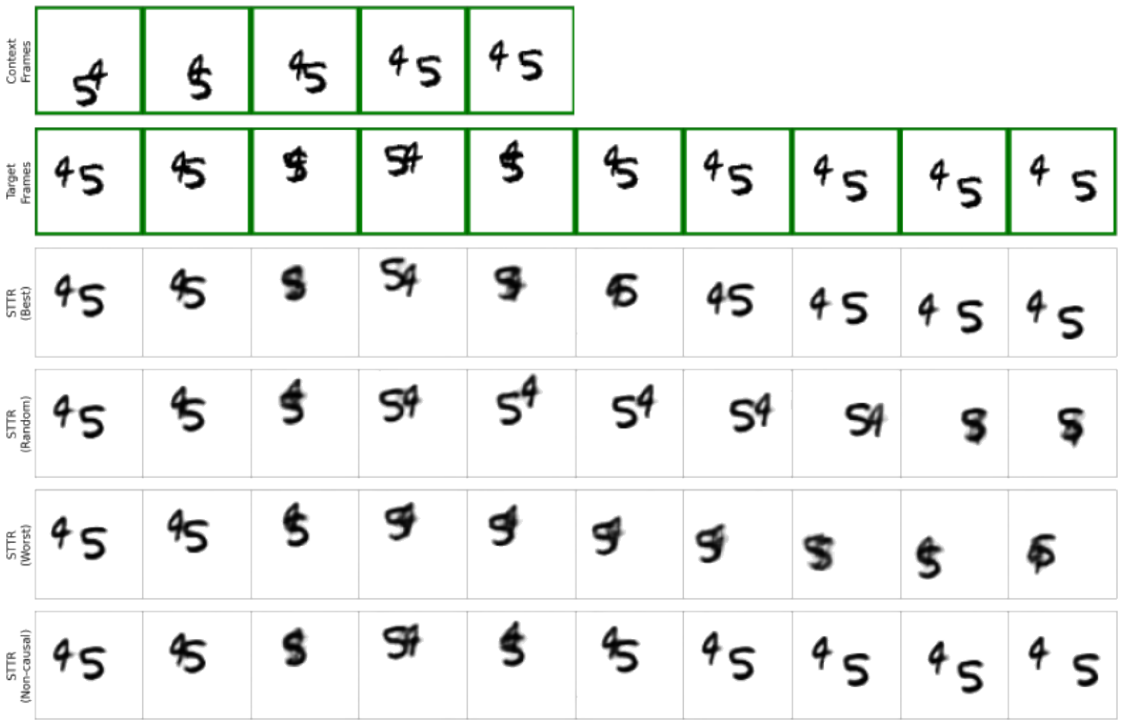


Figure 4.6: Variation from the dataset is captured by our STTR module on a sequence from Stochastic MNIST. As shown in the context frames, the numer '4' and '5' are separating. Due to randomness injected into the motion, they abruptly change directions in the first target frame. Our STTR makes several predictions about the direction of movement, and is able to replicate the stochasticity in the best sample.

As this is the only portion of the Video Prediction model which operates on the temporal domain, it is the key to synthesizing logical continuations of the video. Improvements to the predictor result in:

- Accurate modeling of motion in the predicted sequence.
- Continuation of the action performed in the context.
- Variety of predictions from the stochastic modules.

4.2.3 Decoder

The decoder takes the prediction at the current timestep and outputs the predicted frame $\hat{X}_t$. This is done through a CNN, which is a mirrored copy of the encoder. Pooling layers are replaced by spatial up-sampling and convolutions are replaced with Transposed Convolutions.

The decoder also contains skip-connections that come from the encoder. These bypass the predictor layer and are used to retain back-ground information in the image. The skip-connections allow the predictor layer to focus more on motion and the difference between the last-seen frame and the new prediction.

4.3 Human Pose Prediction

Human Pose Prediction is an interesting problem for three reasons: human poses abstract away background information present in images; longer term predictions are more possible than with videos; and there is a large variation in what could be considered valid motion.

By formatting the input as human skeletons, we remove a lot of background information irrelevant for prediction. This is a way of simplifying the task and allowing us to highlight the advancement of our predictor.

This task is further useful for testing the long-term capabilities of our predictor. A distinct advantage of transformers is their ability to maintain long term temporal dependencies, which we can showcase with Human Pose Prediction. If we try to forecast, for example the walking motion, on videos, then we will at some point run into the issue of the person walking out of the frame. This is why we turn to Human Pose prediction for generating long sequences.

The last reason we evaluate using Human Pose Prediction, is because there is a much larger variation in what could be considered normal motion. In terms of inverse kinematics, there are nearly infinite ways to the human skeleton to arrange itself and still pick up an object on the ground.

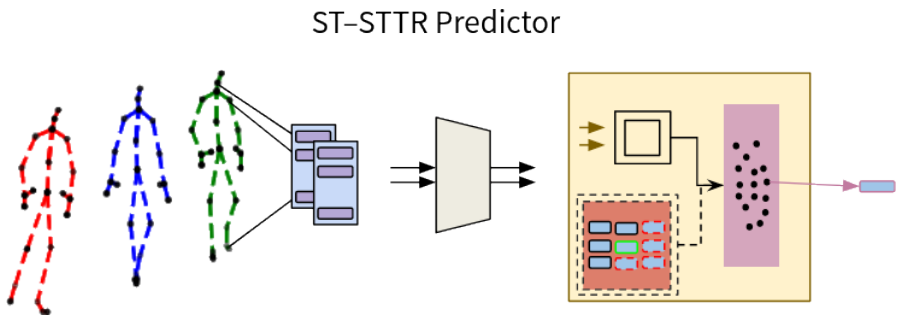


Figure 4.7: Our STTR Predictor module modified for Human Pose Prediction. Attention blocks are replaced by ST- Parallel Attention

In this section, we expand our video prediction model to the task of Human Pose Prediction. Taking the parallel attention mechanism from Aksan et al.[4], we introduce the Spatio-Temporal STochastic TRansformer (ST-STTR) shown in Figure 4.7. The ST-STTR is a predictor which performs separate attention over spatial and temporal modalities.

Problem Formulation

We formulate the problem of Human Pose Prediction with relation to Video Prediction. Instead of being an image with the shape $X_t \in \mathbb{R}^{H \times W \times Chan}$, each timestep is a human skeleton with J 3 Dimensional joints. Each joint is parameterized by their (x, y, z) coordinates in Euclidean space.

Given N context poses, we are asked to predict the next M poses of continued motion $\hat{X} \in \mathbb{R}^{M \times J \times 3}$.

Mean Per Joint Positional Error

We now introduction the loss metric used during training and evaluation. The Mean Per Joint Positional Error (MPJPE) averages the distance of each predicted joint from its ground truth corespondent. This is given by the equation:

$$\mathcal{L}_{Recon} = \frac{1}{J \times M} \sum_{j=1}^{J} \sum_{t=1}^{M} \|\hat{x}_t^{(j)} - x_t^{(j)}\|. \tag{4.11}$$

Where: $\|\cdot\|$ is the l_2 norm; $x_t^{(j)}$ is the ground truth position of joint j at timestep t; and $\hat{x}_t^{(j)}$ is the model prediction at the same point.

4.3.1 Spatial Temporal Transformer

The encoder for our Human Pose Prediction model is a single linear layer projection of all joints. This projection is simply:

$$\mathbf{x}_{t-1} = W_{enc} X_{t-1} + b_{enc} \tag{4.12}$$

where $W_{enc} \in \mathbb{R}^{J*dim \times J*3}$. We rearrange the encoding such that $\mathbf{x}_{t-1} \in \mathbb{R}^{J \times dim}$.

As a deterministic baseline for our model, we implement and test the Spatio-Temporal Transformer (ST-Transformer) from [4]. The architecture is outlined in Figure 4.8b.

The key difference between the ViT Encoder Transformer block and the ST-Transformer is the parallel attention over spatial and temporal modalities. Spatio-temporal attention is shown in Figure 4.8a.

In spatial attention, the transformer can attend to any token at the current timestep. At this timestep, self attention is performed over joints in the human body. Because knowledge of the right ankle is extremely relevant for prediction of the right foot, we decouple this modality from the temporal domain.

Similarly, temporal attention is decoupled from the spatial domain. For prediction of the next timestep, the joint is able to examine its position in the current and previous timesteps. In a cyclical motion like walking, it makes sense to allow strong temporal focus of the joint to its previous states.

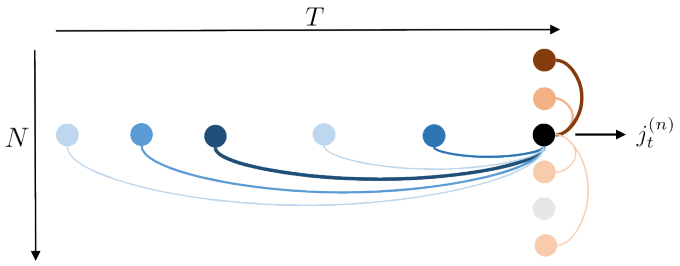
We implement the ST-Transformer as a deterministic baseline for comparison of our model. We detail in the next section how we can expand our STTR with this parallel attention mechanism.

4.3.2 ST-STTR

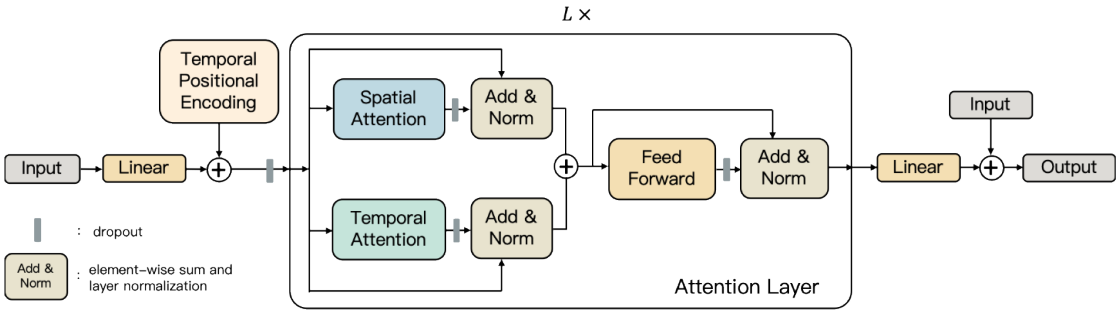
With minimal modifications, we adapt our STTR to use the Spatio-Temporal attention. This leads to the Spatio-Temporal STochastic TRansformer model (ST-STTR).

We replace each MHA module in the STTR with its equivalent parallel attention. Tokens are passed through the STTR having both a temporal and spatial dimension. After separate, parallel attention, the two modalities are combined through addition as in [4].

Miner modifications are also made to the sampling module. We combine all joints in a frame by stacking them along their feature dimension. The hyperbolic tangent and linear layer then project this to a μ and σ. The latent sample is repeated, in addition to every timestep, also for every joint.



(a) **Spatio-temporal Attention** [4]. Spatial attention (beige) attends to all other joints at the current timestep. Temporal attention (blue) attends to the current joint at previous timesteps.



(b) **ST-Transformer Architecture** [4]. Attention is performed in parallel over the spatial and temporal modalities.

4.4 Training and Implementation Details

In this section, we present out hyper-parameter selection and implementation details of our networks. All models are implemented in Pytorch [41] using Adam optimizer [29] with $\beta = (0.9, 0.999)$.

Video Prediction Encoder/Decoder

Input images are 64×64 grayscale for all datasets. For the encoder and decoder, we use DCGAN-like [43] and VGG-like [49] architectures. Implementation was done by Denton et al. [10] in their public github repository[1].

Skip connections are placed between each features map of the encoder and decoder. We save the feature maps from the last context frame, and use them to decode all predicted frames.

Hyper-parameters for the STTR (Stochastic Transformer) are in table 5.1.

The **KL Weight** is parameter β in Equation 4.10. Additionally, we use KL Annealing [8] to more heavily weight the reconstruction loss during the first several gradient update steps.

[1] https://github.com/edenton/svg

Parameter	STTR (Stochastic Transformer) (S)MMNIST / KTH / H3.6M
Batch Size	256 / 100 / 64
Total Gradient Update Steps	3e5 / 2e5 / 6e4
Encoder /Decoder	DCGAN / VGG16 / FC
Framerate	NA / 25 FPS / 25 FPS
LR	1e-3 / 3e-4 / 1e-4
LR Warmup	250 / 2500 / 2500
LR Burnout	0 / 50000 / 10000
KL Weight	1e-4 / 1e-6 / 1e-3
KL Annealing	250 / 5000 / 2000
dim	128 / 128 / 128
depth	5 / 5 / 4
zdim	10 / 32 / 10

Table 4.1: Hyperparameters Selection for the STTR across our tested datasets.

We linearly increase the learning rate for **LR Warmup** steps and decrease it to zero for the last **LR Burnout** steps.

For model hyper parameters:

- **dim** is the encoded feature dimension.
- **depth** is the number of repeated deterministic transforms after the STTR block.
- **zdim** is the dimension of the latent sample.

5 Experiments

In this section, we present experimental results for comparison of our STTR Model. The quantitative evaluation is performed on an unseen test set which was removed from the training and validation. Following [17] [23], stochastic models make 100 predictions and pick the prediction which is most similar to the target. We further expand insight into our predictor through detailed qualitative examples.

This chapter continues as follows:

5.1 **Datasets**: We introduce both synthetic and real datasets, test splits, and generalizations behind the data.

5.2 **Video Prediction**: We present our metrics used for evaluation and further discus result on the Video Prediction task.

5.3 **Human Pose Prediction**: We give results for Human Pose Prediction and capacity for long-term pose generation.

5.1 Datasets

5.1.1 Moving-MNIST

Moving-MNIST [51] (M-MNIST) is a commonly used synthetic dataset, typically to verify a proof of concept. The dataset comprises sequences showcasing two handwritten digits from the MNIST [33] dataset in motion. Each digit is given a random starting position, and placed in a 64×64 pixel frame. The digits move deterministically from their pre-defined position with a constant speed, changing direction when they encounter a border so that they bounce off the image boundary.

Images are grayscale, having a single channel dimension. We generate training sequences on-the-fly, by randomly sampling digits from the train split in MNIST. For testing, we use 10,000 sequences based on digits from the MMNIST test split. In both training and evaluation, we condition our models on 5 frames and predict the next 10.

To successfully predict the next frames in the video, the model must learn the shape of each digit from the context frames, as well as their direction and speed. Identifying when the digit will bounce off of the side of the frame is difficult, as even though this is deterministic, the handwritten MMNIST digits are padded to the same length. The model must also successfully disentangle overlapping digits, which requires recalling elements from the context.

5.1.2 Stochastic Moving-MNIST

We further test our models by injecting randomness into the Moving MNIST dataset and refer to it as Stochastic Moving MNIST (S-MNIST). While sharing the same name as the dataset used in [10] and [17], we inject randomness in a different way. Every 5th frame, we randomly select a new velocity for the digits (while maintain the same direction). Thus in the 5 conditioned frames, the last frame has positions of the digits calculated based off of the new, randomly sampled velocity.

The additional challenge in this dataset is that stochastic models must learn a large variance on the 5th frame, and then continue from this sample deterministically for 5 more frames. We demonstrate the effectiveness of our model by plotting the variance over time.

5.1.3 KTH Actions

We further extend our methods to evaluation on real videos. The KTH Actions dataset [45] consists of 25 people performing 6 actions (walking, jogging, running, boxing, hand waving, and hand clapping). Videos are taken over homogeneous backgrounds using a static camera running at 25 frames per second. We down sample each image from 160×120 to 64×64 and remove the color dimension.

For training/evaluation procedure, we follow the methodology used by [17]. We condition our model on 10 frames and during training predict the next 10 frames ($10 \rightarrow 10$). During evaluation, we predict the next 30 frames ($10 \rightarrow 30$). This shows how the model is able to make predictions outside the training horizon.

We remove the last 5 persons in KTH Actions from the training/validation and use them only for testing. We test by generating 1000 random evaluation sequences using a fixed random seed[1].

[1] https://github.com/edouardelasalles/srvp/blob/master/preprocessing/kth/make_test_set.py

	MovingMNIST 5→10			Stochastic MMNIST 5→10			KTH 10→30		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Det-LSTM	21.06	0.891	0.090	—	—	—	21.62	0.812	0.122
Det-Trans	21.02	0.883	0.092	17.70	0.684	0.355	<u>23.34</u>	0.825	0.116
Stoc-LSTM	**23.60**	**0.941**	**0.043**	**19.26**	**0.852**	**0.063**	**24.16**	<u>0.838</u>	**0.094**
Stoc-Trans	<u>21.57</u>	<u>0.896</u>	<u>0.082</u>	<u>17.88</u>	<u>0.803</u>	<u>0.112</u>	22.70	**0.841**	<u>0.095</u>

Table 5.1: Video Prediction Collective Quantitative Results. Our model gives competitive performance on more realistic settings when compared with nearly identical training procedures and minimal hyper-parameter tuning.

5.1.4 Humans 3.6M

For the task of human pose prediction, we perform evaluation on the dataset Humans 3.6M (H36M) [28]. This dataset is widely used in Human Pose Prediction and consists of a large (3.6 million) number of 3D human poses. H36M contains 7 actors performing 15 different actions (e.g. *Walking*, *Eating*, *Phoning*, ...). Following previous work [50], We us subject 11 (S11) for validation and subject 5 (S5) for testing. The rest of the subjects are used for training.

The skeletons of the actors are represented using 32 joints. We translate the skeletons so that the root joint is fixed at the zero position. The root joint is then removed, along with redundant joints, to create a 22-joint skeleton for training/evaluation. The original data is in the form of an exponential map. Using forward kinematics[2], we transform the skeletons to (x, y, z) coordinates in Euclidean space. Poses are captured at 50FPS, but we drop every second to reduce the frame rate to 25FPS.

During both training and evaluation, we condition on 10 frames and predict the next 25 ($10 \rightarrow 25$). By seeding motion every 10'th frame of a captured video, we create 256 test sequences per action.

5.2 Video Prediction

In this section, we present results over our Video Prediction models. Results are summarized in Table 5.1.

[2] `https://github.com/FraLuca/STSGCN/blob/main/utils/forward_kinematics.py`

5.2.1 Metrics

We make quantitative and qualitative comparisons against the ground truth using three statistics; Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM) [55], and Learned Perceptual Image Patch Similarity (LPIPS) [57]. These statistics operate on images individually.

PSNR

The first metric, Peak Signal-to-Noise Ratio (PSNR), has an inverse relationship to Mean Squared Error (MSE) and is calculated on a logarithmic scale. For a predicted image $\hat{X}$ and ground truth image X each with shape $H \times W$, the MSE and PSNR are calculated:

$$MSE = \frac{1}{HW} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \left[\hat{X}[i,j] - X[i,j]\right]^2 \tag{5.1}$$

$$PSNR = 10 \log_{10} \frac{I^2}{\text{MSE}} \tag{5.2}$$

where I is the maximum pixel luminance value (e.g. 255 for 8-bit representation). PSNR alone however, can be misleading. A study done by [27] demonstrates this by comparing two videos with the same PSNR having different subjective quality scores.

SSIM

A more perceptual-based method is Structural Similarity (SSIM) [55]. SSIM places importance on various perceptional phenomenon, for example luminance masking. Luminance masking is the phenomenon where distortions in the image are more difficult to perceive in bright regions. SSIM further considers nearby pixels to have a high level of dependency and that regions in the image should be accounted for when computing the similarity between two images. SSIM is a common evaluation metric which correlates with perceived similarity.

LPIPS

Further improvements to determining the realism of image frames is made by the introduction of the Learned Perceptual Image Patch Similarity (LPIPS) [57]. LPIPS evaluates the perceptual distance between two images (or 64x64 image patches) by comparing the scaled l_2 distance of the features. The authors of [57] argue that the features space of pretrained deep networks such as VGG provide

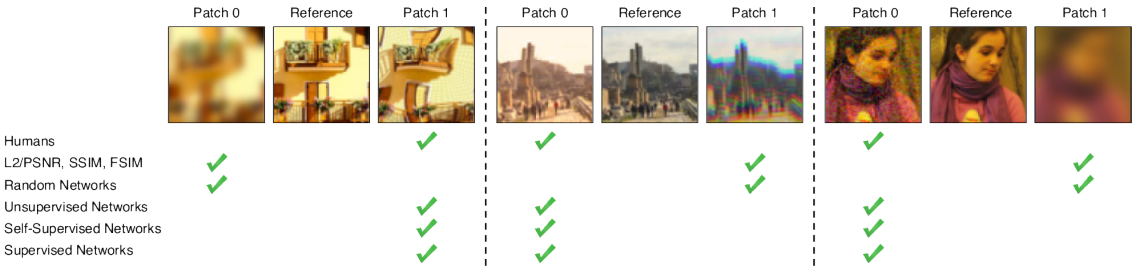


Figure 5.1: Which image is more similar to the reference? Evidence that deep features in pretrained models provide more realistic image comparisons.

a better perceptual loss than shallow functions such as PSNR and SSIM. As [57] show in Figure 5.1, both SSIM and PSNR prefer more blurry or otherwise distorted images than what humans considered to be a better fit.

We emphasize more perceptually based loses and highlight that an advantage to stochastic prediction is sharper images, where-as more blurry predictions may prefer metrics like PSNR.

5.2.2 Moving-MNIST

Our quantitative results for the Moving MNIST (M-MNIST) dataset are reported in Table 5.2. While the deterministic models are comparable, our STTR is outperformed by the SVG. We highlight in the table, that the posterior does not give strong improvements. In the toy dataset, neither model is very successful on integrating randomness into their sampling process. This is further shown by taking a single prediction from the prior, without seeing too large of a drop in performance. Stochasticity is not significantly shown on the deterministic Moving-MNIST dataset by either of our stochastic models.

In Figure 5.2, we show a particularly difficult Moving-MNIST example. Digits in the context sequence are very difficult to disentangle. The deterministic models show blurriness in their predictions, while the stochastic methods give a sharper image. The SVG is able to correctly split the digits and makes the most accurate prediction.

5.2.3 Stochastic Moving-MNIST

The stochastic version of the Moving-MNIST (S-MNIST) dataset gives a nice opportunity to evaluate diversity of stochastic samples. Every 5th frame, the speed of the digits change while direction is maintained. Changing only the speed of the digit makes the randomness more subtle in this variant of S-MNIST. The quantitative results are reported in Table 5.3.

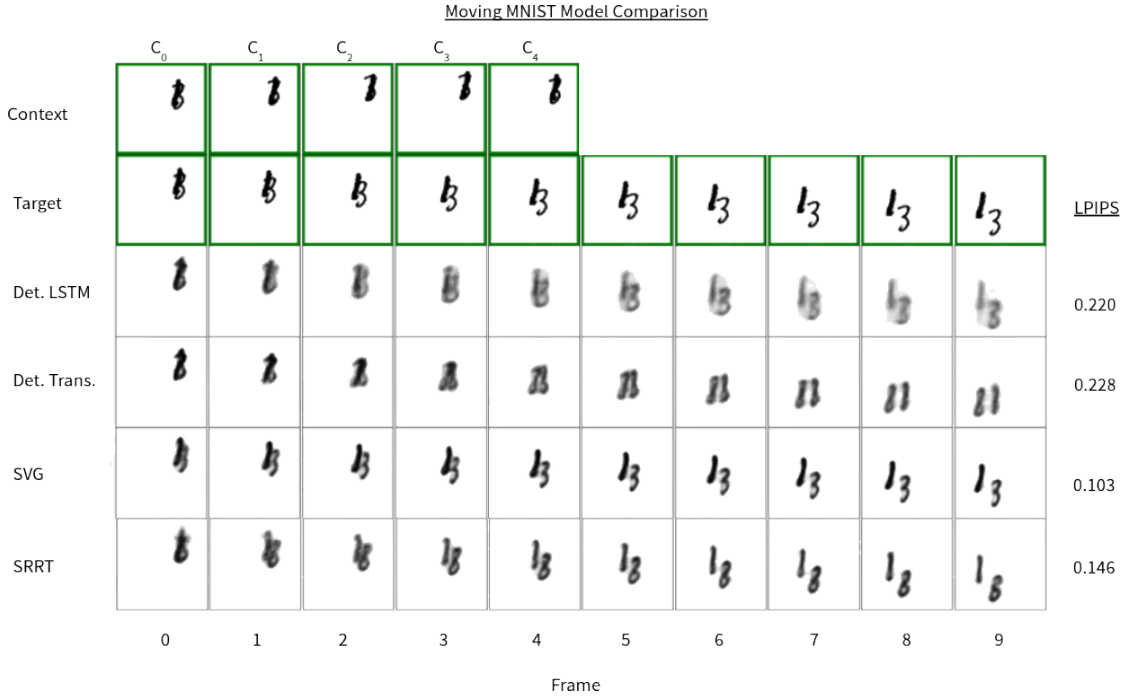


Figure 5.2: A difficult example from Moving-MNIST. The context frames contain both overlapping digits and similar motion patterns. The SVG successfully disentangles the digits.

MNIST Quantitative Results

	PSNR↑	SSIM↑	LPIPS↓
Det-LSTM	21.06	0.891	0.090
Det-Trans	21.02	0.883	0.092
SVG 100 Pred	**23.60**	**0.941**	**0.043**
SVG 1 Pred	—	—	0.054
SVG Posterior	—	—	0.044
STTR 100 Pred	21.57	0.896	0.082
STTR 1 Pred	—	—	0.093
STTR Posterior	—	—	0.084

Table 5.2: Our prediction models applied to the deterministic Moving-MNIST dataset. In addition to the normal prediction of 100 Samples, we also test making a single prediction from the stochastic models (in "... 1 Pred"). We further report results using the non-causal posterior branches of the stochastic models.

SMNIST Quantitative Results

	PSNR↑	SSIM↑	LPIPS↓
Det-Trans	17.70	0.684	0.355
SVG	**19.26**	**0.852**	**0.063**
STTR	17.88	0.803	0.112

Table 5.3: The Quantitative results of four models tested on Stochastic Moving-MNIST

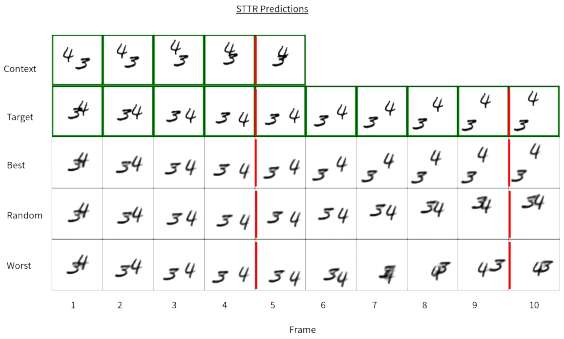


Figure 5.3: Predictions on the DIR-SMMNIST Dataset. Change occurs in the last context frame, which is continued for all predictions until Frame 5. Direction changes again and variance is shown in the Best/Random/Worst Predictions

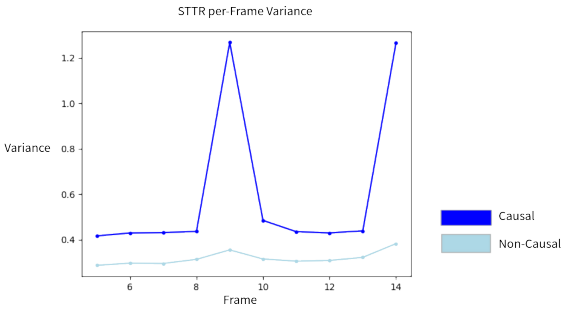


Figure 5.4: At every 5 frames the direction changes in DIR-SMMNIST. This corresponds to strong variance in the prior distribution.

To more cleanly illustrate the variance of our predictions, we simplify the stochasticity. In Dir-SMMNIST, we change the direction of the digit instead of the speed, which stays constant. At every 5 frames our STTR gives a spike in variance as shown in Figure 5.4, thus indicating that the STTR has learned the underlying stochasticity of the data. Also, note the lack of any variance spike in the posterior as it has non-causal knowledge of the future. This is shown next to a qualitative example of DIR-SMMNIST in Figure 5.3.

5.2.4 KTH Actions

In this section, we make qualitative analysis over sequences selected from the KTH dataset. Through these, we will show our models latent space capacity by comparisons of best/worst predictions. We also draw examples from the posterior, which maximizes potential of the latent sample.

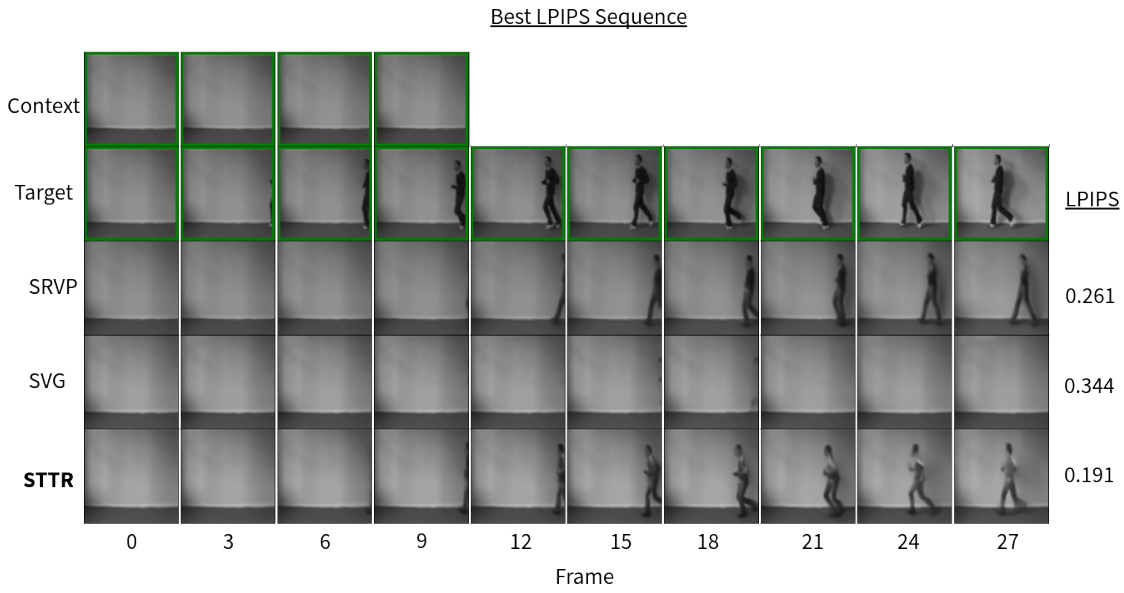


Figure 5.5: Blank-context prediction of stochastic models. We take the best LPIPS Score out of 100. The SVG does not make a person appear in this scene (on the indoor background). The STTR (Ours) and SRVP [17] successfully predict the appearance of a person.

Quantitative results are given in the earlier Table 5.1 and Framewise LPIPS scores are shown in Figure 5.7. Our STTR perform very similarly in per-frame LPIPS to the SVG. The Deterministic Transformer outperforms its LSTM counterpart.

The KTH Dataset is filmed with a static background and an unmoving camera. When we feed context images containing only the background, we investigate the sampling capacity of the prior distribution. Predictions from three stochastic modules are shown in Figure 5.5. In this situation, the SVG fails to predict the appearance of a person, while our STTR and the SRVP [17] succeed.

Our next example in Figure 5.6 highlights the expressive power of the STTR latent vector. Samples were sorted according to their mean LPIPS score. We display the best, worst, a random and posterior samples. Each latent vector makes a unique prediction of the target. The worst sample successful generates a person, but does so too late and is out of sync with the target. The result is it having the worst LPIPS score among 100 predictions, despite being a reasonable guess.

As a small ablation, we restrict the max number of tokens that the transformer can attend to. We hypothesized, that the positional embedding is difficult to extrapolate past the training prediction horizon. This is confirmed in Figure 5.8 by allowing the transformer to attend over all previous tokens, and giving worse

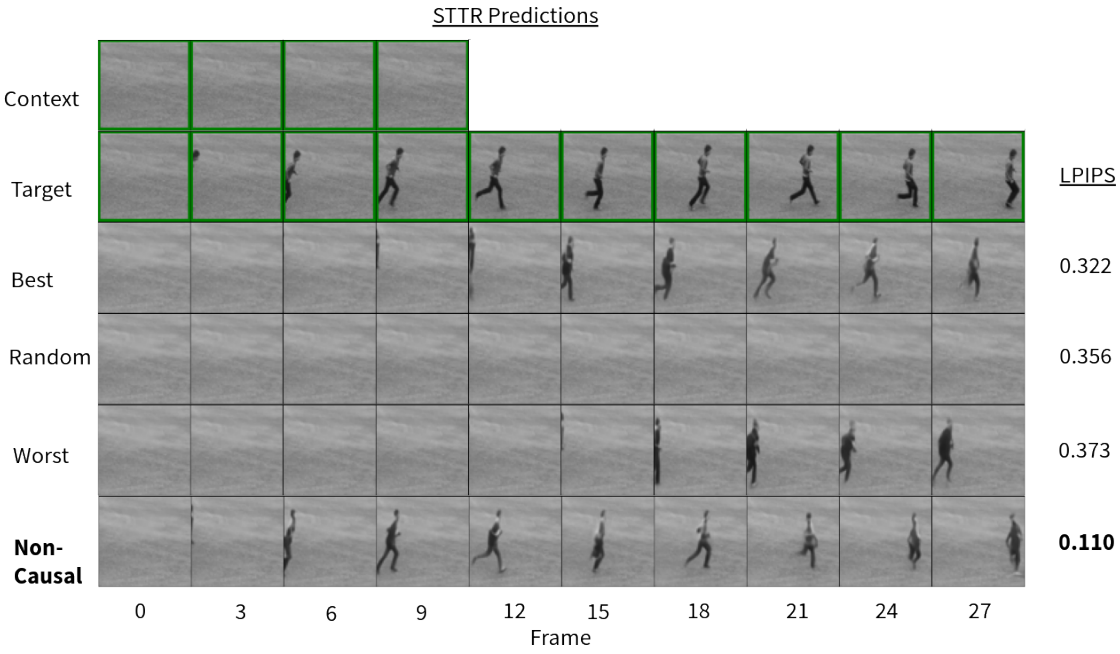


Figure 5.6: Evaluation of the STTR on test sequence #518. We show the different results over selections of the top 100 samples. We also include the non-causal sample to show the expressive power of the latent vector.

performance. There is still a sharp decrease in performance at around Frame 23, when we begin to remove context tokens from the input. We report the better results by attention over the last 20 frames.

Our analysis on KTH shows clear potential for the STTR. Despite only out predicting the SVG in one qualitative metric, we are confident with additional training time and hyper-parameter tuning these results can see further improvements. Our model captures strong measures of uncertainty in its prior distribution, and uses these to make realistic guesses of the future frames.

5.3 Human Pose Prediction

In this section, we briefly present results on H3.6M for the task of human pose prediction. Due to long training times required in hyper-parameter optimization, we only report results using a single metric on a few actions, although our models used all actions during training. Further optimizations and evaluation on more metrics are left as future work.

We start by reintroducing the metric used during training, the MPJPE. Quantitative results from this metric are in Table 5.4. The we present qualitative analysis of our models using two samples taken from the walking action.

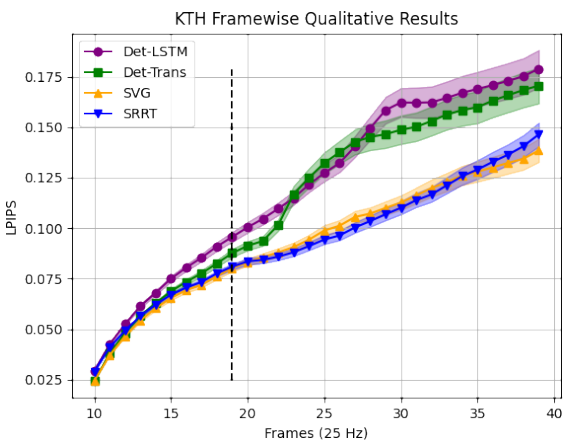


Figure 5.7: Quantitative analysis of all our models with the SVG and STTR showing comparable performance. 95% Confidence interval is shaded.

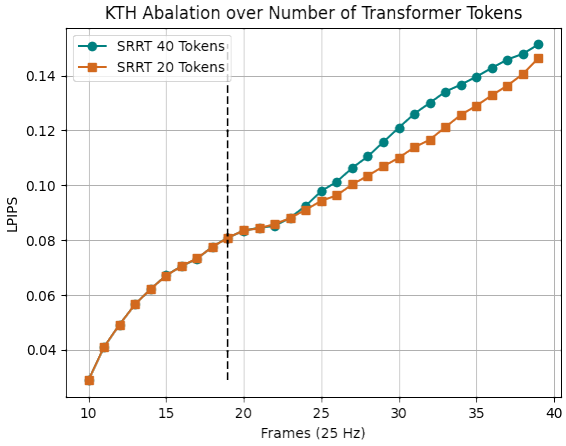


Figure 5.8: Ablation over the number of attention tokens given to the STTR. Performance is stronger by restricting the number of tokens to the training prediction horizon.

This action highlights the long-term generation potential of our stochastic block. Our models were trained using all actions.

MPJPE

The Mean-Per-Joint Position Error calculates the end effector of every joint against the ground truth. This is an average distance, recorded in mm. We calculate the MPJPE by:

$$\mathcal{L}_{MPJPE} = \frac{1}{J \times M} \sum_{j=1}^{J} \sum_{t=1}^{M} \|\hat{x}_t^{(j)} - x_t^{(j)}\|. \tag{5.3}$$

Where $\|\cdot\|$ is the l_2 norm and $y_t^{(j)}$ is the position of joint j at timestep t.

This metric does not always depict realism in motion. This is in part because it weights every joint equally, while in real motion the placement of the left hip is more important than the left arm. Furthermore, it loses validity as we get further away from the context sequence. Natural motion can develop in many different ways from a common seed.

It is for these reasons that we do not heavily weight the results in MPJPE, but instead concentrate on the realism of the generated skeletons. Further discussion is added in the next sub-section.

MPJPE (mm) on H3.6M

Walking						
Frame (time):	2 (80ms)	4 (160ms)	8 (320ms)	10 (400ms)	25 (1000ms)	Mean
ST-Trans	**26.1**	**50.3**	86.4	101.3	**135.2**	**92.7**
ST-STTR	28.0	52.7	**83.0**	**90.6**	139.1	93.2
Eating						
Frame (time):	2 (80ms)	4 (160ms)	8 (320ms)	10 (400ms)	25 (1000ms)	Mean
ST-Trans	**13.7**	**26.7**	**47.5**	**56.5**	**98.7**	59.6
ST-STTR	16.2	30.0	49.9	57.5	102.0	**58.8**
Smoking						
Frame (time):	2 (80ms)	4 (160ms)	8 (320ms)	10 (400ms)	25 (1000ms)	Mean
ST-Trans	**15.3**	**30.4**	**54.9**	65.6	**110.4**	**67.1**
ST-STTR	17.8	32.8	55.5	**64.4**	115.4	71.6
Discussion						
Frame (time):	2 (80ms)	4 (160ms)	8 (320ms)	10 (400ms)	25 (1000ms)	Mean
ST-Trans	**21.7**	**41.2**	**74.4**	88.4	144.0	94.5
ST-STTR	24.8	45.8	75.5	**85.6**	**132.6**	**85.2**

Table 5.4: MPJPE For 4/15 Actions on the H3.6M Dataset

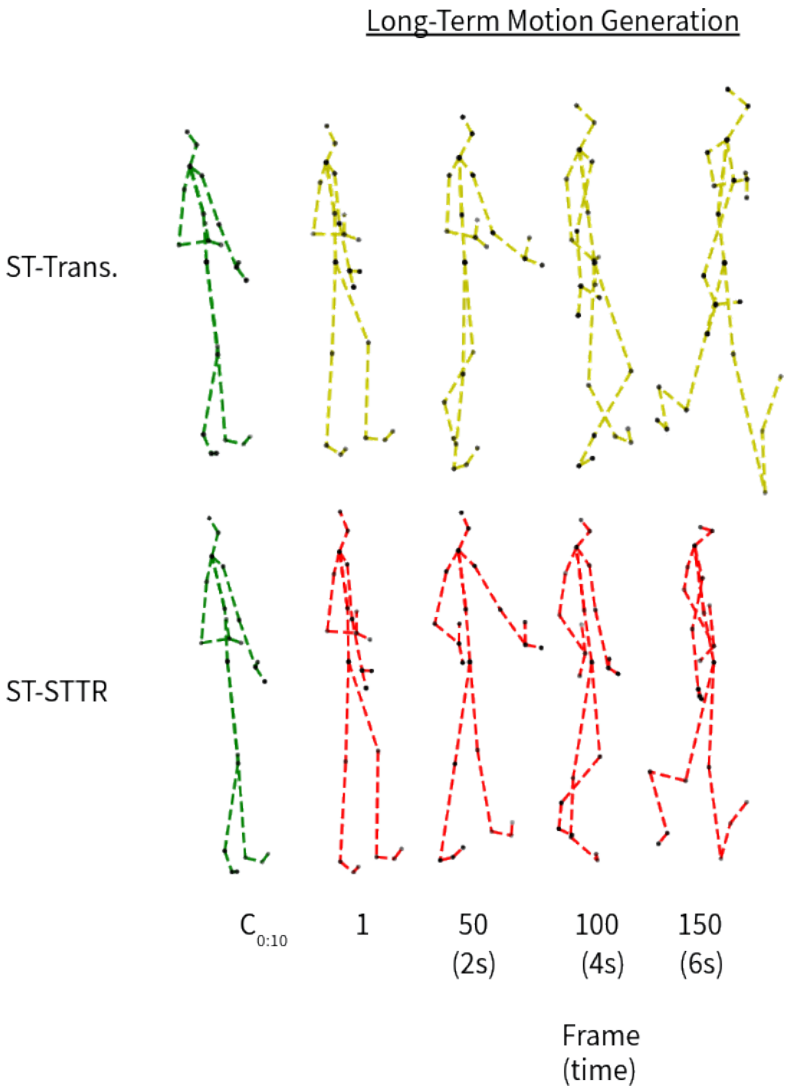


Figure 5.9: We use our ST-STTR and the ST-Transformer to generate predictions 6 seconds into the future. Our ST-STTR continues to show realistic motion until around 4 seconds of generation.

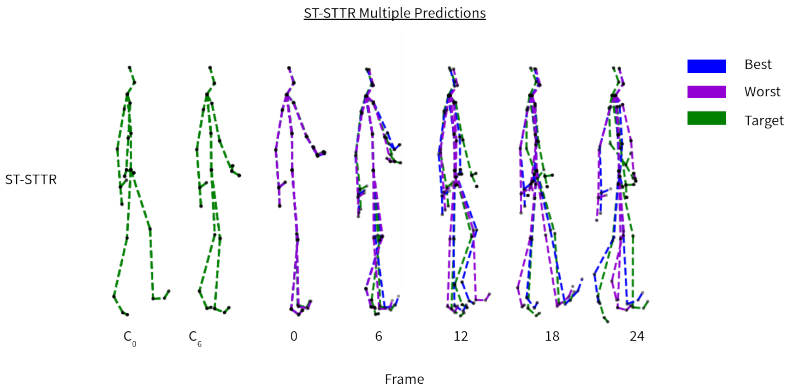


Figure 5.10: We plot the best and worst predictions from our ST-STTR to show variance in the samples. Frame 18 shows that stochasticity is primarily present in the position of the foot. Further deviation occurs in the last frame where our best prediction more closely follows the ground truth.

Results

In Table 5.4, our ST-STTR is outperformed by a deterministic counterpart in almost all situations. This is not surprising, as the ST-Transformer is only optimized over the MPJPE. In our stochastic method, we also minimize the KL Divergence. We see a similar result in the Stochastic Moving MNIST in Table 5.3, where the Deterministic Transformer almost matches performance in PSNR. This signifies a larger problem with l_2 loss functions where the objective is get close to the mean of the data instead of sharp, realistic predictions

Stochastic sampling is a powerful technique which has potential in abstracting small motions into the latent vector. This motion can be used to correct for the drift between the prediction sequence and the ground truth target. Figure 5.10 shows a best and worst prediction, along with the ground truth. Frame 18 of this figure particularly demonstrates the variance learned by our model.

To test the ability to condition and generate based on the models own prediction distribution, we forecast predictions to 150 timesteps as shown in Figure 5.9. This equates to predicting 6 seconds of walking after seeing the motion for 0.4 seconds. At the end of this long generation sequence, both the Spatio-Temporal Transformer and our ST-STTR break down and the skeleton loses form. The results are very encouraging as we succeed in making well-formed prediction far past the frames that the model was originally conditioned on.

6 Conclusion

In this thesis, we proposed and investigated the Stochastic Transformer Module (STTR), a causal self-attention mechanism which learns a distribution of possible futures from a non-causal teacher. Our model is able to maintain strong variation in the prediction, leading to accurate guesses of the sequence continuations despite lack of information in the context. When compared with other state-of-the-art models, ours produces comparable results while having strong predictive powers. We further demonstrate the empirical benefits of a stochastic models with their ability to generate shape, realistic predictions.

Although our model faced difficulty in outperforming metrics, we believe this could be in part due to unconverged training in the case of Human Pose Prediction and overfitting on the KTH Actions dataset. Hyper-parameter tuning and optimizations is a time consuming task to perform on two different modalities. This is especially so in transformers, which increase with a squared complexity to the attention tokens used. A posterior branch that attends over the entire input sequence makes a very powerful predictor, but requires a lot of computation.

Future work could be led in applying additional regularization to our model. Discriminators have been popularly used to increase the fidelity of predictions [39]. We believe that regularization would give a strong boost to our model performance and propose use in future work.

This work could also be scaled up to larger images and videos. A ViT [12] based frame encoder would create both spatially and temporally localized tokens that could be directly used by our ST-STTR. The ViT encoder, however, requires large datasets to overcome its lack of spatial biases [34].

We conclude with a summary of our contributions in this thesis. We proposed a novel model, the STTR, which gives competitive performance against strong baselines, particularly in more realistic videos. Then we run a series of experiments on the STTR to better understand the predictive power of our model. Finally, we proposed an extension of our model to separate temporal and spatial information and test it on the task of Human Pose Prediction.

7 Appendix

This appendix adds two additional figures from our models on SMNIST, and KTH.

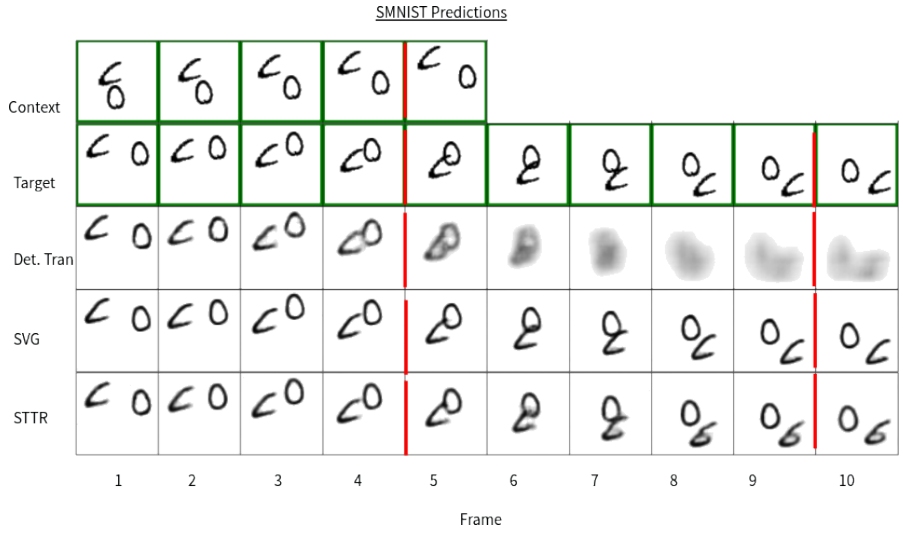


Figure 7.1: This figure shows the failure of determinstic models to make predictions on stochastic datasets. The speed of the digits changes on the red lines. The determinstic transformer is force to predict the mean over possible locations of the digits. Note that because the first four prediction frames are a determinstic continuation of the context, it is only at Frame 5 that the determinstic transformer fails.

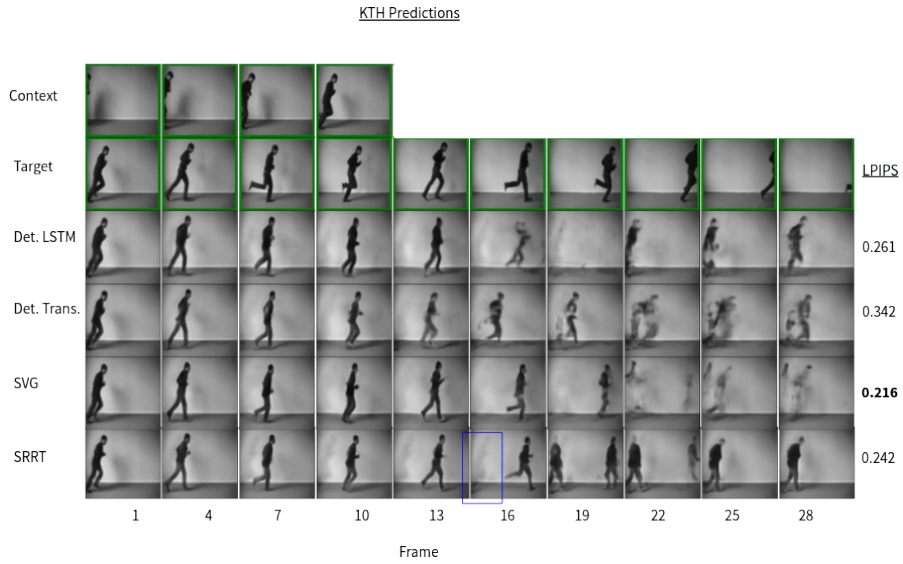


Figure 7.2: This figure shows a prediction on the KTH Dataset. The SVG performs best according to LPIPS, but losses prediction quality in later frames. The STTR learns strong priors from the underlying distribution of the dataset. Due to this, the STTR manifests an additional person to appear at frame 19. We believe this is due to the shadow (blue box) which occurs three frames earlier.

List of Figures

List of Tables

Bibliography

[1] Accessed on 11.07.2023. URL: https://www.ibm.com/topics/gradient-descent.

[2] Accessed on 19.07.2023. URL: https://docs.nvidia.com/deeplearning/performance/dl-performance-fully-connected/index.html.

[3] Accessed on 19.07.2023. URL: https://en.wikipedia.org/wiki/Rectifier_(neural_networks)#/media/File:ReLU_and_GELU.svg.

[4] Emre Aksan et al. "A spatio-temporal transformer for 3d human motion prediction." In: *2021 international conference on 3d vision (3dv)*. IEEE. 2021, pp. 565–574.

[5] Mohammad Babaeizadeh et al. "Stochastic variational video prediction." In: *Arxiv preprint arxiv:1710.11252* (2017).

[6] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: a deep convolutional encoder-decoder architecture for image segmentation." In: *Ieee transactions on pattern analysis and machine intelligence* (2017).

[7] Uttaran Bhattacharya et al. "Take an emotion walk: perceiving emotions from gaits using hierarchical attention pooling and affective mapping." In: *European conference on computer vision*. Springer. 2020, pp. 145–163.

[8] Samuel R Bowman et al. "Generating sentences from a continuous space." In: *Arxiv preprint arxiv:1511.06349* (2015).

[9] George Cybenko. "Approximation by superpositions of a sigmoidal function." In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[10] Emily Denton and Rob Fergus. "Stochastic video generation with a learned prior." In: *International conference on machine learning*. PMLR. 2018, pp. 1174–1183.

[11] Jacob Devlin et al. "Bert: pre-training of deep bidirectional transformers for language understanding." In: *Arxiv preprint arxiv:1810.04805* (2018).

[12] Alexey Dosovitskiy et al. "An image is worth 16x16 words: transformers for image recognition at scale." In: *Arxiv preprint arxiv:2010.11929* (2020).

[13] Vincent Dumoulin and Francesco Visin. "A guide to convolution arithmetic for deep learning (2016)." In: *Arxiv preprint arxiv:1603.07285* (2016).

[14] Hafez Farazi and Sven Behnke. "Frequency domain transformer networks for video prediction." In: *Arxiv preprint arxiv:1903.00271* (2019).

[15] Chelsea Finn and Sergey Levine. “Deep visual foresight for planning robot motion.” In: *2017 ieee international conference on robotics and automation (icra)*. IEEE. 2017, pp. 2786–2793.

[16] Katerina Fragkiadaki et al. “Recurrent network models for human dynamics.” In: *Proceedings of the ieee international conference on computer vision*. 2015.

[17] Jean-Yves Franceschi et al. “Stochastic latent residual video prediction.” In: *International conference on machine learning*. PMLR. 2020, pp. 3233–3246.

[18] Kunihiko Fukushima. “Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.” In: *Biological cybernetics* (1980).

[19] Juergen Gall. Universtität Bonn Video Analystics Class. 2021.

[20] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation.” In: *Proceedings of the ieee conference on computer vision and pattern recognition*. 2014.

[21] Ian Goodfellow. “Nips 2016 tutorial: generative adversarial networks.” In: *Arxiv preprint arxiv:1701.00160* (2016).

[22] Liang-Yan Gui et al. “Adversarial geometry-aware human motion prediction.” In: *Proceedings of the european conference on computer vision (eccv)*. 2018.

[23] Agrim Gupta et al. “Maskvit: masked visual pre-training for video prediction.” In: *Arxiv preprint arxiv:2206.11894* (2022).

[24] Kaiming He et al. “Deep residual learning for image recognition.” In: *Proceedings of the ieee conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[25] Dan Hendrycks and Kevin Gimpel. “Gaussian error linear units (gelus).” In: *Arxiv preprint arxiv:1606.08415* (2016).

[26] Anthony Hu et al. “Probabilistic future prediction for video scene understanding.” In: *Computer vision–eccv 2020: 16th european conference, glasgow, uk, august 23–28, 2020, proceedings, part xvi 16*. Springer. 2020, pp. 767–785.

[27] Quan Huynh-Thu and Mohammed Ghanbari. “The accuracy of psnr in predicting video quality for different video scenes and frame rates.” In: *Telecommunication systems* 49 (2012), pp. 35–48.

[28] Catalin Ionescu et al. “Human3. 6m: large scale datasets and predictive methods for 3d human sensing in natural environments.” In: *Ieee transactions on pattern analysis and machine intelligence* 36.7 (2013), pp. 1325–1339.

[29] Diederik P Kingma and Jimmy Ba. “Adam: a method for stochastic optimization.” In: *Arxiv preprint arxiv:1412.6980* (2014).

[30] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes." In: *Arxiv preprint arxiv:1312.6114* (2013).

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems* 25 (2012).

[32] Solomon Kullback and Richard A Leibler. "On information and sufficiency." In: *The annals of mathematical statistics* (1951).

[33] Yann LeCun. "The mnist database of handwritten digits." In: *Http://yann. lecun. com/exdb/mnist/* (1998).

[34] Seung Hoon Lee, Seunghyun Lee, and Byung Cheol Song. "Vision transformer for small-size datasets." In: *Arxiv preprint arxiv:2112.13492* (2021).

[35] Yijun Li et al. "Flow-grounded spatial-temporal video prediction from still images." In: *Proceedings of the european conference on computer vision (eccv)*. 2018, pp. 600–615.

[36] Zhaojiang Lin et al. "Variational transformers for diverse response generation." In: *Arxiv preprint arxiv:2003.12738* (2020).

[37] William Lotter, Gabriel Kreiman, and David Cox. "Deep predictive coding networks for video prediction and unsupervised learning." In: *Arxiv preprint arxiv:1605.08104* (2016).

[38] Julieta Martinez, Michael J Black, and Javier Romero. "On human motion prediction using recurrent neural networks." In: *Proceedings of the ieee conference on computer vision and pattern recognition*. 2017, pp. 2891–2900.

[39] Michael Mathieu, Camille Couprie, and Yann LeCun. "Deep multi-scale video prediction beyond mean square error." In: *Arxiv preprint arxiv:1511.05440* (2015).

[40] Sergiu Oprea et al. "A review on deep learning techniques for video prediction." In: *Ieee transactions on pattern analysis and machine intelligence* 44.6 (2020), pp. 2806–2826.

[41] Adam Paszke et al. "Pytorch: an imperative style, high-performance deep learning library." In: *Advances in neural information processing systems* 32 (2019).

[42] Viorica Patraucean, Ankur Handa, and Roberto Cipolla. "Spatio-temporal video autoencoder with differentiable memory." In: *Arxiv preprint arxiv:1511.06309* (2015).

[43] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." In: *Arxiv preprint arxiv:1511.06434* (2015).

[44] Hasim Sak, Andrew W Senior, and Françoise Beaufays. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling." In: (2014).

[45] Christian Schuldt, Ivan Laptev, and Barbara Caputo. "Recognizing human actions: a local svm approach." In: *Proceedings of the 17th international conference on pattern recognition, 2004. icpr 2004.* Vol. 3. IEEE. 2004, pp. 32–36.

[46] Younggyo Seo et al. "Reinforcement learning with action-free pre-training from videos." In: *International conference on machine learning*. PMLR. 2022, pp. 19561–19579.

[47] Xingjian Shi et al. "Convolutional lstm network: a machine learning approach for precipitation nowcasting." In: *Advances in neural information processing systems* 28 (2015).

[48] Xingjian Shi et al. "Deep learning for precipitation nowcasting: a benchmark and a new model." In: *Advances in neural information processing systems* 30 (2017).

[49] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." In: *Arxiv preprint arxiv:1409.1556* (2014).

[50] Theodoros Sofianos et al. "Space-time-separable graph convolutional network for pose forecasting." In: *Proceedings of the ieee/cvf international conference on computer vision*. 2021, pp. 11209–11218.

[51] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. "Unsupervised learning of video representations using lstms." In: *International conference on machine learning*. PMLR. 2015, pp. 843–852.

[52] Ashish Vaswani et al. "Attention is all you need." In: *Advances in neural information processing systems* 30 (2017).

[53] Angel Villar-Corrales et al. "Mspred: video prediction at multiple spatio-temporal scales with hierarchical recurrent networks." In: *Arxiv preprint arxiv:2203.09303* (2022).

[54] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. "Anticipating visual representations from unlabeled video." In: *Proceedings of the ieee conference on computer vision and pattern recognition*. 2016, pp. 98–106.

[55] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity." In: *Ieee transactions on image processing* 13.4 (2004), pp. 600–612.

[56] Tianfan Xue et al. "Visual dynamics: probabilistic future frame synthesis via cross convolutional networks." In: *Advances in neural information processing systems* 29 (2016).

[57] Richard Zhang et al. “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.